Høgskulen på Vestlandet

Bacheloroppgave Elektro (ING3055)

ING3055-BAC-2021-VÅR-FLOWassign

Predefinert informasjon

Startdato:	28-01-2021 09:00	**Termin:**	2021 VÅR
Sluttdato:	21-05-2021 12:00	**Vurderingsform:**	Norsk 6-trinns skala (A-F)
Eksamensform:	Bacheloroppgave		
SIS-kode:	203 ING3055 1 BAC 2021 VÅR		
Intern sensor:	(Anonymisert)		

Deltaker

Navn:	Joar Kyo Berg
Kandidatnr.:	204
HVL-id:	577027@hvl.no

Informasjon fra deltaker

Tittel *:	Digitalisering i produksjonsbedrift
Antall ord *:	16586
Engelsk tittel *:	Digitalization in production company

Sett hake dersom besvarelsen kan brukes som eksempel i undervisning?:	Ja	**Egenerklæring *:**	Ja
		Inneholder besvarelsen konfidensielt materiale?:	Nei
Jeg bekrefter at jeg har registrert oppgavetittelen på norsk og engelsk i StudentWeb og vet at denne vil stå på vitnemålet mitt *:	Ja		

Gruppe

Gruppenavn:	EL01
Gruppenummer:	3
Andre medlemmer i gruppen:	Sindre Sævareid, Olav Ludvig Skjelstad

Jeg godkjenner avtalen om publisering av bacheloroppgaven min *

Ja

Er bacheloroppgaven skrevet som del av et større forskningsprosjekt ved HVL? *

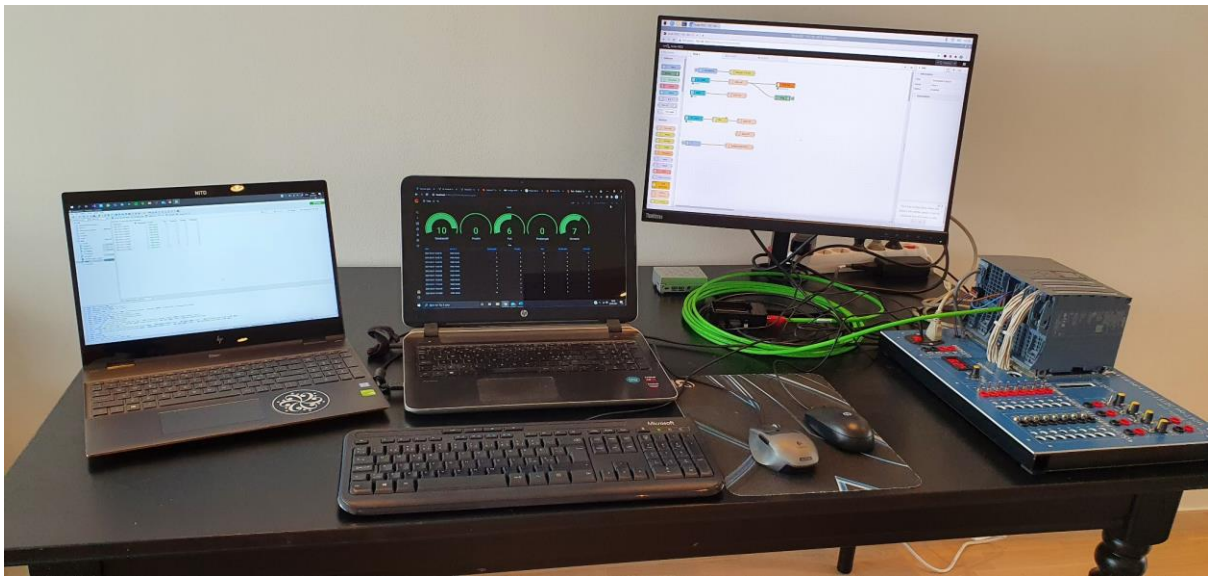
Nei

Er bacheloroppgaven skrevet ved bedrift/virksomhet i næringsliv eller offentlig sektor? *

Ja, Westcon Power & Automation

Høgskulen på Vestlandet

Bacheloroppgåve



Digitalisering i produksjonsbedrift

Elektro Y-veg	
Emnekode:	ING3055
Joar Kyo Berg	204
Sindre Sævareid	202
Olav Ludvig Skjelstad	207

BACHELORPROSJEKT

	Joar Kyo Berg
Studenten(e)s namn:	Sindre Sævareid
	Olav Ludvig Skjelstad

Linje & studieretning Bachelor i ingeniørfag, Elektro Y-veg

Oppgåvas tittel: *Digitalisering i produksjonsbedrift*

Oppgåvetekst:

Denne oppgåva vart gitt av Westcon Power & Automation på vegne av Norcable AS. Prosjektet kjem frå bedriftskunden Norcable, som er ei bedrift som driv med produksjon av aluminiumskablar. Norcable AS ynskjer å styrka digitaliseringa av produksjonen for å oppnå reduserte kostnadar, betre kvalitet og forbetra effektivitet.

Oppgåva går ut på å gjera ei låg-kostnad datainnsamling frå maskinane sine eksisterande PLS-ar (Programmerbar Logisk Styring). Dette gjeld til dømes hastigheiter, tidsbruk og produkt-ID for lagring i PDB (Produksjonsdatabase) og ERP (Enterprise Resource Planning).
Produksjonsdata skal kunne analyserast og visualiserast. I denne oppgåva blir det tatt i bruk ein SBC (Single-board computer), som eit mellomledd for å lesa PLS-variablar og lagra dei i PDB/ERP.
Open-source programvare som skal brukast er primært Node-red, InfluxDB og Grafana. Formålet med å bruka open-source programvare er å redusera kostnad for kunden, ettersom closed-source programvare er forholdsvis kostbar.

Endeleg oppgåve gitt: *(seinast 26.02-'21)*

Innleveringsfrist: Fredag 21.mai 2021 kl. 12.00

Intern vegleiar : Olav Skjølingstad

Ekstern vegleiar: Kjell Mannes

E-postadresse : kjell.mannes@westcon.no

Godkjent av studieansvarleg: 15/04-21

Dato: H. Spångberg

Forord

Denne bacheloroppgåva blei mogleg å gjera gjennom Westcon Power & Automation og Norcable AS. Oppgåva blei gjennomført i vårsemesteret 2021, som slutten av eit treårig bachelorutdanning innan elektro Y-veg industriell automatisering. Med digitalisering som oppgåva vart kunnskapane frå faga 'ING3064 Programmering og mikrokontrollarar' 'ING2051 Styringsteknikk' nyttig. For å få full forståing av rapporten så kan det vera lurt å ha forkunnskapar innan programmering og data.

Det har vore veldig mykje nytt å læra og mange utfordringar som dukka opp. Men kunnskapen som er fått her er gull verdt å ha, no som digitalisering og industri 4.0 blir stadig viktigare for industrien.

I perioden vi har jobba med oppgåva har vi fått aktiv hjelp og støtte frå forskjellige aktørar. Og med det vil vi i bachelor gruppa hjarteleg takke:

Olav Skjølingstad – Intern vegleiar
Kjell Mannes – Ekstern vegleiar
Jan Magne Dybvik – Tilsett i WPA
Bengt Haugland – Dagleg leiar i Norcable AS

Og ynskjer også takk til dei arbeidarane i frå Norcable AS og Informasjonskontroll AS som har vert involverte i arbeidet vårt. Til sist ynskjer vi å takke Westcon Power & Automation som gav oss oppgåva og den store engasjementet frå Norcable AS.

Haugesund
Stad

20/05/21
Dato

Signatur:

Joar Kyo Berg | Sindre Sævareid | Olav Ludvig Skjelstad

Samandrag

For at bedrifter skal overleva den harde internasjonale konkuransen, må industrien klare å legga seg i forkant når det gjeld teknologiskutvikling. Det som stadig blir viktigare er å automatisera flyt av datainformasjon, som vil vera ein del av industri 4.0.

I denne rapporten blir det tatt opp problemstillinga om datainnsamling i bedrifta Norcable AS. Denne bedrifta driver med produksjon av aluminiumsleiarar, kor data blir samla undervegs av produksjon. Problemet bedrifta stod ovanfor var at dei ikkje hadde eit system for å lagre data frå produksjonen. Oppgåva som vart gitt var å klare å lage eit datainnsamlingssystem, som kan automatisert henta, lagra og visualisera desse data frå produksjonen. Sida dette skal vera eit lågkostnad system så skal open kjeldekoda programvare nyttast.

Dei open kjeldekoda programvara som blir nytta er Node-RED, MariaDB, InfluxDB og Grafana. Node-RED er bindeleddet og administreringa av datatrafikk. MariaDB og InfluxDB er databaseprogram for lagring av data, og Grafana for visualisering av data.

I rapporten blir det sett gjennom programvara som blir brukt, val av framgangsmetode, teknisk utføring og testing av løysning. Formåla er å kunne forbetra leveringspresisjonen til Norcable med dei data som blir produserte, og visa at open kjeldekoda programvarer kan nyttast for digitalisering av bedrifta.

Denne oppgåva har resultert med eit fungerande datainnsamlingssystem som er implementert i bedrifta og er i bruk. Det er også lagt til tilleggsfunksjon for å forbetra ytinga av produksjonen. Systemet vil ha moglegheiter til vidareutvikling og forbetring i framtida.

Summary

For companies to survive the harsh international competition, the industries must be able to keep ahead when it comes to technological development. One example of development is the automated flow of data information, which is a part of Industry 4.0.

This report will address the issue of data collection in the company Norcable AS. This company operates the production of aluminium conductors, where data is collected during production. The problem the company faced was they did not have a proper system to properly store data from the production. So, the task given to us was to make this data collection system, which can automatically retrieve, store, and visualize the production data. The system also had to be low-cost made, so the system is built by using open-source software's.

The open-source software's used in this task are Node-RED, MariaDB, InfluxDB and Grafana. Node-RED is the link and the administrating tool of data traffic. MariaDB and InfluxDB are the database programs for storing data, and Grafana is used for visualizing the collected data.

The report will review the software's used, choice of solution, system setup and execution, and testing of selected solution. The purpose of the system is to be able to improve the delivery precision of Norcable with the data that is produced and show that open-source software's can be used for digitalization of the company.

This task has resulted in a functioning data collection system that is implemented in the company and is in use. Additional tools have also been added to improve the performance of the production. The system will have opportunities for further development and improvement in the future.

Innhald

Figurliste

Tabelliste

Definisjonar og forkortingar

Namn	Forkorting	Definisjon
API Endpoint		Endepunkt, angir kor API kan få tilgang til ressursar. Inngangspunkt for kommunikasjon. Dette blir oppgitt som URL.
Application Programming Interface	API	Programmeringsgrensesnitt for utveksling av data mellom forskjellige applikasjonar.
Closed-Source/Proprietary software		Betalt/Lisensiert programvare som er betalt for. Ikkje gitt tilgang til endra eller modifisera programvare.
Datablokk	DB	Eit minneområde i PLS-en kor ulike datatypar kan bli lagra i.
Datatype: Boolsk variabel	BOOL	Datavariabel som har berre to verdiar, sann eller usann.
Datatype: Character	CHAR	Datavariabel som kan lagre ein enkel bokstav.
Datatype: Double Integer	DINT	Datavariabel av heiltal. Datastorleik varierer etter programvare og programspråk. Dobbelt så stor som INT.
Datatype: Floating-point number	REAL	Datavariabel som kan ha heiltal og desimaltal. Datastoleik varierer etter programvare og programspråk
Datatype: Integer	INT	Datavariabel av heiltal. Datastorleik varierer etter programvare og programspråk.
Datatype: String	STR	Datavariabel som lagre sekvensar av bokstavar eller teikn.
Datatype: Timestamp		Tidsmessig datatype som innehalde data om dato og tid.
Effektmål		Mål for langsiktig verknad for ein verksemd.
Enterprise Resource Planning	ERP	Programvare som dekkjer behova til bedrifter innanfor finans, prosjektering, rapportering, analyse, lønning og HR.
Extensible Markup Language - Remote Procedure Call	XML-RPC	Ein metode som brukar XML språket for å lage funksjonar eller kall over nettverket til ein anna datamaskin.
Graphical user interface	GUI	Grafisk brukargrensesnitt. Visuelt samhandling med maskinvarer.
Industriell Ethernet – switch		Eit robust nettverkskomponent med fleire Ethernet portar. Sambinding

		mellom fleire komponentar i eit nettverk.
Internett protokoll– adresse	IP-adresse	Ein identifisering av eining tilkopla eit datanettverk.
JavaScript Object Notation	JSON	Tekstbasert standard for lagring og utveksling av data. JavaScript objekta kan bli konvertert til JSON og tilbake.
Local Area Network	LAN	Lokal datanettverk. Geografisk avgrensa nettverk.
Manufacturing Execution System	MES	Datasystem brukt i produksjonen for å spora og dokumentera produktet frå start til slutt, frå råvare til det endelege produktet.
Minimum viable product	MVP	Minste brukbare produkt. Det minste grensa for kunden for at produktet gir verdi.
Open Platform Communications United Architecture	OPC UA	Kommunikasjonsmetode som gjer moglegheit for kryptering av datakjelde.
Open-Source		Open kjeldekode er programvarer som er gratis å bruka og er open for alle å endra og manipulera på etter kva ynskje ein har.
Operativsystem	OS	Grunnprogramvara i ein datamaskin.
Produksjonsdatabase	PDB	Database som innehalde data brukt for produksjon.
PROFINET		Kommunikasjonsprotokoll for utveksling av data mellom maskinvarer.
Programmerbar logisk styring	PLS	Programmerbar datamaskin som i hovudsak blir brukt i industri.
Single-board computer	SBC	Ein komplett datamaskin på eit enkel kretskort.
Structured Query Language	SQL	Programmeringsspråk for databasar. Brukt til og i MariaDB, InfluxDB og Grafana.
Totally Integrated Automation Portal	TIA	Siemens sitt visuelle grensesnitt for digitale automatiserte system.
Uniform Resource Locator	URL	Internett adresse, www.hvl.no for eksempel.
Unique Identifier	UID	Unik brukar identifikator i eit system.
Westcon Power & Automation	WPA	Bedrift.

1 Innleiing

I dette kapittelet blir det lagt frem kva problemstillinga er, mål og omfanget til oppgåva.

1.1 Bakgrunn

Via Westcon Power & Automation(WPA) fekk me presentert ei oppgåve frå bedriftskunde Norcable AS. Problemstillinga som vart lagt frem var at leveringspresisjonen til Norcable ikkje er tilstrekkeleg. Norcable har ikkje eit godt nok system for innsamling av data, som kan bli brukt til å finna meir nøyaktig anslag av produksjonstida. Når bedrifta lager bestilling så ynskjer dei eit meir nøyaktig tidsanslag på produksjon av produkta sine. Norcable AS driver produksjon av aluminiumsleiarar og drifta skjer mykje automatisk via maskiner. For å kunne oppretthalde gode kundeforhold og optimalisera drifta si, vil dei digitalisera produksjonen meir.

1.2 Formål

Formålet med oppgåva er å kunne lage eit lågkostnad datainnsamlingssystem, der ynskja data frå PLS blir plukka ut og blir sendt til produksjonsdatabase(PDB). Frå PDB skal forretningsverktøyet(ERP/MES) til bedrifta kunne henta ut data, og desse skal kunne bli grafisk framstilt. Systemet skal ha moglegheit for vidare utbygging.

Etter samtaler med Norcable så vart det bestemt kva det minste brukbare produktet (MVP) skal vera. Dette er eit minimumsmål systemet skal oppnå. Oppnår oppgåva å levere tidsdata frå PLS til bedriftas ERP/MES system så er MVP oppnådd.

1.3 Effektmål

Effektmål er den langsiktige verknaden som er ynskja for bedrifta.

- Systemet skal forbetra bedriftas leveringpresisjon.
- Skal lett kunne brukast i framtida for større digitalisering av bedrifta.
- Visualisera innsamla data i PDB, slik at historisk data kan bli samanlikna med dagens data.
- Betra kvalitet, redusera kostnad, betre effektivitet, minimalisera materialsvinn.

1.4 Avgrensingar

I samordning med WPA og bedriftskunde skal denne oppgåva laga eit lågkostnad datainnsamlingssystem. Systemet skal berre bruka open kjeldekoda programvarer samt ha fokuset for leveringspresisjonen til bedrifta.

1.5 Disposisjon

Rapporten begynner det med skildringar av de ulike maskinvarene og programvarene som blir brukt til dette prosjektet.

Deretter kjem eit kapittel med systembeskriving. Kapittelet her skal forklare kva oppgåva går ut på, og korleis datainnsamlingssystemet skal fungera. Etter det vil eit kapittel kome med grunnar for val av programvare og løysningsmetode.

Det er meininga med desse kapitla skal gje lesar innsikt i kva denne oppgåva går ut på. For i neste ledd kjem den tekniske delen av rapporten. Her vil oppsett, flytskjema og bruk av programvarer kome fram. Det er gjort ulike testar på eit testrigg lagd av gruppa og på bedrifta oppgåva gjeld. Problemstillingar og resultat vil bli presentert i kapittelet om testing.

Som avslutning blir det tatt opp moglege svakheiter i systemet med evt. løysningar, og ein konklusjon av oppgåva.

Denne rapporten vil ha med vedlegg av programkoda brukt i dei ulike programvara, JSON filane til systemet og ein manual tillaga for Norcable (nedlasting av programvarer).

2 Skildringar

I dette kapittelet går me gjennom generell skildring av utstyr og uttrykk som er brukt i dette prosjektet.

2.1 Industri 4.0, den fjerde industrielle revolusjonen

Den industrielle revolusjonen inngår i oppgåva sida det denne går ut på er nettopp Industri 4.0.

Denne industrielle revolusjonen skildrar samanslåinga av industriell produksjon og informasjon- og kommunikasjons teknologi. [1]

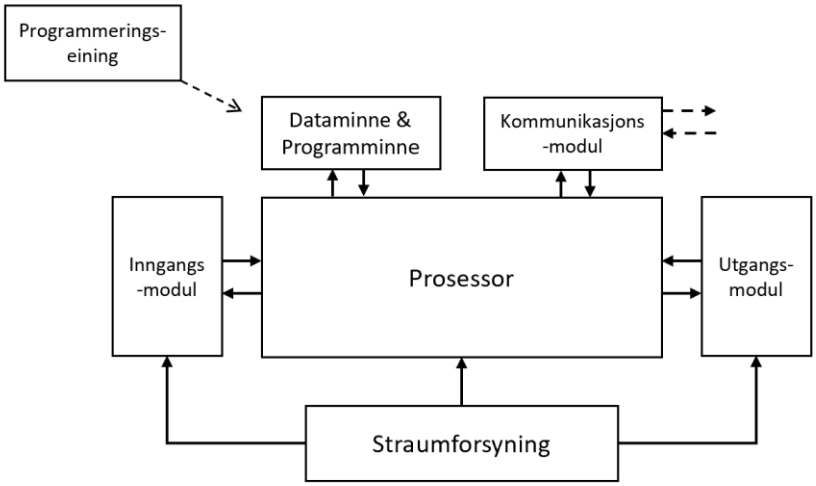
Den første industrielle revolusjonen er den mest kjende, ettersom det er pratet om i historietimane frå ungdomskulen. Denne revolusjonen skjedde på 1700-talet i England, der overgangen gjekk frå handkraft til maskinkraft, drevet av kull og damp. Frå seint på 1800-talet starta den andre industrielle revolusjonen, då elektrisitet og olje vart introdusert til maskin og industriellbruk. Den tredje revolusjon starte rundt 1970-talet og omhandla stadig aukande bruk av automatisering av maskinene og digitalisering i industrien. Noko av dei viktigaste gjennombrota var at PLS-en og roboten vart introdusert for industrien. Det var i denne perioden at informasjons- og kommunikasjonsteknologi (IKT) stadig blei ein viktig del av industrien. [2]

Kort forklart så er industri 4.0 ein oppgradering av digitaliseringa frå den tredje industrielle revolusjonen. Her går internett og informasjonsflyt hand i hand med produksjon og produkta. Nett som eit smarthus kan alt styres via internett i tillegg til få å data og informasjon frå alt som er tilkopla nettet. Og som prosjektet går ut på skal data og informasjonen samla av PLS-en bli tatt hand om og analyserast over nettet.
Derfor er omgrepet industri 4.0 viktig for denne oppgåva.

2.2 Programmerbar logisk styring (PLS)

PLS er ein robust industriell type «datamaskin» som fungerer som ein styring og kontroll for industriell bruk. Denne datamaskinen består av ein prosessoreining som inneheld mikroprosessoren, dette er hjernen som utførar berekningar og databehandling. Prosessoren hentar instruksjonar/program frå dataminnet. Frå ein programmeringseining kan instruksjonane og programma endrast på i minnet til PLS.

PLS utførar ein ynskja handling med hjelp av dei gitte instruksane og data. Med fleire inn- og utgonger så kan PLS ta i mot, og senda ut data- og instruksjonar som kan analyserast. PLS kan behandla diskrete, digitale- og analoge signaler. Frå ein kommunikasjonsmodul så kan PLS henta eller senda data frå andre PLS.



Figur 1 Blokkskjema av PLS

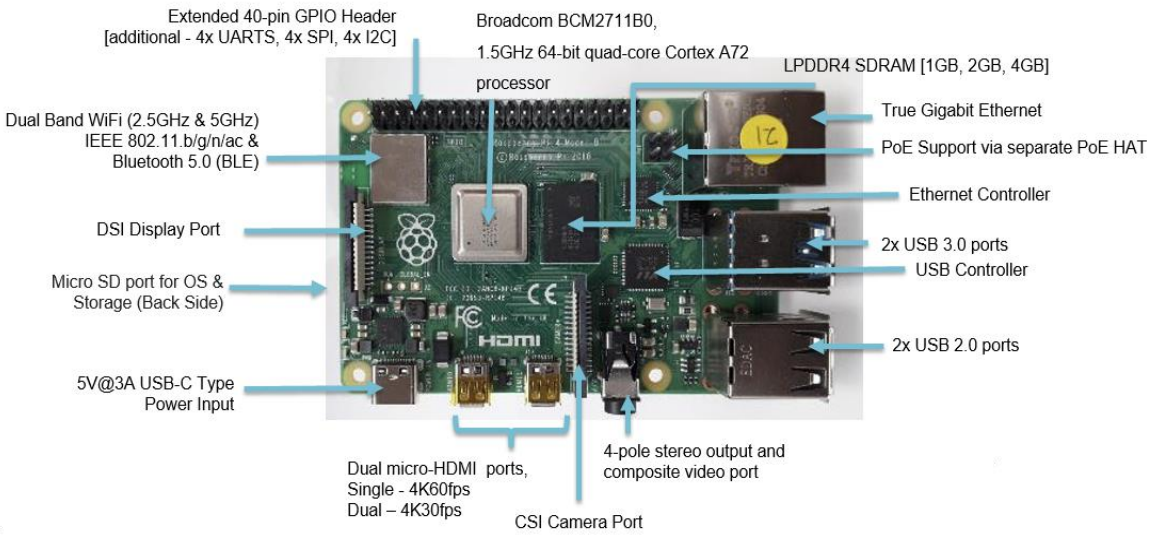
Det er to minnetypar her, dataminne som er eit flyktig minne og programminne som er ikkje-flyktig minne. Med eit flyktig minne så vil data gå tapt ved straumbrot. Mens ikkje-flyktig minne er permanent, som ikkje mister data ved straumbrot. Dataminnet vil vera minnet som vil bli brukt i sanntid, mens programminnet er det som skal lagra data som skal hugsast sjølv ved straumbrot. [3]

2.3 Raspberry Pi 4

I oppgåva er det teken i bruk to SBC(Single-board computer) kalla Raspberry Pi 4, i Norcable som skal vera eit mellomledd frå PLS til PDB. Denne vesle og rimelege datamaskinen som er på størrelsen av eit kredittkort, fungerer mykje likt som ein vanleg datamaskin. Men der er avgrensingar som kjem med at SBC har mindre prosesseringsevne enn ein vanleg PC. Den er hovudsakleg lagd med formål om programmeringslæring. Men ettersom dei er robuste for sin storleik og kostar ikkje mykje, så er det ikkje ukjend at det er brukt i industri.

Med litt kjennskap til programmering kan man bruke denne til eit mangfald av prosjekter [4]. For eksempel:

- Smarthus
- Robot
- Spillkonsoll
- Vanleg datamaskin
- Kamera
- Server
- Sikkerheit



Figur 2 Raspberry Pi 4

2.4 Produksjonsdatabase

Ein produksjonsdatabase er ein database som skal lagre ønska informasjonen frå produksjonen i bedrifta. Produksjonsdatabase fungere på same måte som ein vanlig database, men er kunn meint til å lagre data frå bedriftas produksjon. Ein database er ein samling av data på eit elektronisk medium og blir brukt til mange ulike formål. Datasamlinga blir organisert og strukturert etter ein bestemt strategi eller modell, og blir kontrollert av databasesystemet. Dette databasesystemet vedlikehalde data og administrere kven som har tilgang til databasen. Det finnes fleire ulike måtar å strukturere databasen på etter kva databasemodell og databasesystem som er valen. I dette prosjektet vil det kunn bli lagt vekt på relasjonsdatabase og tidseriedatabase. Det er desse databasemodellane som blir brukt i dette prosjektet.

2.4.1 Relasjonsdatabase

Relasjonsdatabasemodell er den databasemodell som er den mest dominerande i databasemarknaden. Desse databasane er bygget opp av tabellar og relasjonar mellom tabellane. Tabellane kan utformast etter behov og ønske. Det kan til dømes være brukar informasjon til ein nettbutikk, der brukarnamn, adresse, passord og kundenummer blir lagt til som kolonnar. Denne databasemodellen egner seg til å lagre verdiar og data i eit bestemt system og egner seg lite å lagre tekst og filer. [5]

Alle operasjonane på ein relasjonsdatabase blir definert i ein standard som blir kalla SQL-standaren. SQL står for «Structured Query Language» – dette blir på norsk strukturert spørjespråk. Det er dette språket som blir brukt for å lage tabellar, legge inn ny data og hente ut data frå databasen.

2.4.2 Tidsseriedatabase

Tidseriedatabase er ein database som er spesialisert på å lagre tidsseriedata. Tidsseriedata er data som er basert på målingar eller hendingar som blir samla opp over tid. Dette kan vera mengde klikk på ein nettside, mengde sal av ein vare/teneste par minutt eller sensor data som målar temperatur, trykk, vind osv. [6]

Tidsseriedata genererer store mengder data over lengre tidsrom. For at data skal være nyttig så må data sjåast i samanheng over eit gitt tidsrom. Dette tidsrommet kan være ein dag, veke, månad eller fleire år. Ved å bruke ein vanlig database ville dette krevje mykje berekning og

administrasjon av databasen. Mens ein tidsseriedatabase er effektivisert for ein slik bruk, som gjere at slike oppgåver blir gjennomført raskt og effektivt.

2.5 Programvarer

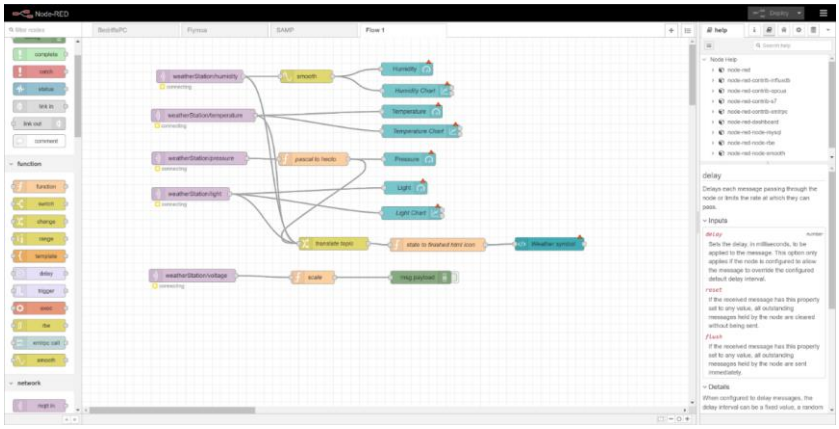
I prosjektet blir det teken i bruk open kjeldekoda programvarer, med unntak av programvarer som bedrifta har allereie betalt og lisensiert for. Open kjeldekode er programvarer tilgjengeleg for alle utan kostnad og kan brukast/programmerast etter kva ein ynskjer. Det motsette er kalla proprietær kjeldekode som må betalast for og må ha lisens eller abonnering. Det er heller ingen tilgang til å sjå/endra på kodane til programvara. Dei største fordelane med Open kjeldekoda programvare er at programvara har ofte fleire brukarar, og dermed mange fleire som kan avdekke feil og utbetra dei. Dei open kjeldekoda programvara tilbyr ekstra tenester for betaling/abonnering for tilleggs tenester. Desse tenesta er ikkje så vitale, slik at de enda er open kjeldekoda.

2.5.1 Node-RED

Node-RED er eit open kjeldekoda flytbasert utviklingsverktøy for visuell programmering. Det var opphavleg utvikla av IBM for å kople saman maskinvareeiningar, API og online tenester. Node-RED tilbyr ein nettlesarbasert flyteditor, som gjer det enkelt å kople saman flyt med eit stort sortiment av noder. [7]

Programmet er veldig anvendeleg og kan vera brukarvennleg etter kor avanserte oppsett som skal til. Det kan brukast til både privat i heimen, for bruk til smart-heim, eller industriellbruk med overføring av data.

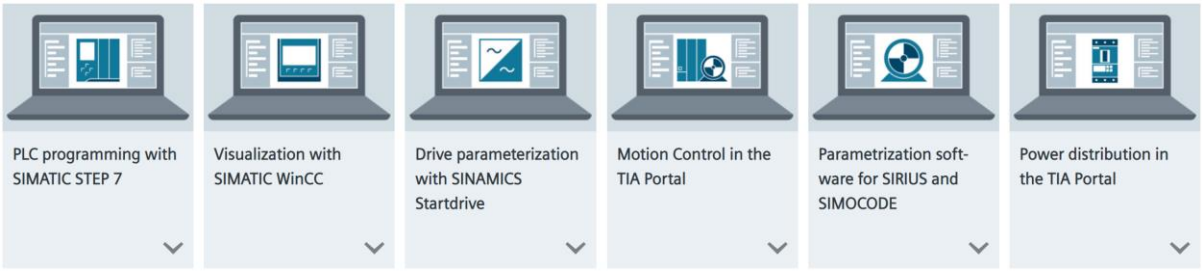
Dette programmet blir brukt i ein PC og to Raspberry Pi 4 i bedrifta. Modell og oversikt av dette blir vist i kapittel 3. SBC-ane blir brukt som bindeledd mellom PLS og PDB, mens i PC er det eit bindeledd mellom PDB og ERP.



Figur 3 Eksempel på eit Node-RED program

2.5.2 Simatic TIA Portal

Simatic TIA (Totally Integrated Automation) er Siemens sin portal som innehalde forskjellige programvare, for å konfigurera og programmera sine PLS. I Figur 4 viser det dei ulike programvarene og kva dei blir brukt til.



Figur 4 Utklipp frå Siemens: https://new.siemens.com/us/en/products/automation/industry-software/automation-software/tia-portal/software.html

Dette er Closed-Source og må lisensierast for å ha full tilgang. Siemens er store innanfor marknaden spesielt for PLS, og er veldig kjend blant industri og annet arbeid innan produksjon. Programmet er anvendeleg og tilbyr forskjellige programmeringsspråk som Ladder-diagram, Funksjonsblokkdiagram, Strukturert tekst osv.

2.5.3 InfluxDB

InfluxDB er eit open kjeldekoda tidsseriedatabase som er brukt for rask lagring og henting av data. Det vil sei programmet skal samla data i bestemde tidsintervall. InfluxDB er eit fint verktøy å bruka for å undersøkja endringar over tid.

2.5.4 MariaDB

MariaDB er eit open kjeldekoda relasjonsdatabase som blei utvikla frå tidlegare utviklarar frå MySQL. Relasjonsdatabase bruker ein tabellstruktur der kvar data får sine eigne identifikasjonar/nøklar, og kan relatere forskjellige tabellar med kvarandre. MariaDB kan ta i mot data og plassere dei i tabell, der innhaldet kan endrast eller behandlast etter kva ynskje ein har.

2.5.5 HeidiSQL

I oppgåva er HeidiSQL brukt som eit administreringsverktøy MySQL/MariaDB. Med dette open kjeldekoda hjelpemiddelet blir det vist eit GUI for å visualisera databaseprogram. Det HeidiSQL gjer er at den kommuniserer med databaseprogramma i SQL kommandoar, i staden for å bruka tekstbaserte-grensesnitt som MariaDB/MySQL. Så med hjelp av dette programmet så blir databaseprogrammane styrt i eit vindauge format i staden for i leietekst. Dette er eit verktøy for databaseprogram og erstattar ikkje programmane.

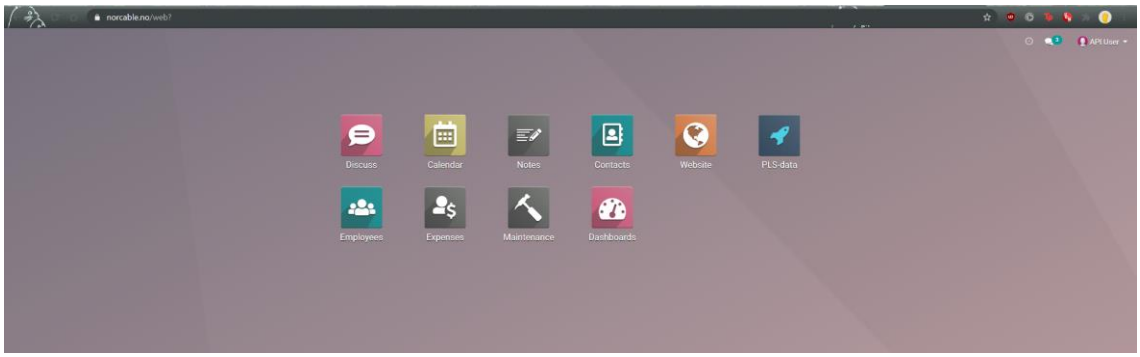
2.5.6 Grafana

Dette er eit open kjeldekoda nettbasert analyse- og visualiseringsprogram. Med dette programmet kan man kople seg til databasar i databaseprogram som InfluxDB og MariaDB for å henta ut data og gi ein visualisering. Med dette kan ein sjå og kartleggja etter korleis oppførselen til produksjonen er. Her kan det framstillast med tabell, søylediagram, tidsserie, div. målarar, logg osv.

2.5.7 Odoo

Odoo er eit forretningsadministrasjons verktøy som blir brukt for å handtera systemet til eit selskap. Det handterer for eksempel lagerbehaldning, HR, økonomi, bestilling, ordreovervaking osv.

Det finst to versjonar av Odoo som forskjellen er mellom open kjeldekoda og proprietær kjeldekoda. Odoo Enterprise er proprietær kjeldekoda og blir betalt for full funksjonalitet. Mens Odoo Community er open kjeldekode versjonen som ikkje har alle funksjonane som Odoo Enterprise har. Eksempelvis har Community versjonen ikkje tilgang til mobil/android/iOS grensesnitt, ulike verktøy for marknadsføring og sal osv. Norcable bruker Odoo Enterprise som ERP/MES system.

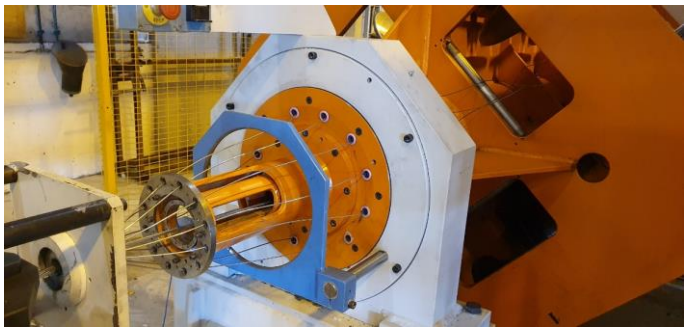


Figur 5 Odoo hovudside

3 Systembeskriving

3.1 Maskinbeskriving

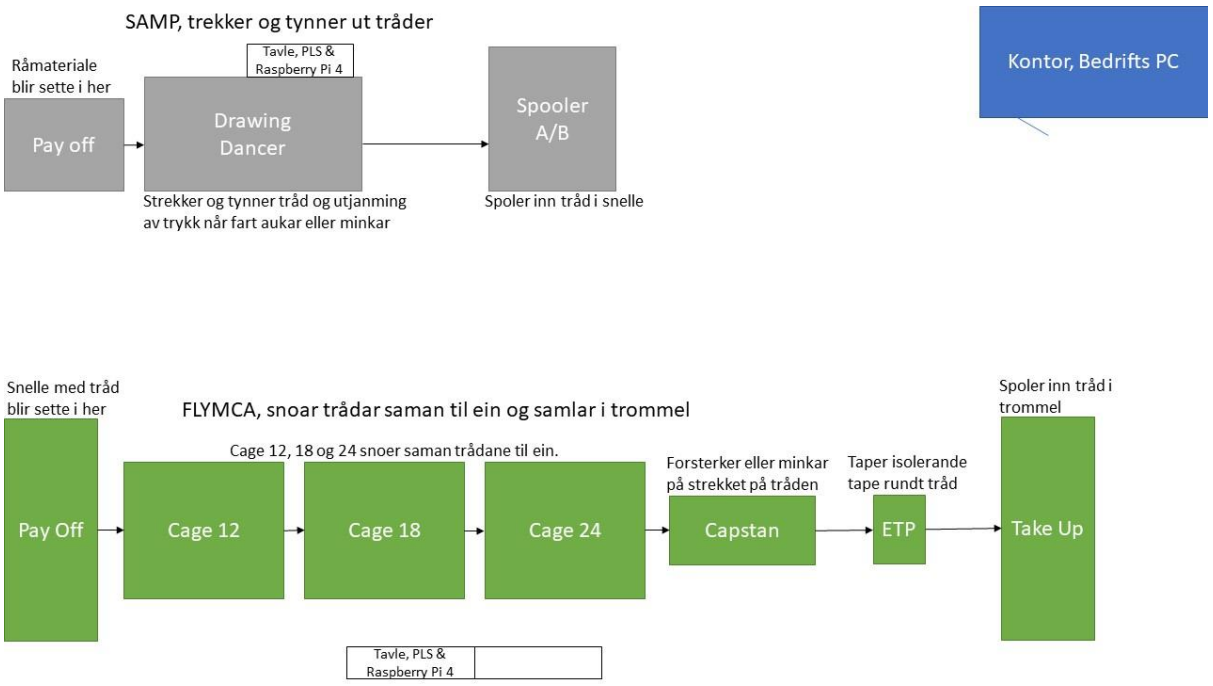
På Figur 7 er det vist frem ein enkel oversikt over produksjonshallen til Norcable.
I hallen er det to produksjonslinjer, som er namngitt SAMP og FLYMCA etter namnet på produsentane av maskinene. Desse produksjonslinjene opererer uavhengig av kvarandre og er utstyrt med kvar sin Siemens S7 PLS.



Figur 6 Norcable Cage maskin

SAMP omarbeider aluminiumstråd til aluminiumskordelar som er råvare til FLYMCA. Dette gjer den ved å endre diameteren og forma på aluminiumssnora som er maskinens råvare. Desse kordelane blir ført på tromlar i ynskja lengde.

FLYMCA er den produksjonslinja som snoer kordelar frå SAMP saman til aluminiumsleiarar som skal i ein kabel eller enkle aluminiumslinjer. Produksjonslinja får det mesta av kordelane frå SAMP, men nokon av kordelane kan vera av eit spesiell eigenskap eller material og blir produsert ein anna plass.



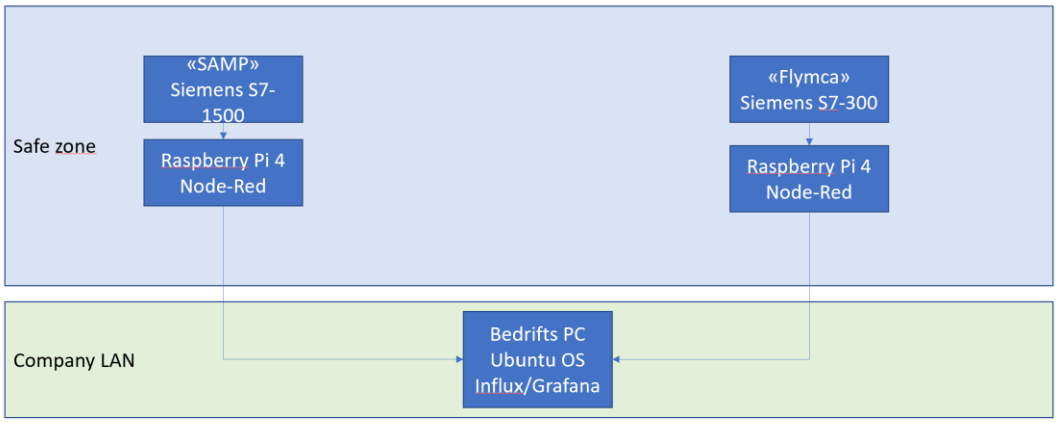
Figur 7 Oversikt over produksjonslinjene i Norcable

3.2 Nettverk og systembeskriving

Frå før av var det satt opp eit nettverk med to Raspberry Pi 4(SBC) og ein bedrifts PC for samling av data. Nettverket og maskinvara er nytta på same måtte i det nye systemet og er ikkje endra av dette prosjektet. Detta nettverket er oppbygd med «Safe-zone» og «Company LAN».

«Safe-zone» er den delen av netverket som er i sikker sone, som vil seie at all kommunikasjon må starte inni dette nettverket for å få tilgang til bedriftsnettverket. SBC som er i den sikresonen, er tilkopla via ein Industriell Ethernet-switch for tilgang til datakommunikasjonsnettverket(PROFINET) til PLS. PROFINET er kommunikasjonsprotokoll som gjer det mogleg for maskinvarer, som SBC og PLS kan kommunisera med kvarandre. «Company LAN» er vanlig bedrift nettverk.

Figur 8 viser korleis systemet var oppbygd frå før av. Dette systemet henta ut utvalt data frå PLS og lagra det på InfluxDB. Data frå InfluxDB vart vist som grafar i Grafana.

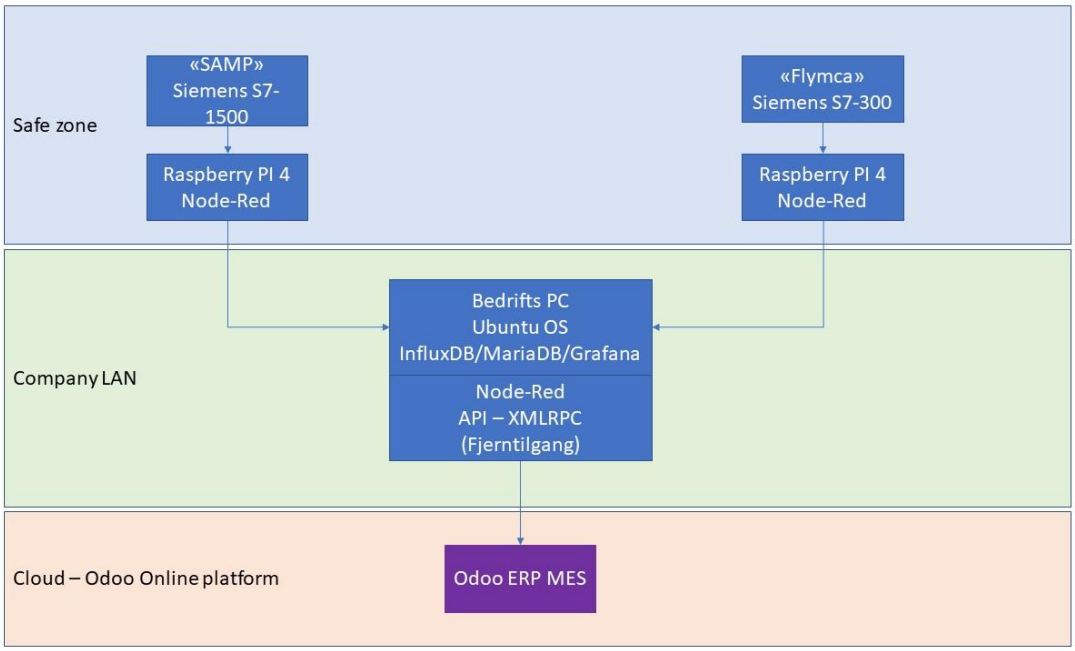


Figur 8 Systemoversikt før bacheloroppgåva starta

I dette prosjektet skal det flyttast data frå dei to PLS til bedrifts PC og være tilgjengeleg for andre PC i bedriftsnettverket. Data skal også vera tilgjengeleg i bedrifta sitt forretningsverktøy, Odoo.

For å få den ynskja funksjonalitet har gruppa bygget opp eit heilt nytt system for innsamling av data. Gruppa har vurdert og valt dei ulike programvarene som skal til for å gjennomføre dei ulike funksjonane til systemet. Det nye systemet er oppbygd med to uavhengige hovudfunksjonar, produksjonsdata og brotanalyse, som er beskrive i delkapittel 3.3 og 3.4. Desse funksjonane har ulike oppgåver, men har den same oppbygning, framgangsmetode og køyre på dei same programvarene.

På Figur 9 viser det systemet bachelorgruppa skal laga til Norcable.



Figur 9 System oversikt bacheloroppgåva

Raspberry Pi 4 (SBC)

På SBC køyre programmet Node-RED. Dette programmet hentar ut, sortere, klargjere og sender data frå PLS til produksjonsdatabase(PDB). Under dette prosjektet er det gjort større endringar i programmet for å kunne senda produksjonsdata og brotanalyse som er dei to ynskja hovudfunksjonane til systemet. Sjå kapittel 5.1.4 for beskriving av korleis dette blir gjort.

Bedrifts PC

PDB er på bedrifts PC og består av to databasestrukturar. Den eine strukturen var frå før av og er ein tidseriedatabase(InfluxDB). Den andre databasestrukturen er lagt til under dette prosjektet og er ein relasjonsdatabase(MariaDB). Sjå kapittel 0 for informasjon om dei ulike databasestrukturane og kapittel 4.2 for kvifor det er to databasestrukturar. Det er MariaDB som blir brukt til å administrera, omarbeide og lagre data for dei to hovudfunksjonane til systemet, sjå kapittel 5.2 korleis dette blir gjort.

Frå PDB blir data overført til Grafana for visualisering, sjå kapittel 5.3. Data frå PDB skal bli overført til Odoo Enterprise som er bedriftas forretningsverktøy. Dette skjer gjennom Node-RED via internett, sjå kapittel 5.4.

3.3 Hovudfunksjon produksjonsdata

Denne hovudfunksjonen er til for å hente ut produksjonstid, og tid for oppsett frå produksjon på kvar enkelt produkt frå produksjonslinjene. Desse tidene skal bli bruk til å kunne betre føresjå, kor lang tid som vil bli brukt for å produsere eit bestemt produkt. Norcable ynskja også å ha den produserte lengda, kva tid produktet var produsert, og trommelnummer for kunne identifisere produksjon i bedriftas forretningsverktøy.

Lagring av data skjer i ein tabell i MariaDB. Her blir dei ynskja tidene, produsert lengde og trommelnummer lagra i samen med produkt ID. Det er lagt til ein kolonne for diameter på linja for å kunne kontrollera at produkt ID stemmer, overeins med produktet som er køyrt gjennom maskinen. Det er operatøren som må legge inn ny produkt ID for kvar type produkt som skal køyre gjennom maskinen. For å gjere det enklare å samanlikne fleire forskjellige lengder av same produkt, er det ein kolonne som reknar ut den gjennomsnittlege farten på denne produksjonen. Denne hovudfunksjonen er både på SAMP og FLYMCA.

3.4 Hovudfunksjon brotanalyse

For å kunne auke produksjonen til FLYMCA må farten til maskinane aukast. Dette kan føre til ulempa, med at sannsynelegheita for brot kan auke. For at operatøren skal ha betre kontroll på kva fart som er den optimale, med omsyn til produktivitet og brothyppigheit, er det laget ulike tabellar og grafar, sjå 5.3.1. Dette er ikkje sette opp i SAMP på grunn av at PLS program var annleis og brot variabel blei ikkje funne.

Systemet er laget for å registrera kva fart FLYMCA køyre på under produksjon og når det skjer eit brot, sjå kapittel 5.1.4 for utførsel. Ideen med systemet er at det skal vera mogleg å henta ut erfaringar frå tidlegare køyringar på eit bestemt produkt. I eit diagram vil erfaringane visa forholda mellom mengde timer køyrt og mengde brot. Då kan man sjå kor mange timer det kan forventast køyrast før eit brot kan skje, sjå delkapittel 5.3.1 for meir info om korleis diagram fungerer. Ved hjelp av denne informasjonen kan farten settast etter det som erfaringsvis gir lav brothyppigheit i forhold til fart. Er brothyppigheita svært låg over tid kan farten aukast for å auke produktiviteten til produksjonslinja. Blir farten for høg vil brotraten gå opp, dette vil vise med at søyla blir lågare, sjå Figur 38.

Denne funksjonen lager også ein feilrapport som beskriver når og kor brot har skjedd på linja. Denne rapporten beskriver også kva del av produksjonslinja som feilen skjedde på.

4 Vurdering og val av programvarer

Det finst mange måtar å løyse problemstillinga ettersom det er finst eit mangfald av programvarer som kan brukast, som kan fungerer likt eller betre. I dette kapittelet blir det lagt frem vurdering og grunnlag til val av programvare.

4.1 Utviklingsverktøy

Siemens TIA portal er programvarene som blir nytta i forbindelse Siemens PLS, og sida det er Siemens PLS brukt i bedrifta så blir dette programmet nytta i oppgåva.

For sin oversiktelegheit og anvende, så er Node-RED vald. Det liknar på ein måte eit Ladder-program, med noder som byggeblokkar, sjå Figur 3.

Kvar node har eigne spesifikke funksjonar som utførar forskjellige utføringar, dette blir meir utbreidd i delkapittel 5.1.2. Node-RED er eit populært verktøy så det finst eit stort forum, kor man kan finna vegleiingar eller spør andre brukarar om hjelp og råd.

Eit anna verktøy som kunne ha blitt vald er Python. Dette er eit programmeringsspråk der ein programmerer med tekst, noko som gir meir friheit enn Node-RED. Men ulempe med Python er at det krevst meir forkunnskapar og er ikkje lika oversikteleg som Node-RED. Av den grunn ynska bedrifta at Node-RED vart nytta.

4.2 Databaseverktøy

Det er mange ulike databaseverktøy og modeller å velje mellom. Valet av databaseverktøy i dette prosjektet er det lagt vekt på kva data som skal bli lagra. Som er i denne oppgåva går ut på å samle inn tidsbruk på forskjellige prosessar i produksjonen av aluminiumkabel. Desse tidene skal også knytast opp mot spesielle produkt og mengde meter som er produserte.

Det vært foreslått frå Westcon og Norcable at det skulle brukast InfluxDB til å utføre denne oppgåva. Frå før av er InfluxDB brukt i Norcable. Men etter undersøking og testing blei det oppdaga at InfluxDB ikkje ville vera det optimale databaseverktøyet, ettersom det er tidsseriebasert(2.4.2). Mens med å bruke relasjonsdatabase(2.4.1) så vil den registrere i database etter kommando, og data kan strukturerast i ein eigendefinert tabell som kan innehalde ynskja datafelt. Ved ein relasjonsdatabase kan også data enkelt hentast ut ved bruk av SQL kommandoar.

Av relasjonsdatabasar var MySQL vurdert å bli brukt først i prosjektet. Det blei funne ut av at MySQL er open kjeldekoda til ein viss grad, ettersom nokre funksjonar var berre tilgjengeleg i den betalte versjonen av MySQL. Med litt undersøking blei MariaDB funnet som var tidlegare ein del av MySQL, men skilte lag etter det blei mindre fokus på open kjeldekoding. [8]

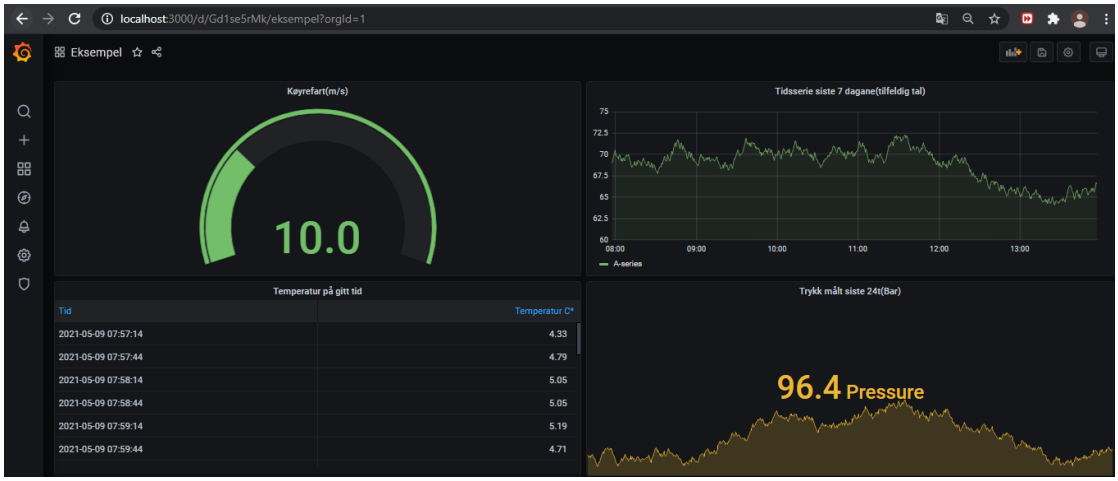
MariaDB vart valt til å være den relasjonsdatabasen i dette prosjektet, ettersom det er eit fullverdig open-kjeldekoda databaseprogram. MariaDB har eit rikt økosystem med mange av tilleggsfunksjonar. Sida MariaDB er basert på MySQL kan mange av funksjonane frå MySQL brukast direkte.

Det var velt å fortsetje å bruke InfluxDB i tillegg til MariaDB. InfluxDB er eit godt verktøy til å lagre data direkte i forhold til tid og kan brukast til å vise maskinens fart og trekkrefter.

4.3 Visualiseringsverktøy

Her også vart det på forhand valt eit visualiseringsverktøy som er Grafana. Dette verktøyet er veldig anvendeleg og kan laga til diagram, grafar og ulike målarar. For overvaking av produksjonen så passer dette programmet veldig bra til tidsseriebasert data.

Dette verktøyet kan brukast i eksterne datamaskinar i same nettverk, eller kan gi tilgang til brukarar i andre nettverk. Det betyr at man kan sitta andre plassar enn bedrifta og sjå visuelt data, både historiske og pågåande produksjon.



Figur 10 Grafana eksempel

4.4 OS-system

Hjå bedrifta Norcable så blir OS Ubuntu hovudsakleg brukt, mens under testing i oppgåva er det brukt Windows. Det er to veldig ulike system og kan i hovudtrekk kan dei skiljast slik:

Windows	**Ubuntu**
Closed-Source OS	Open-Source OS
Windows NT basert	Linux basert
Må ha gyldig lisens for å bruka	Trenger ikkje å ha lisens
Kan ikkje modifisere på kodane til OS	Kan modifisera på kodane til OS
Større ressursbruk	Mindre ressursbruk
Mindre tilpassing moglegheiter	Meir tilpassing moglegheiter
Mindre sikkerheit	Betre sikkerheit
Mindre personvernfokusert	Meir personvernfokusert
Har mange tredjepartsprogramvarer (Meir moglegheit for underhaldning)	Har ikkje lika mange tredjepartsprogram som støtter Linux (Mindre moglegheit for underhaldning)

Tabell 1 Forskjell på Windows og Ubuntu

Så grunnen bedrifta brukar Ubuntu er i all hovudsak at det er mykje meir tryggare å bruka når det gjeld sikkerheit. For i Ubuntu så er ikkje denne OS bunden til lika mange tredjeparts programvarer i forhold til Windows, kor sensitiv informasjon kan vera lagra og bli sende vidare.

Brukarvennlegheita er ein diskusjonssak ettersom Ubuntu er lagd for å vera enklare. Det vart opplevd som vanskeleg å handtera, men den grunnen kjem med at gruppa hadde ingen forkunnskap av bruk av Linux baserte OS.

I denne oppgåva så er testing og forhandsarbeid gjort på Windows. I bedrift er det brukt og implementert med Ubuntu. Bruksmåtane av programvara er like på begge OS, men den store forskjellen ligger på nedlastingsmåte og endring av innstillingar.

Det er eit tredje OS som blir brukt som er Raspberry Pi OS som er brukt i SBC. Raspberry Pi OS er eit gratis operativsystem basert på Debian, optimalisert for Raspberry Pi maskinvarer. [9]

5 Teknisk utføring

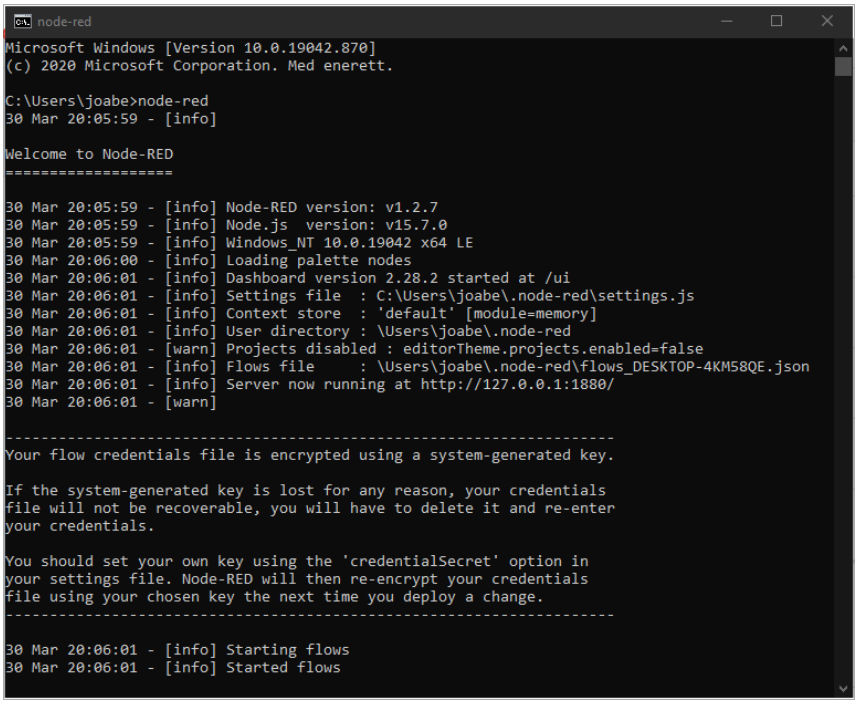
Dette kapitelet beskriver korleis programvarer, innstillingar og programkoder hengjer saman for å få eit velfungerande system. Delkapitla er delt opp i rekkefølgje etter maskinvarer og funksjonalitet.

5.1 Oppsett av SBC

Målet er å henta ut relevante data frå PLS for å optimalisera leveringspresisjon, og skape eit betre oversikt av produksjonen. SBC skal vera bindeleddet mellom PLS og PDB. Her skal data frå PLS klargjerast for PDB.

5.1.1 Oppstart av Node-RED

Node-RED er eit nettbasert verktøy, som vil sei at den opnast og brukast i nettlesar når da er starta. For å lasta ned dette verktøyet så må leietekst i datamaskinen brukast. Når Node-RED er ferdig nedlasta så må det køyrast via leietekst for å brukast, som vist på Figur 11.



Figur 11 Leietekst for start av Node-RED

Etter det så kan det opnast i nettlesar som http://localhost:1880 eller http://127.0.0.1:1880 som er begge lokalvert(lukka nett berre for gjeldande maskinen).

5.1.2 Brukte noder

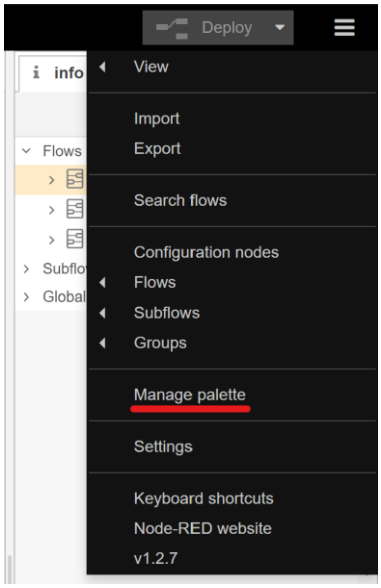
Det er tatt i bruk mange forskjellige noder i dette prosjektet. I dette kapitelet går gjennom dei noda som er brukt i dette prosjektet.

Det finst mange noder som er standard i Node-RED og noda er grupperte i noko som er kalla palettar. Det finst mange palettar som innehalde fleire noder som kan lastast ned etter kva funksjonar som trengst. Desse palettane kan finnast og lastast ned i Node-RED i noko som heiter «Palette Manager».

Av palettar som vart teken i bruk er:

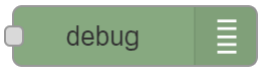
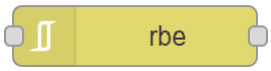
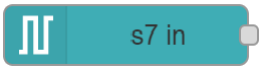
- Node-red-node-mysql, noder for tilkopling til MariaDB/MySQL databasar
- Node-red-contrib-influxdb, noder for tilkopling til InfluxDB databasar
- Node-red-contrib-xmlrpc, noder for å kommunisera med Odoo i denne oppgåva
- Node-red-contrib-s7, noder for Simatic S7 PLS

Desse palettane blir lasta ned i «Palette manager» innpå Node-RED, så Figur 12.



Figur 12 Palette manager

Tabellen under viser og forklarer nodane som blir brukt i denne oppgåva:

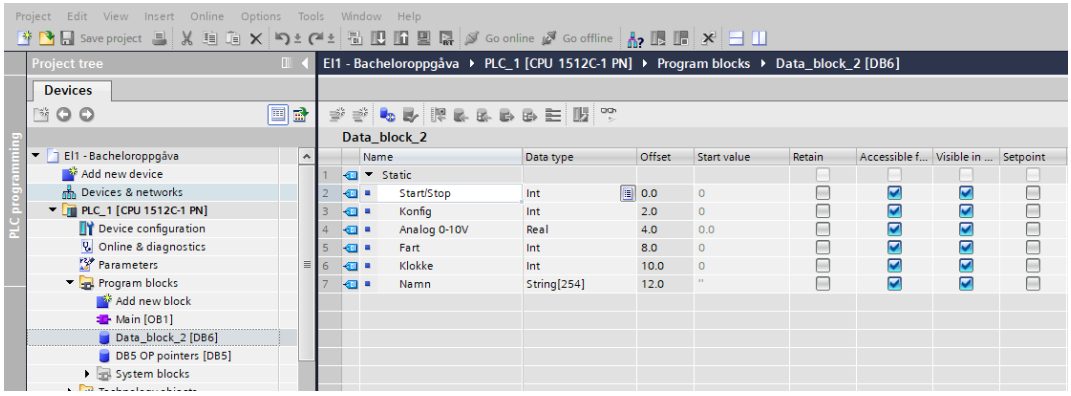
Namn	Bilete	Definisjon
Inject		Sender inn ein melding/puls manuelt eller i eit fast intervall. Innhaldet til meldinga kan vera av ulike typar etter kva ein bestemmer, om det skal innehalde «string», Javascript objekt, den gjeldande tida og vil fungera som ein trigger funksjon.
debug		Denne noden visar heile eller delvis innhaldet til meldinga som blir sendt ut av ein anna node. Dette blir vist i eit vindauge i Node-RED «debug messages».
catch		Tek i mot alle feilmeldingar som blir danna i det gjeldande vindauget i Node-RED. Den kan hjelpa å lokalisera feil og kva feilen kan vera.
function		Dette er ein funksjons node som brukar Javascript programmeringsspråk. Som i tekstbasert programmering kan mykje funksjonar lagast til her. Denne kan ha ein eller fleire utgangar og kan levera ut null eller fleire msg. typar ut.
change		Denne noden kan påverke innhaldet til ein eller fleire data som kome inn Node-RED. Det kan lagra(set) variablar, endre innhald av meldingar, sletta, eller flytta på innhald.
template		Noden her kan generera ein tekst eller verdi ut i frå kva skript som er ført inn i noden. Det den generer kan bli brukt som ein puls med ein kommando.
rbe		Rbe står da for «Report by Exception» og er ein node, som berre slepper forbi data om datainnhaldet har endra seg. Denne kan også innstillast til å sleppa forbi eller blokkera data som går gjennom etter ein spesifikk verdi er nådd.
xmlrpc call		Denne noden utførar eit metodekall til ein XML-RPC server som i dette tilfellet er Odoo. Med metodekall meiner vi at den finner serveren den skal kople seg til, og overføre data.
influx batch		Når denne får melding inn så vil denne senda ein samla mengde data til databasen i InfluxDB.
mysql		Får tilgjenge til MariaDB/MySQL database. Her kan det både sendast inn og tas ut i frå database.
s7 in		Får tilgjenge til Siemens S7 PLS-ar. Denne leser berre i frå uoptimaliserte datablokkar frå PLS. (sjå delkapittel 5.1.3)

Tabell 2 Oversikt av brukte noder

5.1.3 Kommunikasjon frå PLS til Node-RED

Dette kapitel beskriver korleis og kva som skal til for å få kommunikasjon frå PLS til Nore-RED. Dei variablane som er brukt i dette prosjektet er vist i Tabell 3.

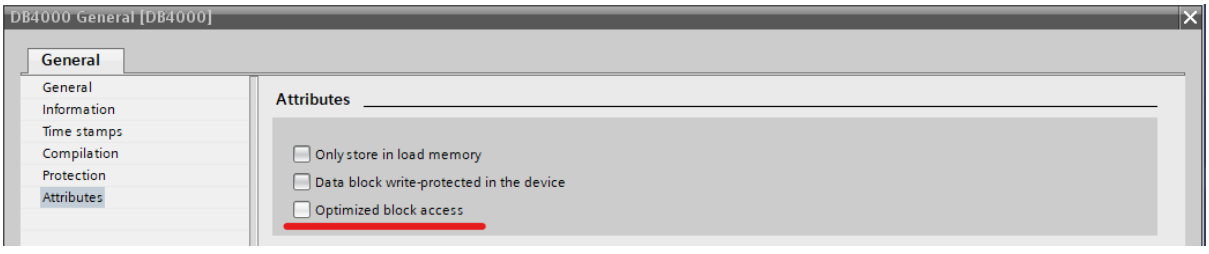
I PLS er det datablokkar(DB) som innehald variablar frå produksjon, i Figur 13 viser det DB brukt for testing. Ein innstilling som er viktig å sjekka i PLS er om DB er uoptimaliserte. Dette er viktig for at det er mogleg for Node-RED å henta innhaldet frå DB. Forskjellen med optimalisert og uoptimalisert DB er korleis data er strukturen. Når data er uoptimaliserte kan data lokalisera med rette adressa.



Figur 13 Datablokk (DB) i PLS program

Optimalisert DB vil arrangera data på den måten at blokkane ikkje vil bruka så mykje minneplass. Nedsida med dette er at datainnhald mister direkte adresse til data. Med uoptimalisert DB så vil det ha ein fast struktur som ikkje sparer så mykje på minneplass, men datainnhaldet vil ha direkte adresse. [10]

Denne innstillinga blir endra med å høgreklikke på den bestemte DB, og vidare på «properties» skal dette vindauget som vist i Figur 14 dukke opp. Sjekk at «Optimized block access» er umerka.



Figur 14 Optimized block access

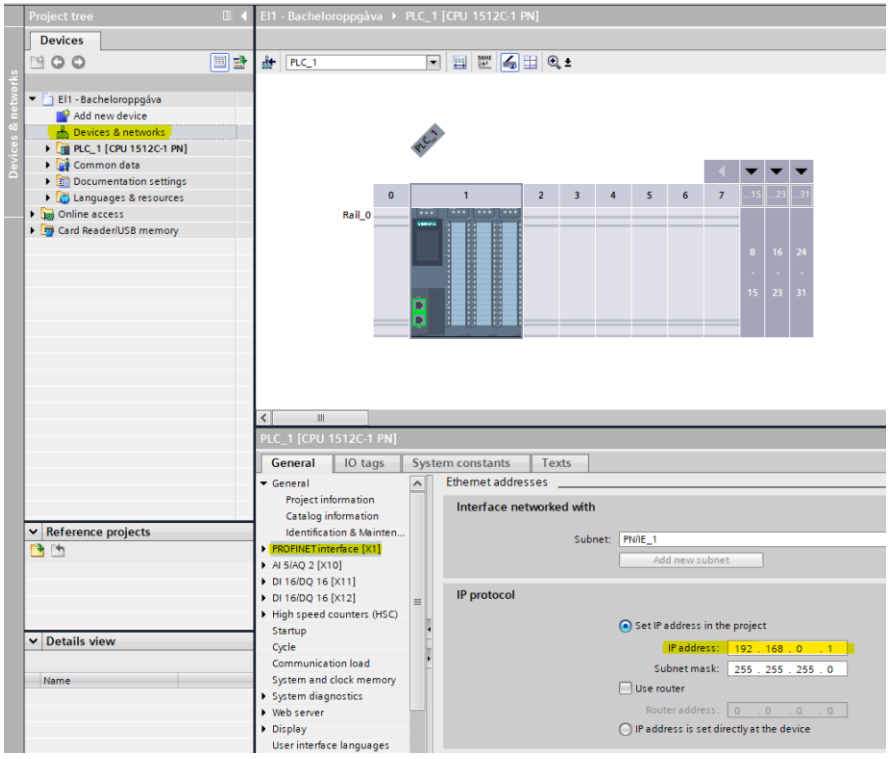
FLYMCA				
DB4000 General				
Linje nr.	Namn	Data Type	Offset	Beskriving
45	Cable length in the bobbin	DINT	28.0	Nåverande total produsert kabel lengde akkumulert i trommel
48	Bobbin identification	DINT	40.0	ID på nåverande trommel på kva kabeltype som blir produsert
51	Reel: Loading cumulative time	DINT	52.0	Tidsteller for kor lengje setup/loading status pågår
52	Reel: Production cumulative time	DINT	56.0	Tidsteller for kor lengje sjølva produksjonen pågår
53	Reel: Alarm cumulative time	DINT	60.0	Tidsteller for alarm
DB4001 Take up				
12	Wire Break	Bool	1.2	Alarm for brot av tråd
74	Pulling force	REAL	82.0	Trekkraft i tråd
DB4002 Pay Off				
12	Wire Break	Bool	1.2	Alarm for brot av tråd
74	Pulling force	REAL	82.0	Trekkraft i tråd
DB4007 Capstan				
34	Line speed	REAL	46.0	Fart på line
36	Cable tension	REAL	54.0	Trekkraft i tråd
DB4008 Cage12 | DB4009 Cage18 | DB4010 Cage24				
33	Wire Break	Bool	3.7	Alarm for brot av tråd
111	Cage speed	REAL	94.0	Cage rotasjons fart
112	Reel angular speed	REAL	98.0	Snoing rotasjons fart
114	Wire tension	REAL	106.0	Trekkraft i tråd
DB5 OP Pointers				
2	Recipe_name	String [254]	0.0	Innehalde namna til alle kabeloppskrifta i maskinen
SAMP				
DB45 Spooler 1 Counter data				
14	OUT_metercounter	DINT	20.0	Teller som teller kvar meter og går i reset når set meter er nådd.
DB100 General FC data				
22	Seconds_counter	DINT	14.0	Teller sekunder under pågåande prosess.
23	Minutes_counter	DINT	18.0	Teller minutt under pågåande prosess.
24	Hour_counter	DINT	22.0	Teller timar under pågåande prosess.
DB105 Speed line data				
1	Speed_line_set	INT	0.0	Farten prosessen er innstilt på.
DB299 Recipe used data				
2	Recipe	«recipe _mask»	2.12	Oppgir identifikator til kva tråd og prosess. Data typen her er ein tillaga med fleire ledd.

Tabell 3 PLS Datablokkar

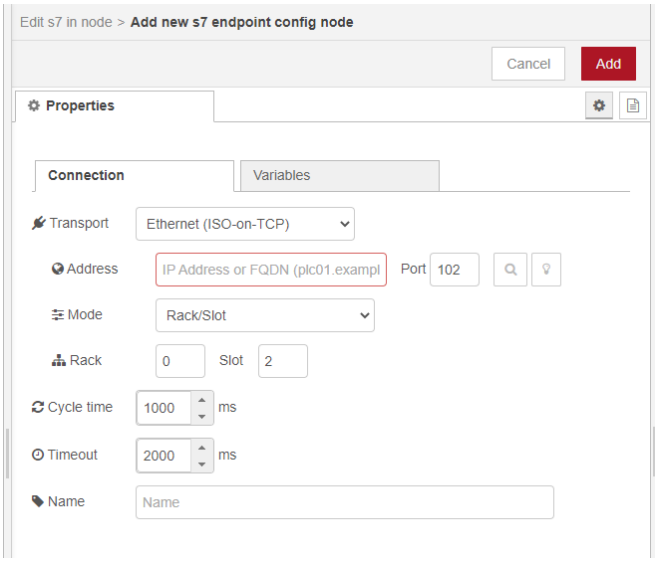
I Tabell 3 blir det vist kva variablar som blir brukt i bacheloroppgåva frå PLS. Der DB, datatype og offset er adressera for kvar enkel variable som er brukt i Node-RED. I tillegg til å henta informasjon om produksjon, så blir det også oppgitt identifikasjonar på kva som blir produsert, farten på produksjonen og alarmar skulle det skje brot på trådane. Det er meir data som kan hentast frå PLS som kunne blitt brukt, men for avgrensa til leveringspresisjon så er det dei oppgitte DB som blir brukt til oppgåva.

Som nemnt i kapittel 5.1.2 så kan det lastast ned forskjellige palettar med ein eller fleire noder som har ulike eigenskapar. For kommunikasjon med PLS så finst det eige palett for S7 PLS «node-red-contrib-s7». Frå denne paletten er det 3 noder, men sida det skal berre lesast frå PLS så er det berre noden «s7 in» som blir brukt. Når noden er dratt ned til vindauget kalla «Workspace» så må noden innstillast til PLS-en som vert brukt.

Med å trykka innpå noden så kjem eit vindauget opp kor IP-adressa til PLS trengst, sjå Figur 16. IP-adressa blir funnet som vist i Figur 15.

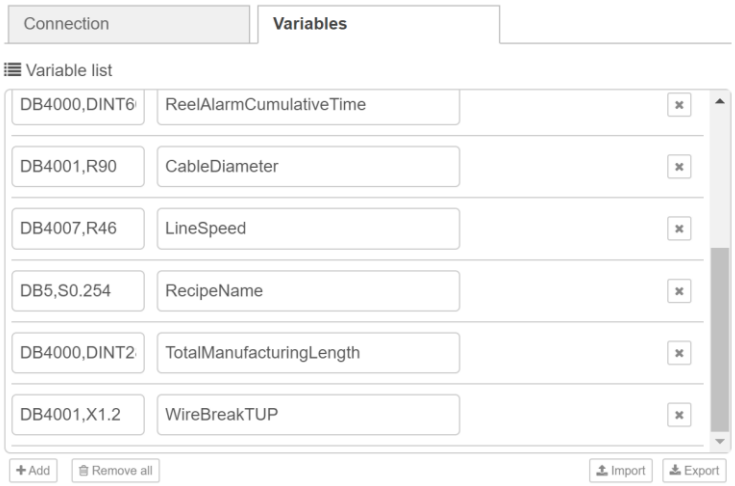


Figur 15 PLS IP-adressa



Figur 16 s7 in node

Etter det er gjort så må DB frå PLS bli oppgjeve i «Variables» i noden. I Figur 17 er det vist korleis DB blir skrevet inn i noden. Det skal starte først med DB namnet etterfølgt med datatype og offset adressa. Nokre datatypar kan førast inn med berre forbokstaven sin, og med BOOL blir det skreve som 'X'.



Figur 17 Innsetting av DB i Variables

Om det er feil med tilkoplinga mellom PLS og Node-RED så vil det kome ein feilmelding på debug vindauget til høgre på sida. Eksemplar på feilmeldingar som har oppstått under oppgåva:

- ERROR: connect EHOSTUNREACH

Når desse feilmeldinga oppstod så var det at det var skrivefeil i «Variables» i PLS noden. I andre tilfelle så er det at feil IP-adresse i noden.

5.1.4 Behandling av data i Node-RED

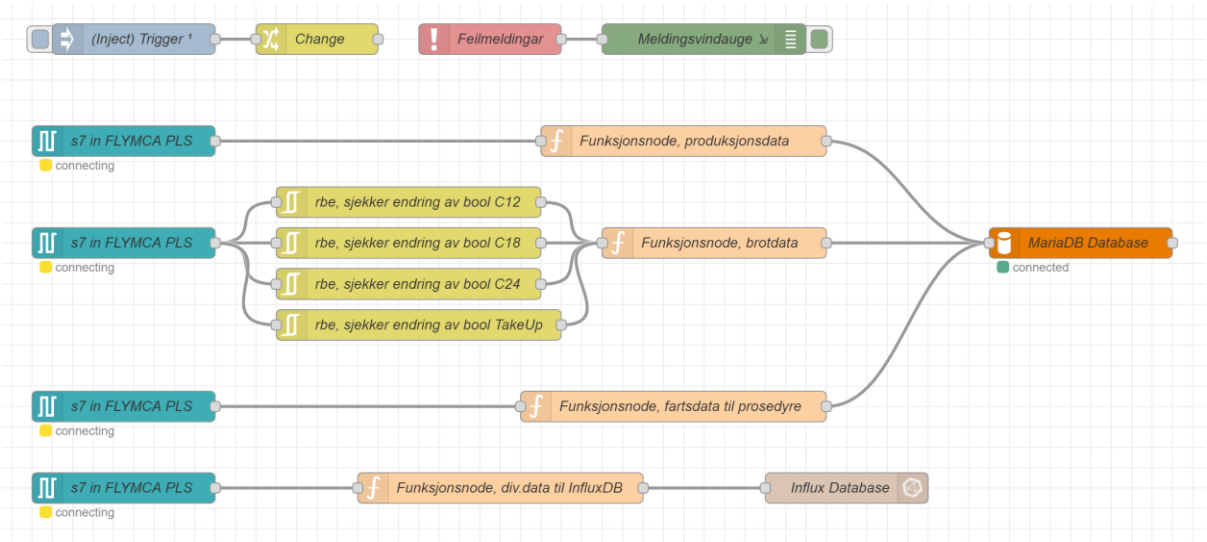
Dette kapitel beskriver korleis Node-RED er bygget opp til for å kunne gjere dei ulike hovudfunksjonane produksjonsdata(3.3) og brotanalyse(3.4). Det er også forklart korleis programmering i Node-RED fungere. Dei siste delkapitla beskriver korleis dei ulike programkodane i funksjonsnoda fungere og kva oppgåver dei gjere.

Oppgåver i Node-RED trenger å gjere:

- Sortere vekk data som ikkje er ynska skal bli sendt inn til PDB
- Bestemme kva tabell eller prosedyre data skal bli sendt til og kva database
- Hugsa verdiane som kjem inn
- Omrekna verdiar til passande måleining

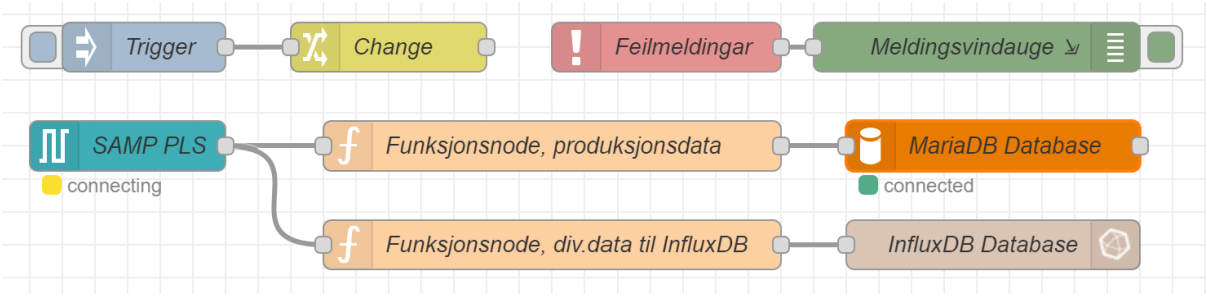
Node-RED er eit meldingsbasert system som gjere at data blir sendt med meldingar. Variablane blir frakta som «message payload», som vil seie at meldinga innehalde dei ulike parametrar. Desse blir kallet opp med «msg.payload.'variabelnamn'». I dette prosjektet er det noden «s7 in» som lager desse meldingane med dei ulike variablane. Desse meldingane følgjer dei grå linjene.

I Figur 18 viser det dataoverførings oppsettet frå FLYMCA PLS til MariaDB. Denne er delt opp i fleire meldingsstraumar som beskriver dei ulike oppgåvene.



Figur 18 FLYMCA Node-RED

For SAMP er Node-RED noko enklare oppbygd enn FLYMCA med færre funksjonar. Hovudfunksjon for brotanalyse er ikkje med for SAMP på grunn av arbeidsmengde.



Figur 19 SAMP Node-RED

s7 in

«s7 in» er noden som spesifisere kva data som skal hentast frå DB, og sende dei som melding vidare til funksjonsnoden. PLS noden sender variablane sine i eit bestemt intervall som blir bestemt i noden, sjå delkapittel 5.1.3 for nøyaktig oppset.

rbe

«rbe» er den noden som kunn sender melding vidare når det skjer ein endring. Denne er kunn brukt for brot, sidan funksjon nodane gjere denne funksjonen for dei andre meldingstraumane.

function

«function» er den noden som gjer det meste av oppgåvene i Node-RED. I denne noden kan det lages funksjonar med hjelp av tekstbasert programmering. Det er vidare forklart korleis programmering i funksjonsnode fungere.

Spørjesetningar er sentral del av programmet i funksjon noda i dette prosjektet. Blir vilkåra oppfylt så ender det med enten eit «CALL» eller «INSERT INTO» kommando. «CALL» blir brukt for å senda eit kall til prosedyre med dei ynskja data, mens «INSERT INTO» legger man direkte verdiane inn til tabell i MariaDB. Skal meldinga bli sendt vidare til databasen blir dei lagt til i ein SQL-kode som er meldinga overskrift(msg.topic). Denne meldings overskrifta blir skrevet som ein tekst datatype, og er samansett av tekst og variablar. Dette blir vist i linje 8 i Figur 20.

```
var KjoreMin

if ( msg.payload.LineSpeed != flow.get('ForgjeFart') && flow.get('ForgjeFart') == flow.get('Forgje2Fart')  && flow.get('ForgjeFart') > 0 )
{
    KjoreMin = msg.payload.ReelProductionCumulativeTime - flow.get('ForgjeKjoreMin');
    if(KjoreMin >= 0)
    {
        msg.topic = "CALL TimerFart('"+global.get('RName')+"','"+flow.get('ForgjeFart')+"', '"+KjoreMin+"','"+global.get('KabelDiameter')+"');";
    }
    else
    {
       msg = null;
    }
    flow.set('Forgje2Fart',flow.get('ForgjeFart'));
    flow.set('ForgjeFart',msg.payload.LineSpeed);
    flow.set('ForgjeKjoreMin',KjoreMin);
    msg.payload= null;
}
```

Figur 20 Utklipp i Program frå funksjonsnode

Programmet som er laget i funksjonsnoden er lagt opp til å slette variablane med kommando «msg.payload = null;» for å spare trafikk. Skal meldinga ikkje vidaresendast, blir kommando «msg=null;» brukt. «Return msg;» på slutten blir brukt for å sende meldings overskrifta vidare.

global.set/flow.set blir brukt for å kunne lagre variablane til neste gang, sånn at funksjonsnoden kan samanlikna variablar. Global.set og flow.set fungere på same måte, men det er ulikt kor dei er tilgjengeleg. Global er tilgjengeleg i heile Node-RED, mens flow er kunn tilgjengeleg i den aktuelle flow (aktuelle vindauget). For å lagre ein ny verdi i desse variablane blir .set brukt, og for å hente variabelen blir .get brukt. Sjå døme i Figur 20 for utførsel.

Blir global og flow brukt i IF-setningar må dei bli satt verdiar ved oppstart. Dette gjerast med å bruka «**inject**» node og «**change**» node. Der «inject» er for å sende ein melding ved oppstart ein gang og til «change» noden som setter variablane global/flow.

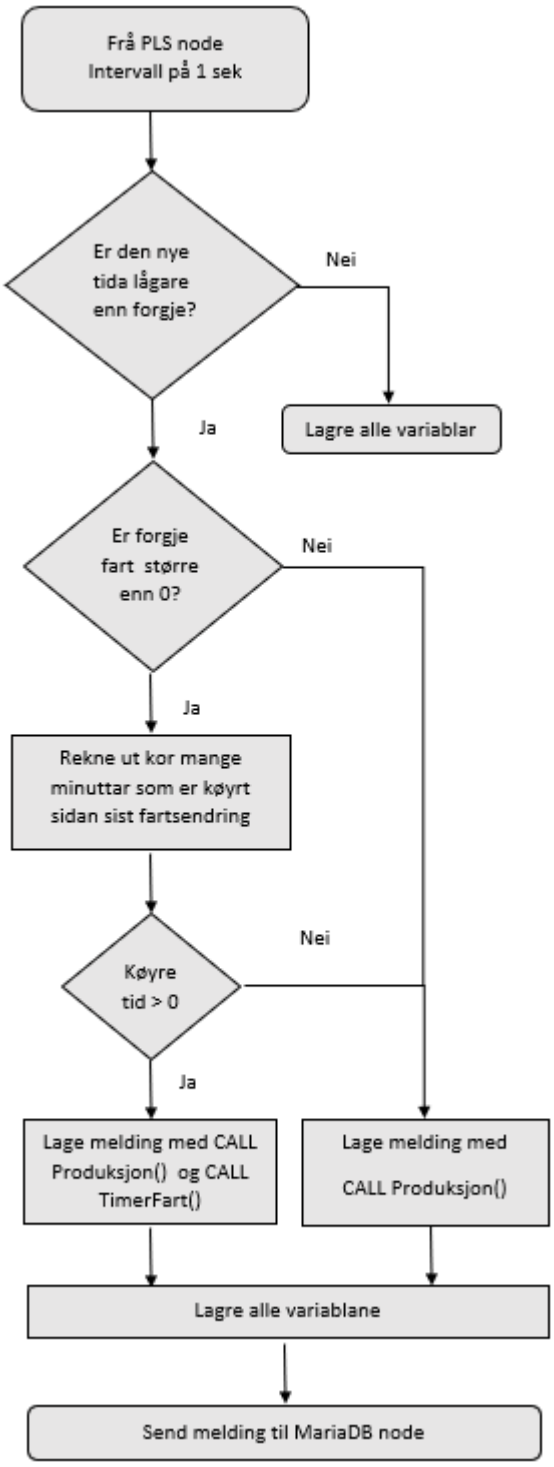
I vedlagte filer ligger det programma i funksjonsnodane i Node-RED. Og i delkapitla under er dei forskjellige programma i funksjonsnoden forklart med flytskjema.

influx batch/mysql

Kvar av meldingstraumane endar i to databasenoder, ein for «mysql» noden, og den andre for «influx batch» noden. I desse nodane må IP-adressene til maskinvare med databaseprogramma bli oppgitt.

5.1.4.1 Funksjonsnode: Produksjonsdata FLYMCA

Denne funksjonen er til for å lagre tidsdata inn i produksjonstabell i databasen. Tidsdata er den tida maskina bruker på å produsere dei ulike produkttypane, og den tida som blir brukt å sette opp maskinen. For å unngå at data frå PLS straumar konstant i tabell, måtte det løysast med at funksjonen må gjenkjenna når ein prosess er ferdig. Dette blir gjort med at funksjonen sjekkar om den aktive produksjonstida «Reel: Production cumulative time» (Tabell 3) er lågare enn den forgje produksjonstid som er lagra(global.set). På denne måten vil funksjonen leggja merke til at prosess er ferdig når parametrane til maskinen blir resetta. Sida alle parameterane er resetta ved denne tida, er ingen av data lengre reelle. Derfor må funksjonen bruke dei lagra parametrar frå sist funksjonen køyrte. I Figur 21 viser det eit flytskjema på korleis prosessen i funksjonsnoden går.



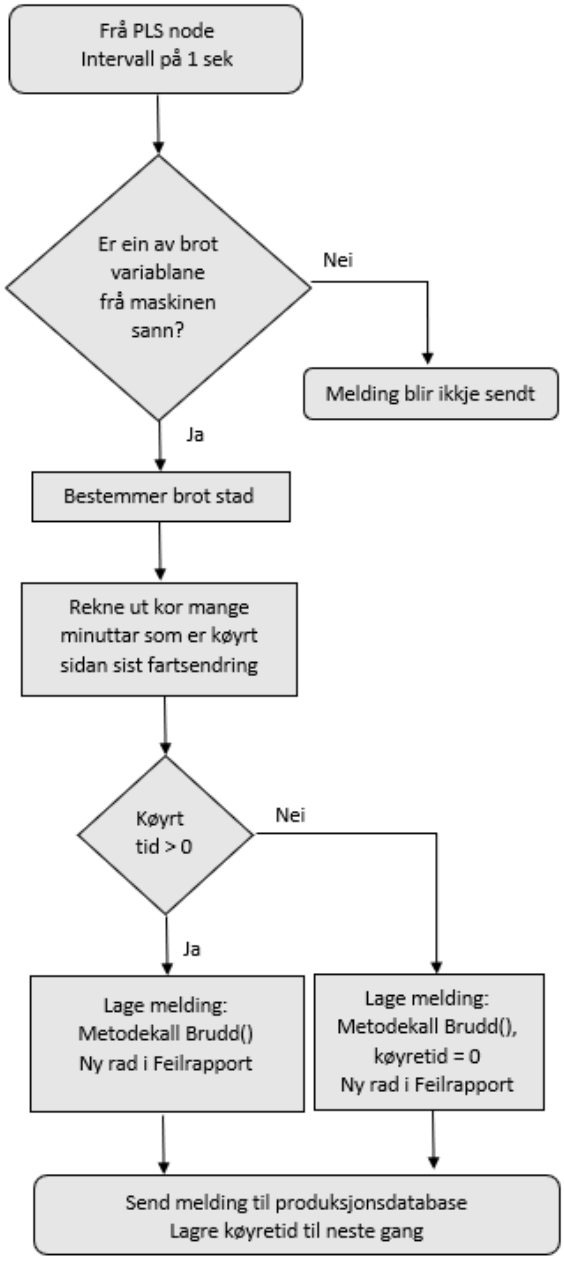
Figur 21 Flytskjema Produksjonsdata FLYMCA

Denne funksjonen kallar opp prosedyren Produksjon i MariaDB og sender produkt ID, produksjonstid, setuptid, trommelnummer og produsert lengde inn til PDB. Sjå kapitel 5.2.4.1 tabellar og prosedyre.

Er ikkje køyretida(køyretid større enn null) til ein fart registrert, vil denne også sende eit kall i prosedyre TimerFart. Det er liten sannsynleg denne delen av funksjonen vil bli brukt, sidan farten har mest sannsynleg har vert null før denne funksjonen køyrte.

5.1.4.2 Funksjonsnodane: Brotdata

Denne funksjonsnoden er til å registrere brot på trådane/kordelar, på FLYMCA. Kvar del av produksjonslinja har kvar sin BOOL variabel for brot på kordelar. Cage- 12, 18, 24 og Take Up (Figur 7) er dei ulike maskindelane til FLYMCA. Når ein av desse variablane blir sann er det første kriteriet oppnådd. For å unngå at denne funksjonen skal køyre fleire gangar ved eit brot, er det brukt «rbe» node for å kunne sende når det skjer ein endring, sjå Tabell 2.



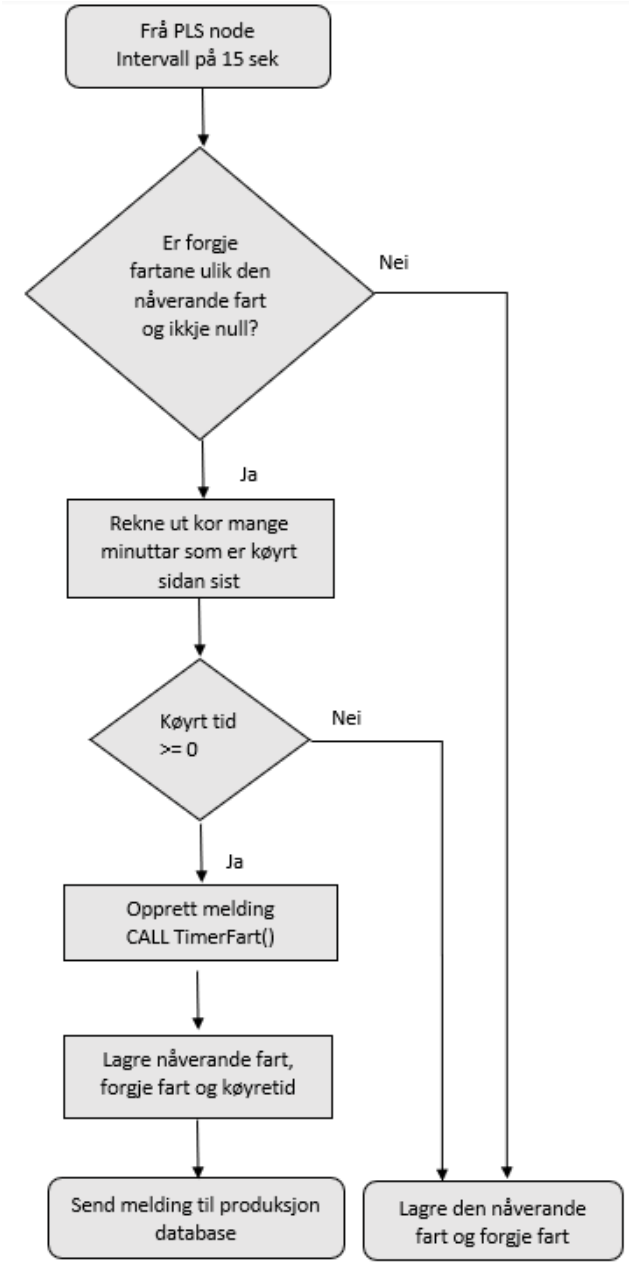
Figur 22 Flytskjema Brot

Etter dette blir det laga ein SQL-kommando for å legge inn ein ny rad i Feilrapport med informasjon som produkttype, trommelnummer, diameter, brotstad, fart og produsert lengde inn til tabell feilrapport i PDB.

Denne funksjonen finner også ut kor mange timer som er køyrt og sender til prosedyre Brudd

5.1.4.3 Funksjonsnode: Fartsdata til prosedyre

Denne funksjonen er til å telje kor mange timar som er køyrt på den spesifikke farten. Skjer det ein fartsendring vil denne funksjonen sende inn kor mange minuttar som er køyrt på den farten den var på. For å hindre at dei små endringane i fart og endringsperioden frå ein fart til ein annan blir registrert, må den same farten vare over to periodar. Altså totalt 30 sekunder i same fart. Er farten lik null skal ikkje dette bli registrert som ein eigen fart. Denne funksjonen sender oppkall til prosedyren TimerFart med variablar produkttype, fart og køyretid. Sjå kapittel 5.2.4.2 for prosedyren.

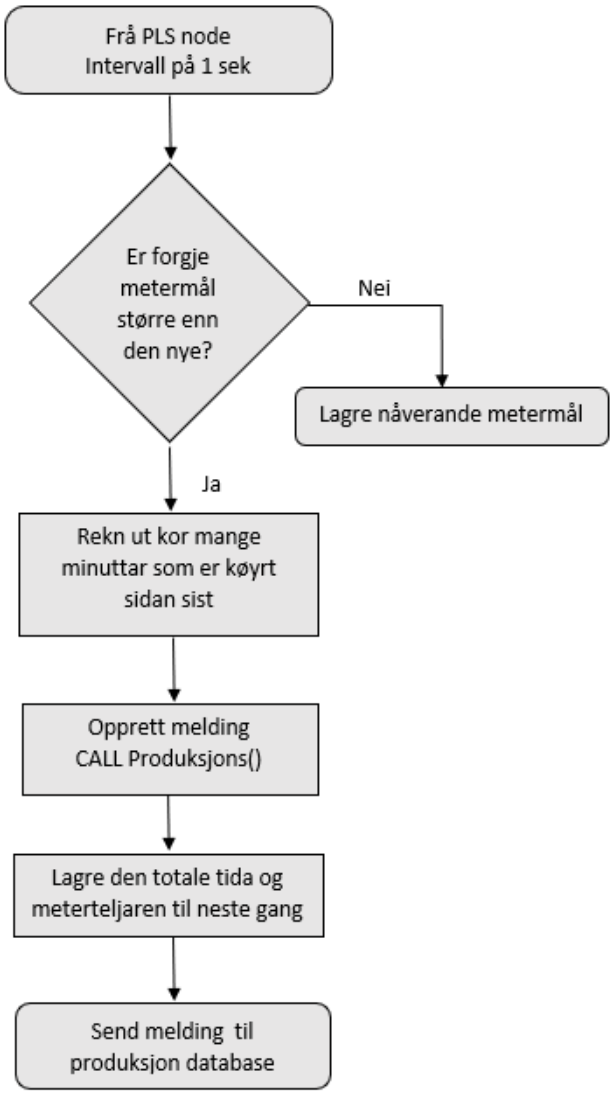


Figur 23 Flytskjema Fartsdata

5.1.4.4 Funksjonsnode: Produksjonsdata SAMP

Denne funksjonen er til for å sende den brukte produksjonstida og produsert lengde på den aktuelle produkttype til PDB. SAMP har ikkje ein eigen teljar for produksjonstida og må derfor bruke andre metodar. For å finne produksjonstida brukar funksjonen dei ulike variablane (sekund, minutt, timer) for timeteljar til produksjonslinja. Timane og sekundane er omgjort til minuttar og adderer i saman. Timeteljaren er den totale produksjonstida akkumulert sida SAMP blei teken i bruk. På grunn av det må produksjonstida bli rekna ut med å finna differansen mellom den nyaste totale produksjonstida, med den totale produksjonstida frå sist køyring:

$$Nyaste\ totale\ produksjonstid - sist\ totale\ produksjonstid = reell\ produksjonstid\ for\ produktet$$



Figur 24 Flytskjema Produksjonsdata SAMP

Det er ingen av variablane som viser når maskinen er ferdig og begynner på neste trommel. Derfor blir meterteljaren brukt. Når meterteljaren er lågare enn forgje metermål, må maskinen vera reset.

Denne funksjonen sender oppkall på prosedyren Produksjon med variablar som produkttype, produksjonstid og produsert lengde. Sjå kapittel 5.2.4.1 for prosedyre.

5.2 Oppsett av PDB

Som nemnd i kapittel 0 så er PDB ein lagringsplass for data frå produksjon. For å bruka og styra database serveren til MariaDB så kan leietekst brukast. I denne oppgåva blir det brukt eit hjelpeverktøy som heiter HeidiSQL (0). Klienten gjer det enklare og meir oversiktleg med tabell, og sjekkar om det kjem data inn frå Node-RED. Val av database program blir utdjupa i kapittel 4.2.

5.2.1 Brukar og database i MariaDB

For å få ein database til å fungera så må det opprettast ein eller fleire brukarar, for ein spesifikk eller fleire databasar. Det vil sei at brukaren må få løyve for å bruka databasen. Desse løyva setter avgrensing med tilgang til databasane, noko som har mykje å sei for datasikkerheita til systemet. For å lage databasar og brukar så må det gjerast i leietekst form. Gjeremåten er lik for både Windows og Ubuntu.

For å lage ein database i MariaDB så må dette skrivast i leietekst:

```
CREATE DATABASE databasenamn_skrives_her;
```

Nedanfor visar eit eksempel på korleis ein kommandoen som skrivast i leieteksten for å opprette brukar, for alle IP-adresser med '%'. For å lage ein for eit bestemt adresse så byter du berre om «%» med ein gyldig IP-adresse eller «localhost». Man kan også velja mellom å bruka passord med bruk av «IDENTFIED BY».

```
CREATE USER 'brukar_namn_her'@'%' IDENTIFIED BY 'passord_skrives_her';
```

Med å oppretta brukar så må det avgjerast kva type privilegium denne brukaren skal ha til databasen. Privilegium som kan gis er eksempelvis moglegheit til å lage/sletta tabellar og lese/skrive i data i databasen.

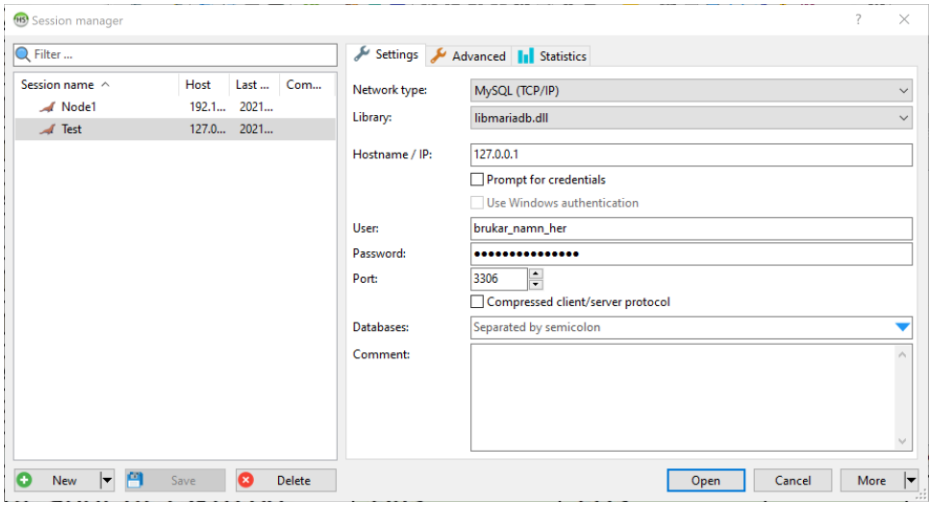
```
GRANT ALL PRIVILEGES ON *.* TO 'brukar_namn_her'@'%' IDENTIFIED BY 'passord_skrives_her';

FLUSH PRIVILEGES;
```

Her er det oppgitt ein kommando som gir full privilegium til brukaren, som vil sei den vil ha fri tilgang til å administrere databasar, brukarar, tabellar, funksjonar og prosedyrar.

For å avgrensa privilegium til ein bestemt database så bytter man «*.*» til «databasenamn_skrives_her.*». Det finst fleire nivå av privilegium, men dei to som er nemnd her er nok for denne oppgåva. «FLUSH PRIVILEGES» oppdaterer dei nye privilegia til brukarane til databasen, slik at man skal sleppe å måtte gå ut av MariaDB og inn igjen for at kommandoane skal slå inn.

Etter det er oppretta ein brukar kan HeidiSQL brukast for å administrere databasen på vindaugeform. I Figur 25 viser det eit eksempel på start vindauget til HeidiSQL og kva som må fyllast inn. Her må det leggast til kva databaseprogram som skal opprettast kontakt med, IP-adressa, brukar, passord og port nummer.

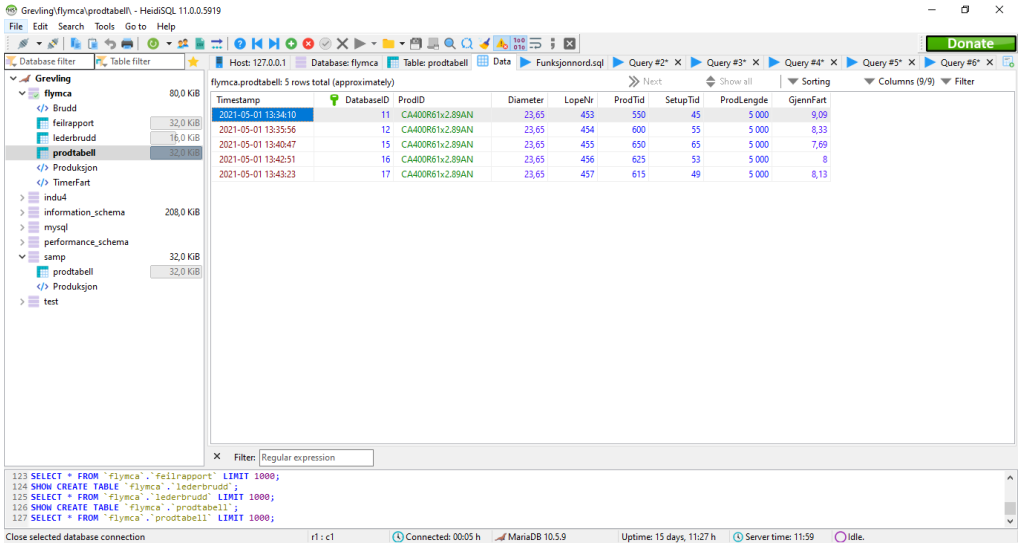


Figur 25 HeidiSQL start vindauge

I delkapittel 5.1.1 er det nemnd at «127.0.0.1» er ein standard IP-adresse for lokalnettverk. Men for at eksterne maskinvarer skal få tilgang til databasen, så må IP-adressa til den gjeldande maskina med databaseprogrammet bli oppgitt i «Hostname/IP».

I Windows kan man enten gå på Wi-Fi-innstillingane eller bruka leietekst og bruka kommandoen «ipconfig». Det dukkar opp fleire linjer med nettverksadresser, men for oppgåva så er det IPv4 adressa i Wireless LAN adapter Wi-Fi feltet som skal brukast.

I andre OS-ar som Ubuntu og Raspberry Pi OS som begge er Debian baserte operativsystem, så blir IP-adressa funne på ganske lik måte som i Windows. I «Terminal» som er det same som leietekst, så blir IP-adressa funne med å skriva inn «hostname -I».



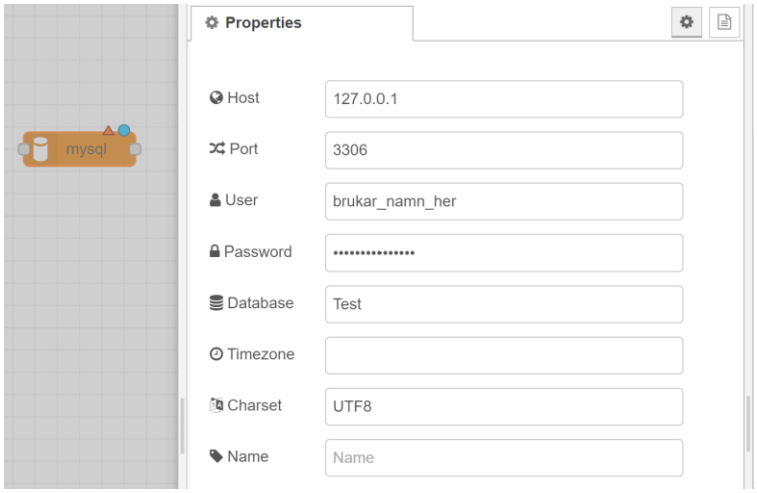
Figur 26 HeidiSQL hovudvindauge

Her vil det vera mogleg å lage til tabellar, prosedyrar og funksjonar for databasen. Her er det viktig å merkja seg, å ikkje sletta noko av dei andre databasane som ligger der som standard (mysql, information_scema) ettersom det kan føre til feil.

5.2.2 Kommunikasjon til MariaDB

I Node-RED så blir det brukt eige palett for MariaDB, «node-red-node-mysql». Sjølv om det står MySQL, så verke det for MariaDB også.

I denne noden blir informasjon ført inn for å få kontakt med den ynskja databasen. Dette er informasjon som IP-adressa, brukar, passord og databasenamnet. Brukaren og databasen blir oppretta på same måte som i kapittel 5.2.1. I Figur 27 er det eit test eksempel av oppsett av noden, og fungerer det så skal det dukke opp ein liten grøn firkant under noden med skrifta «connected» ved sida av.

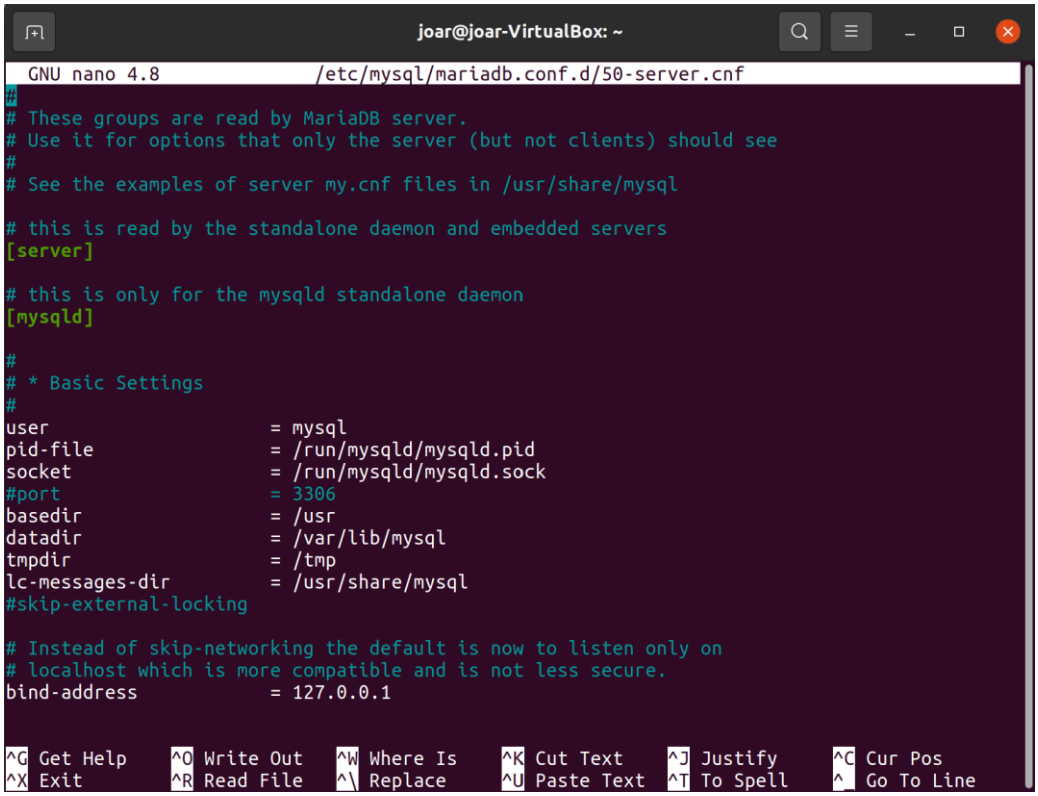


Figur 27 MySQL Node

Køyre databasen på ein Ubuntu maskin er det viktig å merka seg, at databasen er som standard innstilt på kunn på lokal tilkopling. For å endra innstillingar må konfigurasjonsfilen (.cnf -fil) til MariaDB bli lokalisert og endrast på. Kort forklart så må denne kommandoen bli skrevet i terminalen i Ubuntu:

```
sudo nano (‘/etc/mysql/mariadb.conf.f/50-server.cnf’
```

Med den kommandoen vil dette vindauget i Figur 28 dukke opp. Her må «bind-address» endrast til 0.0.0.0 i staden for 127.0.0.1. Dette gjer det mogleg for andre maskiner med andre IP-er kan få tilgang til MariaDB.



Figur 28 MariaDB fjerntilgang

5.2.3 Lagring av data i MariaDB

Det er oppretta to databasar i MariaDB for lagring av data, der ein er for FLYMCA og ein for SAMP. I dei innehalde det tabellar og prosedyrar. Prosedyre blir vist i neste delkapittel 5.2.4.

I Tabell 4 og Tabell 5 viser det ein oversikt over innhald til tabellane oppretta i FLYMCA og SAMP. I tabellane står det kva datatype kolonnen har, og beskriving over kva desse kolonnane skal innehalde.

Som kort nemnd i delkapittel 3.4 er det mindre innhald i SAMP i forhold til FLYMCA. Her er det berre ein tabell for produksjonsdata.

SAMP		
Prodtabell		
Namn	Datatype	Beskriving
Timestamp	TIMESTAMP	Slår inn den gjeldande tid og dato til rader som blir lagt inn i tabell.
DatabaseID	INT	Gir rad ein «index» i form av heiltal som nemnd i «Automatisk ID-nummer»
ProdID	CHAT	Viser produkt ID til det produserte produktet
Fart	DOUBLE	Viser farten som er sett til produksjonen
Diameter	DOUBLE	Viser diameter til det produserte produktet
ProdTid	DOUBLE	Viser den endelege tida brukt under produksjon
ProdLengde	INT	Endelege produserte lengda av produksjonen

Tabell 4 MariaDB SAMP Tabell oppsett

I FLYMCA er det oppretta tre tabellar. Der produksjonstabell innehalde produksjonsdata, feilrapporttabell innehalde brotdata og den siste tabell for anslag for brothyppigheit.

FLYMCA		
Prodtabell		
Namn	Datatype	Beskriving
Timestamp	TIMESTAMP	Slår inn den gjeldande tid og dato til rader som blir lagt inn i tabell.
DatabaseID	INT	Gir rad ein «index» i form av heiltal som nemnd i «Automatisk ID-nummer»
ProdID	CHAR	Viser produkt ID til det produserte produktet
Diameter	DOUBLE	Viser diameter til det produserte produktet
LopeNr	INT	Løpenummeret til trommel til den gjeldande produksjonen
ProdTid	INT	Viser den endelege tida brukt under produksjon
SetupTid	INT	Tid brukt for oppsett av produksjon frå Node-RED
ProdLengde	INT	Endelege produserte lengda av produksjonen
GjennFart	DOUBLE	Gjennomsnittsfarta berekna frå prosedyren Produksjon, sjå 5.2.4.1
Feilrapport tabell		
Tidspunkt	TIMESTAMP	Slår den gjeldande tid og dato til rader som blir lagt inn i tabell.
DatabaseID	INT	Gir rad ein «index» i form av heiltal som nemnd i «Automatisk ID-nummer»
ProdID	CHAR	Viser produkt ID til det produserte produktet
LopNr	INT	Løpenummeret til trommel til gjeldande produksjon
Diameter	DOUBLE	Viser diameter til det produserte produktet
BruddSted	CHAR	Melding kor brot har oppstått i FLYMCA
Fart	INT	Viser køyrefarten produksjonen hadde når brotet skjedde
Lengde	INT	Viser den produserte lengda frem til brot
Lederbrudd tabell		
ProdID	CHAR	Viser produkt ID til det produserte produktet
KjoreFart	INT	Viser den køyrefarten maskinen har køyrt på i meir enn 30 sek.
AntBrudd	INT	Viser antall brot som har skjedd til den gjeldande produktet. Prosedyren Brudd legger saman antall, sjå 5.2.4.2
AntTimer	DOUBLE	Legger til køyrt tid til det gjeldande produktet, sjå 5.2.4.2
Timerbrudd	DOUBLE	Viser anslaget timar det tek for eit brot skal skje, sjå 5.2.4.2

Tabell 5 MariaDB FLYMCA Tabell oppsett

Vidare er det ein generell forklaring av korleis ein tabell blir oppretta i MariaDB.

```
CREATE TABLE `prodtabell` (
	`Timestamp` TIMESTAMP NULL DEFAULT current_timestamp(),
	`DatabaseID` INT(10) NOT NULL AUTO_INCREMENT,
	`Diameter` INT(11) NULL DEFAULT NULL)
```

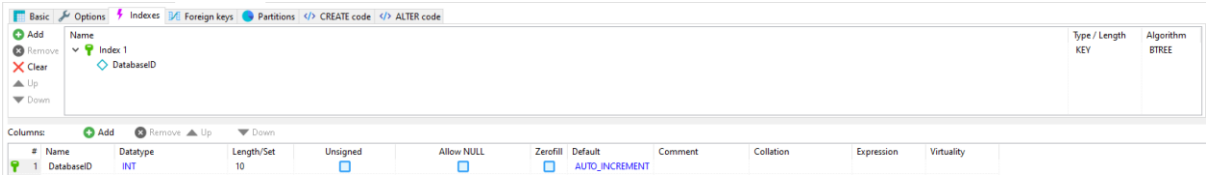
Over er det eit eksempel av SQL kodane som er nytta for å opprette ein tabell. Kolonne starte med namn etterfølgt av datatype, kolonneinnstillingar og avsluttar med eit komma. Det er ved innlegging av ny rad kolonneinnstillingar har noko å seie og er vidare forklart under.

Null/Not null

Dette blir brukt til å spesifisert om det er lov eller ikkje, med ingen data i denne kolonnen. Ved å bruke null er det lov med ingen data og NOT NULL er det ikkje lov med ingen data i denne kolonnen.

Automatisk ID-nummer

AUTO_INCREMENT blir brukt for å automatisk telje opp med ein for kvar gong det blir lagt til ein ny rad. For at dette skal fungere må denne kolonna vera av datatype integer. I denne oppgåva blir denne brukt for å lage «indexes» på kvar rad med data. Ein «index» er ein unik nøkkel som skal representera kvar rad for å gi dei ein identifikator, noko som vil gjera det lett å skilja ut radane.



Figur 29 AUTO_INCREMENT, INDEX

Automatisk tidsstempel

Ved bruk av CURRENT_TIMESTAMP som DEFAULT på kolnonna me TIMESTAMP som datatype, så vil den aktuelle tidspunktet som denne raden blei lagt til bli ført opp automatisk.

5.2.4 Prosedyrar og funksjoner

Prosedyrar og funksjonar blir som eit lite program som køyrer i databaseprogrammet. Desse typane kan brukast til å gjere ulike operasjonar og oppgåver i MariaDB. Forskjellen mellom ein prosedyre og ein funksjon er at ein prosedyre kan kunn ta imot data/variablar, mens funksjonar kan også returnera data/variablar. Prosedyre og funksjonar blir omtalt som prosedyre vidare, men funksjoner har dei same moglegheitene. Først er det ein generell forklaring av korleis ein prosedyre blir oppretta. Deretter er det ein forklaring av dei prosedyrane som er nytta i dette prosjektet, delkapitel 5.2.4.1 og 5.2.4.2.

Oppkall av ein prosedyre blir gjort med kommando `CALL namn ('variabe1','variabel2',…)` hugs det er viktig med rekkefølga er lik, som rekkefølga til dei som er lagt til i prosedyren. Ein prosedyre blir oppretta ved kommando `CREATE PROCEDURE` og deretter ynskja namn på prosedyren. Døme på ein prosedyre:

```
CREATE PROCEDURE `ynskja namn`(
      IN `InnData1` DataTYPE,
      IN `InnData2` DataType,
      IN `InnData3` DataType,
)
NOT DETERMINISTIC
CONTAINS SQL
SQL SECURITY DEFINER
COMMENT ''
BEGIN
'Kva programmet skal gjere'
END
```

IN/OUT/INOUT

Etter namn på prosedyre blir inn og ut variablane deklarert ved å markere dei ved `IN` for variablar som skal inn til funksjonen og `OUT` til dei som skal returnerast. Det er viktig at rekkefølgja blir lagt merke til, sidan dette er viktig for oppkallet av denne prosedyren. Etter namne på denne variabelen blir datatype spesifisert.

Prosedyre/funksjon innstillingar

Når prosedyre blir oppretta blir det definert nokre val som `CONTAINS SQL`, `NO SQL`, `READS SQL DATA`, og `MODIFIES SQL DATA`. Desse blir brukt til å seie til serveren kva prosedyren skal gjere på serveren.

`MODIFIES SQL DATA` betyr at prosedyre/funksjonen innehelder operasjonar som endra på data i databasen. Døme på SQL kommandoar som gjere endringar på data er `DELETE, UPDATE, INSERT ,REPLACE.` `READS SQL DATA` betyr at prosedyren inneheld operasjonar som kunn leser og hentar data frå databasen. Mens `CONTAINS SQL` og `NO SQL` blir lite aktuelt å bruka. `CONTAINS SQL` blir brukt når det innehalde minst ein SQL kommando, men som ikkje endre eller leser data frå databasen. Til dømes `SET og DO.` Mens `NO SQL` er ikkje mogleg å bruke i MariaDB.

Deterministisk / ikkje Deterministisk

Ved å velje DETERMINISTIC / NOT DETERMINISTIC bestemmer om funksjonen returnere same variable på same plass. Dette har kunn noko å sei ved bruk av funksjonar og ikkje prosedyre.

Rutine kroppen

Rutine kroppen «ROUTINE BODY» er programdelen i prosedyren. Det er i denne delen, som det ynskjer oppgåver blir utført. Det er kunn moglegheit for å bruke SQL-språk til å utføre oppgåver. Denne delen starte med `BEGIN` og ender med `END`.
Nokon viktige kommandoar for å kunne programmere:

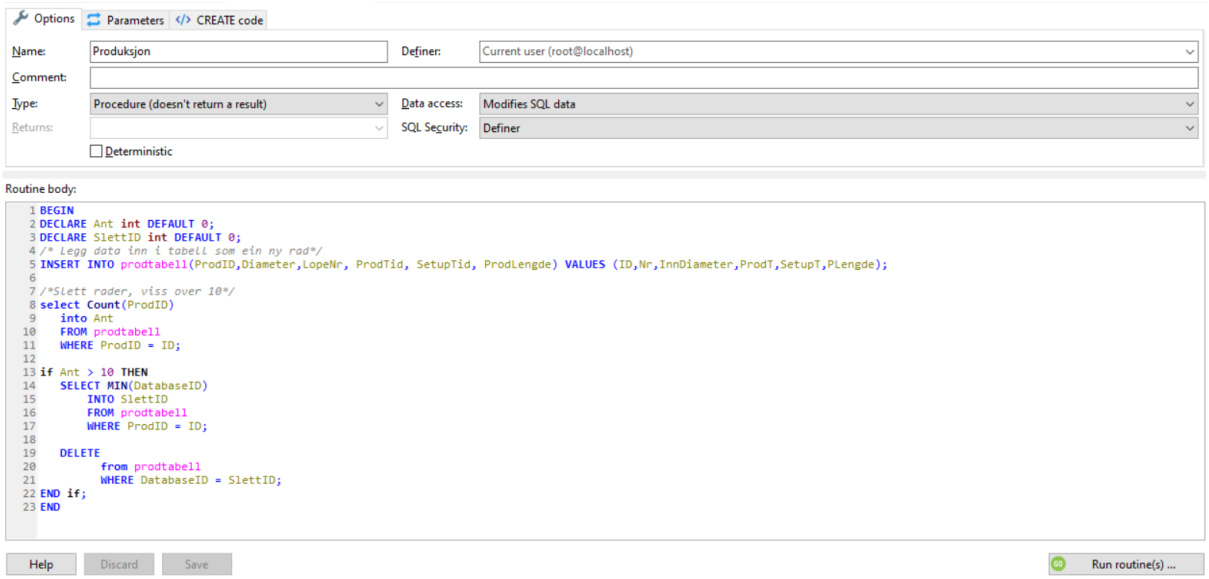
```
DECLARE namn datatype DEFAULT verdi;
```

```
IF (Kriterie) THEN
Oppgåve
END IF
```

```
SET variabelnamn = verdi
```

I dette prosjektet har me valt å bruke HeidiSQL til å køyre SQL kommando inn til MariaDB for oss. For å opprette ein prosedyre eller funksjon i HeidiSQL, høyre klikke på databasen og vel «Stored routine». Da kommer bilde opp i Figur 30. Mykje av dei same vala finst i dette bilde.

- Parameters: er for å sette inn in/out variablane.
- Type: vel om det skal vera funksjon eller prosedyre.
- Data access: innstilling av prosedyren/funksjonen.
- CREATE code: er SQL koden som blir køyrt på MariaDB.

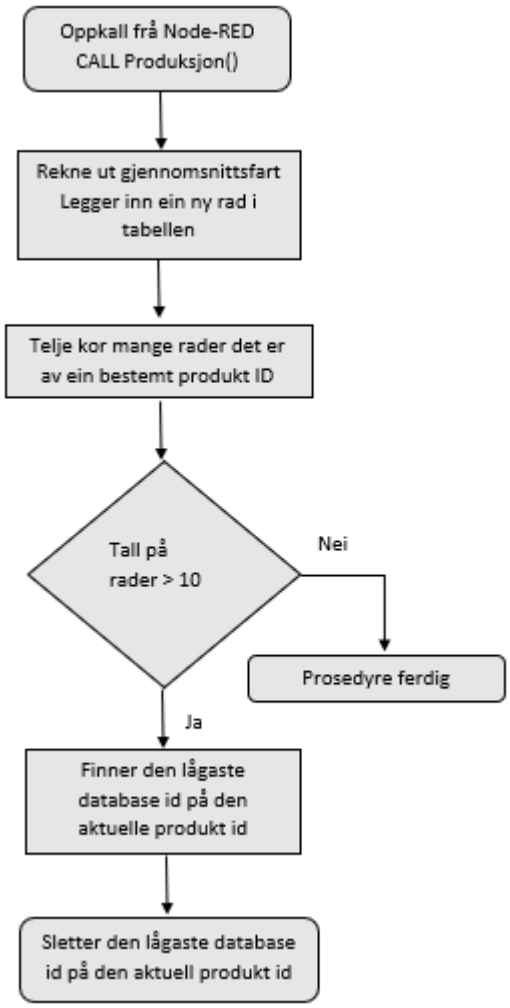


Figur 30 SQL Prosedyre

Figur 31 Parameter vindauge

5.2.4.1 Prosedyre: Produksjon

Denne prosedyren blir brukt til å registrera produksjonsdata frå FLYMCA og SAMP, inn i kvar sin «prodtabell» på kvar sin database. På SAMP er det nokon færre variablar sidan trommelnummer, og setuptid ikkje er på denne maskinen men fungere akkurat likt. Prosedyren skal samtidig sjekke kor mange rader det er av den aktuelle produkttypen frå før av. Er det fleire enn 10 rader, vil den eldste av dei bli sletta.

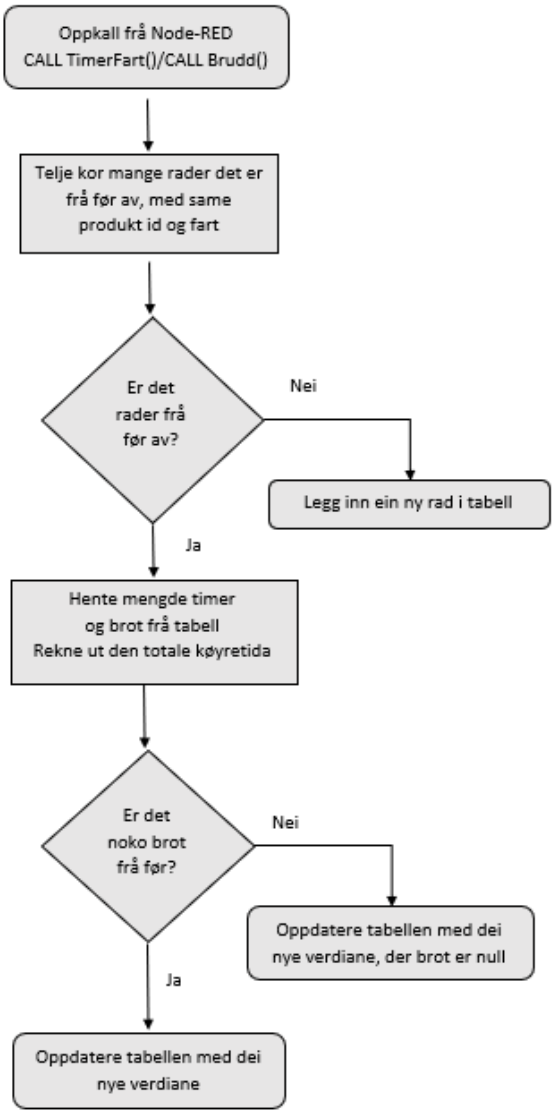


Figur 32 Flytskjema prosedyre Produksjon

5.2.4.2 Prosedyre: Brudd og TimerFart

Prosedyrane Brudd og TimerFart er nesten heilt like. Den einaste forskjellen er at Brudd telje opp med eit brot i tabellen under køyring mens TimerFart gjer ikkje det. Brudd har heller ikkje spørsmålet om det er brot frå før av, sidan dette alltid vil vera minst eit brot. Derfor er det laget kunn eit flytskjema av desse prosedyrane.

Begge blir brukt til å legge til dei timane som er køyrte på den spesifikke farten og produkttypen sidan sist endring. Finnes det ikkje dette frå før av vil den lage ein ny rad i tabellen «lederbrudd». Desse prosedyrane rekne også ut kor mange timer som kan forventast køyrt før eit brot oppstår. Dette blir utrekna med: $\frac{Timar}{Brot} = forventa\ timar\ køyrt\ før\ brot$

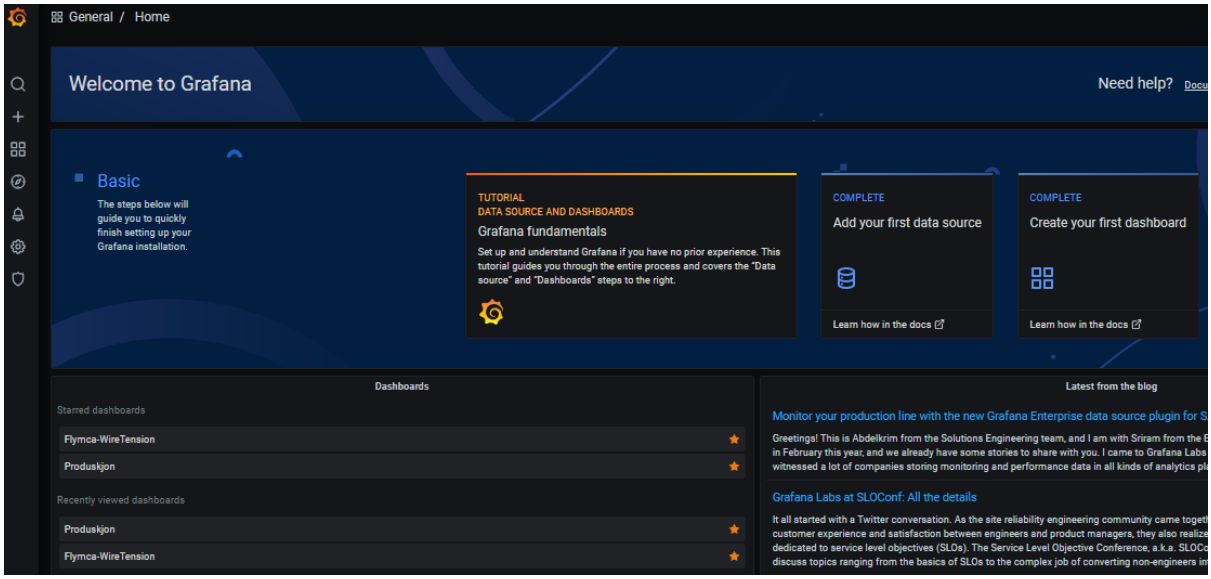


Figur 33 Flytskjema prosedyre Brudd og TimerFart

5.3 Visualisering

Grafana fungerer ganske likt som Node-RED når det gjeld med nedlasting og å ha grensesnittet i nettlesar. I Grafana kan det lagast dashbord som kan bestå av ein eller fleire panelar. Desse panela kan vera grafar, tabellar osv.

For å komme innpå Grafana skal den lokale eller IP-adressa til den maskinen som Grafana køyre på, etterfølgt av port inn i adressefelt. Som standard er Grafana oppgitt med port 3000. Eksempelet «127.0.0.1:3000». Når adressa er skrevet inn vil innloggings-vindauget dukke opp. Som ein standard førstegongs innlogging så er brukarnamn og passord «admin», men vil etter det spør om å lage eit nytt passord.



Figur 34 Grafana hovudside

Etter pålogging kommer hovudvindauget til Grafana opp, som vist i Figur 34. I hovudsida ligger det blant annet dei ulike funksjonar til Grafana og ein liste med dei ulike dashborda som er laget frå før av.

5.3.1 Bruk av dashbord

Når ein opnar Grafana vil eksisterande dashbord visast i ein liste på framsida. I dette prosjektet er det laget to dashbord for å vise data som er samla inn i dette prosjektet. Det eine dashbordet viser produksjonsdata og det andre viser køyreinformasjon. Bruk av desse dashborda er beskrive i dette kapittel.

Dashbord Flymca-WireTension, InfluxDB

Figur 35 viser dashbordet for køyreinformasjon frå FLYMCA maskinen. På dette dashbord viser den nåverande fart, trekkraft og historisk data. Trekkreftene kan brukast til å sjå om det skjer endringar før eit brot.

Figur 35 Dashbord med trekkraft og linjefart som panel

* Namn på dashbord, fortel kva maskin data blir henta frå

* Vel tidsrom for visualiseringane. Ser her frå 4/5-2021 til 12/5-2021.

*** Viser to panelar, den nåverande verdi som speedometer og historisk data som graf. X-akse = tid, Y-akse = verdi

* Viser maskina sin hastigheit.

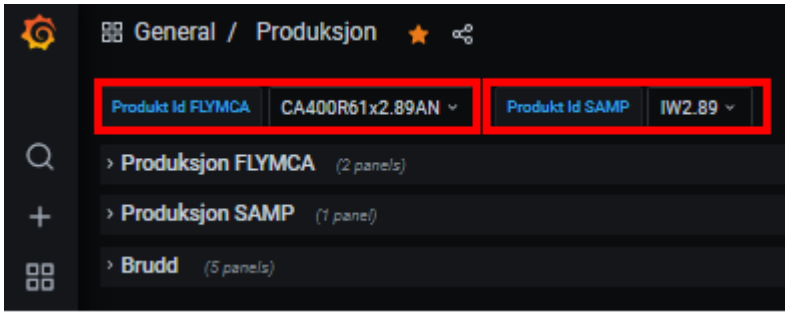
* Visar trekkraft på linja i maskindel Capstan(Figur 7).

* Viser trekkrafta på linja i maskindel Cage12, 18 og 24.

Dashbord Produksjon, MariaDB

I dette dashbordet blir det vist informasjon frå tidlegare produksjonar frå både SAMP og FLYMCA. Dashbordet består av fleire rader(Produksjon FLYMCA, Produksjon SAMP og Brudd) med forskjellige panelar. Desse radene kan leggast opp og ned etter ynskje.

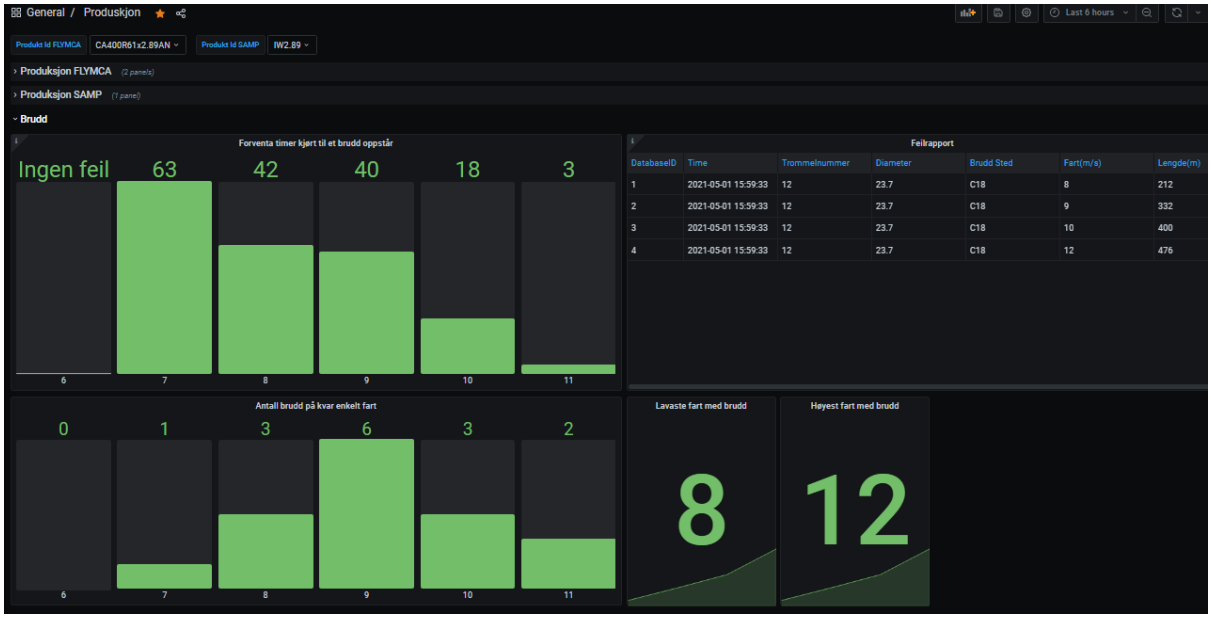
Øvst på dashbord er det to valmenyar(ringet rødt) Figur 36, her kan man velja kva produkt ID som skal framstillast.



Figur 36 Valmenyar/Variablar

Øvre rad på Figur 37 som blir kalla «Produksjon FLYMCA» framstiller produksjonsdata som tabell. Data framstilt i søylediagrammet er mengde timar køyrd på kvar enkelte fart. Med det kan operatørar sjå kva fart som har vore mest hyppig brukt.
Nedste rad som blir kalla Produksjon SAMP blir det framstilt produksjonsdata.

Figur 37 Dashbord med produksjonsdata frå MariaDB



Figur 38 Visualisering for fart og brot FLYMCA

Figur 38 visar visualisert data om brot frå MariaDB. I søylediagrammet øvst til venstre viser det anslag på kor mange timar køyring(y-akse) det tek, før eit brot skjer i ein bestemd fart(x-akse). I søylediagrammet under viser det samla mengde brot(y-akse) på kvar bestemd fart(x-akse). Tabellen som er vist til høgre, viser brotdata frå kvar enkel brot.

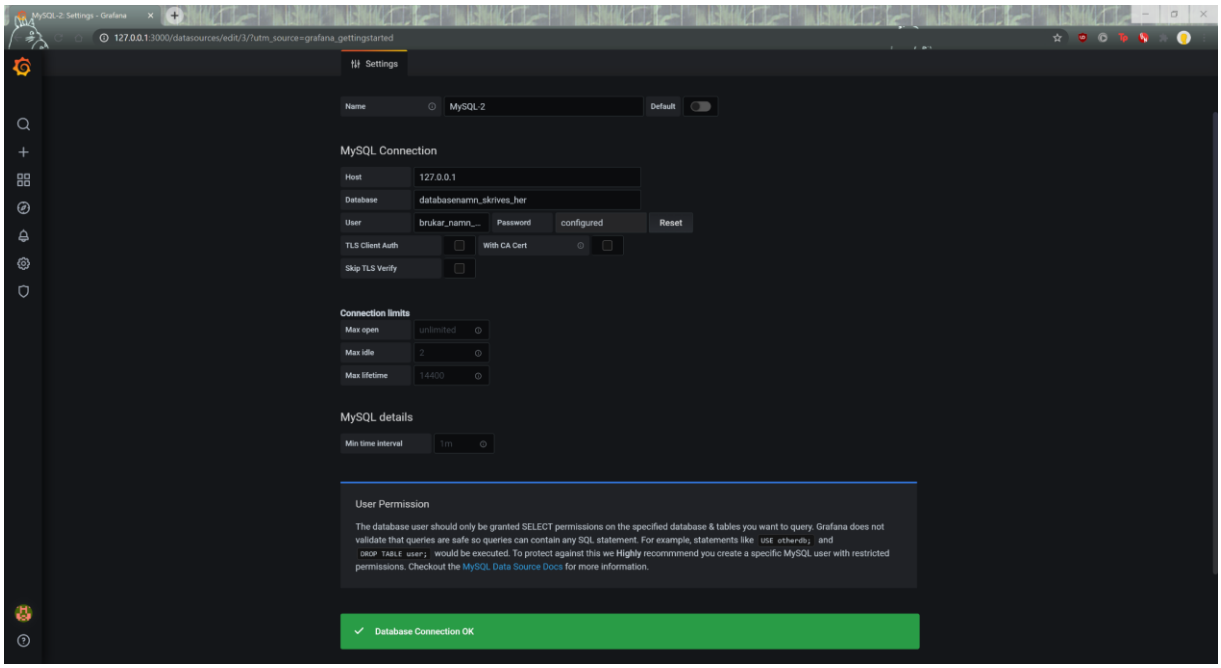
5.3.2 Oppsett av Grafana

I dette kapitelet blir det beskrive kva som skal til for å opprette dashbord som er brukt i dette prosjektet.

Datakjelde

For å kunne hente data inn til Grafana, må det opprettast datakjelde. I dette prosjektet er det InfluxDB og MariaDB som er brukte datakjelda. Ved å velje i «Add your first data source» i valmenyen til venstre kommer det opp eit vindauge. I det vindauget viser Grafana mange ulike val av database programmer. I dette tilfellet er det MySQL og InfluxDB som skal bli brukt for oppgåva.

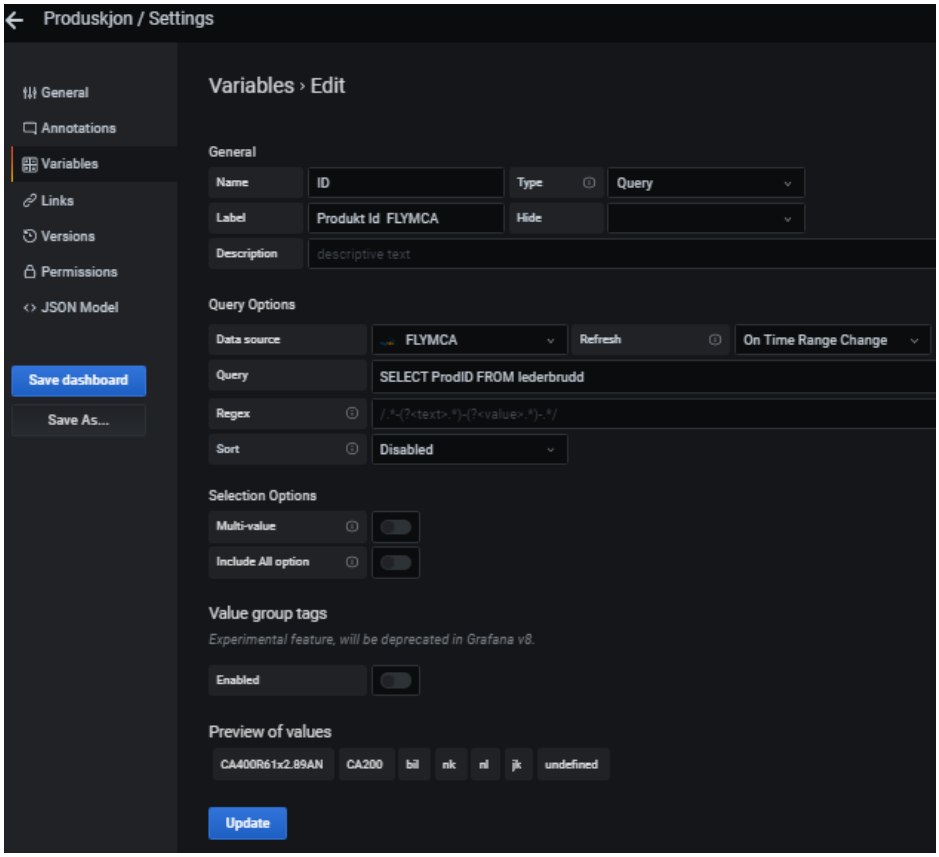
Når databaseprogram er valt kommer vindauge for oppsett, Figur 39. Her trenger det barre å førre in IP-adressa til maskinen databasen køyre på, namnet på databasen, brukar og passord. Køyre databasen og Grafana på same maskin trengst ikkje oppgje IP-adresse. Er det kontakt så skal det stå ein grøn tekst nedst kor det står «Database connection OK», som vist i Figur 39. Då er datakjelde sette opp korrekt.



Figur 39 Kommunikasjon av PDB og Grafana

Grafana variablar

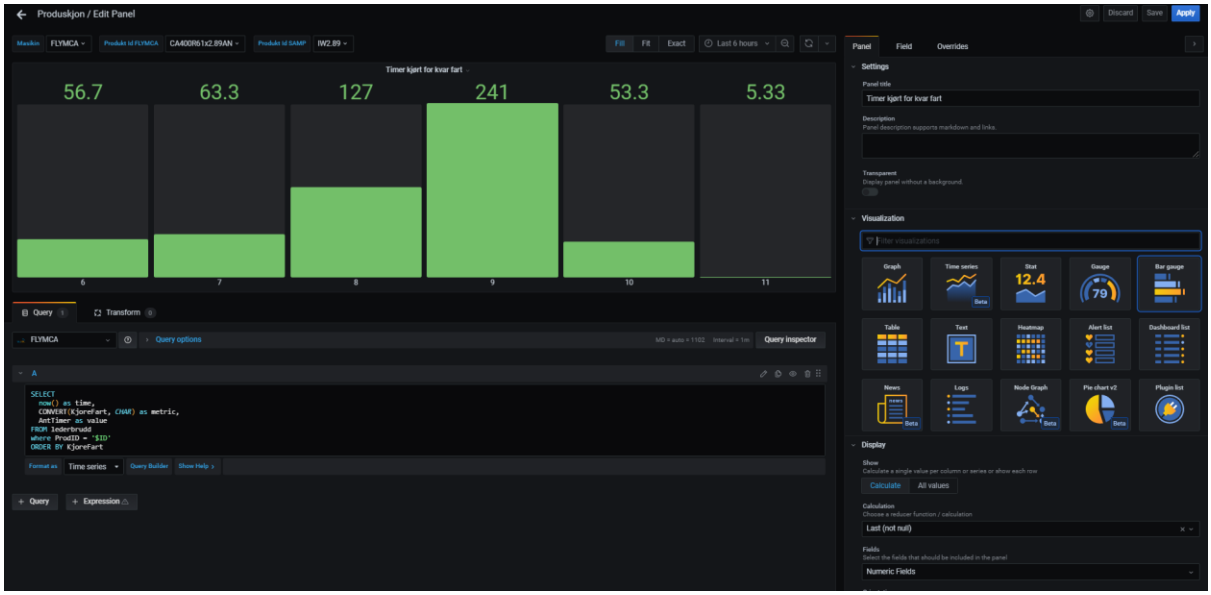
I dashbord innstillingar kan det leggast til nye variablar som vist i Figur 40. Variablane vises som ein valmeny øvst i dashbordet, sjå Figur 36. Her kan det leggast inn val av datakjelde, database osv. I dette prosjektet er det laget eit valmeny av produkt ID til dei ulike maskinen. I feltet «Query» blir SQL kode nytta for å hente ut dei ynskja variablane frå tabellen. Grafana variabelen blir nytta ved å skrive SQL kode «WHERE ynskja variabel i tabell = `$grafana variabel namn`» sjå Figur 42.



Figur 40 Variabeloppsett

Panel og Dashbord

Med oppretta kommunikasjon så må det tillages eit «Dashboard». I dashbordet plasserast det ut ein eller fleire «panels». I desse panela kan det tillagast kva type visualisering som er ynskja, og legger inn dei data fått frå datakjeldane. Vidare med «add new panel» og etterfølgt med «edit panel» vil eit vindauge som vist i Figur 41 dukka opp.



Figur 41 Grafana panel vindauge

Her blir det lagt til datakjeldane som er oppretta, slik at data frå PDB kan innførast til panelet. Data blir valt ut ved hjelp av SQL-kodar. På høgreside i vindauge blir valet med kva type visualisering som er ynskja. Dette valet er avhengig av kva type informasjon det er, og kva databaseprogram det kjem i frå. Det kan også leggjast til parameter og funksjonar som f.eks. trekker ut informasjon eller varslar ved alarm på gitte verdiar. Grafana kan også senda mail som varsling.

Data frå InfluxDB er naturleg lett å framstilla i Grafana, ettersom det er tidsseriebasert. Visualiseringstypane som er eigna for typen informasjon InfluxDB gir, er for eksempel tidsseriegraf som speedometer, trykkmålar, temperatur o.l. Data som kan brukast til desse visualiseringane kan vera nåverande køyrefart, den produserte lengda, spenningsforbruk og trekkekrafta på trådane.

Data frå MariaDB er i dette prosjektet ikkje tidsseriedata. Derfor er det gjort nokre triks for å lure Grafana til å tru at det er tidsseriedata. Dette trikse er å bruke «now() as time» i SQL koden, sjå Figur 42. Da kan søylediagram lagast utan problem.

Figur 42 Eksempel av SQL kode til eit panel

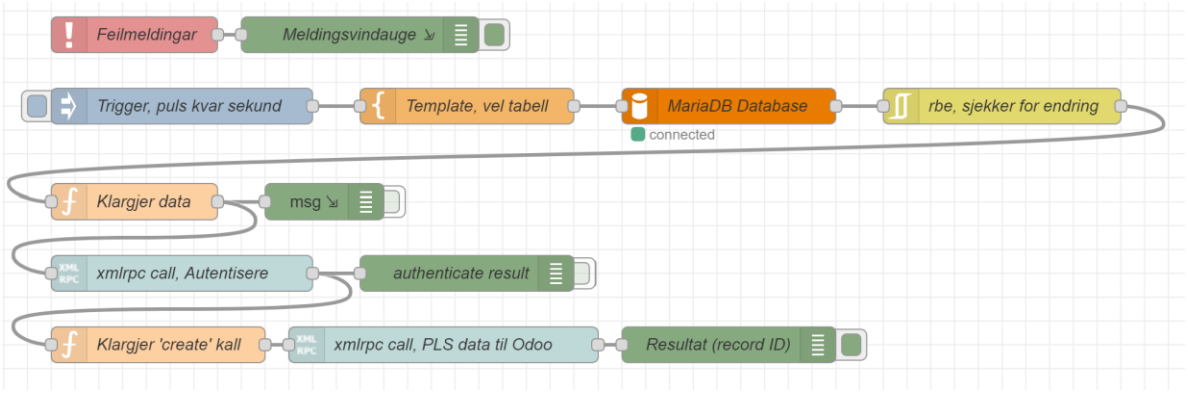
5.4 PDB til ERP/MES

Det siste leddet igjen av oppgåva er å etablere kommunikasjon mellom PDB til forretningsverktøyet til bedrifta, som er Odoo Enterprise. Det er berre laget for FLYMCA og henting av produksjonsdata, hovudsakleg av tidsmessige grunnar.

Innstillingar og oppretting av database i Odoo er gjort av ein tilsett frå Informasjonskontroll AS. Så kapittelet her viser berre korleis kommunikasjon blir oppretta til eit eksisterande Odoo database.

5.4.1 Kommunikasjon til Odoo

For å opprette kontakt med Odoo kan Node-RED bli brukt. For å oppnå kontakt til Odoo så krevst det ein API (Application Programming Interface) for å autentisere data frå bedrifts PC-en. API er eit grensesnittsverktøy for utveksling av data mellom ulike applikasjonar. Det skal gjera det enkelt å kommunisera over system når det er ulike unike kodespråk, for å sleppa å måtte setta seg ned i dei forskjellige språka. I denne oppgåva blir «node-red-contrib-xmlrpc» paletten brukt som API i Node-RED.



Figur 43 Odoo Node-RED

Figur 43 viser det endelege oppsettet av Node-RED programmet. Dette programmet skal berre senda data til Odoo når det skjer endringar i PDB, og trekker ut siste rekka lagt i tabellen «Prodtabell» i MariaDB. Vidare må det gjerast klart for å autentisering og oppkall til Odoo sitt nettverk for å senda data til databasesystemet der. Skriptane til template og funksjonsnodane ligger som vedlegg. I Tabell 2 er nodane brukt forklart.

5.4.2 Oppsett av Node-RED til Odoo

Øvste linja i flyten med feilmelding noden Figur 43, er berre ein måte å fange opp alle feilmeldingar som oppstår i det workspace-vindauget. Det gir då beskjed til debug vindauget.

Vidare i den andre linja i flyten er det ein «inject» node kalla Trigger, som er innstilt til å senda ein melding kvart sekund kontinuerleg. Denne meldinga fungerer som ein puls/trigger, som vil få «template» noden til å senda ein konstant kommando til databasen.

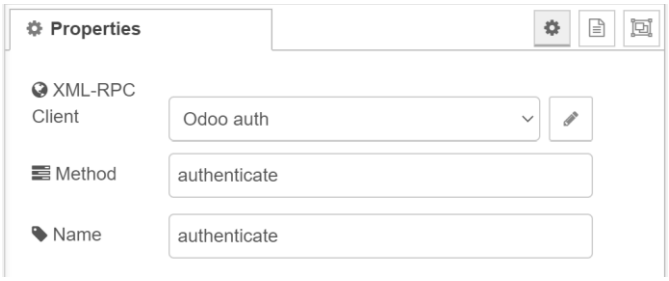
«template», har SELECT – FROM kommando i seg som seie til PDB å velja ut kolonnar frå ein tabell. Namna må tilsvara namna som står i tabell.

I tillegg for å få tidsstempel til å fungera må det konfigurerast for å få godkjend stempel ut frå MariaDB. Som nemnd tidlegare i rapporten så vil «rbe» blokkere meldingar som det ikkje har skjedd endringar i. I dei tre siste linjene er det to «funksjons»- og to «xmlrpc» noder.

- «Klargjer data», gir ein brukaridentifikator (UID) for autentisering til Odoo databasen, og behandlar data frå MariaDB til Odoo sitt kallfunksjon i neste funksjonsnode. Denne plukkar ut berre den siste raden i tabell:

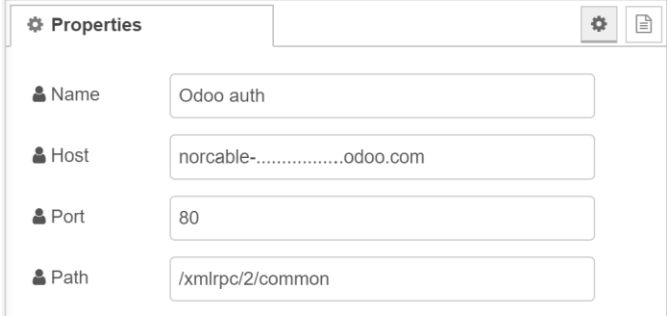
$$Siste\ rad = tabell[tabell.length - 1]$$

- «xmlrpc call» autentisere, sendar eit metodekall til Odoo «authenticate» for å generere API frå «endpoint» /xmlrxp/2/common. «API endpoint» er eit inngangspunkt for å kommunisera med Odoo. Det må innstillast til nettsida til Odoo der data skal til, port nummer og endepunkt.



Figur 44 xmlrpc node (1)

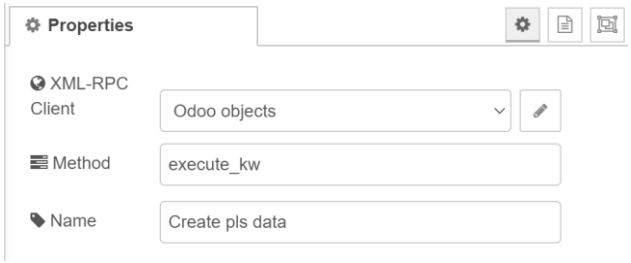
På «Method» må det skrivast ned kommandoen/metodekallet denne noden skal gjera når den kontaktar Odoo. For å kome til vindauget for å skriva inn IP-adresse så trykker man på «penn» symbolet som kan ses på figuren over.



Figur 45 xmlrpc node (2)

IP-adressa og port nummer må bli sette opp i Odoo, men dette blir oppretta av ein tilsett frå Informasjonskontroll AS som jobbar med dette systemet. Med informasjonen så føres IP-adresse, port nummer og endepunktet det skal henta UID i frå.

- «Klargjer 'create' kall», denne får inn eit UID generert frå «authenticate» og data frå funksjonsnoden «Klargjer data». Funksjonen fusjonerer UID og data til ein samla pakke. I tillegg ligger det eit kall som seie kor den skal, og kva pakken skal gjera når den er sendt inn i Odoo.
- «xmlrpc» PLS data til Odoo, koplar seg til Odoo med metodekallet «execute_kw» til endepunktet /xmlrpc/2/object. Det execute_kw gjer er at den tek i mot pakken frå «Klargjer 'create' kall», sjekkar innhaldet, godkjenner det og sender til rett database i Odoo. Kvart kall til «execute_kw» tar følgjande parametrar:
  - Databasen som skal brukast
  - Brukarens ID henta frå autentisering
  - Brukarens passord
  - Modellnamn
  - Metodenamn/action
  - Ein liste over parameterar sendt
  - Ein kartlegging av parameterane for å passera etter nøkkelord (Valfritt)



Figur 46 xmlrpc node (3)

Figur 47 xmlrpc node (4)

6 Testing

Til testing av oppgåva så vart løysningsmetodane frå kapittel 5 implementert og testa på testrigg(PLS, SBC og privat PC) og hjå Norcable.

Sida desse data frå FLYMCA og SAMP kan berre hentast frå maskinane i bedrifta, så måtte det lages til eit testrigg for å kunne testa utanom å vera på bedriftsplass. Frå HVL blei det lånt ut ein S7-1500 PLS. Med bruk av nokre brytarar blei det programmert nokre enkle telleprogram, som skal etterlikna produksjon- og setuptid. Og med eit potensiometeret blei det lagd eit program som gjer ut analog verdi som kan etterlikna fart eller produsert lengde. Verdiane som blir danna blir lagra i datablokk i PLS nett som på dei andre PLS programmane i FLYMCA og SAMP.

Programkodane i Node-RED er testa manuelt med bruk av noden «inject». Denne noden etterliknar datastraumen som ein PLS sender. Mens prosedyrane i MariaDB er testa ved å køyre dei.

6.1 Oppståtte problemstillingar

Gjennom jobbinga med dette prosjektet har det oppstått mange problem som har både vore store og små. I dette delkapittelet vil dei store problema bli nemnd.

I byrjinga av oppgåva blei det tatt i bruk eit virtualiseringsverktøy for å simulera ein maskin med ein anna OS. På denne måten kan den verkelege maskinen med Windows køyra parallelt samtidig med den virtuelle maskinen med Ubuntu. Slik kan bedrifts PC bli simulert med eigen PC. Men under bruk blei det problem med å kommunisera mellom andre maskinvarer med den virtuelle maskinen. Virtualiseringsverktøyet blei droppa tidleg og er derfor ikkje nemnd i rapporten. Men verktøyet har vore i bruk til å lage figurar og bilete som eksemplar.

I bedrifta så er det ikkje så mykje aktivitet i desse tidene, så det har ikkje vore lett å skaffe seg nok med resultat frå systemet. Sida me i gruppa ikkje har fått testa ut systemet i bedrifta ofte, så kan det oppstå uventa feil. Dette kan kome frå skrivefeil i program, eller det kan vera det data som kjem frå DB ikkje leverer som forventa.

Under testing oppstod det kommunikasjonsproblem frå andre maskinvarer med å kopla seg til databasen. Dette kom av at standard innstillingane på MariaDB er berre sette for lokalnett. Dette blei oppdaga med Ubuntu og ikkje på Windows. Løysninga for dette var å finna ein konfigurasjonsfil (.cnf) og endra på innhaldet. Dette er kort nemnd i kapittel 5.2.2.

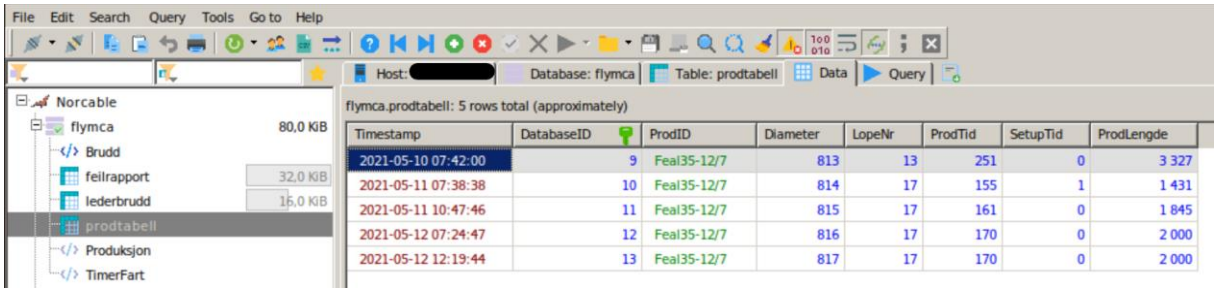
Det blei også problem med å få resultat i database ettersom det blei skreve feil IP-adresse i HeidiSQL programmet. I tillegg vart det kjøpt ny PC i bedrifta og ny ruter, noko som også endra på IP-adressa til datamaskinen. For å hindre slike problem oppstår igjen måtte det ordnast med ein statisk IP-adresse på bedrifts PC, som er ein adresse som vil vera fast for denne datamaskinen. I vedlagt fil Manual Norcable viser det korleis oppsettet og nedlasting blir gjort for datamaskinen.

6.2 Test resultat

Det var utført testar med testrigg for å sjekka om programma i Node-RED og prosedyrane i MariaDB fungerte som ynskja. Når programma fungerte feilfritt i testrigg blei systemet implementert hjå Norcable, for å testa datainnsamlingssystemet i ein reell situasjon.

I dei første testane vart det testa for kommunikasjon mellom maskinvarene, frå PLS til SBC og til bedrifts PC. I seinare testar blei det testa funksjonalitet i oppgradert Node-RED program, og kommunikasjon frå MariaDB til Odoo.

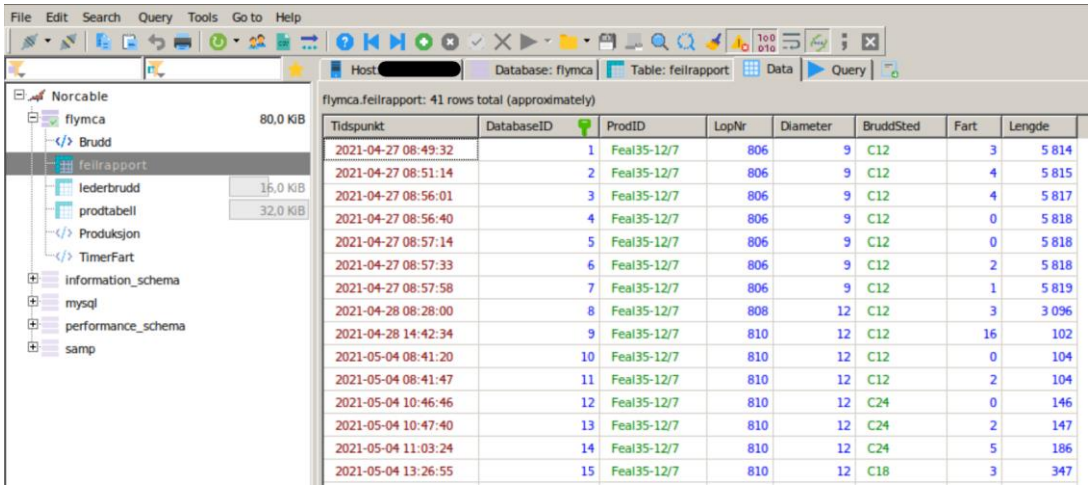
I den siste testen gjort av gruppa kom det lovande resultat. I hovudtrekk så viste resultata at systemet skal fungera, men det var fleire små feil som blei oppdaga.



Timestamp	DatabaseID	ProdID	Diameter	LopeNr	ProdTid	SetupTid	ProdLengde
2021-05-10 07:42:00	9	Feal35-12/7	813	13	251	0	3 327
2021-05-11 07:38:38	10	Feal35-12/7	814	17	155	1	1 431
2021-05-11 10:47:46	11	Feal35-12/7	815	17	161	0	1 845
2021-05-12 07:24:47	12	Feal35-12/7	816	17	170	0	2 000
2021-05-12 12:19:44	13	Feal35-12/7	817	17	170	0	2 000

Figur 48 Testresultat produksjonstabell

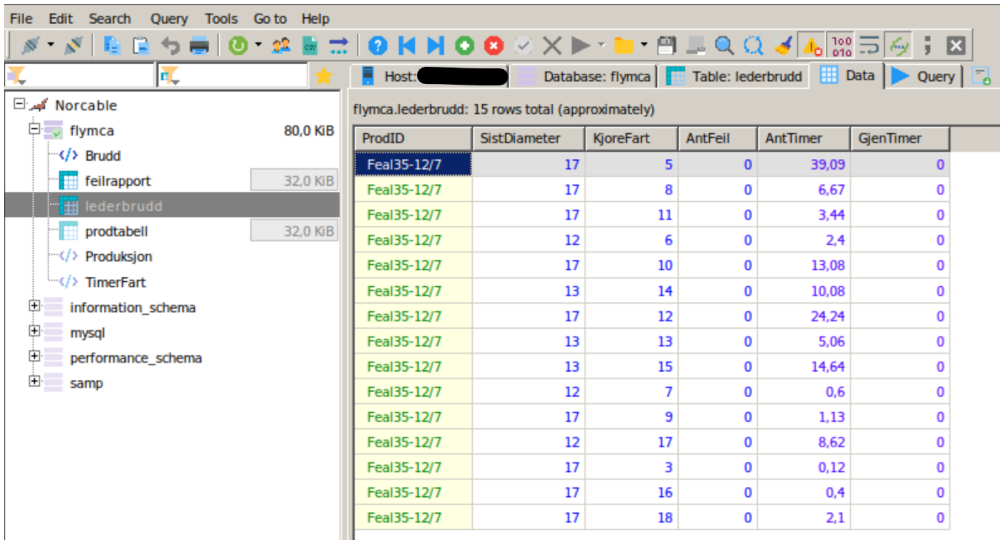
I Figur 48 viser det produksjonsdata som har blitt samla i prodtabell. Ein liten feil som blei oppdaga var at LopeNr-data hadde kome i Diameterkolonnen og Diameterdata til LopeNr-kolonnen. Dette kom frå ein skrivefeil i prosedyren «Produksjon» som er blitt retta på. Det er også lagt merke til at produkt ID er uendra, sjølv om det er ulik diameter på den første linja i forhold til dei andre. Dette kan kome frå at PLS variabel ikkje kjem ut som tiltenkt eller at operatør ikkje har endra oppskrift skikkeleg ettersom diameteren har endra seg mens produkt ID har vore uendra.



Tidspunkt	DatabaseID	ProdID	LopNr	Diameter	BruddSted	Fart	Lengde
2021-04-27 08:49:32	1	Feal35-12/7	806	9	C12	3	5 814
2021-04-27 08:51:14	2	Feal35-12/7	806	9	C12	4	5 815
2021-04-27 08:56:01	3	Feal35-12/7	806	9	C12	4	5 817
2021-04-27 08:56:40	4	Feal35-12/7	806	9	C12	0	5 818
2021-04-27 08:57:14	5	Feal35-12/7	806	9	C12	0	5 818
2021-04-27 08:57:33	6	Feal35-12/7	806	9	C12	2	5 818
2021-04-27 08:57:58	7	Feal35-12/7	806	9	C12	1	5 819
2021-04-28 08:28:00	8	Feal35-12/7	808	12	C12	3	3 096
2021-04-28 14:42:34	9	Feal35-12/7	810	12	C12	16	102
2021-05-04 08:41:20	10	Feal35-12/7	810	12	C12	0	104
2021-05-04 08:41:47	11	Feal35-12/7	810	12	C12	2	104
2021-05-04 10:46:46	12	Feal35-12/7	810	12	C24	0	146
2021-05-04 10:47:40	13	Feal35-12/7	810	12	C24	2	147
2021-05-04 11:03:24	14	Feal35-12/7	810	12	C24	5	186
2021-05-04 13:26:55	15	Feal35-12/7	810	12	C18	3	347

Figur 49 Testresultat feilrapport

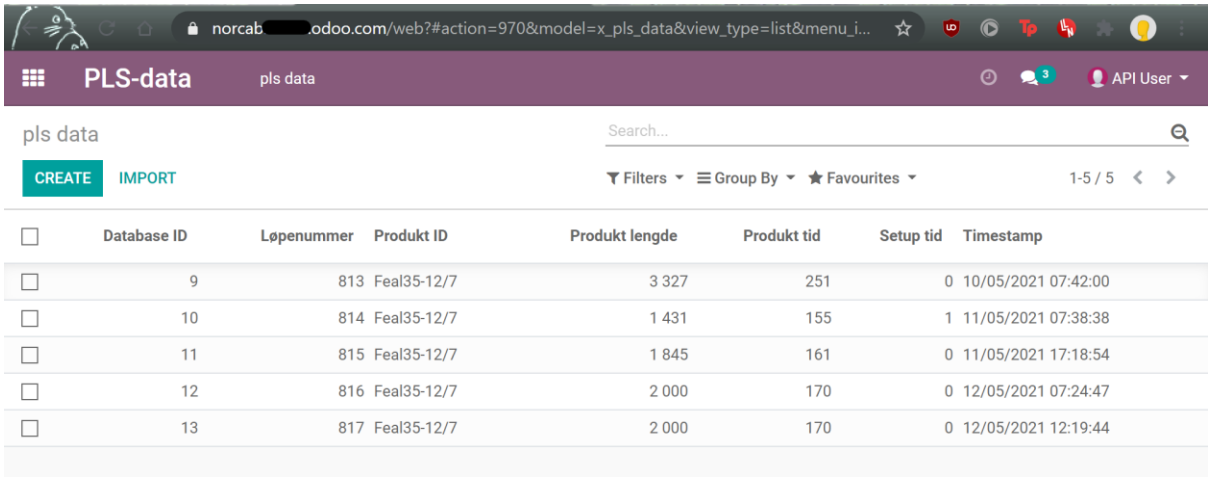
I feilrapport kom det veldig mykje data i forhold til kva som var forventa. Som nemnd i delkapittel 5.1.4.2 så skal tabellen henta produksjonsdata når det oppstår brot av linje under produksjon. I «Tidspunkt» blir det observert at det er korte mellomrom mellom kvar tid, og at i nokre plasser står det at farten er null. Det blir mistenkt at datablokken «Wire Break» registrerer ikkje berre brot, men mogleg brems og andre alarmar. Med undersøking i PLS-programmet er det mogleg funnet rett DB, som skal berre gjelde brot alarm.



ProdID	SistDiameter	KjoreFart	AntFeil	AntTimer	GjenTimer
Feal35-12/7	17	5	0	39,09	0
Feal35-12/7	17	8	0	6,67	0
Feal35-12/7	17	11	0	3,44	0
Feal35-12/7	12	6	0	2,4	0
Feal35-12/7	17	10	0	13,08	0
Feal35-12/7	13	14	0	10,08	0
Feal35-12/7	17	12	0	24,24	0
Feal35-12/7	13	13	0	5,06	0
Feal35-12/7	13	15	0	14,64	0
Feal35-12/7	12	7	0	0,6	0
Feal35-12/7	17	9	0	1,13	0
Feal35-12/7	12	17	0	8,62	0
Feal35-12/7	17	3	0	0,12	0
Feal35-12/7	17	16	0	0,4	0
Feal35-12/7	17	18	0	2,1	0

Figur 50 Testresultat lederbrudd

I lederbrudd viser det null i «AntFeil» som medføre at «GjenTimer» også får null. I feilrapport viste det fleire brot meldingar, noko som skulle medføra at AntFeil skal telle seg opp. Det er mistenkt at dette kjem frå Node-RED programmet. Frå tidlegare versjon var det to separate funksjonsnoder kor den eine noden sendte data direkte til tabell, og den andre til prosedyre. Her var det mistenkt at når begge funksjonane køyrar samtidig blir det konflikt med data sendt til MariaDB. Desse funksjonsnodane er no samla til eit, sjå Figur 18.



Figur 51 Odoo database

Figur 51 viser forretningsverktøyet Odoo har fått produksjonsdata frå FLYMCA i sitt databasesystem. Dette har blitt testa med eige datamaskin, i ein test database i Odoo og har enda ikkje blitt testa i ein reell produksjon. Dette kjem av at bedrifta er driver med oppgradering av Odoo og har dermed ikkje laga eit endeleg system som kan brukast skikkeleg.

6.3 Konklusjon av test

Under testinga av datainnsamlingssystemet i Norcable var det fleire faktorar som var med på påverka resultata.

Det har vore fleire gongar der data frå PLS har ikkje kome som forventa. Sånn som i Figur 49 blei det oppdaga at «Wire Break» er meir enn berre eit brot sensor. Med fleire testingar kunne dette blitt undersøkt og fiksa på.

Men resultata har vist at systemet som er laget i dette prosjektet skal fungera til sitt formål. Med resultata frå Figur 48 så viser at det er oppnådd levering av data frå PLS til PDB. I frå tidlegare test i samarbeid med ein tilsett frå Informasjonskontroll AS blei det laget eit Node-RED system som sender data frå PDB til Odoo, som vist i kapittel 5.4. Tilleggsfunksjon for brotanalyse trenger meir køyretid for å bekrefte om rett DB er satt.

7 Svakheiter

Med sine positive sider med oppgåva så finst det svakheiter. Her vil det bli tatt opp svakheiter av maskinvarer, programvarer og systemet i seg sjølv.

7.1 Maskinvare

Lagringsmediet til SBC skjer på eit micro SD-kort som vil vera ein svakheit til produktet. Bachelor gruppa har fått tips frå Westcon og sjølv erfart frå tidlegare at desse micro SD-kort kan bli korrupt etter fleire overskrivningar. For å minimera risiko kan original kortet byttas til eit micro SD-kort med industriell grad, eller som er laget for videoovervaking som har høgare robustheit. Det kan med fordel lagast ein sikkerheitskopiering av micro SD-kortet, til eit eventuell svikt. Det vil gjer da enklare å setta opp SBC og setta system i drift igjen. Ved straumbrot så vil SBC naturlegvis skru seg av. Men så lengje Node-RED er lasta ned og innstilt at den skal starte ved boot, så vil SBC vera funksjonell ved oppstart utan å måtte logge seg inn.

Bedrifts PC er bindeleddet frå PDB til ERP/MES. Viss denne datamaskina mister straumen eller mister tilkopling nettverket så vil ikkje systemet vera operativt, som vil sei at systemet ikkje vil samla opp data.

7.2 Programvare

I oppgåva så har det ikkje vore fokus på sikkerheita til systemet. Det er fleire moglege smetthol som bør takast omsyn til. Her er nokre eksemplar på sikkerheitshol:

- Med tilgang til lokale nettverket vil vedkommande kunna henta/endra/sletta informasjon frå nettet
- IP-adresser og port nummera som er brukt er ikkje kryptert, men datatrafikk kan krypterast. Det viktigaste grepet vil her vera å ikkje eksponera porter for internett som kan vera letta mål for hacking.
- InfluxDB er det ikkje oppretta brukar, noko som betyr at det er ingen sikkerheit med å få tilgang til databasens innhald.

I PLS typar som er ikkje er nyare enn S7-1500 PLS så må PUT/GET kommunikasjons innstilling aktiverast. Dette er ein kommunikasjonsmetode for PLS-en som vil opna for PUT med å sende inn og endra data, og GET med å ta i mot data. Dette kan vera eit sikkerheitshol med at andre uvedkommande kan skriva over i datablokkene. I oppgåva er det kunn mottak av data, men med PUT/GET aktivert så vil det framleis gje moglegheit for hacking.

Her kan det oppdaterast til OPC UA(Open Platform Communications Unified Architecture) som er eit kommunikasjonssystem som Siemens har tilgjengeleg for PLS [11]. Node-RED har også moglegheit til å implementere OPC UA, med å ta i bruk paletten «node-red-contrib-opcua». Det denne kommunikasjonsprotokollen gjer er å bruka autentisering, signering av data og kryptering av trafikk for å sikra dataflyta. [12]

Alle filar og data frå systemet bør sikkerheitskopierast slik at systemet ikkje går tapt i tilfelle svikt i systemet. På denne måten kan programmer og skript frå Node-RED og MariaDB raskt bli implementert igjen, utan å måtte gå gjennom alle stega med nedlasting og oppsett. Produksjonsdata som har kome frå PLS blir lagra opp i Odoo sitt nettverk, så på den måten vil det fungera som sikkerheitskopiering av data.

Nokre av dei opne kjeldekoda programvara har ikkje brukarstøtte kontakt. Ved eventuelle feil kan forum plattformar brukast for å få hjelp til problem, og det der ingen som er forplikta til å hjelpa. Sikkerheita i desse programvara kan svekkast når fleire utviklarar har tilgang til programmet. Nokon av desse personane kan ha dårlege hensikter og legga inn smetthol i programvara for å nå ut til brukardata. [13]

7.3 System

Systemet er avhengig av produkttypen i PLS er korrekt. Derfor er det viktig at operatøren endre til rett produkttype som skal køyre gjennom maskinen. Blir produkt ID ikkje endra vil feil ID bli registrert i systemet. I produksjonstabellen er det lagt til ein ekstra kolonne om kabeldiameteren, denne kolonnen er for å kontrollera at produkt ID er korrekt. Er det noko skrivefeil i produkt ID så vil systemet setta opp ny rad. Denne raden vil bli rekna med som ein unik ID som ikkje er ein produkttype i systemet. Dette vil føra til at informasjonen frå denne produksjonen kan ikkje samanliknast med dei andre produksjonsdata.

8 Konklusjon

Det har ikkje vore andre tilsvarande system å basere oppgåva på, og arbeidet gjort her er frå botn av. Det har vore mykje stoff å sette seg inn i for å bli kjend med, og forstå korleis alle programvarene fungere. Med mykje prøving og feiling har det resultert med eit velfungerande system.

Hovudmålet med oppgåva var å lage eit datainnsamlingssystem som aukar leveringspresisjonen til Norcable, og nå MVP kravet som blei lagt frem av bedrifta. Den korte tida systemet har samla data i bedrifta har gitt lovande resultat, men vil visa seg over tid om det vil fungera optimalt. Resultata har vist at data har blitt tatt frå PLS og har blitt overført til database. Det var også testa overføring av data til forretningsverktøyet noko som har vist seg å fungere, og dermed er MVP oppnådd. Men under arbeidet har bedrifta jobba med å oppgradera forretningsverktøyet, slik at det ikkje er testa i det nye systemet.

Det har vore oppdaga at brotanalyseverktøyet ikkje har fungert korrekt ettersom at PLS variabel for brot ikkje har fungert rett. Med retting av denne feilen håper me at bedrifta kan nytta seg av denne funksjonen, for å auke produktivitet til produksjon. Produkt ID var uendra i database sjølv med ulike diameter. Dette mistankes at enten PLS variabel ikkje fungerer som tiltenkt, eller at operatør ikkje har ført inn produkt ID.

Denne oppgåva har vist at det er mogleg å laga eit fullt fungerande lågkostnads datainnsamlingssystem. Andre som ynskjer eit liknande system kan godt nytte seg av løysninga som er tatt frem i denne rapporten.

Systemet laget i denne oppgåva vil ha mange moglegheiter til vidareutvikling og forbetring i framtida.

9 Bibliografi

[1] M. Hermann, T. Pentek og B. Otto, «Design Principles for Industrie 4.0 Scenarios,» IEEE, 2016.

[2] K. Sander, «Fire industrielle revolusjoner,» [Internett]. Available: https://estudie.no/fire-industrielle-revolusjoner/.

[3] M. Tengesdal, «Frå transistor til datamaskin,» Universitet i Stavanger, 2018.

[4] Raspberry Pi Foundation, «Raspberry Pi Projects,» [Internett]. Available: https://projects.raspberrypi.org/en/projects.

[5] K. Bratbergsengen, «database,» 11 november 2019. [Internett]. Available: https://snl.no/database.

[6] InfluxData, «What is time series data?,» [Internett]. Available: https://www.influxdata.com/what-is-time-series-data/.

[7] Node JS Foundation, «Node-RED Library,» Open JS Foundation, [Internett]. Available: https://nodered.org.

[8] MariaDB, «MariaDB versus MySQL - Features,» [Internett]. Available: https://mariadb.com/kb/en/mariadb-vs-mysql-features/.

[9] Raspberry Pi Foundation, «Raspberry Pi OS,» [Internett]. Available: https://www.raspberrypi.org/documentation/raspbian/.

[10] Siemens, «What types of access are available in STEP 7 (TIA Portal),» [Internett]. Available: https://support.industry.siemens.com/cs/document/67655611/what-types-of-access-are-available-in-step-7-(tia-portal)-to-access-data-values-in-blocks-and-what-should-you-watch-out-for-with-the-differences-between-the-types-?dti=0&lc=en-WW.

[11] Siemens , «OPC UA – Open Platform Communications Unified Architecture,» [Internett]. Available:

https://new.siemens.com/global/en/products/automation/industrial-communication/opc-ua.html.

[12] Node JS Foundation, «node-red-contrib-opcua,» [Internett]. Available: https://flows.nodered.org/node/node-red-contrib-opcua.

[13] Investintech.com Inc., «Pros & Cons of Open Source in Business,» [Internett]. Available: https://www.investintech.com/resources/blog/archives/7975-pros-cons-open-source-business.html?fbclid=IwAR22U43SXa7YzaETw89NlU2qNp-jGHaqPR0SElV1gJ0tMuI4uLzG-1Ospo4.

[14] W. Bolton, Programmable Logic Controllers, 5 red., Burlington, MA: Newsnes, 2009.

[15] «Stack Overflow,» Stack Overflow Ltd., [Internett]. Available: https://stackoverflow.com/.

[16] «Documentation Ubuntu,» Canonical LTD., [Internett]. Available: https://help.ubuntu.com/.

[17] B. E. Andrew T. Wilson, Open Source Achaeology, Ethics and Practice, De Gruyter Open Poland, 2015, p. 159.

[18] Siemens, «Siemens Industry Online Support,» [Internett]. Available: https://support.industry.siemens.com/cs/start?lc=en-NO.

[19] M. Drake, «Tutorial: How To Allow Remote Access to MySQL,» DigitalOcean, 7 3 2019. [Internett]. Available: https://www.digitalocean.com/community/tutorials/how-to-allow-remote-access-to-mysql.

[20] Odoo, «XML-RPC Web services,» Odoo, [Internett]. Available: https://doc.odoo.com/v6.0/developer/6_22_XML-RPC_web_services/index.html.

[21] Bedriften BI, «Del 2 - Bedriften og de tre industrielle revolusjonene,» Bedriften BI, [Internett]. Available: http://www.bedriftenbi.no/de-tre-industrielle-revolusjonene/.

[22] InfluxData, «InfluxDB 1.8 Documentation,» InfluxData, [Internett]. Available: https://docs.influxdata.com/influxdb/v1.8/.

[23] Siemens, «Software in the TIA Portal,» Siemens, [Internett]. Available: https://new.siemens.com/us/en/products/automation/industry-software/automation-software/tia-portal/software.html.

[24] Odoo, «Compare Odoo Editions,» [Internett]. Available: https://www.odoo.com/page/editions.

[25] W3Schools, «Tutorial,» [Internett]. Available: https://www.w3schools.com/sql/default.asp.

[26] I. M. Liseter, «URL,» [Internett]. Available: https://snl.no/URL.

[27] Odoo, «External API,» Odoo, [Internett]. Available: https://www.odoo.com/documentation/14.0/developer/webservices/odoo.html.

[28] RapidAPI, «Endpoint - What is an API Endpoint?,» [Internett]. Available: https://rapidapi.com/blog/api-glossary/endpoint/.

[29] MariaDB, «MariaDB Knowledge Base,» MariaDB Foundation, [Internett]. Available: https://www.mariadb.com/kb/en/.

[30] N. Ayllon, «What is PROFINET? - PROFINET explained,» [Internett]. Available: https://us.profinet.com/profinet-explained/.

[31] K. Bratbergengen, «relasjonsdatabase,» 30 april 2019. [Internett]. Available: https://snl.no/relasjonsdatabase.

[32] K. Bratbergsengen, «SQL,» 26 september 2017. [Internett]. Available: https://snl.no/SQL.

10 Vedlegg

- Vedlegg 1 Programkodar i MariaDB, Node-RED og Grafana
- Vedlegg 2 .json filar fra Node-RED og Grafana
- Vedlegg 3 .sql fil fra MariaDB
- Vedlegg 4 Manual Norcable (førebels)

  Manual Norcable er produktet til bacheloroppgåva. Det blir lagt som eit førebels dokument ettersom det er uferdig. Leveringsfrist for produktet er 28.05.21

Vedlegg 1 – Programkodar i MariaDB, Node-RED og Grafana

MariaDB Prosedyre

FLYMCA

Brudd

```
BEGIN
DECLARE Timer double DEFAULT 0.0;
DECLARE `Brudd` double DEFAULT 0.0;
DECLARE TimerKjo double DEFAULT 0.0;
DECLARE sjekk int DEFAULT 0;

/*sjekker om det er ein linje frå før av på same ProduktID
og Fart*/
select Count(ProdID)
   into sjekk
   FROM lederbrudd
   WHERE KjoreFart = Fart and ProdID = Prod_ID;

/*Henter data frå tabell og legger på ein feil til*/
if sjekk > 0 THEN
   SELECT AntBrudd
        INTO `Brudd`
        FROM lederbrudd
          WHERE KjoreFart = Fart and ProdID = Prod_ID;

/*Henter antall timer frå tabell*/
   SELECT AntTimer
        INTO Timer
        FROM lederbrudd
          WHERE KjoreFart = Fart and ProdID = Prod_ID;

/*utrekning*/
   SET `Brudd` = `Brudd` + 1.0;
   SET Timer = ROUND(Timer+(KjoreMin/60.0),2);
   SET TimerKjo = Round(Timer/`Brudd`,2);

/*Opptatere tabellen*/
   Update lederbrudd
        SET  Timerbrudd = TimerKjo, AntTimer = Timer,
AntBrudd =`Brudd`
        WHERE ProdID = Prod_ID and Kjorefart = Fart;
/*legger inn ny rad, når ikkje det eksisterar frå før*/
ELSE
   SET Timer = ROUND((KjoreMin/60.0),2);
```

```
    SET TimerKjo = Round(Timer/1.0,2);
    INSERT INTO lederbrudd(ProdID,Kjorefart, AntBrudd,
AntTimer, Timerbrudd)
    VALUES(Prod_ID,Fart,1, Timer, TimerKjo);
END if;
END
```

MariaDB Prosedyre

FLYMCA

Produksjon

```
BEGIN
DECLARE Ant int DEFAULT 0;
DECLARE SlettID int DEFAULT 0;
DECLARE Kvadrat INT DEFAULT 0;
DECLARE GjennomsnittlegFart DOUBLE DEFAULT 0;

/*Rekner gjennomsnittsfart v = s/t*/
SET GjennomsnittlegFart =ROUND( PLengde/ProdT,2);

/* Legger data inn i tabell som ein ny rad*/
INSERT INTO prodtabell(ProdID,Diameter,LopeNr, ProdTid,
SetupTid, ProdLengde, GjennFart) VALUES (ID,InnDiameter, Nr,
ProdT,SetupT, PLengde,GjennomsnittlegFart);

/*Sletter ein rad, viss det blir over 10 rader i tabell*/
select Count(ProdID)
   into Ant
   FROM prodtabell
   WHERE ProdID = ID;

if Ant > 10 THEN
   SELECT MIN(DatabaseID)
       INTO SlettID
       FROM prodtabell
         WHERE ProdID = ID;

   DELETE
             from prodtabell
             WHERE DatabaseID = SlettID;
END if;
END
```

MariaDB Prosedyre

FLYMCA

TimerFart

```
BEGIN
DECLARE Timer double DEFAULT 0.0;
DECLARE `Brudd` int DEFAULT 0;
DECLARE TimerKjo double DEFAULT 0.0;
DECLARE sjekk int DEFAULT 0;

/* Teljer antall linjer, med lik fart og ProduktID*/
select COUNT(ProdID)
   into sjekk
   FROM lederbrudd
   WHERE KjoreFart = Fart and ProdID = Prod_ID;

/*Sjekker om det er nokre rader frå før*/
if sjekk > 0 THEN
   SELECT AntTimer
        INTO Timer
        FROM lederbrudd
          WHERE KjoreFart = Fart and ProdID = Prod_ID;
   set Timer = Round(Timer + (KjoreMin / 60),2 );
/*Henter antall feil frå tabell*/
   SELECT AntBrudd
        INTO `Brudd`
        FROM lederbrudd
          WHERE KjoreFart = Fart and ProdID = Prod_ID;

/*Sjekker om det er noen feil frå før av*/
   if (`Brudd` > 0) THEN
        SET TimerKjo = Round(Timer/`Brudd`,2);
   ELSE
        SET TimerKjo = 0;
   END if;

/*Opptaterer tabellen*/
   Update lederbrudd
        SET  Timerbrudd = TimerKjo, AntTimer =Timer
        WHERE ProdID = Prod_ID and Kjorefart = Fart;
ELSE
/*Legger til ein ny rad*/
   INSERT INTO lederbrudd(ProdID,Kjorefart,
AntTimer,Timerbrudd, AntBrudd)
   VALUES(Prod_ID,Fart,Round(KjoreMin/60.0,2),0,0);
END if;
END
```

MariaDB Prosedyre

SAMP

Produksjon

```
BEGIN
DECLARE Ant int DEFAULT 0;
DECLARE SlettID int DEFAULT 0;

/* Legger data inn i tabell som ein ny rad*/
INSERT INTO prodtabell(ProdID,ProdTid, Diameter, ProdLengde,
Fart) VALUES (ID,InnProdTid, InnDiameter, InnLengde,
InnFart);

/*Sletter eldste rad, viss over 10*/
select Count(ProdID)
   into Ant
   FROM prodtabell
   WHERE ProdID = ID;

/*Sletter ein rad, viss det blir over 5 rader i tabell*/
if Ant > 5 THEN
   SELECT MIN(DatabaseID)
        INTO SlettID
        FROM prodtabell
          WHERE ProdID = ID;

   DELETE
             from prodtabell
             WHERE DatabaseID = SlettID;
END if;
END
```

Node-RED

FLYMCA

Funksjonsnode, produksjonsdata

```
//Henter tidlegare verdi frå ProdTid og samanliknar med den reelle
produksonstida som pågår
if (global.get('ProdTid') >
msg.payload.ReelProductionCumulativeTime)
{
    var ProdID = global.get('RName');
var LopNR = global.get('TrommelID');
var SetupTid = global.get('SetupTid');
var ProdTid = global.get('ProdTid');
var Diameter = global.get('KabelDiameter');
var ProdLengde = global.get('ProdLengde');

    //IF-løkka starte når det er regisrert at farten er større enn
null
    if(flow.get('ForgjeFart') > 0 )
    {
        var KjoreMin = global.get('ProdTid') -
flow.get('ForgjeKjoreMin');

        //Kjoremin er samanlikninga med produksjonstida og tidlegare
produksjonstid, og IF-løkka her går berre når tida er 0 eller større
        if(KjoreMin >= 0)
        {
            //Sender data til prosedyrane i MariaDB kalla Produksjon
og TimerFart
            msg.topic = "CALL
            `Produksjon`('"+ProdID+"','"+Diameter+"','"+LopNR+"','"+
            ProdTid+"','"+SetupTid+"','"+ProdLengde+"');"+
                "CALL
                `TimerFart`('"+ProdID+"','"+flow.get('ForgjeFart')+"
                ','"+KjoreMin+"');";
        }
        else
        {
            msg.topic = "CALL
`Produksjon`('"+ProdID+"','"+Diameter+"','"+LopNR+"','"+ProdTid+"','
"+SetupTid+"','"+ProdLengde+"');";
        }
    }
    else
    {
```

```
        msg.topic = "CALL
`Produksjon`('"+ProdID+"','"+Diameter+"','"+LopNR+"','"+ProdTid+"','
"+SetupTid+"','"+ProdLengde+"');";
    }
    //Lagrer de siste verdiane frå samla opp her for å samanliknast
med neste msg.payload
    flow.set('ForgjeKjoreMin',0)
    global.set('TrommelID',msg.payload.BobbinIdentification);
    global.set('SetupTid',msg.payload.ReelLoadingCumulativeTime);
    global.set('ProdTid',msg.payload.ReelProductionCumulativeTime);
    global.set('RName',msg.payload.RecipeName);
    global.set('KabelDiameter',msg.payload.CableDiameter);
    global.set('ProdLengde',msg.payload.TotalManufacturingLength);
    msg.payload = null;

}
else
{
    //Lagrer de siste verdiane frå samla opp her for å samanliknast
med neste msg.payload
    global.set('TrommelID',msg.payload.BobbinIdentification);
    global.set('SetupTid',msg.payload.ReelLoadingCumulativeTime);
    global.set('ProdTid',msg.payload.ReelProductionCumulativeTime);
    global.set('RName',msg.payload.RecipeName);
    global.set('KabelDiameter',msg.payload.CableDiameter);
    global.set('ProdLengde',msg.payload.TotalManufacturingLength);
    msg = null;
}
return msg;
```

Node-RED

FLYMCA

Funksjonsnode, brotdata

```
var Brot;

//Køyrer når det oppdages linje brot i Cage12,18,24 eller TakeUp
if (msg.payload.WireBreakC12 === true || msg.payload.WireBreakC18
=== true || msg.payload.WireBreakC24 === true ||
msg.payload.WireBreakTUP === true)
{
    //Leggjer til i Brot variabelen kor brotet har skjedd
    if (msg.payload.WireBreakC12){
            Brot = "C12";
```

```
    }
    else if (msg.payload.WireBreakC18){
            Brot = "C18";
    }
    else if (msg.payload.WireBreakC24){
            Brot = "C24";
    }
    else if (msg.payload.WireBreakTUP){
            Brot = "TakeUp";
    }
    else {
        Brot = "Ukjend";
    }

    //Henter lagra variablar frå global
    var ProdID = global.get('RName');
    var LopNR = global.get('TrommelID');
    var SetupTid = global.get('SetupTid');
    var ProdTid = global.get('ProdTid');
    var Diameter = global.get('KabelDiameter');
    var ProdLengde = global.get('ProdLengde');

     //Samanliknar den pågåande produksjonstida med tidlegare
køyretid
    var KjoreMin = msg.payload.ReelProductionCumulativeTime -
    flow.get('ForgjeKjoreMin');

    //Sikrer at IF-løkka køyre berre når verdien er større enn null
    if (KjoreMin > 0)
    {
    //Sender data om feilen til tabellen feilrapport og prosedyren
Brudd i MariaDB
    msg.topic = +
        "INSERT INTO feilrapport(ProdID, LopNr, Diameter, BruddSted,
Fart, Lengde)" +

"VALUES('"+ProdID+"','"+LopNR+"','"+Diameter+"','"+Brot+"','"+flow.g
et('ForgjeFart')+"','"+ProdLengde+"');" +
        "CALL
Brudd('"+global.get('RName')+"','"+flow.get('ForgjeFart')+"','"+Kjor
eMin+"','"+global.get('KabelDiameter')+"');"
    //Lagrer køyretida i flow
    flow.set('ForgjeKjoreMin', KjoreMin);
    msg.payload=null;

    }
    else
    {
```

```
        //Sender data om feilen til tabellen feilrapport og
prosedyren Brudd med 0 i køyretid
        msg.topic = +
        "CALL
Brudd('"+global.get('RName')+"','"+flow.get('ForgjeFart')+"','"+0+"'
);" +
        "INSERT INTO feilrapport(ProdID, LopNr, Diameter, BruddSted,
Fart, Lengde)" +

"VALUES('"+ProdID+"','"+LopNR+"','"+Diameter+"','"+Brot+"','"+flow.g
et('ForgjeFart')+"','"+ProdLengde+"');"
    }
    //Setter køyretida i flow til null
    flow.set('ForgjeFart',0);
    msg.payload = null;
}
else
{
    msg=null;
}
return msg;
```

Node-RED

FLYMCA

Funksjonsnode, fartsdata til prosedyre

```
var KjoreMin

//Køyrer når den pågåande linjefarten er ulik ForgjerFart og at
ForgjerFart er lik Forgje2Fart (Gjer det mogleg at funksjone henter
data rett før endring) og at ForgjerFart er større enn 0
if (msg.payload.LineSpeed != flow.get('ForgjeFart') &&
flow.get('ForgjeFart') == flow.get('Forgje2Fart')  &&
flow.get('ForgjeFart') > 0 )
{
    //Samanliknar den pågåande produksjonstida med tidlegare
køyretid
    KjoreMin = msg.payload.ReelProductionCumulativeTime -
    flow.get('ForgjeKjoreMin');

    //Passer på at køyretida må vera null eller større og ikkje
negativ, dette er berre for å vera sikker
    if(KjoreMin >= 0)
    {
        //Sender data til prosedyren TimerFart i MariaDB
```

```
        msg.topic = "CALL
TimerFart('"+global.get('RName')+"','"+flow.get('ForgjeFart')+"',
'"+KjoreMin+"','"+global.get('KabelDiameter')+"');";
    }
    else
    {
       msg = null;
    }
    //Lagrer verdiane av fartane og tida for samanlikning til neste
gong
    flow.set('Forgje2Fart',flow.get('ForgjeFart'));
    flow.set('ForgjeFart',msg.payload.LineSpeed);
    flow.set('ForgjeKjoreMin',KjoreMin);
    msg.payload= null;
}
else
{
    //Lagrer verdiane av fartane for samanlikning til neste gong
    flow.set('Forgje2Fart',flow.get('ForgjeFart'));
    flow.set('ForgjeFart',msg.payload.LineSpeed);
    msg = null;
}
return msg;
```

Node-RED

FLYMCA

Funksjonsnode, div. data til InfluxDB

```
var elements = { };

elements.payload = [
    {
     //Gir eit kall med å finna database med namnet flymca med
     innhaldet til msg.payload
        measurement: "flymca",
        fields: msg.payload,
    }
];
return elements;
```

Node-RED

SAMP

Funksjonsnode, produksjonsdata

```
var hours = msg.payload.SecondsCounter;
var minutes = msg.payload.MinutesCounter;
var seconds = msg.payload.HoursCounter;

//Henter tidlegare lagra verdi av produsert lengde og samanliknar
med pågåande produserte lengde
if (global.get('OCM') > msg.payload.OutCounterMeter)
{
    //Gjer om timar og sekundar om til minutt og leggjer det saman
    var totalmin = hours * 60 + minutes + seconds / 60;
    //totalmin er ein sum total produsert tid sida maskinen blei
teken i bruk.
    //For å finna den tida produksjonen har brukt så rekner vi
differansen mellom den totale produserte nåverande tida, med den
tidlegare totale tida 'start'
    var tid = totalmin - global.get('start');

    //Sender data til prosedyren Produksjon i MariaDB
    msg.topic = "Call
Produksjon('"+msg.payload.RecipeName+"','"+msg.payload.OutDiameter+"
','"+msg.payload.SpeedLineSet+"','"+tid+"','"+msg.payload.OutCounter
Meter+"')";

    //Hugser på produksjonstida slik at den kan brukast til neste
gong
    global.set('start',totalmin);
}
else
{
    //hugsar på lengden til samanlikning til neste gong
    global.set('OCM',msg.payload.OutCounterMeter);
    msg = null;
}
return msg;
```

Node-RED

SAMP

Funksjonsnode, div. data til InfluxDB

```
var elements = { };

elements.payload = [
    {
     //Gir eit kall med å finna database med namnet samp med
     innhaldet til msg.payload
        measurement: "samp",
        fields: msg.payload,
    }
];
return elements;
```

Node-RED

Bedrifts PC

Template, vel tabell

```
SELECT
DATE_FORMAT(Timestamp, "%Y-%m-%d %H:%i:%s") AS Timestamp,
ID, Namn, Klokke, tid1, Konfig, Analog
FROM start;
```

Node-RED

Bedrifts PC

Funksjonsnode, Klargjer data

```
newMsg = {};

// Common properties
//Database og innloggings krav for å kunne få tilgang til Odoo
Database
newMsg.common = {
    db : "--------------",
    user: "-------",
    api_key: "-----------------"
}

// Used in the authenticate call
//Samler informasjonen over til ein last
newMsg.payload = [
    newMsg.common.db,
    newMsg.common.user,
    newMsg.common.api_key,
    []
];

//Plukkar ut siste rekkja i tabell slik at ikkje alt innhald av
tabell blir dratt med
var my_array = msg.payload;
var last_element = my_array[my_array.length - 1];

// Used in the "Create pls data call"
//Samler tabell data til ein last. Må kalles x_studio for å bli
akseptert i Odoo Database
newMsg.createData = [{
```

```
    x_studio_database_id : last_element.ID,
    x_studio_lpenummer : last_element.Analog,
    x_studio_prod_id : last_element.Namn,
    x_studio_prod_lengde : last_element.Klokke,
    x_studio_prod_tid : last_element.tid1,
    x_studio_setup_tid : last_element.Konfig,
    x_studio_timestamp : last_element.Timestamp,
}]
return newMsg;
```

Node-RED

Bedrifts PC

Funksjonsnode, Klargjer 'create' kall

```
//Samler alt frå den førre funksjonsnoden
// I tillegg til namnet til tabellen den skal til må den ha ein
metodekall for kva den skal utføra i Odoo
var createCall = {
    payload: [
        msg.common.db,      // Database
        msg.payload,        // UID fra auth kall
        msg.common.api_key, // API Key
        "x_pls_data",       // Modell-navn (tabell)
        "create",           // Action
        msg.createData      // Data fra "Prepare data" step
    ]
};

return createCall;
```

Grafana

Bedrifts PC

Timer kjørt for kvar fart

```
SELECT
  now() as time,
  CONVERT(KjoreFart, CHAR) as metric,
  AntTimer as value
FROM lederbrudd
where ProdID = '$ID'
ORDER BY KjoreFart
```

Produksjonstabell FLYMCA

```
SELECT
`Timestamp` as 'Tidspunkt',
ProdID as 'Produkt ID',
Diameter as 'Diameter',
LopeNR as 'Løpenummer',
ProdTid as 'Produksjonstid (min)',
SetupTid as 'Setuptid(min)',
ProdLengde as 'Produsert lengde(m)',
GjennFart as 'Gjen.Fart(m/min)'
FROM prodtabell
where ProdID = '$ID'
```

Produksjonstabell SAMP

```
SELECT
`Timestamp` as time,
DatabaseID as 'databaseID',
ProdID as 'Produkt ID',
Diameter as 'Dimeter',
ProdTid as 'Produksjonstid (min)',
ProdLengde as 'Produsert lengde(m)',
Fart as 'Fart(m/min)'
FROM prodtabell
where ProdID = '$SAMPID'
```

Forventa timer kjørt til et brudd oppstår

```
SELECT
  now() as time,
  CONVERT(KjoreFart, CHAR) as metric,
  Timerbrudd as value
FROM lederbrudd
where ProdID = '$ID'
ORDER BY KjoreFart
```

Feilrapport FLYMCA

```
SELECT
DatabaseID,
Tidspunkt as Time,
LopNR as 'Trommelnummer',
Diameter,
BruddSted as 'Brudd Sted',
Fart as 'Fart(m/s)',
Lengde as 'Lengde(m)'
FROM feilrapport
where ProdID = '$ID'
```

Antall brudd å kvar enkelt fart

```
SELECT
  now() as time,
  CONVERT(KjoreFart, CHAR) as metric,
  AntBrudd as value
FROM lederbrudd
where ProdID = '$ID'
ORDER BY KjoreFart
```