CODEN:LUTEDX/(TEIE- 120 1 57 2023)

Industrial Electrical Engineering and Automation

Mätning och grafisk visualisering av produktionstakten på tillverkningsprocesser

Albin Larsson

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

Mätning och grafisk visualisering av produktionstakten på tillverkningsprocesser

LUNDS UNIVERSITET
Lunds Tekniska Högskola

LTH Ingenjörshögskolan vid Campus Helsingborg
Industriell elektroteknik och automation

Examensarbete:
Albin Larsson

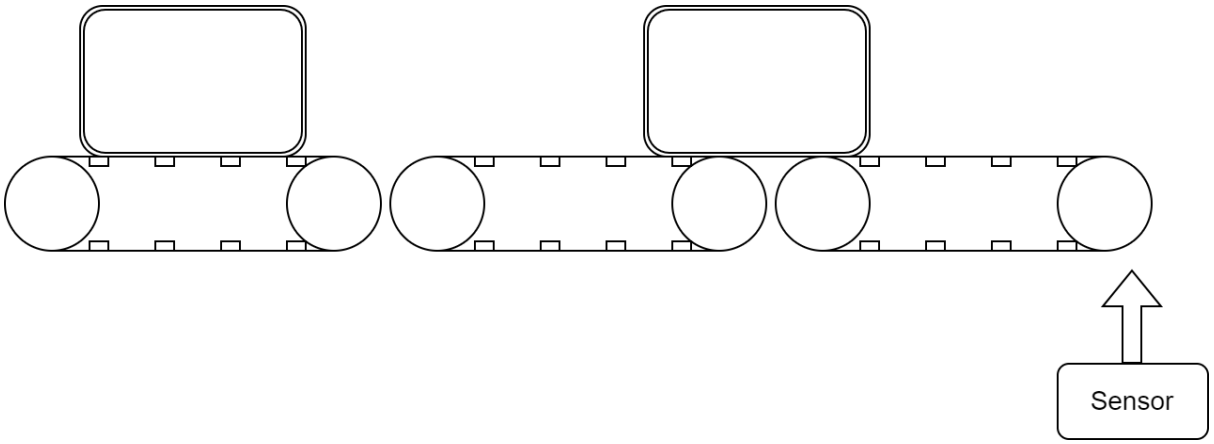
LTH Ingenjörshögskolan vid Campus Helsingborg
Lunds universitet
Box 882
251 08 Helsingborg

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

Tryckt i Sverige
Avd för Industriell Elektroteknik och Automation
Lunds universitet
Lund 2023

Sammanfattning

Målet med det här examensarbetet har varit att ta fram en metod för att mäta produktionstakten på en tillverkningsprocess och grafiskt visualisera den. Det har gjorts för att enkelt kunna avgöra ifall produktionstakten är inom önskat intervall. Produktionstakten beräknas som antal tillverkade produkter dividerat med en bestämd tid. För att genomföra mätningen används en sensor/trigger i slutet av en produktionslinje som illustreras i figur 1.Den framtagna metoden är menad att vara universell och möjlig att flytta mellan olika maskiner. Examensarbetet har utförts hos Automationsteknik AB i Hässleholm.



Figur 1. Produktionslinje med sensor i slutet för mätning av produktionstakten

Examensarbetet började med att ta fram en projektbeskrivning innehållande vad som behövde göras under examensarbetet och vilket resultat som var önskvärt. Därefter undersöktes vilka program som skulle användas för att genomföra examensarbetet. Hårdvarumässigt användes en Siemens PLC och en Siemens IoT-gateway.

Resultatet blev ett operatörsgränssnitt (HMI) där operatören har möjlighet att skriva in information om den tillverkade produkten, en databas för att lagra information om varje tillverkad detalj, två bildskärmar (dashboards) med information om takttiden respektive användningsstatistik om IoT-gatewayen och en operatörspanel för att ändra IP-adress på IoT-gatewayen. I rapporten utvärderas även IoT-gatewayen som användes.

Nyckelord: PLC, Node-Red, HMI, Dashboard, SQL, takttid, produktionstakt

Abstract

The goal of this thesis has been to develop a method to measure the production rate of a manufacturing process and graphically visualize it. This has been done to be able to easily determine if the production rate is within a desired range. The production rate is calculated as the number of manufactured products divided by a certain time. To carry out the measurement, a sensor/trigger is used at the end of the production line as illustrated in figure 2. The developed method is meant to be universal and possible to move between different machines. The thesis has been carried out at Automationsteknik AB in Hässleholm.

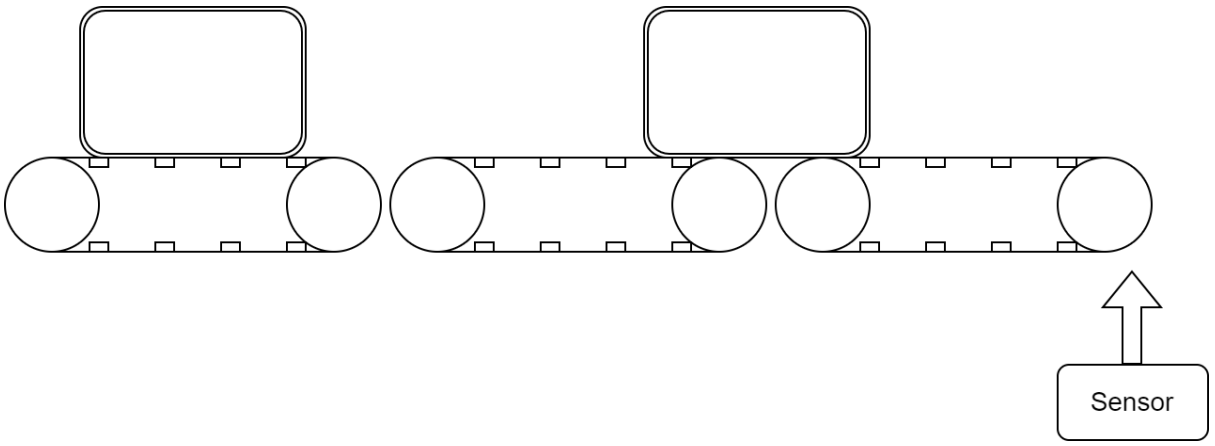


Figure 2. Production line with sensor at the end for measuring the production rate.

The Project began with a project description containing what needed to be done during the project and which result was desired. Next, it was investigated which programs would be used to complete the project. In terms of hardware Siemens PLC and a Siemens IoT-gateway were used.

The result of this thesis was an operator interface (HMI) where the operator has the opportunity to enter information about the manufactured product, a database to store information about each manufactured detail, two monitors (dashboards) with information about the production rate and usage statistics about the IoT-gateway and an operator panel to change the IP-address of the IoT-gateway. The report also evaluates the IoT-gateway that was used.

Keywords: PLC, Node-Red, dashboard, SQL, production rate, production time

Förord

Det här examensarbetet gjordes som en del av utbildningen högskoleingenjör inom elektroteknik vid LTH Ingenjörshögskolan Campus Helsingborg. Det har givit mig en första inblick i automationsbranschen och bättre förståelse för automation överlag.

Jag vill tacka Automationsteknik i Hässleholm AB för att jag fick möjligheten att genomföra mitt examensarbete hos er. Ett extra stort tack till mina handledare på Automationsteknik Ola Fischer och Håkan Marke för ett väl genomfört arbete som handledare. Jag vill också tacka min handledare från universitetet, Morten Hemmingsson och min examinator från universitetet Jörgen Svensson för stöttningen och rådgivning under examensarbetet.

Terminologi

Takttid, Produktionstakt	Tiden mellan varje färdigproducerad detalj på en tillverkningsprocess.
PLC	Programmable Logic Controller.
Open source	Program som är fritt att modifiera.
Fri programvara	Programvara som är gratis att ladda ner och använda.
Raspberry Pi	Enkortsdator.
IoT-gateway	Förmedlar data mellan enheter. Nyare modeller har möjlighet att använda program och hantera data lokalt.
SQL	Structured Query Language. Programmeringsspråk för databaser.
Dashboard	Bildskärm. Grafisk visualisering av värden på en skärm.
Operatörspanel	Kommunikationsplattform mellan en operatör och programvaran där operatören kan ge direktiv genom att modifiera variabler.
HMI	Human Machine Interface. Operatörspanel.

1 Inledning

I inledningen presenteras bakgrunden till examensarbetets genomförande. Företag, syfte, målformulering, problemformulering, motivering och avgränsningar framförs också.

1.1 Bakgrund

Vid driftsättning av nya projekt är det viktigt att tidigt analysera produktionstakten för att avgöra om en maskin producerar detaljer i önskad takt. I det här examensarbetet kommer takttiden och produktionstakten benämnas synonymt. För tillfället görs detta genom att antingen köra maskinen under ett bestämt tidsintervall och sedan räkna hur många detaljer som har producerats eller genom att mäta tiden mellan detaljerna som produceras. Takttiden som mäts sparas för tillfället inte någon längre tid och sättet takttiden grafiskt visualiseras på är med det här examensarbetet tänkt att förbättras. Under examensarbetet beräknas produktionstakten genom att tiden mellan de färdigproducerade detaljerna mäts.

En möjlig lösning för att ta fram en förbättrad metod är att använda en IoT-gateway. Den ska sammankopplas med en PLC som styr tillverkningsprocessen för att därefter insamla information varje gång en detalj tillverkas. Informationen skickas sedan till IoT-gatewayen där takttiden beräknas och sparas i en databas. Från databasen inhämtas takttiden för att grafiskt visualiseras på en dashboard.

IoT-gatewayen som är tänkt att användas är en Siemens Simatic IOT2050. Det är en modul som kan programmeras för att köra program lokalt i sitt eget system för bland annat datahantering och kommunikation med andra enheter. Under examensarbetet kommer även IOT2050 och dess funktionalitet utvärderas på uppdrag av Automationsteknik AB.

Examensarbetet ska genomföras hos Automationsteknik i Hässleholm AB som är inriktade på att ta fram automationslösningar inom bland annat process- och livsmedelsindustrin. De tillhandahåller även konsultation, utbildning inom konstruktionsteknik och PLC-programmering med mera.

1.2 Syfte

Syftet med examensarbetet är att ta fram en metod för att mäta en monterad maskins takttid. Takttiden ska sparas för att göra det möjligt att undersöka den vid ett senare tillfälle. Det ska även vara möjligt att undersöka en specifik produkts takttid vid ett senare tillfälle.

Syftet är även att undersöka IOT2050. Hur den är att arbeta med och hur den kan implementeras i automationsprojekt kommer undersökas.

1.3 Målformulering

Examensarbetet ska resultera i en fungerande metod för att på ett enkelt sätt kunna undersöka takttiden för en tillverkningsprocess på en maskin. Det ska även vara möjligt att gå tillbaka i historiken för en specifik maskins takttider och se vad den har varit för en specifik detalj.

1.4 Problemformulering

Examensarbetet är menat att besvara frågorna:

1. Vilka program behövs för att ta fram en dashboard?
2. Vilka fördelar finns av att spara takttiden under en längre tid?
3. Hur ska mjukvaran och hårdvaran utformas för att bli universell och göra det möjligt att snabbt kunna mäta takttiden hos en ny maskin?

1.5 Motivering

Examensarbetet valdes då det har möjlighet att underlätta vid utvecklingen av framtida projekt där takttiden är en viktig parameter. Examensarbetet ger även en inblick i hur det är att arbeta inom automationsbranschen och kännedom om ny mjukvara.

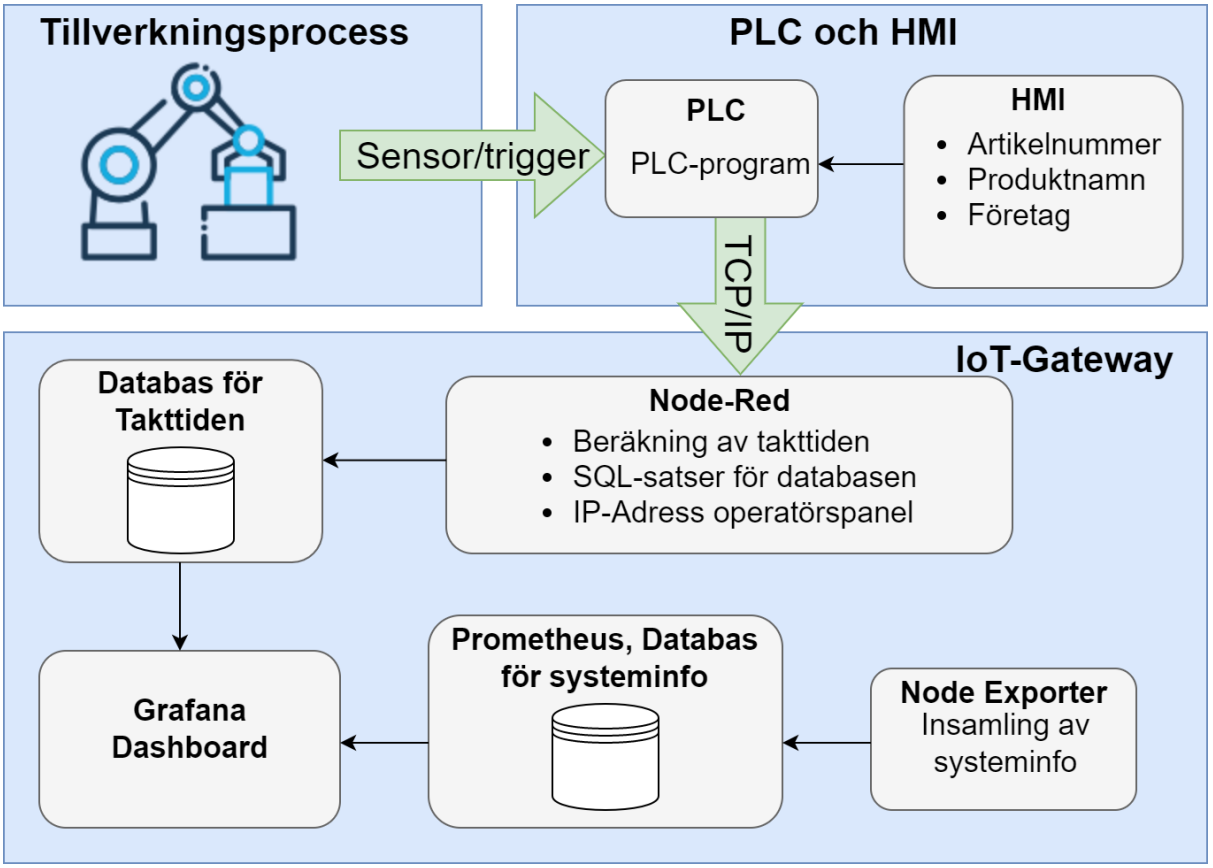
1.6 Avgränsningar

Avgränsningar under examensarbetet är:

- Ingen specifik tillverkningsprocess ska tas fram till examensarbetet.
- Funktionsblocket i PLC-programmet som ansvarar för kommunikationen från PLCn till IoT-gatewayen via TCP/IP är framtagen sedan tidigare och behöver endast modifieras för att implementeras i examensarbetet.
- Dashboarden för användningsstatistik är framtagen sedan tidigare och implementeras endast på IoT-gatewayen.

2 Teknisk bakgrund

För att ge en tydligare bild av examensarbetet kommer detta kapitel att introducera hårdvaran, mjukvaran och kommunikationsprotokollen som använts under examensarbetet. Figur 3 presenterar hur examensarbetet är uppbyggt mjukvarumässigt och vad varje program är menat att fylla för funktion. All mjukvara som används är implementerad i IoT-gatewayen utom TIA Portal som användes för att programmera PLCn och PuTTY som användes för att ansluta till IoT-gatewayen från en dator.



Figur 3. Uppbyggnaden av examensarbetet.

2.1 Hårdvara

2.1.1 PLC

Programmable Logic Controller (PLC) är en industriell dator som fungerar som ett programmerbart styrsystem inom industriella miljöer. En PLC består av beståndsdelarna ingångar, utgångar, minne och processor. Ingångarna tar emot elektriska signaler i form av antingen analoga eller digitala[1]. Digitala ingångar har två möjliga värden, 1 eller 0 medans analoga signaler har flera möjliga värden. Exempelvis kan analoga signaler vara mätning av temperaturer och digitala signaler kan utgöra en strömbrytare.

Utgångarna likt ingångarna levererar antingen digitala eller analoga signaler. Minnet på en PLC sparar PLC-program, data, variabler och är vanligtvis statiskt. Processorn har som funktion att sätta utgångsstatus beroende på vad ingångarna signalerar[2].

De flesta PLC-system följer standarden IEC 61131-3 vilken standardiserar vilka språk en PLC ska kunna programmeras med. Det är följande:

- Structured Text (ST)
- Instruction List (IL)
- Ladder Diagram (LD)
- Function Block Diagram (FBD)
- Sequential Function Chart (SFC)

ST och IL är textbaserade språk medan LD, FBD och SFC är grafiskt baserade[3]. En tillverkare behöver inte stödja alla språk som ingår i standarden.

I takt med att allt mer kommunikation sker med enheter utanför det lokala nätet för datalagring och informationsinhämtning har behovet av att PLCer ska kunna kommunicera med webbsidor och databaser för att exportera och importera data ökat[4].

2.1.2 HMI

För att en operatör ska ha möjlighet att kommunicera med en PLC brukar de vara sammankopplade med ett Human Machine Interface (HMI). Det är vanligtvis någon typ av instrumentpanel. I industriella miljöer brukar ett HMI vara en fristående skärm där operatören kan se information om processen för att övervaka den eller styra processen genom olika funktioner[5].

I detta examensarbete används HMI för att operatören ska kunna skriva in den tillverkade produktens: artikelnummer, produktnamn och företag som tillverkar produkten.

2.1.3 IoT-gateway

IoT-gateway är en modul som kan kommunicera med PLC, IoT-enheter och sensorer. Den kan inhämta data från dem som sedan skickas vidare till t.ex. en molnbaserad tjänst. Nyare modeller av IoT-gateways kan även använda program lokalt i sitt egna system som t.ex. databaser och dashboards. Fördelen med att bearbeta data i program lokalt innan den vidarebefordras är att endast data som behövs kan sorteras ut för att minska kostnaderna för molntjänster [7].

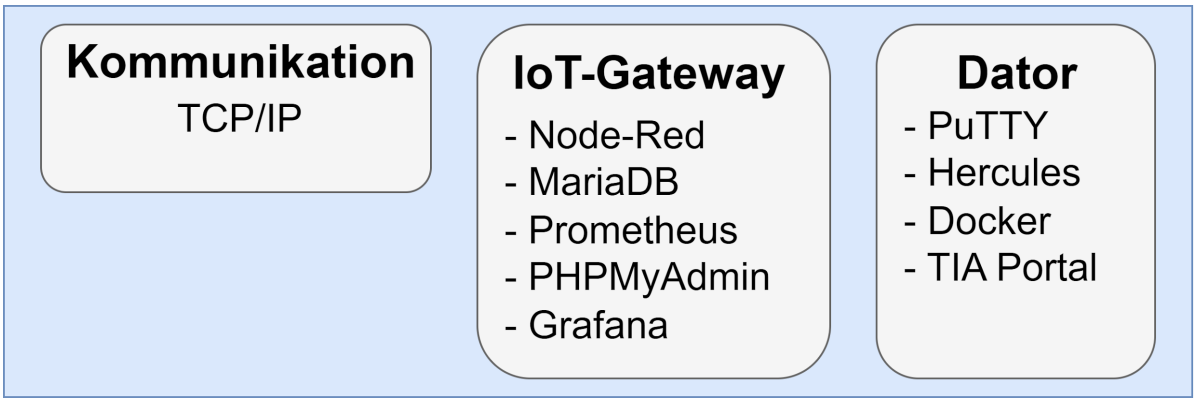
IoT-gatewayen som används under examensarbetet är en Siemens Simatic IOT2050 som visas i Figur 4. Den är designad för industriell IoT, baserad på Linux och använder sig av operativsystemet Debian. Under examensarbetet kommer programmen för databasen och

dashboarden köras lokalt i IoT-gatewayen. All data hanteras lokalt i enhet och ingen kommunikation med molntjänster har gjorts under examensarbetet.

Figur 4. Siemens Simatic IOT2050 i drift.

2.2 Mjukvara

Mjukvaran som har använts under examensarbetet har implementerats i IoT-Gatewayen och datorn som använts under examensarbetet. Undantaget är TCP/IP som är ett kommunikationsprotokoll och inte implementeras på en egen enhet, se figur 5 för redogörelse av vart respektive program är implementerat.



Figur 5. Redogörelse för var respektive program är implementerat.

2.2.1 Programmering av PLC

PLC-programmeringen har gjorts i Totally Integrated Automation Portal (TIA Portal) som är ett program- och verktygspaket utvecklat av Siemens för att ha en centraliserad

utvecklingsmiljö för deras PLCer. Inom utvecklingsmiljön ingår ett stort utbud av program, varav de viktigaste för detta examensarbete är de för programmering, grafisk visualisering av HMI och simulering. PLC-programmeringen görs i programmet Simatic Step7 och kan göras i de flesta språk som ingår i standarden IEC 61131-3. HMI visualiseras grafiskt i Simatic WinCC under utvecklingsfasen och består ofta av en kontrollskärm för operatören. Simulering görs i S7-PLCSIM Advanced[8].

2.2.2 Databas för takttiden

Som databas för att spara värden om takttiden används MariaDB i IoT-gatewayen, MariaDB är en open source relationsdatabas som är lik MySQL i sin struktur. MariaDB utvecklades av det ursprungliga teamet bakom MySQL efter att de blev uppköpta av företaget Oracle, med målet att alltid vara fri programvara. Likt MySQL använder sig MariaDB av SQL (Structured Query Language) som är ett programspråk för att modifiera och hantera data i relationsdatabaser[9].

Det som kännetecknar en relationsdatabas är att data lagras i tabeller som kan ha relationer med varandra. Inom tabellerna struktureras data i rader och kolumner. Varje tabell innehåller vanligtvis ett id för att varje tabell ska vara unik och förhindra dubbletter i databasen[10]. Ett exempel på en MariaDB databas kan vara att det finns tabeller för Tenta, Elev och Kurs. Varje Elev har ett id och ett namn medans varje Kurs har ett id och kursnamn. Id är en primary key och är den kolumn tabellen identifieras med. Informationen från tabellerna används sedan för att skapa kolumner i tabellen Tenta där varje Elev refereras till ett elev_id och varje Kurs till ett kurs_id. Dessa referenser till andra tabeller heter foreign key. Varje tenta kan sedan göras av en elev men varje elev kan göra flera tentor och varje tenta ingår i en kurs medan en kurs kan ha flera tentor. Se Figur 6 för hur exemplet är uppbyggt.

Figur 6. MariaDB databas med Tabellerna Elev, Tenta och Kurs. Elev innehåller kolumnerna id och namn. Kurs innehåller id och kurs_namn. Tenta innehåller id, elev_id och kurs_id varav de två sista refererar till värden från tabellerna Elev och Kurs. Primary keys benämns PK och foreign keys benämns FK.

2.2.3 Administrering av takttid databasen

För att förenkla databas administreringen används PHPMyAdmin som är ett administrationsverktyg för att hantera MySQL och MariaDB databaser. Det är open source och nås efter installation från webbläsaren. Från PHPMyAdmin är det enkelt att strukturera

databaser, lägga till tabeller och modifiera redan existerade tabeller utan att ha någon större kunskap om programmeringsspråket SQL då det är uppbyggt på ett grafiskt användargränssnitt. Det går även att se och sortera data som är infogade i databasen[11].

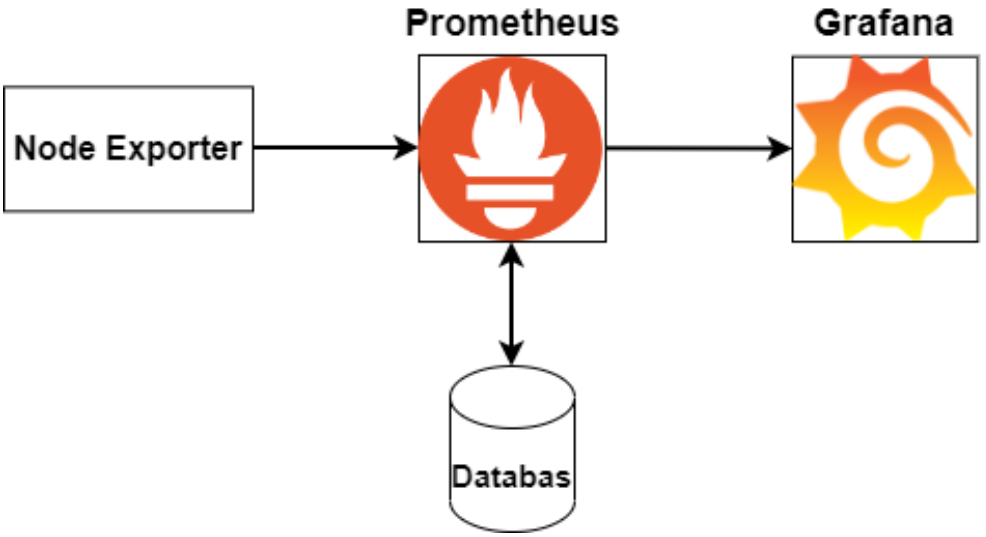
2.2.4 Insamling av användningsstatistik

För att samla användningsstatistik om IoT-gatewayen används Prometheus som är ett program för att övervaka och lagra information om ett system. Den samlar in användningsstatistik och organiserar den efter tidsstämpel. För att utföra detta är prometheus uppbyggt med:

- Prometheus Exports
- Prometheus Server

Prometheus Exports är programvara vars funktion är att hitta och samla in data från systemet vilket görs automatiskt direkt efter nedladdning. Systemet kan vara hårdvara eller mjukvara. Det kan exempelvis vara en dator som använder sig av ett Linuxsystem. En vanlig Prometheus Exporter är Node Exporter som kan samla in data från virtuella- och fysiska maskiners hårdvara.
Prometheus Server samlar in data som Exporten hittat och sparar den med en tidsstämpel i en databas.

Prometheus används ofta tillsammans med något externt program för att grafiskt visualisera data som samlats in, vilket illustreras i Figur 7. Under detta examensarbete kommer den grafiska visualiseringen att göras i programmet Grafana[12].



Figur 7. Node Exporter hittar information om systemet, Prometheus lagrar den i en databas, Grafana visualiserar data som förmedlas av Prometheus.

2.2.5 Grafisk visualisering av databaser

För att grafiskt visualisera databaserna har dashboards tagits fram i programmet Grafana. Det görs genom att konfigurera vilken datakälla som information ska hämtas från och sedan göra förfrågningar om data beroende på vad som ska visualiseras på panelen. Backenden kan vara av en stor variation av datakällor som MySQL, MariaDB, PostgreSQL och Google sheets. Den grafiska visualiseringen kan göras i tidsdiagram, tabell eller grafer för att nämna några. Då all data måste vara tidsbaserad har användaren även möjlighet att själv ställa in under vilken tidsperiod informationen ska inhämtas emellan. Denna tidsperiod kan exempelvis vara senaste 5 minuterna eller under specifik dag. Användaren har även möjlighet att ställa in hur ofta panelerna ska uppdateras.

Flera olika paneler kan sedan sammanställas i en gemensam dashboard där varje enskild panel presenterar information beroende på hur dess förfrågan mot databasen ser ut. Se Figur 8 för hur en dashboard i Grafana kan vara utformad för en webbplats[13].

Figur 8. Grafana dashboard med tidsstaplar- och tidsserie grafer.

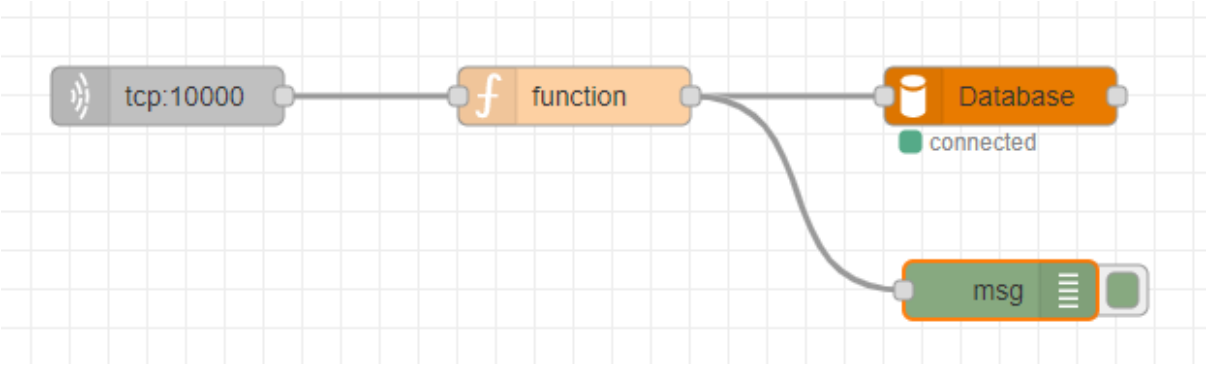
1. Meny för att skifta mellan dashboards.
2. Vilken tidsperiod information ska inhämtas emellan.
3. Värden från databasen representerade i siffror. Från vänster i figuren: Inloggningar, nya användare, användare som lämnar och antal supportsamtal.
4. presentation av värden från databasen i tabeller och tidsserie grafer.

2.2.6 Informationsbearbetning från PLCn

För att bearbeta data från PLCn innan den infogas i databasen används Node-Red i IoT-gatewayen. Det är en webbläsarbaserad utvecklingsmiljö för kommunikation mellan olika program, ursprungligen framtagen av IBM men numera förvaltad av JS Foundation.

Operativsystem Node-Red stödjer Windows, Mac, Linux vilket gör den vanlig att använda till enkortsdatorer och IoT-gateways. Node-Red kan kommunicera genom MQTT, TCP och Modbus för att nämna några.

Dess huvudsakliga funktion är att sammankoppla och underlätta kommunikationen mellan olika program med hjälp av block som kallas noder. Node-Red är byggt på Node.js och alla flöden sparas som JavaScript Object Notation (JSON) vilket gör att noder med olika protokoll kan kommunicera med varandra då de har ett gemensamt format[14]. Figur 9 visar ett exempel på hur Node-Red ser ut då en TCP-node lyssnar på port 10000 och sedan sänder vidare all inkommande data till en mySQL databas via en javaScript funktion. All data som går ut från funktionen går även till en debugger-node som kan läsa av inkommande data.



Figur 9. Node-Red flöde med noderna för TCP, function, databas och debugger.

Node-Red har även stöd för att bygga upp enklare dashboards. Exempelvis kan dashboarden bestå av knappar, tidsdiagram och mätare av olika slag.

Det finns över 225.000 olika noder att ladda ner för att anpassa flödet efter varje unikt behov ett system behöver[15]. Tabell 1 innehåller de noder som har använts under examensarbetet.

Tabell 1. Noderna som användes under examensarbetet och dess funktioner.

Noder	Funktion
TCP IN	Lyssnar efter inkommande data som skickas med TCP protokollet till en specifik port.
JSON	Konverterar ett text-objekt till ett JSON-objekt.
SQLstring-format	Gör det möjligt att skriva in SQL kommandon för att senare exekveras i en databas.
Function	Används för javascript funktioner.
MySQL	Används för att sammankoppla Node-Red med en databas.
Debug	Upptäcker all inkommande data vilket gör det enklare att upptäcka fel i flödet och åtgärda de.
EXEC	Exekvera ett inskrivet kommando i systemets terminal.
Button (dashboard)	Representerar en knapp på Node-Reds inbyggda dashboard.
Text-input (dashboard)	Gör det möjligt att skriva in text till Node-Red från dess dashboard.
Text (dashboard)	Visar en text från Node-Red på en dashboard.

Under examensarbetet används Node-Red i IoT-gatewayen för att ta emot och bearbeta data från PLCn. För att ha möjlighet att ändra IP-adressen på IoT-gatewayen togs en operatörspanel fram i Node-Red där operatören kan lägga till och ta bort IP-adresser på IoT-gatewayen.

2.2.7 Anslutning till IoT-gateway

För att kunna ladda ner programvara till IoT-gatewayen används programmet PuTTY som är ett program för Windows som använder sig av SSH (Secure Shell) eller Telnet. Det används för att upprätta en anslutning till andra enheter via en terminal för att kunna genomföra

kommandon i dess system. Vanligtvis handlar det om anslutning till Linux- och Unixsystem för att ladda ner och modifiera programvara[16].

2.2.8 Test av TCP/IP kommunikation

För att testa TCP/IP kommunikationen används programmet Hercules. Hercules är ett program som kan skicka och ta emot paket via TCP/IP. Det görs genom att Hercules antingen sätts upp som en TCP klient eller en TCP server. Under examensarbetet användes Hercules för att testa TCP-kommunikationen mellan PLCn och IoT-gatewayen innan PLC-programmet var implementerat vilket endast var under utvecklingsfasen[17].

2.2.9 Programvara-nedladdning

För att ladda ner relevanta program för inhämtning av användningsstatistik databasen används Docker som är ett program för att ladda ner, starta och uppdatera programvara. Inom Docker finns containrar som är paket innehållande allt som krävs för att starta en applikation. Containers är skilda från varandra men använder sig av en gemensam operativsystemskärna vilket bidrar till att de använder sig av mindre resurser än vid vanlig nedladdning av program.

Några fördelar med Docker är:

- Enhetens resurser delas mellan containrar.
- Snabb uppstart av program.
- Det finns många färdiga containrar att ladda ner som är konfigurerade att användas på specifika enheter vilket minskar antalet justeringar användaren behöver göra innan programvaran kan börja användas[18].

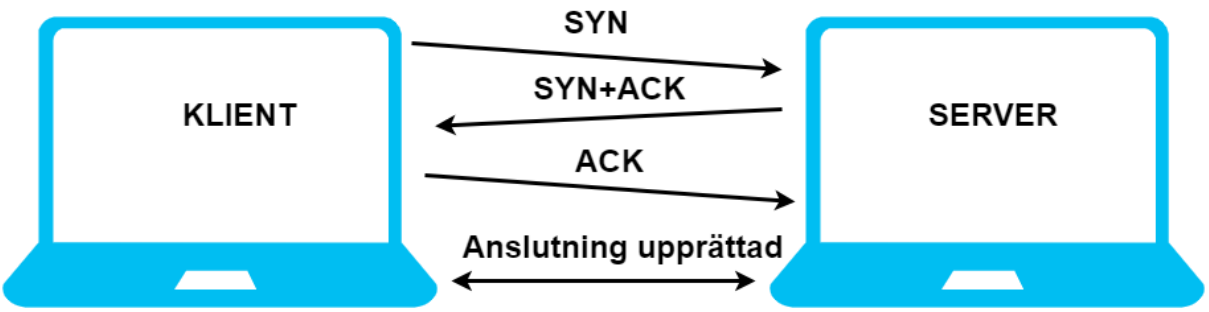
2.3 Kommunikationsprotokoll

2.3.1 Kommunikation mellan PLC och IoT-gateway

För att kommunicera mellan PLC och IoT-gateway används Transmission Control Protocol/Internet Protocol (TCP/IP) som är två protokoll vars uppgift är att specificera hur data och paket ska skickas mellan olika enheter. TCP delar upp data som ska skickas i flera mindre paket som är numrerade. Mottagaren vet sedan i vilken ordning paketen ska sättas ihop för att få fram den tilltänkta ordningen på den skickade informationen och den märker ifall delar av informationen saknas. Fördelen med att inte skicka allt i ett stort paket är att de små paketen kan ta olika vägar till mottagaren vilket gör att paketen kan levereras till mottagaren snabbare. IP protokollet definierar enheternas adress. Det har som uppgift att se till att paketen når rätt mottagare[19].

TCP/IP använder sig av klient-server modell som illustreras i Figur 10. Klient är den som ska skicka paket och server är den som är mottagare. För att upprätta kommunikationen mellan klient och server använder sig TCP/IP av en 3-vägs handshake:

1. Klienten börjar med att skicka till en server att den har paket att skicka (SYN-synchronize).
2. Servern svarar med att skicka tillbaka SYN och ACK (Synchronize-acknowledge). ACK skickas för att verifiera att den har mottagit klientens SYN-meddelande.
3. Klienten skickar tillbaka ACK-meddelandet till servern för att bekräfta serverns svar. Anslutningen är därefter upprättad[20].



Figur 10. TCP 3-vägs handshake mellan en klient och en server.

Fördelarna med att använda TCP/IP är:

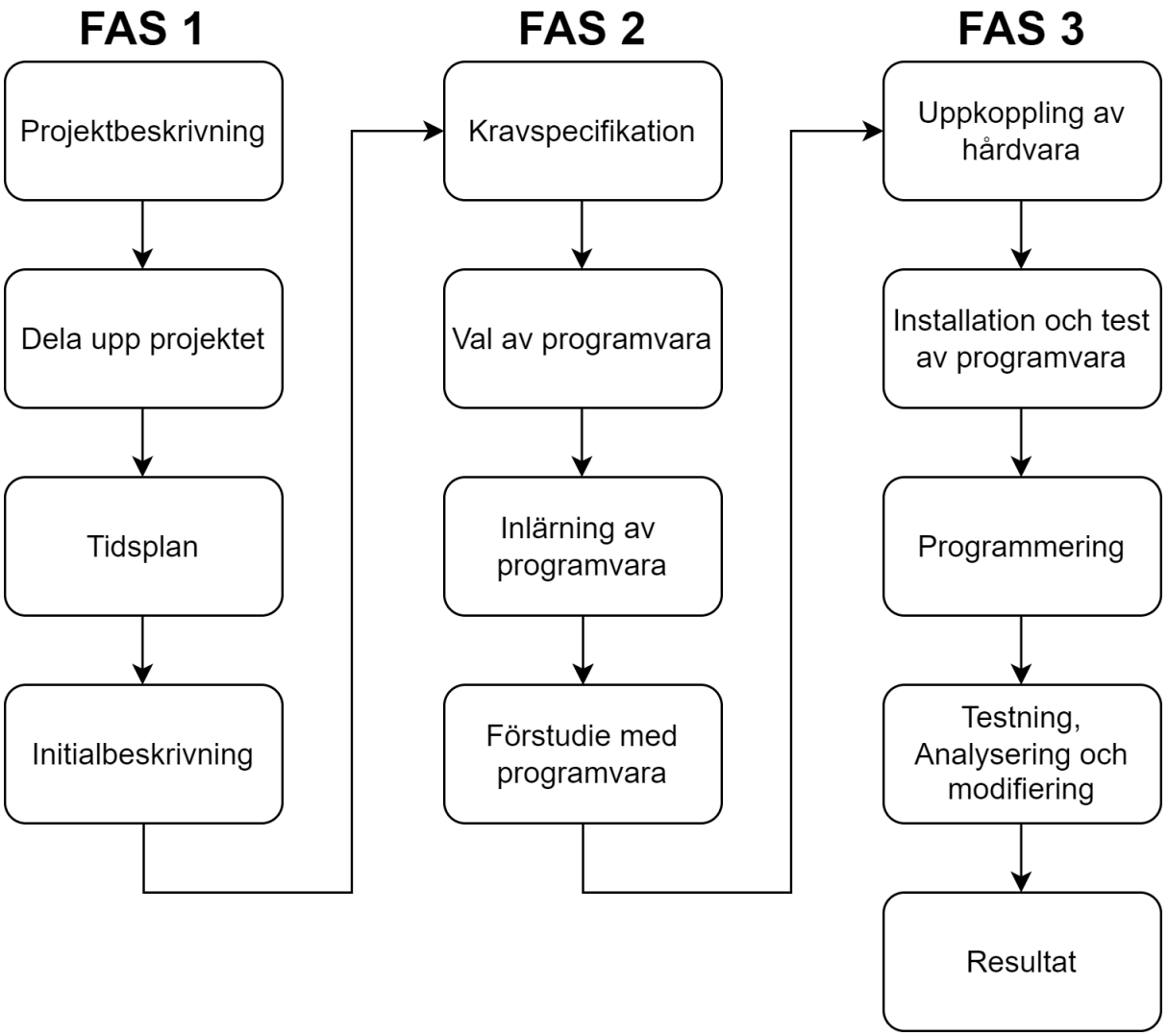
- TCP/IP kan verifiera att data som har skickats har mottagits.
- TCP/IP kan upprätta anslutningar mellan enheter och servrar vilket gör att mottagaren bara behöver verifieras vid första paketet.
- TCP kan kontrollera storleken och flödet av paket som skickas, vilket kan förhindra att paket blockerar anslutningen[19].

TCP/IP används under examensarbetet för kommunikation mellan PLCn och IoT-gatewayen.

3 Metod

3.1 Arbetsprocess

Detta kapitel kommer att redovisa arbetsfaserna som examensarbetet delades upp i. Arbetsprocessen bestod av tre olika faser med ett antal moduler i varje för att enklare lägga upp arbetet. Den första fasen beskriver förberedelserna för att komma igång med examensarbetet. Fas 2 går igenom inlärningen av tekniken som lagt grunden för examensarbetet och första arbetet med hårdvaran och mjukvaran. Den tredje och sista fasen beskriver hur arbetet gick till vid implementeringen av tekniken för att få fram det slutgiltiga resultatet. Resonemanget till varför arbetsprocessen varit utformad som den är kommer att presenteras i kapitel 4. Figur 11 visar ett flödesschema för arbetsprocessen under examensarbetet.



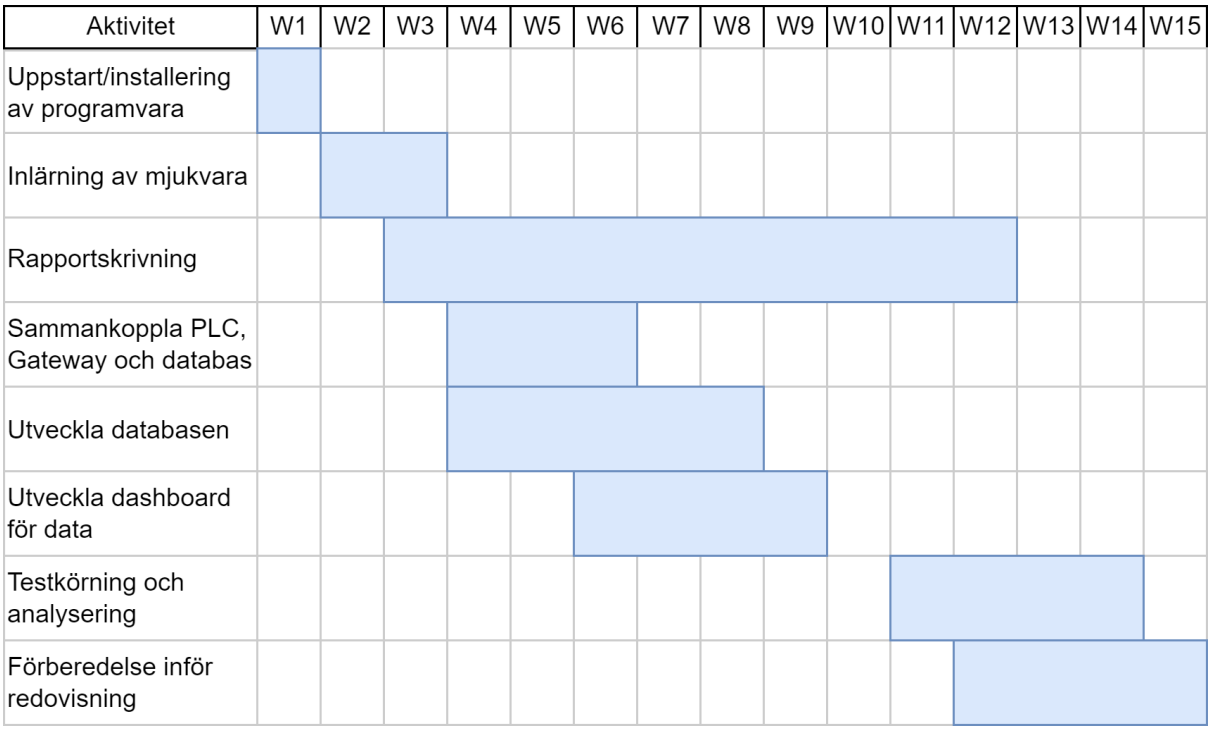
Figur 11. Flödesschema över arbetsprocessen under examensarbetet.

3.2 Fas 1: Initiering och planering

Efter första kontakt med Automationsteknik AB om vad för inriktning på examensarbetet som var aktuellt återkom de med en första projektbeskrivning av examensarbetet.

Efter ett möte där projektbeskrivningen diskuterades vidare och egen efterforskning gjordes om ämnet togs en egen projektbeskrivning fram för att få bättre förståelse för vilka resurser som skulle behövas för att genomföra examensarbetet.

Examensarbetets olika huvuddelar, PLC, databas och dashboard, delades sedan upp för att enklare lägga upp en tidsplan och besluta vad som behövde göras för att genomföra examensarbetet. Figur 12 illustrerar tidsplanen som användes under examensarbetet.



Figur 12. Tidsplanen för examensarbetet.

När projektbeskrivning och tidsplan var gjorda togs en initialbeskrivning fram innehållande bakgrund, syfte, målformulering, problemformulering, motivering, metod, resurser och tidsplanen som tidigare tagits fram. I initialbeskrivningen presenterades även potentiella program som kunde tänkas behövas för att genomföra examensarbetet.

3.3 Fas 2: Förstudie

Det första som gjordes under förstudien var att ta fram en kravspecifikation för att få en uppfattning om hur mjukvaran skulle användas, exempelvis hur takttiden ska beräknas och hur databasen ska byggas upp för att varje sak som ska sparas har plats i en kolumn. Vilken mjukvara som behövdes för att genomföra examensarbetet fastställdes när det stod klart vilken information om tillverkningsprocessen som var av intresse att lagra och vad som

skulle behöva göras med denna data innan den läggs in i databasen. Vilken PLC och IoT-gateway som skulle användas under examensarbetet var beslutat sedan tidigare.

När det var klart vilka resurser som skulle användas blev nästa steg att bekanta sig med programmen som skulle användas för att få en grundläggande förståelse för dem. Programvaran behövde även experimenteras med för att få en känsla för hur programmen fungerar och utvärdera ifall det var rätt program för examensarbetet, som att ta fram enklare dashboards och bygga upp flöden i Node-Red. På grund av halvledarbrist var leveranstiden av IoT-gatewayen ett par månader så allt under förstudien gjordes på en Raspberry Pi. Kommunikation genom TCP/IP mellan PLCn och Raspberry Pi sattes upp för att testa PLC-programmet som skulle användas för TCP kommunikationen. En första version av databasen för att spara information om tillverkningsprocessen byggdes också upp under denna fas.

3.4 Fas 3: Genomförande

När en IOT2050 fanns tillgänglig började den sista fasen som var genomförande. Det första som gjordes var att sätta upp all nödvändig hårdvara och konfigurera den. IoT-gatewayen var i sin design snarlik Raspberry Pi vilket underlättade denna del av fasen då kunskap om dess system hade införskaffats i förstudiefasen.

Nästa steg när hårdvaran var på plats och den fungerade som den skulle var att ladda ner alla nödvändiga program. Då IoT-gatewayen inte hade någon desktop skapades en anslutning mellan en dator och IoT-gatewayen genom programmet PuTTY. Det var sedan via PuTTY möjligt att ladda ner program till IoT-gatewayen.

Databasen var det första som konstruerades då de andra delarna av examensarbetet behövde byggas upp runt en datakälla. Databasen skapades från programmet PHPMyAdmin. Insättning av data i databasen från Node-Red gjordes för att säkerställa att databasen fungerade som önskat. En dashboard som grafiskt visualiserar data från databasen togs sedan fram i Grafana.

För att säkerställa att TCP-noden fungerade som den skulle i Node-Red användes programmet Hercules för att skicka meddelanden till IoT-gatewayen från en dator.

Nästa steg blev att ta fram ett HMI där operatören ska ha möjlighet att skriva in information om processen. PLC-programmet som användes för TCP kommunikationen mellan PLCn och IoT-gatewayen modifierades för att implementeras på detta examensarbete.

När databas, dashboard, PLC och HMI var på plats och kommunikationen mellan de olika modulerna fungerade bra fördes diskussion med handledarna ifall det fanns fler funktioner som hade varit användbara att implementera. Det som togs upp då var om det var möjligt att:

- Ta fram en metod för att få fram användningsstatistik om IoT-gatewayen och redovisa den på en dashboard.
- Ändra IP-adress för de båda fysiska portarna på IoT-gatewayen från en operatörspanel i Node-Red.

Det som särskilde implementeringen av dessa funktioner jämfört med tidigare var att Docker användes för att ladda ner den nödvändiga programvaran.

3.5 Källkritik

Källorna [1-6, 13, 14] kommer från företag inom automationsbranschen, dock inte information direkt relaterad till deras produkter. Informationen som tagits från dessa källor är mer relaterade till ämnet automation och tekniken som produkterna bygger på som exempelvis programmeringsspråk.

[7, 8, 21, 22] Är källor från Siemens och Raspberry Pi om deras egna produkter. De anses trovärdiga då det är väletablerade företag och det ligger i deras intresse av att visa upp en rättvis bild av sina produkter för att inte skada sina rykten.

Källa [10] är från Microsoft Azure och innehåller allmän information om relationsdatabaser och inte specifikt om några produkter relaterade till Microsoft och anses därför trovärdig.

[9, 11, 12, 15-17] Är källor från företag och stiftelser som erbjuder fri programvara. De har inget eget vinstintresse för användandet av dessa program och de är väletablerade och kan därför anses vara trovärdiga.

Källorna [18, 20] är främst menade att förmedla information och utbildningar inom diverse ämnen.

[19] Internetstiftelsen är en organisation som har som mål att främja utvecklingen av internet. De handhar driften om alla svenska .se domäner och informationen de förmedlar på sin sida är allmänt om tekniker som används hos datorer, som i detta fallet är TCP.

4 Implementering

Det här kapitlet redogör för implementeringen av hårdvaran och mjukvaran som har använts under examensarbetet. Kravspecifikationen presenteras tillsammans med de problem som har uppstått under examensarbetet och hur de löstes.

4.1 Kravspecifikation

För att veta hur programmeringen skulle genomföras behövdes en kravspecifikation tas fram. Kravspecifikationen innehöll vad som skulle sparas om tillverkningsprocessen och vad som skulle presenteras på dashboarden.

Operatören ska ha möjlighet att i ett HMI skriva in:

- Artikelnummer.
- Produktnamn.
- Företag.

Dashboarden över takttiden ska presentera:

- Artikelnummer.
- Produktnamn.
- Företag.
- Medeltakttiden.
- Senaste takttiden.
- Antal producerade detaljer.

4.2 Konfigurering av hårdvaran

Leveransen av Simatic IOT2050 som har använts under examensarbetet var försenad på grund av halvledarbrist vilket gjorde att den ersattes under förstudiefasen med en liknande enhet. Det gjordes för att bli bekväm med både en enhet som är uppbyggd på ett liknande sätt samt med programmen som skulle användas. Då förstudiefasen under examensarbetet bestod av att bli bekväm med programvaran så bidrog det inte till några förseningar för tidsplanen. Valet landade på en Raspberry Pi då båda enheterna använder sig av Linux och har stöd för programmen som var tänkta att användas under examensarbetet. Se Tabell 2 för en jämförelse mellan enheterna.

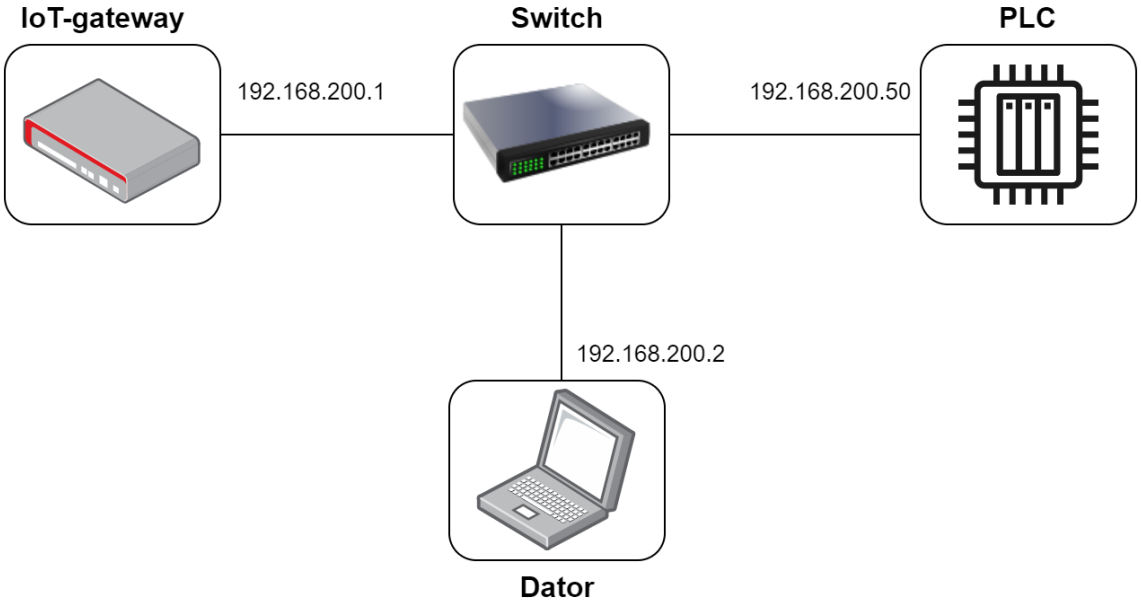
Tabell 2. Specifikationer för IOT2050 och Raspberry Pi 4.

Specifikationer	**IOT2050**[22]	**Raspberry Pi 4**[21]
Processor	ARM TI AM6548 HS	Quad core Cortex-A72 (ARM v8) 1.5GHz
Ram minne	2GB DDR4 RAM	1-8GB
Operativsystem	Debian	Debian finns som valmöjlighet

Övrig hårdvara som användes under examensarbetet var PLCn Siemens 1512SP-1 PN, spänningskälla, switch och nödvändiga kablar. Spänningskällan, IOT2050 och PLCn visas i Figur 13.

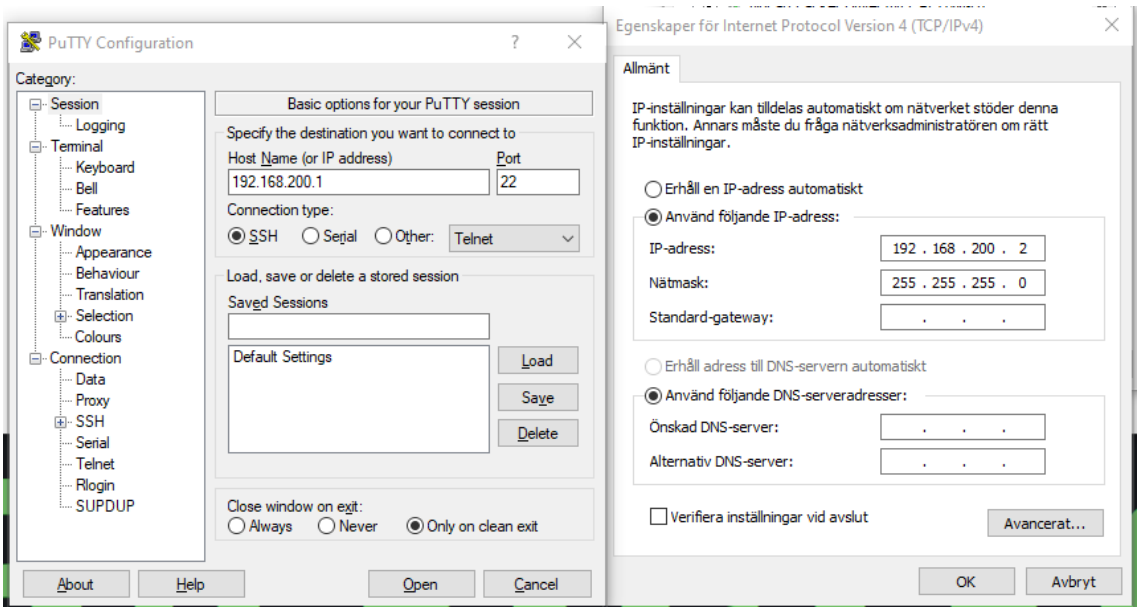
Figur 13. Hårdvaran som användes under examensarbetet. Från vänster, spänningskälla, IoT-gateway och PLC.

För att kunna upprätta kommunikationen mellan IoT-gatewayen, PLCn och datorn som användes under examensarbetet var det viktigt att enheternas IP-adresser var konfigurerade att tillhöra samma IP-nät. Det gjordes genom att de tre första delarna av IP-adressen på IoT-gatewayen, PLCn och datorns fysiska port var samma medan den sista delen var unik. Se Figur 14 för vilka IP-adresser som har använts på enheterna under examensarbetet.



Figur 14. IP-adresser som användes för IoT-gateway, PLC och datorns fysiska port.

För att ansluta till IoT-gatewayen via PuTTY från en dator fylldes IP-adressen för IoT-gatewayen in i översta kolumnen och SSH markerades. Se Figur 15 för PuTTY inställningarna till vänster och datorns portinställningar som gjordes till höger. Datorns port behövde ligga på samma nät som IoT-gatewayen för att det skulle vara möjligt att kommunicera med den från datorn. Efter att anslutningen via PuTTY var genomförd var det möjligt att logga in på IoT-gatewayen via PuTTY med valfri användare.



Figur 15. Anslutningen mellan dator och IoT-gatewayen. Programmet PuTTY till vänster med IP-Adress för IoT-gatewayen inskriven. Till höger syns IP-adressinställningar för datorns fysiska port.

4.3 Databas för takttid

Databasen som valdes ut för att spara information om tillverkningsprocessen var MariaDB då den är lik MySQL som tidigare har arbetats med under utbildningen. De båda har liknande syntax och använder sig av språket SQL för att skapa och modifiera databaser. Den största skillnaden är att MariaDB är framtagen med målet att alltid vara fri att använda medan MySQL ägs av företaget ORACLE vilket gör dess framtid som kostnadsfri osäker. Tabell 3 redovisar tabellerna och kolumnerna som databasen bestod av.

Tabell 3. Tabeller och kolumner i databasen för takttiden.

Tabeller	**Kolumner**
Product	• ID • Article_number • Product_name
Company	• ID • Company_name
Detail	• ID • Takt_time • Article_id • Company_id • Time_done

I tabellerna Product och Company är Article_number och Company_name unika nycklar. Unika nycklar användes för att undvika dubbletter i databasen då varje rad som infogades i kolumnen behövde vara unik.

Article_id och Company_id i tabellen Detail är foreign keys för de tillverkade produktens ID respektive företagets ID. Foreign keys användes för att länka ihop tabeller.

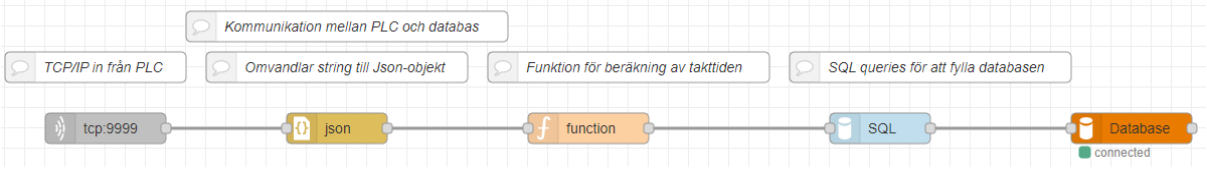
Alla ID kolumner är primary keys och autogenererade. Figur 16 illustrerar databasen i sin helhet.

Figur 16. Databasen för examensarbetet med tabellerna Product, Detail och Company.

För att administrera databasen användes PHPMyAdmin som grundade sig i att det behövdes ett administrationsverktyg för att snabbt kunna undersöka databasen vid felsökning. Från PHPMyAdmin går det att modifiera databaser utan att ha större kunskaper inom SQL då det erbjuder mallar att fylla i för att skapa nya tabeller.

4.4 Bearbetning av inkommande data från PLCn

Node-Red var ett program som beslutades av Automationsteknik AB att användas under examensarbetet. Det implementerades i IoT-gatewayen för att bearbeta inkommande data om den producerade detaljen från PLCn. Se Figur 17 för Node-Red flödet som togs fram under examensarbetet.



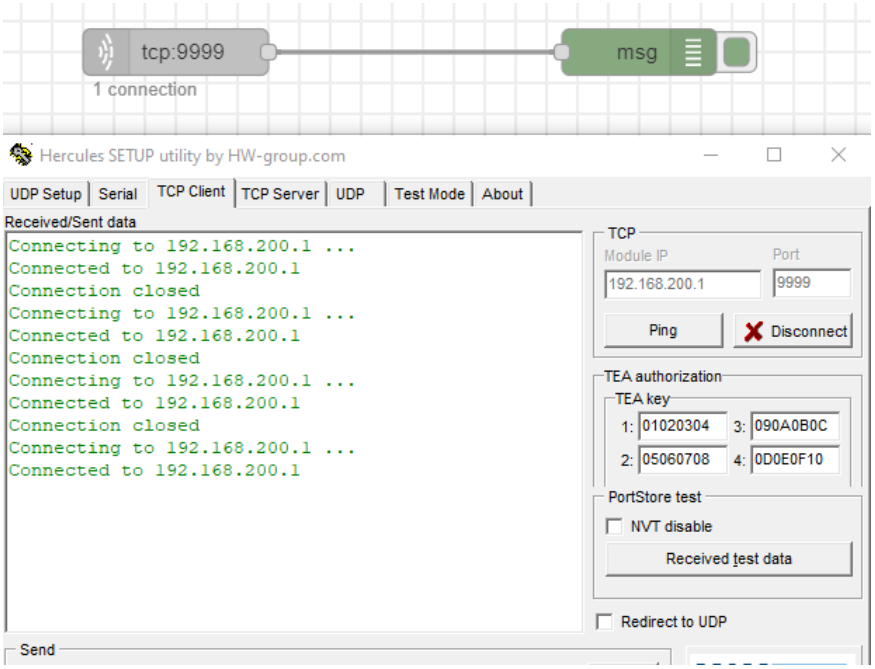
Figur 17. Flödet för kommunikationen mellan PLC och databasen.

Flödet bestod av:

- TCP-node.
- JSON-node.
- Function-node.
- SQL-node.

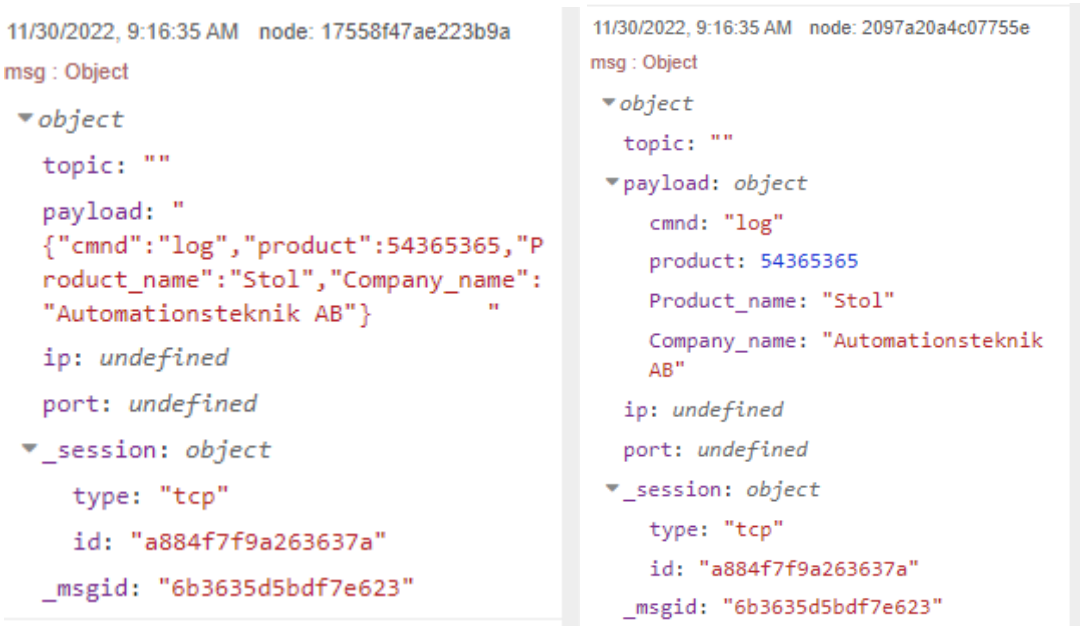
- Databas-node.

TCP-noden tog emot data från PLCn via TCP protokollet på en specifik port. Under examensarbetet användes port 9999. För att kontrollera att TCP anslutningen fungerade användes programmet Hercules för att tillfälligt upprätta en anslutning mellan en dator och Node-Red. I Figur 18 visas uppkopplingen som gjordes mellan Hercules och TCP-noden.



Figur 18. Hercules ansluten till Node-Red genom TCP-noden.

JSON-noden användes för att omvandla den inkommande datan från ett text-objekt till ett JSON-objekt. Det gjordes för att värdena från PLCn skulle användas som variabler i SQL-noden. Figur 19 visar hur datan såg ut före och efter den passerade JSON-noden.

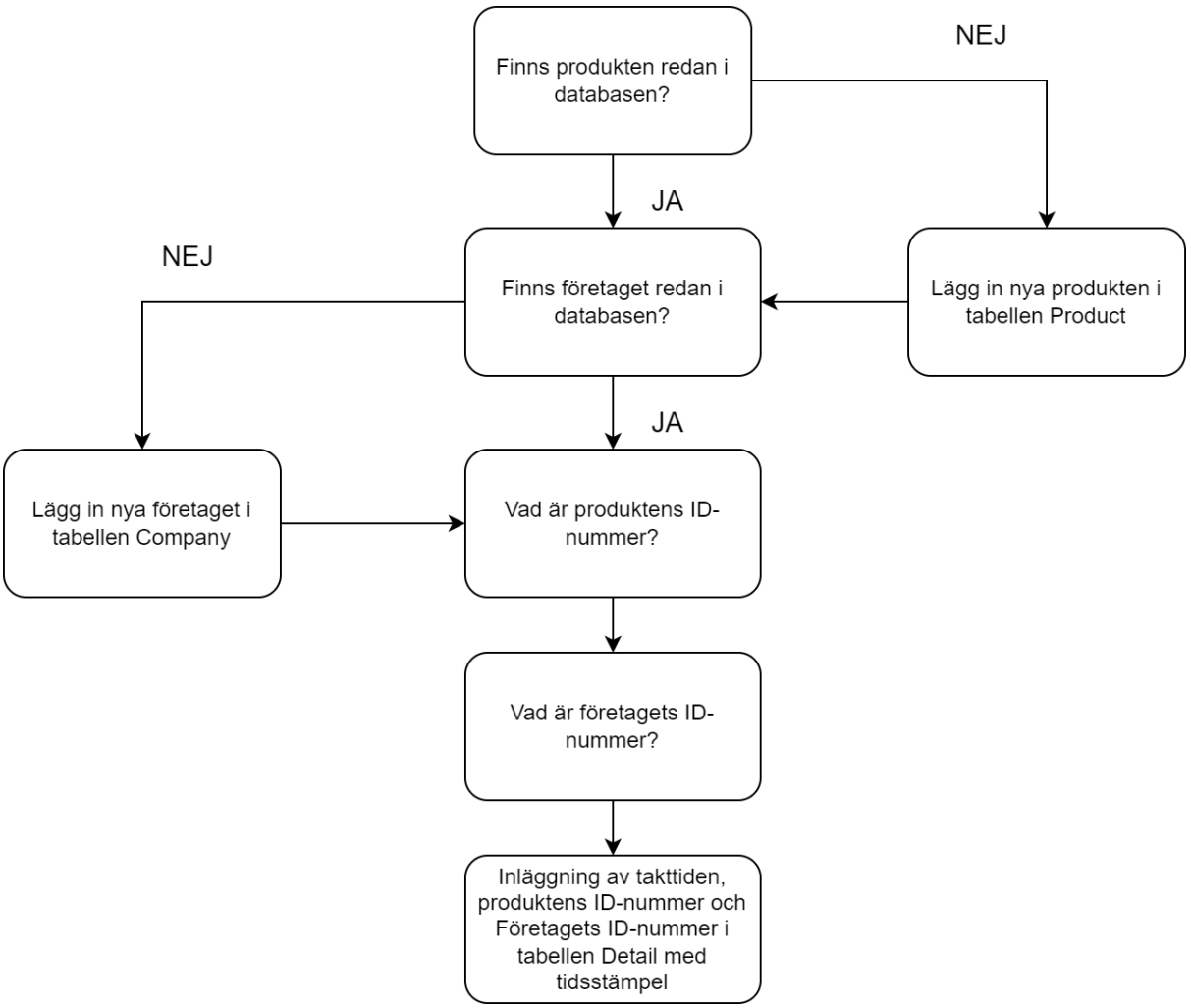


Figur 19. Datan före och efter JSON-noden där den omvandlades från ett text-objekt till ett JSON-objekt.

När datan hade ombildats till ett JSON-objekt förändrades payloaden och innehöll istället variablerna cmnd, product, product_name och company_name. De tre sista var värdena som skrevs in på HMI-skärmen och kunde nu användas som variabler i SQL-noden.

Innan datan kom fram till SQL-noden gick den igenom en functions-node där takttiden beräknades. Det gjordes genom att den föregående detaljens sluttid som tillverkades sparades i en variabel. Den detaljens sluttid blev på så sätt den senaste detaljens starttid. För att få fram takttiden beräknades tidsskillnaden mellan sluttiden och starttiden för detaljen. Den beräknade takttiden skickades sedan vidare till SQL-noden tillsammans med resten av värdena från JSON-noden.

I SQL-noden fanns SQL-satser vars funktion var att infoga värdena i databasen. Figur 20 visar ett flödesschema för hur koden i SQL-noden var uppbyggd. För att se koden i sin helhet se Appendix A. Alla förfrågningar i flödesschemat gjordes till databasen.



Figur 20. Flödesschema över koden i SQL-noden.

4.5 Dashboard för takttiden

För att ta fram en dashboard för grafisk visualisering av takttiden användes Grafana. Det första som gjordes var att konfigurera vilken datakälla Grafana skulle inhämta information från.

Nästa steg var att ta fram panelerna för att bygga upp en dashboard. Det gjordes genom att implementera SQL-satser för att göra förfrågningar till databasen. Se Figur 21 för en förfrågan som gjordes för att få fram takttiden och medeltaktiden från tabellen Detail i databasen. Figur 21 består av:

1. Tidsdiagram över takttiden. Grön linje är varje enskild takttid och gul linje är medeltakttiden över ett förbestämt antal detaljer.
2. Konfigurering av vilken databas informationen ska inhämtas från.
3. SQL-förfrågningar till databasen.

Figur 21. Panel över takttiden. SQL-satsen är förfrågan till databasen.

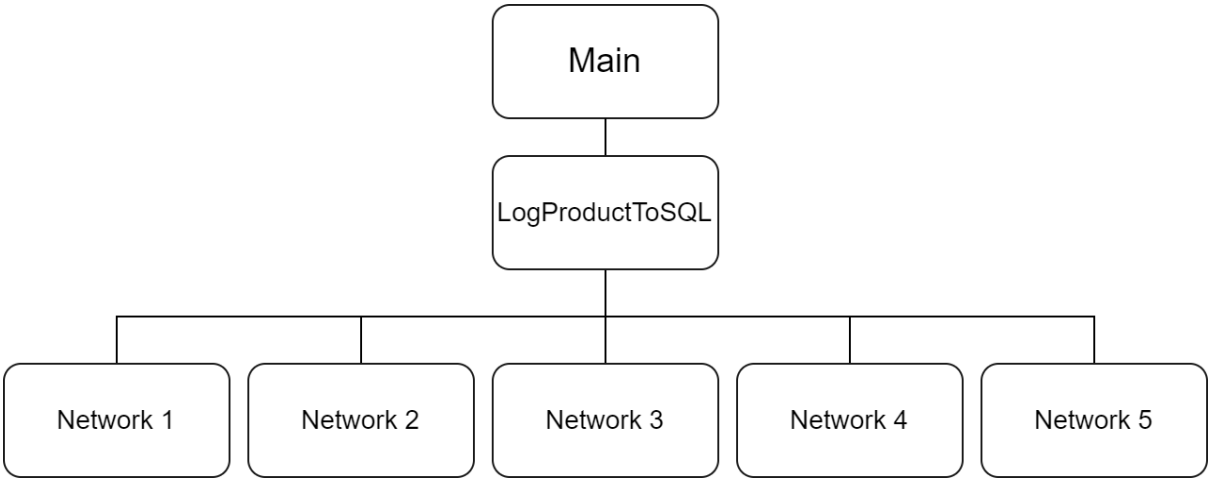
Fler paneler implementerades sedan för att grafiskt visualisera det som nämndes under rubriken kravspecifikation i kapitel 4.

4.6 PLC-programmering och HMI

PLC-programmet som användes under examensarbetet var ett sedan tidigare framtaget program som modifierades för att implementeras på examensarbetet. Programmet var skrivet i TIA Portal och innehöll ett Main block som är högst upp i hierarkin av programmet. I Main finns ett block som heter LogProductToSQL som är uppbyggt av fem olika nätverk som har olika funktioner. Tabell 4 sammanfattar de olika nätverkens funktioner och Figur 22 illustrerar hierarkin i programmet. För att se alla nätverk i sin helhet se Appendix B.

Tabell 4. Nätverken som PLC-blocket LogProductToSQL byggdes upp på.

Nätverk	**Funktion**
Nätverk 1	Sätter IP-adressen för mottagaren av data. IP-adressen skrivs in på blocket LogProductToSQL i programmet TIA Portal.
Nätverk 2	Samlar all data som skrivits in i HMI till ett text-objekt.
Nätverk 3	Startar sändningen av data.
Nätverk 4	Uppbyggt av ett TSEND-block som gör sändning över TCP möjligt.
Nätverk 5	Nollar minnet av programmet när sändningen är färdig.



Figur 22. Hierarkin för PLC-programmet som har använts under examensarbetet.

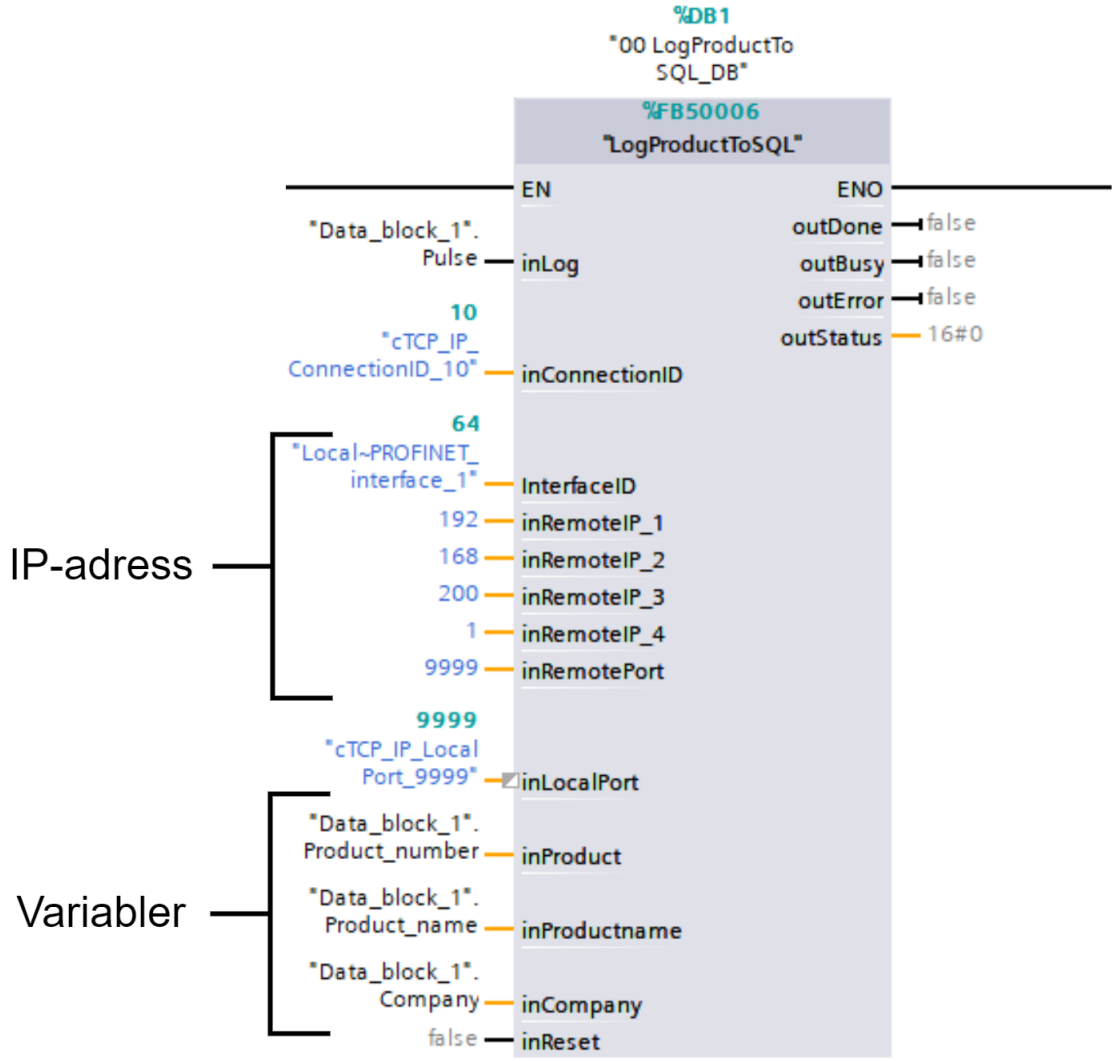
Det som har gjorts för att anpassa programmet till examensarbetet var att skapa ett nytt datablock med variabler som sparade information om produkten som tillverkades. Figur 23 visar de nya variablerna för att spara produktnamn, artikelnummer, företag och för att trigga igång sändningen av data.

Data_block_1 (snapshot created: 11/16/2022 1:34:41 PM)

	Name	Data type	Start value	Retain	Accessible f...	Writa...	Visible in ...	Setpoint
1	Static							
2	Pulse	Bool	false					
3	Product_number	LInt	0					
4	Product_name	String	''					
5	Company	String	''					

Figur 23. Datablocket som användes under examensarbetet.

Nätverk 2 modifierades så att text-objektetet som skickade vidare data innehöll de nya variablerna. Variablerna sammankopplades sedan med ingångarna på blocket LogProductToSQL där även mottagarens (IoT-gatewayen) IP-adress skrevs in, vilket visas i figur 24.



Figur 24. PLC-blocket LogProductToSQL.

Ett HMI togs fram där operatören har möjlighet att skriva in artikelnummer, produktnamn och vilket företag produkten tillverkas på. Även en knapp för att simulera tillverkningen inkluderades. När knappen för simulering är intryckt blir variabeln InLog från blocket LogProductToSQL sann och när den släpps blir den falsk, vilket simulerar triggern som är tänkt att representera en producerad detalj.

4.7 Databasen för användningsstatistik

För att samla in och visualisera användningsstatistiken om IoT-gatewayen användes programmet Prometheus. Det framstod som ett användarvänligt alternativ för informationsinhämtning om system då det fanns gott om färdiga dashboards att ladda ner i

Grafana som utan några modifikationer kunde sammankopplas med en Prometheus. Prometheus gjorde det även enkelt att modifiera hur länge information skulle sparas i databasen.

För att samla in informationen åt Prometheus användes tillägget Node Exporter. För att Prometheus skulle veta vart data skulle hämtas från konfigurerades en textfil som innehöll information om hur ofta informationen skulle inhämtas och vart den skulle inhämtas från vilket illustreras i Figur 25. Vart den skulle inhämtas är portarna som Prometheus och Node Exporters använde sig av.

Figur 25. Konfigureringsfilen. Scrape_intervall är vilket intervall som information inhämtades och scrape_configs är vart data skulle inhämtas från.

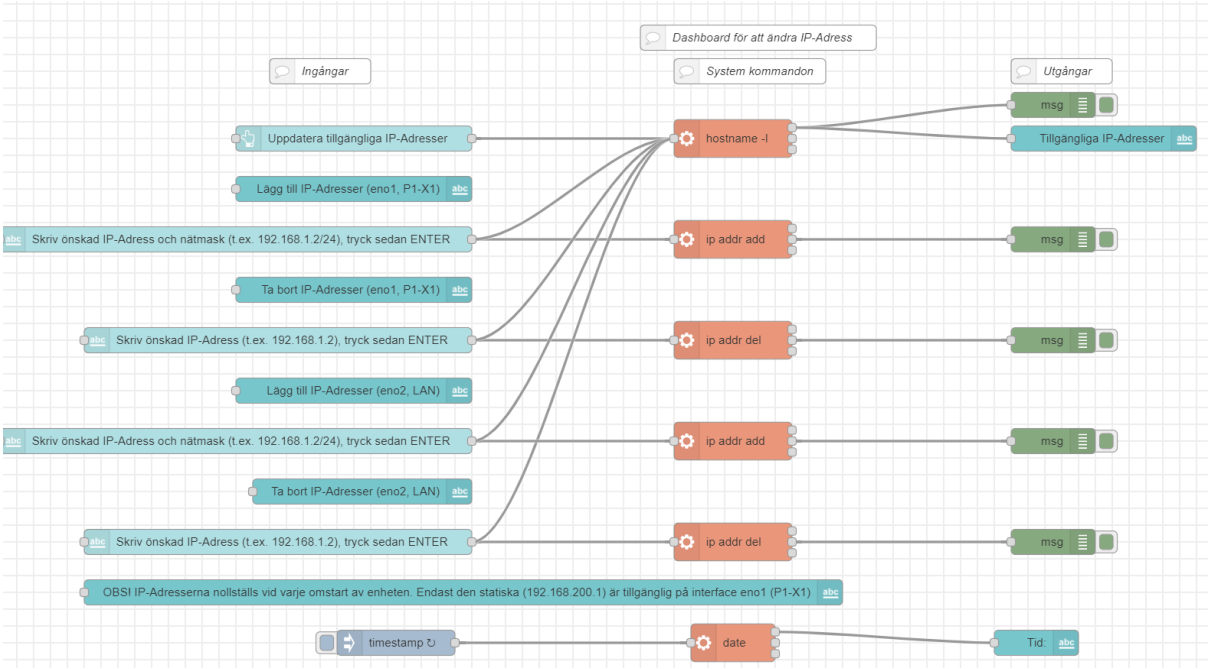
För att grafiskt visualisera den insamlade informationen användes programmet Grafana. En färdig mall vid namn “Node Exporter Full” användes och det som konfigurerades var vilken datakälla Grafana skulle inhämta informationen från.

4.8 Operatörspanel för modifiering av IP-adresser

Operatörspanelen för modifiering av IP-adresser togs fram i Node-Red och byggdes upp i ett flöde som innehöll noderna:

1. EXEC
2. Debug
3. Button
4. Text-input
5. Text
6. Timestamp

Noden Text-input användes för att operatören skulle ha möjlighet att skriva in den nya IP-adressen för IoT-gatewayen. Button användes för att uppdatera en lista över tillgängliga IP-adresser. Text användes för att redovisa tillgängliga IP-adresser och vad den nuvarande tiden var. EXEC användes för att utföra kommandon i systemets kommandotolk för att byta IP-adresser. Timestamp användes för att varje sekund skicka ett kommando för att uppdatera tiden. Figur 26 illustrerar Node-Red flödet för IP-adress operatörspanelen i sin helhet och Tabell 5 presenterar Kommandona som utfördes i EXEC-noderna.



Figur 26. Node-Red flödet för IP-adress operatörspanelen.

Tabell 5. Kommandon för EXEC-noderna.

Kommandon	**Funktion**
Hostname -I	Visa tillgängliga IP-adresser.
ip addr add “Angiven IP-adress” dev eno1	Lägger till en IP-adress på port eno1 (WAN).
ip addr del “Angiven IP-adress” dev eno1	Tar bort en IP-adress på port eno1 (WAN).
ip addr add “Angiven IP-adress” dev eno2	Lägger till en IP-adress på port eno2 (LAN).
ip addr del “Angiven IP-adress” dev eno2	Tar bort en IP-adress på port eno2 (LAN).
date	Uppdaterar tiden.

4.9 Implementering på en tillverkningsprocess

Under examensarbetet har inte den framtagna metoden implementerats på en PLC-styrd tillverkningsprocess på grund av tidsbrist, men om det ska göras vid ett senare tillfälle ska följande genomföras:

1. Koppla samman IoT-gatewayen och PLCn.
2. Implementera PLC-programmet som används under examensarbetet på PLCn som styr tillverkningsprocessen.
3. Skapa en trigger varje gång en detalj är färdigtillverkad som kopplas till ingången inLog på PLC-blocket LogProductToSQL.
4. Konfigurera IP-adress på IoT-gatewayen med hjälp av den framtagna operatörspanelen för IP-adress modifiering.

4.10 Problem som uppstått under examensarbetet

4.10.1 Beräkning av medeltiden i Grafana

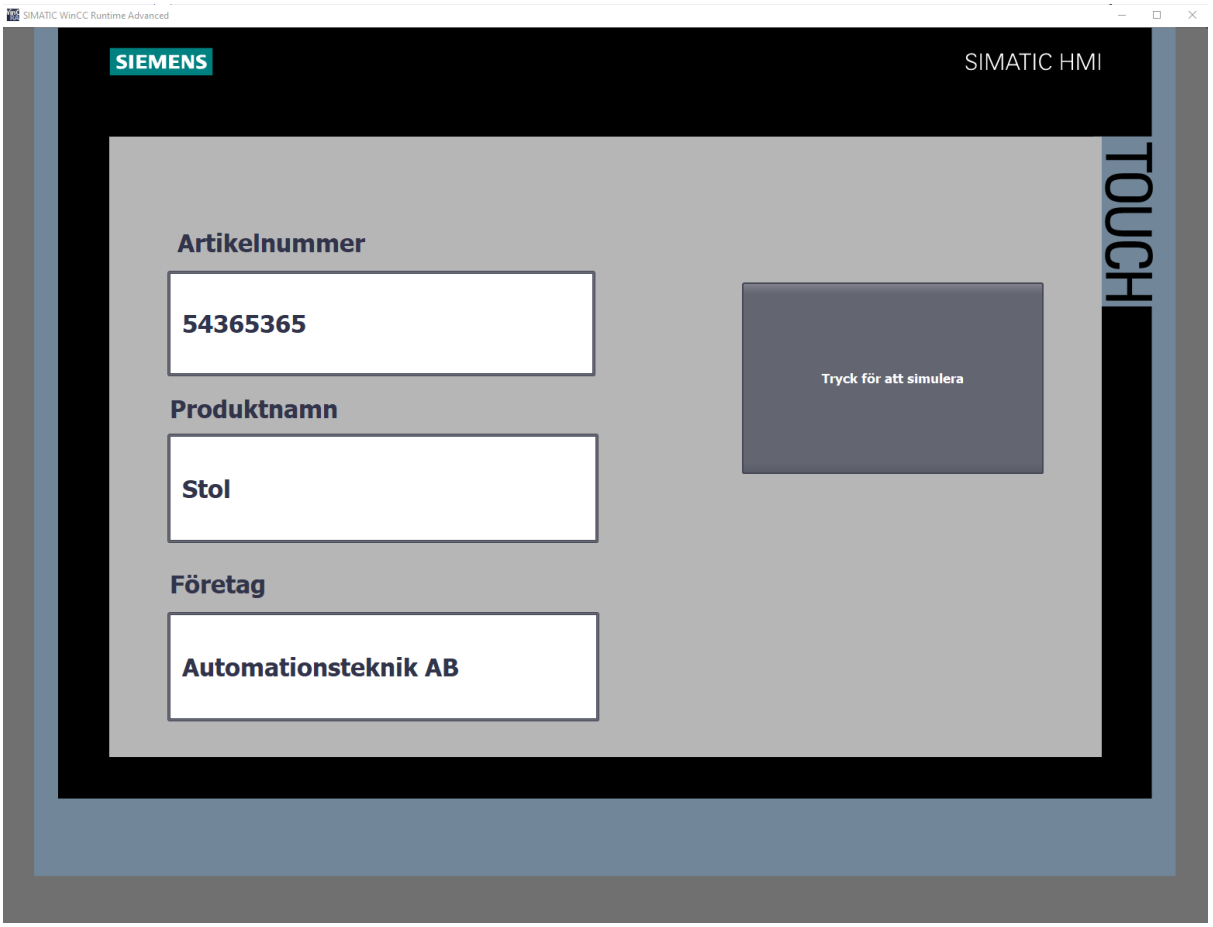
I programmet Grafana var det ej möjligt att beräkna medeltakttiden vilket var tanken från början. Detta då Grafana endast kan utföra enklare beräkningar och beräkning av medeltiden under en längre tid blev för avancerat för programmet. Lösningen blev att istället göra en kolumn för takttiden i tabellen Detail och använda en funktion för att beräkna takttiden i Node-Red. Därefter infogades takttiden i databasen tillsammans med artikelnummer, produktnamn och företag. Medeltakttiden behövdes då endast visualiseras i Grafana och ej göra några beräkningar.

5 Resultat

Det här kapitlet presenterar resultatet av examensarbetet. Det har delats upp i HMI, dashboard och operatörspanel.

5.1 Operatörs HMI

Ett HMI som grafiskt användargränssnitt mellan operatören och PLCn togs fram vilket visas i Figur 27. I HMI har operatören möjlighet att skriva in produkten som ska tillverkas: artikelnummer, produktnamn och företag. När operatören trycker på enter efter att informationen är inskriven i panelen ändras variablerna som skapades för att spara de inskrivna värdena. En knapp för att simulera tillverkningen av en detalj implementerades på HMI-skärmen för att testa TCP kommunikationen med IoT-gatewayen.



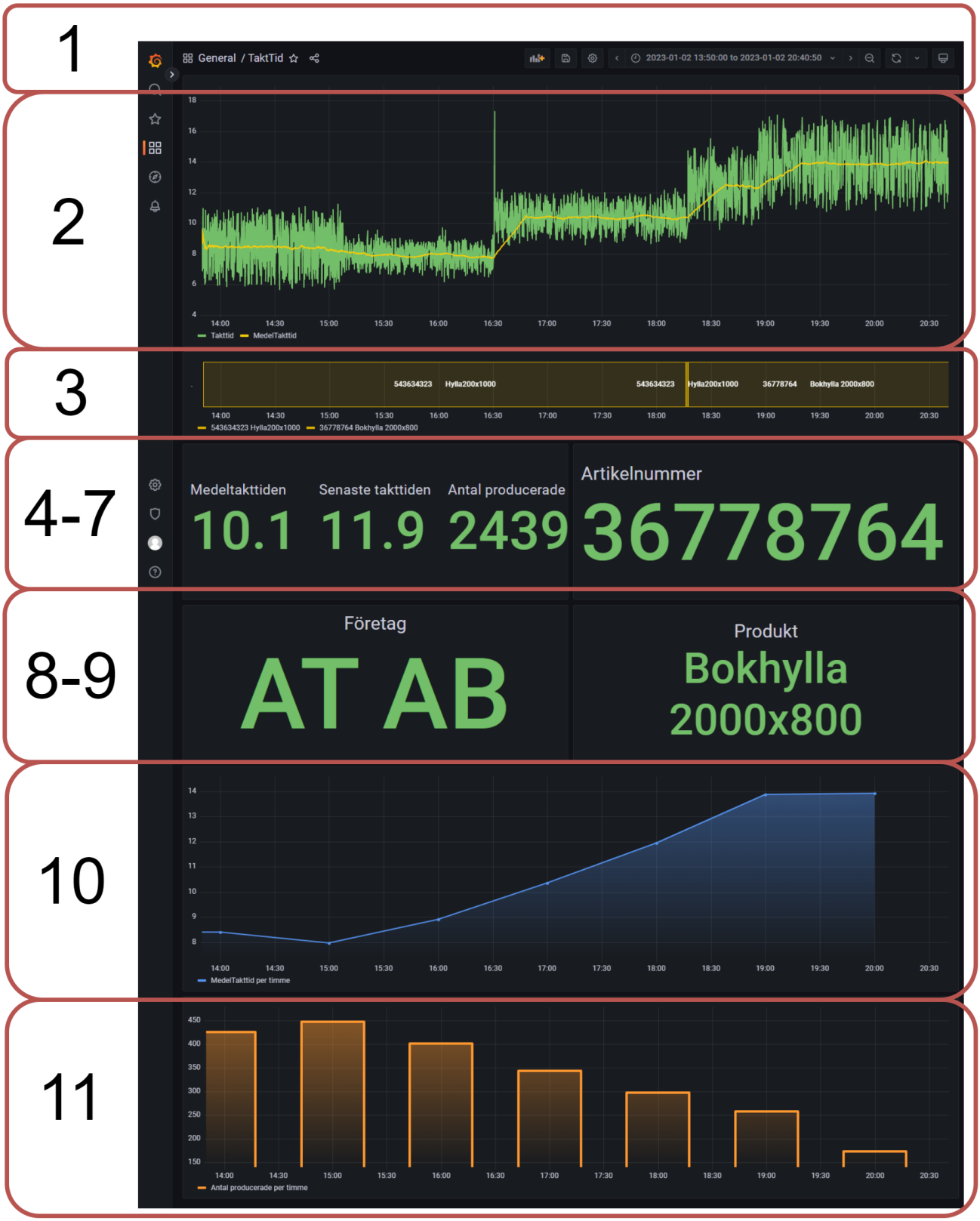
Figur 27. HMI-skärm med kolumner för artikelnummer, produktnamn och företag.

5.2 Dashboards för Takttid och användningsstatistik

Under examensarbetet har två dashboards tagits fram i Grafana, en relaterad till takttiden och den andra relaterad till användningsstatistik om IoT-gatewayen.

Dashboarden för takttid som visas i figur 28 innehåller paneler för:

1. Meny för vilken dashboard som ska visas och vilket tidsintervall som ska visas.
2. Takttiden och medeltakttiden i tidsserie.
3. Tillverkad produkt i tidsserie.
4. Medeltakttiden.
5. Senaste takttiden.
6. Antal producerade detaljer.
7. Artikelnummer.
8. Företaget som tillverkar detaljerna.
9. Produktnamn.
10. Medeltakttiden per timme.
11. Antal producerade detaljer per timme.

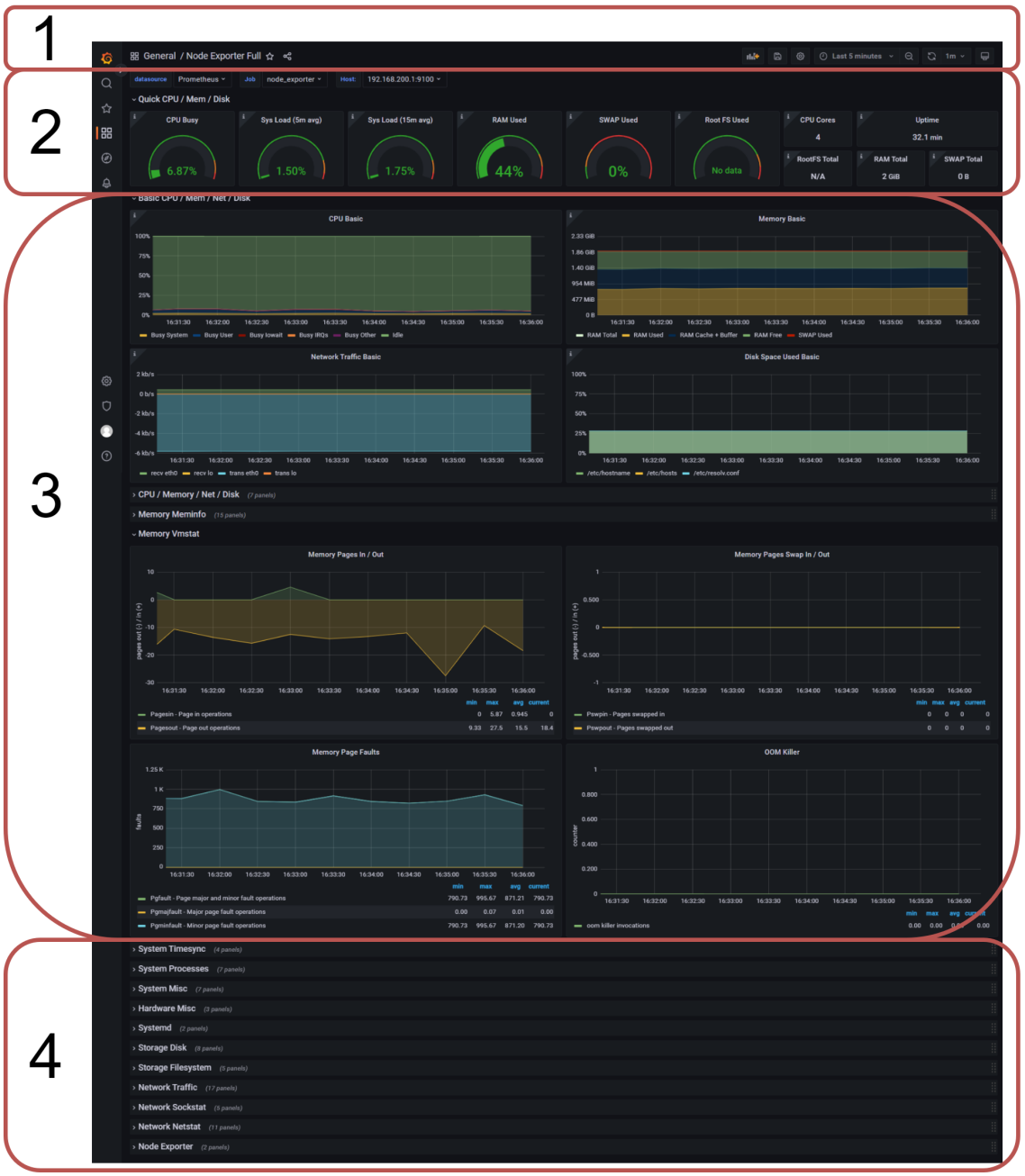


Figur 28. Dashboard över takttiden.

Appendix C presenterar hur koden bakom samtliga paneler i dashboarden i figur 28 är uppbyggda.

Figur 29 visar dashboarden som används för att visualisera användningsstatistik om IoT-gatewayen. Det är en mall som implementerats på examensarbetet och den innehåller bland annat:

1. Meny för vilken dashboard som ska visas och vilket tidsintervall som ska visas.
2. Paneler för indikering hur mycket CPU, RAM och minne används.
3. Övergripande paneler för hur mycket nätverk som används och data som överförs på IoT-gatewayens portar.
4. Utfällbara stycken för paneler som mäter data i detalj.

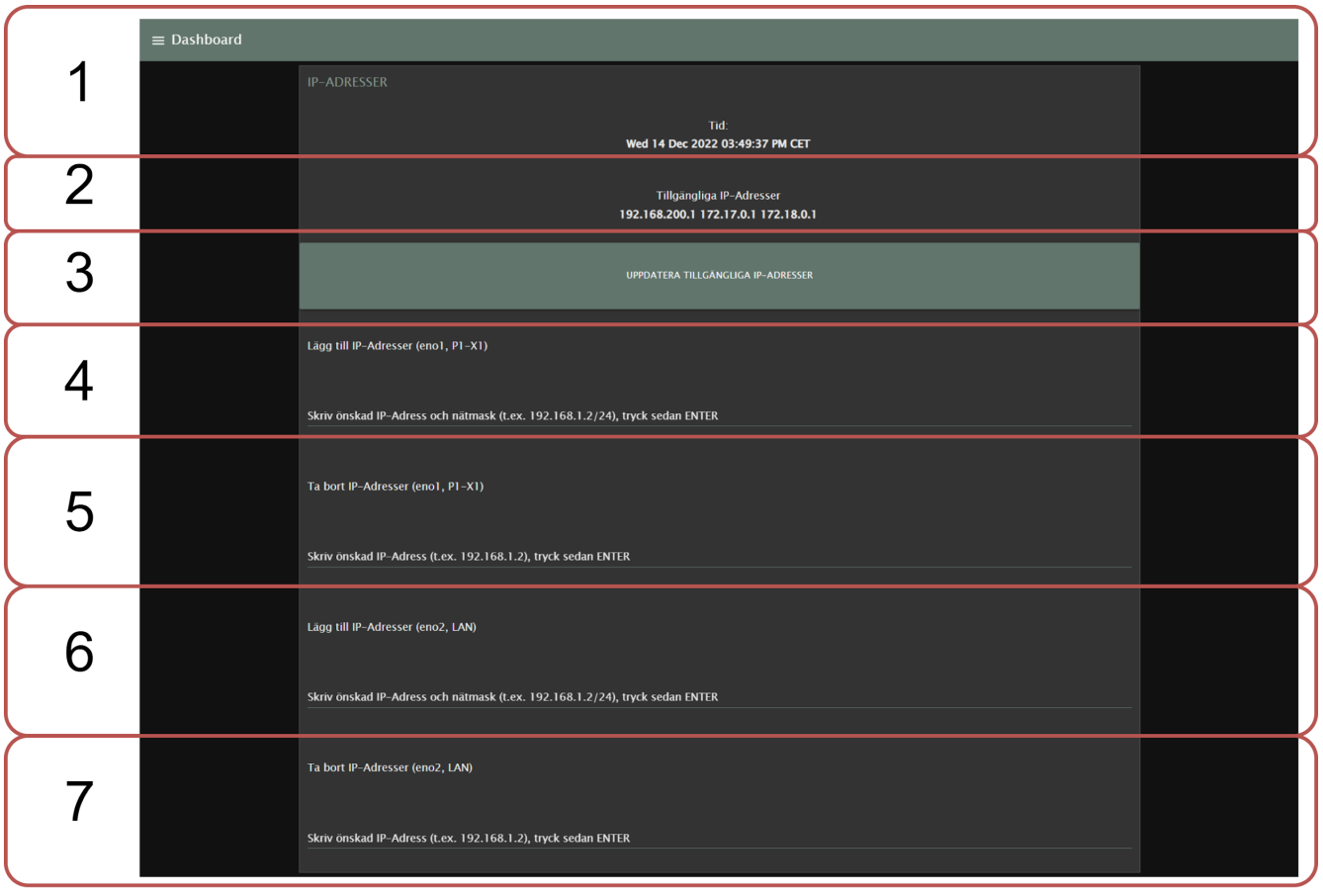


Figur 29. Dashboard över användningsstatistik på IoT-gatewayen.

5.3 IP-adress operatörspanel

Figur 30 visar operatörspanelen för att modifiera IP-adressen på IoT-gatewayen som togs fram under examensarbetet för att förenkla förflyttningen av IoT-gatewayen mellan PLC-styrda tillverkningsprocesser. Operatörspanelen innehåller:

1. En rad innehållande datum och tid.
2. En rad med tillgängliga IP-adresser.
3. Knapp som uppdaterar IP-adresslistan.
4. Textrad för att lägga till ny IP-adress på port wan.
5. Textrad för att ta bort IP-adress på port wan.
6. Textrad för att lägga till ny IP-adress på port Lan.
7. Textrad för att ta bort IP-adress på port Lan.



Figur 30. Operatörspanel för modifiering av IP-adressen på IoT-gatewayen.

Efter att en IP-adress skrivits in trycker operatören på enter för att genomföra kommandot. IP-adressen ändras direkt och ingen omstart av programmet eller IoT-gatewayen krävs.

6 Slutsats

Det här kapitlet presenterar de slutsatser som dras av examensarbetet. Även en reflektion över etiska aspekter och framtida utvecklingsmöjligheter diskuteras.

Examensarbetet var menat att besvara frågorna:

1. Vilka program krävs för att ta fram en dashboard för takttiden?
2. Vilka fördelar finns av att spara takttiden under en längre tid?
3. Hur ska mjukvaran och hårdvaran utformas för att bli universell och göra det möjligt att snabbt kunna mäta takttiden hos en ny maskin?

För att ta fram en dashboard behövs ett program för att grafiskt visualisera data. Under examensarbetet användes Grafana men ett annat alternativ hade varit att använda sig av Node-Red för att bygga upp en dashboard.

Fördelarna med att spara takttiden under en längre tid är att det går att testa ifall någon modifiering på tillverkningsprocessen lönar sig långsiktigt då takttiden kan jämföras med tidigare takttider för samma produkt. Förändringar i takttiden kan även vara ett resultat av slitage som kan upptäckas vid ett tidigt skede.

För att göra examensarbetet portabelt behövs en IoT-gateway, PLC, spänningskälla och en sensor som ska känna av de färdigtillverkade detaljerna på tillverkningsprocessen. IoT-gateway, PLC och givare fästes på en flyttbar monteringsskiva. Sensorn fästs i slutet av produktionslinjen. Den kan då enkelt förflyttas mellan olika maskiner då själva PLCn och IoT-gatewayen inte har någon permanent placering vid maskinen och program inte behöver föras över till PLCn som styr tillverkningsprocessen.

En deluppgift under examensarbetet har varit att undersöka IoT-gatewayen som använts på uppdrag av Automationsteknik AB. Slutsatsen som kan dras av IOT2050 är att det är en IoT-gateway som är lätt att arbeta med. Den är lik Raspberry Pi så innehar användaren kunskap om den är det inga problem att börja arbeta med en IOT2050.

Beskrivningen av IOT2050 som en industriell Raspberry Pi är sanningsenlig och inga större skillnader under detta examensarbete har upptäckts mjukvarumässigt samt prestandamässigt. Hårdvarumässigt upplevs IOT2050 mer kraftfull i sin konstruktion och har en kylfläns monterad vilket håller nere temperaturen vid drift. Det kan möjligtvis göra den mer lämplig för industriella tillämpningar men då varken IOT2050 eller Raspberry Pi har fått utså längre drift i en industriell miljö under examensarbetet kan denna slutsats ej dras i den här rapporten. Det kan även vara så att skillnaden mellan de båda enheterna först visar sig vid användning inom större projekt än vad detta examensarbete utgör.

6.1 Reflektion över etiska aspekter

“Ingenjören bör i sin yrkesutövning känna ett personligt ansvar för att tekniken används på ett sätt som gagnar människa, miljö och samhälle.” — Sveriges ingenjörers hederskodex.

Examensarbetets mål var att underlätta vid utveckling av tillverkningsprocesser och kan det bidra till att minska antalet timmar som produktionslinjen behöver vara i drift för att mäta takttiden hade det varit gynnsamt för miljön och för Sveriges elnät. En stor tillverkningsprocess kan göra av med stora mängder el under korta perioder.

6.2 Framtida utvecklingsmöjligheter

Då en implementering på en fysisk tillverkningsprocess inte gjordes under examensarbetet hade nästa steg varit att göra det.

En framtida utvecklingsmöjlighet är att använda flera sensorer för att på så sätt mäta flera mindre takttider över olika delar på produktionslinjen. På så sätt hade flaskhalsar i produktionen kunnat upptäckas och åtgärdas.

En annan utvecklingsmöjlighet hade varit att med en mer permanent implementation kan takttiden användas för att upptäcka när produktionslinjen behöver service och när den börjar bli sliten genom att larma när takttiden ökar ovanligt mycket.

7 Referenser

[1] AMCI. What is a PLC?. 2022. [Online]. Tillgänglig: https://www.amci.com/industrial-automation-resources/plc-automation-tutorials/what-plc/ (Hämtad: 16 Oktober 2022).

[2] Induo AB. Analoga I/O, digitala I/O eller pulser?. 2022. [Online]. Tillgänglig: <https://www.induo.com/b/analoga-io-digitala-io-pulser-guide/> (Hämtad 25 Oktober 2022).

[3] RealPars. What are the Most Popular Types of PLC Programming Languages?. 2018. [Online]. Tillgänglig: https://realpars.com/plc-programming-languages/ (Hämtad 25 Oktober 2022).

[4] Unitronics. What is PLC ? Programmable Logic Controller - Unitronics. 2018. [Online]. Tillgänglig: https://www.unitronicsplc.com/what-is-plc-programmable-logic-controller/ (Hämtad 23 Oktober 2022).

[5] COPA-DATA. Vad är HMI - human-machine interface (Människa-maskingränssnitt). 2022. [Online]. Tillgänglig: https://www.copadata.com/sv/produkter/zenon-software-platform/visualisering-kontroll/vad-aer-hmi-human-machine-interface-maenniska-maskingraenssnitt-copa-data/ (Hämtad 18 November 2022).

[6] IXON. PLC Routers and IoT Gateways explained - VPN, Edge, IoT. 2020. [Online]. Tillgänglig: https://www.ixon.cloud/knowledge-hub/vpn-edge-iot-different-type-of-plc-routers-and-gateways-explained (Hämtad 25 Oktober 2022).

[7] Siemens. [Online]. Tillgänglig: https://new.siemens.com/global/en/products/automation/pc-based/iot-gateways/simatic-iot2050.html (Hämtad 26 Oktober 2022).

[8] Siemens. [Online]. Tillgänglig: https://new.siemens.com/global/en/products/automation/industry-software/automation-software/tia-portal.html (Hämtad 2 November 2022).

[9] MariaDB Foundation. MariaDB in brief. 2022. [Online]. Tillgänglig: https://realpars.com/plc-programming-languages/ (Hämtad 18 November 2022).

[10] Microsoft Azure. Vad är en relationsdatabas? 2022. [Online]. Tillgänglig: https://azure.microsoft.com/sv-se/resources/cloud-computing-dictionary/what-is-a-relational-database/#whatis (Hämtad 18 November 2022).

[11] PHPMyAdmin. Bringing MySQL to the web. 2022. [Online]. Tillgänglig: https://www.phpmyadmin.net/ (Hämtad 19 November 2022).

[12] Prometheus . What is Prometheus? 2022. [Online]. Tillgänglig: https://prometheus.io/docs/introduction/overview/ (Hämtad 18 November 2022).

[13] S. Sarawagi. What is Grafana? why use it? everything you should know about it. 2022. [Online]. Tillgänglig: https://realpars.com/plc-programming-languages/ (Hämtad 18 November 2022).

[14] Impulse embedded Limited. What is Node-Red? [Online]. Tillgänglig: https://www.impulse-embedded.co.uk/info/what-is-node_red.htm (Hämtad 18 November 2022).

[15] OpenJS Foundation. Node-Red. 2022. [Online]. Tillgänglig: https://nodered.org/ (Hämtad 18 November 2022).

[16] SSH. PuTTY Home - Free Downloads, Tutorials, and How-Tos. 2022. [Online]. Tillgänglig: https://www.ssh.com/academy/ssh/putty (Hämtad 18 November 2022).

[17] HW Group. Hercules SETUP utility. [Online]. Tillgänglig: https://www.hw-group.com/software/hercules-setup-utility (Hämtad 18 November 2022).

[18] V. Chaturvedi. What is Docker & Docker Container: A deep dive into Docker !. 2021. [Online]. Tillgänglig: https://www.edureka.co/blog/what-is-docker-container (Hämtad 18 November 2022).

[19] Internetstiftelsen. TCP. 2021. [Online]. Tillgänglig: https://internetkunskap.se/artiklar/ordlista/tcp/ (Hämtad 18 November 2022).

[20] ScienceDirect. Three-Way Handshake. [Online] Tillgänglig: https://www.sciencedirect.com/topics/computer-science/three-way-handshake (Hämtad 23 November 2022).

[21] Raspberry Pi. Raspberry Pi 4 Tech Specs. [Online] Tillgänglig: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/ (Hämtad 8 December 2022).

[22] Siemens. SIMATIC IOT2050. [PDF] Tillgänglig : https://support.industry.siemens.com/cs/attachments/109779016/iot2050_operating_instructions_en_en-US.pdf?download=true (Hämtad 8 December 2022).

8 Appendix

A. Node-Red funktioner

Funktions-noden i Node-Red.

```
//Ta in en ny tidsstämpel och spara den föregående
msg.payload.c = new Date().getTime();

var tidigare = flow.get("tidigarevarde") || msg.payload.c

flow.set("tidigarevarde", msg.payload.c);

//Jämför tidigare sluttid med nuvarande sluttid
var skillnad = msg.payload.c -tidigare;
msg.payload.c=skillnad/1000;

//Lösning för att inte fylla databasen med Null värden för product
if ( msg.payload.product == null && msg. payload.Product_name == null){
msg.payload.product=0;
}

return msg;
```

SQL-noden i Node-Red.

```
SET @Ano =?;
SET @Aname =?;
SET @Company =?;
SET @taktTime =?;

-- Inläggning i tabellen product

INSERT IGNORE INTO Product (Article_number, Product_name)
VALUES(@Ano, @Aname)
ON DUPLICATE KEY UPDATE Product_name=@Aname;

-- Inläggning i tabellen Company

INSERT IGNORE INTO Company (Company_name)
VALUES(@Company);
```

```
-- Få fram vilket id det valda artikelnumret har

SET @var3 = (
SELECT id
FROM Product
WHERE Article_number =@Ano OR Product_name=@Aname
ORDER by id DESC
limit 1
);

-- Få fram vilket id det valda Företaget har

SET @var4 = (
SELECT id
FROM Company
WHERE Company_name = @Company
ORDER by id DESC
limit 1
);

-- Inläggning i tabellen Detail

INSERT INTO Detail (takt_time, Article_id, Company_id)
VALUES(@taktTime, @var3, @var4);



FROM Company
WHERE Company_name = @Company
ORDER by id DESC
limit 1
);

-- Inläggning i tabellen Detail

INSERT INTO Detail (Cycle_Time, Article_id, Company_id)
VALUES(@CycleTime, @var3, @var4);
```

B. PLC-kod

Variablerna i LogProductToSQL-blocket.

00 LogProductToSQL_DB (snapshot created: 11/16/2022 1:34:41 PM)

	Name	Data type	Start value	Retain	Accessible f...	Writa...	Visible in ...	Setpoint
1	Input			☐	☐	☐	☐	☐
2	inLog	Bool	false	☐	☑	☑	☑	☐
3	inConnectionID	CONN_OUC	16#0	☐	☑	☑	☑	☐
4	InterfaceID	HW_ANY	0	☐	☑	☑	☑	☐
5	inRemoteIP_1	UInt	0	☐	☑	☑	☑	☐
6	inRemoteIP_2	UInt	0	☐	☑	☑	☑	☐
7	inRemoteIP_3	UInt	0	☐	☑	☑	☑	☐
8	inRemoteIP_4	UInt	0	☐	☑	☑	☑	☐
9	inRemotePort	UInt	0	☐	☑	☑	☑	☐
10	inLocalPort	UInt	0	☐	☑	☑	☑	☐
11	inProduct	LInt	0	☐	☑	☑	☑	☐
12	inProductname	String	''	☐	☑	☑	☑	☐
13	inCompany	String	''	☐	☑	☑	☑	☐
14	Output			☐	☐	☐	☐	☐
15	outDone	Bool	false	☐	☑	☑	☑	☐
16	outBusy	Bool	false	☐	☑	☑	☑	☐
17	outError	Bool	false	☐	☑	☑	☑	☐
18	outStatus	Word	16#0	☐	☑	☑	☑	☐
19	InOut			☐	☐	☐	☐	☐
20	inReset	Bool	false	☐	☑	☑	☑	☐
21	Static			☐	☐	☐	☐	☐
22	TSEND_C_Instance	TSEND_C		☐	☑	☑	☑	☑
23	DATA	Array[0..100] of Char		☐	☑	☑	☑	☑
24	sData	String	''	☐	☑	☑	☑	☐
25	cnt	UInt	0	☐	☑	☑	☑	☐
26	connect	TCON_IP_v4		☐	☑	☑	☑	☐
27	send	Bool	false	☐	☑	☑	☑	☑
28	P_TRIG_Log	Bool	false	☐	☑	☑	☑	☐

Network 1.

Network 1: Connection data

Comment

```
#connect.InterfaceId := #InterfaceID;
#connect.ID := #inConnectionID;
#connect.ConnectionType := 11;
#connect.ActiveEstablished := TRUE;

#connect.RemoteAddress.ADDR[1] := UINT_TO_BYTE(#inRemoteIP_1);
#connect.RemoteAddress.ADDR[2] := UINT_TO_BYTE(#inRemoteIP_2);
#connect.RemoteAddress.ADDR[3] := UINT_TO_BYTE(#inRemoteIP_3);
#connect.RemoteAddress.ADDR[4] := UINT_TO_BYTE(#inRemoteIP_4);
#connect.RemotePort := #inRemotePort;
#connect.LocalPort := #inLocalPort;

```

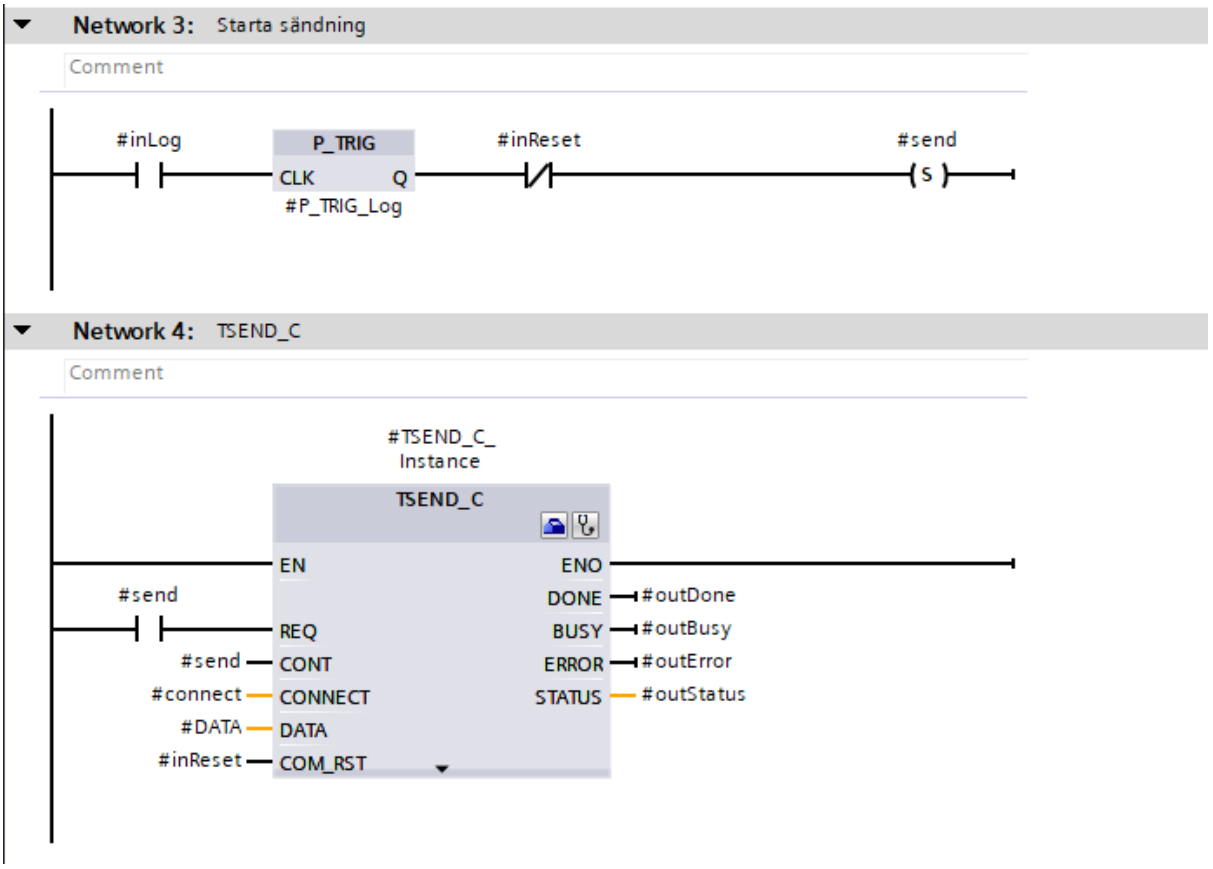
Network 2.

Network 2: Data att skicka

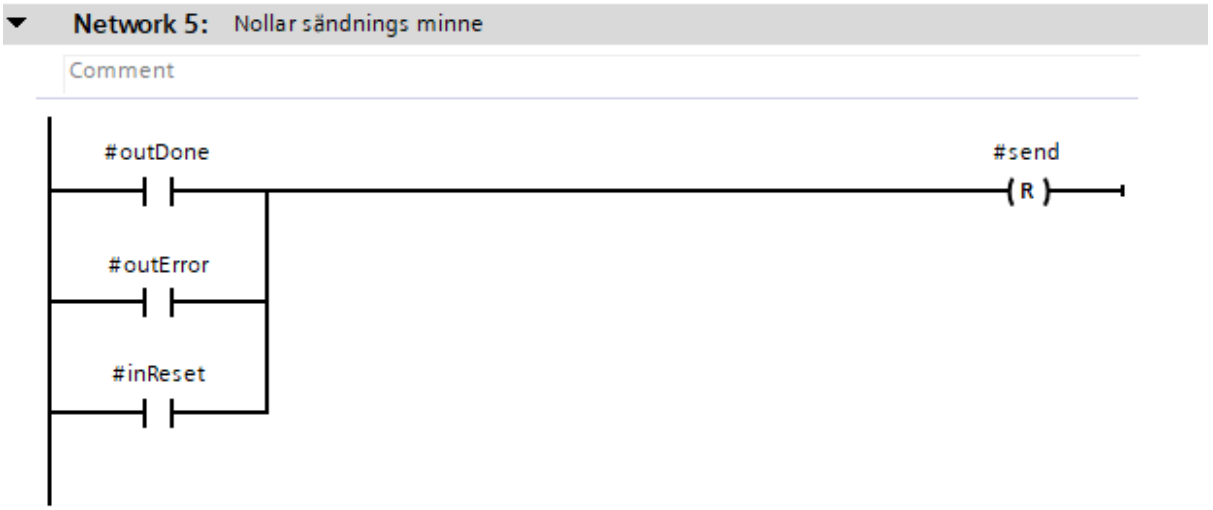
Comment

```
1
2  // Skapar string
3
4 #sData := CONCAT_STRING(IN1:=  '{"cmnd":"log","product":', IN2:=
5                         CONCAT_STRING(IN1 := "fcLInt2String"(#inProduct),
6                                       IN2:= ',"Product_name":"', IN3:= "fcTrimString"(#inProductname),
7                                       IN4:= '","Company_name":"',  IN5:= "fcTrimString"(#inCompany),
8                                       IN6:=  '"}'));
9
10
11
12
13  // Nollar char array
14 FOR #a := 0 TO 100 DO
15
16     #DATA[#a] := ' ';
17
18 END_FOR;
19
20
21  // String till char array
22 Strg_TO_Chars(Strg:=#sData,
23                         pChars:=0,
24                         Cnt=>#cnt,
25                         Chars:=#DATA);
26
```

Network 3 och network 4.



Network 5.


Network 5: Nollar sändnings minne
Comment
#outDone
#send
R
#outError
#inReset

C. Grafana SQL-satser

takttiden tidsserie.

```
SELECT
$__timeGroupAlias(Time_done, 1s),
takt_time
AS "Takttid"
FROM Detail
WHERE
  $__timeFilter(Time_done) AND takt_time < 10000
ORDER BY $__timeGroup(Time_done,1s)
```

medeltakttiden tidsserie.

```
SELECT
$__timeGroupAlias(Time_done, 1s),
avg(takt_time)
OVER(ORDER BY Time_done ROWS BETWEEN 100 PRECEDING AND CURRENT ROW)
AS "MedelTakttid"
FROM Detail
WHERE
  $__timeFilter(Time_done) AND takt_time < 10000
ORDER BY $__timeGroup(Time_done,$__interval)
```

Producerade produkter under tidsserien.

```
SELECT
$__timeGroupAlias(Time_done, 1s),
CONCAT_WS("       ", CONVERT (Product.Article_number, char(100)),
Product.Product_name) as Produkt
FROM Detail
inner JOIN Product on Product.id=Detail.article_id
WHERE
  $__timeFilter(Time_done) AND Product.Article_number > 10
ORDER BY $__timeGroup(Time_done,$__interval)
```

medeltakttid.

```
SELECT
$__timeGroupAlias(Time_done, $__interval),
avg(takt_time) as "Medeltakttiden"
FROM Detail
WHERE
$__timeFilter(Time_done)
ORDER BY Time_done
```

Senaste takttiden.

```
SELECT
  $__timeGroupAlias(Time_done, $__interval),
  takt_time as "Senaste takttiden"
FROM Detail
WHERE
  $__timeFilter(Time_done)
ORDER BY Time_done
```

Antal producerade inom valda intervall.

```
SELECT
$__timeGroupAlias(Time_done, $__interval),
  count(*) as "Antal producerade"
FROM Detail
WHERE
  $__timeFilter(Time_done)
```

Artikelnummer på senaste producerade detaljen.

```
SELECT
  Time_done AS "time",
  Product.Article_number as "Artikelnummer"
FROM Detail
inner JOIN Product on Product.id=Detail.Article_id
WHERE
  $__timeFilter(Time_done)
ORDER BY Time_done
```

Företag som tillverkade senaste detaljen.

```
SELECT
  Time_done AS "time",
  Company.Company_name as "Företag"
FROM Detail
inner JOIN Company on Company.id=Detail.Company_id
WHERE
  $__timeFilter(Time_done)
ORDER BY Time_done
```

Produktnamn på senaste producerade detalj.

```
SELECT
  Time_done AS "time",
 Product.Product_name as "Produkt"
```

```
FROM Detail
inner JOIN Product on Product.id=Detail.article_id
WHERE
  $__timeFilter(Time_done)
ORDER BY Time_done
```

Medeltakttiden per timme.

```
SELECT
$__timeGroupAlias(time_done,1h,0),
avg(takt_time)
AS "MedelTakttid per timme"
FROM Detail
WHERE $__timeFilter(time_done)
GROUP BY 1
ORDER BY 1
```

Antal producerade per timme.

```
SELECT
$__timeGroupAlias(time_done,1h,0),
  count(*) as "Antal producerade per timme"
FROM Detail

WHERE $__timeFilter(time_done)
GROUP BY 1
ORDER BY 1
```