CODEN:LUTEDX/(TEIE-3104)/1-42(2022)

Industrial Electrical Engineering and Automation

Trådlös larmkommunikation från robotcell till Android-applikation

Anton Ekelund

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

LUNDS
UNIVERSITET

BACKER

EXAMENSARBETE

Anton Ekelund

Trådlös larmkommunikation från robotcell till Android-applikation

Department of Electrical and Information
Technology Faculty of Engineering,
LTH, Lund University SE-221 00 Lund, Sweden

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 2022

Sammanfattning

Examensarbetet som denna rapport avhandlar har utförts hos Backer AB i Sösdala.

I den moderna tillverkningsindustrin blir allt fler processer automatiserade och robotstyrda. En stor del av de automatiserade maskiner som finns i Backers fabrik är utrustade med principen att ha ett magasin för obearbetade produkter, vilket resulterar i att operatören inte alltid behöver vara närvarande vid maskinen. Detta ger operatören möjlighet att köra flera automatiserade maskiner samtidigt, men med konsekvensen att uppsikten över maskinerna minskar.

Det här examensarbetet har undersökt hur ett system som meddelar berörd operatör om oplanerade produktionsstopp i automatiserade maskiner kan se ut. Under arbetet har en fungerande prototyp av systemet tagits fram och testats på en robotcell i industriell miljö. Överföring av larmsignaler sker trådlöst från robotcellen, via en PLC, till en datorenhet. Där behandlas signalerna och skickas sedan vidare till en applikation för Android som ger operatören möjlighet att ta emot information om produktionsstopp. Systemet är tillämpligt på stora delar av Backers maskinpark.

Eftersom operatören blir meddelad direkt när produktionsstopp sker går mindre tid förlorad och produktionen kan effektiviseras.

Nyckelord

robotcell, PLC, Node-RED, MQTT, larm, Android

Abstract

The bachelor thesis that this report discusses has been carried out at Backer AB in Sösdala.

In the modern manufacturing industry, more and more processes are becoming automated and robot controlled. A large part of the automated machines available in Backer's factory are equipped with the principle of having a magazine of unprocessed products, which means that the operator does not always have to be present at the machine. This gives the operator the opportunity to run several automated machines at the same time, but with the consequence that the supervision of the machines decreases.

This thesis has investigated how a system that notifies the relevant operator of unplanned production stoppages in automated machines can be designed. During the work, a working prototype of the system has been developed and tested on a robot cell in an industrial environment. Alarm signals are transmitted wirelessly from the machine, via a PLC, to a computer unit. There, the signals are processed and then forwarded to an application for Android that gives the operator the opportunity to receive information about production stoppages. The system is applicable on large parts of Backer’s machine park.

Since the operator is notified immediately when the production stops, less time is lost and production can be made more efficient.

Keywords

Robot cell, PLC, Node-RED, MQTT, alarm, Android

Förord

Jag vill tacka Backer AB för att jag fick möjligheten att göra mitt examensarbete hos er. Jag vill även ge ett stort tack till min handledare Alexandros Soulakis och övrig involverad personal för de råd och erfarenheter ni delat med er av.

Terminologi

Robotcell	Ett inhägnat område där robot och andra automatiserade maskiner opererar.
API	Applikationsgränssnitt
Protokoll	En standard eller regelsamling för IT-kommunikation
I/O	In- och utgångar
Emulator	En mjukvara eller hårdvara som möjliggör ett datorsystem att efterlikna ett annat datorsystem.

Innehållsförteckning

1. Inledning

I detta avsnitt beskrivs företaget där examensarbetet utfördes. Därefter beskrivs och förklaras examensarbetets syfte, mål, problemformulering, motivering och avgränsningar.

1.1. Bakgrund

Backer AB grundades 1949 i Sösdala, där huvudkontoret fortfarande är beläget. Företaget har specialiserat sig på utveckling och tillverkning av elektriska rörvärmeelement. Inom Backergruppen ingår 70 bolag och 10000 medarbetare vilket gör Backer till den främsta aktören inom sin bransch.

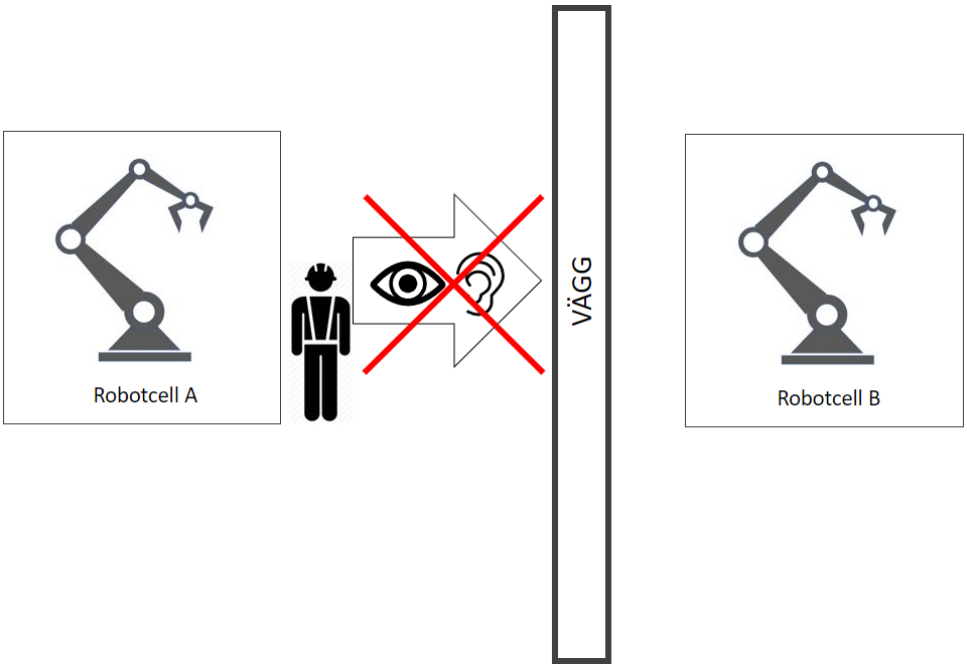
Inom produktionsindustrin blir allt fler arbetssysslor ersatta av automation och robotisering. Även om automationen sköter sig själv, krävs ibland tillsyn och assistans av robotoperatörer. Det finns flera orsaker till att en robotcell behöver tillsyn, t.ex. att

- magasinet av oprocessade artiklar behöver fyllas på.
- ytan där processade artiklar placeras är full.
- något i maskinen gått sönder eller blivit fel.

Detta resulterar då i att cellen automatiskt stoppas och därmed stoppas även produktionen. Det är först när operatören startar upp cellen igen som produktionen återupptas.

Eftersom varje robotcell ofta ej har ett bemanningsbehov på 100%, ansvarar en robotoperatör ofta över flera robotceller. Det medför att operatörens uppmärksamhet pendlar mellan cellerna. Dessa celler kan vara långt ifrån varandra eller kan de ha väggar emellan sig. Därmed har operatören inte uppsikt över robotcell A när vederbörande arbetar vid robotcell B.

Det finns tillfällen då operatören inte finns tillgänglig direkt när ett produktionsstopp sker i en robotcell, t.ex. vid arbete vid annan robotcell eller annan maskin. Vid dessa tillfällen får operatören i nuläget ingen signalering om att robotcellen är stoppad, utan förblir ovetandes om detta tills dess att operatören är fysiskt närvarande vid cellen. Detta medför att produktionen i robotcellen kan stå stilla under onödigt lång tid.



Figur 1. Problematiken illustrerad

1.2. Syfte

Syftet med examensarbetet har varit att undersöka om tiden då robotceller står stilla på grund av oplanerade stopp kan reduceras genom att låta operatören få ett elektroniskt meddelande om produktionsstopp från robotcellen till en trådlös handenhet eller smartklocka. Det förväntade resultatet är en ökad produktion till följd av minskad tid som robotcellen är under produktionsstopp.

1.3. Målformulering

Examensarbetet har undersökt en metod att meddela en maskinoperatör angående produktionsstopp i en robotcell eller annan automatiserad maskin. Detta genom att låta en larmsignal skickas från maskinen till operatörens handenhet eller smartklocka. Eftersom företagets automatiserade maskiner och robotceller har olika slags utförande skulle examensarbetet ta fram en modulär lösning som med enkla metoder kan anpassas till varje maskin.

1.4. Problemformulering

Examensarbetet har undersökt och svarat på följande frågor:

- Vilken typ av kommunikation mellan automatiserad maskin och datorenhet är lämplig för ändamålet?
- Ska operatören få information om produktionsstopp via en applikation, e-post eller på annat vis?
- Hur ska datorenheten skicka ett larmmeddelande till en handenhet och vilka protokoll ska användas?
- Om en smartphone-applikation ska användas, ska denna då programmeras eller ska en redan befintlig applikation användas?

1.5. Motivering

Efter kontakt med Backer AB presenterades flera förslag på examensarbete där majoriteten verkade intressanta. Examensarbetet valdes på grund av att arbetet kring problemställningen omfattar flera av de grundläggande momenten i utbildningen. Dessutom var det ett system som var användbart och hade potential att öka företagets produktion.

1.6. Avgränsningar

Examensarbetet har omfattat utvecklingen av en modulär lösning för Backer AB som i teorin ska fungera på majoriteten av företagets automatiserade maskiner. Examensarbetet har dessutom tagit fram en prototyp som är utvecklad och testad för en specifik robotcell i Backers fabrik. Därmed har systemets funktion inte testats på andra typer av maskiner och kan därför inte garanteras.

2. Teknisk bakgrund

I detta avsnitt kommer den tekniska bakgrunden av de hårdvaruenheter, program och tekniker som använts under examensarbetet att beskrivas och förklaras.

2.1. PLC

En PLC (Programmable Logic Controller) är ett programmerbart styrsystem som ofta används i automatiserade maskiner och anläggningar inom tillverkningsindustrin. PLC:n som har använts är en Mitsubishi Electric FX5U [1].

2.2. HMI

Human-Machine Interface är ett användargränssnitt som möjliggör mänsklig kommunikation med en maskin. Detta kan vara en operatörspanel, knappsats eller en pekskärm som är ihopkopplad med maskinen. HMI:et används bland annat till att kunna styra eller programmera maskiner. HMI:et som användes i tester och simuleringar av systemet under detta examensarbete var en Mitsubishi Electric GT2508-VTBD [2].

2.3. Handenhet

Handenheten som har använts är av typen Unitech Ea510 och är en Androidbaserad enhet anpassad för industriell miljö [3]. Operatörerna på Backer AB har nyligen blivit tilldelade dessa enheter för stämpling och orderhantering och med tanke på att dessa enheter alltid bärs av operatören fanns det ett önskemål från företaget att inkludera dessa i kommunikationslösningen.

Systemet är även kompatibelt med vanliga Android-mobiltelefoner som inte är avsedda för industriell miljö.

2.4. Node-RED

Node-RED är ett verktyg för sammankoppling av hårdvaruenheter, API:s och internettjänster. Verktyget är utvecklat av IBM och är gratis att ladda ner och använda [4].

Programmeringen är flödesbaserad och görs direkt i webbläsaren. Ett flöde i Node-RED består av en serie sammankopplade noder, där varje nod representerar en viss funktion. Noderna ser ut som block och placeras på en canvas genom drag-and-drop [4]. Noderna använder ett *message*-objekt för att överföra data mellan varandras in- och utgångar [5].

Noderna finns förprogrammerade men de kan också programmeras för hand. Det finns olika typer av noder och de som använts i detta examensarbete beskrivs nedan.

- **inject**
  inject-noden startar ett flöde genom att mata in ett *message*-object genom ett musklick. Noden kan också programmeras att autostarta vid vissa tidpunkter eller vid vissa intervaller [6].

- **debug**
  Noden används för att visa innehållet i ett message-objekt. Denna nod är användbar vid felsökning av flödet [6].

- **mqtt-out**
  Denna nod används för att skicka MQTT-meddelanden, det vill säga att den fungerar som en publisher [6].

- **change**
  Noden kan användas för att modifiera message-objektet, t.ex. ändra, sätta eller radera ett värde [6].

- **function**
  Denna nod gör det möjligt att köra ett JavaScript på det message-objekt som förs över [6].

- **filter**
  filter-noden kan programmeras att filtrera bort vissa värden eller upprepningar av indata [7].

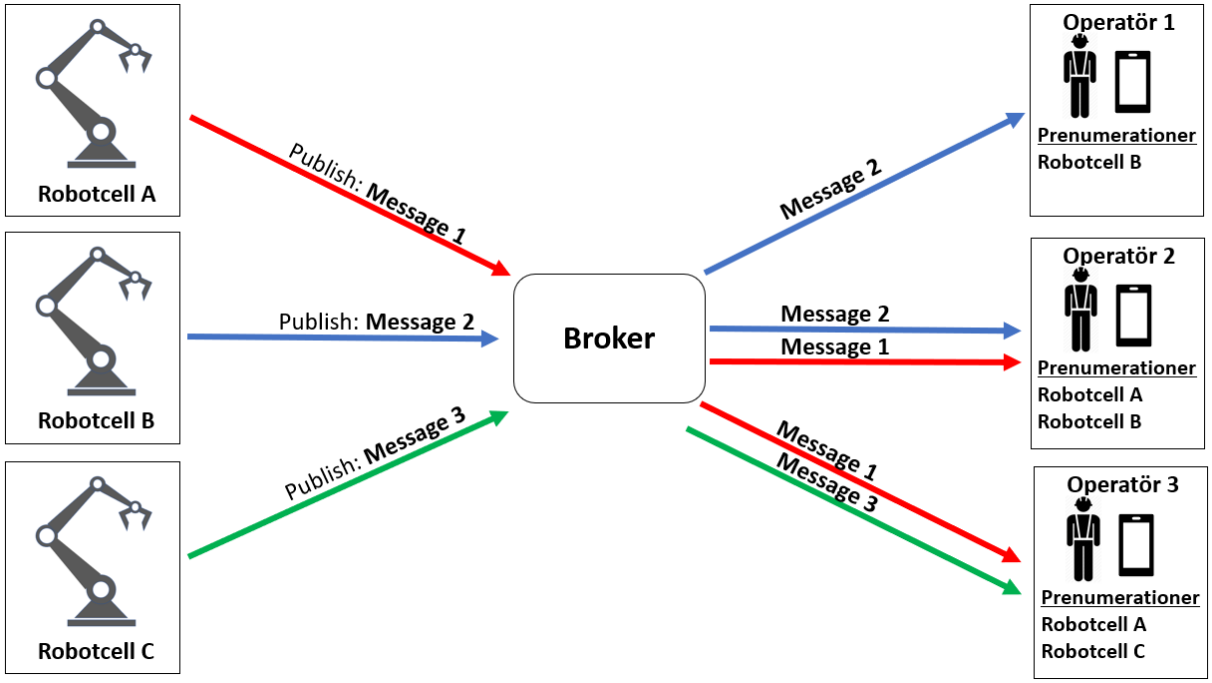
- **MC protocol read**
  Noden använder Mitsubishi Electrics egna kommunikationsprotokoll MELSEC för att avläsa värden från en PLC [8].

2.5. MQTT

MQTT står för MQ Telemetry Transport och är ett protokoll för kommunikation mellan sändare och mottagare. MQTT är ett så kallat publish/subscribe-protokoll, där sändare och mottagare inte är direkt anslutna med varandra utan kommunicerar via en broker som fungerar som en mellanhand [9].

Sändarens roll är att ansluta sig till en broker och där publicera de meddelanden som den vill skicka ut. Varje meddelande publiceras i ett specifikt ämne (topic). Sändare kan t.ex. utgöras av en sensor [9].

Brokerns roll kan liknas vid ett postkontor, där den håller koll på vilka klienter som prenumererar på vilka topics. När ett meddelande publicerats från en sändarklient skickar brokern vidare detta till de klienter som prenumererar på det topic som meddelandet publicerats i. För att ta emot meddelanden måste mottagarklienten prenumerera på det topic som meddelandet skickas ut i. Ett meddelande från en sändarenhet kan därför tas emot av flera mottagenheter [9].



Figur 2. MQTT-protokollet

Ett topic kan vara organiserat hierarkiskt, separerade med ett snedstreck. Detta system liknar det fil- och mappsystem som används i en PC [9].
Ett topic kan se ut på följande sätt:
Backer / FabrikA / avdelning1 / maskin1 / komponent1

2.6. Mosquitto

Mosquitto är en MQTT-broker utvecklad av Eclipse. Mjukvaran är gratis att ladda ner och använda. Mosquitto har inget grafiskt användargränssnitt utan alla inställningar görs i programmets konfigurationfil [10].

2.7. Android Studio

Android Studio är en IDE (integrerad utvecklingsmiljö) för utveckling av applikationer för Android. Android Studio utvecklades av Google och kom ut på marknaden 2014. Programmeringen av applikationer kan göras i programmeringsspråken Kotlin, Java och C++. Programmet är gratis att ladda ner och använda [11].

En viktig funktion är den emulator som finns tillgänglig för att testa och debugga applikationer under utvecklingen.

2.8. Foreground Service

När Android 8.0 Oreo lanserades 2017 infördes förändringar för att förbättra prestandan på Android-enheter. För att undvika osynliga processer som konsumerar batteri och RAM-minne infördes begränsningar för vad en applikation kan göra när den körs i bakgrunden [12].

En så kallad *Foreground Service* är en process i Android som utför en operation som är synlig för användaren genom en notifikation. Servicen används när en process ska kunna fortgå även om användaren inte interagerar med applikationen vid den tidpunkten. När servicen är igång visas en notifikation i notifikationspanelen på enheten och denna går ej att ta bort utan att tvinga fram ett stopp av antingen servicen eller applikationen.
Ett exempel på en Foreground Service är en mediaspelare som spelar ljud även när andra applikationer används eller om enheten är i viloläge. Genom notifikationen i en förgrundsprocess blir användaren uppmärksam på att applikationen är igång [13].

2.9 Robotcell och lokal

Robotcellen som valts ut av företaget för installation av den protoyp som examensarbetet tagit fram är placerad i en fabrikslokal där det även finns andra maskiner och tillverkningsprocesser. Robotcellen tillverkar genomströmningsvärmare och tillverkningen sker i flera olika moment. När operatören som ansvarar för robotcellen befinner sig vid *arbetsplats B* är sikten mot robotcellen skymd av väggar, pallställ, hyllor och andra arbetsytor.

Figur 3. Ritning av fabrikslokalen

3. Metod

I detta avsnitt kommer arbetsmetoden kring de olika delmomenten i examensarbetet beskrivas.

3.1. Planering och initial beskrivning

I en inledande fas hölls flertalet möten med handledare och berörd personal på Backer AB, där examensarbetet beskrevs, krav på systemet presenterades och en preliminär tidplan togs fram. Det gjordes även en rundvandring i fabriken där observationer gjordes hur och var robotcellen som skulle utnyttjas för testning och prototyp var placerad. Därefter kunde en initial beskrivning av examensarbetet skrivas. När den initiala beskrivningen av examensarbetet var godkänd av handledare och examinator från LTH kunde arbetet påbörjas.

3.2. Förstudier och systemdesign

I förstudiernas början togs det fram data och tekniska detaljer om de befintliga komponenter och system som användes i robotcellen och i det befintliga nätverket. Undersökningsarbetet utgjordes av studier av tekniska specifikationer och intervjuer av berörd personal på Backer AB och dess samarbetspartners. Fortsättningsvis delades förstudierna in i två sammanhörande delar för att få en bättre överblick.

Den första delen utgjordes av informationshämtning angående extrahering av data från styrsystemet. Här studerades främst vilka överföringsprotokoll som skulle kunna bli aktuella och vilken mjukvara och hårdvara som lämpade sig för hämtningen. Inledningsvis gjordes en bredare sökning efter information på internet på de nyckelord som ansågs vara relevanta. För att få en djupare förståelse vilka möjligheter som fanns angående styrsystemets kommunikation gjordes därefter flera samtal och intervjuer, både internt med kunnig personal på Backer och externt med samarbetspartners.

Den andra delen av förstudierna bestod av research angående vilket kommunikationsmedium som skulle kunna komma att användas i överföringen mellan datorenhet och handenhet. Här söktes information om vilka protokoll som lämpade sig bäst i överföringen och vilken typ av mjukvara som krävdes i datorenheten och mottagarenheten. Informationen hittades främst på internet men även här rådfrågades personal på Backer.

Genom att sedan studera och kombinera de alternativ som kommit fram i de två delarna av förstudien kunde en preliminär modell av systemet skissas fram. Denna modell undersöktes vidare genom att ta fram djupare information för att bekräfta dess genomförbarhet. En specifikation gjordes på de funktioner och komponenter som krävdes av varje delmoment i systemet.

En undersökning utfördes angående de tekniska detaljerna och hur de olika delarna i systemet förhöll sig och överensstämde med varandra. De komponenter som ingick i systemet skulle även uppfylla de krav som ställdes på dem från företagets IT-avdelning. Eftersom examensarbetet har en tidplan var det dessutom viktigt att arbetet kring utvecklingen av systemet inte skulle kräva orimligt mycket tid. Även om projektet inte hade en övre gräns rent ekonomiskt gjordes en utvärdering för att säkerställa att utgifterna för inköp av komponenter höll sig på en resonlig nivå.

3.3. Hårdvara och mjukvara

När en modell tagits fram över hur systemet skulle se ut och vilka typer av komponenter som skulle användas utfördes en djupare undersökning för att bestämma vilken specifik hårdvara som skulle användas. Här var det framförallt funktion och tillgänglighet som prioriterades men även användarvänlighet, ekonomiska fördelar och kvalitet togs hänsyn till i valet.

I valet av mjukvara för hämtning av data från PLC:n gjordes en grundlig jämförelse av de alternativ som undersökts. Här var enkelhet och användarvänlighet prioriterat.

I valet av kommunikationsmedium mellan sändande maskin och mottagande operatör gjordes en jämförelse av de alternativ som undersökts. Att kunna lägga till mottagarenheter utan att behöva programmera om sändarenheten var prioriterade.

Efter testet med flertalet applikationer som möjligen skulle lämpa sig som meddelandemottagare för MQTT, kunde en undersökning göras där funktioner som behövdes i applikationen listades.

3.4. Programmering och tester

Utvecklingsfasen i examensarbetet sattes igång med att bygga en rigg för testning och simulering av systemet. Detta gjordes genom att programmera en PLC och ett HMI så att ett produktionsstopp kunde simuleras.

Därefter kunde programmeringen av mjukvaran för hämtning av data från PLC:n påbörjas. Genom att följa internetbaserade guider kunde rätt portar öppnas och lämpliga kommunikationsprotokoll väljas med hjälp av PLC-programmeringsmjukvaran. Med hjälp av Node-RED kunde sedan en PC koppla upp sig för hämtning av data från PLC:n. Webbsökning efter information om tillvägagångssätt gjordes parallellt med programmeringen av mjukvaran.

Information och specifikationer hämtades för mjukvaran för meddelandekommunikation som sedan laddades ner, installerades och programmerades. En trådlös router användes för att skapa ett trådlöst nätverk som den meddelandemottagande handenheten kunde ansluta sig till. En färdigutvecklad applikation för Android installerades på handenheten för att testa MQTT-uppkopplingen mellan handenhet och broker. Denna applikation saknade vissa grundläggande funktioner och användes därför inte i den slutgiltiga versionen av systemet.

För att testa systemet med flera insignalskällor byggdes testriggen ut med ytterligare en PLC allt eftersom arbetet fortskred.

För att kunna programmera den applikation som skulle användas för meddelandemottagning i handenheten laddades Android Studio ner och installerades. Programspråket Java användes under programmeringen och guider och information kunde hämtas från studentlitteratur, Androids och programspråksutvecklarens webbsida. Programmeringen inleddes med en planeringsfas där de grundläggande funktionerna listades. Därefter kunde varje funktion programmeras. Applikationen testades flitigt under programmeringens gång med hjälp av en Android emulator i arbetsdatorn.

Eftersom handenheterna ska användas av operatörer upp till 9h per dag utan att batteriet laddas, gjordes ett test för att beräkna batterikonsumtionen för den applikation som utvecklats. Testet utfördes genom att ladda handenheten till 100% och sedan starta applikationen. Applikationen var uppkopplad till brokern och tog emot 10 meddelanden per timme. När 9 timmar hade gått lästes batteriladdningsprocenten av och noterades. För jämförelse gjordes ett kontrolltest där handenheten inte hade applikationen igång.

Den trådlösa routerns räckvidd var inte specificerad på tillverkarens hemsida och telefonsupporten kunde heller inte ge besked om dess räckvidd. Därför utfördes ett test för att bestämma om routern hade tillräcklig räckvidd för ändamålet. Routern placerades i närheten av den robotcell som prototypen skulle installeras på. Genom att etablera en uppkoppling till routern med en mobiltelefon och förflytta denna till de punkter där uppkopplingen förlorades kunde routerns räckvidd uppskattas**.** Den trådlösa åtkomstpunkten som skulle kopplas till PLC:n testades på liknande vis.

3.5. Kommunikation

En viktig del av examensarbetet har varit den kommunikation som hölls med handledare och personal på Backer AB. Samtal hölls löpande under examensarbetet som uppdaterade handledare om arbetets gång och vad som behövdes för att fortsätta arbetet. Samtalen har också gett en klarare målbild som i sin tur förenklat arbetet.

3.6. Källkritik

Samtliga källor utgår ifrån tillverkarens dokumentation kring dess produkt. Dessa anses därför vara tillförlitliga.

4. Analys och genomförande

I detta avsnitt behandlas den analys som låg till grund för de beslut som fattades under examensarbetets gång. Dessutom beskrivs examensarbetets genomförande mer ingående.

4.1. Kravspecifikation

De krav och önskemål som företaget hade på systemet presenteras här i punktform.

- Systemet får inte ha några löpande kostnader, såsom avgiftsbelagda licensprenumerationer eller abonnemang.
- Systemet ska med liten modifikation kunna hantera hämtning av larmsignaler från en stor del av Backers automatiserade maskiner. Detta innebär att den del av systemet där data från maskinen samlas in ska vara anpassad efter den maskin systemet ska installeras på. Eftersom varje robotcell och automatiserad maskin i företagets fabrik är unik är förutsättningarna olika för dessa och därför skall ett system tas fram som fungerar oberoende av maskinens utförande.
- Varje operatör har nyligen blivit tilldelad varsin handenhet för orderhantering och därför fanns önskemål om att använda denna som meddelandemottagare i lösningen.

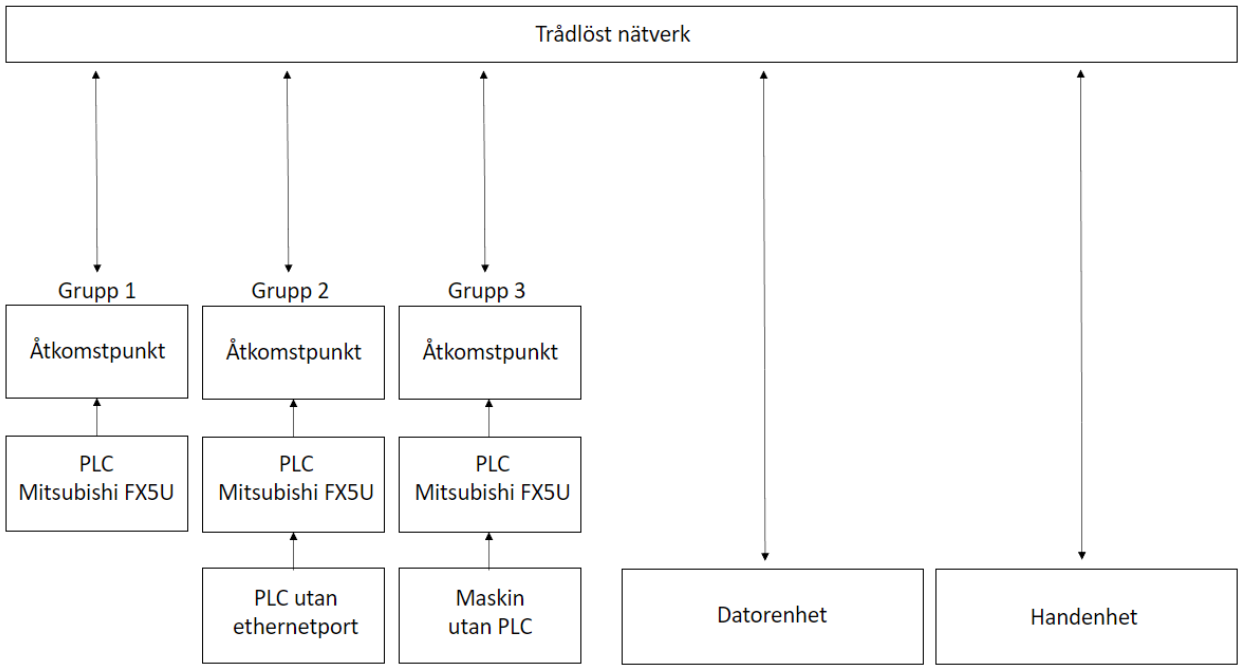
4.2. Styrsystem

De automatiserade maskinerna i Backers maskinpark kan delas in i tre grupper.

1. Maskiner med nyare PLC med möjlighet för trådbunden nätverksuppkoppling.
2. Maskiner med äldre PLC utan möjlighet för trådbunden nätverksuppkoppling.
3. Maskiner utan PLC.

Systemet som tagits fram i detta examensarbete bygger på att larmsignaler läses ut från en modern PLC med ethernetport. Detta betyder att en PLC med möjlighet för trådbunden nätverksuppkoppling måste installeras på de maskiner som tillhör grupp 2 och 3.

Av de maskiner som tillhör grupp 1 och 2 används nästan uteslutande PLC:er av fabrikatet Mitsubishi Electric. Eftersom systemet är utvecklat för att även kunna installeras på maskiner utan PLC med ethernetport, var det naturligt att då använda sig av just Mitsubishi Electrics PLC även på dessa, dels med anledning av att kunskap och erfarenhet fanns tillgängligt bland personal på företaget och dels för att inte behöva ändra metod vid inläsning av data från PLC:n.



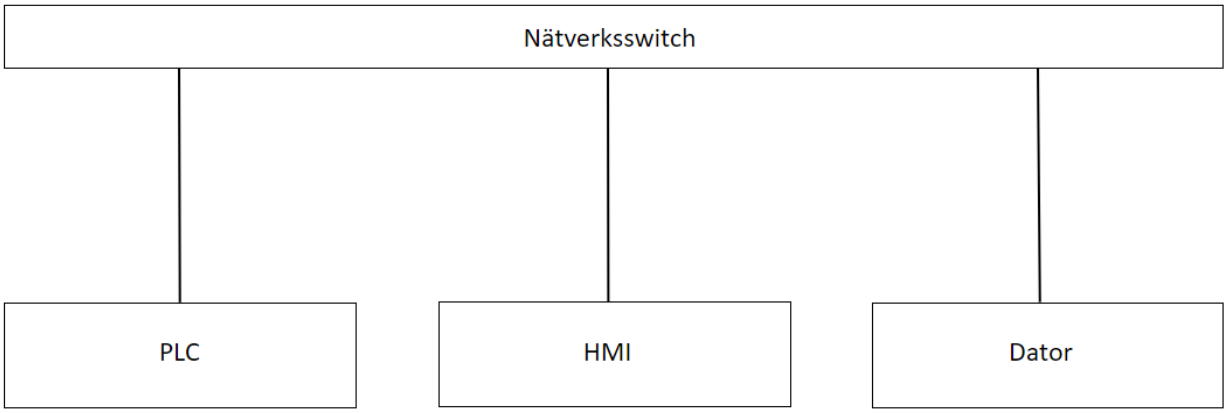
Figur 4. Gruppering av maskiner

Vid samtal med säljare från Mitsubishi Electric rekommenderades en Mitsubishi Electric FX5U Compact Edition. Compact Edition betyder att PLC:n är mindre än standardvarianten och tar därmed upp liten plats i installationsskåpet. Eftersom leverantören av denna PLC hade leveransproblem användes standardvarianten, Mitsubishi FX5U, istället.

4.3. Hämtning av data från styrsystem

I början av förstudierna var det oklart hur inläsningen av data från PLC:n skulle se ut. Initialt var det beskrivet att systemet skulle innehålla en mikrokontroller eller Raspberry Pi som skulle vara kopplad till PLC:ns utgångar. Då alla typer av mikrokontrollers och Raspberry Pi inte levde upp till företagets höga säkerhetskrav, gällande nätverk och enheter kopplade till detta, avslutades vidare studier för denna design.

När en skiss tagits fram på hur ett system skulle kunna se ut, byggdes en testrigg för att kunna testa systemet i allmänhet och testa hämtning av data från PLC:n i synnerhet. Denna testrigg bestod initialt av en PLC (Mitsubishi Electric FX5U), ett HMI (Mitsubishi Electric GT2508-VTBD), en nätverksswitch (Netgear ProSafe GS108), en dator (Windows 10) och diverse kablage. Denna testrigg kom att byggas ut allt eftersom arbetet fortlöpte.



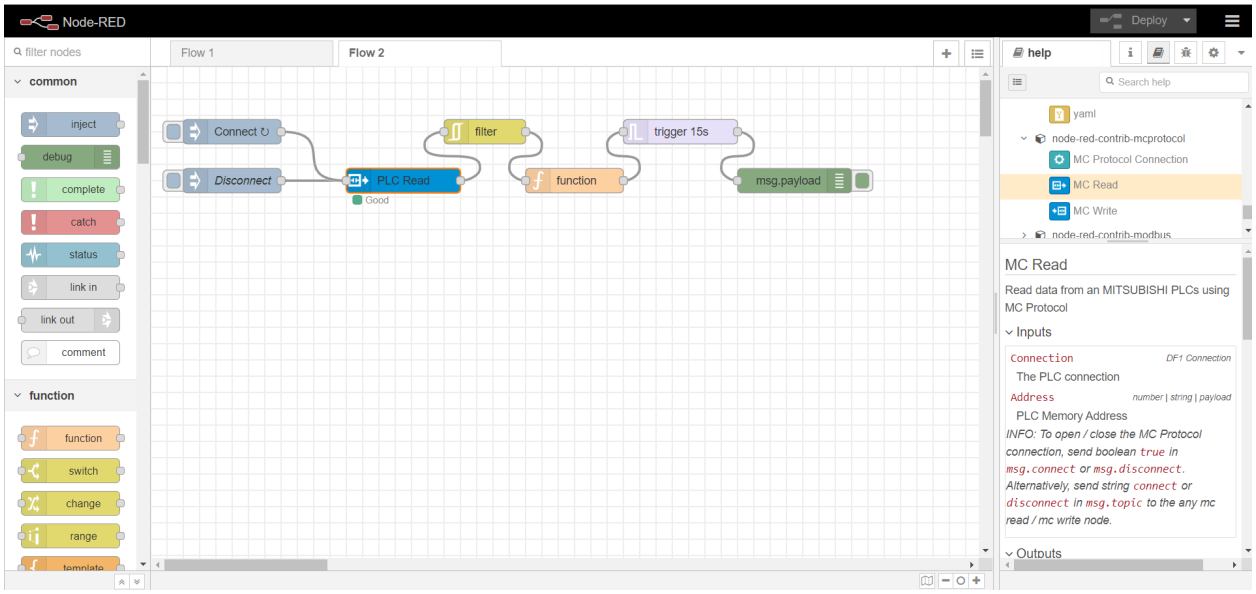
Figur 5. Testrigg

PLC:n och HMI:et programmerades för att kunna simulera ett produktionsstopp. Genom ett tryck på en knapp på HMI:et hoppade PLC:n ur sin programcykel.

För att kunna hitta PLC:n och HMI:et i nätverket programmerades IP-adresser för dessa i mjukvaran *GX Works3*. I samma mjukvara öppnades portar och protokoll i PLC:n vilket möjliggjorde kommunikation med PC:n. PC:n som användes för hämtning behövde dessutom ha samma subnätmask som PLC:n och HMI:et.

Flera olika kombinationer av kommunikationsmetoder och protokoll testades för att hitta det bäst lämpade. Efter att studerat möjligheterna för att extrahera data från PLC:n med Node-RED, hittades en metod för kommunikation som använder Mitsubishi Electrics egna protokoll MELSEC. Genom att använda noden *MC protocol* kunde data från specifika minnesceller hämtas från PLC:n.

När PLC:n kör sin programcykel stod minnescell M104 som *true*. Cykeltiden har på HMI:et programmerats till 10s. Varje gång cykeln slutförts ställs minnescellen till *false* i 0,5s innan cykeln börjar om och minnescellen återigen ställs till *true*.



Figur 6. Flöde i Node-RED

I Node-RED har flödet för att upptäcka produktionsstopp på testriggen programmerats enligt följande:

1. En *Inject*-nod (grå) startar flödet med frekvensen 10 gånger per sekund, vilket innebär att *MC Read* -noden (blå) hämtar data från en specifik minnescell i PLC:n. Hämtningen sker med MELSEC Protocol som är Mitsubishi Electrics egna kommunikationsprotokoll.
2. *Filter*-noden (gul) filtrerar bort upprepningar i indatan och skickar därför bara vidare flödet vid ändringar i indatan, dvs. när indatan ändras från *true* till *false* eller tvärtom.
3. *Function*-noden (orange) skickar bara vidare flödet om indatan är *true.*
4. *Trigger*-noden (lila) startar en timer varje gång noden får indata. När en given tid (här 15s) gått och ingen ny indata anländ löses triggern ut och message-objektet når slutdestinationen.

I valet av mjukvara för hämtning av data från PLC valdes Node-RED. Detta på grund av dess enkelhet och användarvänlighet. Node-RED hanterar inte enbart hämtning av data från PLC, utan även bearbetning av denna data och meddelandesändning till Android-applikation.

4.4. Kommunikationsmedium

När valet av kommunikationsmetod mellan datorenhet och handenhet skulle göras fanns flera parametrar att ta hänsyn till. Systemet skulle vara lättmanövrerat och inte kräva uppdateringar av mjukvara i tid och otid. Det skulle vara enkelt att lägga till nya användare på mottagarsidan men även på sändarsidan. Systemet fick heller inte använda sig av någon prenumerationstjänst som i förlängningen skulle vara kostsam. Initialt testades e-post som kommunikationsmetod då detta var enkelt att implementera i Node-RED. Efter en närmare undersökning kunde en större nackdel urskönjas; att det kan bli invecklat att lägga till och ta bort användare i mejllistor.

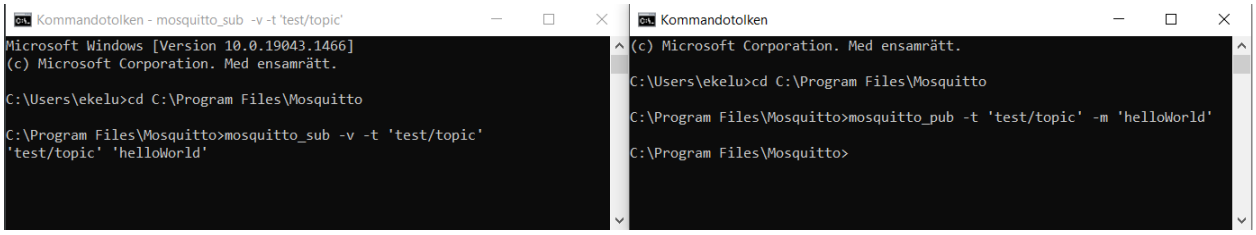
Därefter undersöktes också ett system där Node-RED sänder data till en MySQL-databas. En Android-applikation i operatörens handenhet skulle kontinuerligt kontrollera databasen efter uppdateringar och hämta data om uppdateringar gjorts. Denna data skulle presenteras i applikationen, dels via en notifikation men även i logg. På grund av stor batterikonsumtion och ineffektiv systemdesign avbröts undersökningen även för detta system.

Då MQTT-kommunikation finns som standard-nod i Node-RED var protokollet enkelt att testa. I MQTT-protokollet finns ingen direkt sammankoppling mellan sändare och mottagare eftersom brokern agerar mellanhand. Detta betyder att det inte behövs någon programmering på sändarenheten när nya mottagare läggs till och därför lämpar sig detta protokoll väl till uppgiften. Protokollet är dessutom lättviktigt och belastar därför inte nätverket i någon större utsträckning.

4.5. Broker

När beslut fattats att använda MQTT för kommunikation mellan datorenhet och handenhet gjordes en webbsökning efter information om lämplig broker. Då det fanns mest relevant information om brokern Eclipse Mosquitto laddades denna ner och installerades. Motiveringen till valet baseras också på goda erfarenheter kring tillverkarens övriga produkter.

Installationen på arbetsdatorn gick smärtfritt och brokern kunde börja testas. Till en början användes en testserver för att testa uppkopplingen. När detta lyckats, testades Mosquitto genom att öppna två separata terminalfönster. Det ena fönstret användes som *subscriber* och det andra som *publisher.* Testning skedde genom att skicka ett meddelande från publisher-fönstret till subscriber-fönstret.



Figur 7. Publish / subscribe med Mosquitto

4.6. Android-applikation

Under förstudierna testades en befintlig Android-applikation som laddades ner från Google Play Store. Detta för att se hur MQTT-protokollet fungerade och för att skapa en uppfattning om vilka funktioner som behövdes för att systemet skulle kunna fungera optimalt. Applikationen som testades saknade vissa nödvändiga funktioner och hade dessutom många funktioner som var onödiga för systemet. Därefter testades flera applikationer av samma typ men med liknande resultat. Detta gav motivation att programmera en applikation på egen hand för att optimera funktion och användarvänlighet.

Google har sedan 2019 rekommenderat app-utvecklare för Android att använda sig av programmeringsspråket Kotlin. På grund av bristande erfarenhet av detta språk programmerades appen i språket Java. Under programmeringen av applikationen användes programmet Android Studio.

De funktioner som ansågs vara viktiga för att systemet skulle fungera optimalt och dess beskrivningar och motiveringar listas nedan.

- **Skapa uppkoppling till broker**
  För att ta emot meddelande måste enheten vara uppkopplad till en broker på det lokala nätverket.
- **Lägga till prenumerationer på nya maskiner**
  Att med enkelhet kunna lägga till nya prenumerationer är en av de största fördelarna med MQTT.
- **Redigera inställningar för Brokern**
  Att i appen kunna redigera inställningarna för brokern, t.ex. dess IP-adress eller lösenord, var en viktig funktion. Detta för att slippa behöva uppdatera hela appen om någon av brokerns inställningar skulle behöva ändras.
- **Logg**
  En logg med de 10 senaste mottagna meddelandena som sparas i en fil.

- **Radera och redigera befintliga prenumerationer**
  Att kunna radera eller redigera någon av de befintliga prenumerationerna.
- **Förgrundsprocess**
  Eftersom appen ska kunna ta emot meddelanden även när den är nedstängd eller om Android-enheten är i viloläge behövdes en så kallad *foreground service* programmeras. Denna gör att applikationen kan köras i bakgrunden, även när användaren använder sig av andra appar eller om enheten är i viloläge.
- **Notifikationer**
  För att operatören ska kunna få reda på när ett meddelande tas emot var en essentiell funktion att kunna få notifikationer vid meddelande. Notifikationerna skulle visas visuellt genom ett meddelande i skärmen, vibrera och en meddelandesignal skulle ljuda.
- **Aktivera/inaktivera prenumerationer**
  Operatören behöver ha möjligheten att kunna inaktivera meddelanden från vissa maskiner för tillfällen då de inte arbetar vid dem.

4.7. Batterikonsumtion

Eftersom operatören använder handenheten upp till 9 timmar per dag var det viktigt att applikationen inte skulle förbruka för mycket batteri under denna tid. För att testa hur applikationen påverkade handenhetens batterikonsumtion gjordes ett test där:

1. Handenheten laddades till 100%.
2. Applikationen startades och började ta emot meddelanden.
3. Batterinivån lästes av efter 9 timmar.

För jämförelse gjordes ett kontrolltest med punkt 1 och 3, det vill säga utan att ha applikationen igång.

Förfluten tid	**Test med applikation**	**Kontrolltest**
0 h	100%	100%
4,5 h	96%	99%
9 h	91%	98%

Tabell 1. Batterikonsumtionstest

I handenheten fanns även en funktion där batterikonsumtionen för dess appar presenterades. Batterikonsumtion för applikationen beräknades där till 2% för de senaste 9 timmarna.

4.8. Testkörning i industriell miljö

När en prototyp tagits fram som fungerade i kontorsmiljö gjordes förberedelser för tester på systemet i industriell miljö, vilket var den miljö som systemet var tänkt att operera i.

Den robotcell som valts ut av företaget för installation programmerades för att kunna skicka larmsignaler till systemet. Elektriker installerade därefter en PLC som kopplades för att ta emot dessa larmsignaler via I/O. Routern positionerades på så vis att dess signaler kunde tas emot vid operatörens sekundära arbetsplats.

Initialt testades kommunikationssystemet genom att manuellt trigga de larmsignaler som roboten skickar vidare till PLC:n. Således kunde larmmeddelande mottagas i applikationen utan att något faktiskt larm från roboten löst ut. Detta var tidssparande då larm från roboten inträffar relativt sällan.

När systemets funktion hade bekräftats kunde ett test genomföras där operatören bar handenheten med parkopplad smartklocka under en arbetsvecka. Därmed kunde kommunikationens tillförlitlighet testas under en längre tid. Då testet gav ett positivt resultat kunde systemets funktion i industriell miljö bekräftas.

4.9. Problem och lösningar

Stora delar av examensarbetet har bestått av att finna lösningar på de problem som formulerades i den initiala beskrivningen men även de problem som uppkommit under examensarbetets gång.

På grund av IT-säkerhetsskäl kunde inte systemet testköras på företagets befintliga nätverk och av ekonomiska skäl kunde inte ett nytt trådlöst nätverk som möter företagets säkerhetskrav publiceras. Därför har prototypen som tagits fram i detta examensarbete endast testkörts på ett nätverk som byggts upp provisoriskt för detta ändamål.

4.10. Verktyg och hjälpmedel

De verktyg och hjälpmedel som använts under examensarbetet listas här.

Instrument

- Kalkylator
- PC (Windows)
- Smartphone

Mjukvara

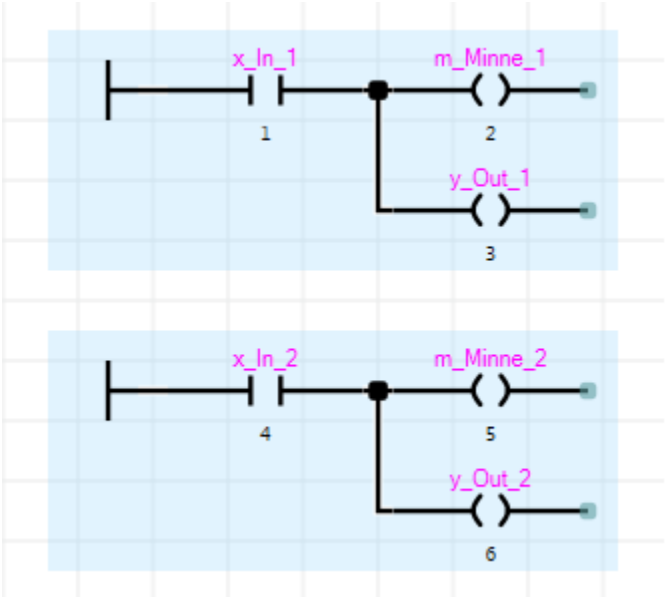
- Node-RED
- Mosquitto
- Android Studio
- Android emulator
- Notepad++
- Google docs

5. Resultat

I följande avsnitt kommer resultatet av det arbete som gjorts presenteras utifrån den prototyp som tagits fram.

5.1. Styrsystemet

En PLC av typen Mitsubishi Electric FX5U installerades i den robotcell som valts ut av företaget för testning av systemet. Med hjälp utav en åtkomstpunkt är PLC:n uppkopplad till det trådlösa nätverket. PLC:n används för att ta emot larmsignaler från robotcellen via PLC:ns I/O-modul. PLC:n är programmerad att sätta en specifik minnescell till *true* vid inkommande larmsignal.



Figur 8. PLC-program för inkommande larmsignal

Till testningen av prototypen i industriell miljö programmerades roboten att vidarebefordra följande tre larm till PLC:n.

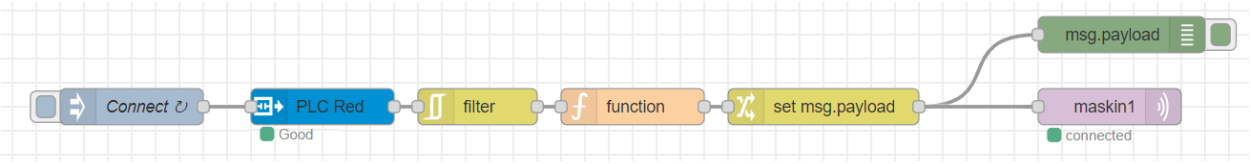
1. Magasinet för oprocessade artiklar är tomt.
2. Något i cellen har blivit fel eller gått sönder.
3. Magasinet för oprocessade artiklar är snart tomt.

5.2. Datorenhet

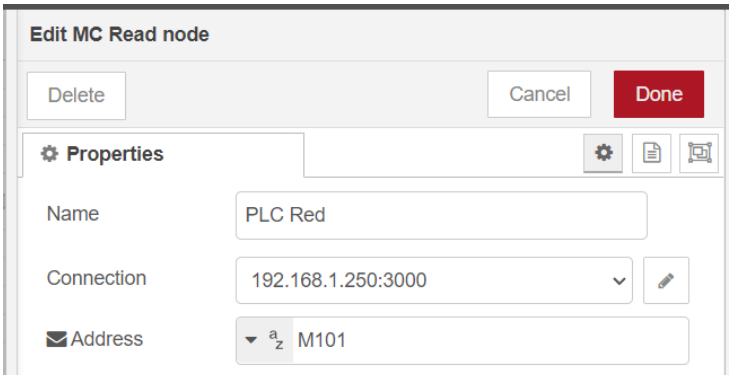
I datorenheten, som är uppkopplad till nätverket, finns Node-RED och Mosquitto installerade. Node-RED fungerar som den centrala motorn i systemet. Datorenheten kan i teorin hantera inflöden från obegränsat antal maskiner, och därför krävs det inte ytterligare datorenheter när nya maskiner kopplas upp till systemet.

De två flöden som skapades i programmet hämtar data från PLC:n genom *MC protocol*-noden. Denna data är antingen *false,* vilket betyder att allt är i sin ordning, eller *true* vilket betyder att ett larm sänts ut.

- *MC read*-noden skapar en anslutning till PLC:n. Varje sekund hämtas data från den minnescell i PLC:n som ställs till *true* när ett larm skickas ut.
- *Filter*-noden filtrerar bort alla upprepningar i indatan.
- *Function*-noden filtrerar bort alla *false*.
- *Set*-noden skapar det meddelande som ska skickas ut med MQTT-protokollet. T.ex. ”Maskinen har stannat!”.
- *Mqtt out*-noden kopplar upp sig till brokern (Mosquitto) och skickar iväg ett meddelande med angivet *topic* när ett larm har löst ut i robotcellen.



Figur 9. Flöde för hämtning, filtrering och sändning av data i Node-RED



Figur 10. Inställningar för mqtt out-noden i Node-RED

5.3. Android-applikation

Android-applikationen är utvecklad för att på ett enkelt vis upplysa operatören om att en robotcell behöver tillsyn. När något skett i en robotcell som kräver operatörens uppmärksamhet skickas ett meddelande till operatörens handenhet via MQTT-protokollet. Operatören meddelas genom att en notifikation skickas ut, vilket gör att det vibrerar och ringer i handenheten. Notifikationen visar ett meddelande som talar om vilken maskin som larmet gäller samt vilken typ av larm. Handenheten kan även parkopplas via bluetooth med en smartklocka och därmed blir meddelandena mer lättåtkomliga och svårare att missa. Larmmeddelandet loggas även i applikationen.

Vissa funktioner såsom *Inställningar* och *Redigera prenumeration* har medvetet gjorts svårare att komma åt. Detta för att användaren inte ska kunna redigera nödvändiga inställningar av misstag.

5.3.1. Hemskärmen

På hemskärmen visas de prenumerationer som lagts till av användaren. De kan aktiveras och inaktiveras med en switch. Det senast inkomna meddelandet visas under prenumerationens namn. Om rätt inställningar har sparats till brokern och denna befinner sig på samma nätverk som användaren sker en anslutning automatiskt. Annars visas *Disconnected* med röd text tills dess att användaren gör en lyckad anslutning.

Figur 11. Hemskärmen i applikationen

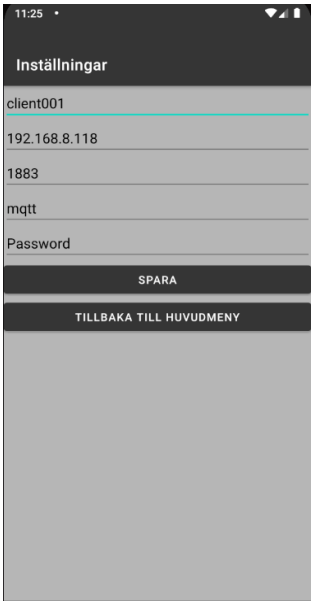
5.3.2. Lägg till prenumeration

Genom att fylla i namn och topic läggs en ny prenumeration till.

Figur 12. Lägg till ny maskin

5.3.3. Inställningar

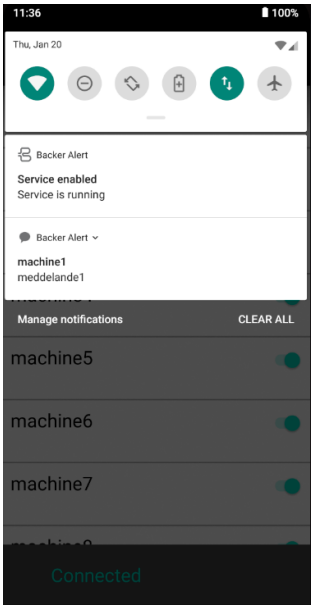
Här redigeras de nödvändiga inställningarna till brokern, såsom klient-ID, IP-adress, portnummer, samt användarnamn på broker och dess lösenord.



Figur 13. Inställningar

5.3.4. Notifikation

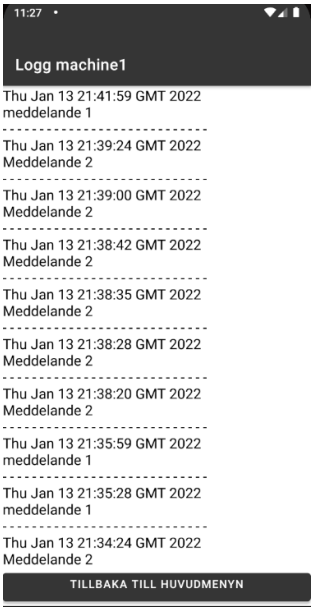
När applikationen körs visas en notifikation som talar om att *foreground servicen* är igång. När ett nytt meddelande anländer visas en notifikation i notifikationspanelen samt att handenheten vibrerar och ringer. Är dessutom handenheten parkopplad med en smartklocka visas notifikationen i denna, förutsatt att klockan stödjer notifikationer.



Figur 14. Notifikation

5.3.5. Logg

I loggen visas de 10 senast inkomna meddelandena där de senast inkomna sorteras överst.



Figur 15. Logg

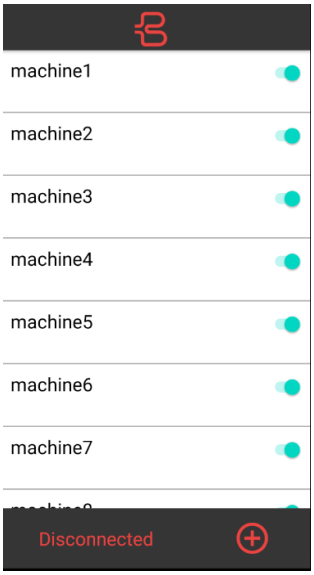
5.3.6. Redigera prenumeration

För att redigera en befintlig prenumeration görs ett långklick på dess namn på startskärmen. Här kan man sedan välja att redigera namn eller topic, radera prenumerationen eller radera alla prenumerationer.

Figur 16. Redigera prenumeration

5.3.7. Disconnected

Disconnected visas när uppkopplingen till brokern ännu inte initierats eller om uppkopplingen förlorats. Genom att lång-klicka på *Disconnected* visas Inställningar-skärmen och genom att klicka vanligt görs ett försök att ansluta till brokern.

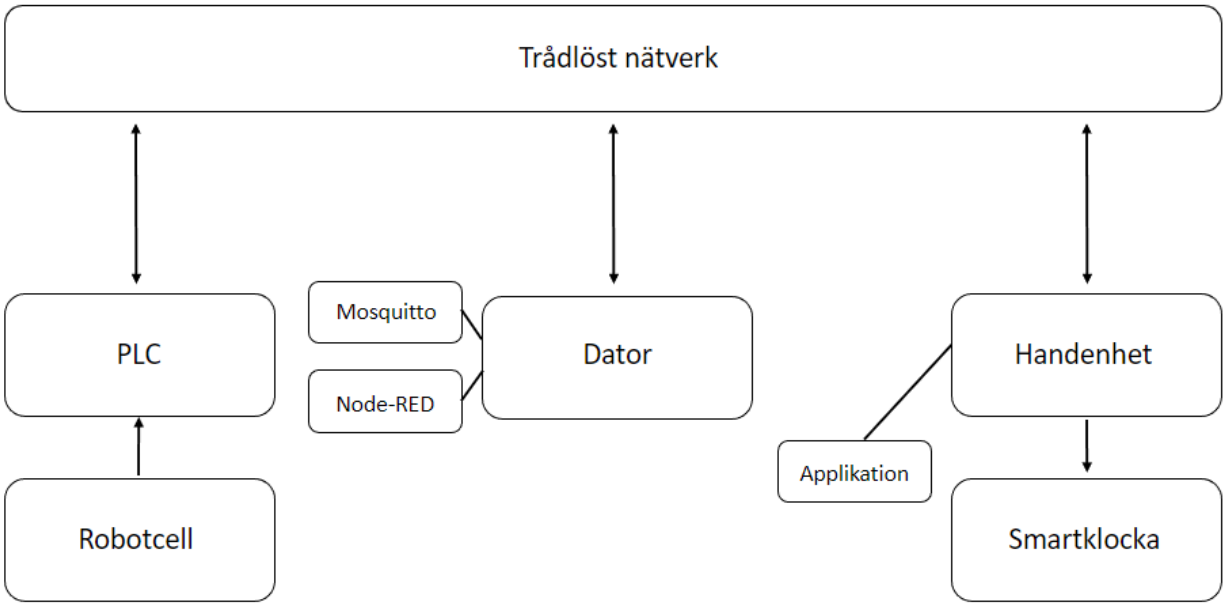


Figur 17. Disconnected

6. Slutsats

Examensarbetet har tagit fram en prototyplösning bestående av 3 delar.

1. En PLC som tar emot larmsignaler från en robotcell. PLC:n som är utrustad med ethernetport kopplas till en åtkomstpunkt som möjliggör trådlös uppkoppling till nätverket.
2. En datorenhet som hämtar, behandlar och skickar data.
3. En Android-applikation för mottagande och visning av data på handenhet. Handenheten kan parkopplas med en smartklocka för en lättåtkomlig visning av larmmeddelanden.



Figur 18. Slutgiltig modell av systemet

6.1. Frågor och svar

Följande frågor har besvarats utifrån prototypens lösning.

Vilken typ av kommunikation mellan robotcell och datorenhet är lämplig för ändamålet?

Med hjälp utav Node-RED ansluter sig datorenheten till den PLC som installerats i anslutning till robotcellen. Node-RED avläser de minnesceller i PLC:n som aktiveras när larm från roboten inträffar. Kommunikationen använder sig utav MELSEC-protokollet.

Ska operatören få information om produktionsstopp via en applikation, e-post eller på annat vis?
Operatören får en notifikation (vibration, ljudsignal, text) via en applikation i handenheten när produktionsstopp sker. Handenheten kan även parkopplas med en smartklocka.

Hur ska datorenheten skicka ett meddelande till en handenhet?
Datorenheten hämtar med hjälp av Node-RED data från den PLC som installerats i robotcellen och skickar vid behov ut ett felmeddelande till berörda operatörer via MQTT-protokollet när maskinen har larmat. Handenheten tar emot felmeddelandet via en applikation.

Om en smartphone-applikation ska användas, ska denna då programmeras eller ska en redan befintlig applikation användas?
För testning av systemet användes inledningsvis en redan befintlig applikation för mottagning av larm från robotcellen. Denna applikation mötte inte de krav på funktioner och användarvänlighet som ställts och därför har en skräddarsydd applikation för Android utvecklats.

6.2. Brister
De brister i systemet som upptäckts under utvecklingen av detsamma presenteras här.

- Vissa buggar i applikationen har upptäckts, däribland duplicerad loggning av meddelanden. Det har gjorts försök att åtgärda denna bugg men eftersom felet inte är konsekvent har arbetet försvårats. Buggen har bedömts att inte vara utav det allvarliga slaget och därmed har ytterligare tid inte lagts på att åtgärda problemet.

- På grund av tidsbrist har inte några tester beträffande IT-säkerheten kunnat utföras.

6.3. Systemets fördelar

De fördelar som har kunnat utrönas listas nedan.

- Systemet är inte beroende av externa servrar.
- Systemet är installerat på ett lokalt nätverk utan anslutning till internet och försvårar därför för dataintrång.
- PLC:n är kopplad via I/O till roboten och försvårar därför för dataintrång.
- Lättviktig kommunikation som inte belastar nätverket i någon större utsträckning.
- Systemet är modulärt och kan i teorin appliceras på majoriteten av företagets automatiserade maskiner.
- Inga löpande kostnader.
- Nya klienter kan läggas till med enkelhet.
- Systemet anses ha en relativt låg initialkostnad.
- Flexibelt och utbyggbart.

6.4. Reflektion över etiska aspekter

Om systemet fungerar optimalt innebär det ökad drifttid för automatiserade maskiner. Detta kan ur en etiskt aspekt vara olyckligt då det i vissa fall kan reducera behovet av mänsklig arbetskraft.

6.5. Framtida utvecklingsmöjligheter

Systemet har utvecklats för att med enkelhet kunna addera funktioner, både i Node-RED och i applikationen. Med tanke på att Node-RED är ett multifunktionellt verktyg som kan anpassas till många applikationer, kan detta system utvecklas brett.

För att underlätta felsökning kan en detaljerad larmlista skapas i robotcellen. Denna ger ingående information om var i robotcellen felet skett och vilka komponenter som är involverade. På så vis kan tiden för felsökningsprocessen reduceras.

I uppföljningssyfte kan en databas skapas och spara de larm som löst ut. Är larmlistan dessutom tillräckligt detaljerad kan statistik föras på larmorsakerna. Här kan sedan åtgärder sättas in för att lösa de problematiska momenten i maskinen.

7. Källhänvisning

[1] Mitsubishi Electrics FX5U, hämtad 2022-03-13
https://se.mitsubishielectric.com/fa/products/cnt/plc/plcf/cpu-module/fx5u-32mr-es.html#tab-bltfbdbcff971f9343a

[2] Mitsubishi Electrics GT2508-VTBD, hämtad 2022-03-13
https://www.mitsubishielectric.com/fa/products/faspec/detail.page?kisyu=/got&formNm=GT2K_GT25_GT2508-VTBD_87&lang=2&word=GOTs&category=ex&id=spec

[3] Handenhet, hämtad 2021-11-12
https://www.ute.com/eu/products/detail/EA510

[4] About: Node-RED, hämtad 2022-03-13
https://nodered.org/about

[5] Messages, hämtad 2022-01-14
https://nodered.org/docs/user-guide/messages

[6] Noder, hämtad 2021-12-21
https://nodered.org/docs/user-guide/nodes#template

[7] Filter-nod, hämtad 2022-12-21
https://flows.nodered.org/node/node-red-node-rbe

[8] MC protocol, hämtad 2021-12-21
https://flows.nodered.org/node/node-red-contrib-mcprotocol

[9] MQTT, hämtad 2022-01-14
https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.pdf

[10] Broker, hämtad 2021-12-03
https://mosquitto.org/

[11] Android Studio, hämtad 2021-12-03
https://developer.android.com/studio/intro

[12] Android Oreo, hämtad 2021-12-03
https://developer.android.com/about/versions/oreo/android-8.0-changes

[13] Foreground Service, hämtad 2022-01-04
https://developer.android.com/guide/components/foreground-services