CODEN:LUTEDX/(TEIE-3132)/1-46/(2024)

Industrial Electrical Engineering and Automation

Undersökning och implementering av IoT-lösning för tillgång till driftinformation och remote support för HiFlex-system

Alma Flygare
Emil Jakobsson

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

Undersökning och implementering av IoT-lösning för tillgång till driftinformation och remote support för HiFlex-system

LUNDS UNIVERSITET
Lunds Tekniska Högskola

LTH vid Campus Helsingborg
Avdelningen för industriell elektroteknik och automation

Examensarbete:
Alma Flygare
Emil Jakobsson

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 2024

Sammanfattning

Det här examensarbetet har handlat om att ta fram en IoT-lösning för Eltech Automations HiFlex-system, som är en standardprodukt med robot, PLC och HMI i samma produkt. Målet med arbetet var att ta fram en lösning för att kunna skicka data från PLC, via en gateway, till en molntjänst för att sedan kunna hämtas och ses av operatörer eller andra användare av HiFlex på en webbsida. Datan som skulle vara tillgänglig var driftinformation som exempelvis batchnummer, antal packade enheter och driftstatus med mera. Det behövde även finnas en lösning för att ansluta sig till PLC:n med VPN för att få remote access och kunna göra enklare felsökningar och omprogrammeringar utan att Eltech Automations ingenjörer skulle behöva ta sig till HiFlexen.

Resultatet blev att en gateway, som hade en färdig lösning för remote access, kombinerades med en virtuell dator. På den virtuella datorn installerades Node-RED, en MySQL-databas samt en Apache2 webbserver vilken tillhandahöll en webbsida. Gateway:n läste av variabler från PLC:n och skickade dessa med HTTPS till Node-RED. Node-RED i sin tur extraherade datan och la in den i databasen. Vid anslutning till eltechconnect.com kom användaren till en inloggningssida där ett konto med mailadress och lösenord skapades för att sedan användas för inloggning. På den inloggade sidan visades datan, som hämtades av webbsidan från databasen, tillhörande de HiFlex som användaren hade behörighet till. Gateway:n som valdes konfigurerades också så att remote support genom VPN var möjlig.

Resultatet blev alltså en IoT-lösning där data från HiFlex visas på en webbsida, med inbyggd möjlighet till remote support via VPN.

Nyckelord: IoT, gateway, HTTPS, PLC, virtuell dator

Abstract

This thesis has focused on developing an IoT solution for Eltech Automation's HiFlex system, which is a standard product with robot, PLC, and HMI integrated into one unit. The objective of the project was to create a solution for sending data from the PLC, via a gateway, to a cloud service, which could then be accessed and viewed by operators or other users of the HiFlex on a web page. The information that needed to be available included production data such as batch numbers, number of packaged units, operational status and more. Additionally, there needed to be a solution for connecting to the PLC via VPN to enable remote access for easier troubleshooting and reprogramming without requiring Eltech Automation's engineers to physically access the HiFlex.

The result was the integration of a gateway, which had a pre-existing solution for remote access, with a virtual machine. Node-RED, a MySQL database, and an Apache2 web server were installed on the virtual machine, providing the web page interface. The gateway read variables from the PLC and sent them via HTTPS to Node-RED. Node-RED extracted the data and stored it in the database. Upon connecting to eltechconnect.com, the user was directed to a login page where an account with an email address and password was created for subsequent logins. Once logged in, the user could view the data associated with the HiFlex systems they had authorization for, as fetched from the database. The gateway that was used was also configured so that remote access was possible using a VPN.

The result was thus an IoT solution where data from HiFlex is displayed on a webpage, with built-in capability for remote support via VPN.

Keywords: IoT, gateway, HTTPS, PLC, virtual machine

Förord

Detta examensarbete inom högskoleingenjörsutbildningen i elektroteknik med automation på Lunds Tekniska Högskola utfördes på Eltech Automation i Lomma.

Eltech Automation har varit mycket hjälpsamma och vi är mycket tacksamma för möjligheten att fått skriva vårt arbete hos er. Extra stort tack till vår handledare Fredrik Svensson och även Peter Ahrens som stöttat oss genom arbetet.

Vi vill även tacka vår handledare Bernard Schmidt och vår examinator Mats Lilja på Lunds Tekniska Högskola för deras stöd och vägledning genom arbetet.

Innehållsförteckning

1. Inledning

Inledningen beskriver syfte, bakgrund och frågeställningarna som arbetet besvarar. I detta kapitel motiveras även arbetet.

1.1. Bakgrund

Eltech Automation har utvecklat en komplett produkt kallad HiFlex som inkluderar robot, PLC-system, HMI och givare med mera. HiFlex-systemen säljs till kunder tillsammans med supporttjänster från Eltech Automation. Idag behöver Eltech Automations ingenjörer ta sig till HiFlex-roboten för att kontrollera status eftersom larm och annan status som exempelvis batchnummer och antal packade enheter för roboten endast syns på HMI:n. Om en kund behöver support måste personal från Eltech Automation åka ut och göra felsökning på plats.

Ett problem är att det i dagsläget inte finns en färdig lösning för hur operatörsinformation visas i en applikation eller på en webbsida. Operatörs-information som vore intressant och hjälpsamt att se är exempelvis produktions-status, batchnummer och antal packade enheter. Ett annat problem är att det inte finns en lösning för hur Eltech Automations ingenjörer kan få remote access till deras HiFlex när den står ute hos kund. Remote access hade möjliggjort remote support vilket hade inneburit att Eltech Automation hade kunnat undvika resor ut till kund vid enklare problem.

En lösning på problemet är att undersöka om det går att kommunicera med HiFlex med en färdig produkt som via 4G eller 5G kan kommunicera med en färdig applikation eller webbsida. Lösningen skulle sedan kunna implementeras efter Eltech Automations behov och krav för att visa operatörsinformation som exempelvis produktionsstatus, batchnummer och antal packade enheter. För att möjliggöra remote support är en uppkoppling via VPN-tunnel en lösning. Detta hade gett Eltech Automations personal möjligheten att koppla upp sig till HiFlex för att se driftinformation och göra eventuella ändringar i program-koden.

Eltech Automation är ett automationsföretag, grundat 1997, med kontor i Lomma och Stockholm med cirka 20 anställda. Deras kunder finns i södra Sverige och Stockholmsregionen (Eltech Automation 2022 a). De erbjuder helhetslösningar

inom automation inom områden som förpackningar, livsmedel och konsument-produkter med mera (Eltech Automation 2022 b).

1.2. Syfte

Syftet med examensarbetet är att undersöka och jämföra olika gateways för att se vilken som är lämpligast. Till gateway:n ska det även finnas en lösning för remote access som möjliggör remote support så att ansluten PLC kan anslutas till och programmeras via VPN. Det ska även undersökas om det finns en färdig lösning för att se produktionsdata som kan implementeras. Om det inte finns ska en sådan tas fram för att sedan verifieras och utvärderas. Det förväntade resultatet är att effektivisera och förenkla användandet samt underhållet av HiFlex både för Eltech Automation och deras kunder som använder sig av HiFlex.

1.3. Målformulering

Examensarbetet ska undersöka om det finns en färdig gateway-lösning som kommunicerar till en färdig app-miljö som sedan kan konfigureras och anpassas efter Eltech Automations behov. Om det inte går att hitta en färdig app-lösningen som är lämplig för ändamålet ska mjukvara i form av en webbsida utvecklas. I app- eller webbside-miljön ska följande operatörsinformation kunna avläsas:

- Produktionsstatus, drift/stopp/larm
- batchnummer
- antal packade enheter
- aktivt larm
- Eltech Automations logga

Det ska även finnas möjlighet för remote access till HiFlex för Eltech Automations ingenjörer. Det vill säga att de ska kunna ansluta sig till HiFlex trådlöst, var de än är, via dator och kunna se potentiella problem samt kunna göra ändringar i programkoden för att ha möjlighet att ge kunden support utan att behöva ta sig till HiFlexen.

1.4. Problemformulering

Dessa frågor kommer att besvaras under examensarbetet.

1. Hur ansluter Eltech Automations ingenjörer sig till HiFlex idag?
2. Vilken typ av IoT-lösning är kompatibel med HiFlex?
3. Vilka kriterier ska användas för att välja IoT-lösning?
4. Vilken IoT-lösning ska väljas?
5. Hur ska IoT-lösningen utvärderas?
6. Hur ska ett testprogram som ska verifiera total funktionalitet utformas?

1.5. Motivering av examensarbete

Examensarbetet valdes för att få möjligheter att fördjupa kunskaperna i trådlös kommunikation och automation. Dessutom fås kännedom om hur arbetet på ett mindre företag som tillverkar och säljer helhetslösningar inom automation ser ut.

Eltech Automation får ett verktyg för att kunna ansluta till sina HiFlex-system utan att behöva vara på plats vilket sparar både tid och pengar. Arbetet är fördelaktigt för omgivande samhället då det förenklar användandet och underhållet av HiFlex-robotar som ersätter människor med monotona uppgifter och lyft som sliter på kroppen.

1.6. Avgränsningar

I examensarbetet ingår inte att utveckla en applikation från grunden. Endast den aktuella driftinformationen ska finnas tillgänglig att se i applikationen eller på webbsidan, och historiken behöver inte lagras.

1.7. Arbetsfördelning

Tabell 1. Arbetsfördelning i procent.

	Alma Flygare	Emil Jakobsson
Studier av produkter	50	50
Programmering av PLC för “dummy”-värden		100
Kommunikation mellan PLC och gateway		100
Implementering av HTTP POST i gateway		100
Implementering av virtuell dator	100	
Databasimplementering	100	
Implementering av Node-RED-flöde	50	50
Stresstest av virtuella datorn		100
Konstruktion av webbsida	100	
Domän och HTTPS	60	40
Remote access		100
Rapportskrivning	50	50
Poster	50	50
Förbereda presentation	50	50
Presentation	50	50

För att effektivisera arbetet delades det upp i implementeringsstadiet. Emil fick mer ansvar för hårdvarans programmering och konfigureringen av den samt dess kommunikation med servern. Alma ansvarade för den virtuella maskinen och dess mjukvara. Uppdelningen gjorde så att tiden någon behövde vänta på att den andra blev färdig blev minimal och bådas tid kunde användas så effektivt som möjligt, se tabell 1.

2. Teknisk bakgrund

I det här kapitlet förklaras begrepp, koncept och tekniker som har använts i arbetet som kräver djupare förståelse.

2.1. Gateway

En gateway är, i nätverksammanhang, en dator eller ett program på en server som sitter mellan nätverk och gör det möjligt för en klient som tidigare inte har kunnat kommunicera med utomstående nätverk möjlighet till det. Den gör det genom att översätta de protokoll som används på det lokala nätverket till att fungera med övriga internet. Gateway:n arbetar på OSI-lager 3, det vill säga nätverkslagret (IT-ord u.å.).

Inom industriell automation är gateway:s vanligt förekommande och används för att ge läs- och skrivrättigheter till fältbussdatapunkter för att möjliggöra operationsprocesser och övervakning. Det krävs då ofta att gateway:n har en viss beräkningsförmåga för att omvandla kommunikationsprotokoll och även data (Sauter, T & Lobashov, M. 2011).

2.2. OPC UA

För att kommunikationen mellan PLC och gateway ska vara möjlig måste gateway:n vara kompatibel med något av de kommunikationsprotokoll som PLC:n använder. Ett vanligt förekommande protokoll, som är en internationell standard, är OPC UA. OPC UA är en modell som bygger på kommunikation mellan noder, en server och klienter. Ett exempel på noder är sensorer. Servern lyssnar sedan efter förändringar i noder. Till servern finns det sedan klienter som prenumererar på olika ämnen. Det kan exempelvis vara att en klient prenumererar på värden från en givare och får då data från servern när det finns (Cavalieri, S., Mulè, S. 2021).

Omrons PLC NX102-1000 kan konfigureras till att agera som server. En annan enhet kan då konfigureras som klient och kan då prenumerera på olika ämnen eller variabler (Omron Industrial Automation EMEA 2020).

2.3. Node-RED

Node-RED är ett open-source, flödesbaserat programmeringsverktyg som kan användas för att automatisera olika processer. Programmering i Node-RED görs genom att olika noder placeras och kopplas samman i ett flöde. Olika noder har olika funktioner och kan konfigureras efter behov. Flödesbaserad programmering

är visuell och lätt att förstå, även för individer som inte har tidigare kunskaper i Node-RED (Node-RED u.å. a).

2.4. Webbsida

För skapandet av en dynamisk webbsida är PHP ett vanligt förekommande programmeringsspråk. PHP är en akronym för PHP: Hypertext Preprocessor. Vid användning av PHP-filer kan samverkan med andra programmeringsspråk så som exempelvis HTML eller JavaScript ske enkelt i samma fil. När PHP-filen exekveras sker det på servern som sedan skickar resultatet till webbläsaren som HTML-kod. PHP har också inbyggda funktioner för att kunna kommunicera med databaser (W3Schools u.å. a).

2.5. HTTP(S)

HTTP, Hypertext Transfer Protocol, är ett protokoll för kommunikation mellan datorer på internet. Protokollet bygger på att det finns en klient och en server. Klienten, ofta en webbläsare, skickar olika förfrågningar till en server. Förfrågan görs till en URL. I HTTP finns flera instruktioner för om data ska hämtas eller skickas. Om en klient vill hämta information från en server kan den använda instruktionen GET och vill den skicka information kan instruktionen POST användas (Nationalencyklopedin u.å. a). HTTP är i sig inte krypterat men HTTPS (HTTP över SSL) är. All kommunikation mellan klient och server är då krypterad. För att HTTPS ska fungera behöver ett SSL-certifikat utfärdas av en Certificate Authority för domänen som servern är kopplad till (Stallings & Brown 2018, s. 697-698).

2.6. Virtuell dator

En virtuell dator eller VM (Virtual Machine) är mjukvara som skapar en virtuell version av en dator med dedikerat minne och CPU. Mjukvaran kan köras på en lokal dator men är ofta tillhandahållen av olika molntjänster. Användare betalar då för att ha tillgång till en virtuell dator i en serverhall som blir tillgänglig över nätet (Microsoft Azure u.å. b).

2.7. VPN

VPN (Virtual Private Network) är en teknik för att ansluta till ett annat nätverk än det man befinner sig på. VPN kan användas för att komma åt resurser på ett privat nätverk från ett annat nätverk. För att det ska fungera behövs en VPN-klient som ansluter till en VPN-server. VPN-servern tilldelar då klienten en IP-adress på det privata nätverket och klienten och programmen på klienten får då åtkomst till det privata nätverket (Nationalencyklopedin u.å. d).

2.8. Reverse Proxy

En reverse proxy är en typ av proxyserver eller mellanserver som exempelvis kan användas för att fördela belastningen mellan servrar. En klient ansluter då till en reverse proxy som sedan skickar klienten vidare till servern som hanterar anropet från klienten. Ett annat användningsområde för en reverse proxy kan vara autentisering (Nationalencyklopedin u.å. c).

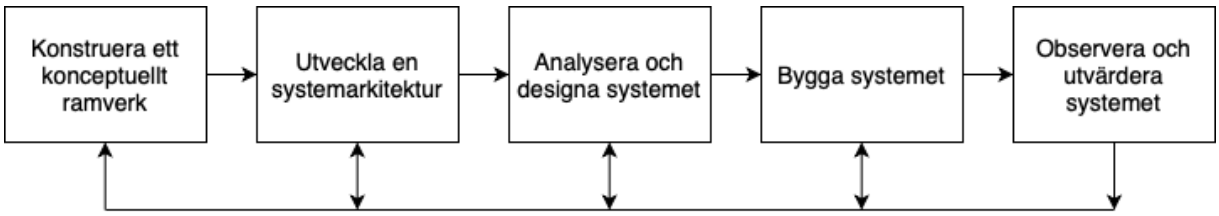
3. Metod

I det här kapitlet förklaras alla steg som har tagits under arbetets gång och hur de gjordes. Kapitlet börjar även med att förklara den valda metodiken och hur den har tillämpats.

3.1. Val av metod att arbeta utifrån

För att kunna utföra examensarbetet och för att det skulle ske med ett vetenskapligt tillvägagångsätt behövdes en lämplig metod att arbeta utifrån väljas. Eftersom arbetet kretsade mycket kring design, utveckling och implementering av hård- och mjukvara behövdes en metod som var applicerbar på dessa områden. Det var även viktigt att i alla skeden av arbetet kunna gå tillbaka till ett tidigare stadie och göra ändringar och utvecklingar. En lämplig metod var därför System development research method (SDRM). Metoden togs fram av Nunamaker, Chen och Purdin (1990) och delar upp undersöknings-processen i fem steg. Stegen har en bestämd ordning och metodens utformning är linjär men det går att gå tillbaka till tidigare steg i processen var än man är. Möjligheten att kunna gå tillbaka till tidigare steg har varit viktig för arbetet och förenklat processen när upptäckter som påverkade tidigare steg gjorts. På grund av vald arbetsmetod ska inte underrubrikerna i metoden tolkas som att de har skett i strikt kronologisk ordning. Många processer har skett parallellt och ändringar i processer har gjorts efter upptäckter i senare steg.

Metoden är uppdelad i fem steg, se figur 1, som förklaras mer ingående nedan.



Figur 1. Flödesdiagram för SDRM

3.1.1. Konstruera ett konceptuellt ramverk

Eftersom många delar av projektet var främmande behövdes det först skaffas en god uppfattning över vad det efterfrågade arbetet handlade om. Här formulerades de övergripande frågeställningarna som behövde besvaras. De övergripande frågeställningarna var följande:

- Vilka funktioner är det som efterfrågas?
- Vad är det för hårdvarulösning som krävs?
- Vad finns det för olika alternativ av gateways på marknaden?
- Vad krävs det för mjukvara?
- Finns det färdiga helhetslösningar som uppfyller alla krav?
- Vad för typ av lösningar kan implementeras och tas fram på egen hand?
- Beroende på olika val, vad behöver implementeras och vad finns det för alternativ där?
- Vad är kostnaden för olika alternativ?

Information samlades ihop från olika tillverkare men även information om hur andra hade löst liknande problem och vad som krävdes.

3.1.2. Utveckla en systemarkitektur

I det här steget påbörjades skissning på hur ett system skulle kunna se ut. Vilka komponenter och funktioner krävs och hur fungerar de tillsammans? Vad behövdes göras på egen hand och vilka färdiga lösningar kunde användas?

3.1.3. Analysera och designa systemet

Här togs det fram ett antal olika förslag på hur problemet kunde lösas. Lösningarna samlades i ett Excel-ark som sedan delades med Eltech Automation som därefter valde ett alternativ.

3.1.4. Bygga systemet

I det här steget samlades mycket förståelse för vad som fungerade och hur det fungerade. Här gjordes all konfigurering och programmering av systemet. Arbetet delades upp i delar som arbetades med parallellt. Delarna testades var för sig för att bekräfta funktionalitet för att sedan kopplas samman med varandra.

3.1.5. Observera och utvärdera systemet

För att kunna bekräfta att systemet fungerade som det skulle behövdes varje del testas och utvärderas. Det blev därför naturligt att en stor del av arbetet pendlade mellan steg fyra och fem.

För att simulera en HiFlex skrevs ett PLC-program för att skicka dummy-värden från PLC. Med hjälp av det kunde funktionen på stegen längre fram utvärderas. I det här steget gjordes även ett stresstest av systemet för att analysera och få en inblick i hur det beter sig i ett scenario med flera gateways.

3.2. Studie av produkter

Arbetet påbörjades med ett möte tillsammans med Eltech Automation för att specificera vad examensarbetet skulle lösa. Därefter började sökandet av produkter, både mjukvara och hårdvara, som tillsammans kunde uppfylla samtliga kriterier och mål. Efter en första orientering på området där dess produkter, kommunikationsprotokoll och övrig terminologi undersöktes, inriktades studierna på ett fåtal specifika lösningar. Produkterna, med utgångspunkt gateway, jämfördes på följande punkter:

- Vilket modem (4G eller 5G) innehar produkten?
- Vilka kommunikationsprotokoll stöds?
- Vad kostar hårdvaran?
- Vilka molntjänster krävs för egen lösning och vad kostar det?
- Vad är den totala kostnaden för egen lösning?
- Vad kostar den färdiga lösningen?
- Vad innebär den färdiga lösningen?
- Är produkten tillgänglig?

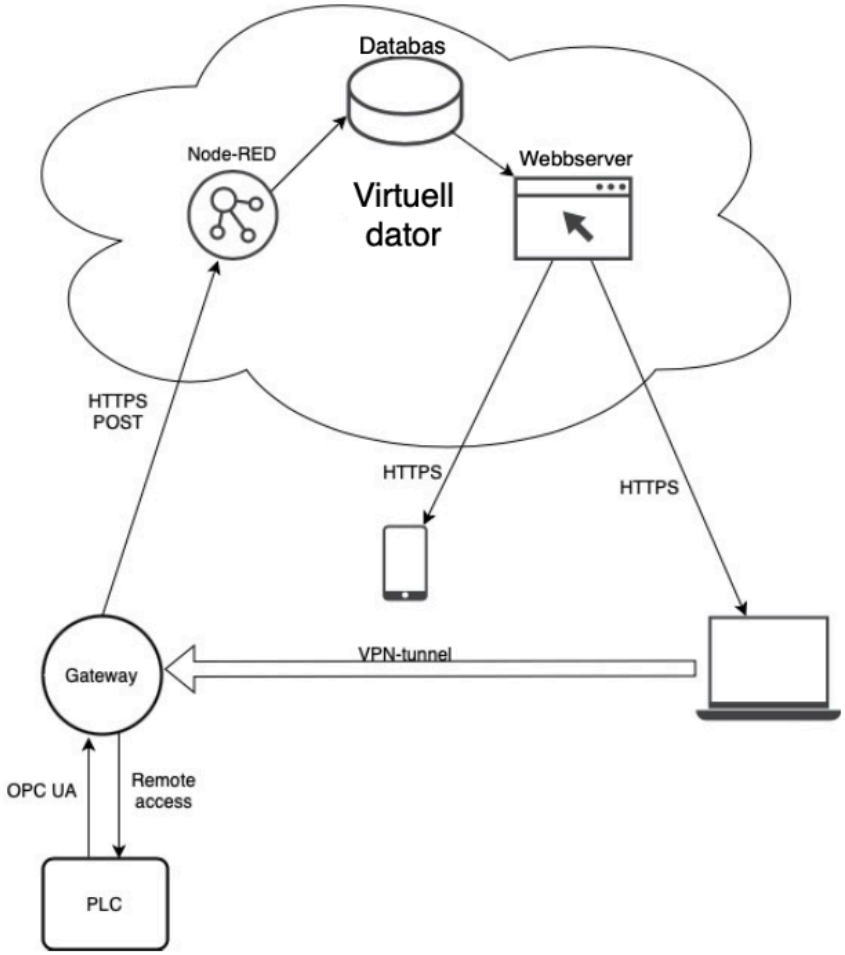
För att minimera faktorerna togs beslutet att utgå från endast en egen lösning på mjukvara som skulle fungera till samtliga gateways. Antalet gateways begränsades till fem stycken för att minska beslutsunderlaget och därmed snabbare kunna fatta beslut. Den egna lösningen var inte helt fastställd i alla steg vid beslut om lösning men de delar som medförde kostnader så som molntjänst och domännamn var bestämda.

All datainsamling gjordes utanför företaget. Detta för att datainsamlingen inte krävde daglig kontakt med företagshandledare och för att det var lättare att få en uppfattning om vad uppgiften innebar och vilka kriterier som skulle has i åtanke om arbetet utfördes på egen hand. När frågor uppkom eller möten skulle planeras in gjordes detta över mail.

Efter insamling av data hölls ett nytt möte med Eltech Automation där beslut fattades om hård- och mjukvara. Beslutet togs av Eltech Automation baserat på de punkter som det tagits fram underlag för samt tidigare erfarenheter av kvalité på hårdvaran.

3.3. Systemarkitektur

Eftersom den valda hårdvaran krävde att mycket av systemet behövde designas och implementeras togs en design fram för att visualisera systemets arkitektur, se figur 2.



Figur 2. Visualisering av systemets design.

3.4. Experimentuppställning

För att kunna programmera PLC och gateway samt kunna få dem att kommunicera behövdes en testuppställning att jobba med. Eltech Automation tillhandahöll uppställning med PLC, gateway, nätaggregat, kopplingsplint och brytare. PLC och gateway var kopplade till varandra med en ethernetsladd. Gateway:n var sedan kopplad med ethernetsladd från WAN-porten till nätverksuttaget.

3.5. Programmering av Omron PLC för "dummy"-värden

PLC:n som var tillgänglig hade inte programmet för HiFlex sparat och eftersom HiFlex-roboten inte var monterad och i funktionsdugligt skick när arbetet påbörjades så behövde PLC:n programmeras så att några dummyvärden ändrades konstant. Programmeringen av PLC gjordes i Sysmac Studio, ett program för att programmera Omron-PLC. En rung med en timer som räknar upp till 10 och sedan nollställdes gjordes. Datatypen som användes då var time. Tre rung:s där outputen, "larm1", "larm2" och "larm3", pendlade mellan noll och ett med olika intervall gjordes. Datatypen var då boolean. Sist gjordes en rung med en räknare, med inputen lampa från tidigare nämnda rung. Räknaren räknade upp till 100 och nollställdes sedan.

3.6. Kommunikation mellan PLC och gateway med OPC UA

För att skicka data mellan PLC:n och gateway:n användes protokollet OPC UA. Protokollet behövdes aktiveras i Sysmac Studio. Här kunde även portnummer konfigureras men det förvalda portnumret 4840 användes. För att variablerna, från programmet som förklarades ovan, skulle kunna skickas med OPC UA behövdes de konfigureras som globala. Variablerna behövdes även ställas in, enligt Omrons instruktioner, som "Publish Only" under "Network Publish" (Omron Industrial Automation EMEA 2020) enligt figur 3.

	Name	Data Type	Initial Value	AT	Retain	Constant	Network Publish
	timespend	TIME			☐	☐	Publish Only
	Start	BOOL			☐	☐	Publish Only
	data	INT			☐	☐	Publish Only
	Q	BOOL			☐	☐	Publish Only
	reset	BOOL			☐	☐	Do not publish
	Larm1	BOOL			☐	☐	Publish Only
	count	DINT			☐	☐	Publish Only
	Larm2	BOOL			☐	☐	Publish Only
	Larm3	BOOL			☐	☐	Publish Only

Figur 3. Globala variabler i Sysmac Studio.

För att PLC:n och gateway:n skulle kunna kommunicera behövde de vara placerade i samma subnät. Subnätsmasken som gateway:n använde var 255.255.255.0. IP-adresser som låg i samma subnät valdes därför för PLC och gateway. För att det skulle gå att kommunicera mellan dator och PLC samt dator och gateway behövde även datorns ethernetport ha en IP-adress som låg i samma subnät. Med IP-adresser konfigurerade och OPC UA-server aktiverad i PLC:n var konfigureringen av PLC:n färdig och nästa steg var att konfigurera gateway:n.

Programmeringen av gateway:n var tillgänglig i webbläsaren genom att skriva in gateway:ns LAN-IP i URL-fönstret. Under fliken “IOServer” och sedan ”OPC UA” kunde server-IP-adressen läggas in. Under “Values” kunde sedan nya “Tags” läggas till. Alla variabler från PLC-programmet tilldelades lämpliga “Tag Name” och “Server Name” valdes till OPC UA. Adressen ställdes sedan in så att den hämtade motsvarande variabel. Variablerna var av olika datatyper för att kunna bekräfta att dataöverföringen fungerade med flera olika datatyper.

3.7. Implementering av HTTP POST i BASIC

För att kunna skicka vidare datan från gateway:n till en Node-RED-nod på den virtuella maskinen implementerades en metod för att kunna skicka data över nätet. Ewon Flexy 205, som var den gateway som valdes för examensarbetet, har en inbyggd BASIC IDE som utnyttjades för detta. Metoden som valdes för att skicka datan var HTTP. Metoder för HTTP-requests fanns redan implementerade i BASIC-IDE:n och implementerades efter Ewons instruktioner (Ewon 2021).

Alla tags som hämtades från PLC:n lästes av och sparades i strängvariabler. De lades sedan samman i en strängvariabel som skrevs i JSON-format. För att kunna

göra en HTTP-request behövdes URL:en till Node-RED-noden, vald HTTP-metod som här var POST, samt HTTP-header som här var “Content-Type=application/json” (Ewon 2021).

Samtlig kod för att skicka datan från PLC:n lades in i metoden “HTTPREQUEST" som i sin tur lades i en funktion, kallad “Postdata”, för att underlätta exekvering. Det var eftersträvansvärt att en HTTP-POST gjordes varje gång ett värde ändrades. Detta kunde lösas genom att varje tag övervakades och vid ändring kallades funktionen “Postdata”. Detta gjordes med hjälp av den färdiga funktionen “ONCHANGE” (Ewon 2021).

3.8. Implementering av virtuell dator

För att implementera den virtuella datorn användes Microsoft Azure där en virtuell dator med operativsystemet Linux Ubuntu 22.04 skapades. Anslutning till den virtuella datorn gjordes med terminalemulatorn PuTTy vilket förenklade anslutningen från en Windows-dator till en Linux-dator genom SSH. Därefter installerades Node-RED genom Node-REDs instruktioner för installation i en Microsoft Azure virtuell maskin (Node-RED u.å. b) men med skillnaden att version 21 istället för version 12 av node.js installerades eftersom det var den senaste versionen vid installationstillfället (Node.js 2023). Node-RED användes för att hantera datan från gateway:n och skicka datan vidare till databasen. I installationsinstruktionerna ingick även att skapa en ny brandväggsregel i Azure-portalen för att tillåta trafik till port 1880, det vill säga den port som Node-RED installerades på (Node-RED u.å. b).

När Node-RED installerats användes HTTP med den virtuella datorns publika IP-adress följt av “:1880” för port 1880 i webbläsaren för åtkomst till verktyget. För ökad säkerhet justerades verktygets inställningsfil där endast en användare och tillhörande hashat lösenord lades till. Detta innebar att åtkomst till Node-RED nu endast gavs efter inloggning med rätt användarnamn och lösenord för att undvika att obehöriga gavs åtkomst och möjlighet att redigera Node-RED- flödet.

3.9. Databasimplementering

Databashanteraren som valdes var MySQL och den installerades på den virtuella datorn via PuTTy. Tillsammans med MySQL installerades även MySQL secure för att öka säkerheten. Med secure-tillägget togs möjligheten till anonym åtkomst till databasen bort tillsammans med möjligheten till att logga in som root-användare från annan IP-adress än den där databasen är installerad. Secure-tillägget tog även bort den förinlagda databasen "test" som är förinställd till att vara åtkomlig till alla användare, både anonyma och icke anonyma (MySQL u.å. a).

Vidare skapades databasen "Hiflex" som skulle hantera all information från Hiflexen och göra informationen tillgänglig för webbsidan. Därefter skapades en användare med full åtkomst till alla databaser i databashanteraren och möjlighet att ansluta från vilken IP-adress som helst. För att utveckla databasen "Hiflex" skapades ett UML-diagram där tabellerna och dess nycklar specificerades. Databasen konstruerades sedan utefter specifikationen i UML-diagrammet vilket innebar skapande av tabellerna "Users", "Access" och "HiflexData".

3.10. Implementering av Node-RED-flöde

Node-RED användes för att ta emot datan från gateway:n och sedan för att skicka datan vidare till databasen. För att kunna göra detta behövdes ett Node-RED-flöde implementeras.

En nod för inkommande HTTP POST-requests lades in i flödet. Här valdes också förlängningen på URL:en, det vill säga namnet på noden. För att inte få felmeddelanden i Flexyns BASIC IDE och för att metoden HTTPREQUEST skulle fungera korrekt krävdes det att Node-RED skickade en HTTP-response och en sådan nod lades därför till och kopplades direkt till den inkommande POST-noden. Till POST-noden kopplades även en debugger-nod för att förenkla felsökandet och kunna se datan som kom från POST-noden. Allt detta gjordes efter Node-RED:s instruktioner för hur en HTTP-POST-flöde för att hantera JSON-filer programmeras (Node-RED u.å. c). Implementeringen av Node-RED skedde parallellt med implementeringen av HTTP-POST-funktionen i gateway:n. För att kontrollera om Node-RED-flödet fungerade användes curl-kommandot i

terminalen på en annan dator för att kontrollera att det gick att göra HTTP-POST till servern och för att se vad svaret var.

Den tredje och sista noden som kopplades till POST-noden var en funktionsnod. I den skrevs kod för att extrahera datan i JSON-objektet. För att hantera datan från funktionsnoden och skicka vidare datan till databasen installerades Node-RED-node-mysql vilket är ett tillägg till Node-RED som gör det möjligt att kommunicera med MySQL-databaser (Node-RED u.å. d). Därefter skapades en MySQL-nod som konfigurerades för att uppdatera tabellerna i MySQL-databasen som installerats och konfigurerats på servern.

3.11. Stresstest av virtuella datorn

Under arbetets gång fanns det endast tillgång till en gateway. För att se hur servern hanterade flera POST-requests samtidigt och direkt efter varandra från flera klienter behövde flera enheter skicka POST-request samtidigt. För att lösa detta skrevs ett shellscript som kunde köras i terminalen. Scriptet valde slumpmässiga tal för varje variabel i JSON-objektet. Detta gjordes för att simulera att värdena ändrades samt att det kom data från olika HiFlex. 100 användare lades in i databasen med HiflexID mellan 1 – 100. Shellscriptet kördes från två datorer samtidigt och det gjordes ett POST-request var hundradels sekund från vardera dator. Inkommande data samt eventuella felmeddelanden kunde ses i Node-RED-konsollen. För att se att datan i databasen uppdaterats som förväntat kontrollerades även MySQL-tabellen “HiflexData”.

3.12. Konstruktion av webbsida

3.12.1. Uppbyggnad av webbsida

Vid skapandet av webbsidan gjordes valet att använda Apache2 som webbserver. Apache2 laddades ner på den virtuella datorn, via PuTTy, enligt Ubuntus instruktioner och användes för att tillhandahålla webbsidan (Ubuntu 2023). I Azure-portalen gjordes en ny brandväggsregel som tillät all TCP-trafik till port 80 vilket är standardporten för HTTP. För att säkerställa att installationen gått rätt till testades åtkomsten till den virtuella datorns publika IP-adress i webbläsaren. I webbläsaren syntes då Apache2s index-fil som meddelade att webbservern fungerade. Därefter skapades en mapp där samtliga filer till webbsidan skulle placeras och mappen i sin tur placerades där Apache2 hämtar filer på den virtuella datorn.

För att skapa en webbsida, som visar den mest aktuella informationen från databasen utan krav på manuell uppdatering av sidan, krävdes utveckling av en dynamisk webbsida vilket gjordes med programmeringsspråket PHP. För att möjliggöra användning av PHP installerades PHP-FPM vilket är ett verktyg för att tolka PHP-koden åt Apache2.

I mappen för webbsidan skapades en PHP-fil, “welcome.php” för huvudsidan som skulle visa samtlig data. Eltech Automations logga sparades även i mappen och PHP-kod för att vissa denna och texten “Welcome” skapades. Därefter installerades PHP-MySQLi genom PuTTy för att möjliggöra anslutning till MySQL-databasen via PHP-filer. Innan MySQLi kunde användas ändrades PHPs initieringsfil där bortkommenteringen av “extension=mysqli” togs bort. Anslutning till databasen “Hiflex” gjordes via "welcome.php” där även kod för att hämta all data från databasens tabell “HiflexData” skrevs. Att anslutningen lyckats och att data gick att hämta bekräftades sedan genom att betrakta webbsidan i webbläsaren.

Vidare implementerades kod för att hämta data från databasen en gång i sekunden. Detta gjordes med AJAX som är ett sätt att använda JavaScript och XML tillsammans för att hämta och visa data från en server (W3Schools u.å. b). För att underlätta användandet av AJAX användes jQuery som är ett JavaScript-bibliotek som innehåller färdiga metoder (W3Schools u.å. c). I den ursprungliga filen “welcome.php” lades kod in som hämtar jQuery och funktioner som efterfrågar data av typen JSON som sedan tolkas till HTML och visas på webbsidan om det fungerar och visar “AJAX error” om det ej fungerar. Kontakten med databasen flyttades över till en ny PHP-fil “fetch_data” som därefter innehöll SQL-kod som hämtade data från databasen Hiflex och skickade tillbaka datan i JSON-format.

3.12.2. Användarautentisering

För hantering av användarautentisering valdes verktyget Auth0 som tillhandahåller tjänster för att skapa, verifiera och styra åtkomst för användare samt lagring av lösenord med mera (Auth0 by Okta u.å. a). Implementering av autentiseringen gjordes genom att skapa ett konto hos Auth0 för att sen skapa av

en "regular web application” i Auth0-portalen. I portalen ställdes applikationens namn in tillsammans med URL för webbsidans huvudsida och applikationens logga. Därefter installerades modulen mod_auth_openidc genom PuTTy och sedan justerades modulens konfigureringsfil till applikationen enligt Auth0s instruktioner för användning med Apache2 (Auth0 by Okta u.å. b).

Vidare implementering av Auth0 gjordes genom konfigurering av Auth0-PHP SDK vilket möjliggör användningen av färdiga metoder från Auth0 i koden för webbsidan. Detta gjordes genom att först, via PuTTy, installera Composer vilket är ett verktyg för att hantera PHP-bibliotek (Composer u.å.). Därefter installerades unzip och mbstring, vilket är två PHP extensions som hanterar zip-filer och multibyte character, tillsammans med curl som möjliggör kommunikation med flera typer av servrar och protokoll. I PHP’s initieringsfil togs bortkommenteringen av “extension=curl” och “extension=mbstring” bort.

För att lagra uppgifterna till Auth0 och koppla dessa till webbsidan skapades en fil “.env” i projektmappen. Sedan, via PuTTy, laddades biblioteket PHP Dotenv ner för att PHP skulle kunna läsa “.env”-filen (Auth0 by Okta u.å. c). Därefter skapades filen “index.php” i projektmappen. I “index.php” lades kod in för att hämta variablerna från “.env”-filen och för att med dessa initiera klassen Auth0. Utefter Auth0s instruktioner lades sedan metoder in för att kontrollera om användaren är inloggad eller ej (Auth0 by Okta u.å. d). Om ej inloggad dirigerade koden användaren till inloggningssidan och om användaren var inloggad visades huvudsidan med texten “Authenticated!”. Att autentiseringen och omdirigeringen fungerade bekräftades genom test av webbsidan.

3.12.3. Webbdesign

När autentiseringen var fullt implementerad och fungerande i filen “index.php” användes denna som ny huvudfil för webbsidans huvudsida. Den tidigare AJAX-koden och JavaScript-koden från “welcome.php” lades nu över i “index.php” och dess funktion bekräftades genom att betrakta webbsidan som inloggad användare.

För att endast visa användare den data de är behöriga till implementerades kod för att hämta mailadress från inloggad användare i “index.php”. Mailadressen

jämfördes sedan i “fetch_data.php” med tabellen “Access” i databasen där inlagda mailadresser kopplats till de hiflexIDn de är behöriga att se. Därefter hämtade koden all data i tabellen “HiflexData” för aktuellt hiflexID och skickade tillbaka till “index.php” i JSON-format.

En logga ut-knapp implementerades för att ge användaren möjlighet att logga ut efter användning. Detta gjordes i HTML genom använde av taggen “<button>” som skapar en klickbar knapp. I Javascript-delen i “index.php”, som uppdaterar sin data en gång per sekund, lades en funktion för logga ut-knappen in. Funktionen satte variabeln logout till sann genom att dirigera användaren till sidan med förlängningen “?logout=true”. Metoden logout i Auth0-klassen genomfördes också vid klickande på logga ut-knappen vilket medför att sessionen bryts och att samtliga data från sessionen raderas samtidigt som användaren skickas till logga in-sidan.

Genom HTML-taggen “<Style>” i filen “index.php” definierades utseendet på webbsidans innehåll. Presentationen av data önskades vara tydlig och lättöverskådlig och placerades därför i en utstickande ruta mitt på webbsidan. Bakgrundsfärgen på sidan gjordes blå för att matcha Eltech Automations logga och rutan gjordes vit för att tydligt sticka ut från bakgrunden och visa datan. Eltech Automations logga placerades tillsammans med logga ut-knappen och texten “Logged in as:” följt av nuvarande användares mailadress i sidhuvudet med svart bakgrund. För att synliggöra aktiva larm skrevs kod för att göra larmtext röd vid aktivt larm och svart vid inaktivt larm. Även robotens status förtydligades men detta med grön text för körande robot och svart text för stillastående.

För att webbsidan skulle passa för både mobilskärmar och datorskärmar skrevs kod som anpassar innehållets storlek efter skärmens storlek. Majoriteten av storlekar programmerades sedan med procent istället för antalet pixlar.

Eftersom webbsidan skulle vara så lättåtkomlig som möjligt önskades åtkomst direkt från IP-adressen istället för att tydliggöra exakt mapp och fil. Detta löstes genom skapandet av fil “index.html” som placerades direkt i mappen där

Apache2 hämtar filer till skillnad från “index.php” som placerats i ytterligare en mapp. I “index.html” lades kod in som dirigerar användaren direkt till “index.php”.

3.13. Domän och HTTPS

Eftersom lösningen ska användas av kunder så behövdes en domän istället för att datan var tillgänglig genom IP-adressen. Vd:n på företaget valde en domän som var lämplig för ändamålet. Med en domän kunde ett SSL-certifikat utfärdas. Detta gjordes med hjälp av certbot. Med ett SSL-certifikat kunde HTTPS användas istället vid anslutning till webbsidan. För att kommunikationen mellan gateway:n och Node-RED också skulle vara över HTTPS behövdes en reverse proxy konfigureras. Reverse proxyn ställdes in så att kommunikation till /node-red skickades vidare till Node-RED. Detta gjordes i filen eltechconnect.com-le-ssl.conf, vilken styr domänens SSL-inställningar, genom att lägga följande rader:

ProxyPass /node-red http://localhost:1880/

ProxyPassReverse /node-red http://localhost:1880/

(Kossi D. T. S. 2019). För att gateway:n sedan skulle kunna använda HTTPS ändrades metoden för HTTP POST till REQUESTHTTPS. URL:en som POST-requesten gjordes till ändrades till https://eltechconnect.com/node-red/node-red.

3.14. Remote access

Gateway:n kom med en färdig lösning för remote access med VPN men den behövde konfigureras för att det skulle gå att komma åt PLC:n från ett annat nätverk. Ett konto på Ewons Ecatcher gjordes och sammankopplades med gateway:n. I inställningar för gateway:n i Ecatcher kunde man sedan ställa in brandväggen och vilka IP-adresser anslutna till gateway:n som det skulle gå att ansluta till. Här lades IP-adressen till PLC:n in. I kommunikations-inställningarna i Sysmac Studio valdes sedan anslutning med ethernet via hub samt PLC:ns IP-adress. För att sedan kunna ansluta till PLC:n i Sysmac Studio från en dator på ett annat nätverk kopplades datorn upp via VPN i Ecatcher.

3.15. Källkritik

Här motiveras valet av källor som har använts och varför de kan antas vara trovärdiga.

Auth0 by Okta

Källan Auth0 by Okta har använts för att hämta information om vad Auth0 är och hur det används. Eftersom informationen gäller de som själva gett ut den och då instruktionerna fungerade anses källan vara trovärdig.

Apache

Apache som källa har använts för att samla generell information om apache. Eftersom Apache är en så välkänd och vanligt förekommande webbserver kan källan antas vara trovärdig.

Composer

Informationen som har hämtats från Composer är information om deras eget verktyg och hur det installeras. Eftersom instruktionen fungerade anses källan vara trovärdig.

Computer Security

Computer Security är en bok inom genren facklitteratur och behandlar ämnet datorsäkerhet. Boken har bland annat använts som kurslitteratur i kursen säkerhet på LTH och kan därför antas vara trovärdig.

Eltech Automation

Eltech Automation är företaget examensarbetet har gjorts hos. Informationen härifrån antas därför vara trovärdig.

***How to Access Factory Floor Information Using Internet Technologies and Gateways* - Artikel**

Publicerades i IEEE Transactions on Industrial Informatics. IEEE är världens största ingenjörsförening med över 375000 medlemmar och är bland annat aktiva inom standardiseringsarbetet (Nationalencyklopedin u.å.). Källan kan därför anses vara trovärdig.

How to configure OPC-UA on SYSMAC NX-102 controllers – **Video**

Klippet är uppladdat av Omrons egna youtube-konto *Omron Industrial Automation EMEA*. Eftersom klippet användes för att konfigurera OPC UA och det fungerade så kan källan antas vara trovärdig.

***How to use Apache to redirect requests for Node-RED?* - Foruminlägg**

Källan var ett foruminlägg på stackoverflow. Eftersom stackoverflow är omodererat ska information hämtas därifrån med försiktighet. Instruktionerna som hämtades härifrån bedömdes dock vara trovärdiga och i och med att de fungerade så var de också det.

***Interoperability between OPC UA and oneM2M* - Artikel**

Publicerades först i *Journal of Internet Services and Applications* men finns numera tillgänglig hos SpringerOpen som är en portfolio med vetenskapliga publikationer. Alla publikationer har genomgått en hög nivå av Peer-to-peer-granskningar (SpringerOpen u.å.). Samtliga källor använda i artikeln är med i referenser och den ena författaren, Salvatore Cavalieri, är professor i datavetenskap vid universitetet i Catania (Orcid u.å.). Källan kan därför antas vara trovärdig.

IT-ord

IT-ord är ett uppslagsverk över IT-termer och begrepp. Sidan drivs av Computer Sweden som ägs av IDG. “Computer Sweden ingår i ett globalt nätverk av redaktionella publikationer inom professionell it” (Computer Sweden u.å.). Computer Sweden är en webbtidning och kan därför tänkas vara partisk och ha

en agenda i vissa sammanhang. Eftersom informationen som hämtades därifrån var ren fakta kan IT-ord dock ändå ses som en pålitlig källa.

Let's Debug
Let's Debug är ett verktyg för att felsöka när Let's Encrypt inte fungerar vilket var vad det användes till här. Informationen som kunde utläsas stämde och källan kan därför antas vara trovärdig.

Microsoft Azure
Microsoft som källa har här bara använts för instruktioner vid initiering och konfigurering av virtuell maskin samt för att se var de har sina serverhallar. Microsoft är ett företag som har ekonomiska intressen och kan därför tänkas ha en agenda i vad de skriver. Eftersom den hämtade information var instruktioner för hur deras tjänst används samt platsen för var de verkar kan dock Microsoft här anses vara en trovärdig källa.

MySQL

Källan MySQL har använts för att hämta information om databashanterarens eget program ”MySQL secure” och information om vad MySQL är. Den hämtade informationen handlar alltså om de som själva gett ut den och kan därför anses trovärdig. MySQL är dessutom en vanligt förekommande databashanterare med en stor mängd användare som förlitar sig på tjänsten.

NE.se
NE är ett “ledande digitalt kunskapsföretag” och Nationalencyklopedin är Sveriges nationella uppslagsverk. De artiklar på ämnen de har är skrivna av ämnesexperter och skribenter. På NE.se går det också att läsa att allting är faktagranskat och objektivt (Nationalencyklopedin u.å. e). NE används av flera skolor och universitet däribland Lunds universitet och kan därför antas vara trovärdig.

Node.js
Node.js har endast använts som källa för information om vilken den senaste utgivna versionen av Node.js var. Eftersom informationen som hämtats på sidan handlar om plattformens egen utgåva anses källa vara trovärdig.

Node-RED
Faktan och instruktionerna hämtade från Node-RED har varit information och instruktioner om programmet i sig. Instruktionerna har varit korrekta och det har fungerat när de har följts vilket ökat trovärdigheten. Programmet är open source och är därmed kostnadsfritt. Det finns därför inga ekonomiska incitament och har därför antagits vara trovärdigt.

Programming reference guide
Programming reference guide är en referensguide för programmering i BASIC i Flexy-IDE:n. Guiden är framtagen av Ewon som har utvecklat gateway:n. De instruktioner som har följts har fungerat och man kan anta att Ewon vill göra sina produkter så användarvänliga som möjligt och källan kan därför antas vara pålitlig.

***Systems Development in Information Systems Research* – Artikel**
Publicerades först i *Journal of Management Information Systems* som är ett forum för forskning inom *"organizational information systems"*. Metoden som beskrivs refereras även till i andra vetenskapliga artiklar och kan därför anses vara trovärdig.

Ubuntu

Ubuntu som källa har använts för att få instruktioner i hur Apache2 installeras på en Ubuntuserver. Samtliga av de hämtade instruktionerna fungerade. Ubuntu som källa har även använts för att hämta information om utgivningsår av operativsystem. Källan kan därför antas vara trovärdig.

W3Schools

W3Schools har i examensarbetet använts för information om olika programmeringsspråk och enklare användning av dessa. Samtlig kod som hämtats från källan har fungerat som beskrivet och därför anses källan trovärdig.

4. Analys

Här förklaras de olika val som gjordes och varför under arbetet. De problem som uppkom under arbetets gång och hur det löstes tas också upp i det här kapitlet.

4.1. Gateway

Vad som avgör om en gateway är kompatibel med en PLC är att enheterna måste använda sig av samma gränssnitt och de måste stödja ett gemensamt kommunikationsprotokoll. HiFlex-systemet använder sig av en PLC från Omron. Gränssnittet för kommunikationen är Ethernet och exempel på dess protokoll är MODBUS och OPC UA.

4.1.1. Kravspecifikation på gateway

För att välja bäst lämpade produkter ställdes följande krav på mjuk- och hårdvara:

- Hårdvaran ska stödja möjlighet för mobil uppkoppling med modem för antingen 4G eller 5G, där 5G är att föredra.
- Produkten ska stödja något av kommunikationsprotokollen som vald PLC använder (Omron eller Siemens) ex. Modbus eller OPC UA, där OPC UA är att föredra.
- Hårdvarans kostnad får inte vara för hög i förhållande till liknande alternativ på marknaden.
- Hårdvaran ska antingen komma med eller ge möjlighet att köpa till mjukvara som går att anpassa efter Eltech Automations behov och/eller vara kompatibel med egen mjukvarulösning.
- Hårdvaran ska finnas tillgänglig för inköp och får inte ha längre leveranstid än två veckor.
- Produkten ska komma från en pålitlig tillverkare.

En pålitlig tillverkare är, i det här sammanhanget, en tillverkare som är väletablerad och välkänd inom branschen och som det går att hitta mycket information om vid sökningar. Det ska också vara en tillverkare som Eltech Automation inte har dåliga erfarenheter av sedan tidigare. Det är också fördelaktigt om det finns en stor grupp användare av produkten eftersom det kan vara ett tecken på att det är en omtyckt produkt och att det finns många med mycket kunskap man kan vända sig till i exempel forum om hjälp skulle behövas.

4.1.2. Val av gateway

Valet gjordes av Eltech Automation och baserades på ovan ställda krav tillsammans med företagets samlade erfarenhet av produkterna. De två Secomea-alternativen ansågs för dyra och Teltonika-alternativen valdes bort på grund av tidigare erfarenheter från fallerade Teltonika-gateways. Eltech Automation ansåg att Ewon Flexy 205 hade rimligt pris och omfattande anpassningsmöjligheter samt kom från en pålitlig tillverkare med en stor användargrupp. Ewon Flexy 205 hade även ett forum där tidigare användare ställt frågor men också möjlighet för oss att kunna ställa frågor om det skulle behövas. En annan fördel var att Eltech Automation redan hade den valda gateway:n hemma och hade experimenterat med den till viss del tidigare. Det behövdes därför inte beställas något och arbetet kunde påbörjas omgående.

Den valda gateway:n är också modulär och tillbehör för att utöka dess funktionalitet kan köpas till. Ett exempel på det är ett 4G-modem som kan köpas till om mobil uppkoppling efterfrågas.

4.2. Kommunikation mellan PLC och gateway

Flera kommunikationsprotokoll för kommunikationen mellan PLC och gateway fanns att välja på. Några exempel på protokoll är OPC UA, MODBUS och FINS. Det var efterfrågat att lösningen skulle vara lätt att anpassa till att fungera med en PLC av andra tillverkare. FINS är ett protokoll som bara används av Omron och var därför inte lämpligt. MODBUS används av alla stora PLC-tillverkare men är en förhållandevis gammal standard som inte är särskilt lätthanterlig om man jämför med OPC UA som blir mer och mer standardiserat och som är förhållandevis lätthanterlig.

4.3. Mjukvarulösning

Studier av de olika lösningarna på mjukvara, antingen inkluderad i hårvarulösningar eller separata, visade att det inte fanns något på marknaden som tillgodosedde de krav Eltech Automation hade satt. Lösningen var då att ta fram en egen mjukvarulösning som kunde skräddarsys efter Eltech Automations behov. Från början var också tanken att mjukvaran skulle vara i form av en mobilapplikation. Eftersom mjukvaran skulle utvecklas från grunden istället för att köpas in valdes det istället att göras en webbsida. Beslutet togs då det inte

fanns någon efterfrågad funktion som en mobilapplikation erbjuder som en mobilanpassad webbsida inte gör. Lösningen med en webbsida kombinerades med en server som kör mjukvara för att organisera datan som skickas från gateway:n.

4.3.1. Node-RED

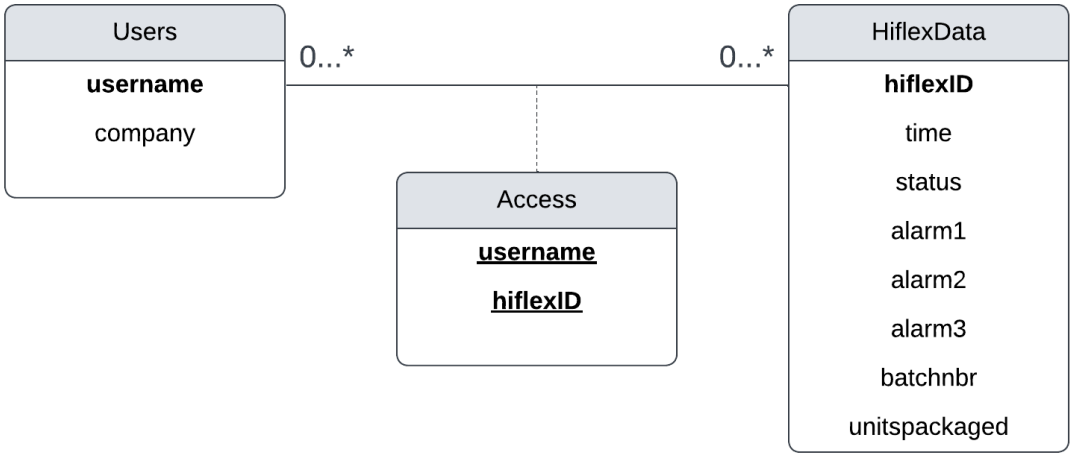
För den egna lösningen gjordes valet att använda Node-RED, som var ett förslag från handledaren på LTH, i en molntjänst för att lägga in och tillfälligt lagra den insamlade datan i en databas. Inom molntjänsten skulle även en webbsida finnas som visar efterfrågad information från databasen. Node-RED valdes eftersom det ansågs vara ett enkelt verktyg för att hantera den inkommande datan och det gjorde det lätt att komma igång och få en fungerande lösning. Dessutom är Node-RED kostnadsfritt och kräver relativt lite förkunskaper för att användas.

4.3.2. Databas

Vid val av databashanterare övervägdes typen, hur vanligt förekommande systemet är, kostnad och funktioner. Tidigare erfarenheter innefattade endast relationsdatabaser och det ansågs därför bäst lämpat. Vidare konstaterades att databashanteraren inte behöver ge några speciella funktioner och för att ha så låg kostnad som möjligt gjordes valet att använda en open-source-mjukvara. Valet föll på MySQL eftersom det är en vanligt förekommande databashanterare (MySQL u.å. b) som är open-source och därmed är gratis. Eftersom MySQL har en stor mängd användare finns också en stor mängd dokumentation om olika problem och dess lösningar vilket underlättar om liknande problem skulle uppstå under examensarbetet.

Uppbyggnaden av databasen gjordes genom att först skapa ett UML-diagram för att få en tydlig översikt, se figur 3. Databasen behövde inte innehålla särskilt mycket data och fokus var att på enklast möjliga vis sortera datan som skulle visas på webbsidan och koppla datan till behöriga användare. För att åstadkomma detta användes tabellerna “Users”, “HiflexData” och “Access”. I “Users” skrevs attributet användarnamn vilket är mailadressen som användaren registrerat på webbsidan. Användarnamnet används som primary key för att unikt identifiera varje användare. För att underlätta för Eltech Automation och hanteringen av användare lades även attributet företag in vilket kan användas för att ha bättre

koll på vilka användare som tillhör vilka företag. I “HiflexData” används ett unikt id, kallat hiflexID, som primary key. HiflexID är av datatypen varchar(75) för att ge Eltech Automation möjlighet att utforma HiflexID efter egna preferenser. Med datatypen varchar(75) kan hiflexID skrivas antingen som nummer eller som mer förklarande text, exempelvis företagsnamn för kund som använder den specifika HiFlexen. I “HiflexData” läggs också data som tid för när datan uppdaterades i databasen, status, batchnummer, packade enheter och status för tre olika larm in. För att sammankoppla de två tabellerna “Users” och “HiflexData” används tabellen “Access” som har användarnamn och hiflexID som både primary key och foreign key för att unikt verifiera varje behörighet med både användarnamn och hiflexID , se figur 4.



Figur 4. UML-diagram för hur databasen är uppbyggd. Primary keys är markerade med fetstil och foreign key med understrykning.

4.3.3. Virtuell dator

Node-RED och MySQL-servern behövde installeras på någon typ av dator. För att minimera behovet av underhåll valdes det att köra allt på en virtuell dator. Eftersom molntjänsten ska innehålla flertalet tjänster som ska kommunicera med varandra underlättar det att välja en virtuell maskin istället för specifika molntjänster för varje tjänst. Kriterierna för val av virtuell datorn var kostnad, pålitlighet samt plats för serverhallen. Valet landade på Microsoft Azure. Kostnaden för en virtuell dator från Azure var i samma nivå som konkurrenter. Fördelen för Azure är att de har flera serverhallar i Sverige varav den närmsta ligger i Staffanstorp (Microsoft Azure u.å. a). Azure ägs av Microsoft som är ett välkänt it-bolag och deras pålitlighet ansågs därför vara hög. Vid konfigurering

av en virtuell dator väljs datorns hårdvaruspecifikationer. Här valdes en av de enklare varianterna, med två kärnor och 4GB RAM, för att applikationen inte troddes behöva särskilt krävande hårdvara. Fördelen med en virtuell maskin är också att om en kraftfullare dator skulle behövas är det bara en fråga om att ändra i inställningarna för hanteraren av den virtuella datorn.

4.3.4. Operativsystem

Vid val av operativsystem valdes Ubuntu, som är baserat på Linuxkärnan. Fördelen är att det är ett vanligt förekommande operativsystem för servrar och det finns många resurser på internet samt mycket mjukvara som är utvecklat för det. En annan fördel är att kostnaden för Ubuntu är betydligt lägre jämfört med Windows. Det finns även gratis versioner av Ubuntu men för att få underhålls- och säkerhetsuppdateringar i tio år istället för fem år valdes PRO-versionen som fortfarande har en avsevärt lägre kostnad jämfört med Windows. När installering av operativsystemet gjordes var den senaste versionen av Ubuntu 22.04 och därför valdes den (Ubuntu u.å.).

4.3.5. Webbsida

Webbsidan valdes att göras dynamiskt för att visa den mest aktuella informationen utan behov av manuell uppdatering av sidan. Valet att använda PHP gjordes eftersom det är ett vanligt förekommande programmeringsspråk för webbsidor med dynamiskt innehåll. AJAX valdes eftersom även det är vanligt förekommande för dynamiska webbsidor och för att det med relativt lite kod möjliggjorde automatiskt uppdatering av webbsidans data.

Apache2 valdes som HTTP-server för att det är en välkänd webbserver som har funnits sedan 1995. Webbservern har en stor användargrupp och det finns därför många resurser och stöd för den på internet vilket underlättat när egna problem uppstod under arbetet. En annan fördel är att Apache2 är gratis (Apache u.å.).

4.3.6. Lösenordshantering och autentisering

På grund av att lösenordshantering är känsligt och kräver att man tar hänsyn till många säkerhetsaspekter var detta något som var lämpligast att använda ett tredjepartsprogram till. Det finns flera autentiseringsprogram som fyller i princip

samma funktion och det som valdes att användas var Auth0. Detta var för att det är gratis att använda upp till 7500 aktiva användare (Auth0 by Okta u.å. e), vilket är långt fler än vad arbetet efterfrågade. Det finns också en stor mängd användare av tjänsten samt ett internetforum där användare ställt frågor och diskuterat problem och lösningar vilket var fördelaktigt.

4.4. Problem och lösningar

Under arbetets gång stöttes det på diverse olika problem som behövde lösas. Här tas de problemen upp, varför de uppkom samt hur de löstes.

4.4.1. Problem med IP-adress på grund av DHCP

Ett problem som uppstod en dag när det var många på Eltech Automations kontor var att gateway:n inte kunde ansluta sig mot internet trots att WAN-porten var inkopplad. Detta var för att kontorets DHCP hade tilldelat gateway:n en WAN-IP som var för lik LAN-IP. Det räckte inte med att sista oktetten var annorlunda. För att detta inte längre skulle vara ett problem så ändrades gateway:ns LAN-IP och PLC:ns IP till adresser som med mycket låg sannolikhet kommer att krocka med de som DHCP:n delar ut.

4.4.2. SSL-certifikat och HTTPS

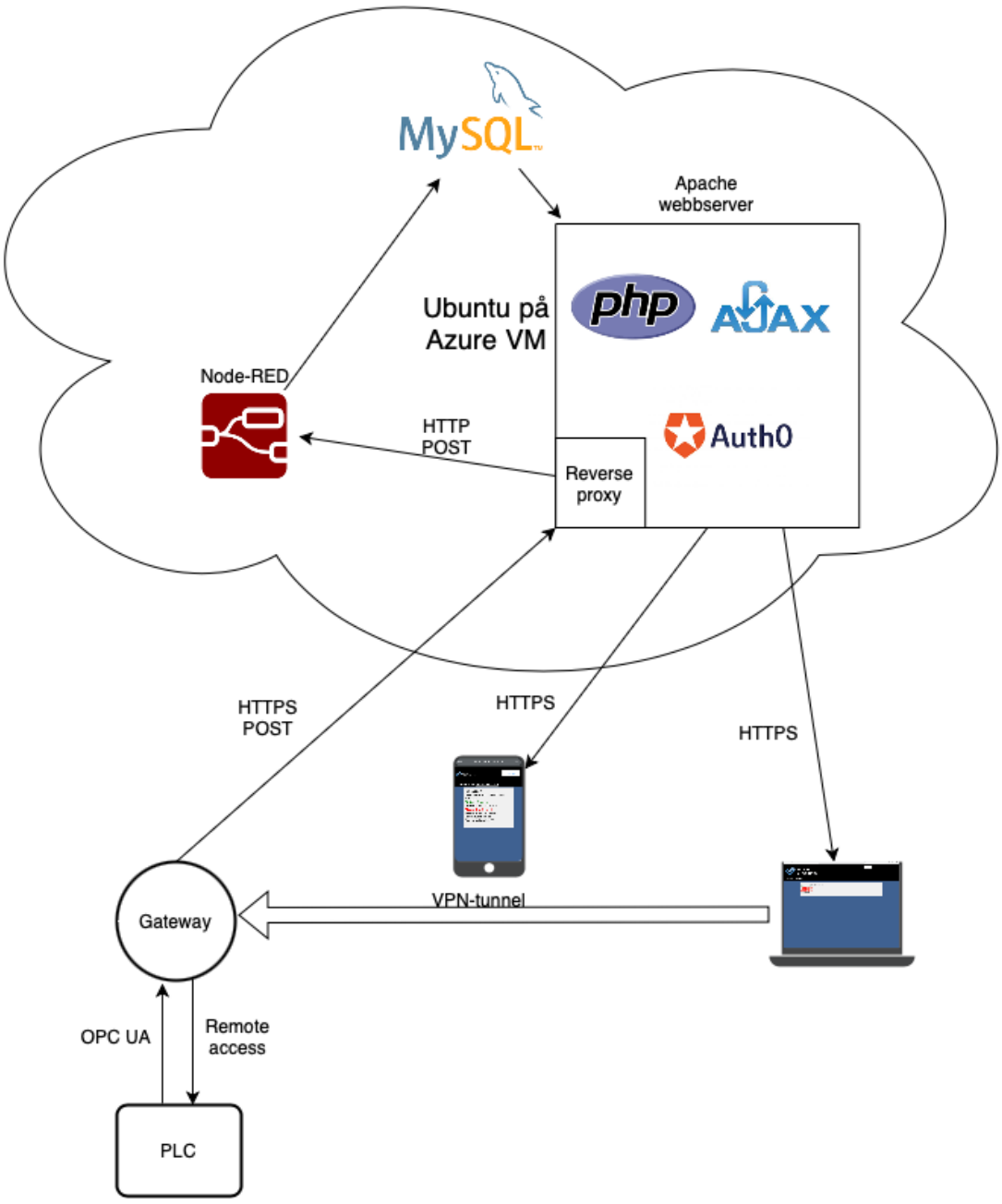
Vid hämtning av SSL-certifikat användes programmet certbot tillsammans med tjänsten Let's Encrypt som tillhandahåller gratis SSL-certifikat. Hämtningen av certifikatet försökte göras direkt i Apache2. Flera försök gjordes sedan med certbot för att hämta certifikatet men alla misslyckades. För att felsöka användes tjänsten Let's debug som är ett verktyg vid felsökning när det inte går att utfärda ett certifikat med Let's Encrypt (Let's debug u.å.). URL:en skrevs in i Let's debug och det resulterade i felmeddelandet "MultipleIPAddressDiscrepancy". Felmeddelandet berodde på att eltechconnect.com hade flera IP-adresser i sitt DNS-register. I DNS-registret fanns det IPv6-adresser kopplade till domänen men servern hade ingen IPv6-adress konfigurerad. För att åtgärda felet behövde man gå in i DNS-tjänsten och ta bort alla AAAA register för IPv6. När detta var åtgärdat gick det att utfärda ett SSL-certifikat.

4.4.3. Reverse proxy

Datan som skickades mellan gateway:n och Node-RED behövde också konfigureras så att det skedde över HTTPS. Först testades att lägga in certifikatet och nyckeln i Node-RED:s inställningsfil. Detta fungerade aldrig och felmeddelandet som uppkom var att Node-RED inte hade behörighet till certifikat- och nyckel-filerna. Behörigheterna försökte ändras med chmod-kommandot men utan framsteg. Lösningen på problemet var att använda Apache2 som en reverse proxy för Node-RED. Istället för att HTTPS-POST från gateway:n gjordes direkt till Node-RED så gjordes de till eltechconnect.com/node-red. Apache2 var sedan inställd till att skicka dessa paket vidare till Node-RED. Eftersom datan som skickades mellan gateway och Apache2 var över HTTPS och Apache2 sedan skickade HTTP-requesten vidare lokalt så var dataöverföringen nu säker.

5. Resultat

I detta kapitel presenteras resultaten av undersökningarna och analyserna som genomfördes under olika stadier av arbetet, inklusive studien av produkter, produktutvecklingsprocessen och testningen av den slutliga produkten. Den slutgiltiga systemarkitekturen visas i figur 5 och förklaras mer i kapitlet nedan.



Figur 5. Visualisering av hur systemet är uppbyggt.

5.1. Lösningsunderlag, studie av produkter

Efter studier av olika produkter och lösningar på marknaden togs ett underlag fram. I underlaget, se tabell 2, jämförs fem olika alternativ och vad de kostar. Här jämförs också vad de olika tillverkarnas färdiga lösningar kostar och vad en egenutvecklad lösning kostar. Kostnaderna är ungefärliga och användes för att få en uppfattning och på så sätt kunna jämföra olika alternativ.

Tabell 2. Lösningsunderlag som skickades till Eltech Automation**.**

Vad?	**Ewon Flexy 205**	**Secomea Sitemanager 1539**	**Secomea Sitemanager 3539**	**Teltonika TRB140**	**Teltonika TRB500**
Modem (5G/4G)	4G	4G	4G	4G	5G
Kommunikation	OPC UA, Modbus, MQTT, SNMP, HTTPs	Modbus, OPC UA		OPC UA, Modbus, MQTT, HTTPS	
Inköpskostnad hårdvara	8995 kr + 3195 kr (4G extension card) = 12190 kr (Dustin.se)	7 900 kr	10 000 kr	1 249 kr	3 595 kr
Vår lösning kräver följande molntjänster och det kostar ungefär[1]	VM hos Azure med 32 Gb HDD 400kr/mån, SSL certifikat 500/år, domännamn 195/år Total ca månadskostnad: 460 kr				
Kostnad vår lösning Engångskostnad: Månadskostnad:	12190 kr 460 kr	7900 kr 460 kr	10000 kr 460 kr	1249 kr 460 kr	3595 kr 460 kr
Kostnad färdig lösning	Engångskostnad: 12190 kr (gateway) Årligen: 10017 kr (Talk2M Pro)"	Engångskostnad: 7900 kr (gateway) Årligen: från 8478 upp till 27691 (GateManager Professional) + 10800 (Data Collection Cloud, visualisering) = 19278 kr till 38491 kr totalt per år (TR Electronic Nordic AB)"	Engångskostnad: 10000 kr (gateway) Årligen: från 8478 upp till 27691 (GateManager Professional) + 10800 (Data Collection Cloud, visualisering) = 19278 kr till 38491 kr totalt per år (TR Electronic Nordic AB)"	30 kr/månad och enhet (instantbuyrms.com) 30 kr/månad och enhet (instantbuyrms.com)	

[1]De exakta kraven för maskinen för den egna lösningen specificerades inte, men den föreslagna lösningen bör vara lämplig i de flesta fall.

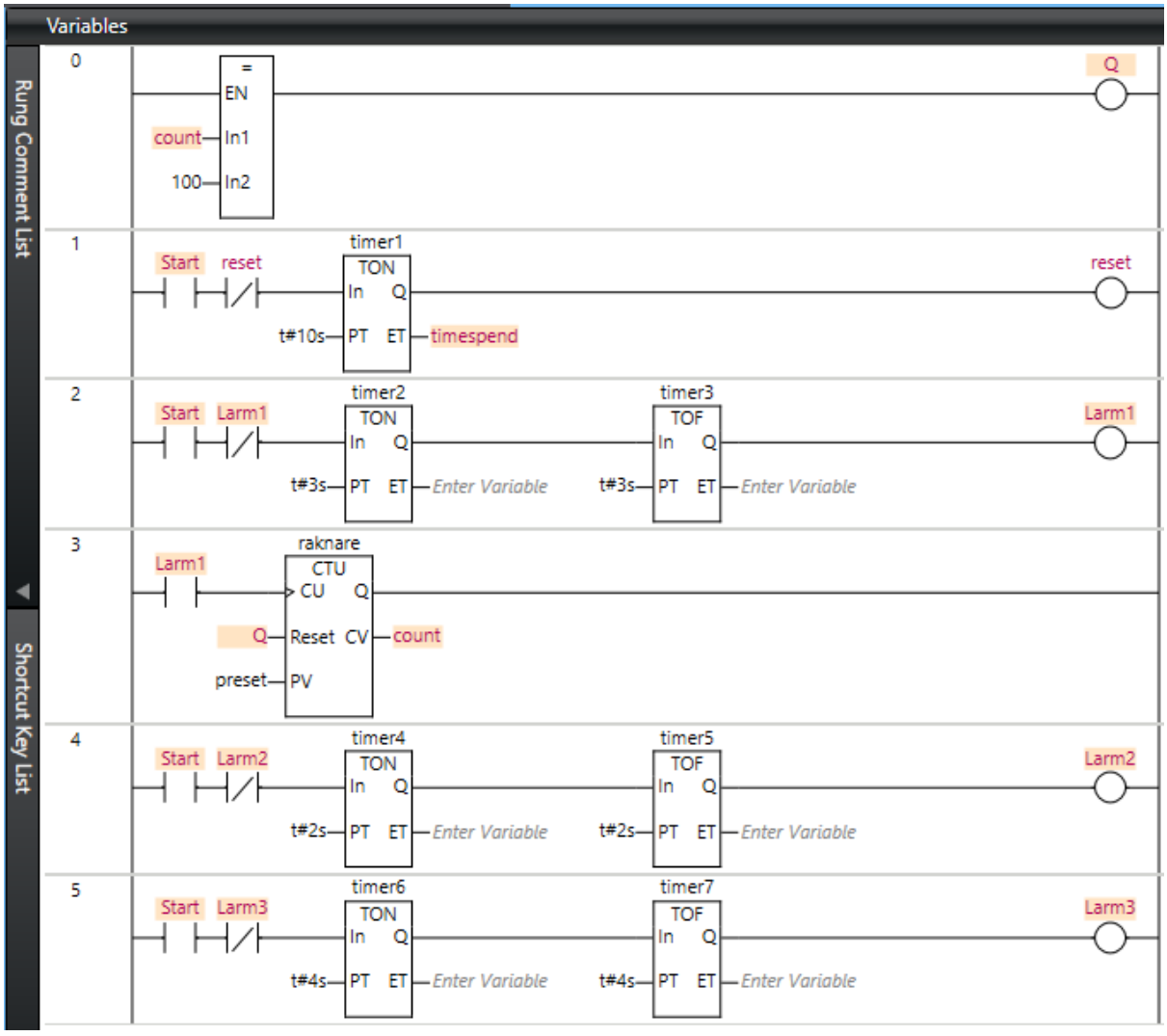
Färdig lösning innebär	Remote Access + Cloudtjänst med datainsamling visualisering		Remote access + AWS Cloud-tjänst, Inte så anpassningsbart interface, inte särskilt kundanpassat
Tillgänglighet	1 st gateway + 2 st modemkort i lager på Dustin	I lager hos TR Electronic Nordic AB	I lager på Dustin

5.2. Gateway

Eltech Automation valde att använda Flexy 205 som är en gateway från tillverkaren Ewon. Gateway:n erbjuder en färdig lösning för remote access, har inbyggt stöd för OPC UA samt en BASIC IDE där egna program kan skrivas för exempelvis HTTPS-kommunikation.

5.3. Ladder-program

Programmering av PLC gjordes i Sysmac Studio enligt beskrivningen i metoden, se figur 6. Programmet kunde sedan överföras till PLC:n och i Sysmac Studio kunde “Start”-variabeln manuellt sättas till hög eller låg för att starta och stoppa programmet.



Figur 6. Ladder-program för att generera ”dummy”-värden.

Variablerna som skulle övervakas i gateway:n sattes som globala samt “publish only”. De skickades sedan med OCP UA till gateway:n. Värdena övervakades direkt i gateway:ns gränssnitt. Programmet användes sedan för att verifiera funktionen av IoT-lösningen genom hela examensarbetets gång.

Gateway:n konfigurerades till att vara en OPC UA-klient ansluten till PLC:n. I gateway:ns BASIC IDE exekverades kod för att posta datan. Variablerna gjordes om till strängar som sedan lades samman i en sträng skriven i JSON-format. Strängen var för lång att ha i en variabel och fick därför delas upp i två och sedan läggas ihop. Strängarna skrevs enligt följande:

```
json1$ = '{"Start": "' + startstr$ + '",' + '"Batchnbr": "' + batchnbr$
+ '",' + '"Unitspackaged": "' + countstr$ + '",' + '"Alarm_1": "' +
larm1str$ + '",'

json2$ ='"Alarm_2": "' + larm2str$ + '",' + '"Alarm_3": "' + larm3str$
+ '",' + '"HiFlexId": "' + HiFlexId$ + '"}'
```

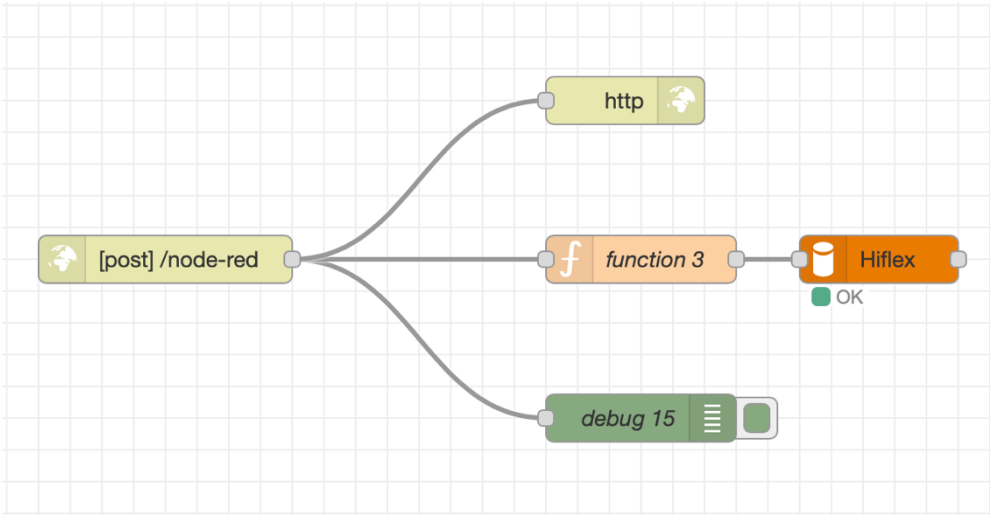
För att göra ett HTTP-request användes följande kod:

```
MyHTTPURL$ = "https://eltechconnect.com:1880/node-red/node-red"
MyHTTPMethod$ = "POST"
MyHTTPHeaders$ = "Content-Type=application/json"
MyHTTPPostData$ = jsontot$
MyHTTPFileData$ = ""
MyHTTPFileAnswer$ = ""
MyHTTPProxy$ = ""
//Gör HTTP-request
REQUESTHTTPX MyHTTPURL$, MyHTTPMethod$, MyHTTPHeaders$,
MyHTTPPostData$, MyHTTPFileData$, MyHTTPFileAnswer$, MyHTTPProxy$
```

Koden kunde sedan exekveras och inga felmeddelanden uppkom i konsolen.

5.4. Node-RED

Node-RED-flödet programmerades. POST-noden med URL-förlängningen "/node-red" tog sedan emot POST-requesten från gateway:n, se figur 7. JSON-objektet som mottogs kunde sedan utläsas i debugger-konsollen, se figur 8.



Figur 7. Node-RED-flöde för mottagning av HTTP-POST.

Figur 8. JSON-objekt i debugger-konsoll i Node-RED.

Till POST-noden kopplades även en HTTP-response-nod som skickade en HTTP-response till gateway:n.

För att extrahera datan i JSON-objektet skrevs kod i funktionsnoden enligt följande:

```
var status = msg.payload.Start;
var hiflexID = msg.payload.HiFlexId;
```

Till funktionsnoden kopplades sedan databasnoden och i den skrevs uppgifterna för anslutning till databasen. När en anslutning till databasen upprättats blev en ruta under noden grön med texten “OK” bredvid, se figur 7.

5.5. Stresstest av virtuell dator och Node-RED

Stresstestet av Node-RED och databas på den virtuella datorn gjordes, som beskrivet i metoden, med hjälp av curl, i ett shell-script, för att göra HTTP-POST och användes enligt följande exempel:

```
curl -X POST \       -H "Content-Type: application/json" \       -d "$json_data" \
http://74.241.129.138:1880/node-red
```

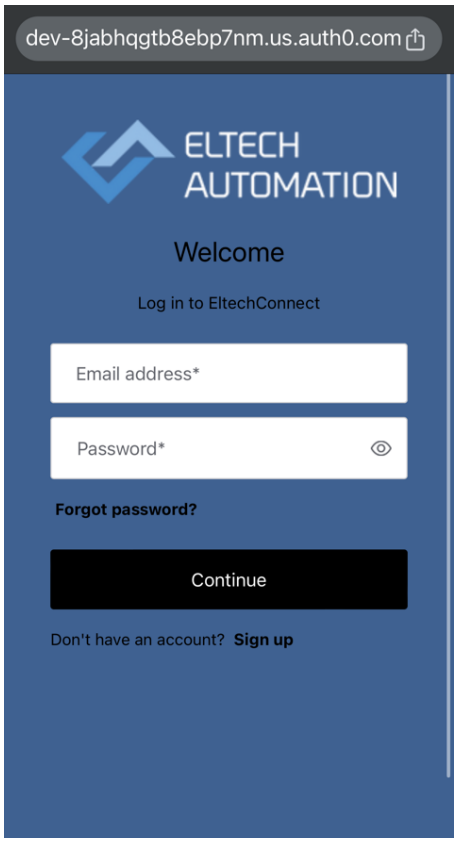
Jämförelser med vad som skickades, vad som kunde övervakas i Node-RED-debuggern samt kontroller av MySQL-databasen visade att det inte fanns några problem att hantera alla HTTP-POST trots högre belastning på systemet.

5.6. Webbsida

Resultatet blev en webbsida med en inloggningssida och en huvudsida enligt nedan.

5.6.1. Autentisering

Hanteringen av autentiseringen resulterade i en webbsida som använder sig av Auth0. Med Auth0 skapades en anpassad inloggningssida med Eltech Automations logga och möjlighet att logga in, skapa ny användare eller återställa lösenord, se figur 9.

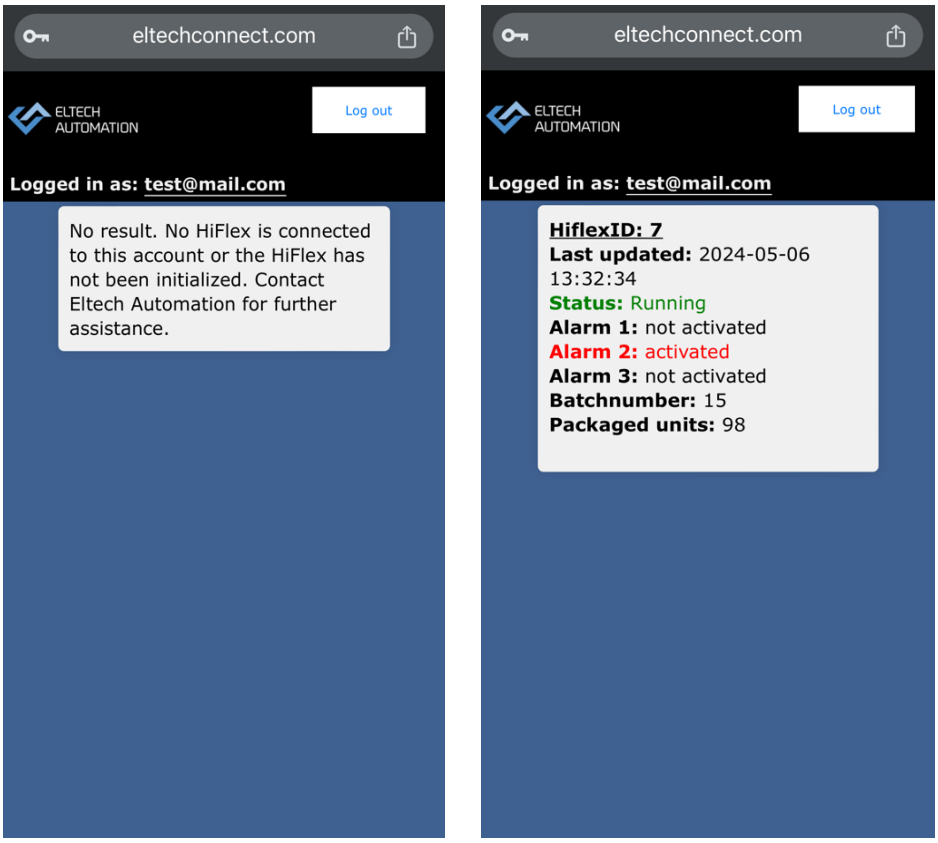


Figur 9. Bild på webbsidans inloggningssida.

5.6.2. Huvudsida

Utvecklingen av en sida för att visa data från HiFlex resulterade i en webbsida som, efter inloggning, visar den data som användaren är behörig till. Webbsidan kontrollerar vilken mail som loggats in med och jämför den med mailadresser i databasens tabell “Access”. Om mailadressen finns i “Access” hämtas de HiflexID:n som mailadressen är kopplad till för att sedan hämta datan för de HiflexID:n i tabellen “HiflexData”. I fall att mailadressen ej finns i tabellen “Access” eller om det kopplade HiflexID:t inte finns i tabellen “HiflexData” så visas en text som förklarar att användaren behöver kontakta Eltech Automation för hjälp, se figur 10. Om mailadressen och det kopplade HiflexID:t finns i databasen så visas datan på webbsidan. Vid körande Hiflex visas texten för status i grönt och vid icke körande Hiflex visas texten för status i svart. Aktiva larm visas med röd text och icke aktiva larm visas i svart text, se figur 11.

Den inloggade användarens mailadress visas i sidhuvudet tillsammans med Eltech Automations logga och en knapp för att logga ut.



Figur 10 (vänster). Inloggad användare med mailadress som inte är kopplad till HiflexID.

Figur 11 (höger). Inloggad användare med mailadress som är kopplad till HiflexID.

5.7. Domän och HTTPS

Den resulterande webbsidan använder sig av domännamnet eltechconnect.com vilket administreras av webbhotellet Websupport och är kopplat till den virtuella datorns offentliga IPv4-adress. Webbsidan använder HTTPS vilket möjliggjorts genom SSL-certifikat utfärdat av Let's Encrypt. SSL-certifikaten är giltiga i 90 dagar och förnyas automatiskt när utgångsdatumet är inom 30 dagar.

6. Slutsats

I det här avsnittet knyts arbetet ihop och slutsatser dras. Här besvaras frågorna som ställdes i problemformuleringen. Här reflekteras även kring etiska aspekter och sist tas förbättringsmöjligheter upp.

6.1. Svar på problemformulering

Här besvaras de frågor som ställdes i problemformuleringen i inledningen. Varje fråga är här en egen underrubrik.

6.1.1. Hur ansluter Eltech Automations ingenjörer sig till HiFlex idag?

Idag ansluter sig Eltech Automations ingenjörer till HiFlexen genom att koppla upp sig till PLC:n med en ethernetsladd. Detta innebär att utan en lösning för remote access så behöver de resa till kunden varje gång en ändring i programmet behöver göras eller vid enklare felsökning.

6.1.2. Vilken typ av IoT-lösning är kompatibel med HiFlex?

De IoT-lösningar som är kompatibla med HiFlex är i det här fallet de IoT-lösningar som är kompatibla med en Omron PLC. IoT-lösningen måste stödja något av de kommunikationsprotokoll som PLC:n använder. I det här arbetet användes OPC UA vid kommunikation mellan PLC och gateway.

6.1.3. Vilka kriterier ska användas för att välja IoT-lösning?

De kriterier som har använts vid val av IoT-lösning har varit:

- Möjlighet till anslutning till mobilt nätverk med modem som är inbyggt i gateway:n eller som finns som tillval.
- Gateway:n ska stödja ett kommunikationsprotokoll som PLC:n använder.
- Kostnaden för hårdvaran ska vara så låg som möjligt. Vid val mellan två likvärdiga alternativ är det kostnaden som avgör valet.
- Kostnaden för mjukvara och digitala tjänster ska vara så låg som möjligt och samtidigt erbjuda de funktioner som har efterfrågats.

- Gateway:n ska kunna konfigureras för att kunna anslutas till via VPN för remote access.
- Hårdvaran ska ha en så leveranstid som är mindre än två veckor.
- Hårdvaran ska komma från en pålitlig tillverkare.

6.1.4. Vilken IoT-lösning ska väljas?

IoT-lösningen som valdes var en kombination av en färdig produkt tillsammans med en egen lösning för hur datan skulle vara tillgänglig och presenteras.

Gateway:n Flexy 205 valdes för att den hade många anpassningsmöjligheter med en inbyggd programmeringsmiljö och stöd för flera kommunikationsprotokoll som OPC UA och HTTPS. Gateway:n har även en färdig lösning för remote access med mjukvara för att med VPN ansluta till ett annat nätverk. Gateway:n kombinerades med en virtuell maskin hos Microsoft Azure. På den virtuella maskinen kördes en Node-RED-server, en MySQL-databas samt en Apache2 webbserver.

6.1.5. Hur ska IoT-lösningen utvärderas?

Lösningen utvärderades på ett antal sätt. Stresstest gjordes för att verifiera att servern klarade av flera HTTP-requests direkt efter varandra. Det testade både Node-RED och databasen samt den virtuella maskinen i sin helhet och verifierade att den valda virtuella datorn var tillräckligt kraftfull för uppgiften.

För att utvärdera hur datan skulle presenteras på webbsidan så presenterades olika revisioner för anställda på Eltech Automation och de kom med förslag på ändringar och förbättringar.

För att verifiera remote access och utvärdera dess funktion anslöts datorn, med Sysmac Studio på, till ett annat nätverk. En VPN-tunnel upprättades sedan och anslutning till PLC:n gjordes i Sysmac Studio. Det gick då att programmera PLC:n genom VPN-tunneln.

6.1.6. Hur ska ett testprogram som ska verifiera total funktionalitet utformas?

Tanken var från början att HiFlexen skulle programmeras och köras för att se att värdena på webbsidan uppdaterades. Eftersom HiFlexen inte var monterad under perioden som examensarbetet utfördes på Eltech Automation så fick funktionen istället verifieras med PLC-programmet med "dummy"-värden. Eftersom det är en enkel process att välja vilka värden som ska skickas till gateway:n från PLC och sedan vidare till webbsidan kommer det att vara förhållandevis enkelt för Eltech Automation att implementera IoT-lösningen i deras HiFlex.

6.2. Reflektion av etiska aspekter

I det här delkapitlet reflekteras över två etiska aspekter kopplat till arbetet. Först reflekteras över konfidentiell information och därefter reflekteras över en punkt från Sveriges Ingenjörers hederskodex.

Konfidentiell information

IoT är en stor del av den fjärde industriella revolutionen men det är inte helt oproblematiskt. I och med att en stor mängd information, ofta konfidentiell, kan skickas över nätet är det ytterst viktigt att den informationen kan skickas på ett säkert sätt och att den inte har blivit tillgänglig för obehöriga under sändningsprocessen. Det är även viktigt att datan som lagras är skyddad och att obehöriga inte har tillgång till den. För att, i det här arbetet, minska risken att information hamnar på fel ställe har det setts till att all data som skickas över nätet är krypterad genom användning av HTTPS.

För vidare skydd av konfidentiell information är systemets gateway endast tillgänglig att ansluta till, för att göra ändringar i, om man befinner sig på samma nätverk. Det betyder att om någon vill göra ändringar i gateway:ns inställningar eller kod så måste de befinna sig på samma lokala nätverk. De behöver sedan logga in i gateway:n med korrekt användarnamn och lösenord. Detta är också något som ökar säkerheten för att garantera att informationen som skickas från PLC:n inte är tillgänglig för obehöriga. I fall att någon vill göra ändringar i Node-RED måste de ha användarnamn och lösenord. Ska ändringar i den virtuella datorn göras måste användare logga in med SSH tillsammans med användarnamn

och lösenord. SSH kan också stängas av i brandväggsreglerna för den virtuella datorn och på så sätt göras mer svårtillgänglig.

Den data som kommer att skickas över nätet om våran IoT-lösning tas i bruk kommer förmodligen inte vara speciellt känslig men om stora mängder data kan samlas så kan en god förståelse för företagets produktion fås. Detta hade kunnat göras av ett konkurrerande företag som hade kunnat dra nytta av informationen. Det är därför viktigt och har varit centralt under arbetets gång att all information som skickas över nätet görs på ett säkert sätt.

Den andra etiska aspekten är en punkt från Sveriges Ingenjörers hederskodex:

Ingenjören bör sträva efter att förbättra tekniken och det tekniska kunnandet i riktning mot ett effektivare resursutnyttjande utan skadeverkningar (Sveriges Ingenjörer 2023)**.**

Det här arbetet har förbättrat Eltech Automations produkt HiFlex genom att göra den möjlig att koppla upp mot nätet. Under arbetets gång har möjlighet att utveckla förståelse för datorkommunikation och hur olika enheter kommunicerar erhållits. Arbetet kommer sedan att redovisas för anställda på Eltech Automation som jobbar med programmering och de kommer då att få tillfälle att lära sig mer om datorkommunikation och hur den här IoT-lösningen är uppbyggd.

Eftersom lösningen gör det möjligt för operatörer att se driftsinformation direkt i en webbläsare behöver de inte gå nära en HiFlex och kolla dess HMI för att ta reda på driftinformation. Operatörer kan därför använda sin tid mer effektivt. Möjligheten till remote access för att koppla upp sig mot PLC:n sparar även tid för Eltech Automations ingenjörer då de undviker resor till kund för att göra enklare felsökning eller omprogrammering.

6.3. Framtida utvecklingsmöjligheter

Det finns mycket med arbetet som skulle kunna utvecklas men på grund av tidsbegränsningen behövde avgränsningar göras. Några utvecklingsmöjligheter är:

Fler lägen för status

På webbsidan finns det bara två lägen för maskinens status. Dessa är “Running” och “Stopped”. Fler statuslägen hade varit önskvärt exempelvis “Idle” om maskinen till exempel väntar på mer material.

Bekräfta mailadress vid kontoregistrering

När en användare ska göra ett nytt konto behöver den inte bekräfta att det faktiskt är användarens egen mailadress som den anger. Det påverkar inte säkerheten avsevärt då varje mailadress ändå måste läggas in i databasen manuellt men det hade varit ett extra lager av säkerhet.

Inloggning med arbetsmail/sociala medier-konto

För att underlätta för användaren hade inloggning med sociala medier eller genom olika mailtjänster kunnat implementeras. Detta hade förenklat det för användaren eftersom den då inte behöver komma ihåg ytterligare inloggnings-uppgifter utan enkelt kunna använda något befintligt konto. Om användaren har arbetsmail i Google eller Microsoft hade denna kunnat användas för att undvika att dela privata kontouppgifter till arbetsrelaterade webbtjänster.

Möjlighet att spara eller logga data

Som databasen är konfigurerad nu så sparas ingen data utan det är bara de senaste värdena som visas. Det hade varit fördelaktigt om det var möjligt att gå tillbaka i någon typ av logg för att exempelvis se när olika larm har skett eller när maskinen har varit ur drift.

Visa data som grafik

Viss typ av data hade varit mer överskådlig om den visades i form av grafik eller symboler. Larm hade exempelvis kunnat symboliseras med en röd lampa eller status “Running” med en grön. Grafiken hade även kunnat kombineras med loggade data för att exempelvis jämföra produktionen över flera dagar med stapeldiagram.

7. Terminologi

AJAX - "Asynchronous JavaScript and XML". Verktyg som kombinerar HTTP-request för att hämta data från en webbserver med JavaScript och HTML DOM för att visa eller använda den hämtade datan. Möjliggör uppdatering av data utan att ladda om en hel webbsida.

BASIC - Programmeringsspråk.

Curl - Mjukvara som möjliggör kommunikation mellan en klient och en server genom en terminal.

DHCP - "Dynamic Host Configuration Protocol". Ett protokoll som tilldelar IP-adresser till datorer på ett nätverk.

DNS - "Domain Name Service".

IDE - "Intergrated Development Environment". Utvecklingsmiljö för programmering.

JSON - Ett textformat som är oberoende av programmeringsspråk.

LAN - "Local Area Network".

OIDC - "OpenID Connect". Protokoll för identitetsautentisering för att bekräfta autentisering mellan applikationer utan att dela känslig användardata.

PuTTy - Terminalemulator som användes istället för terminalen vilket förenklade anslutningen från en Windows-dator till den virtuella Linux-datorn genom SSH.

Rung - Rad eller gren i ladder-program.

SDK - "Software development kit". Utvecklingsverktyg för att bygga applikationer till specifika plattformar.

Shellscript - Kod som automatiserar exekvering av program i terminalen.

SSH - "Secure Shell" Metod. Protokoll för att ansluta sig säkert till andra datorer. Används här för anslutning till den virtuella datorn.

Sysmac Studio - Programmeringsmiljö för Omron PLC.

URL - Adressen till en unik resurs på internet.

WAN - "Wide Area Network".

8. Källförteckning

Auth0 by Okta (u.å. a). *Aut0 Overview.* https://auth0.com/docs/get-started/auth0-overview [Hämtad 2024-04-24].

Auth0 by Okta (u.å. b). *Apache.* https://auth0.com/docs/quickstart/webapp/apache [Hämtad 2024-04-24].

Auth0 by Okta (u.å. c). PHP: Getting Started using Auth0-PHP. https://auth0.com/docs/libraries/auth0-php [Hämtad 2024-04-25].

Auth0 by Okta (u.å. d). *PHP: Logging in, out, and returning user profiles with Auth0-PHP*. https://auth0.com/docs/libraries/auth0-php/auth0-php-basic-use [Hämtad 2024-04-25]

Auth0 by Okta (u.å. e). *Flexible pricing for developers & companies.* https://auth0.com/pricing [Hämtad 2024-04-24]

Apache (u.å.). *The Number One HTTP Server On The Internet.* https://httpd.apache.org/ [Hämtad 2024-05-02].

Cavalieri, S., Mulè, S. (2021) Interoperability between OPC UA and oneM2M. *J Internet Serv Appl* 12, 13. https://doi.org/10.1186/s13174-021-00144-9

Composer (u.å.). *Introduction.* https://getcomposer.org/doc/00-intro.md [Hämtad 2024-04-24].

Computer Sweden (u.å.). *Om oss.* https://computersweden.se/om-oss/ [Hämtad 2024-05-06].

Eltech Automation (2022 a). *Ett utvecklingsföretag inom Automation och Mekanik.* https://www.eltechautomation.se/om-oss [Hämtad 2024-04-22].

Eltech Automation (2022 b). *Empowering growth through industrial automation and robotics.* https://www.eltechautomation.se/ [Hämtad 2024-04-22].

Ewon (2021). *Programming Reference Guide.* https://developer.ewon.biz/system/files_force/rg-0006-01-en-basic-programming.pdf [Hämtad 2024-04-08].

IT-ord (2019). *Gateway.* https://it-ord.idg.se/ord/gateway/ [Hämtad 2024-04-23].

Kossi D. T. S. (2019). *How to use Apache to redirect requests for Node-RED?.* [foruminlägg], 3 Maj. https://stackoverflow.com/questions/55975504/how-to-use-apache-to-redirect-requests-for-node-red [Hämtad 2024-05-07].

Let's Debug (u.å.). *Let's Debug*. https://letsdebug.net/ [Hämtad 2024-05-06].

Microsoft Azure (u.å. a). *General availability: Microsoft Azure available from new cloud region in Sweden.* https://azure.microsoft.com/sv-se/updates/microsoft-azure-available-from-new-cloud-region-in-sweden/ [Hämtad 2024-05-02].

Microsoft Azure (u.å. b). *Virtual machines: virtual computers within computers.* https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-a-virtual-machine [Hämtad 2024-05-04].

MySQL (u.å. a). *6.4.2 mysql_secure_installation — Improve MySQL Installation Security.*
https://dev.mysql.com/doc/refman/8.0/en/mysql-secure-installation.html
[Hämtad 2024-04-05].

MySQL (u.å. b). *1.2.1 What is MySQL?*.
https://dev.mysql.com/doc/refman/8.3/en/what-is-mysql.html
[Hämtad 2024-04-08].

Nationalencyklopedin (u.å. a). *HTTP*.
http://www.ne.se/uppslagsverk/encyklopedi/lång/http [Hämtad 2024-05-07].

Nationalencyklopedin (u.å b). *IEEE*.
https://www.ne.se/uppslagsverk/encyklopedi/l%C3%A5ng/ieee
[Hämtad 2024-05-08].

Nationalencyklopedin (u.å. c). *proxyserver*.
https://www.ne.se/uppslagsverk/encyklopedi/l%C3%A5ng/proxyserver
[Hämtad 2024-05-08].

Nationalencyklopedin (u.å. d). *VPN*.
http://www.ne.se/uppslagsverk/encyklopedi/lång/vpn [Hämtad 2024-05-07].

Nationalencyklopedin (u.å. e). *Om oss*.
https://faq.ne.se/vad-ar-ne/ [Hämtad 2024-05-04].

Node.js (2023). *Node.js 21 is now available!*.
https://nodejs.org/en/blog/announcements/v21-release-announce
[Hämtad 2024-04-09].

Node-RED (u.å. a). *About*.
https://nodered.org/about/ [Hämtad 2024-04-23].

Node-RED (u.å b). *Running on Microsoft Azure.* https://nodered.org/docs/getting-started/azure [Hämtad 2024-04-09].

Node-RED (u.å. c). *Post JSON data to a flow.* https://cookbook.nodered.org/http/post-json-data-to-a-flow [Hämtad 2024-04-23].

Node-RED (u.å. d). *node-red-node-mysql.* https://flows.nodered.org/node/node-red-node-mysql [Hämtad 2024-05-02].

Nunamaker, J. F., Chen, M., & Purdin, T. D. M. (1990). Systems Development in Information Systems Research. *Journal of Management Information Systems*, *7*(3), s.89–106. http://www.jstor.org/stable/40397957.

Omron Industrial Automation EMEA (2020). *How to configure OPC-UA on SYSMAC NX-102 controllers* [video]. https://www.youtube.com/watch?v=JiJK1pU9XMU [Hämtad 2024-04-18].

Orcid (u.å.). *Salvatore Cavalieri.* https://orcid.org/0000-0001-9077-3688 [Hämtad 2024-05-06].

Sauter, T & Lobashov, M. (2011). How to Access Factory Floor Information Using Internet Technologies and Gateways. *IEEE Trasactions on Industrial Informatics,* 7(4), s-699-712. https://ieeexplore.ieee.org/abstract/document/6009198/authors#authors

SpringerOpen (u.å). *About SpringerOpen.* https://www.springeropen.com/about [Hämtad 2024-05-06].

Stallings, W. & Brown, L. (2018). *Computer Security: Principles and Practice.* 4 uppl., Pearson.

Sveriges Ingenjörer (2023). *Hederskodex.* https://www.sverigesingenjorer.se/om-forbundet/organisation/hederskodex/ [Hämtad 2024-05-08].

Ubuntu (2023). *How to install Apache2.* https://ubuntu.com/server/docs/how-to-install-apache2 [Hämtad 2024-04-11].

Ubuntu (u.å.). *The Ubuntu lifecycle and release cadence*. https://ubuntu.com/about/release-cycle [Hämtad 2024-05-02].

W3Schools (u.å. a). *PHP Introduction.* https://www.w3schools.com/php/php_intro.asp [Hämtad 2024-05-09].

W3Schools (u.å. b). *AJAX Introduction*. https://www.w3schools.com/xml/ajax_intro.asp [Hämtad 2024-05-02].

W3Schools (u.å. c). *jQuery Get Started.* https://www.w3schools.com/jquery/jquery_get_started.asp [Hämtad 2024-05-02].