

CHALMERS

Utvärdering av alternativ till fysiska PLC-system

Examensarbete inom högskoleingenjörsprogrammet Mekatronik

ERIK JOSEF BELLMAN
JOHANNA KÄLLSTRÖM

INSTITUTIONEN FÖR ELEKTROTEKNIK

CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2023
www.chalmers.se

INSTITUTIONEN FÖR ELEKTROTEKNIK,
Göteborg, Sverige 2023

EXAMENSARBETE

Utvärdering av alternativ till fysiska PLC-system

ERIK JOSEF BELLMAN
JOHANNA KÄLLSTRÖM



CHALMERS
UNIVERSITY OF TECHNOLOGY

Institutionen för Elektroteknik
Chalmers tekniska högskola
Göteborg, Sverige 2023

Utvärdering av alternativ till fysiska PLC-system
Examensarbete inom högskoleingenjörsprogrammet Mekatronik
Erik Josef Bellman och Johanna Källström

© Erik Josef Bellman
© Johanna Källström

Handledare: Anton Andersson, Acobia
Examinator: Veronica Olesen, System- och Reglerteknik

Institutionen för elektroteknik
Chalmers Tekniska Högskola
SE-412 96 Göteborg, Sverige
Telefon: +46 (0)31-772 10 00

Förord

Detta examensarbete har genomförts av två mekatronikstudenter vid institutionen för Elektroteknik på Chalmers tekniska högskola på uppdrag av automationsföretaget Acobia. Arbetet omfattar 15 högskolepoäng och genomfördes under perioden januari till juni 2023.

Idén bakom examensarbetet är framtagen tillsammans med vår handledare Anton Andersson på Acobia. Vi riktar därför ett stort tack till Anton och de kollegor som hjälpt oss på vägen med allt från problemlösning till lån av utrustning.

Examensarbetet har varit en lärorik och utmanande resa som har gett oss möjlighet att tillämpa teoretiska kunskaper i praktiska situationer. Vi har fått en stor inblick i hur det är att arbeta på ett företag, och hur detta skiljer sig från miljön på Chalmers tekniska högskola, men även på vilka sätt det är likt.

Vi vill också tacka Chalmers Tekniska Högskola för utbildningen och för möjligheten att genomföra examensarbetet hos Acobia.

Vi hoppas att detta arbete kommer att bidra till Acobias fortsatta utveckling och önskar företaget all framgång framöver.

Sammanfattning

PLC-system används idag nästan exklusivt inom industrin för att styra processer som kräver hög tillförlitlighet till bibehållen drift. Med hög efterfrågan på PLC-enheterna som krävs för att exekvera programmen i kombination med materialbrist blir väntetiderna längre och fakturorna dyrare. Detta leder till frågan; Finns det alternativ som är mer tillgängliga?

Denna fråga utvärderades med en rad tester i programmet OpenPLC och teorievaluering i de fall där tester inte var möjliga. Ett testsystem byggdes upp med lampor och knappar kopplade till en PLC som konfigurerats om för att agera nätverks-I/O. I/O:n kopplades med nätverkskabel till en serverdator på vilken OpenPLC-programmet exekverades.

Testerna i kombination med teorievaluering ledde till tillräckliga resultat för att överväga att i vissa specifika fall byta ut PLC mot alternativa lösningar. System som inte alltid kräver 100% driftsäkerhet, och system som inte skulle komma att utsättas för cyberattacker kan vara lämpliga för denna typ av substitut. Detta betyder att mycket pengar och tid kan sparas in i de fall företag har serverdatorer men inte PLC och skulle behöva implementera styrsystem som kräver medelhög driftsäkerhet. I vissa testfall förekom resultat som speglade mycket hög driftsäkerhet, men eftersom faktorer som belastning på CPU:n påverkar detta finns det ingen garanti för att det alltid gäller ens i dessa fall.

Arbetet har utförts på Acobia AB:s kontor på Lindholmen.

Abstract

PLC systems are almost exclusively used in today's industry to control processes that require high levels of stability. However, there is a large shortage of PLC-units, leading to long delivery times and expenses going up. This leads to the question; Are there other options?

This question was evaluated by testing in the program OpenPLC, and by looking at theory in the cases where tests were not possible. A testing system was developed, consisting of lamps and switches connected to a PLC that was reconfigured to act as an I/O-unit. The I/O was then connected to a server computer on which the OpenPLC-program was executed.

This led to results that were convincing enough to in specific cases consider alternatives to the PLC unit. Systems that don't always need 100% reliability in performance, and systems that are not likely subjects to cyber-attacks seem appropriate for this substitution. This means that a lot of money and time can be saved, especially if the company in question already has server-computers but not PLC's and needs a control system with medium-high reliability.

There is good potential for the alternative system to be applicable in even more cases, since some test results indicate very high reliability being possible (within reasonable circumstances), but there are still some problems with load that need to be dealt with to ensure such performance.
This project has taken place in Acobia AB's office space at Lindholmen, Gothenburg.

Innehållsförteckning

Terminologi/Förkortningar 1
1. INLEDNING 2
1.1 Bakgrund 2
1.2 Syfte 3
1.3 Avgränsningar 3
1.4 Precisering av frågeställning 3
2. TEORETISK/TEKNISK BAKGRUND 3
2.1 PLC och Soft-PLC 3
2.2 Nätverksbaserad kommunikationsmodul 4
2.3 EcoStruxure Automation Expert 5
2.4 OpenPLC 6
2.5 Redundans 6
2.6 Säkerhetskritiska system 7
3. METOD 8
3.1 Tidsfördröjning/svarstid 8
3.2 Funktionskontroll 9
3.3 Belastningstest 9
3.4 Avbrott i kommunikationen 10
3.5 Nätverksfördröjning 10
4. KONSTRUKTION 10
4.1 Konstruktion av systemet 10
4.2 Serverdator 11
4.3 Nätverkskommunikation 11
4.4 I/O-modul och kommunikationsprotokoll 12
5. GENOMFÖRANDE 12
5.1 Installation OpenPLC 12
5.2 Konfigurering av Modicon M241 13
5.3 Styrning via OpenPLC 15
5.4 Testprogram lampstyrning fördröjning med brytare 15
5.5 Testprogram prestanda över tid 16
6. RESULTAT 17
6.1 Tidsfördröjning/svarstid 17
6.2 Funktionskontroll 18
6.3 Belastningstest 19
6.4 Avbrott i kommunikationen 20

6.5 Nätverksfördröjning 21
7. DISKUSSION OCH SLUTSATS 22
7.1 Testresultat 22
7.2 Redundans 22
7.3 Säkerhetssystem 22
7.4 Cybersäkerhet och nät-trafik 23
7.5 Fördelar och nackdelar med OpenPLC 23
7.6 Implementeringsmöjligheter 24
7.7 Teknikens roll i samhället 25
7.8 Fortsatt arbete 25
Referenser 26

Terminologi/Förkortningar

CPU:	“Central processing unit”, är processorn/hjärnan i en dator eller annan beräknande enhet.
Cykeltid:	Tid som går mellan varje exekvering av programmet i ett PLC-system
EAE:	“Ecostruxure Automation Expert”, Schneider Electrics eget automationsprogram.
FBD:	“Function block diagram”, en vanlig metod för att programmera PLC-system enligt standarden IEC 61131–3.
I/O:	“Input/Output”, inport eller utport för signaler från eller till enheter som är uppkopplade i systemet.
Packet loss:	Om data som skickas till en enhet inte kommer fram som det ska, eller om inget svar ges från enheten till avsändaren, kallas detta packet loss.
PLC:	“Programmable logic controller”, refererar antingen till systemet i sin helhet eller till datorn som exekverar programmet, beroende på om PLC-system eller PLC används.
TCP/IP:	“Transmission control protocol/Internet protocol”, ett kommunikationsprotokoll som tillåter att skicka och motta meddelanden mellan enheter via internet.
UDP:	“User datagram protocol”, ett alternativ till TCP/IP som endast skickar meddelande utan att ta emot ett svar från andra enheter.

1. INLEDNING

Inom området för industriell automatisering och styrning spelar PLC (Programmable Logic Controller) en avgörande roll. PLC är en robust industridator som används för att automatisera och övervaka olika processer och maskiner som kräver hög tillförlitlighet [1]. En PLC utför en sekvens av instruktioner som är programmerade att förändras för att ta hänsyn och ge svar på olika insignaler till systemet. PLC har tidigare skilt sig ganska mycket från en PC, men har under de senaste åren alltmer börjat likna stationära datorer. Funktionaliteten hos PLC har utvecklats till att omfatta sekventiell relästyrning, rörelsestyrning, processtyrning, distribuerade styrsystem och nätverk. De börjar också likna varandra i datahantering, lagring, processorkraft och kommunikationsförmåga. PLC-liknande programmering kombinerat med en nätverksstyrd I/O-hårdvara, gör att en stationär dator för allmänna ändamål i vissa fall kunna ersätta en PLC. Att använda stationära datorer som styrenhet har i allmänhet inte accepterats inom industrin eftersom stationära datorer körs på mindre stabila operativsystem än PLC.

PLC:n exekverar endast en enda uppsättning eller följd av uppgifter, med överlägsen prestanda och tillförlitlighet. PLC genererar inte oförutsägbara händelser, såsom en plötslig omstart för att installera en uppdatering, eller att den börjar utföra andra kommandon än planerat. En PC är designad för att utföra ett varierande antal funktioner samtidigt med olika prioriteter. PLC har också vissa funktioner som inte återfinns i vanliga datorer, så som tolerans mot temperatur, fuktighet och vibrationer, och livslängden för en PLC-processor skiljer sig från den av en PC [2]. I projektet undersöks möjligheten att byta ut fysiska PLC-enheter mot alternativ som kan vara mer flexibla, exempelvis ett program som ligger helt på en serverdator.

1.1 Bakgrund

PLC utgör en effektiv lösning för att styra system som kräver hög grad av stabilitet och pålitlighet. I händelse av ett omfattande system med flera samtidiga processer kan dock behovet av flera PLC-enheter innebära betydande kostnader [3]. Av den anledningen är det fördelaktigt att utvärdera alternativa lösningar ur både ett kostnads- och hållbarhetsperspektiv. Detta blir särskilt relevant om det finns en befintlig serverdator på plats som möjligtvis kan fungera som styrenhet och därmed eliminera behovet av att investera i nya PLC-enheter.

Hårdvaran för PLC-system är ofta låst till mjukvaran och företag som köpt in en programvara blir därför begränsad till att använda hårdvara av samma fabrikat och vice versa. I kristider eller i andra omständigheter är det inte en självklarhet att hårdvaran för PLC-system finns tillgänglig och framför allt blir tillgängligheten begränsad när man måste förhålla sig till specifika fabrikat. Genom att använda standardiserad dataöverföring via nätverksprotokoll och enhetsanslutningar direkt från PC till en I/O-modul görs systemet mer flexibelt och obundet till någon specifik tillverkare. Användaren blir då oberoende av tillverkarspecifika standarder. Systemet kan enklare utökas eller modifieras [4].

Företag som redan påbörjat utveckling av mjukvaruprogram som inte är strikt kopplade till specifik hårdvara är Schneider Electric som en del av UniversalAutomation.org och OpenPLCProject. Schneider Electrics programvara EAE utlovar ett system som kan kopplas direkt till distribuerade I/O-moduler och är kompatibel med flerta typer av hårdvara. OpenPLCProject erbjuder en gratis programvara, OpenPLC, som också ska kunna hantera direkt styrning av olika I/O-moduler via standardiserad nätverkskommunikation.

1.2 Syfte

Syftet med projektet är att utvärdera alternativ till fysisk PLC för att se om denna nya lösning kan utföra ett motsvarande jobb. Fördelar och nackdelar med lösningen ska dokumenteras. Vidare ska det undersökas för vilka system dessa lösningar med fördel kan tillämpas.

Målet är att hitta konkreta fall där det kan vara mer eller mindre lämpligt med ett alternativ till den annars mycket tåliga PLC, samt att jämföra säkerheten.

1.3 Avgränsningar

I detta projekt kommer endast OpenPLC utvärderas som alternativ till fysisk PLC. EAE kommer kort nämnas då denna programvara övervägdes, men då denna programvara skulle innebära inlärning av ett helt nytt programmeringsspråk, till skillnad från OpenPLC som utgår från samma standard som traditionell PLC-programmering hamnar EAE utanför omfattningen av arbetet. Andra avgränsningar som görs är att frågor som redundans, systemintegration och cybersäkerhet endast kommer diskuteras teoretiskt och inte kommer omfattas av systemtester.

1.4 Precisering av frågeställning

De frågor som kommer hanteras är:

1. Finns det fall/situationer där PLC-system skulle kunna ersättas av alternativa lösningar?
2. Vilka processer skulle vara lämpliga?
3. Vilken/vilka alternativa lösningar skulle kunna användas på systemet?
4. Vilka för- och nackdelar finns med den alternativa lösningen?
5. Skulle en alternativ lösning kunna implementeras på ett system som kräver redundanta CPU:er?

2. TEORETISK/TEKNISK BAKGRUND

Teoriavsnittet innefattar teoretisk bakgrund som behövs för att förstå hur ett traditionellt PLC system skulle kunna ersättas med ett nätverksbaserat system som styrs direkt via en serverdator.

2.1 PLC och Soft-PLC

En traditionell PLC är en fysisk enhet som vanligtvis består av en robust industriell dator med inbyggd hårdvara som in- och utgångsmoduler, processorer och minne. Den är konstruerad för

att tåla de tuffa förhållandena i en industriell miljö, inklusive vibrationer, elektromagnetiskt brus och temperaturvariationer. En standard PLC har anslutningar till en I/O för att styra sensorer, ställdon och andra enheter i systemet. Programvaran laddas ner och körs på PLCn och är därför innesluten i hårdvaran. Mjukvaran skrivs i något av programmeringsspråken enligt standarden 61131–3 [2].

En soft-PLC är en programvarubaserad PLC som körs på en vanlig dator eller industriell styrenhet. I stället för att använda en speciell PLC-hårdvara för att styra in- och utgångar kan en soft-PLC använda standardiserade datorgränssnitt som Ethernet, USB eller seriell kommunikation för att styra sensorer och ställdon via en enkel I/O-modul. Mjukvaran för en soft-PLC är installerad på den valda plattformen och kan vara ett fristående program eller en del av en större automatiseringsprogramvara.

2.2 Nätverksbaserad kommunikationsmodul

För att system ska kunna kommunicera med varandra över ett nätverk används så kallade kommunikationsprotokoll. Ett kommunikationsprotokoll är en uppsättning regler och standarder som styr hur kommunikationen sker. Protokollet definierar reglerna för hur data ska struktureras, överföras, verifieras och tolkas av sändare och mottagare för att säkerställa en pålitlig och korrekt kommunikation.
Ett kommunikationsprotokoll specificerar olika aspekter av kommunikationen, inklusive dataformatet, överföringshastigheten, felhanteringen, adresseringen, autentiseringen och säkerheten. Det fastställer också vilken enhet eller komponent som har rätt att initiera eller svara på kommunikationsbegäran.

För att kunna styra en I/O enhet direkt från en serverdator via ett nätverksprotokoll krävs en specifik hårdvara som stöder valt kommunikationsprotokoll. Denna hårdvara kallas för en nätverks-baserad kommunikationsmodul. En nätverk-I/O-modul fungerar som en mellanhand mellan programvaran och I/O-enheter, och möjliggör direkt kommunikation mellan dem via ett nätverksprotokoll. Det finns ett stort utbud av nätverksmoduler tillgängliga på marknaden och valet av modul beror på behoven i det specifika systemet.

Att styra ett system direkt via en server kan innebära begränsningar i antalet och typerna av I/O-enheter som används eftersom I/O-enheterna behöver ha protokollkompatibilitet, samma drivrutiner och kommunikationsgränssnitt. Om inte alla I/O-enheter stöder de nätverksprotokoll som används blir det svårt att konfigurera. Även serverkapacitet kan hämma antalet I/O-enheter eftersom en server har begränsningar när det gäller processorkraft, minne och nätverkskapacitet. Om antalet enheter och datamängden blir för stort kan servern överbelastas.

2.2.1 Kommunikationsprotokoll Modbus

Modbus är ett seriellt kommunikationsprotokoll som används för att möjliggöra kommunikation mellan industriella enheter. Modbus-protokollet bygger på en master-slave-arkitektur, där en master-enhet initierar kommunikationen och begär data från eller skickar

instruktioner till en eller flera slave-enheter. Protokollet stöder olika typer av funktioner för att utföra olika uppgifter, till exempel att läsa från eller skriva till register, kontrollera digitala in- och utgångar eller utföra kommandon. Modbus har blivit en populär standard inom industriell automation på grund av sin enkelhet, tillförlitlighet och stora användarbas. Protokollet är kompatibelt med många olika enheter och kan integreras i olika system och plattformar.

Modbus TCP/IP är en variant av Modbus-protokollet som används för att möjliggöra kommunikation mellan enheter över ett Ethernet-nätverk. Protokollet är baserat på TCP/IP-protokollen som förklaras mer detaljerat i stycket nedan.

2.2.2 Modbus TCP/IP

TCP/IP protokollet består av två huvudprotokoll: Transmission Control Protocol (TCP) och Internet Protocol (IP). TCP använder sig av en funktion som kallas för "acknowledgement" (ACK), vilket innebär att mottagaren av data skickar tillbaka en bekräftelse på att data har mottagits. Dessa bekräftelser används för att säkerställa att all data har mottagits korrekt och för att upprätthålla ordning i dataöverföringen. Genom att använda TCP/IP-protokollet kan Modbus garantera att all data överförs korrekt och att enheterna kommunicerar effektivt med varandra i nätverket [5].

Den data som hanteras i Modbusprotokollet är organiserad i ett eller flera register. För att skicka data i Modbusprotokollet delas den upp i paket. Ett paket innehåller information om vilken enhet som ska ta emot data, vilken typ av funktion som ska utföras, samt vilket registerdata ska skrivas till eller läsas från [6].

Paketen skickas sedan som segment över nätverket genom TCP/IP-protokollet. Varje segment innehåller en sekvens av paketet och inkluderar ett sekvensnummer för att hålla reda på vilken ordning segmenten ska sättas samman i på mottagarsidan [6].

Master-enheten initierar kommunikationen genom att skicka en förfrågan till en specifik slave-enhet med information om vilket register som ska läsas eller skrivas till. Slave-enheten svarar sedan på förfrågan med det efterfrågade värdet eller en bekräftelse på att värdet har skrivits till registret [6].

2.3 EcoStruxure Automation Expert

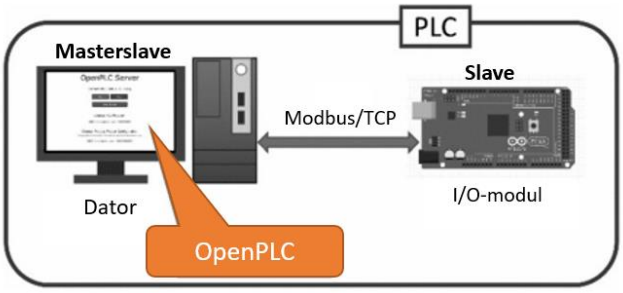
EcoStruxure Automation Expert (EAE) är Schneider Electric:s egna mjukvaruprogram som erbjuder ett alternativ till traditionella PLC-system med decentraliserade I/O-moduler och hårdvara som inte är låst till specifik mjukvara. EAE har en runtime-komponent som när den körs möjliggör automatisering utan behovet av fysiska PLC.

För att använda EcoStruxure Automation Expert som en soft-PLC, är det nödvändigt att skapa en projekt- och programvarustruktur i plattformen. Det är möjligt att använda en integrerad utvecklingsmiljö (IDE) för att utveckla den önskade automationslogiken. EAE har ett eget

språk där funktionsblock används likt FBD. EAEs språk utgår dock från standarden IEC 61499 till skillnad från FBD som utgår från 61131-3.

2.4 OpenPLC

OpenPLC är en programvara med öppen källkod som utvecklats av Thiago Rodrigues Alves. Programvaran gör det möjligt att simulera PLC-funktioner med hjälp av en dator och en I/O-enhet (se figur 2.1). OpenPLC fungerar på både Microsoft Windows och Linux-operativsystem [5].



Figur. 2.1. OpenPLC system. Modifierad från [7]. *Figuren visar ett PLC-system där en dator exekverar programkod via OPenPLC-runtime för att styra en nätverks-I/O via Modbus TCP. Här agerar datorn som en Modbus-master och I/O-modulen som en Modbus-slave.*

OpenPLC kan användas som en soft-PLC där datorn direkt styr I/O-modulen med Modbus/TCP-protokoll vilket visas i figuren ovan. I/O-enheten styr sedan hårdvara enligt datorns instruktioner.

Utvecklingsmiljön, OpenPLC Editor, är baserad på 61131-3 standarden dvs Ladder Diagram, Structured Text, Function Block Diagram, Instruction List samt Sequential Function Chart. Dessa fem språk ger olika tillvägagångssätt för att programmera industriella styrsystem baserat på specifika behov och preferenser. När programmet exporteras till OpenPLC runtime översätts valt programspråk till strukturerad text [5].

2.5 Redundans

Redundans är ett uttryck som ofta används i uppbyggnad av PLC-system, och det betyder “överflödig utrustning”, vilket innebär utrustning som ej används i normal drift. Ett exempel på detta är användning av en extra givare som är kopplad till samma mätpunkt som en annan och som bara blir aktiv om den primära givaren skulle sluta fungera. En annan användning av redundans är redundanta master-enheter, vilket skulle innebära att om den primära mastern skulle sluta fungera eller bli för belastad för att kunna utföra sina instruktioner, kommer den redundanta mastern ta över programexekveringen, och således bli den primära mastern. Den tidigare primära mastern kommer då bli redundant master och kommer vänta på att behöva ta över exekveringen på samma sätt. I/O-moduler kan i de flesta fall konfigureras för den här typen av styrstruktur[8]

2.6 Säkerhetskritiska system

Ett säkerhetskritiskt system är ett system vars felaktiga funktion kan orsaka allvarliga skador på människor, miljö eller egendom. Det kan också definieras som ett system vars korrekta funktion är kritisk för att upprätthålla säkerheten. Detta kan vara till exempel i industriella processer, transport, medicinsk utrustning, kraftverk eller andra tillämpningar där en felaktig funktion kan leda till allvarliga konsekvenser. Ett säkerhetskritiskt system måste ha hög tillförlitlighet, precision och robusthet för att uppfylla de krav som ställs på säkerhet [9].

2.6.1 Säkerhetssystem

Funktionell säkerhet handlar om att utforma och implementera skyddsåtgärder för att reducera riskerna för oönskade händelser. För att uppnå funktionell säkerhet används säkerhetssystem (SS). I SS implementeras nödvändiga säkerhetsfunktioner som kan upptäcka farliga förhållanden och leda till en säker status genom att garantera att en önskad åtgärd, till exempel ett säkert stopp, utförs. Funktionell säkerhet är också viktigt för att säkerställa att processen eller maskinen fungerar korrekt genom att hantera möjliga operatörsfel, hårdvarufel och förändringar i miljön [10]

SS ingriper om det grundläggande styrsystemet och operatörens åtgärder resulterat i en farlig situation för processen eller maskinen. Detta kan antingen vara en direkt fara eller potential till fara. SS fungerar självständigt parallellt med styrsystemet och består av en eller flera säkerhetsloopar vilka övervakar och hanterar faror. I de fall fara upptäcks leder SS styrsystemet till en fördefinierad säker status baserat på bedömd risk. Säkerhetssystemet använder specialdesignad instrumentering, fristående reläer, modulära reläer eller säkerhets-PLC för logisk bearbetning [10,11].

Kraven på ett säkerhetssystem kan variera beroende på applikationen och tillämpningen men generellt ska ett säkerhetssystem uppfylla följande krav:

1. Tillförlitlighet: Systemet måste vara tillförlitligt och fungera korrekt under alla förhållanden och i alla miljöer. Detta innebär att systemet ska ha hög driftsäkerhet och inte misslyckas vid en felaktig funktion.
2. Redundans: Systemet måste ha inbyggd redundans för att undvika enstaka fel. Detta innebär att systemet ska ha dubblerade komponenter och säkerhetskopior som säkerställer att systemet kan fortsätta fungera vid ett fel.
3. Felhantering: Systemet måste kunna upptäcka och hantera fel och felaktig funktion. Detta innebär att systemet ska ha inbyggda diagnostiska funktioner och avbrottsrutiner som kan hantera problem och minimera risken för skador eller skada på utrustning.
4. Säkerhetsnivåer: Systemet måste uppfylla relevanta säkerhetsstandarder och föreskrifter som ställer krav på säkerhetsnivåer och bedömning av säkerhet.
5. Övervakning och rapportering: Systemet måste ha funktioner för övervakning och rapportering av säkerhetshändelser. Detta innebär att systemet ska kunna övervaka och logga säkerhetsrelaterade händelser och varningar, samt att kunna rapportera dessa händelser till relevanta personer eller system.

6. Användarvänlighet: Systemet måste vara lätt att använda och tillgängligt för användare med olika nivåer av expertis och utbildning. Detta innebär att systemet ska ha ett användarvänligt gränssnitt och klart utformad dokumentation och instruktioner.

2.6.2 Säkerhets-PLC

En säkerhets-PLC är en PLC som är utformad för att hantera säkerhetskritiska applikationer där tillförlitlighet och säkerhet är avgörande. En säkerhets-PLC stöder alla applikationer som en standard-PLC men har också integrerade säkerhetsfunktioner för att övervaka och hantera säkerhetssignaler.

Säkerhets-PLC:er är byggda med högsta tillförlitlighet och tillgänglighet. Själva hårdvaran i systemet är konstruerad för att klara av tuffa och ogynnsamma miljöförhållanden. Det kan handla om höga eller låga temperaturer, hög luftfuktighet, damm eller smuts, vibrationer eller andra liknande påfrestningar. För att säkerställa hög tillförlitlighet och tillgänglighet vid användning av säkerhets-PLC, används redundans och omfattande diagnostik för att upptäcka fel innan farliga situationer uppstår. Interna testsignaler skickas mellan programcyklerna för att kontrollera att funktionerna är fungerande och utför sitt arbete korrekt. Säkerhets-PLC:s programvara är speciellt utvecklad för att ha en mängd felupptäckande och övervakningsåtgärder, vilket garanterar att programmodulerna alltid fungerar korrekt [11]

Standard-PLC skiljer sig från Säkerhets-PLC i att de saknar specifika säkerhetsapplikationer. Enligt Sharma KLS [11] är standard-PLC begränsade i flera områden, inklusive:

- Felsäkra egenskaper
- Tillförlitlighet (oupptäckta och farliga fel) på grund av brist på omfattande diagnostik
- Stabilitet med mjukvaruversioner
- Flexibilitet och säkerhet
- Skydd i kommunikation
- Redundans

3. METOD

Följande kapitel täcker de olika tester som utförts för att verifiera funktionen och tillförlitligheten hos servern och dess kommunikation med I/O-enheten. Utförandet av varje test beskrivs i detalj, inklusive de steg som krävdes för att genomföra testet och de mätningar som utfördes.

3.1 Tidsfördröjning/svarstid

Syftet med att testa tidsfördröjning och svarstid är att analysera fördröjning av kommunikationssvar från serverdatorn till I/O-enheten via nätverkskabeln. Två lampor används till testet. I testet jämförs responstiden vid tändning av en lampa med en brytare via programmet mot en lampa som är direkt seriekopplad till brytaren med sladd.

3.1.1 Utförande tidsfördröjning/svarstid

För att genomföra testet skrivs ett program som styr en lampa genom att använda en switch. Programmet gör att lampan tänds eller släcks baserat på värdet på switchen. För att göra detta kopplas switchen och den ena lampan till I/O-enheten.

Den andra lampan seriekopplas med switchen. Detta innebär att den andra lampan kommer att tändas direkt när switchen vrids på.
Efter att kopplingen har gjorts, körs programmet och switchen vrids på och av för att testa funktionaliteten. Med mätutrustning beräknas skillnaden i tiden det tar för lampan att tändas via programmet jämfört med tiden det tar för den andra lampan att tändas via strömkabeln.

För att säkerställa tillförlitligheten i resultaten upprepas testet flera gånger och resultatet analyseras för att säkerställa att tidsfördröjning inte blir ett för stort problem för att den alternativa lösningen ska vara genomförbar.

3.2 Funktionskontroll

Syftet med att testa funktionen är för att kontrollera tillförlitlighet. I testet mäts antalet in- och utsignaler som skickas från serverdatorn till I/O-enheten via nätverkskabeln.

I testet skickas en puls som tänder en lampa varje 0,5 sekunder. Programmet har två räknare, en som räknar varje gång signal har skickats till I/O för att tända lampan, och en som räknar varje gång lampan fysiskt tänts. Testet gör det möjligt att få en uppfattning om funktionssäkerheten i systemet och dess förmåga att tillförlitligt skicka signaler till I/O-enheten.

3.2.1 Utförande funktionskontroll

En lampa ansluts till en ut- och en ingång på I/O-modulen. Ett program skrivs som skickar hög signal till utgången varje 0,5 sekunder, och två räknare som räknar upp enligt tidigare beskrivning. Programmet laddas upp till OpenPLC runtime och startas. Testet pågår under en längre tid för att läsa av data och få resultat som är lättare att analysera. Testet upprepas med flera olika cykeltider och olika tider mellan klockpulser testas för att erhålla ytterligare data.

Skillnader mellan de två räknarna kan sedan användas för att analysera hur olika cykeltider påverkar funktionen av programmet såväl som generell funktionsduglighet.

3.3 Belastningstest

Belastningstestet går ut på att mäta antalet ut- och insignaler som kommer fram genom att skicka en signal till en lampa som lyser i 0,5 sekunder och sedan är släckt i 0,5 sekunder. Se funktionskontroll ovan. Testet utförs på samma sätt som i funktionskontrollen, men med en

högt belastad CPU. Syftet är att mäta hur väl serverdatorn klarar av att hantera en hög belastning utan att det påverkar funktionaliteten i systemet och därmed mätresultaten.

3.4 Avbrott i kommunikationen

Testet syftar till att verifiera systemets förmåga att vid avbrott i kommunikationen mellan serverdatorn och I/O på ett tillförlitligt sätt återhämta sig utan att förlora data eller funktionalitet. Testet utförs genom att dra ut nätverkskabeln för att skapa ett avbrott och sedan observera systemets respons och prestanda.

Systemet förbereds med ett program som hanterar både in- och utgångssignaler cykliskt. Nätverkskabeln dras ut och ett avbrott i kommunikationen mellan serverdatorn och I/O uppstår. Vad som händer med systemet när avbrott inträffar analyseras och variationslösa tendenser noteras. Nätverkskabeln återansluts och systemets beteende analyseras även nu. Den viktigaste frågan är om systemet återhämtar sig från avbrottet och om det återgår till sitt normala tillstånd, utan att förlora data eller funktionalitet. Testet upprepas flera gånger för att se hur väl systemet hanterar avbrott och kan återhämta sig utan att förlora data eller funktionalitet.

3.5 Nätverksfördröjning

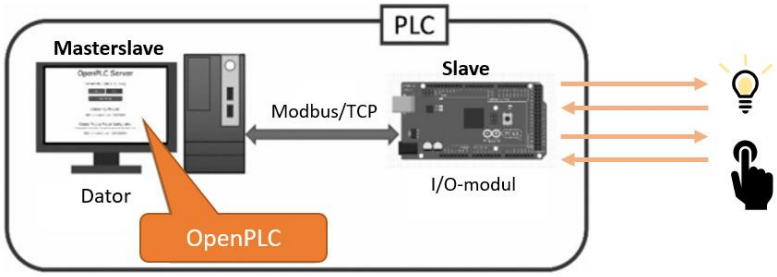
För att testa nätverksfördröjningen utförs ett ping-test, där I/O-modulens IP pingas med hjälp av kommandofönstret på serverdatorn, och packet loss och responstid läses av och analyseras.

4. KONSTRUKTION

I följande kapitel beskrivs systemkonstruktionen för det OpenPLC-baserade testsystemet. Systemet är utformat för att testa hårdvarufunktioner i en labbmiljö, vilket gör att antalet ingångar och utgångar i I/O-modulen inte är av största vikt. Valet av I/O-modul är därför gjort i samråd med projektbeställaren.

4.1 Konstruktion av systemet

Figur 4.1 visar testsystemets konstruktion. En serverdator med OpenPLC sammankopplas med en modicon M241 I/O-modul via en nätverkskabel. Modbus TCP/IP-protokollet används för att läsa och skriva data till och från I/O modulen genom att använda dess modbusadresser.
I det här systemet fungerar OpenPLC som en Modbus-master, och I/O-modulen fungerar som en Modbus Slave. OpenPLC skickar förfrågningar till slave-enheterna för att hämta data från deras ingångar, eller för att skicka data till deras utgångar. För att testa kommunikationen används en lampa och switch.



Figur 4.1: Konstruktion av OpenPLC testsystem. Modifierad från [7]. *Figuren visar ett PLC-system där en serverdator styr och kontrollerar en lampa samt en switch via Modbus-TCP. Här agerar datorn som en Modbus-master och I/O-modulen som en Modbus-slave. Datorn och I/O-modulen är ihopkopplade med en nätverkskabel.*

4.2 Serverdator

I systemet används en serverdator men det är också möjligt att använda en vanlig PC. Skillnaden mellan en serverdator och en vanlig PC på en arbetsplats ligger främst i deras funktioner och användningsområden. Varför en serverdator används och rekommenderas är främst för att den inte är avsedd för individuell användning. Den är optimerad för kontinuerlig drift, hantering av tunga arbetsbelastningar och skalbarhet för att möta behoven hos flera användare eller applikationer samtidigt. Dessutom har serverdatorn inbyggda funktioner för fjärråtkomst, övervakning och konfiguration, vilket gör det möjligt att hantera dem utan att fysiskt vara på plats.

4.3 Nätverkskommunikation

Att använda nätverksbaserad kommunikation mellan dator och I/O modul är ett val som baseras på flera olika aspekter. Det finns många fördelar med nätverkskommunikation om man exempelvis jämför med kommunikation via USB som också skulle kunna vara möjligt i detta fall. Nätverksbaserad kommunikation, såsom Ethernet eller trådlöst nätverk, möjliggör centraliserad styrning av flera I/O-enheter från en enda plats. Dessutom kan fjärråtkomst och övervakning av enheter göras från vilken plats som helst med nätverksanslutning, vilket ger en hög grad av hanterbarhet.

Nätverksbaserad kommunikation ger även flexibilitet och skalbarhet. Genom att utnyttja befintliga nätverksinfrastrukturer blir det enkelt ansluta och styra I/O-enheter över stora avstånd. Detta är särskilt användbart om enheterna är fysiskt spridda över ett område eller om det önskas lägga till eller ta bort enheter utan omfattande fysiska anslutningsändringar.

En annan fördel med nätverksbaserad kommunikation är enkel integration med andra system och applikationer. Genom att använda standardiserade nätverksprotokoll går det integrera I/O-styrning med exempelvis databaser, molntjänster eller överordnade styrningssystem. Detta öppnar för automatisering, datainsamling och bredare användning av I/O-data.

4.4 I/O-modul och kommunikationsprotokoll

En I/O modul som fungerar ihop med OpenPLC väljs. Antalet ingångar och utgångar är inte till stor vikt då funktionen testas i labbmiljö där det endast krävs ett fåtal in- och utgångar för funktionstesterna. I en anläggning eller i ett verkligt projekt där metoden ska implementeras är det väldigt viktigt att välja en I/O-modul som har tillräckligt med ingångar och utgångar för projektspecifika behov. Det är också viktigt att modulen stöder de I/O-typer som krävs för applikationer som används i projektet, såsom digitala ingångar och utgångar, analoga ingångar och utgångar, eller pulserande ingångar.

4.4.1 Val av I/O-modul

I/O modulen som valts är en Modicon M241 PLC och den valdes då den är kompatibel med olika kommunikationsprotokoll men framför allt att den fanns tillgänglig fysiskt hos Acobia. Modulen måste dock programmeras om för att agera som en enkel nätverks-I/O eller se till att kommunikationen från serverdatorn sker direkt till I/O-portarna utan att ladda ner logik till CPU:n.

4.4.2 Val av kommunikationsprotokoll

I projektet används kommunikationsprotokollet Modbus TCP/IP. Protokollet är en standard som används för Modicon M241-modulen. Modbus är också ett av de vanligaste förekommande kommunikationsprotokollen i industriella informations- och styrsystem i flera verksamhetsområden i Sverige [12]. Modbus TCP/IP är enligt öppen standard, vilket innebär att det är standardiserat och tillgängligt för allmän användning.

4.4.3 Kommunikationshastighet

Modbus TCP/IP stöder en räckvidd av kommunikationshastigheter, vanligtvis från 10 Mbps till 100 Mbps för effektiv dataöverföring. I projektet används Modicons standardinställningar vilket är 100Mbps.

5. GENOMFÖRANDE

Kapitlet beskriver genomförandet, vilket innefattar installationer och förberedelser samt hur testprogrammen är konstruerade.

5.1 Installation OpenPLC

OpenPLC editor och OpenPLC runtime installeras enkelt genom att följa instruktioner via OpenPLC:s hemsida. OpenPLC Editor kan köras på vilken plattform som helst som har stöd för Python. Det finns officiella installationsprogram för: Windows Linux (Debian, Ubuntu, Fedora och varianter) macOS (i beta-version). Installationsguide finns för dessa via hemsidan.

Cykeltid för programexekveringen kan väljas i inställningarna på OpenPLC runtime och allt från 1 ms uppåt går att ställa in. Cykeltiden är den tid som går mellan varje exekvering av programmet.

5.2 Konfigurering av Modicon M241

Modicon M241 programmeras med Ecostruxture Machine Expert (tidigare version SoMachine).

5.2.1 Modbus TCP slave device

För att I/O-modulen ska kunna ta emot och skicka signaler i rätt protokoll måste den konfigureras. I detta fall konfigureras modulen till Modbus TCP slave device enligt Schneiders Modicon M241 User Guide sida 126 [13]. Modbus-kommunikation sker via port 502 och därför måste denna port vara öppen i brandväggsinställningarna.

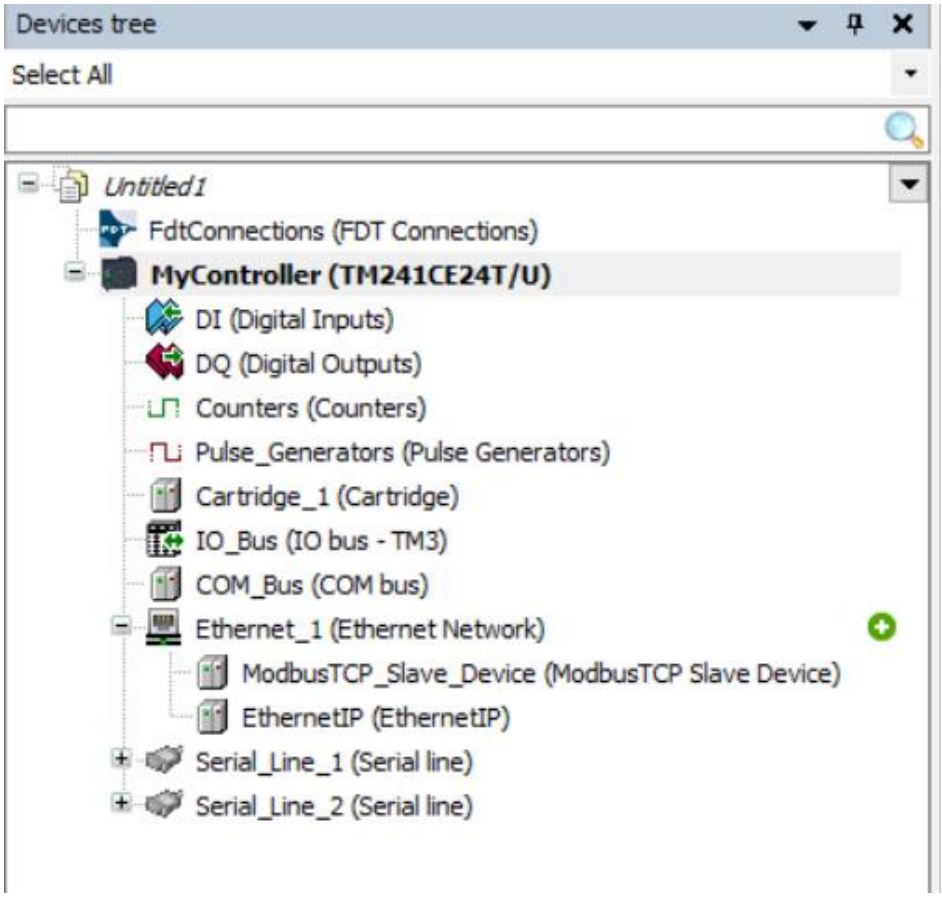
5.2.2 I/O mappning

Modicon har en inbyggd PLC och denna kopplas bort för att I/O-modulen ska kunna agera som en simpel nätverks-I/O. I ett verkligt system är tanken att använda en simpel nätverks-I/O direkt och detta steg är därför inte relevant för ett verkligt system. För komma runt CPUn kopplas I/O-adresserna på Modicon till modbusadresserna genom att tilldela respektive modbusadressvärde till I/O adressen och vice versa. Resultatet blir att de fysiska I/O-portarna kan styras via Modbus direkt med hjälp av OpenPLC som agerar som en modbus master.

5.2.3 Omprogrammering till nätverks-I/O

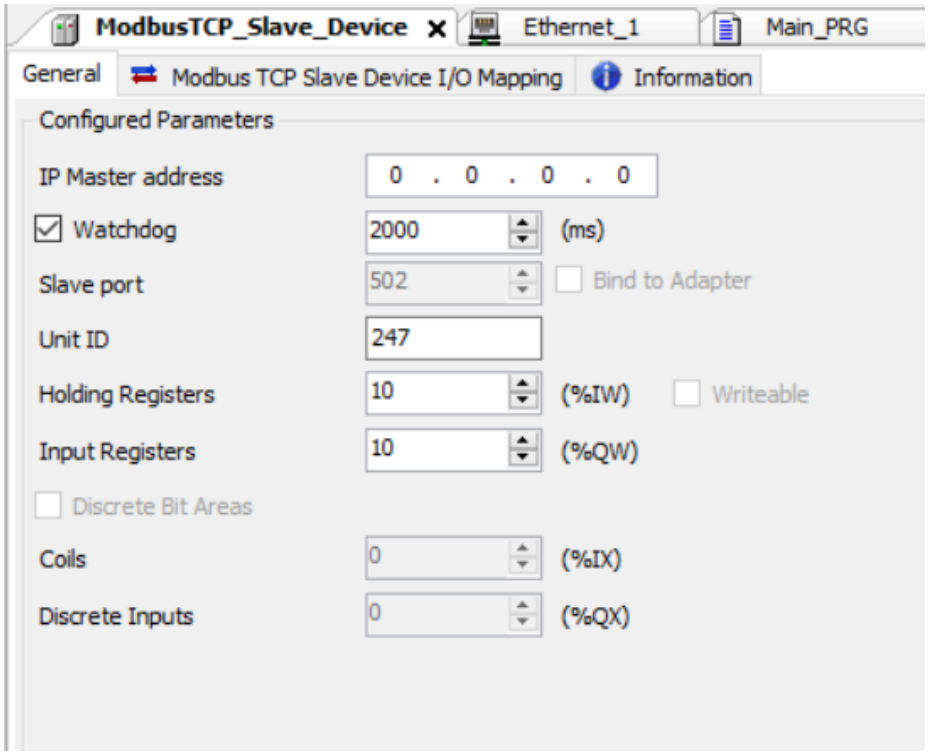
Ett program skapas i Machine Expert och logik implementeras för att koppla de mjukvarubaserade I/O-punkterna till modulens fysiska I/O-adresser. Konfigurationen anpassas automatiskt genom att välja rätt enhet i Devices tree, i detta fall en Modicon TM241CE24T, se figur 5.1. Därefter hanteras inställningar för nödvändiga holding- och inputregister i ModbusTCP_Slave_Device, se figur 5.2. För att säkerställa att in- och utgångsvärdena returneras korrekt och att Modicon-enheten fungerar som en enkel nätverks-I/O, ansluts MyController till slave-enheten genom att använda två rader strukturerad textkod se Figur 5.3. Värdet på "Digital Inputs" har skrivits till "slave device input registers", och värdet på "Slave device holding registers" har skrivits till "Digital Outputs".

Efter att Modicon anslutits via USB-porten på enheten har programmet laddats upp till modulen. Sedan kan modulen användas som en distribuerad I/O-modul och kommunicera via Modbus TCP-protokollet. När konfigurationen är klar kan Modicon M241-modulen klassificeras som en enkel I/O-modul som agerar som Modbus slave. Machine Expert behövs sedan inte mer för att använda I/O:n.



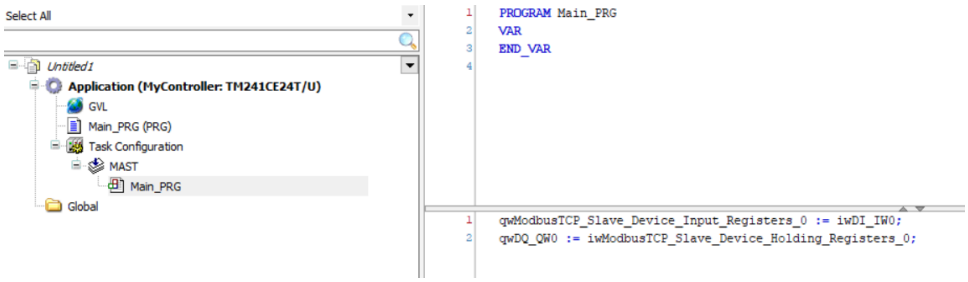
Figur 5.1: Devices tree i programmet Machine Expert

Plustecknet till höger om Ethernet_1 i figur 5.1 används för att lägga till en slave device.



Figur 5.2: Inställningar slave device

Det gör ingenting att fler in- och utgångsregister konfigureras i inställningarna (figur 5.2) än vad som kommer användas. Det kommer inte påverka prestandan.



Figur 5.3: Strukturerad text-kod för att ge värden till in- och utgångar

Figur 5.3 visar de rader kod som krävs i denna konfigurering för att returnera värden till in- och utgångar.

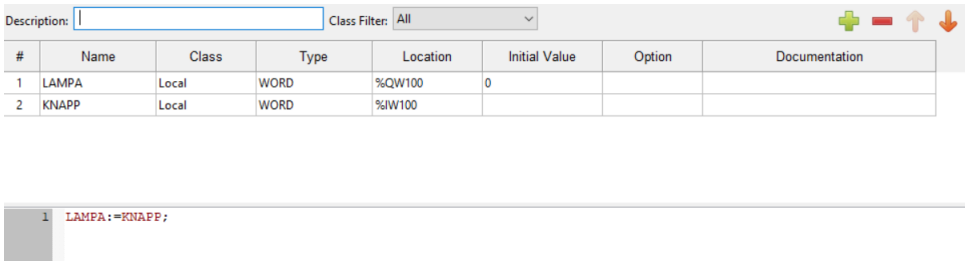
5.3 Styrning via OpenPLC

För att styra I/O-modulen direkt via OpenPLC används OpenPLC-Runtime. Modulen måste konfigureras, vilket görs via en flik i Runtime där startadress samt antal adresser för varje register fylls i för in- och utgångar på varje I/O-modul. PLC-kod skrivs sedan in i OpenPLC-editor som använder dessa register för att läsa och skriva data till I/O-modulen [14]. Programmet laddas upp och körs i PLC-Runtime. Runtime för Linux- och Windows har en Modbus-server som körs som standard på TCP-port 502. Det krävs därför bara en nätverkskabel för att kommunicera mellan servern och I/O-modulen.

Viktigt att notera är att kommunikation mellan I/O-modul och serverdator sker på samma nätverk som all annan kommunikation serverdatorn utför, vilket betyder att prioriteringar och belastning kan skapa fördröjningar som kan vara oönskade och skapa problem. Hänsyn till detta krävs under testning såväl som analys av testresultat.

5.4 Testprogram lampstyrning fördröjning med brytare

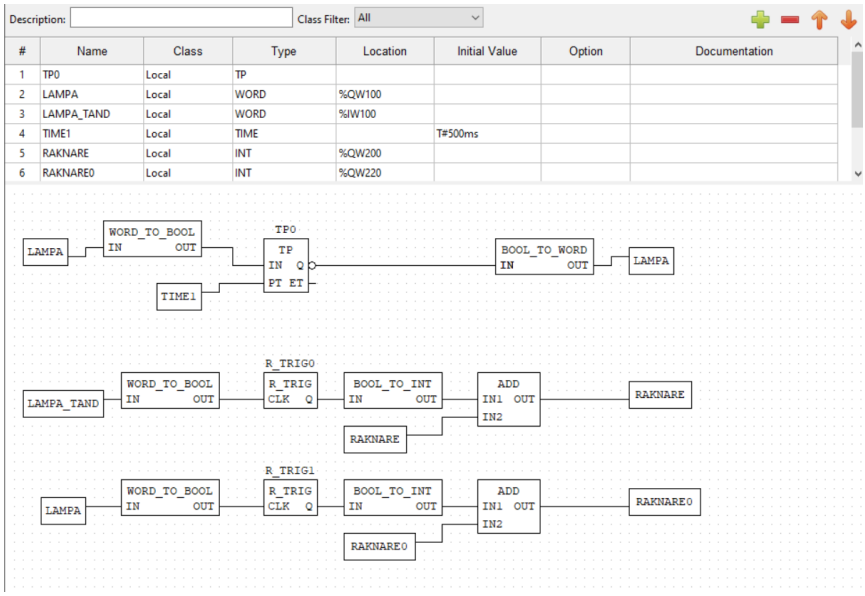
Testprogrammet skrevs med en rad kod på språket strukturerad text i OpenPLC-editor, där värdet på knappen skrivs över till lampan varje gång en ny cykel körs. Se figur 5.4 nedan.



Figur 5.4: Testprogram för lampstyrning

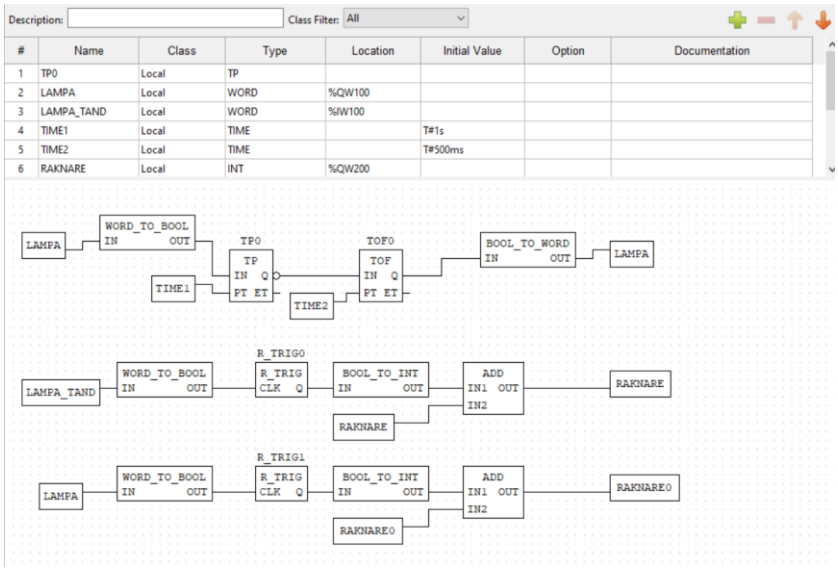
5.5 Testprogram prestanda över tid

För att undersöka om alla signaler som borde levererats fram och tillbaka mellan programmet och I/O faktiskt levererats utvecklades följande program bestående av en klocka som skickar en klockpuls till lampan varje 500ms, och två räknare. RAKNARE0 räknar upp varje gång en klockpuls skickas från programmet till lampan. När lampan fysiskt lyser tar systemet emot en signal från lampan och RAKNARE räknar upp. Programmet visas nedan i figur 5.5.



Figur 5.5: Testprogram funktionalitet för pulser

En variation på klockpulsprogrammet utvecklades också, vilket är nästan identiskt till det första, med skillnaden att signalen som skickas till lampan är hög i 500ms, och att klockan skickar signalen en gång i sekunden istället för varje 500ms. Detta betyder att lampan tänds i 500ms och släcks i 500ms, vilket upprepas så länge programmet kör. Programmet visas nedan i figur 5.6.



Figur 5.6: Testprogram funktionalitet med signal hög 500ms

6. RESULTAT

I detta kapitel presenteras de resultat som kommits fram till genom testerna, samt mer ingående analys av resultaten.

6.1 Tidsfördröjning/svarstid

Till testet användes programmet från avsnitt 5.4, Testprogram lampstyrning fördröjning med brytare.

Cykeltid sattes först till 1 ms, och vid test skiljde det 30 ms mellan lampornas tändning. Testet gjordes sedan om för 5, 10, 20, 40, 60, 80, 100 och 120 ms och resultatet sammanställdes i tabell 6.1 nedan.

Tabell 6.1: Responstider för olika cykeltider

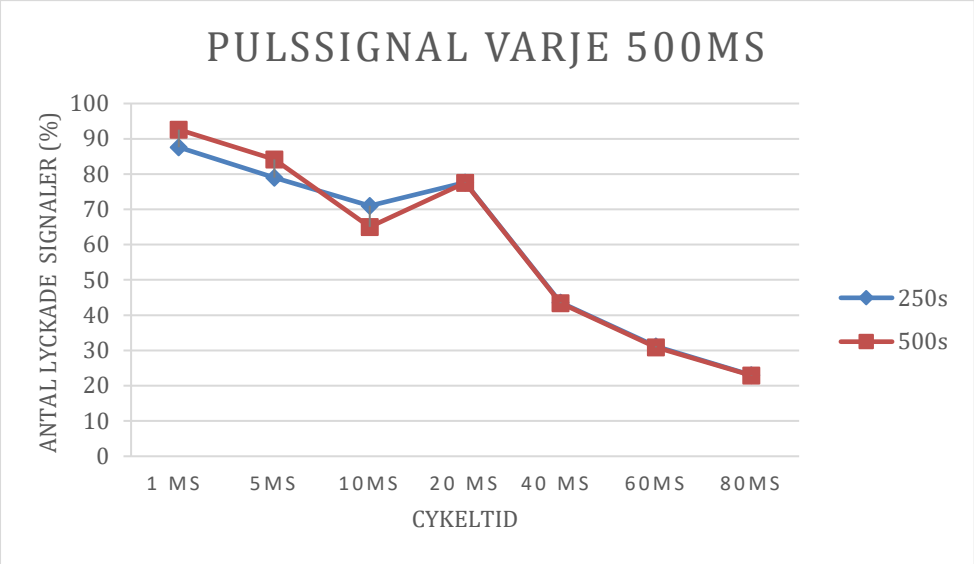
Cykeltid (ms)	1	5	10	20	40	60	80	100	120
Snabbast uppmätt responstid (ms)	30	50	30	30	70	90	150	140	150
Långsammast uppmätt responstid (ms)	30	60	60	90	90	140	160	230	360

För de olika cykeltider som testades var de snabbaste svaren 30 ms mer än cykeltiden. Några av cykeltiderna tog längre tid än 30 ms som snabbast men detta beror mest sannolikt på tajming, vilket betyder att om testet utförts fler gånger på dessa skulle fördröjningen bli 30 ms även där. Andra avvikelser som upptäcktes var att vid cykeltiderna 10 och 20 ms respektive förekom fördröjningar som var kortare än 30 ms, vilket skulle betyda att signalerna ibland kunde skickas snabbare än annars uppmätt. Dessa avvikelser förekom dock ytterst få gånger av många tester och därför tolkas resultatet som att en fördröjning på 30 ms i allmänhet förekommer vid kommunikationen mellan programmet på serverdatorn och I/O-enheten, oavsett inställd cykeltid.

Val av cykeltid blir en fråga om prioriteringar. Om hög snabbhet är högsta prioritet vore 1 ms cykeltid lämplig då detta garanterar snabbast möjliga respons vid normalbelastad drift, men om prestanda och tålighet anses viktigare kan högre cykeltider vara mer lämpliga då detta inte belastar serverdatorn lika mycket, eftersom detta innebär att färre cykler körs under samma tidsperiod. Det är också värt att notera att fördröjningen kan öka om det finns andra applikationer som körs samtidigt på serverdatorn och tar upp dess resurser (se avsnittet om belastning).

6.2 Funktionskontroll

Funktionskontrollen gjordes med pulssignaler varje 500 ms med olika cykeltider, enligt program ett i 5.5 Testprogram prestanda över tid. För varje cykeltid lästes data av efter 250 och 500 sekunder, och en tredje gång efter en tillräckligt lång tid för att mätdata inte längre skulle variera. Se figur 6.1 och tabell 6.2 för de första två värdena.



Figur 6.1: Prestanda för pulssignal vid olika cykeltider

Tabell 6.2: Prestanda för pulssignal vid olika cykeltider

Cykeltid	1ms	5ms	10ms	20ms	40ms	60ms	80ms
Antal överförda signaler under 250s	87,6 %	79 %	71 %	77,6 %	43,6 %	31,2 %	23 %
Antal överförda signaler under 500s	92,6 %	84,2 %	65 %	77,5 %	43,4 %	30,9 %	22,9 %

Efter en längre tid konvergerade resultatet för respektive cykeltid mot medelvärdet mellan data för 250 och 500 sekunder. Eftersom det inte var vid en exakt tid den tredje nämnda data mättes utan bara efter godtyckligt lång tid exkluderas data från tabell och figur. Att data efter längre tid närmade sig medelvärdet mellan data för de två kortare tiderna visar på att det inte förekommer stora avvikelser i prestandan. Procentsatserna uppnåddes genom att dividera antalet gånger lampan fysiskt tändes med antalet gånger den borde tänts, multiplicerat med 100.

Funktionstestet med pulssignaler under normalbelastad drift fungerade inte bra, och detta berodde med hög sannolikhet på att signalerna var höga för kort tid för att de skulle kunna levereras i tid, med tidsfördröjning inräknat. Med högre cykeltider blev problemet bara större, och med en cykeltid på 40ms var lyckade signaler nere på under 50%.

Det andra testet gjordes med program två i 5.5 Testprogram prestanda över tid, och procentsatserna uppnåddes på samma sätt som i föregående test. Se tabell 6.3 för resultat.

Tabell 6.3: Resultat på test där signalen är hög längre än cykeltiden

	Överförda signaler under 24h	
Cykeltid	469ms	450ms
Överförda signaler	99,98%	100%

Under normalbelastad drift med signaler som skickades höga i 500ms varje sekund lyckades 100% av signalerna tända lampan i ett dygn, så länge cykeltiden var 450ms eller lägre. Detta visar på en hög funktions- och driftsäkerhet i systemet och nätverkskommunikationen. Anledningen till att 100% av signalerna inte lyckades när cykeltiden var högre berodde troligtvis på att tidsfördröjningen kunde variera något, men i och med att Modbus-TCP/IP användes kunde signaler levereras en liten stund efter att de skickats och på så sätt komma ifatt igen. Detta beror på att TCP/IP kontrollerar att varje signal kommit fram. Att signalerna kunde levereras lite sent gav extra marginal till systemet vilket gjorde att även om tidsfördröjningen kan ha ökat vid något tillfälle blev det aldrig för mycket för att signalen skulle komma fram, så länge cykeltiden var max 50ms mindre än tiden signalen var hög.

6.3 Belastningstest

För belastningstestet återanvändes båda programmen från 5.5 Testprogram prestanda över tid. Den stora skillnaden är att serverdatorns CPU under dessa tester konstant belastas mer än 94% och minnet mer än 90%. Själva belastningen uppnåddes med hjälp av tunga Matlab-simulationer som kördes parallellt med testningen av programmet. Belastningsvärdena lästes kontinuerligt av med hjälp av aktivitetshanteraren. Lyckade signaler är här på samma sätt som i avsnitt 6.1 definierat som antal signaler som skickades av programmet som fysiskt tände lampan. Resultat är presenterat i tabell 6.4.

Tabell 6.4: Belastat system, pulssignaler

Antal signaler skickade	1000 signaler
Lyckade signaler, Cykeltid 1 ms	60%

När systemet med pulssignaler belastades högt visade sig prestandan bli så dålig att det inte fanns någon anledning att utföra vidare tester på systemet. I detta beslut vägdes även resultatet från funktionstestet in.

Tester med signal hög i 500ms utfördes på liknande sätt som för pulssignaler, men data mättes efter 1000, 3000 och över 10 000 signaler och olika cykeltider användes för att uppnå så hög prestanda som möjligt, vilket är framställt i tabell 6.5 nedan.

Tabell 6.5: Belastat system, signal hög i 500ms

Antal signaler skickade	1000	3000	Över 10 000
Lyckade signaler, Cykeltid 400 ms	97,1%	97,8%	98%
Lyckade signaler, Cykeltid 300 ms	99,4%	99,5%	99,6%
Lyckade signaler, Cykeltid 250 ms	-	-	99,97%

En upptäckt som gjordes under testning är att när CPU är 95%-100% belastad är det bara 98% av signalerna från programmet som skickas iväg till I/O, oavsett vad cykeltiden är satt till och oavsett om signalen är en puls eller en längre signal. Detta betyder att om 1000 signaler skickats är det ungefär 1021 signaler som borde skickats. Detta fenomen benämns vidare som belastningsbortfall. Se tabell 6.6 nedan och jämför med tabell 6.5.

Tabell 6.6: Belastningsbortfall, signal hög i 500ms

Antal signaler som borde skickats	1000	3000	Över 10 000
Lyckade signaler, Cykeltid 400 ms	95,16%	95,84%	96,04%
Lyckade signaler, Cykeltid 300 ms	97,41%	97,51%	97,61%
Lyckade signaler, Cykeltid 250 ms	-	-	97,97%

När serverdatorn belastades nära sin maxkapacitet blev responstiden längre och många signaler som skulle levererats i normalbelastad drift kom aldrig fram. Antal lyckade signaler gick ner med upp till 3% jämfört med det normalbelastade fallet. I detta fall var cykeltiden satt på 400 ms och signalen som skickades var hög i 500 ms. När denna längre responstid togs mer hänsyn till och lägre cykeltider testades för systemet blev resultatet dock bättre. Upp till 99,97% lyckade signaler uppnåddes när cykeltiden var satt på 250ms under en längre period. Ett annat problem som dök upp var belastningsbortfallet som upptäcktes under dessa tester, vilket innebar att under hög belastning (över 94%) var det 2% av signalerna som borde skickats av programmet som aldrig skickades, vilket innebär att även om 99,97% av signaler som skickades levererades var det bara ungefär 98% av de som borde skickats som levererades. Detta visar på vikten av att dedicera tillräckligt med prestanda till servern eller datorn som skall agera master för att belastningen aldrig skall uppkomma i dessa nivåer.

Trots att signaler föll bort vid hög belastning finns det inget som tyder på att signaler tolkats på fel sätt i kommunikationsprotokollet, vilket betyder att även om inte alla signaler kommit fram som tänkt, påverkades inte de signaler som faktiskt skickades.

6.4 Avbrott i kommunikationen

För detta test användes program två från kapitel 5.5, Testprogram lampstyrning tid. Programmet startades och nätverkssladden rycktes ut, vilket resulterade i att signalen för att

tända lampan fortsatte skickas från programmet, men I/O:n tog ej emot signalen och lampan kunde därför inte tändas.

Lampan behöll sitt senaste värde vid utryckning av sladden, vilket betyder att om den var tänd när sladden rycktes ut, fortsatte den vara tänd tills uppkoppling återupprättats. Efter att nätverkssladden kopplades in igen tog det tre sekunder för signalen att återupprättas. Eftersom programmet fortsatte köra även om det tappat signalen till I/O:n resulterade detta i att de räknare som är inkluderade i testprogrammet fick en konstant förskjutning mellan sig. Detta test upprepades flera gånger med samma resultat.

Vid avbrott bröts kommunikationen mellan I/O och serverdatorn, och I/O:n behöll sina senaste värden på samtliga ingångar. Att senaste värdet behölls beror på konfigureringen i I/O-enheten och kan sannolikt ändras utifrån preferens. Det största problemet som uppstod vid avbrott var just att programmet fortsatte exekvera även när ingen uppkoppling fanns till I/O, vilket i testfallet innebar att den ena räknaren (som räknade upp varje gång signal skickades) fortsatte räkna, medan den andra räknaren (som räknade upp varje gång signal levererades) inte räknade upp alls. Detta skulle kunna bli ett stort problem i en processkedja med flera ingående sekvenser, då ett kommunikationsavbrott i många fall borde stoppa exekveringen där den är. OpenPLC går ej ut med tydlig information om detta går att ta hänsyn till någonstans i inställningarna, vilket betyder att detta är en stor förbättringspotential hos denna alternativa lösning.

En typ av avbrott som skulle kunna förekomma är att serverdatorn som utför programmet skulle kunna starta om för att installera en uppdatering eller liknande. Detta förekom aldrig under tester men det är viktigt att ta i beaktning och utvärdera hur denna typ av avbrott skulle skada systemet, och hur det skulle kunna undvikas. Ett sätt att undvika komplikationer är genom att i inställningarna på datorn bestämma specifika tider då denna typ av omstarter kan förekomma och att de behöver godkännas av administratör, vilket skulle ta bort slumpfaktorn i det hela. Processer som kör 24 timmar om dygnet är därför i nuläget inte helt lämpliga för denna typ av system såvida inte redundans kan tillämpas.

6.5 Nätverksfördröjning

Testet utfördes både över Wifi och med nätverkskabel. Vid 200 genomförda pings över wifi var packet loss 0% och responstiden låg i ett spann mellan 1 och 20 ms med ett snitt på 4 ms. Vid 200 genomförda pings med nätverkskabel var packet loss 0% och responstiden var konstant mindre än 1ms.

Nätverksfördröjningstesterna visade på en pålitlig och jämn prestanda, och att det även skulle vara möjligt att köra systemet trådlöst om I/O-enheten tillåter denna typ av anslutning. För bästa möjliga prestanda är det dock alltid bättre att använda nätverkskabel än trådlös anslutning.

7. DISKUSSION OCH SLUTSATS

7.1 Testresultat

Resultaten från samtliga tester visade på att det är fullt möjligt att med OpenPLC implementera system som utför instruktioner på ett tillförlitligt sätt. Några viktiga saker att tänka på är att programmera så att hänsyn tas till signalfördröjningar och cykeltider, och att systemet inte riskerar att belastas alltför mycket.

Under normal belastning är det testade systemet högt jämförbart med ett fysiskt PLC-system, med liknande svarstid och mycket hög prestanda och tillförlitlighet, så länge hänsyn tas till de faktorer som nämns i stycket ovan. Eftersom inget fysiskt PLC-system testats med exakt samma funktioner kan ingen direkt jämförelse göras mer än i teorin.

Det är högst lämpligt att använda ett separat nätverkskort till systemet och inte dela kommunikationen med andra processer på serverdatorn, eftersom detta skulle kunna skapa oförutsägbara fördröjningar. Dessa fördröjningar skulle i sin tur kunna skapa problem om det krävs att systemet levererar signaler inom ett jämnt intervall eftersom kommunikationsprotokollet TCP/IP kan leverera signaler sent och ibland missa deadlines helt. TCP/IP är därför inte heller lämpligt för system som ställer dessa krav. Mer ingående analys av testerna finns under respektive resultatavsnitt.

7.2 Redundans

Redundanta master-enheter kan i de flesta fall konfigureras för vanliga PLC-system, och det finns goda chanser att det även skulle kunna fungera för ett system uppbyggt i OpenPLC, men i nuläget finns det inga bekräftade fall där ett sådant system programmerats, och testning av redundans exkluderades från detta arbete då varken hårdvara eller tillräcklig tid fanns för att utförligt ta reda på om det skulle vara möjligt. Därför kommer utgångspunkten vara att OpenPLC inte i nuläget är lämpligt för system som kräver redundanta CPU:er, men det är en intressant fråga för framtida undersökningar.

7.3 Säkerhetssystem

När det gäller säkerhetssystem är det viktigt att systemet är robust och tillförlitligt för att uppfylla sitt syfte. En viktig faktor i säkerhetssystem är redundans, det vill säga att det finns backup-system som kan ta över om huvudsystemet fallerar. Det är också viktigt att systemet har hög tillgänglighet och att eventuella fel upptäcks så snabbt som möjligt för att minimera risken för skador och olyckor.

Det finns flera säkerhetsaspekter att ta hänsyn till när man använder en server för att styra I/O via runtime. För det första är det viktigt att systemet är konfigurerat korrekt och att all kommunikation är säkerställd så att obehöriga inte kan få åtkomst till systemet. Det är också

viktigt att ha en backup-plan för om servern skulle fallera, för att säkerställa att systemet fortfarande kan fungera och skydda mot farliga händelser.

Då ingen djupdykning gjorts i redundans är det svårt att veta om möjligheten för säkerhetsfunktioner går att styra med denna typ av system. Slutsatsen som kan dras med den data som framkommit är att serverstyrd PLC kan användas för icke säkerhetskritiska funktioner, där kraven på säkerhet inte är lika höga. En PLC som körs via en server är billigare och väldigt flexibel med hårdvara vilket gör den lämplig för mindre kritiska applikationer där säkerhetskraven är lägre. Något som kan vara intressant att studera är att kombinera en serverstyrd I/O i mer kritiska system där en säkerhets-PLC används för de säkerhetskritiska funktionerna. OpenPLC kan tillhandahålla tillräckligt med funktionalitet för att utföra logik som krävs för icke säkerhetskritiska funktioner.

7.4 Cybersäkerhet och nät-trafik

Det finns flera säkerhetsaspekter att ta hänsyn till när man använder ett serverbaserad system. Kommunikationen mellan servern och I/O-enheterna kan vara sårbar för attacker från externa nätverk, särskilt om man använder en öppen kommunikationskanal som internet. Detta kan leda till obehörig åtkomst, sabotage, dataintrång och andra säkerhetsproblem. Servern i sig kan också vara sårbar för angrepp, särskilt om man inte har tillräcklig säkerhet på plats, t.ex. brandväggar, antivirusprogram och så vidare. Om servern blir utsatt för en attack kan det resultera i att I/O-enheter kontrolleras på ett felaktigt sätt eller inte alls, vilket kan leda till farliga situationer i systemet.

7.4.1 Separat nätverk

Ett separat/avskärmat nätverk som används endast till utrustningen kan minska sårbarheten för säkerhetsproblem. Risken för att obehöriga ska kunna få tillgång till programmet och manipulera det minskar genom att använda ett separat nätverket. Ett separat nätverk kan också minska risken för att nätverket blir överbelastat, vilket kan orsaka förseningar och fel i överföringen av data.

Trots att ett separat nätverk kan minska sårbarheten, bör man ta till hänsyn till andra säkerhetsåtgärder. Ett privat avskärmat nätverk kan fortfarande bli angripet genom felaktiga konfigurationer, obehörig åtkomst eller användning av osäkra protokoll.

Ett separat nätverk är inte är en absolut lösning på säkerhetsproblemen som kan uppstå. Istället bör det användas som en del av en större säkerhetsstrategi som inkluderar andra tekniska och organisatoriska åtgärder.

7.5 Fördelar och nackdelar med OpenPLC

OpenPLC är ett kostnadseffektivt och flexibelt alternativ eftersom användare kan nyttja befintlig hårdvara i stället för att köpa PLC-hårdvara som ofta är låst till specifik mjukvara och

annan specifik hårdvara. Systemet innebär däremot en större sårbarhet för cyberattacker eftersom programvaran körs på en serverdator som är ansluten till ett nätverk, vilket skulle kräva fler steg för de som riskerar bli utsatta. Dessutom kan det finnas prestandaproblem om programvaran inte optimeras för den specifika hårdvaran.

Sammanfattningsvis, fördelar och nackdelar som går att presentera utifrån studien:

Fördelar med OpenPLC:

1. Kostnadseffektivt: OpenPLC är gratis och har öppen källkod, vilket innebär att det inte finns några kostnader för programvarulicenser.
2. Flexibilitet: Eftersom OpenPLC körs på en serverdator, kan användare enkelt uppdatera programvaran och konfigurera om I/O-anslutningar utan att behöva byta ut hårdvaran.
3. Användarvänlighet: OpenPLC har ett användarvänligt gränssnitt för att skapa programvara. Det finns också många exempelprogram och instruktioner tillgängliga online för att hjälpa användare att komma igång. I editorn används de standardiserade PLC-programmeringsspråken.
4. Integrering med andra system: OpenPLC använder standardiserade protokoll som Modbus TCP för kommunikation med andra enheter, vilket gör det lätt att integrera med andra system och enheter.
5. Tålighet: Tål relativt höga belastningar utan att prestandan påverkas, vid väldigt höga belastningar försvinner viss funktionalitet.
6. Funktionalitet: Fungerar i enkla fall lika bra som en traditionell PLC.

Nackdelar med OpenPLC:

1. Säkerhet: Eftersom OpenPLC körs på en uppkopplad dator, finns det en högre risk för cyberattacker jämfört med traditionella PLC-system.
2. Pålitlighet: Eftersom OpenPLC är beroende av en serverdator, kan det uppstå driftstopp om serverdatorn stannar eller förlorar anslutningen till nätverket.
3. Prestanda: Prestandan på OpenPLC kan vara lägre jämfört med traditionella hårdvarubaserade PLC-system beroende på hur kraftfull processor serverdatorn har.
4. Funktionalitet: I vissa specifika fall där högsta precision och tillförlitlighet krävs lämpar sig inte OpenPLC så bra som alternativ.

7.6 Implementeringsmöjligheter

Det finns helt klart potential hos OpenPLC som alternativ till klassiska PLC-system, och med teori och tester i åtanke är några av de processer som skulle kunna vara lämpliga för denna typ av alternativ:

1. Ventilationssystem i miljöer där specifikt klimat inte är kritiskt, exempelvis i en lagerlokal eller kontorsmiljö. Där en sekund hit eller dit inte gör någon större skillnad och framförallt inte gör någon skada på person eller utrustning. Samma funktioner som ett traditionellt PLC-system skulle erbjuda är okomplicerat uppnått eftersom OpenPLC-

programmet kan programmeras på precis samma sätt, och det är bara i prestandan det kan finnas små skillnader.
2. Belysningsautomation
3. Enkla, mindre delar av processkedjor, exempelvis ett rullband med givare som läser av olika förpackningar och sorterar dem i olika fack.

En stor anledning till att vi inte rekommenderar denna lösning till styrning av större processer är att det i nuläget verkar osäkert med att eventuella kommunikationsavbrott orsakar missmatchningar i exekveringen av programmet. Om OpenPLC skulle ha eller skulle utveckla alternativ för att pausa programexekveringen om kommunikation bryts skulle större processer också vara möjliga att implementera, men det skulle också kräva mer fördjupade tester för att säkerställa att prestandan håller.

7.7 Teknikens roll i samhället

Då denna lösning i praktiken inte skiljer sig speciellt mycket från traditionella PLC-system skiljer sig inte heller de etiska aspekterna. Implementering är snarlik och erbjuder snarare ett mer flexibelt sätt att styra processer. Den största skillnaden är just att OpenPLC erbjuder en uppkopplad lösning, vilket leder till större behov av beaktning när det kommer till cybersäkerhet, men som diskuterats tidigare finns det metoder för att skydda även denna typ av system. Miljömässigt är det bättre att bruka materiel som finns på plats än att behöva köpa in nya PLC-enheter, vilket innebär att det i många fall skulle vara bättre att använda denna alternativa lösning i de fall serverdatorer finns tillgängliga.

7.8 Fortsatt arbete

En viktig fråga att besvara för att utvidga implementeringsmöjligheterna markant är den om redundanta CPU:er. En fråga som vi inte lyckades besvara men som är högst relevant inom fältet och som skulle göra stor skillnad för resultatet av detta arbete.

En annan intressant fråga är hur man åtgärdar problemet med fortlöpande programexekvering vid kommunikationsavbrott. Om redundans är möjlig, kan dessa frågor kopplas ihop? Med svar på redundansfrågan kommer vi också in mer på säkerhets-PLC och diskussionen från kapitel 7.3 blir alltmer aktuell.

Andra mjukvaror än OpenPLC skulle även kunna testas och en ”Proof of Concept” skulle kunna utföras på ett mer verkligt system.

Referenser

[1] X. Mao, Y. Zhang, J. Shi, Y. Huang, and Q. Li, “A refinement development approach for enhancing the safety of PLC programs with Event-B,” *Sci Comput Program*, vol. 215, p. 102763, Mar. 2022, doi: 10.1016/J.SCICO.2021.102763.

[2] U. George-Andrei, P. Olga, and U. Maria, “PLCs’ Inputs and Outputs Response Time Testing Application,” *SIELMEN 2021 - Proceedings of the 11th International Conference on Electromechanical and Energy Systems*, pp. 57–60, 2021, doi: 10.1109/SIELMEN53755.2021.9600400.

[3] A. John, R. Varghese, S. S. Krishnan, S. Thomas, T. A. Swayambu, and P. Thasneem, “Automation of 11 kv substation using raspberry pi,” *Proceedings of IEEE International Conference on Circuit, Power and Computing Technologies, ICCPCT 2017*, Oct. 2017, doi: 10.1109/ICCPCT.2017.8074264.

[4] Changwen Xie, (2003) "PLC or PC Card" in Fieldbus technology: industrial network standards for real-time distributed control, Berlin Heidelberg: Springer-Verlag

[5] W. Richard. Stevens and K. W. Fall, “TCP/IP illustrated. Volume 1, The protocols,” 2011, Accessed: Apr. 24, 2023. [Online]. Available: https://www.kobo.com/ww/en/ebook/tcp-ip-illustrated-volume-1-the-protocols-2

[6] “MODBUS Application Protocol Specification V1.1b3 Modbus,” 2012, Accessed: Apr. 24, 2023. [Online]. Available: http://www.modbus.org

[7] S. Fujita, · Kosuke Hata, · Akinori Mochizuki, K. Sawada, S. Shin, and S. Hosokawa, “OpenPLC based control system testbed for PLC whitelisting system,” vol. 26, pp. 149–154, 2021, doi: 10.1007/s10015-020-00635-1.

[8] S. Qiao *et al.*, “Research on redundant control mode of coordinated controller based on dual control cores”, doi: 10.1051/e3sconf/202021302006.

[9] D. J. Smith and K. G. L. Simpson, “Safety Critical Systems Handbook: A Straight forward Guide to Functional Safety, IEC 61508 (2010 EDITION) and Related Standards, Including Process IEC 61511 and Machinery IEC 62061 and ISO 13849,” *Safety Critical Systems Handbook: A Straight forward Guide to Functional Safety, IEC 61508 (2010 EDITION) and Related Standards, Including Process IEC 61511 and Machinery IEC 62061 and ISO 13849*, pp. 1–270, Jan. 2010, doi: 10.1016/C2010-0-65791-9.

[10] Sharma, KLS. (2017). “18.1 Introduction “ in Overview of Industrial Process Automation (2nd Edition). Elsevier. Retrieved from https://app.knovel.com/hotlink/pdf/id:kt0119G1H1/overview-industrial-process/safety-sys-introduction introduction&hierarchy=kt0119G1G4&view=collapsed&zoom=1&page=6

[11] Sharma, KLS. (2017). “18.2.4.2 Safety Programmable Logic Controller “ in Overview of Industrial Process Automation (2nd Edition). Elsevier. Retrieved from https://app.knovel.com/hotlink/pdf/id:kt0119G1O7/overview-industrial-process/safety-programmable-logic

[12] “NCS3 - Virtualisering inom industriella informations- och styrsystem : en översikt.” https://www.msb.se/sv/publikationer/stabsmetodik--introduktion2/ (accessed May 14, 2023).

[13] S. Electric, “Modicon M241 Logic Controller - Programming Guide,” 2015, Accessed: Apr. 24, 2023. [Online]. Available: www.schneider-electric.com

[14] “2.6 Slave Devices – OpenPLC.” https://openplcproject.com/docs/2-6-slave-devices/ (accessed May 04, 2023)

INSTITUTIONEN FÖR ELEKTROTEKNIK
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2023
www.chalmers.se



CHALMERS