Examensarbete Industry 4.0

LUNDS UNIVERSITET

Examinator: Mats Lilja
Handledare: Christian Nyberg
Handledare Rejlers: Sten Pettersson

Examensarbetare: Morris Ljung

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 2018

Sammanfattning

Detta examensarbete har i huvudsak undersökt begreppet Industry 4.0 och möjligheten att skicka data från en temperatursensor upp till en molntjänst, en process som är ett exempel på termen Internet of Things. Undersökningen har gjorts för konsultföretaget Rejlers för att ge VD samt teknisk chef en bredare uppfattning om hur de skulle kunna implementera IoT vid uppdrag för företag i framtiden. För att kunna visa en tydligare bild har ett pilotsystem skapats där data har hanterats och omvandlats i olika steg på vägen upp mot molntjänsten. Innan dessa steg implementerades gjordes en grundlig förstudie för att definiera ett tydligt tillvägagångssätt att lösa uppgiften på och jämföra de alternativ på hårdvara och mjukvara som fanns tillgängligt. Genom att följa en skiss som blev presenterad tillsammans med arbetet baserades strukturen på arbetet utifrån ett verkligt exempel som en eventuell kund skulle kunna efterfråga.

Från underlaget i förstudien startade den konkreta fasen där fysikaliska storheter skulle hanteras och omvandlas till digitala data. Bland de val som fanns till hands gjordes ytterligare utredning utöver förstudien då det krävs en praktisk förståelse för komponenterna för att kunna göra en tydlig bedömning som inte bara fungerar på papper utan även är möjlig att realisera.

Resultatet av examensarbetet är en pilotanläggning som kan användas för att på ett tydligare sätt förklara kommunikationskonceptet med Industry 4.0. Genom att binda ihop den tidigare kunskapen från "Industry 3.0" där givare kopplas ihop med PLC och lägga till en kommunikation till molntjänsten kan vi skapa en mer dynamisk analys. Möjligheten att komma åt data, nyckeltal och felmeddelande överallt där det finns internetanslutning öppnas vilket kan leda till snabbare beslut och ökad effektivisering. Pilotanläggningen har genom kommunikation och datahantering mätt en fysikalisk storhet med hjälp av en givare, omvandlat denna signal i en PLC, skickat den vidare till en gateway som i sin tur överfört denna data till en server uppe i en molntjänst.

Nyckelord

Industry 4.0, IoT, Internet of Things, MQTT, Molntjänst, Broker.

Abstract

This scientific report has mainly studied the concept of Industry 4.0 and the possibility to send data from a temperature sensor to the cloud, a process which is an example of the term Internet of Things. The study has been conducted for the consultant business Rejlers to give the CEO as well as the Technical Manager a broader view of how they could implement IoT on behalf of costumers in the future. In order to show a clearer picture a pilot system has been created where data has been handled and transformed into different steps on its way to the cloud. Before these steps where implemented a thorough study to define a clear approach to solve the assignment on and compare the hardware options and the software options available was made. By following a sketch that was presented together with the assignment the structure of the report was based on a true example as a potential costumer would ask for.

From the background of the preliminary study started the concrete phase where physical quantities should be handled and converted into digital data. Among those choices at hand further investigation was made in addition to the pre-study since a practical understanding for the components is required to make a clear decision that does not just work on paper but is possible to realize.

The result of the thesis is a pilot system that can be used in order to explain in a clearer way the communication concept of Industry 4.0. By combining the previous knowledge from “Industry 3.0” where sensors are connected with a PLC and add a communication to the cloud we can create a more dynamic analysis. The ability to access data, key figures and error messages wherever there is internet connection opens up which can lead to faster decisions and increased efficiency. The pilot system has through communication and data management measured a physical quantity with a sensor, converted this signal within a PLC, sent it to a gateway which in turn transferred this data to a server in the cloud.

Keywords

Industry 4.0, IoT, Internet of Things, MQTT, Cloud service, Broker.

Innehållsförteckning

Förord

Jag vill tacka Magnus Lindau för möjligheten att få skriva mitt examensarbete hos Rejlers. Jag vill även tacka mina handledare Sten Pettersson och Christian Nyberg samt ingenjörerna Daniel Kjellberg och Ola Attlehed för den hjälp och vägledning jag har fått under arbetets gång. Från arbetet har jag fått en klarare bild av hur en framtida arbetsplats kan fungera och de områden jag finner intressanta inom ingenjörsyrket har förtydligats.

1 Inledning

1.1 Bakgrund

Detta examensarbete bygger på ett scenario som kunde varit hämtat ur Rejlers konsulters normala arbetsvardag. Rejlers AB är ett konsultföretag med ca 2 000 experter anställda som arbetar med projekt och IT-lösningar inom bygg och fastighet, energi, industri och infrastruktur. Där finns flera specialiserade ingenjörer och de fortsätter att växa snabbt och finns idag på 80 orter runt om i Sverige, Finland och Norge.

Ute i industri och produktion runt om i världen diskuteras ofta begreppet Industry 4.0. Det handlar om den uppkopplade industrin. Allt fler industriprodukter har någon form av Ethernet-anslutning och allt fler fältbussar blir Ethernetbaserade. Molntjänster blir allt vanligare och det pratas om allt som kan göras i en molntjänst. Men vad är egentligen Industry 4.0 och är verkligen allt uppkopplat bara för att styrsystem har frekvensomformare med Ethernetuttag? Svaret är enkelt – Nej!
Detta examensarbete har arbetat med att fram ett exempel och en form av väg man kan gå för att komma in i Industry 4.0. En pilotanläggning togs fram där man kunde skapa en uppkopplad produktionsanläggning. Från att endast ha automatiserade processer (Industry 3.0) som styrs av en PLC och måste styras på plats, till att allting kan kommunicera via internet och bli mer lättillgängligt online, vilket gör att vi kan förutspå kommande problem, är ett stort steg i rätt riktning mot effektivisering av industrin.

Examensarbetet undersöker möjligheten med ett realtidsuppdaterat system vilket är ett nytt arbetssätt som introducerats de senaste åren. Då det inte är en standard med onlinebaserade system än finns det få andra arbeten inom området. Att lagra data och analysera den på en server online växer fram mer och mer, främst hos de stora företagen som helst håller sin utveckling hemlig, vilket gör att den offentliga informationen inom ämnet är begränsad. Vid sökning i Libris [1] hittades inget arbete som liknar detta.

1.2 Syfte

De flesta kunder har ett behov av att veta mer om sin egen produktion. Man vill kunna få information från produktionen mer eller mindre i realtid och man vill kunna få analyser och nyckeltal även om man inte ens är i närheten av produktionen. Man vill med detta kunna få stöd för beslut, investeringar och avvecklingar. Bara för att molntjänster finns innebär det inte att kunden vet mer om sin anläggning och bara för att kundens anläggning kopplas mot nätverket betyder inte det att data vare sig blir tillgänglig eller ens användbar. Syftet med detta arbete är att skapa ett exempel på hur Industry 4.0 fungerar, framförallt för att visa möjligheterna med molntjänster, men även för att skapa ett exempel som kan användas för att illustrera vart man behöver starta och vilka steg man behöver gå igenom för att skicka upp sensordata till en molntjänst För många är vägen mellan PLC och moln diffus, men den pilotanläggning som skapats visar en mer detaljerad bild och belyser de aspekter som utgör grunden för en sådan kommunikation. Detta för att snabbt kunna ge någon med mindre kunskaper inom området en tydlig bild av hur omfattande det är, vilka komponenter som krävs och hur kommunikationen mellan de olika delarna hänger samman.

1.3 Mål

Målet med detta examensarbete har varit att skapa ett system som kan hämta information från ett temperatursystem, skicka upp denna informationen till en molntjänst för att sedan hämta ner denna informationen på en ”Dashboard” som kan presentera användbar information, såsom nyckeltal, larm och orsaker i realtid för VD eller teknisk chef.

1.4 Problemformulering

Nedan finns de problemformuleringar som ansetts vara de mest relevanta.

- Vad krävs för att kunna skicka upp och spara data i en molntjänst?
- Vilket typ av protokoll bör användas för att skicka data från en Raspberry Pi till en molntjänst?
- Vilka typer av komponenter behövs och går dessa att byta ut?
- Hur kopplas en ”dashboard” ihop med en molntjänst för att få ut

data i realtid?

1.5 Motivering av examensarbetet

Att kunna kommunicera med ett företags hela anläggning kommer ligga till grund för framtiden inom automation och därför är uppdraget att undersöka IoT väldigt relevant. Examensarbetet valdes framförallt för att studenten ska kunna få en ny inblick i hur framtiden kommer att formas. Rejlers konsulter har själva inte tid att göra denna undersökning, men tycker att det är ett spännande och utvecklande område. Genom detta examensarbete kommer företaget, studenten och samhället kunna förstå fördelarna med att kunna se en produktion i realtid och med hjälp av analys förhindra problem redan innan de uppstår. För samhället i stort kan detta leda till effektivare produktion med mindre materialspill, mindre slit för de anställda med att reparera maskiner som kunde förhindrats genom att förutspå vilka delar som börjar bli slitna. Samhället skulle även kunna dra nytta av utvecklingen då industrierna som producerar material för att bygga upp vägar, hus etc. kan arbeta mer effektivt då feedback kan nå utvecklare tidigare vilket t.ex. förkortar tid för nya prototyper.

1.6 Avgränsningar

Detta examensarbete har avgränsat sig till maximalt 2 stycken processorenheter såsom Arduino, Raspberry Pi, PLC för att samla in data från givare och sedan skicka vidare denna till en molntjänst. En undersökning har gjorts för att bestämma vilka som är mest passande. För att skicka upp data till molntjänsten kommer endast ett protokoll att användas, ett MQTT-protokoll. Av de många molntjänster som finns att välja mellan valdes endast en för att användas i pilotanläggningen.

1.7 Resurser

Examensarbetet krävde hårdvara i form av PLC/Arduino för att samla in data vilket tillhandahölls av Rejlers. Mjukvara som behövdes tillhandahölls av Rejlers och en del övriga program för att bygga en Dashboard, sätta upp kommunikation, testa olika funktioner samt skicka och ta emot data från en molntjänst fanns tillgängligt att ladda ner från utvecklare. Tillgång till arbetsplats, hårdvara samt kunniga handledare fanns tillgängligt hos Rejlers i Malmö. Intervjuer med

samarbetspartners till Rejlers genomfördes för att få mer information om komponenter som kan användas inom IoT.

2 Teknisk bakgrund

2.1 Mqtt

Mqtt står för Message Queue Telemetry Transport och är ett lättviktsprotokoll som passar bra för enheter där låg energiförbrukning är önskvärd. [2] Genom att hålla protokollet enkelt passar det bra för enheter inom Internet of Things som inte har mycket processorkraft. Mqtt skickar paket med väldigt liten storlek vilket gör att det går snabbt att skicka ett paket. För att kunna skicka och ta emot Mqtt-paket med data mellan klient och server behövs en mellanhand, en digital brevbärare, som kallas för Broker. Det är en programvara som ställs in att lyssna på en digital port som sänder över internet och sedan skickar vidare den inkommande informationen till rätt destination. [3] Mqtt-paket kan skickas väldigt enkelt och snabbt utan större säkerhet via port 1883, men kan även skicka informationen på port 8883 över TLS/SSL om mer säkerhet önskas. I det sistnämnda krävs autentisering med användarnamn och lösenord eller giltiga certifikat. [4]

2.2 Broker

En broker kan vara en programvara som installeras på en dator, virtuell maskin eller liknande maskin. Det kan även vara program som finns förinstallerade på en enhet som i fallet med gatewayen från Weintek. Det finns även färdiga bibliotek till Arduino som tillåter användaren att skicka via Mqtt genom att endast lägga till biblioteket. [5]

Sändaren, en så kallad "Publisher" skickar ett meddelande som innehåller t.ex. data till en mottagare i form av DNS-adress eller IP-adress. Meddelandet märks med ett "Topic" som kan ses som en adress på en postlåda. Topic används för att beskriva vad meddelandet innehåller och kan anges med t.ex. Vardagsrum/Temperatur/Givare1 och beskriver då Givare 1 som mäter temperaturen i vardagsrummet. Ett topic delas upp i olika nivåer med "/" där den första nivån är den högsta. Det räcker att ha en nivå, men man kan preciseras tydligare med flera nivåer om det finns mycket data att hålla reda på. [6]

Mottagaren, en "Subscriber", prenumererar på ett Topic och mottager den information som är kopplad till det. Sändaren kan välja vilken

information denne vill ha genom att öppna ("subscribe") brevlådan och ta de brev som denne är intresserad av. Genom att subscribe på "Vardagsrum/#" så tas all data från vardagsrummet emot. # är ett tecken som betyder "wild card" och det innebär att alla topics på den nivån som finns tillgängliga och sänder data hämtas. [4] Det är särskilt effektivt om man vill hämta all data som rör temperatur i ett hus eller analysera all data som kommer från vardagsrummet.

En Mqtt-klient är en användare eller enhet som prenumererar eller sänder meddelanden (Se Subscribe/Publish). En klient kan både skicka och ta emot data på samma gång. Den kan t.ex. skicka sensordata och ta emot feedback. Klienten kan vara allt från en liten mikrokontroller som Ardiuno till en fullskalig server som Microsoft Azure. Kravet för att räknas som en Mqtt-klient är att den kan hantera TCP/IP och använder Mqtt. Mqtt-bibliotek finns i de flesta programspråk vilket gör att de flesta enheter kan bli klienter. [7]

2.3 Dashboard

En dashboard är ett grafiskt gränssnitt där data kan hämtas och visas visuellt. Här kan rapporter samlas och data visas på ett lättöverskådligt sätt för användaren. Det kan t.ex. vara grafer, figurer eller text som beskriver och visar hur temperaturen förändrats över tid. En dashboard kan designas precis hur man vill och kan anpassas med olika sidor för att visa information på olika sätt. En startsida kan visa en lättöverskådlig bild på de allra viktigaste nyckeltalen i t.ex. en fabrik, genom att klicka sig vidare kan en tekniker se rapporter och information som rör larm och fel i anläggningen medan en chef kan granska viktiga siffror på en annan sida i dashboarden. En dashboard kan designas online och vara tillgänglig genom att logga in på ett konto online eller laddas ner som en app för att enkelt komma åt informationen på mobilen. Den kan kopplas till en molntjänst med hjälp av säkerhetsnycklar för att få högre säkerhet och realtidsuppdatering av data som kommer in till molntjänsten. [7]

2.4 Molntjänst

En molntjänst är ett sätt att hyra plats på en server som någon annan tillhandahåller och är åtkomlig via internet. I synnerhet funktioner

som traditionellt sätt sköts på den egna datorn kan skötas av någon annan i molntjänsten. För IoT är det främst serverprogram samt lagring av data som är intressant. Molntjänsterna som använts i examensarbetet omfattas av programvara på nätet (SaaS, Software as a Service). Med en molntjänst behöver inte användaren ha en kraftfull dator då programvara körs på servern. Underhåll kräver mindre eftertanke då företaget som tillhandahåller molntjänsten sköter mycket av detta åt användaren. Även säkerhetskopiering görs av servern vilket ger en extra säkerhet för användaren. Några exempel på molntjänster är Dropbox, Gmail och Soundcloud. [8]

2.5 PLC

De två PLC-enheter som använts i pilotanläggningen har båda haft sammankopplingsmöjlighet via Ethernet. Detta lämpar sig väl för IoT då data önskas skickas över nätverk. M221 TM221CE16T är modellnamnet på den PLC från Schneider som användes. Denna enhet har 2st analoga ingångar som tar emot signaler på 0-10V. Den har en anslutningskontakt av typen RJ45 för att kommunicera med en dator eller för att skicka vidare data via Ethernet. Kommunikation med Modbus TCP stöds på M221:an vilket är kompatibelt med gatewayen från Weintek. För att programmera denna PLC krävs Schneiders egna program SoMachine Basic som finns tillgängligt att ladda ner. [9] Siemens S7-1500 är modellbeteckningen på den andra PLCn som användes. Denna var förprogrammerad i labbsalen och data hämtades via Ethernet med Modbus TCP.

2.6 Virtuell maskin

En virtuell maskin/dator även förkortat VM från engelskans "Virtual machine" är en datorfil, en avbildning som fungerar som en fysisk dator. På den installeras ett operativsystem och det fungerar aom om operativsystemet var installerat på datorn. Flera VM kan köras samtidigt på en fysisk dator och körs i varsitt fönster som vanliga program. En VM sägs vara avskärmad och program som installeras eller körs här kan ej ta sig ut eller infektera den fysiska datorn med virus. Det passar bra i utvecklingssyften av program, operativsystem etc. som saknar implementerad säkerhet eller löper risk att manipulera delar av en fysisk dator. Den VM har virtuellt minne, processorer

o.s.v. vilket gör att flera VM kan köras på en fysisk dator eller server för att spara pengar på hårdvara eller energi i form av kylning. [10]

2.7 Modbus TCP/IP

Modbus TCP/IP är som ett vanligt modbus-protokoll som körs i en internetmiljö över TCP/IP-protokollet. Det passar bäst för PLC-enheter som kopplas via Ethernet. Det räknas som ett standardprotokoll inom automation och är väl implementerat. [11] Protokollet publicerades av nuvarande Schneider Electrics och används som förväntat på den PLC från Schneider som använts i pilotanläggningen. Protokollet ansluter över port 502. [12]

2.8 Gateway

Enheten cMT-G01 från Weintek som använts i pilotanläggningen är en gateway med möjlighet att behandla data som ett HMI. En gateway tar emot data från enheter, behandlar denna och omdirigerar den vidare dit användaren önskar. Med Ethernet kopplas t.ex. en PLC till gateway som tar emot informationen och med hjälp av programvaran EasyBuilder Pro behandlas denna data och skickas sedan vidare med Mqtt till en server. Port, destinationsadress, taggar o.s.v. konfigureras i programvaran som sedan laddas ner till en gateway. Enheten kan ta emot, simulera och skicka vidare data. Den kan skicka information direkt till ett Scada-system och agerar som en samlingspunkt för data från PLC-enheter. En gateway har egenskaper som ett HMI t.ex. spara händelser, logga data. TLS/SSL stöds vilket ger säkra anslutningar. [13]

3 Metod

3.1 Planering

Det första momentet som la grunden för examensarbetet bestod av att planera, undersöka samt söka information om ämnet Industry 4.0 för att kunna få en bild av tillgången på information samt att veta i vilken ordning de olika delarna skulle utföras. Handledaren Sten Pettersson skapade en skiss att utgå från som innehöll de grundläggande delarna i kedjan från givare till dashboard. Bred sökning på internet gjordes på de nyckelord som var relevanta för att sedan kunna smalna av sökningen när en helhet fanns. Det krävdes en hel del informationsinhämtning för att kunna uppskatta tiden det skulle ta att lära sig de olika delarna och från det skapades ett Gant-schema för arbetet. Det var först efter att allting var strukturerat och ett sammanhang var skapat som det andra momentet, förstudien, kunde påbörjas.

3.2 Förstudie

En grundanalys på vilka protokoll och komponenter som kunde användas gjordes främst genom informationssökning på internet, tester och genom samtal med samarbetspartners samt kunniga ingenjörer på kontoret. Då arbetsområdet är nytt och inte spridit sig i särskilt stor utsträckning fanns det väldigt begränsat med information att få fram. Varje del av arbetet fick delas upp i små delar som undersöktes för sig för att sedan försöka kunna kopplas ihop med varandra när tillräckligt mycket information var insamlad. Mycket av arbetet består av praktiska delar och nästan obefintlig information fanns att hämta från publicerad litteratur. Handledaren Sten Pettersson tillhandahöll en skiss med de delar som arbetet i stort skulle bestå av. En bred grundsökning för att hämta information och förstå dessa små delar gjordes där tips och guidning kunde fås från framförallt forum där andra med tekniskt intresse testat liknande tillvägagångssätt. Dessa källor kunde självklart inte användas som någon faktabakgrund, men mycket av dessa tips och idéer kunde realiseras praktiskt vilket bekräftade att det fungerade. För att bestämma vilken hårdvara som skulle användas gjordes framförallt en jämförelse mellan dessa sinsemellan för att se vad som var lämpat med tanke på tidigare lärd kunskap från utbildningen och hur tillgången på övriga komponenter

såg ut.

För att avgöra vilken programvara som skulle användas gjordes en grundlig jämförelse mellan de olika alternativen där användarvänlighet samt stabilitet prioriterades högst. Genom sökning på internet framkom det även att några få huvudaktörer användes mer flitigt av andra tekniker vilket ledde till att mer information om hur de programmen skulle användas fanns tillgängligt.

Vid valet av molntjänst stod valet av någon av de tre mest omtalade och mest utvecklade företagen IBM Cloud, Microsoft Azure eller Amazon AWS detta för att vara säker på att säkerhet och support var tillräckligt genomarbetad. Denna jämförelse gjordes framförallt genom att jämföra funktioner, programspråk, priser, datamängder, kompabilitet med tidigare valda programvaror, säkerhet, support etc.

3.3 Programmering

Arbetet fortskred med programmering och testning för att kunna skicka upp data till molntjänsten. I detta momentet lades även säkerheten i fokus för att senare slippa intrång i systemet. För att kunna programmera hårdvaran behövde först de valda komponenterna kopplas ihop. För mikroprocessorn fanns det mycket tidigare projekt att få inspiration från på diverse forum och med kunskap från tidigare Arduinoprojekt kopplades en lösning ihop. [14][15][16]
Program som hämtade värden från en givare utvecklades fortlöpande för att beskriva processen på ett tydligare och mer professionellt sätt.

Det följande steget var att programmera PLCn och få den att kommunicera med en IoT-Gatewayen som skickade vidare data. Programmeringen gjordes med ladderprogrammering i Schneiders egen programvara SoMachine Basic. Det fanns inte mycket information att hitta på internet om hur programmeringen skulle utföras. Datablad studerades, men för att riktigt kunna förstå hur koden skulle skrivas hjälpte teknikern Sten Petterson från Rejlers till med att förklara vilka konverteringar av signaler som behövde utföras samt hur detta skulle realiseras i programmet. När det kom till exporten av taggarna kontaktades Schneiders telefonsupport som

hjälpte till att lösa problemet.

Kopplingen mellan PLC och molntjänsten sköttes med hjälp av ett HMI som kan ses som en gateway. Enheten från Weintek räknas som en "IoT device" just för att den har Ethernetuppkoppling och möjlighet att skicka data med Mqtt-protokollet. Gatewayen som användes konfigurerades via en inbyggd server och ställdes in genom att följa datablad. PLCn kopplades samman till denna med Ethernetkablar och konfigurerades i programvaran som fanns att ladda ner på tillverkarens hemsida.

3.4 Kommunikation

Brokern Mosquitto som valdes ut i förundersökningen laddades ner och försök gjordes att installera den på arbetsdatorn. Detta alternativ var mycket bättre anpassat för Windows och installerades utan problem. På den virtuella maskinen där Linux var installerat kunde Mosquitto installeras som tänkt. När den virtuella maskinen skapats och Linux (Ubuntu) installerats färdigt kunde brokern laddas ner.

I fasen när data skulle skickas upp till molntjänsten provades det först att göra detta via Arduinon. Exempel på hur tidigare personer hade skickat data via Mqtt studerades och testprogram skrevs. [17] Först testades att skicka data direkt till IoT Huben, men då detta efter en del efterforskning visade sig vara svårt p.g.a säkerhetsskäl riktades arbetet om till att skicka data direkt till brokern på den virtuella maskinen för att lättare kunna analysera problemet. Då det var svårt att se om data kom fram till brokern i Azure gjordes försök att istället skicka data till en testserver för att säkerhetsställa att kommunikationen fungerade. Detta moment var ganska tidskrävande då det inte fungerade som det skulle och det fanns en mängd faktorer som spelade in. De olika felkällorna fick utvärderas en efter en för att kunna eliminera problemen.
Brokern på Linuxservern som låg utanför det lokala nätverket började att undersökas för att möjliggöra kommunikation mellan brokern och IoT-Huben. Den installerades med kommandon i seriekonsolen och testmeddelande skickades samt hämtades till en testserver för att se till att den fungerade korrekt. När kommunikationen till testservern

fungerade försöktes det istället skickas testmeddelande till IoT Huben vilket misslyckades. Alternativet att skicka data via port 8883 med TLS/SSL började implementeras.

För att komma vidare i arbetet startades fasen där data skulle skickas från gateway till den virtuella maskinen på Azureservern. I EasyBuilder valdes inställningar för MQTT alternativet, där IP-adress, taggar och port valdes.

De olika problemen försökte lösas ett efter ett i laborationsrummet på Rejlers kontor, men kommunikationsproblemet kvarstod. Supporten kontaktades angående nätverkets säkerhet, men svar uteblev. För att lösa nätverksproblemet togs mikroprocessorn hem för att testas på ett hemnätverk utan begränsningar i brandväggen och kommunikationen mellan Arduino och testserver fungerade. Efter telefonkontakt med supporten konstaterades att port 1883 var stängd i Rejlers nätverk p.g.a säkerhetsskäl vilket innebar att testning ej kunde fortsätta på plats.

3.5 Visualisering

När allting fungerade som tänkt med att lagra data skulle en dashboard skapas för att sedan kunna hämta hem data från molntjänsten och visa den på ett vettigt och effektivt sätt. Då data aldrig kunde skickas upp till molntjänsten beslutades det att en konfiguration av en dashboard ej skulle ske.

3.6 Källkritik

De källor som använts i det här examensarbete har granskats och ansetts pålitliga i den mån att de valts att användas.

[1] är hämtad från kurshemsidan och bör vara granskad av institutionen och är därför tillförlitlig.

[2], [9] och [13] är hämtade från hårdvaruutvecklare. Det kan anses pålitliga eftersom att det förutsätts att de anger korrekt information om sina produkter, men för att vara säker har logisk rimlighetsbedömning gjorts och i de fall osäkerhet har funnits har någon av de ingenjörer

som jobbar på Rejlers rådfrågats.

[3], [5], [6], [7], [8], [10] och [17] är hämtade från produktutvecklares hemsidor. Det kan anses pålitliga eftersom att det förutsätts att de anger korrekt information om sina produkter och tjänster, men för att vara säker har logisk rimlighetsbedömning gjorts och i de fall osäkerhet har funnits har någon av de ingenjörer som jobbar på Rejlers rådfrågats.

[4], [11] och [12] har hämtats från hemsidor med elektronikexperter. De kan anses vara tillförlitliga då informationen stämmer med de som undervisats under utbildningen, kontrollerats av ingenjörer på Rejlers eller som jämförts med uppgifter från produkttillverkare för att se att informationen ej skiljer sig åt.

[14], [15] och [16] har hämtats från diverse elektronikforum som har beskrivit tillvägagångssätt. Eftersom att dessa tillvägagångssätt har testats praktiskt har de visats sig stämma vilket gör just de källorna pålitliga.

4 Analys

4.1 Planering

Under planeringsfasen skapades ett dokument där användbar information lagrades. Från skissen som Sten Pettersson skapat började internetsökning utgående från nyckelorden och de delar som ansågs viktiga kopierades till dokumentet för att kunna sammanställa en helhetsbild. De internetlänkar som innehöll information som kunde vara användbar sparades undan till förstudien. Diskussioner med handledare och andra ingenjörer gjordes även i detta steg för att få hjälp med att hitta en ände att börja i. Alla komponenter som fanns tillgängliga togs fram för att sedan kunna jämföras under förstudien.

4.2 Förstudie

Det fanns två möjliga mikroprocessorer som Rejlers erbjöd att välja mellan, en Arduino eller en Controllino Maxi. Då det fanns bättre information samt fler komponenter tillgängliga till den förstnämnda blev detta valet att använda. Den senare krävde ett spänningsaggregat på 24V vilket också gjorde det otympligare och mindre praktiskt att arbeta med. Nödvändiga bibliotek laddades hem och den hårdvaran som användes kopplades ihop. Ett "PubSubClient"-bibliotek användes som tillåter kommunikation med Mqtt-protokollet och detta konfigurerades för att fungera tillsammans med det lokala nätverket via Ethernet. Bland annat kontrollerades Arduinos IP adress och en ledig MAC-adress valdes.

4.2.1 Broker

Det fanns en del olika Brokers att välja på, men efter sökningar på internet hittades tre stycken möjliga alternativ. De andra alternativen hade bristfällig information och kändes inte pålitliga nog. Det första alternativet var Mosquitto, ett gratis "open source" program skapat av Eclipse som passar till C-programmering i väldigt små enheter såväl som på större servrar. Då Mosquitto är open source används det flitigt i pilottester bland hobbyprogramerare och det fanns en mängd info och tips att samla på sig på diverse forum.
Detta var det överlägset mest rekommenderade alternativet vilket underlättade vid eventuell felsökning. Anledningen till att en broker laddas ner till arbetsdatorn var för att kunna göra tester och se hur

långt kommunikationen nådde.
Denna broker skulle fungera både på Windowsmaskiner samt enheter som kör Linux. Mosquitto var dock väldigt omständligt att installera på windows då programmet först skulle installeras, men inte slutföras, sedan skulle det laddas ner några filer från olika hemsidor som skulle läggas in i speciella mappar innan installationen sedan gjordes om och slutfördes. Trots detta fungerade inte programmet som tänkt och fick bytas ut mot det andra alternativet från förundersökningen. På Linux var installationen enklare och med några få kommandon laddades det ner och fungerade direkt. När brokern installerats behövde programmet justeras för att passa till ändamålet, det krävdes bland annat att rätt portar öppnades. Brokern som placerades på VM hade som funktion att ta emot data från gatewayen och sedan skicka den vidare till IoT-Huben. Det andra brokeralternativet var HiveMQ som är gratis för ett visst antal uppkopplingar, men kostar om man vill använda det i större sammanhang med fler enheter. Då HiveMQ kan tjäna pengar på programmet läggs det även en del energi på att förbättra programvaran och gränssnittet samt användarvänligheten är bra. HiveMQ var enkelt att installera på Windows, men inte lika enkelt att arbeta med.
CloudMQTT som var det sista alternativet består i grunden av Mosquitto brokern med ett extra skal och har ett gratisalternativ att ladda ner för en begränsad mängd anslutningar och dataöverföring. Då det inte består av en självutvecklad broker utan bygger på en tidigare utvecklad programvara fanns det ingen anledning att inte använda grundbrokern istället.

4.2.2 PLC

Rejlers hade både en Mitsubishi FX5 samt en Schneider M221 PLC som kunde användas under examensarbetet. Båda dessa enheter hade Ethernetuttag och var lämpade att använda för att skicka sensordata vidare till den IoT-gateway som handledaren tillhandahöll. Båda PLC-enheterna programmerades i sina egna program. Temperaturgivaren som var tänkt att användas fungerade till båda enheterna vilket innebar att det inte fanns några större fördelar med att välja den ena före den andra. Då FX5:an även behövde användas till ett annat projekt valdes M221:an från Schneider Electric för att kunna ha möjlighet att arbeta med den när det behövdes samt för att slippa koppla om och flytta

givaren.
Schneiders programvara var tyvärr svår att använda och det fattades dessutom en del funktioner. Den var ganska enkelt uppbyggd, men samtidigt krävdes det detektivarbete för att hitta vissa funktioner som t.ex. tagghantering vilket kunde varit tydligare placerat. Efter ett samtal med deras telefonsupport konstaterades att även supporten tyckte att programvaran var dåligt uppbyggd.
Mitsubishis PLC programmerades i deras egna program, GX Works 3, vilket är ett mycket avancerat och komplett program. För att komma igång med programmet erbjöds en två dagars kurs av Mitsubishi vilken var väldigt lärorik och tydlig. För att kunna få så mycket kunskap som möjligt togs även chansen att gå på en endagsutbildning hos Beijer Electronics. Denna kurs var lärorik, men inte lika informativ och givande som Mitsubishis.

I valet av mikroprocessor erbjöds möjligheten att välja mellan en Arduino och en Controllino Maxi där det senare är mer lik en PLC, men som programmeras i samma miljö som Arduinon. Då Controllinon drivs med 12/24V jämfört med Arduinon som drivs av 5V via USB var det en bättre kontrast att välja en mer portabel enhet då en PLC som drevs via en spänningskub med 24V redan valts. Detta skulle möjliggöra testning av uppkopplingen på olika ställen även om det inte fanns möjlighet att ha en spänningsomvandlare med sig. Då det även fanns fler projekt inom Mqtt till Arduino att hämta inspiration ifrån bedömdes detta vara ett bättre val om problem skulle uppstå och hjälp behövdes. Även om båda enheterna programmeras i Arduinos egna programmeringsmiljö kan det ibland vara lättare att kunna följa en guide med likdana komponenter när man ska felsöka.

4.2.3 Gateway

För att skicka data från PLC upp till en molntjänst hade handledaren, Sten Pettersson, ordnat en HMI/Gateway från Weintek. Denna enhet har som huvudsyfte att ta emot information från andra enheter, behandla denna information och sedan skicka iväg den behandlade informationen via Mqtt till angiven IP-adress via angiven port. Enheten är liten och har en avancerad medföljande programvara. Den hade endast en ingång och en utgång via Ethernet vilket begränsar möjligheten att koppla ihop flera enheter på samma gång, den lämpar

sig därför bättre för arbetsmoment med färre enheter. Detta för att slippa koppla ihop flera switchar vilket kan göra eventuell felsökning svårare. Beijer samt Beckhoff har egna motsvarande enheter för samma syfte, men vid kursen hos Beijer berättades det att deras "Beijer Box 2" som enheten heter ej var lanserad än och det fanns ingen testenhet att låna. Vid telefonsamtal med Beckoff för att diskutera deras enhet "EK9160" framgick det att den ej heller var lanserad och deras intresse för att erbjuda en testversion var minimalt. En mailadress till deras tekniska support gavs för att kunna erbjuda mer information angående enheten, men detta mail besvarades inte. Den tekniska supporten hos Weintek kontaktades för frågor angående programvaran och ett utförligt svar mottogs kort därefter. Då en användbar gateway presenterades redan från början lades mindre tid på att jämföra de olika alternativen, men baserat på dokumentation, användarvänlighet och support var valet att använda Weinteks produkt bra.

4.2.4 Molntjänst

Bland de tre stora molntjänster som undersöktes var AWS den största, Azure nästföljande och IBM den minsta. [17] AWS startade 2006 och har därför utvecklats under längre tid än de andra vilket gjort att de kunnat bygga upp ett större förtroende hos användare. En annan fördel med att ha arbetat med molntjänster under lång tid är att de stött på de flesta problemen som kan tänkas uppstå. Azure startades 2010 och har också haft många år på sig att utveckla sin verksamhet. IBM har inte särskilt stor del av marknaden jämfört med de tidigare nämnda vilket gör att sannolikheten för att det ska hålla i längden är mindre. Ju mer pengar företaget har investerat desto större chans finns det att de anstränger sig mer för att behålla eller förbättra sin position på marknaden. Alla tre alternativen erbjuder gratisalternativ för test som är begränsade antingen på mängden data som du får skicka per dag eller antalet meddelanden som är möjligt att skicka per dag. Amazons molntjänst är mer anpassad för open source än vad Microsofts alternativ är. AWS är även det mest avancerade alternativet, men det kan även bli det dyraste vilket gör det bäst anpassat för stora företag. Microsofts alternativ fungerar som gissat väldigt bra tillsammans med Windows, men har på senaste tiden utvecklats för att passa bra med open source. De har även ett unikt samarbete tillsammans med Red

Hat vilket innebär att de anser det vara värt att satsa på Linux. En stor fördel med Microsoft är deras fokus på säkerhet vilket är viktigt för Rejlers som jobbar med andra kunders data och inte vill få något problem med att ha läckt ut information från sina kunder. Azure har bredare support för SDKs och programmeringsspråk. AWS stödjer C och NodeJS, men Azure tillhandahåller även .Net och Java vilket ger mer flexibla lösningar.

4.2.5 Dashboard

Det fanns inte särskilt många alternativ att välja på när det kom till dashboards, men de två alternativen som skulle kunna fungera till pilotanläggningen var thingsboard.io och Thinger.io. Båda alternativen är open source och har möjlighet att användas online. Det förstnämnda alternativet skulle ha en snabb koppling med Mqtt och även kunna laddas ner som ett program på datorn. Båda alternativen hade en version som var gratis upp till ett visst antal enheter. Thingsboard kändes som det mer utvecklade alternativet och det fanns även mer hjälp att få online än hos Thinger och det blev därför alternativet som var tänkt att användas.

4.3 Programmering

4.3.1 PLC

För att visa på ett mer verklighetsbaserat exempel som skulle kunna användas i en fabrik programmerades en Schneider "TM221ME16T" PLC som hade Ethernetuttag och därför var lämpad för att användas inom IoT. Till denna PLC kopplades en temperaturgivare in på en av de två tillgängliga analoga ingångarna. Dessa analoga signaler som gick från 0-10V skulle omvandlas till digitala signaler som kunde variera mellan –50 till 50 grader Celsius. För att göra detta fick de analoga signalerna först omvandlas till heltal, sedan omvandlades heltalet till ett flyttal, flyttalet delades sedan med en faktor 10 då det analoga värdet varierar mellan 0-1000 och det endast var intressant med en skala på 0-100, för att sedan kunna simulera rimliga värden minskades detta med 50 för att få ett intervall som gick mellan –50 grader till 50 grader. Detta utfördes med hjälp av ladderprogrammering i programvaran SoMachine Basic V1.6. PLC:n fick konfigureras i programmet genom att välja rätt modell på

enheten, uppdatera firmware, definiera taggar på rätt I/O, sätta en fast IP adress för Ethernetport 1 och sedan göra en simulering i programvaran för att se så att allting fungerade som det skulle.

Även en Siemens S7-1500 PLC programmerades för att kunna avgöra hur många grader axeln på en encoder snurrat. En ingenjör på Rejlers, Daniel Kjellberg, hade förprogrammerat denna PLC och definierat taggar samt adresser som kunde användas för att få upp värdena till en Gateway/HMI som skulle skicka data vidare till en molntjänst. Programmeringen hade gjorts i Siemens egna programvara "Totally Integrated Automation Portal" med ladderprogrammering.

4.3.2 Gateway

För att samla in data behövde rätt adresser konfigureras och anges i Weinteks egna programvara EasyBuilder Pro vilken laddades ner från deras hemsida. De PLC-enheter som skulle användas kopplades samman via Ethernet med en switch. I EasyBuilder skapades inställningar för varje PLC där IP-adress, port, protokoll, kommunikationsväg och PLC-typ skulle väljas. I varje PLC-inställning skulle sedan taggarna konfigureras, hos dessa skulle namn, typ (Bit/Word), adress och vilken typ av adress väljas. Dessa varierar beroende på vilken tillverkare det är, vilken sorts signal du försöker hämta samt vad signalen är tänkt att användas till. För att se att alla inställningar var korrekta användes en simulering i EasyBuilder som visade värdena om det var rätt. På så sätt kunde tester utföras för att förstå vilka delar som skulle anges på vilket sätt.

4.4 Problem

4.4.1 Förstudie

Den första svårigheten som uppenbarade sig var att hitta relevant information om ämnet i förundersökningen. Då Internet of Things ännu inte finns i någon större kommersiell skala och mestadels bedrivs i utvecklingssyfte finns det ytterst lite information om storskaliga IoT-system i drift tillgängligt. Detta medför att mycket av informationen är spekulationer och exempel som är anpassade för specifika system. Även en del av det arbete kring IoT som finns är gjort av företag som helst inte delar med sig av detaljer utan hellre erbjuder en färdig

lösning att köpa. För att kunna sätta ihop ett pilotsystem för att skicka upp till en molntjänst blir det därför nödvändigt att sätta sig in i små detaljer för att kunna se hur ett helt system kan se ut och hur de olika delarna kan samverka. Det som gör detta tidskrävande är framförallt de många spekulationer och antaganden som måste utvärderas, jämföras och testas för att se vad som egentligen stämmer och är praktiskt genomförbart.

4.4.2 PLC

En annan svårighet var programmeringen av PLC. SoMachine Basic V1.6 var ett program med bristfällig dokumentation, få färdiga exempel och i kombination med låg erfarenhet av PLC-programmering sedan tidigare var det svårt att komma igång. Som tur var kunde handledaren på Rejlers vägleda, förklara och visa tillvägagångssättet så att det fungerade som tänkt. När sedan taggarna för de olika variablerna skulle exporteras uppkom nästa problem. Det fanns ingen tydlig meny där exportalternativet fanns och handledaren hade inte arbetat tillräckligt mycket i programmet för att veta vart alternativet fanns. Supporten ringdes upp och de förklarade hur detta gjordes, men även supporten ansåg att programmet behövde utvecklas mer.

4.4.3 Broker

När programmeringen var klar tog det inte lång tid innan det första problemet med att installera brokern på arbetsdatorn upptäcktes. Installationen skulle först startas för att sedan avbrytas enligt instruktionerna. När installationen avbrutits och en mapp för programfilerna skapats skulle två separata filer från utomstående webbplatser hämtas och kopieras in i mappen som skapats. Installationen startades om och fullföljdes, trots att instruktionerna följdes fungerade det ej att starta programmet och en annan programvara fick installeras som fungerade korrekt.

4.4.4 Molntjänst

Brokern installerades på en virtuell maskin i den valda molntjänsten. Först behövde ett konto skapas i molntjänsten där en "Iot Hub" valdes och där all data ska samlas. Enheterna som anslöts valdes och tilldelades valda topics samt korrekta inställningar för adresser och

taggar. I molntjänsten behövdes även en Virtuell maskin, som kör Linux skapas som tillhandahåller brokern och är länken mellan att få in data och skicka den vidare till IoT-Huben. För att sätta upp en virtuell maskin i Azure krävdes det att en hel del val gjordes. Bland annat skulle en gemensam säkerhetsnyckel från IoT-Huben ställas in, det skulle väljas vilken region i världen som man önskde att lägga VM:en i, vilket operativsystem man önskade använda och sedan skulle man välja ett komponentalternativ på sin VM som man tror kunde passa ens behov. Det fanns en mängd olika alternativ att välja med olika RAM-minne, storlek på disken, antal kärnor etc. och där varje alternativ kostade en viss summa per månad.

Att skapa ett konto hos molntjänsten var ingen större svårighet mer än att ett kreditkort behövde registreras för att skapa kontot. Detta kreditkort tillhandahölls av Rejlers, men av förståeliga säkerhetsskäl tog det en längre tid vilket gjorde att arbetet med att starta molntjänsten försenades. Att skapa en IoT-Hub var enkelt, men att skapa den virtuella Linuxmaskinen var svårare. Användaren valde en av de färdiga VM som Azure erbjöd där antal kärnor, minne och storlek var fördefinierat. Dessa alternativ skiljde sig beroende på vilken region som önskades. Det "rekommenderade" alternativet i norra Europa kostade omkring 27500kr per månad vilket absolut inte accepterades. Ett rimligt alternativ på Azure hittades och valdes att installeras med Ubuntu, men installationen kunde ej slutföras och ett nytt alternativ fick väljas. Efter ytterligare sökning hittades ett nytt alternativ som var inom det accepterade prisintervallet och det installerades korrekt med Ubuntu.

4.4.5 Kommunikation

Det följande problemet uppkom när projektet i EasyBuiler Pro skulle konfigureras. Det var en del inställningar för PLC:n som var komplicerade att utföra utan tidigare erfarenhet. Handledaren Sten Pettersson hos Rejlers rådfrågades och hjälpte till att reda ut begreppen samt inställningarna.

När data sedan skulle skickas vidare med Mqtt till brokern misslyckades kommunikationen. Kommunikationen från Arduinon ut från det lokala nätverket kunde inte upprätthållas och det kunde bero

på fel på IP-adresser, mac-adresser, Ethernetmodul, lokala nätverket, kablar, brandvägg, programmeringskod m.m. Då det fanns många felkällor och flera av dessa behövde fungera samtidigt blev det svårt att felsöka. De mest grundläggande ändringarna började undersökas. Då detta inte hjälpte bryggades en annan dator in på nätverket mellan gateway och Azureservern. Utsignalerna analyserade med hjälp av Wireshark för att se om det fanns någon kommunikation ut från enheten. Broadcastmeddelanden kunde ses, men data från sensorerna kunde ej visas. Det visade sig att kommunikationen hade problem vilket ej kunde förklaras just då. Supporten för Rejlers nätverk kontaktades för att se om de kunde förklara problemet.

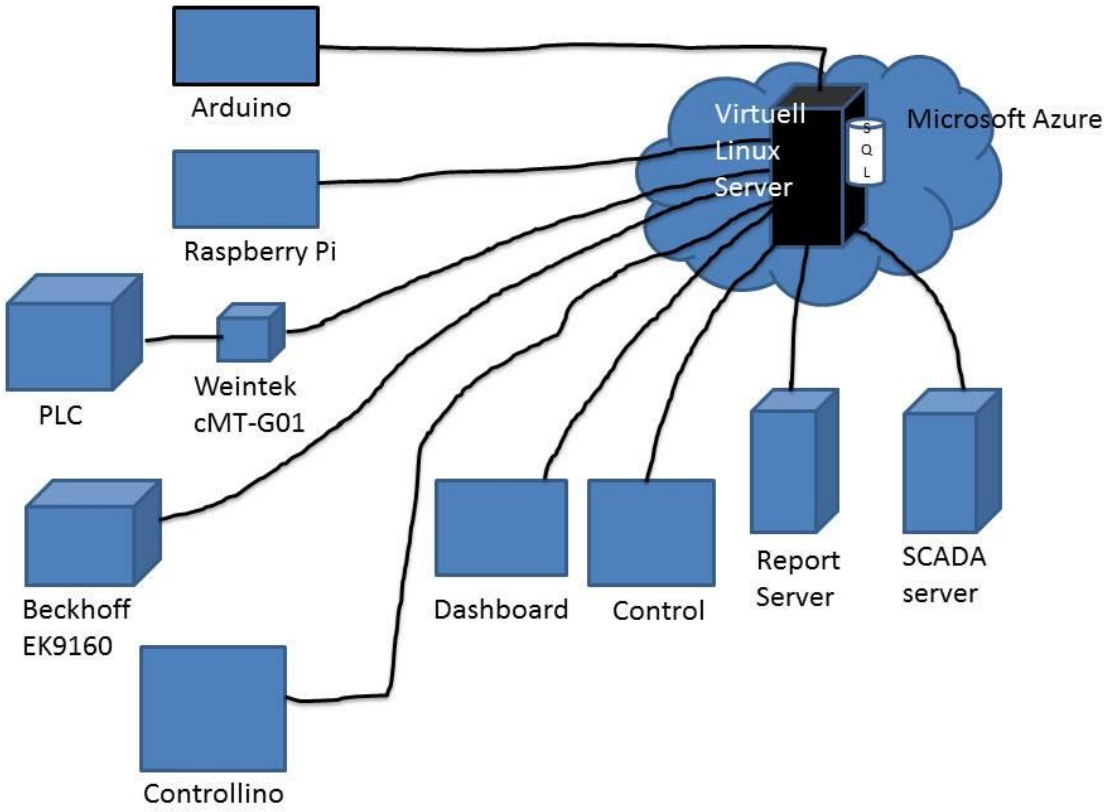
Under tiden supporten behandlade ärendet skickades testmeddelanden från broker till IoT-Hub på Azureservern. Dessa meddelanden kunde ej tas emot på den vanliga porten utan behövde skickas på den säkrare porten 8883. Efter ytterligare efterforskning visade det sig att ett certifikat mellan server och klient krävdes för att skapa förbindelsen. Ett sådant certifikat bör utfärdas av en CA vilket ställde till problem då varken tid eller pengar fanns att lägga på det. Lösningar för att komma runt problemet undersöktes, men de guider som behandlade ämnet var ej anpassade för en VM. Detta innebar att en fungerande lösning för certifikat ej kunde hittas. Efter diskussion med handledaren hos Rejlers bestämdes att data endast skulle försöka skickas upp till en VM och inte vidare till någon dashboard.

Då problemet med kommunikationen kunde ligga i nätverkets säkerhetsinställningar testades Arduinon i ett hemnätverk utan speciella säkerhetsåtgärder vilket bekräftade att problemet som tidigare varit svårt att sätta fingret på berodde på säkerhetsinställningar som begränsade vissa portar i företagsnätverket. Supporten hos Rejlers kontaktades via telefon och problemet konfirmerades. Ett nytt ärende som undersökte ifall port 8883 var låst lades upp. När supporten undersökt ärendet visade det sig att även port 8883 var låst vilket blockerar alla Mqtt-tester på företagsnätverket.

5 Resultat

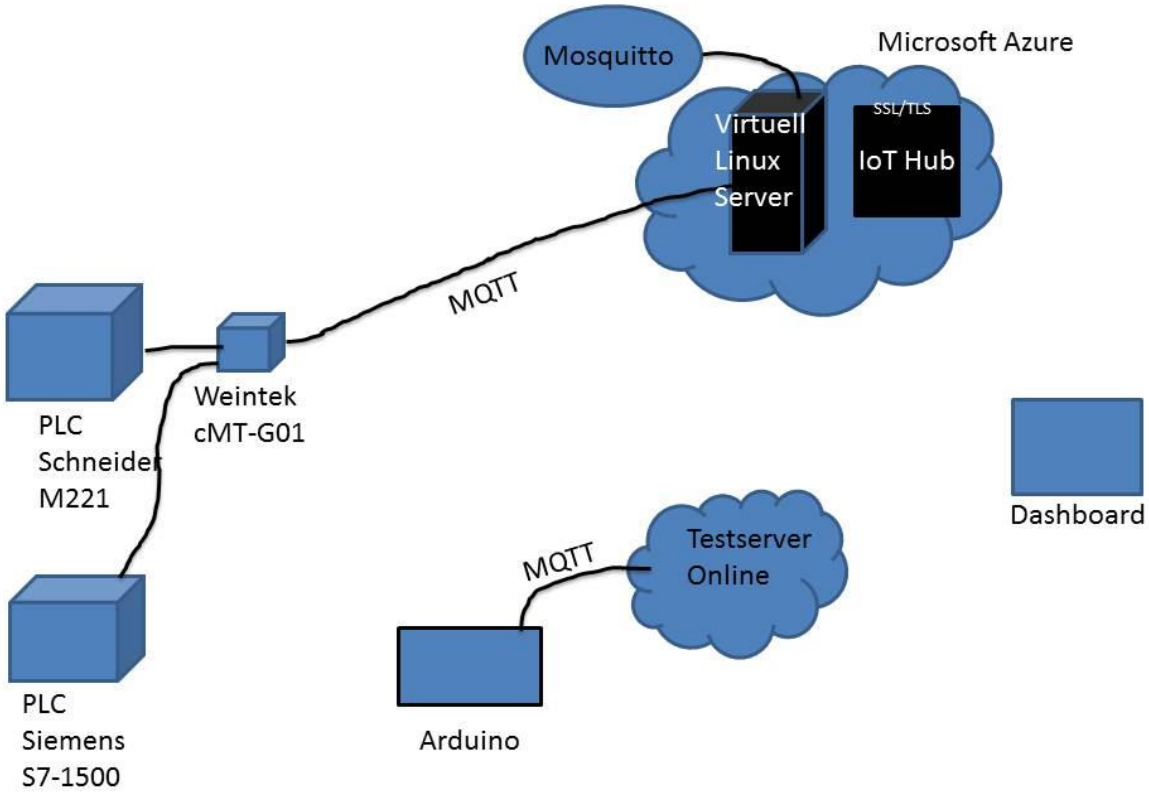
5.1 Slutgiltiga systemet

Vid presentationen av examensarbetet visade handledaren Sten Pettersson en skiss (figur 1) som visade en hur ett pilotsystem skulle kunna vara uppbyggt.



Figur 1: Första skiss över pilotsystemet från handledaren Sten Pettersson.

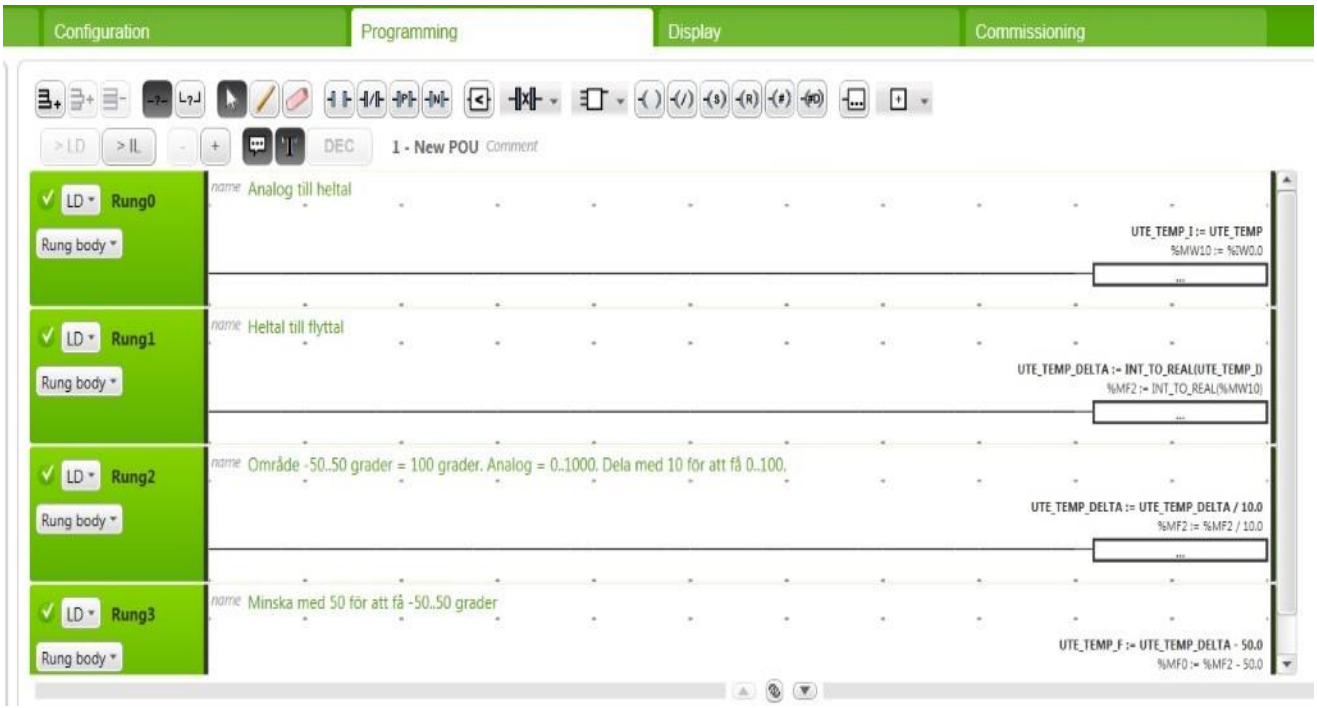
Figur 1 förklarade att någon form av mikroprocessor eller PLC skulle ge data som sedan skulle skickas till en molntjänst där en broker först tog emot informationen och sedan dirigerade denna vidare till servern som låg placerad online. Dessa värden från givare skulle sedan visualiseras på en dashboard och i mån av tid även placeras på en SCADA server.



Figur 2: Slutgiltiga pilotsystemet

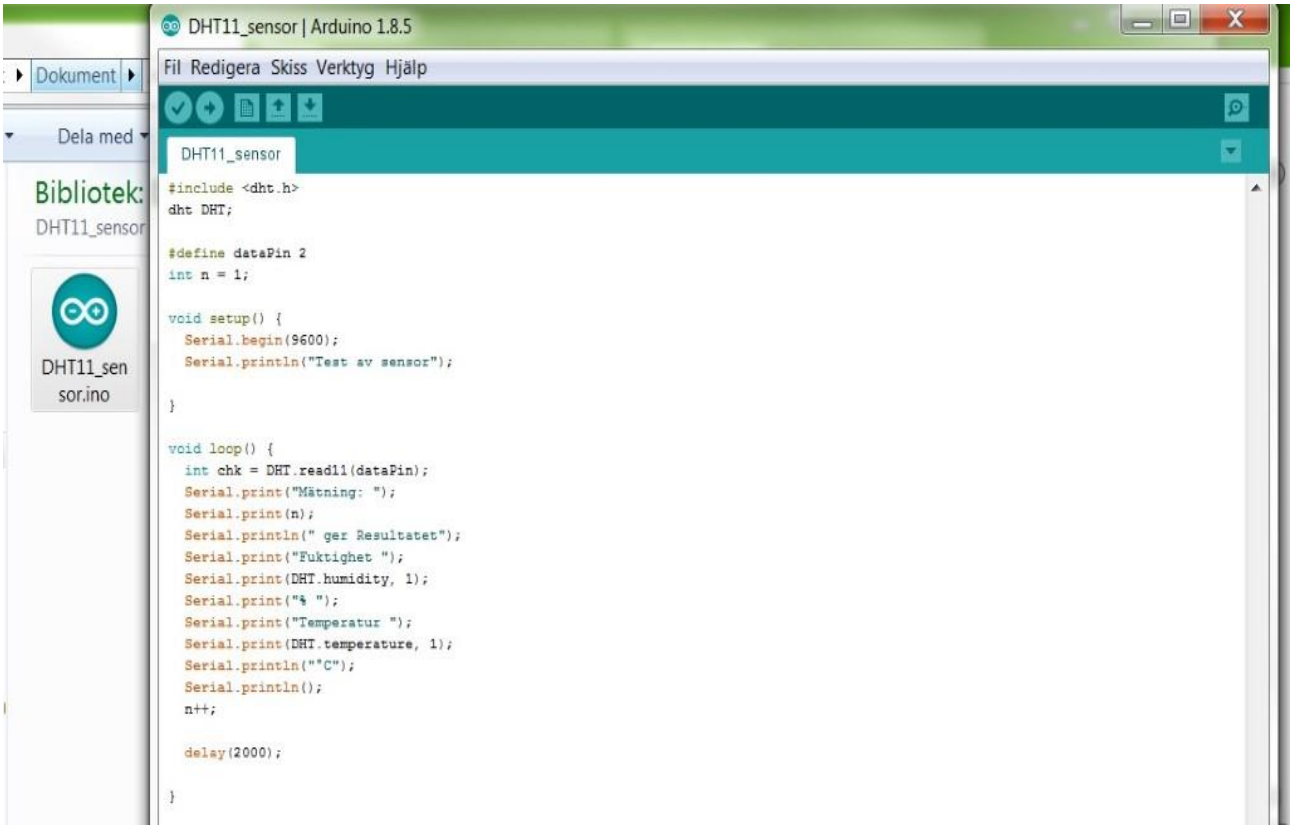
Figur 2 visar hur det slutgiltiga systemet såg ut. De två PLC-enheterna på vänster sida i figuren skickade sin sensordata till gatewayen via Modbus. Denna försökte sedan sända denna data vidare till den virtuella maskinen i molntjänsten via Mqtt. Mikroprocessorn till höger i figuren samlade in sin data från givaren och lyckades sedan sända denna data till en testserver online. Då något certifikat ej fanns tillgängligt kunde inte IoT-Huben ta emot någon data och dashboarden kunde inte heller kopplas samman med IoT-Huben.

5.2 Programmering



Figur 3: Programmering i SoMachine Basic.

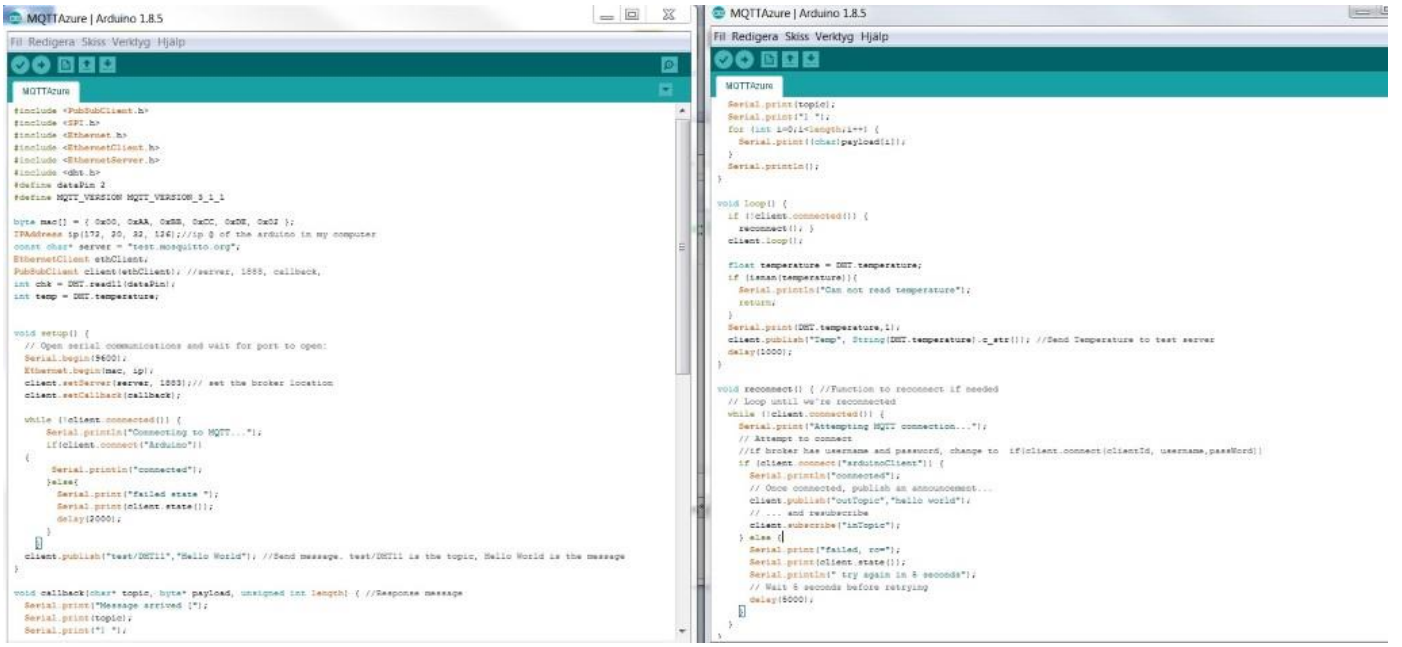
Resultatet vid programmering i SoMachine Basic gav ladderdiagrammet enligt figur 3. Vid simulering ändrades värdet i %MF0 till den aktuella temperaturen givaren kände av.



Figur 4: Kod för programmering av temperatur samt fuktighetssensor.

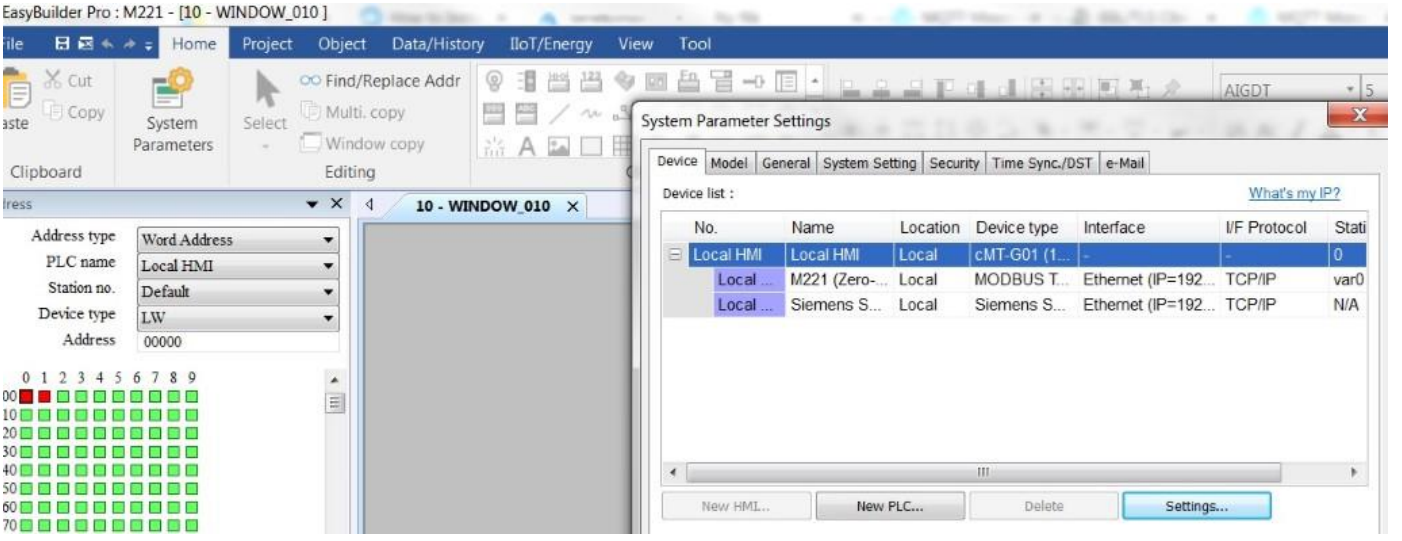
Koden enligt figur 4 användes för att testa kopplingen mellan sensor och mikroprocessor. Koden skriver ut fuktighet samt temperatur i Arduinos konsol, "Seriell monitor".

5.3 Kommunikation



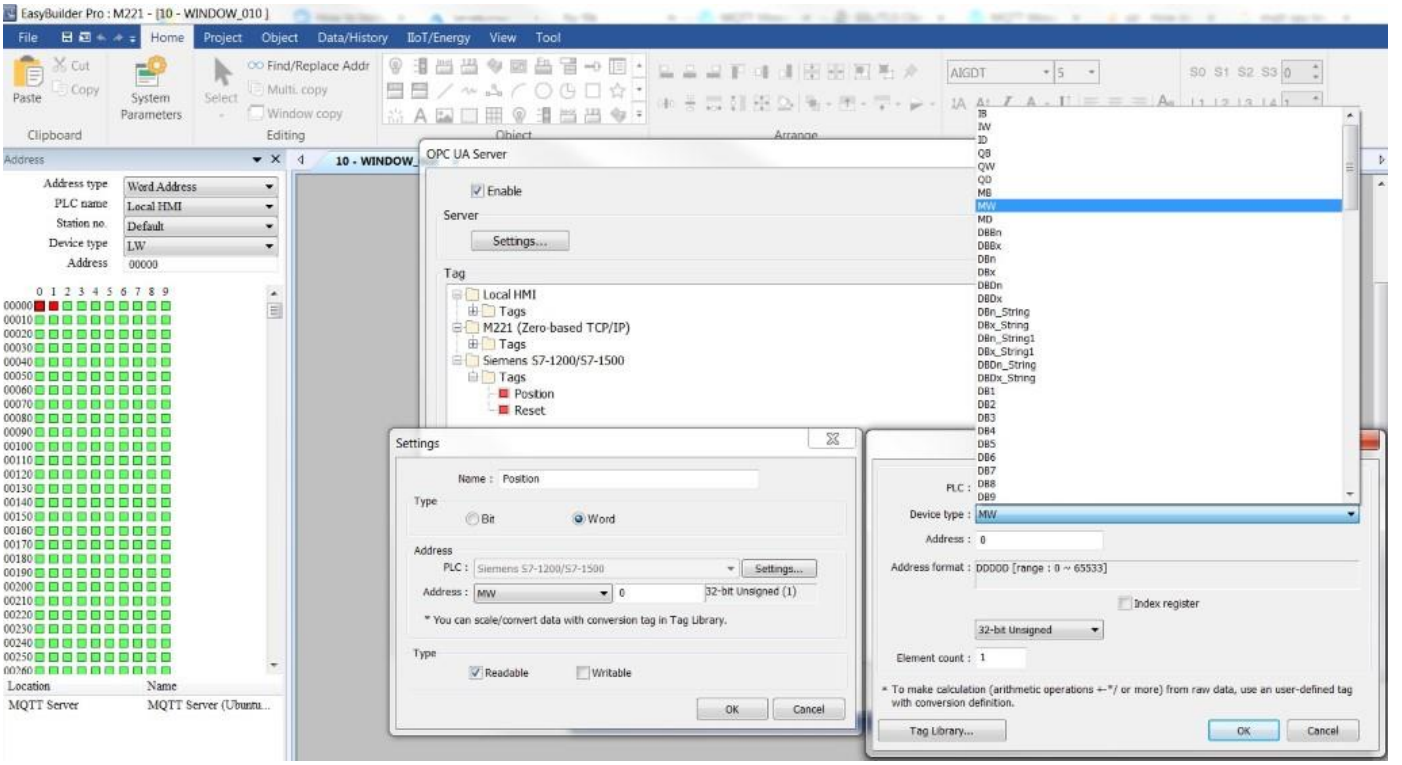
Figur 5: Kod för att skicka temperatur från Arduino till testserver via MQTT.

Koden i figur 5 ställer in MAC- och IP-adress för mikroprocessorn, DNS-adress för testservern, portnummer att skicka via och ansluter sedan till servern med hjälp av ett bibliotek för att sända Mqtt-meddelande. Uppkopplingen säkerhetsställs och ett meddelande med sensordata skickas.

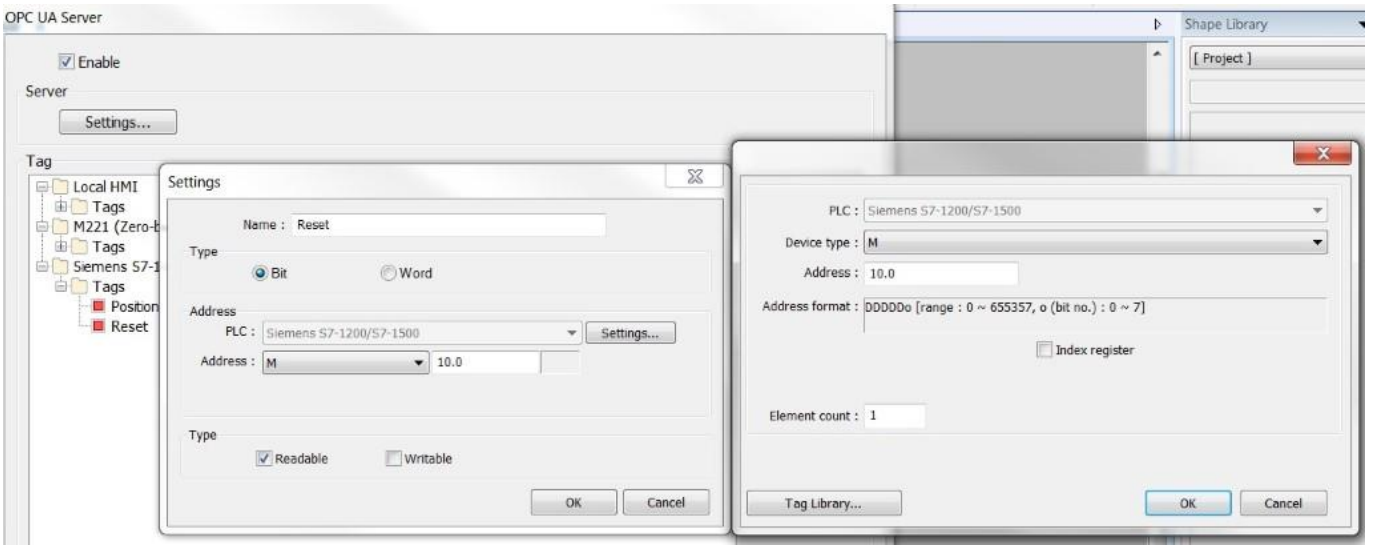


Figur 6: Konfigurering av PLC i EasyBuilder Pro.

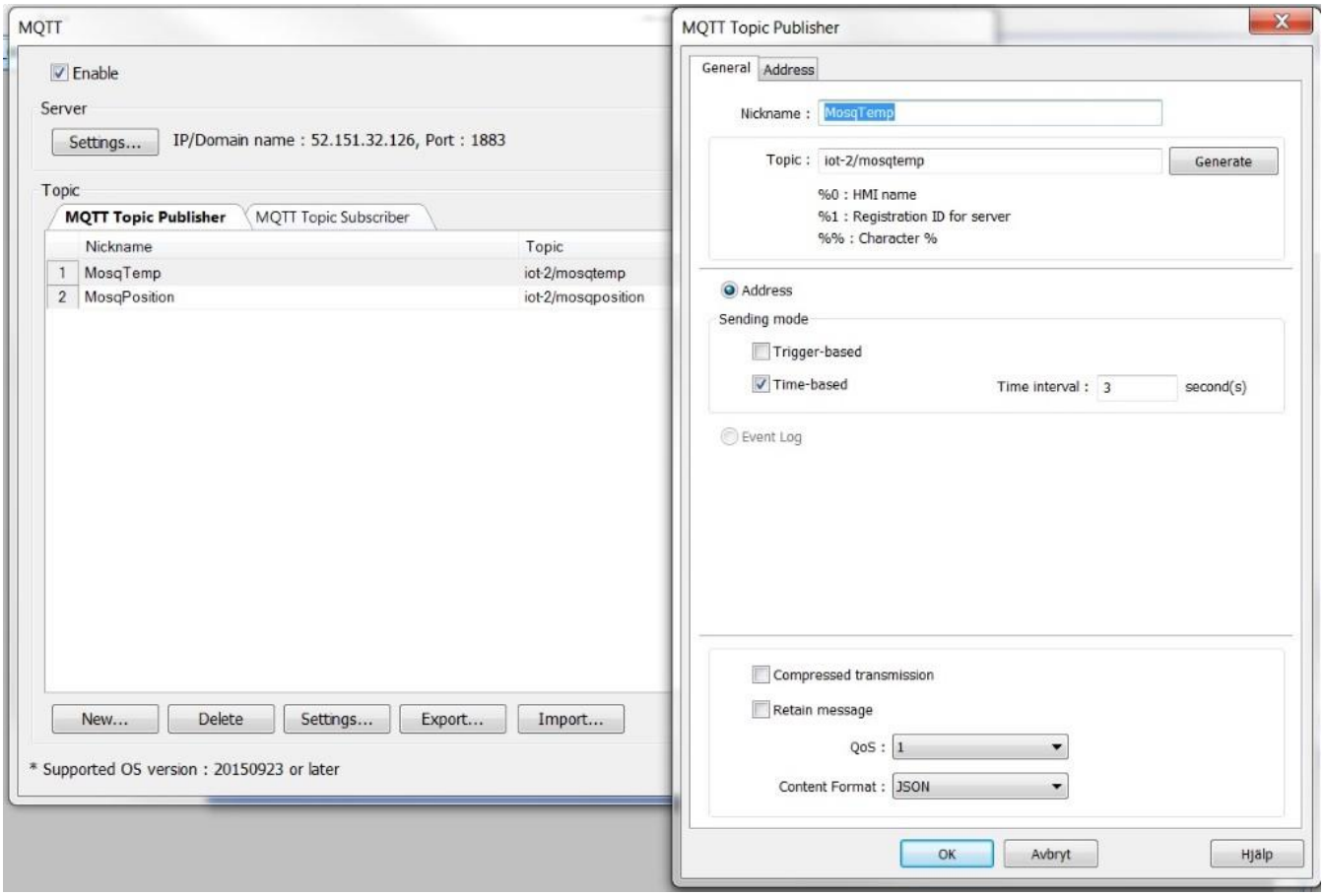
De två PLC enheter som programmerats ställdes in i EasyBuilder Pro, programvaran för gatewayen från Weintek. Både S7-1500 från Siemens och M221 från Schneider använde MODBUS TCP/IP för att överföra data från givare till program.



Figur 7: Inställningar för taggen Position ställs in i EasyBuilder Pro.



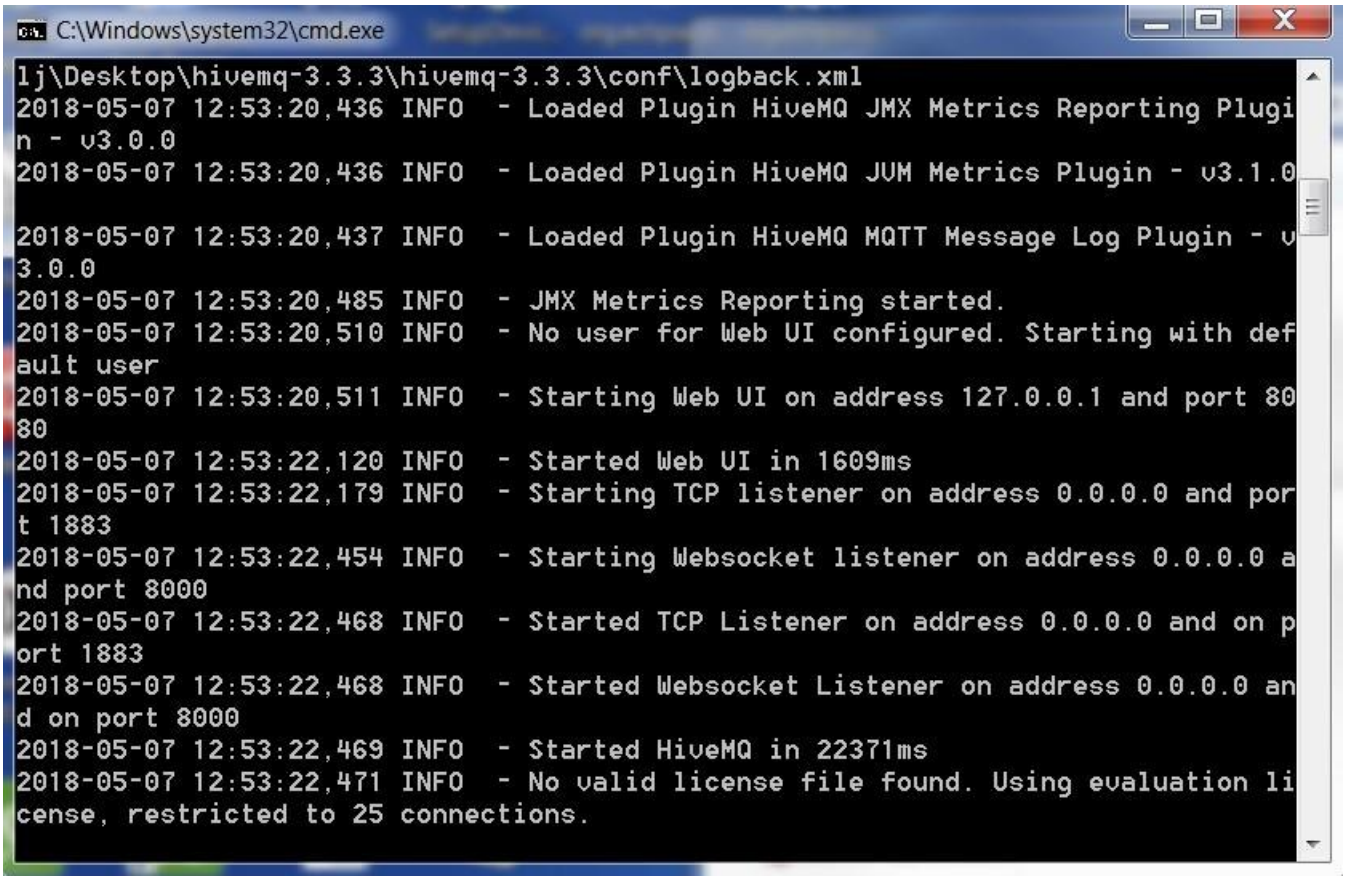
Figur 8: Inställningar för taggen Reset ställs in i EasyBuilder Pro.



Figur 9: Inställningar för Mqtt ställs in i EasyBuilder Pro.

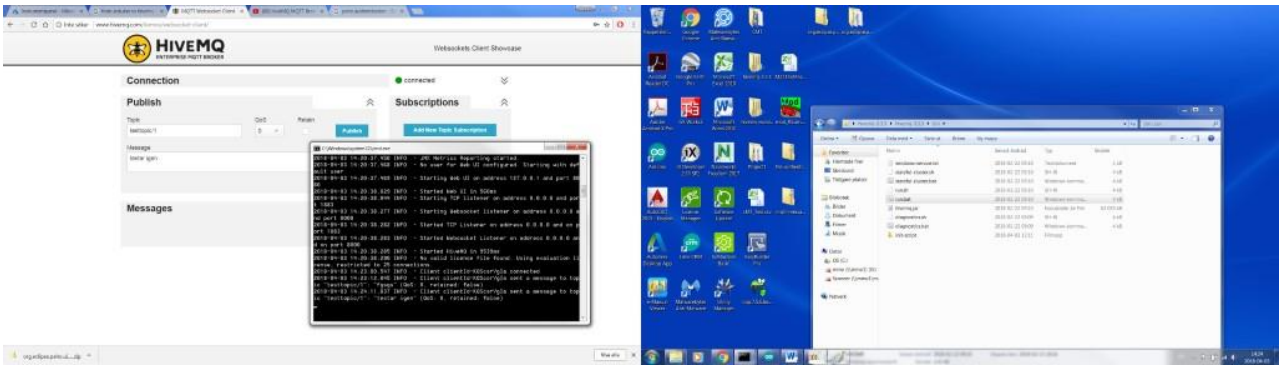
Som figur 6,7,8 och 9 visar var det flera inställningar som skulle ställas in och som varierade beroende på tillverkare och komponent.

5.4 Broker



Figur 10: Status på HiveMQ vid start.

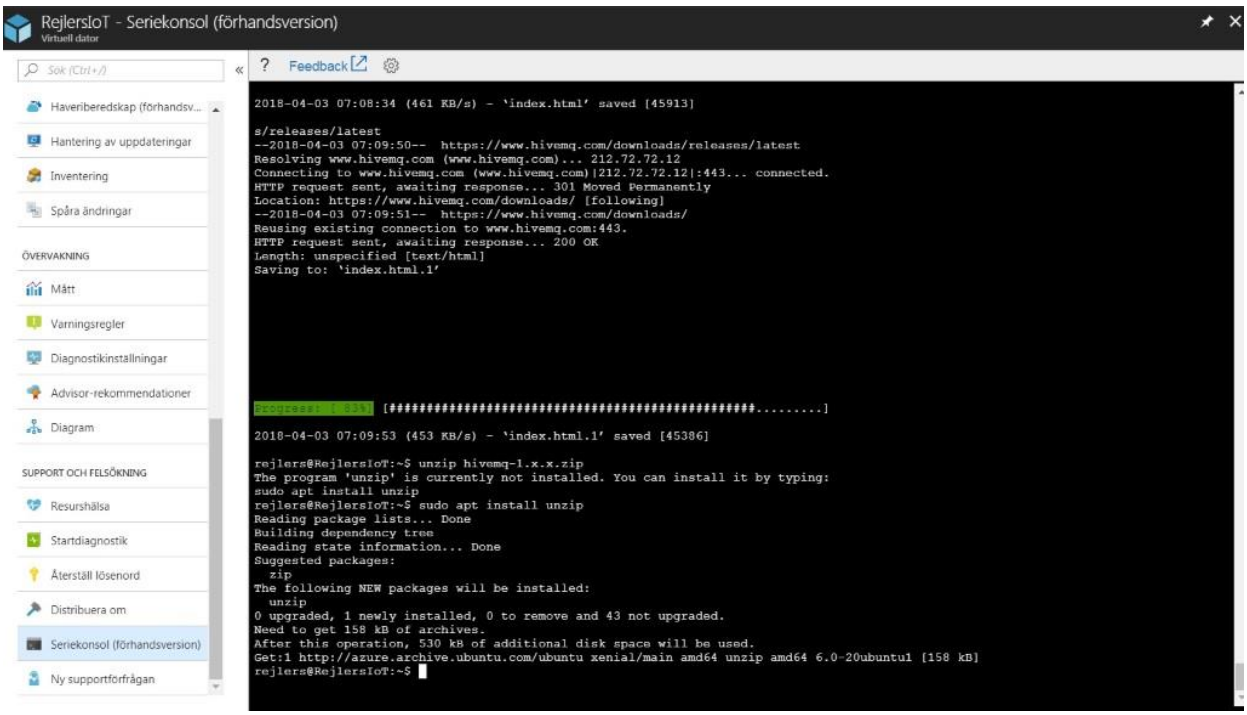
När brokern HiveMQ installerats på arbetsdatorn kunde enligt figur 10 ses att programmet lyssnade på port 1883 vilket indikerade att Mqtt meddelande kunde sändas på denna port.



Figur 11: Onlinetest av websocketport för HiveMQ.

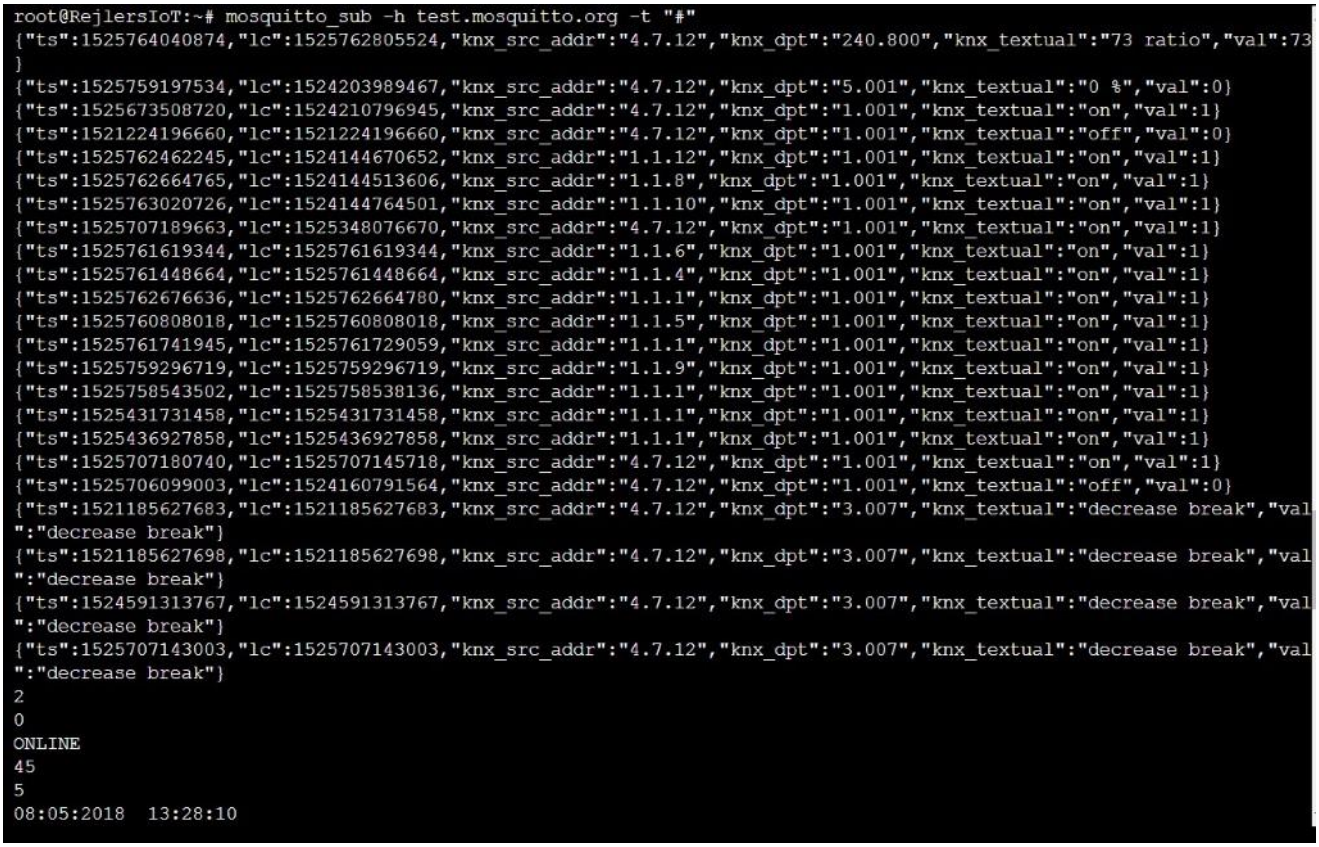
För att kontrollera att brokern installerats korrekt på arbetsdatorn användes en onlinetjänst som tillhandahölls av företaget, HiveMQ. Detta program kontrollerade endast websocketporten 8000 och ej port

1883 som används för Mqtt. Anledningen till detta test var för att säkerhetsställa att programmet installerats utan problem.



Figur 12: Installation av Mosquitto på virtuell maskin Azure.

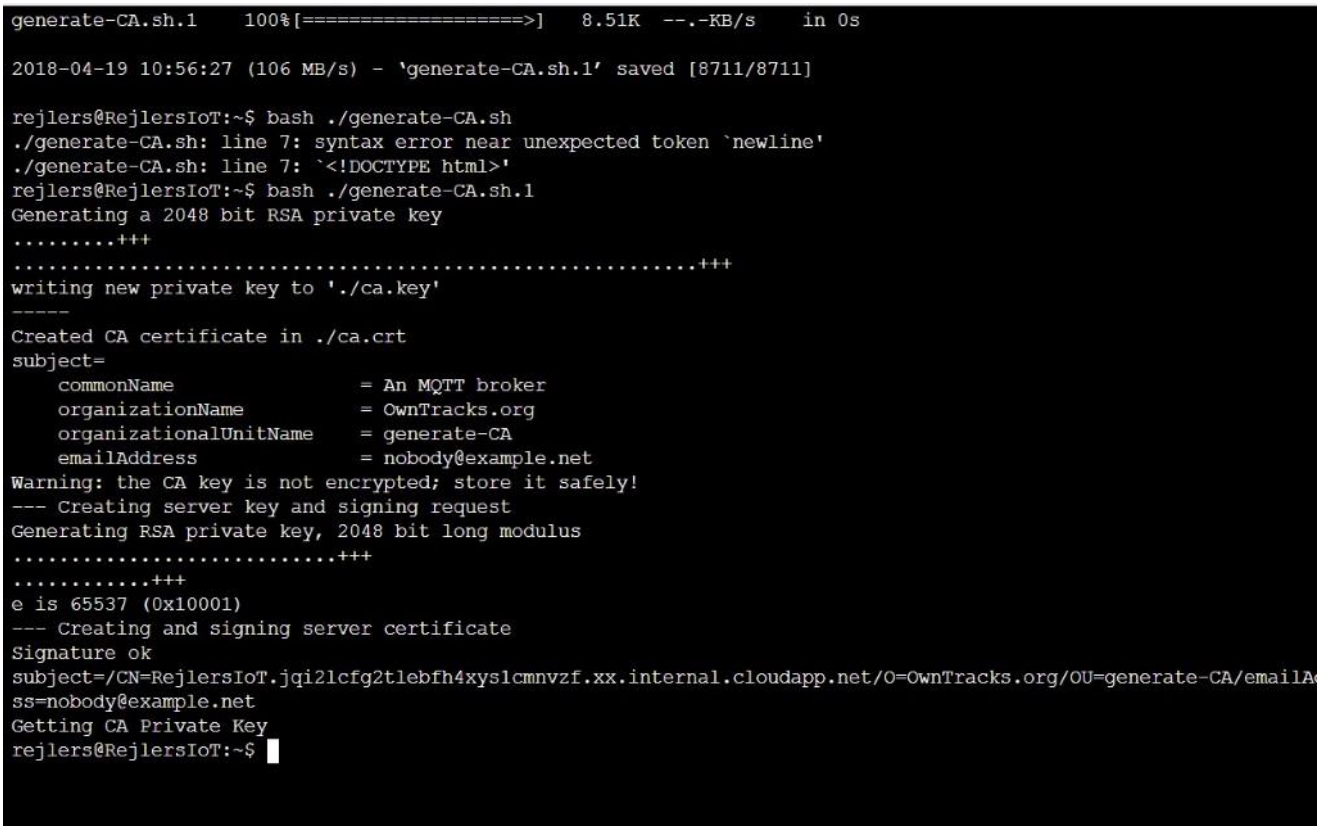
Figur 11 visar kod samt steg i installationen av brokern Mosquitto på den Linuxbaserade VM i Microsoft Azure molntjänst.



Figur 13: Test av broker på VM för att ta emot meddelande från testserver.

Översta raden i figur 13 "mosquitto sub" beskriver ett test för att "Subscribe" d.v.s hämta ett meddelande från testservern 'test.mosquitto.org' med ett wildcard (#) angivet som "Topic".

5.5 Certifikat



Figur 14: Försök att generera användbart certifikat på VM.

Figur 14 visar processen när ett certifikat mellan klient och server försökte skapas. Detta fungerade ej att använda i praktiken. Ett giltigt certifikat måste utfärdas av CA.

Resultaten enligt figur 10 till 14 visar att två olika brokers installerades, en på arbetsdatorn samt en broker på den VM i Azure. Ett certifikat för att skapa en säker förbindelse mellan VM och IoT Hub försökte skapas utan framgång. Då pengar ej kunde läggas på att ordna ett fungerande certifikat kunde ej processen med att skicka vidare data till en dashboard och presentera detta fortskrida och utredningen om att skicka upp data till IoT Huben fick avlutas. För att kunna få ut värden som presenteras på en dashboard måste dessa hämtas direkt från IoT Huben, då detta ej var möjligt bestämdes efter möte med handledaren att endast försöka skicka data till brokern på den virtuella maskinen.

5.6 Gateway

Testmeddelande hämtades från Mosquittos testserver med Mosquitto-brokern. När kommunikationen verkade fungera skickades ett meddelande från Arduinon ut från Rejlers företagsnätverk till en testserver online, detta utan framgång. Samma testmeddelande skickades senare, här med framgång, från Arduinon till testservern när enheten istället var kopplad till ett hemnätverk. Detta meddelande kunde sedan hämtas med hjälp av Mosquitto brokern som var installerad på VM.

Efter kontakt med IT-supporten på Rejlers visade det sig att både port 1883 samt port 8883 var blockerade i säkerhetssyfte. Detta innebar att vidare tester kopplade till företagsnätverket ej kunde fortsätta. Tester med Arduino fortsatte att göras hemma och data skickades upp till adafruit.io, en onlineserver med privat konto, för att visa en enklare skiss på hur IoT fungerar. Detta för att kunna visa en tydligare bild när data från PLC ej kunde sändas och realiseras.

6 Slutsats

6.1 Slutsats

Detta examensarbete som har undersökt möjligheten att hämta data från en fysisk givare och omvandla temperatur till digitala värden på en server i en molntjänst har rört flera intressanta delar inom området Industry 4.0. Arbetet har bland annat visat att det är svårare att genomföra detta praktiskt än vad som tidigare har trotts. Säkerheten är den i särklass största utmaningen, men även kommunikationen mellan de olika komponenterna är ett steg som behöver undersökas noggrant för att allting ska fungera korrekt.

Arbetet har visat att data från olika PLC-enheter kan samlas i en programvara som sedan laddas ner till en gateway, som sedan efter att korrekta mottagarinställningar angivits kan skicka med Mqtt till en molntjänst. Detta ger en möjlighet att bygga ett IoT-system så länge PLCn har en Ethernetutgång, den behöver alltså inte vara klassad som en IoT-enhet. Att kunna samla all information på en plats och endast sända detta vidare från en enhet är en effektiv metod istället för att behöva ha en gateway för varje PLC. Det är dock en fördel att endast koppla små grupper av enheter till en gatway för att få bättre kontroll och prestanda samt för att kunna urskilja de olika delarna i exempelvis en anläggning. På så sätt kan man strukturera data som kommer in till molnservern efter olika faser i en tillverkningsprocess.

Resultatet bevisar även att testmeddelanden kan skickas från en broker via Mqtt. En undersökning samt installation av en broker har genomförts och visat på fördelar samt nackdelar med olika utvecklare och metoder att installera programvaran på. Vid tester med brokern konstaterades att certifikat krävs för att skapa en säker anslutning mellan klient och server, något som kräver mer kunskap samt pengar än vad som var möjligt för denna undersökning. Detta skapade problem med fortsatt kommunikation mellan delarna i pilotanläggningen då data inte kunde analyseras som väntat. När det konstaterats att problemet inte gick att lösa inom tidsplanen för arbetet fick systemet skalas ner och istället visa på möjligheterna med molntjänster och hur vägen dit kan se ut samt vilka svårigheter som kan dyka upp med att skicka data till en molntjänst. Förutom problem

med certifikat fanns det även en begränsning i företagsnätverket hos Rejlers som av säkerhetsskäl blockerade de portar som behövde användas. Fortsatta tester med att skicka meddelande kunde av denna anledning ej fortsätta på plats utan endast småskaliga tester med en Arduino kunde genomföras i det privata hushållet hos examensarbetaren på ett hemnätverk med öppna portar.

Examensarbetet har förhoppningsvis gett en inblick i vad Industry 4.0 är och vilka steg som behöver genomföras för att ta sig från givare till molntjänst. Det har visats hur denna molntjänst ska skapas och konfigureras samt hur en användare måste tänka i ett säkerhetsperspektiv för att undvika att data kommer till fel händer eller manipuleras.

6.2 Frågeställningar

- ***Vad krävs för att kunna skicka upp och spara data i en molntjänst?***

För att kunna skicka upp data och spara denna i en molntjänst krävs det först och främst att data samlas in på något sätt till exempel från en givare. Detta värde måste sedan tas emot av en enhet och omvandlas till digitala signaler. Enheten måste sedan använda sig av ett program som har möjlighet att skicka via Mqtt-protokollet eller överföra data till en enhet som kan skicka via Mqtt. Ett konto hos en molntjänst måste finnas som kan ta emot data via Mqtt. För att sedan kommunikationen ska fungera krävs ett certifikat mellan server och klient.

- ***Vilket typ av protokoll bör användas för att skicka data från en Raspberry Pi till en molntjänst?***

För att skicka data från en Raspberry Pi, eller någon annan enhet för den delen, till en molntjänst är det lämpligast att använda protokollet Mqtt. Detta är ett lättviktsprotokoll som lämpar sig utmärkt för mindre enheter med begränsad processorkraft såsom en Raspberry Pi. Detta protokoll skickas via internet från port 1883 eller 8883 där det senare alternativet har ökad säkerhet med SSL/TLS. Inom IoT används gärna mindre enheter och därför är det viktigt att använda ett protokoll som Mqtt för att minska risken för överbelastning eller liknande fel.

- ***Vilka typer av komponenter behövs och går dessa att byta ut?***

Vid jämförelsen av komponenter visades att en enhet för att skapa data (PLC, mikroprocessor etc.), givare samt en IoT-gateway kan behövas. Alla dessa komponenter finns i flera olika utföranden och går i de flesta fall att byta ut till något likvärdigt. Komponenterna måste dock undersökas för att vara säker på att de fungerar ihop med de andra komponenterna. Till pilotanläggningen valdes en Schneider PLC med en matchande temperatursensor för att skapa data som kunde användas för att skicka vidare till molntjänsten. Rejlers lånade ut en Gateway från Weintek som skulle vara lämpad för Iot. Denna enhet kopplades ihop med PLC via Ethernet och skickade ut data på nätet via Mqtt. I programvaran för denna kopplades taggar ihop och DNS-adressen för IoT-Huben i molntjänsten angavs som slutpunkt för informationen. Denna gateway använde säkerhet med certifikat och skickade ut data på port 8883 för att hålla hög säkerhet. Ett sådant certifikat försökte skapas, men detta skedde utan framgång vilket ledde till att undersökningen fick avgränsas till att endast undersöka möjligheten att skicka upp data utan säkerhet. Detta hade även kunnat göras med en IoT gateway från en annan tillverkare, men resultatet skulle förmodligen blivit detsamma då Microsoft Azure kräver certifikat för att ta emot data.

- ***Hur kopplas en ”dashboard” ihop med en molntjänst för att få ut data i realtid?***

Dashboarden som ej skapades, men studerades i förundersökningen kopplas ihop med IoT-Huben i Microsoft Azure. Med hjälp av säkerhetsnycklar som finns i Azure ska, enligt de guider som finns tillgängliga på utveklarens hemsida, dashboarden vara enkel att installera. Dashboarden kan antingen användas genom att logga in på sitt konto online eller genom att ladda ner en app på vilken du kan se värden och resultat i telefonen eller på en surfplatta. För att säkerheten ska vara bibehållen måste säkerhetsnycklar hämtas från IoT-Huben där inloggning först måste ske.

6.3 Samhällsnytta

Att utvecklingen för Internet of Things kommer att gynna samhället råder det inga tvivel om. Genom att koppla upp enheter, givare och att samla information lätt och snabbåtkomligt i en molntjänst kan vi driva

utvecklingen framåt i snabbare takt än tidigare. Lättillgängligheten, snabbheten och effektiviteten är de tre största faktorerna som ger Industry 4.0 fördelen över det tidigare industrisystemet samhället är baserat på. Genom att kunna samla in värden i realtid och sedan analysera dessa i en databas borde vi kunna åtgärda problem snabbare och kanske t.o.m. förebygga problem innan de uppstått. Genom att placera små givare vid t.ex. kullager på stora maskiner och sedan samla in denna data i ett system kan vi över tid se förändringen av vibrationer i kullagret och byta det när funktionen bedöms vara opålitlig. Detta skulle kunna göra att underhållet på maskinen blir enklare och snabbare än om kullagret går sönder och fler komponenter såsom remmar brister som följd av haveriet ifall åtgärd först görs vid problem.

Att slippa ha egna servrar eller flera fysiska hårddiskar i en produktionslokal kan också vara en säkerhet ifall ex brand eller strömavbrott skulle kunna förstöra arbetet. Genom att ha backup och analyser på en server i en molntjänst minskar risken för att förlora allt arbete eftersom att det är mer osannolikt att både den egna anläggningen samt servern i molntjänsten skulle lägga av samtidigt. Det är även smidigt för en utomstående person att felsöka anläggningen utan att behöva vara på plats fysiskt. Genom att ha data och statistik i en molntjänst kan denne person få åtkomst till värdefull information och börja fundera på en lösning innan något arbete sker på plats. För fabriker med små marginaler kan denna tidsskillnad vara avgörande och problem längre fram i processen kan undvikas.

För den enskilde individen kan livet göras enklare, bekvämare och mer spännande med hjälp av t.ex. smarta hem som bygger på att enheter kommunicerar med varandra. Vid utvecklingen av IoT i framtiden finns möjligheten att kylen skickar ett sms till din telefon när mjölken börjar ta slut eller att du kan fjärrstyra din ugn ifall den glömts att stängas av. Dessa möjligheter kan bidra till minskad stress, större säkerhet samt ett ökat intresse för utveckling av samhället. Med största sannolikhet kommer det finnas möjlighet att anpassa egna lösningar med hjälp av programmering och den teknikintresserade kan på så sätt lära sig att anpassa sitt eget hem med kreativa lösningar.

6.4 Framtida utvecklingsmöjligheter

Då utredningen under arbetet visade att ett certifikat krävdes för att fortsätta finns det en hel del utveckling kvar som arbetet kan bygga vidare på. För det första hade ett certifikat behövt utfärdas vilket i sin tur hade öppnat upp möjligheten att skicka data från PLC till IoT-Huben i Microsoft Azure. När detta sedan fungerat kunde mer omfattande arbete med att skicka större mängder data gjorts. För att komma vidare med problemet med säkerhetsinställningar på företagsnätverket hade ett mobilt nätverk med eget SIM-kort kunnat användas för att koppla förbi Ethernetportarna som används till företagsnätverket vilket gjort det möjligt att använda port 1883 samt 8883 för att skicka via Mqtt. Ett annat alternativ skulle vara att dela internet från mobilen för att komma utanför företagsnätverket, men då hade en router som omvandlar från mobilen trådlösa signal till trådbunden signal med Ethernet krävts då gatewayen som användes ej har någon WiFi-modul.

Vidare utveckling kan bestå av utveckling av en dashboard tillsammans med ett snyggt visuellt gränssnitt och praktiska funktioner för rapportering och analys av data. Med ett komplett system hade en liknande lösning kunnat presenteras för andra företag och öppna en affärsmöjlighet som ett första steg in i Industry 4.0. Då tekniken är ny och förmodligen kommer att ge bättre lönsamhet för många processer finns det mycket arbetsmöjligheter inom området. En gateway kan då presentera med ett lättöverskådligt gränssnitt vilka problem som uppstår i t.ex. en fabrik utan att teknikern behöver vara uppkopplad på nätverket inne i anläggningen. Detta kan vara tidseffektivt samt spara pengar då en tekniker kan lösa problem på distans istället för att vara stationerad på plats för att se förloppet.

7 Terminologi

Broker – En "mellanhand" som skickar eller tar emot information som sänds över Mqtt.
Publish – En enhet som sänder meddelande.
Subscribe – Lyssna på meddelande skickade av en "publisher".
Topic - Ämne/Adress som Mqtt meddelande märks med.
Gateway – Enhet som tar emot och sänder vidare data.
Wildcard - Hämtar information från alla topics på angiven nivå.
Dashboard – Ett grafiskt gränssnitt som presenterar data visuellt.
Molntjänst - En server som går att komma åt när som helst via internet.
IoT-Hub – En samlingsplats för data på en molnserver.
Encoder – En elektromekanisk enhet som omvandlar rörelse från en axel till ett analogt värde.
VM – Virtuell maskin, programvara som simulerar ett operativsystem.
Mqtt – Ett särskilt protokoll
Ethernet – Teknik för att skicka data, främst använt i LAN, MAN och WAN.
SDK – Utvecklingsverktyg för tredjepartsutvecklare.
CA – Certificate Authority.
SSL/TLS – Säkerhetsprotokoll.
Modbus – Överföringsprotokoll.
Molntjänst – Tjänst som hyr ut plats på en server.
Wireshark – Program för att analysera data.
Switch – Nätverksenhet för att koppla ihop flera enheter.

8 Källförteckning

[1] Libris http://uppsok.libris.kb.se/sru/uppsok 2018-05-21

[2] Weintek http://www.weintekusa.com/globalw/News/News_Content.aspx?Nid=20160217_0002&C=TechNews) 2018-05-13

[3] HiveMQ https://www.hivemq.com/blog/mqtt-essentials-part-5-mqtt-topics-best-practices 2018-05-15

[4] Mqtt DZone https://dzone.com/refcardz/getting-started-with-mqtt?chapter=1 2018-05-15

[5] Arduinobibliotek https://pubsubclient.knolleary.net/ 2018-05-16

[6] HiveMQ https://www.hivemq.com/blog/mqtt-essentials-part-3-client-broker-connection-establishment 2018-05-16

[7] Thingsboard https://thingsboard.io/docs/getting-started-guides/what-is-thingsboard/ 2018-05-15

[8] Microsoft Azure https://azure.microsoft.com/en-us/overview/what-is-cloud-computing/ 2018-05-15

[9] Schneider https://www.schneider-electric.com/en/product/TM221CE16T/controller-m221-16-io-transistor-pnp-ethernet/ 2018-05-16

[10] Microsoft Azure https://azure.microsoft.com/sv-se/overview/what-is-a-virtual-machine/ 2018-05-16

[11] Modbus RTA automation https://www.rtaautomation.com/technologies/modbus-tcpip/ 2018-05-16

[12] Modbus Simplymodbus http://www.simplymodbus.ca/TCP.htm 2018-05-16

[13] Weintek http://www.weintek.com/globalw/News/News_Content.aspx?Nid=20170126_0001&C=ProductNews 2018-05-16

[14] Guide Arduino http://www.iotsharing.com/2017/05/how-to-use-mqtt-to-build-smart-home-arduino-esp32.html 2018-03-20

[15] Guide Arduino http://www.instructables.com/id/How-to-interface-Humidity-and-Temperature-DTH11-Se/ 2018-03-20

[16] Guide Arduino https://forum.arduino.cc/index.php?topic=236119.0 2018-03-20

[17] Microsoft Azure https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-mqtt-support 2018-04-05

[18] Jämförelse molntjänster https://paolopatierno.wordpress.com/2015/10/13/an-iot-platforms-match-microsoft-azure-iot-vs-amazon-aws-iot/ 2018-04-15