Høgskulen på Vestlandet

Bacheloroppgave

ELE350

Predefinert informasjon

Startdato:	08-05-2023 09:00 CEST	**Termin:**	2023 VÅR
Sluttdato:	22-05-2023 14:00 CEST	**Vurderingsform:**	Norsk 6-trinns skala (A-F)
Eksamensform:	Bacheloroppgave		
Flowkode:	203 ELE350 1 O 2023 VÅR		
Intern sensor:	Endre Håland		

Deltaker

Navn:	Lars Askild Skauhellen Aarvik
Kandidatnr.:	202
HVL-id:	587850@hvl.no

Informasjon fra deltaker

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:	BO23EB-37
Gruppenummer:	31
Andre medlemmer i gruppen:	Sander Vatne Andersen

Jeg godkjenner avtalen om publisering av bacheloroppgaven min *

Ja

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

Nei

Høgskulen på Vestlandet

BACHELOROPPGAVE:
BO23EB-37 Bølgesimulering og evaluering av metoder for aktiv hiv kompensering

Sander Vatne Andersen
Lars Askild Skavhellen Aarvik

22.05.2023

Dokumentkontroll

Rapportens tittel: BO23EB-37 Bølgesimulering og evaluering av metoder for aktiv hiv kompensering.	Dato/versjon: 22.Mai.2023/1.4
	Rapportnummer: BO23EB-37
Forfatter(e): Sander Vatne Andersen Lars Askild Skavhellen Aarvik	Studieretning: Automatisering og robotikk
	Antall sider m/vedlegg: 242
Høgskolens veileder(e): Endre Håland	Gradering: Åpen
Eventuelle merknader: Vi tillater at oppgaven kan publiseres.	
Oppdragsgiver: Scantrol AS	Oppdragsgivers referanse:
Oppdragsgivers kontaktperson(er) (inklusiv kontaktinformasjon): Espen Karlsen (espen@scantrol.com, 977 78 043) Øystein Aase (oystein@scantrol.com)	

Revisjon	**Dato**	**Status**	**Utført av**
1.0	27.03.2023	Avklart rammeverk	Sander Vatne Andersen, Lars Askild Skavhellen Aarvik
1.1	12.05.2023	Første utkast	Sander Vatne Andersen, Lars Askild Skavhellen Aarvik
1.2	19.05.2023	Andre utkast	Sander Vatne Andersen, Lars Askild Skavhellen Aarvik
1.3	20.05.2023	Korreksjonslesing	Sander Vatne Andersen, Lars Askild Skavhellen Aarvik
1.4	21.05.2023	Ferdigstilt rapport	Sander Vatne Andersen, Lars Askild Skavhellen Aarvik

Forord

Vi er nå ferdig med tre år ved Høgskulen på Vestlandet, og i dette siste semesteret har vi skrevet vår avsluttende bacheloroppgave i Automatisering med Robotikk. Oppgaven har vært både praktisk og teoretisk. Arbeidet har omhandlet å utvikle et styresystem for en bevegelsesplattform med to frihetsgrader og en vinsj som er benyttet for å evaluere metoder for aktiv hiv kompensering.

Vi ønsker å gi en stor takk til Scantrol AS for muligheten til å jobbe på ett så spennende og lærerikt prosjekt. Læringskurven vår har vært bratt dette semesteret og vi sitter igjen med mye ny kunnskap og gode erfaringer i automasjonsfaget. Vi ønsker også å takke vår veileder Endre Håland ved HVL og interne veiledere hos Scantrol, Øystein Aase og Espen Karlsen for gode diskusjoner og oppfølging underveis i prosjektet.

Bergen 22. mai 2023
Sander V. Andersen og Lars Askild S. Aarvik

Sammendrag

Dette bachelorprosjektet er utført i samarbeid med Scantrol AS og Høgskulen på Vestlandet. Prosjektet har gått ut på å fornye styresystem og brukergrensesnitt for en ferdig konstruert bevegelsesplattform med to frihetsgrader, en motor og en vinsj.

Oppgaven er inndelt i tre hoveddeler:
Del 1: Oppgradering og erstatning av styring for bevegelsesplattformen.
Del 2: Utarbeidelse av styring for fastmontert vinsj på plattformen.
Del 3: Evaluering av ulike metoder for aktiv hiv kompensering.

Alle kravene fra Scantrol om brukergrensesnittet som betjener og styrer systemet er implementert. Det er også lagt inn flere tilleggsfunksjoner som lar bruker betjene systemet i ulike moduser. To ulike metoder for aktiv hiv kompensering ble evaluert, hvor den første metoden var beregnet aktiv hiv kompensering og den andre var å benytte Scantrol sitt egenutviklede stereokamera for aktiv hiv kompensering.

Beregnet aktiv hiv kompensering gav gode resultater ved evaluering. Ved bruk av denne metoden stod vinsjen tilnærmet i ro da plattformen simulerte bølgebevegelser. Aktiv hiv kompensering med bruk av stereokamera gav ikke optimale resultater. Utførte tester viste at dette var grunnet uklare signaler og treg data fra stereokameraet.

Videre, etter oppgradering av styresystemet for plattformen og vinsjen i tillegg til evaluering av metoder for aktiv hiv kompensering, er systemet klart for at Scantrol kan benytte det til ytterligere videreutvikling av aktiv hiv kompensering.

Abstract

This bachelor's project is performed in collaboration with Scantrol AS and Høgskulen på Vestlandet. The project has involved renewing the control system and user interface for a fully constructed movement platform with two degrees of freedom, a motor, and a winch.

The project is divided in three main parts:
Part 1: Upgrading and replacing the motion platform control.
Part 2: Control for fixed winch on the platform.
Part 3: Evaluation of different methods for active heave compensation.

All the requirements from Scantrol regarding the user interface that operates and controls the system have been implemented. Further, several additional functions have been added to allow the user to operate the system in different modes. Two different methods for active heave compensation were evaluated. The first method was calculated active heave compensation and the second method was to use Scantrol's own developed stereo camera for active heave compensation.

Calculated active heave compensation gave good results in the evaluation. When using this method, the winch was almost at rest when the platform simulated wave movements. Active heave compensation with the use of a stereo camera, did not give optimal results. Tests carried out showed that this was due to unclear signals and slow data from the stereo camera.

Furthermore, after upgrading the control system for the platform and winch in addition to an evaluation of methods for active heave compensation, the system is ready for Scantrol to use the platform for further development and testing of new methods for active heave compensation.

Innhold

1 Innledning

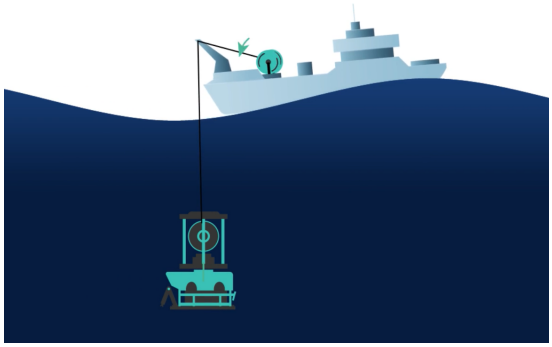
1.1 Oppdragsgiver

Oppdragsgiveren for denne oppgaven er Scantrol AS, videre i rapporten referer vi til firmaet som Scantrol. Scantrol er lokalisert i Sandviken i Bergen og har omtrent 20 ansatte. De er en uavhengig leverandør av overvåknings- og kontrollsystemer innenfor maritim industri, og leverer primært automatiske fiske- og trålesystemer i tillegg til *active heave compensation* (AHC) systemer til offshore applikasjoner. Scantrol leverer systemer til skip over hele verden og har fokus på effektive systemer som har høy brukerverdi. Referanse [2] henviser til Scantrol sin hjemmeside.

1.2 Problemstilling

Scantrol hadde en delvis fungerende, egenutviklet bevegelsesplattform med to frihetsgrader, som har blitt brukt til testing og optimalisering av *motion reference unit* (MRU) og AHC systemer.

AHC er en teknikk for å minimere innflytelsen bølger har på offshore applikasjoner. Denne teknikken blir mye brukt på offshore båter med kraner, fjernstyrte undervannsfarkoster (ROVer), annet utstyr som opererer nærme havbunnen eller der det er viktig at lasten står i ro. AHC blir vanligvis implementert ved at en MRU sensor måler bølgebevegelsene, og at et kontrollsystem både regner ut hvor fort og hvor mye vinsjen må rotere for å motvirke denne bølgebevegelsen. Systemet gjør at vinsjen til enhver tid jobber mot bølgene, slik at lasten tilkoblet vinsjen holder seg mest mulig i ro, dette blir illustrert i figur 1.1.



Figur 1.1: Illustrasjon av aktiv hiv kompensering. Hentet fra [1], 01.05.2023.

Scantrol ønsket å se på muligheten for å benytte AHC ved flytting av en last fra et fartøy til et annet, eller fra en stasjonær plattform til et fartøy. Slike systemer finnes allerede i dag, men er gjerne mer komplekse enn vanlige AHC systemer. Ved flytting av en last fra et fartøy til et annet, er bevegelsene i hvert fartøy uavhengig av hverandre. I det tilfellet er det bevegelsene i fartøyet som tar i mot lasten som er interessant for AHC systemet. Systemene som brukes i dag tar gjerne i mot MRU signaler trådløst fra fartøyet som skal motta lasten, og bruker dette til å kompensere for bevegelsene til fartøyet. Scantrol ønsket å se på muligheten for å benytte alternative måle- og reguleringsmetoder, som blant annet baserte seg på bruk av avstandsmåling og visjonsteknologi for å løse dette problemet.

Oppdraget fra Scantrol var å fornye styringen av bevegelsesplattformen, i tillegg til styringen av fastmontert vinsj. Videre skulle mulighetene for bruk av avstandsmåling og visjonsteknologi for å styre vinsjen montert på bevegelsesplattformen med AHC undersøkes.

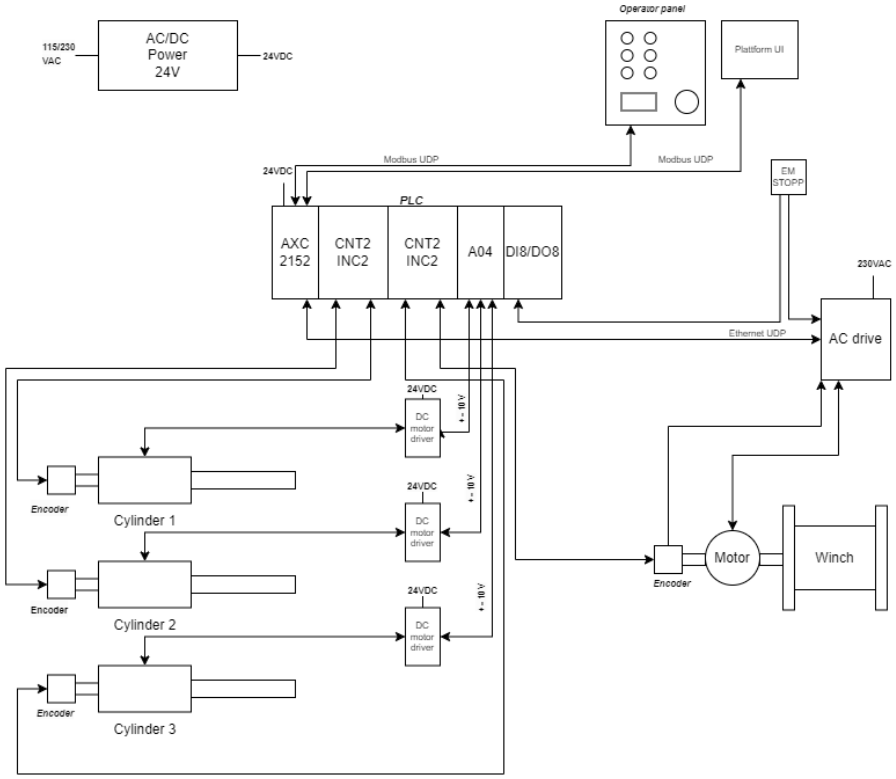
1.3 Systemoversikt

Scantrol hadde en forenklet, konstruert bevegelsesplattform, men til dette prosjektet ble all *hardware* og styring byttet ut. Det eksisterende systemet besto av en bevegelsesplattform med påmontert motor og vinsj. Bilde 1.2 viser det oppgraderte systemet med alle tilhørende komponenter. Systemet skulle brukes til dette prosjektet og videre arbeid hos Scantrol. Det skal kunne benyttes for å utvikle og teste ulike metoder for AHC, i tillegg til å demonstrere Scantrol sine AHC systemer for kunder og lignende. Bevegelsesplattformen skal simulere bølgebevegelsene til et fartøy i sjøen. Det er både billigere og enklere å utvikle og teste systemer på en nedskalert modell enn på ett ekte fartøy. Plattform og vinsj har følgende spesifikasjoner:

- **Tre sylindre:** Plattformen har tre lineært aktuerte sylindre med 30[cm] sylinderutslag.
- **To frihetsgrader:** Plattformen har to frihetsgrader, *heave* og *roll*. Det vil si at de to fremste sylindrene alltid skal gå synkront.
- **Motor/vinsj:** Det er montert en 0.37[kW] tre fase asynkron induksjonsmotor som er koblet på en vinsj via en $\frac{1}{10}$ girkasse.
- **Operatørpanel:** For styring av vinsj.
- **HMI:** Utviklet til dette prosjektet for styring og monitorering av plattform og vinsj.

Figur 1.2: Oversiktsbilde over bevegelsesplattform med vinsj, operatørpanel og HMI.

En oversikt over komponentene i systemet blir illustrert i blokkskjemaet i figur 1.3.



Figur 1.3: Blokkskjema over PLS, sylindre, operatørpanel, frekvensomformer, motor og vinsj.

Tabell 1 viser en oversikt over utførte oppgaver i prosjektet og hvem som har gjort hva.

Utførte oppgaver:	**Utført av studenter**	**Utført av Scantrol**	**Kort oppgavebeskrivelse:**
Bygget fysisk plattform modell		X	Plattformen og vinsj er ferdigbygget fra tidligere.
Ny hardware og oppkobling	X		Fjernet gammel hardware. Valg av komponenter for ny styring av plattform. Koblet nytt system.
Software HMI og PLS kode	X		Laget helt nytt brukergrensesnitt i C#, WPF design. Laget helt ny software og styring av plattform/vinsj, PLS-kode er egenutviklet med unntak av tre stk funksjonsblokker "fbRamping$_V1_001$","$fbSpeed_V1_021$","$fbEncoderCount_V1_001$".
Dokumentasjon	X		Dokumentasjon av tester, protokoller, tuning, tegninger, formler, etc... Har benyttet Scantrol sitt standard protokoll dokument for kommunikasjon, innholdet er utført av oss.
Evaluert metoder for AHC	X		Testet metode for beregnet AHC og AHC med stereokamera.

Tabell 1: Oversikt over utførte oppgaver i prosjektet.

1.4 Oppgavebeskrivelse

Bacheloroppgaven består av tre hoveddeler i tillegg til ett kapittel om HMS. Del 1 omhandler oppgradering og erstatning av styringen for bevegelsesplattformen, Del 2 dreier seg om utarbeidelse av styring for fastmontert vinsj på plattformen, Del 3 tar for seg evaluering av ulike metoder for aktiv hiv kompensasjon. Foruten om dette er det utviklet software for HMI som er lagt ved som vedlegg 15 og for Phoenix PLC next som vedlegg 16. Detaljert oppgavebeskrivelse fra Scantrol er som følger:

Del 1:
Utvikle et styresystem for en sylinderstyrt bevegelsesplattform med to frihetsgrader, tilt og hiv, herunder:

- Valg av drivere for styring av sylindermotorer.
- Lese sylinderposisjon fra inkrementelle enkodere for å beregne hastighet og posisjon.
- Sette sammen og koble opp PLS og IO moduler for sylinderstyring.
- Lage reguleringssløyfe for posisjon- og hastighetskontroll av sylindre, samt synkronisering av bevegelse mellom sylindre.
- Lage programvare for å styre plattformen med ulike bevegelser, både ved parametrisering og ved lesing av loggfil fra bølgebevegelser fra skip. Programmet må også fremstille plattformens bevegelse og nøyaktigheten til regulatoren som benyttes.

Del 2:
Lage et PLS-program for styring av en elektrisk vinsj som er montert på bevegelsesplattformen. Denne har en motor koblet til en Vacon frekvensomformer:

- Koble opp frekvensomformer og inkrementell enkoder til PLS.
- Kommunisere med frekvensomformer over nettverk med UDP.
- Styre hastighet og dreiemoment på vinsj.
- Måle posisjon på vinsj basert på inkrementell enkoder.
- Forklare prinsippet bak elektrisk motor og frekvensomformer.
- Benytte standard operatørpanel med joystick, knapper og potensiometer for å styre vinsj.

Del 3:
Teste ut ulike metoder for å hivkompensere vinsjen når plattformen følger forskjellige bevegelser. Dette kan inkludere å utvikle regulatorer for ulike metoder. Nøyaktigheten til de ulike metodene skal dokumenteres og diskuteres. Relevante metoder kan være, men er ikke begrenset til:

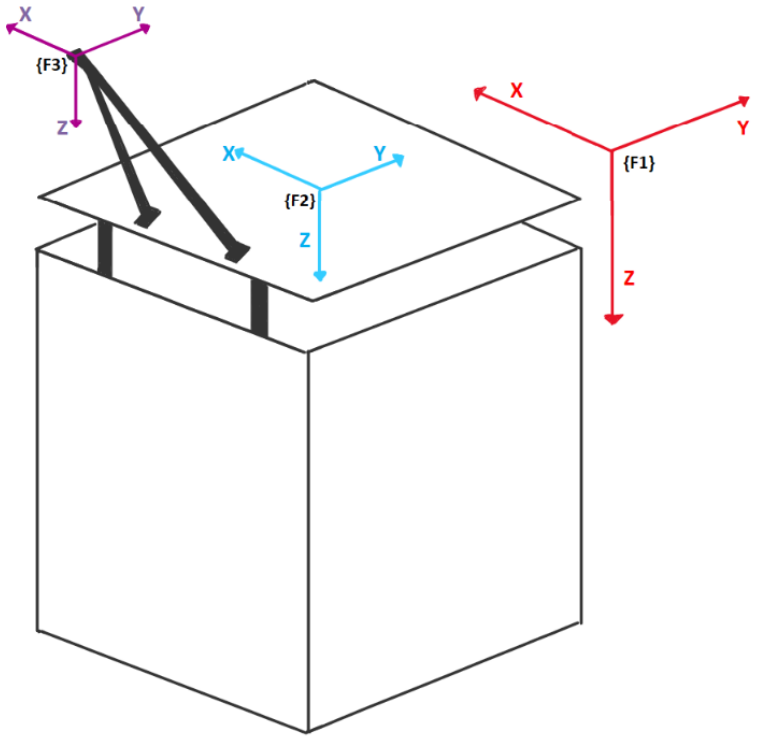
- Benytte Scantrol AS sin mTrack AHC controller.
- Benytte radarsensor eller laser for å måle avstand fra last i krok til gulv.
- Benytte Scantrol AS sitt stereokamera for å bestemme avstand fra last til gulv.

1.5 Forkortelser og ordforklaringer

- AHC – Aktiv hiv kompensering.
- Feedforward – Foroverkopling, regulering uten tilbakemelding.
- GUI – Graphical user interface.
- Heave position – bevegelse langs Z - akse.
- HMI – Human machine interface.
- HMS – Helse, miljø og sikkerhet.
- MP – Measuring point.
- MRU – Motion reference unit.
- Pitch angle – rotasjon rundt Y - akse.
- PLS – Programmerbar logisk styring.
- Roll angle – rotasjon rundt X - akse.
- PPS – Pulses per second.
- PPR – Pulses per revolution.
- K_k – Ziegler Nichols kritisk forsterkning.
- T_k – Ziegler Nichols kritisk periodetid.
- WPF – Windows presentation foundation.

2 Bevegelsesplattform

Figur 2.1 viser en illustrasjon av bevegelsesplattformen med tre forskjellige koordinatsystem, *world-frame* (F1), *plattform frame* (F2) og *crane-tip frame* (F3). Videre i rapporten omtales rotasjon rundt X-aksen som *roll angle* og bevegelse langs Z-aksen som *heave position*. Positiv rotasjon rundt aksene er definert mot klokken, negativ rotasjon er definert med klokken. Senterpunktet for de simulerte bølgene er 15[cm] *heave position*, dette er halvparten av fullt sylinderutslag.



Figur 2.1: Illustrasjon av bevegelsesplattformen med tilhørende koordinatsystem.

2.1 Valg av hardware

Plattformsystemet består av en del hardware for styring og monitorering. Detaljert materialliste med kilder til komponentene er dokumentert i vedlegg 10.6.

- **Strømforsyning**
  Ettersom styringselektronikken og sylindrene går på 24[V] ble det benyttet en 20[A], 24[V] strømforsyning, denne er stor nok for å drive sylindrene og resten av styresystemet.
- **Kontroller**
  Systemet blir kontrollert av en PLCnext ax2152 kontroller. Dette er PLSen som Scantrol vanligvis bruker for styring.
- **I/O**
  Det er benyttet diverse I/O moduler for innganger og utganger fra PLSen til systemet.
  - **CNT2/INC2**
    Det er benyttet to «CNT2 INC2» moduler, som brukes som inngang for enkodere. De har to enkoder-innganger hver, som gir mulighet for tre enkodere fra sylindre og en fra vinsj.
  - **AO4**
    Det er benyttet en «AO4» modul som har fire analoge utganger, hvor tre av dem benyttes for analog hastighetskommando til sylindrenes motordrivere som styres med et ± 10[V] signal.

- **DI8/DO8**
  Det er benyttet en «DI8/DO8» inngangs-/utgangsmodul. Denne har åtte digitale innganger og åtte digitale utganger. Her benyttes blant annet aktivering av enable rele og inngang for nødstopp.

- **Motordriver**
  Sylinderene er av typen *linear actuator* som styres av en elektrisk DC motor. Denne var tidligere styrt av en DC motordriver, som ikke kunne gjenbrukes fordi den analoge inngangen for hastighetstyringen var ±12[V]. PLSen som benyttes kan kun gi ut ±10[V], derfor ble motordriverne byttet ut med et tilsvarende alternativ som styres med riktig spenning.

- **Enable rele**
  Siden motordriverene styres ved ±10[V] signal var det hensiktsmessig å installere enable rele på analogutgangen. Hensikten med disse releene var å kunne sette referanse inngangen på motordriveren til 0[V] ved behov. Dette gjør at man unngår at sylindrene «drifter» når PLSen sender 0% i hastighetskommando, da den i praksis kan sende små spenninger uansett. Når releene er aktive sendes hastighetssignal fra PLS, og når de er innaktive sendes 0[V].

- **Nødstopp rele**
  For å ivareta sikkerheten i systemet, ble det montert nødstopp. Denne er *normally closed* (NC), slik at eventuelle feil på kabling og lignende blir oppdaget. Nødstoppbryteren ble direkte koblet til et 3polt rele. Dersom nødstoppen blir aktivert kuttes strømmen til releet, som igjen kutter strømmen til sylindrenes motordrivere. Dette medfører at plattformen stopper tilnærmet momentant. Som følge av at sylindrene er styrt av en DC motor som er induktiv, ble det montert diode-rekkeklemmer på tilførselen til motordriverne, disse kan brenne vekk eventuell strøm fra motorene over diodene i motsatt retning ved strømbrudd.

2.1.1 Inkrementell enkoder

Det ble benyttet inkrementelle enkodere i dette systemet for posisjons og hastighetsmåling av sylindrene og vinsj. Enkoderen har to kanaler, A og B, som sender ut pulser når akselen roterer. Enkoderne som er benyttet i dette systemet har også kanalene A^{-1} og B^{-1} som er invertert av A og B. Kanalene A og B sender ut pulstog som er 90° faseforskjøvet i forhold til hverandre. Basert på om de er positivt eller negativt faseforskjøvet kan man tolke retningen akselen roterer. Ved å telle antall pulser kan man vite posisjonen til utstyret som er koblet sammen med enkoderen, hyppigheten av pulsene gjør at man kan beregne rotasjonshastigheten. Nøyaktigheten av posisjons- og hastighetsmålinger er avhengig av oppløsningen til enkoderen. Jo flere pulser det er per rotasjon, desto bedre oppløsning og nøyaktighet oppnås. Inkrementelle enkodere har, til forskjell fra absolutt enkodere, ingen data på hvor de befinner seg. PLSen må dermed selv telle og beregne hvor sylindrene/vinsjen befinner seg til enhver tid. Det er viktig at systemet har gode muligheter for å resette posisjonen sin slik at PLSen har et referansepunkt å beregne riktig posisjon fra.

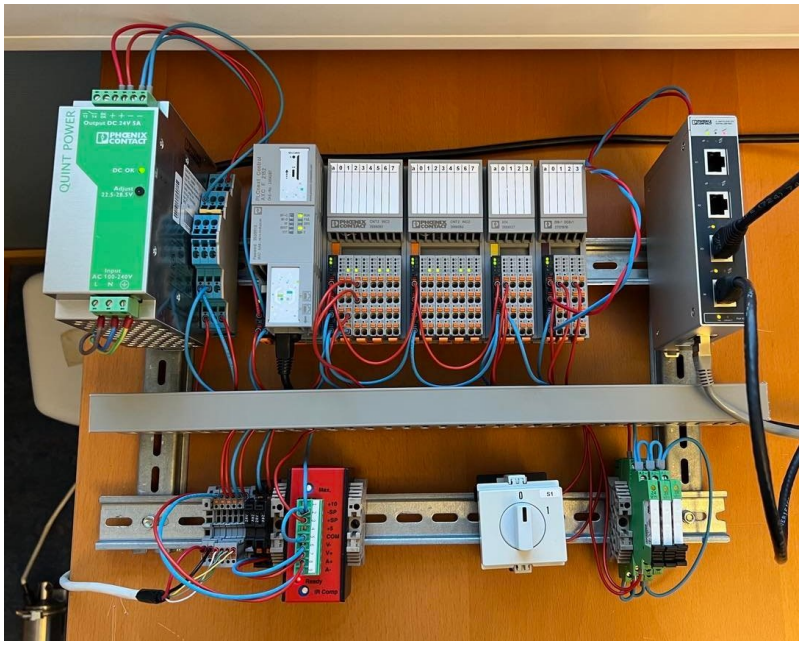
For sylindrene i dette systemet ble det ikke oppgitt hvor mange pulser per rotasjon enkoderene hadde, istedet ble det målt hvor mange pulser som ble mottatt på hele sylinderutslaget. Resultatet ble 800 pulser på 30[cm], dette gir en oppløsning på 0.375[mm/puls]. I PLSen ble det implementert en funksjonsblokk 1 som signalbehandler enkoderpulsene som input-signal og gir output-signal posisjon[m].

```
1  ///*************************************************************************
2  //Code information:
3  //Function block that scales pulses given from encoder to position in meter.
4  //
5  //Author: Sander V. Andersen and Lars Askild S. Aarvik
6  //Date: 14.02.2023
7  //*************************************************************************/
8
9
10 udiRawDataUDINT           :=      TO_UDINT(dwRawDataEncoder); // For calulation
11
12 rOutValueInMeter          :=      TO_REAL(udiRawDataUDINT) * (rDistance/rPulses); // Outvalue -
       position[m]
```

Listing 1: Funksjonsblokk som skalerer pulser fra enkoder til sylinder posisjon i meter.

2.2 Testoppsett

Figur 2.2 viser bilde av testoppkoblingen for en sylinder. Figur 2.3 viser sylinderen alle innledende tester ble utført på, denne testsylinderen er av samme modell som sylindrene montert på plattformen. Tester og kalibreringer vil bli beskrevet i kapittel 2.3 og 2.4.

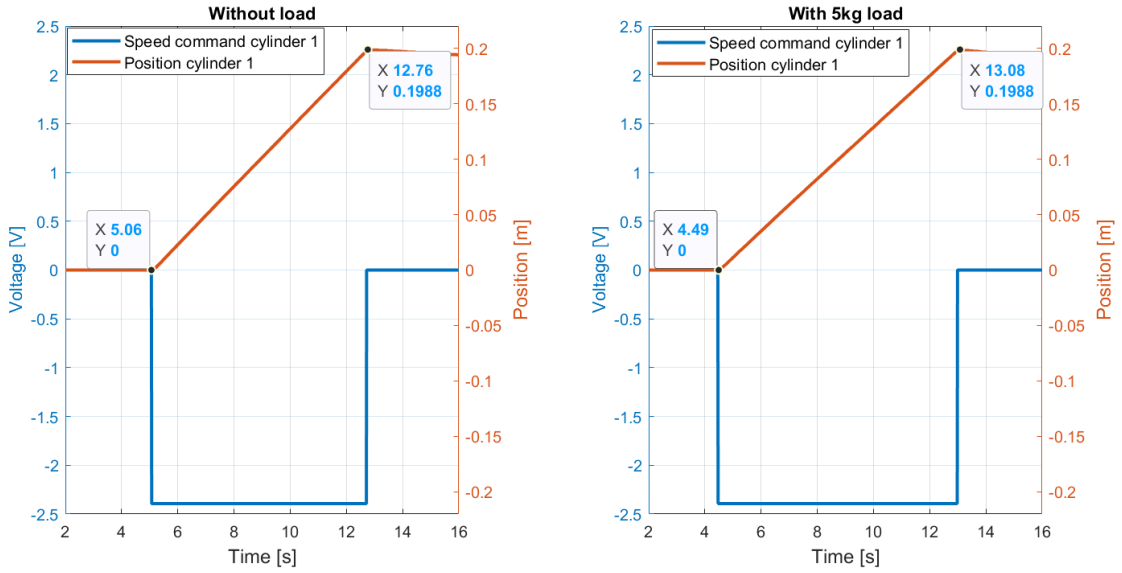


Figur 2.2: Hardware oppkobling for testoppsett av en sylinder.

Figur 2.3: Testsylinder. Alle innledende tester og kalibreringer ble utført på denne testsylinderen.

2.3 Lastavhengighet

Det ble utført diverse lasttester på sylindrene for å avdekke en eventuell lastavhengighet. Dette var viktig med tanke på hvordan systemet skulle designes videre. Dersom sylindrene var veldig lastavhengige, ville karakteristikken til sylindrene endret seg basert på lasten, som igjen går ut over posisjonsreguleringen. På testoppsettet ble det utført flere tester der sylinderen kjørte fra 0[cm] til 20[cm] med fast hastighetskommando (-2.3[V]). Testene ble utført en gang uten last og en gang med et 5[kg] lodd festet til sylinderen. Figur 2.4 viser resultatene av de utførte testene. Tidsdifferansen var 7.7[s] uten last og 8.59[s] med 5[kg] last. Resultatet viste at sylinderen med last brukte 0.89[s] lengre tid på samme avstand og ved samme hastighetskommando enn sylinderen uten last. Gjennomsnittlig hastighetsforskjell for sylinderen med og uten last var på ca. 0.0027[m/s], dette ble tatt hensyn til for videre valg av regulator for systemet.



Figur 2.4: Last-avhengighetstest av sylinder med og uten 5[kg] last, hvor sylinder har kjørt 20[cm] med fast hastighetskommando[V]. Resultatet viste at sylinder med last bruker 0.89[s] lengre tid på samme avstand og hastighetskommando enn sylinder uten last.

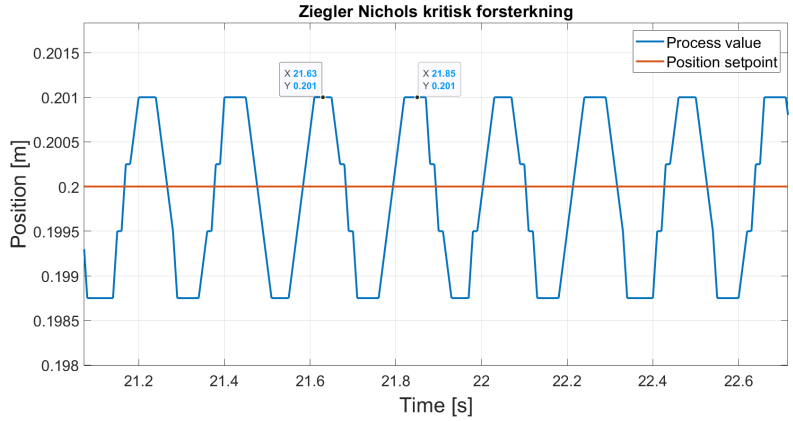
2.4 Optimalisering av prosess

2.4.1 Posisjonsregulering

Sylindrene plattformen er utstyrt med har bevegelighet på 30[cm] og blir regulert etter posisjon i [m] ved bruk av en posisjonsregulator. En posisjonsregulator mottar tilbakemelding av sylinder-posisjonen og sammenligner den med ønsket posisjon. Deretter korrigerer regulatoren prosessen med en kalkulert forsterkning[7]. Videre i kapittelet vil optimaliseringsprosessen av regulatorene til plattformen bli beskrevet.

2.4.1.1 Ziegler Nichols første metode Ziegler Nichols første metode baserer seg på å finne systemets kritiske forsterkning, der prosessen beveger seg i harmoniske svingninger[7]. Denne metoden ble brukt for å finne ett utgangspunkt til forsterkning for videre optimalisering av P-regulatoren. Figur 2.5 viser at systemet går i harmoniske svingninger med en tilført forsterkning K_k på 50 000[V/m]. Figur 2.5 gir følgende informasjon:

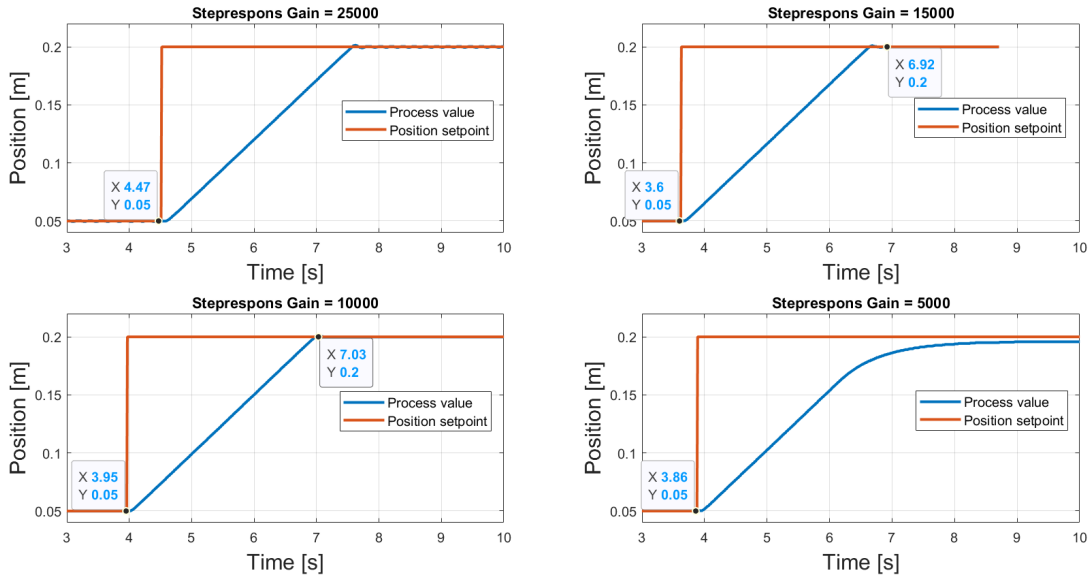
- $T_k = 0,22[s]$



Figur 2.5: Plot som viser stabile harmoniske svingninger under utførelse av Ziegler Nichols første metode, det er harmoniske svingninger ved en periodetid på 0.22[s] og en forsterkning på 50 000[V/m].

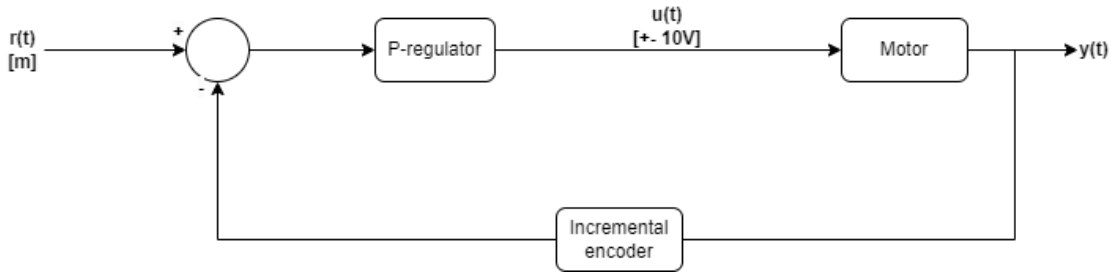
2.4.1.2 P-regulator og steprespons Etter gjennomført Ziegler Nichols metode, ble posisjonsregulatoren optimalisert med ulike forsterkninger for å finne raskest mulig stigetid uten statisk avvik. Ved å bruke formel 1 der K er forsterkningen, ble det beregnet en forsterkning på 25 000[V/m] som utgangspunkt for P-regulatoren[7]. Figur 2.6 viser stepresponser ved ulike forsterkninger. Forsterkning på 10 000[V/m] ga raskest responstid uten statisk avvik mellom ønsket-verdi og er-verdi. Forsterkningen ga også god stabilitet uten oversving, som var ønskelig i prosessen. Denne forsterkningen ble benyttet videre for P-regulatoren.

$$K = K_k \cdot 0.5 \tag{1}$$



Figur 2.6: Steprespons av testsylinder ved forskjellige forsterkninger, steg fra 0.05[m] til 0.20[m]

Reguleringssløyfen for P-regulatoren som ble benyttet under stepresponsene ovenfor er illustrert i figur 2.7.



Figur 2.7: Blokkskjema som illustrerer reguleringsløyfen til en enkel P-regulator.

Ligning 2 beskriver pådragsignalet beregnet av P-regulatoren, hvor e(t) er avviket mellom er-verdi og ønsket-verdi og u(t) er pådragssignalet $\pm 10[V]$.

$$u(t) = K \cdot e(t) \tag{2}$$

2.4.1.3 Valg av posisjonsregulator Bevegelsesplattformen har hele tiden ett flyttende referansesettpunkt, det vil dermed ikke bygges opp et statisk avvik over tid, I-leddet benyttes dermed ikke. D-leddet baserer seg på å bruke den deriverte av feilen til prosessen, raske endringer i settpunkt kan dermed medføre raske endringer i feilen, den deriverte av dette vil kunne føre til en uønsket forsterkning på systemet[7]. Ettersom en enkel P-regulator viste gode resultater med rask stigetid og et akseptabelt, statisk avvik, ble dette vurdert som en god regulator for prosessen.

2.4.2 Hastighetsregulering

Ettersom en P-regulator venter på tilbakemelding fra prosessen, vil den kunne bli hengende etter settpunktet sitt, og det vil kunne oppstå statisk avvik i prosessen. Dermed ble det utarbeidet en foroverkopling som kan forutse posisjonsendringene i settpunktet før de skjer, og gi ett pådrag basert på dette. Ved å derivere posisjonssettpunktet får man et hastighetssettpunkt i [m/s] som kan brukes til å styre sylindrene. Kjente hastigheter ved et gitt pådrag kan benyttes for å styre sylindrene direkte uten å vente på tilbakemelding fra prosessen. Det vil kunne gi en raskere prosess som følger settpunktet sitt bedre, enn ved en ren P-regulator.

2.4.2.1 Feedforward regulator Foroverkopling forutsier prosessens endringer før de oppstår, i motsetning til en regulator med tilbakekobling som reagerer på prosessens endring[7]. Ved riktig kalibrering kan man dermed oppnå en rask og nøyaktig regulering. Ettersom en foroverkopling ikke mottar tilbakemelding om prosessens oppførsel, vil systemet lett kunne påvirkes av for eksempel lastendringer, og begynne å «drifte» ut av ønsket posisjon.

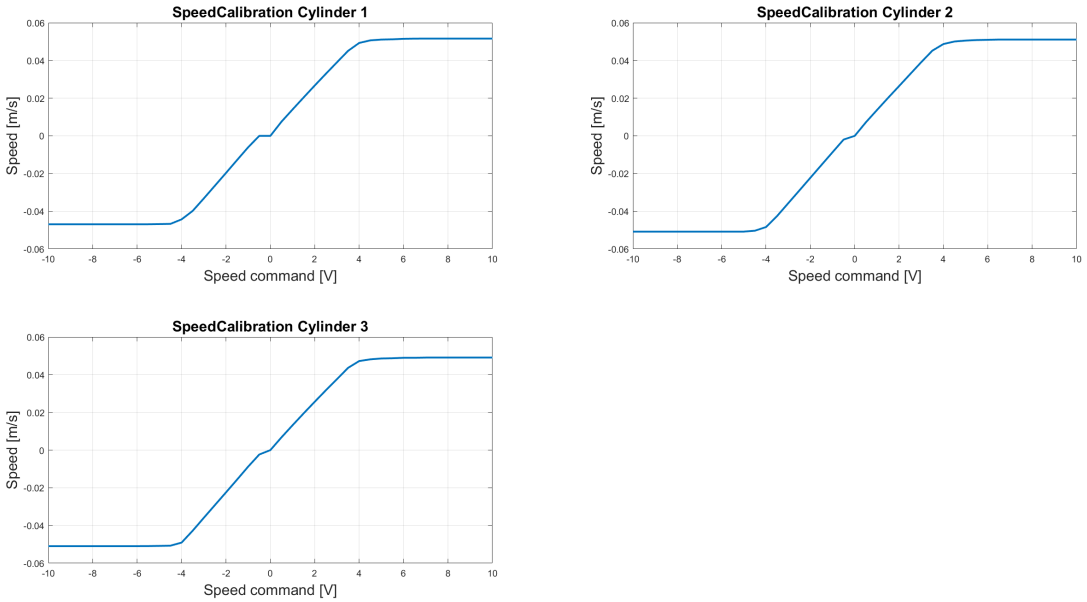
2.4.2.2 Hastighetskalibrering For å kunne oppnå ønsket regulering med Feed-Forward regulatoren, ble sylindrene hastighetskalibrert hver for seg. Ved å utføre denne kalibreringen vet PLSen hvilket pådrag som gir hvilken hastighet. Hastigheten til sylindrene ble beregnet ved å kjøre hver sylinder 20[cm] med forskjellige hastigheter og ta tiden sylinderen brukte på distansen. Dette ble gjort med hastighetskommandoer fra -10[V] til +10[V].

Kodeutklipp 2 viser implementert funksjonsblokk i PLS som er brukt for å utføre hastighetskalibreringen av sylindrene.

```
1  ///**************************************************************************
2  //Code information:
3  //Function block that calculates cylinders average velocity with a given speed command [V].
4  //
5  //Author: Sander V. Andersen and Lars Askild S. Aarvik
6  //Date: 01.02.2023
7  //**************************************************************************/
8
9  IF(xActivate) THEN // Bool that activates the function block
10
11     IF((ABS(rCylinderPos - rStartPos) < rTestRunLength) AND NOT(xPosReached)) THEN
12         rSpeedCmdOut            := rSpeedCmd;
13         xEnableRelay            := TRUE;
14         iTimerCount             := iTimerCount + 1;
15         iTimeElapsed            := iTimerCount * 10;
16
17     ELSE
18
19         rSpeedCmdOut            := 0;
20         xEnableRelay            := FALSE;
21         iTimeElapsed            := iTimerCount * 10;
22         if(NOT(xPosReached))THEN
23             rAvgVelocity        := 1000.0* ABS(rCylinderPos - rStartPos)/(TO_REAL(iTimeElapsed)
       *0.001); // Avg cylinder vel in [m/s]
24             END_IF;
25         xPosReached             := TRUE;
26         END_IF;
27
28
29 ELSE // Resetting the current meassurment
30     iTimerCount                 := 0;
31     rStartPos                   := rCylinderPos;
32     rAvgVelocity                := 0;
33     iTimeElapsed                := 0;
34     rSpeedCmdOut                := 0;
35     xEnableRelay                := FALSE;
36     xPosReached                 := FALSE;
37     END_IF;
```

Listing 2: Funksjonsblokk for hastighetskalibrering av sylindrene.

Figur 2.8 viser et plot av hastighetskommando ±10[V] mot faktisk hastighet for hver sylinder. Merk dødbåndet rundt 0[V] hastighetskommando og at maksimal hastighet i hver retning er nådd allerede ved omtrent ±5[V]. Resultat av detaljerte hastighetskalibreringer er dokumentert i tabell 12.2.



Figur 2.8: Plot som viser sammenhengen mellom hastighetskommando og hastighet for hver sylinder.

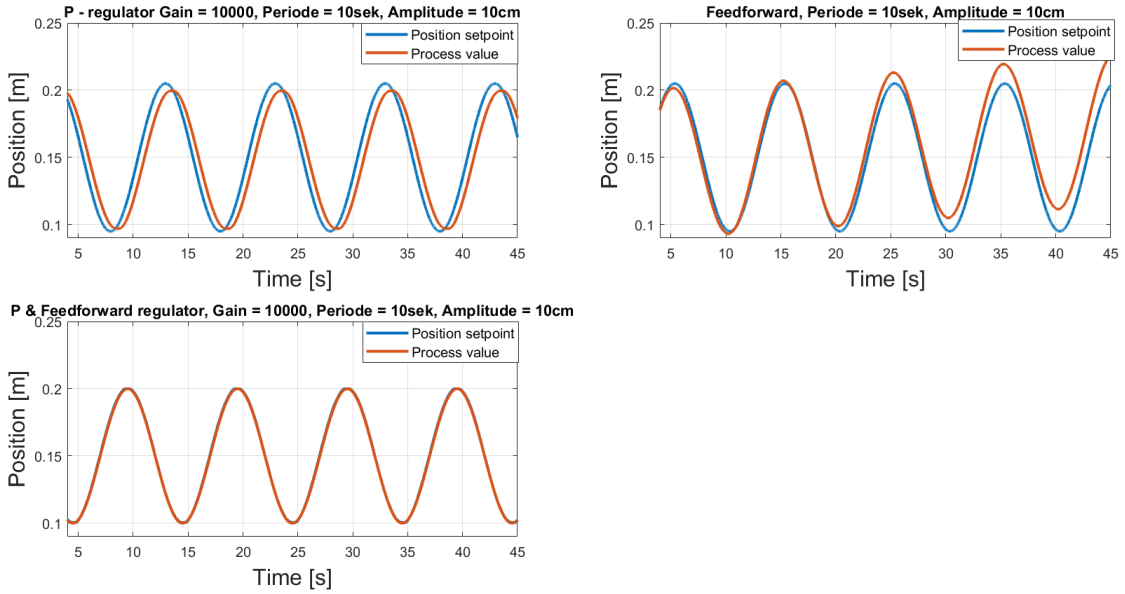
Resultatene av kalibreringen brukes videre i funksjoner laget i PLSen. Funksjonene har ønsket hastighet i [m/s] som inngangsargument (Speed_SP), og returnerer hastighetskommando til gitt sylinder som et ±10[V] signal. I funksjonen er det lagt inn 20 konstanter som tilsvarer de kalibrerte verdiene fra hastighetskalibreringen. Mellom hvert punkt er det lineært interpolert, dette medfører at dersom ønsket hastighet ligger mellom to punkter, returneres det et punkt som ligger på den lineære linjen mellom dem, slik som vist i figur 2.8. Formel 3 viser beregningen som er benyttet for å lineært interpolere mellom punktene.

- SP – Hastighetssettpunkt i [m/s].
- Cmd – ±10[V] hastighetskommando til sylindrene.
- x – Hastighetskommando ved ett gitt punkt [V].
- y – Hastighet ved ett gitt punkt i [m/s].

$$Cmd = \frac{x_1 - x}{y_1 - y} \cdot (SP - y_1) + x_1 \tag{3}$$

2.4.3 Valg av regulator

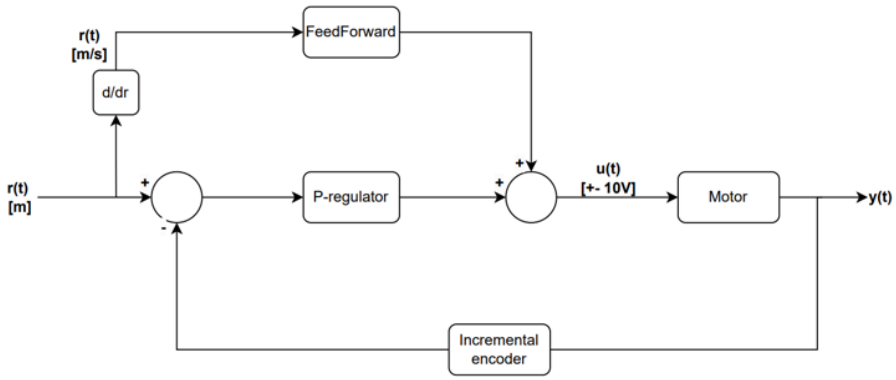
For å oppnå en optimal regulering av sylindrene i systemet, ble det valgt en kombinasjon av en hastighetsregulator (feedforward) og en posisjonsregulator (P - regulator). Plottet øverst til venstre i figur 2.9 viser at prosessen henger etter settpunktet ved bruk av en P - regulator alene. Systemet sliter også med å nå topp og bunn, spesielt ved endring i last, grunnet lastavhenigheten til sylindrene. Plottet øverst til høyre i figur 2.9 viser at prosessen kan «drifte» ved en ren hastighetsregulering (feedforward). Ettersom PLSen ikke får tilbakemelding om hva som faktisk skjer i prosessen, klarer ikke regulatoren å rette opp eventuelle feil som oppstår underveis. Plottet nederst i figur 2.9 viser en kombinasjon av disse to regulatorene, her kan man se at posisjon og posisjonssettpunktet følger hverandre optimalt.



Figur 2.9: Plot som viser resultat av forskjellige regulatorer for sylinderstyring.

Blokkskjemaet i figur 2.10 viser reguleringsløyfen for den implementerte regulatoren for systemet. Formel 4 viser utregning av pådragsignalet for P- og feedforwardregulatoren der d(t) er avviket mellom er-verdi og ønsket-verdi, FF representerer pådraget fra feedforwardregulatoren og u(t) er pådragssignalet til sylindrene ±10[V].

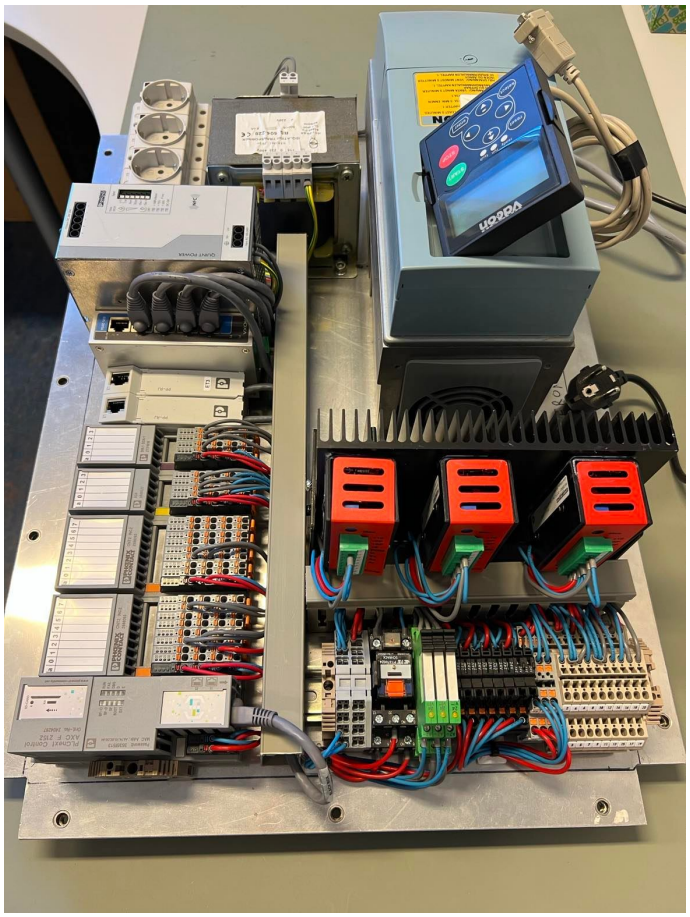
$$u(t) = K \cdot d(t) + FF \tag{4}$$



Figur 2.10: Blokkskjema av reguleringsløyfe med FeedForward og P-regulator for regulering av bevegelsesplattform etter posisjon og hastighet.

2.5 Overgang til bevegelsesplattform

Etter arbeidet i kapittel 2.3 og 2.4, som omhandlet optimalisering og regulering av testsylinderen, startet overgangen til bevegelsesplattformen. Her startet arbeidet med å fjerne systemets tidligere *hardware*. Videre i prosessen tegnet vi elektriske tegninger for nytt egenutviklet styresystem i «*haBit software*», disse er dokumentert i vedlegg 14. Ferdig oppkobling av ny hardware er vist i figur 2.11. Etter oppkoblingen ble PLS software utvidet fra å styre en sylinder til tre sylindre.



Figur 2.11: Bilde som viser ferdig oppkobling av hardwarekomponenter, platen er fastmontert helt i bunnen av bevegelsesplattformen.

2.6 Beregning av sylinderposisjonssettpunkt

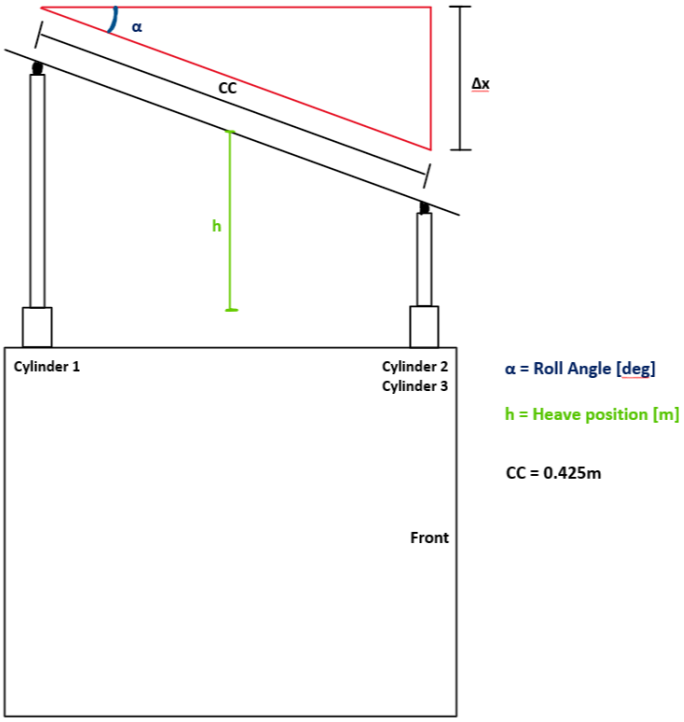
Figur 2.12 viser kinematikken til bevegelsesplattformen. Plattformen består av 3 sylindre, som hver kan bevege seg fra 0-30[cm]. Sylinder 2 og 3 skal bevege seg synkront, dette innebærer at plattformen får to frihetsgrader, *heave* og *roll*. MRU loggfilene plattformen skal bevege seg etter inneholder *heave position* og *roll angle* data. Det må dermed beregnes sylinderposisjoner for en gitt *heave* og *roll*. Formel 5 viser utregning av sylinder 1 sitt posisjonssettpunkt gitt *heave position* og *roll angle*. Formel 6 viser utregning av sylinder 2 og 3 sitt posisjonssettpunkt gitt *heave position* og *roll angle*.

- CC = 0.425[m]
- h – *Heave position* [m].
- Syl_{SP} – Sylinderposisjonssettpunkt.

$$Syl1_{SP} = h + \frac{\Delta X}{2} \tag{5}$$

$$Syl2\_3_{SP} = h - \frac{\Delta X}{2} \tag{6}$$

$$\Delta X = \sin(\alpha) \cdot CC \tag{7}$$



Figur 2.12: Illustrasjonen viser mål som er brukt for beregning av sylinderposisjoner basert på *heave positon* og *roll angle*.

Kodeutsnitt 3 viser utregning av sylinderposisjoner, gitt *heave position* og *roll angle* i Platform HMI.

```
1 // Cylinder 1 position
2 Cylinder1_Pos.Add(Convert.ToSingle(Heave_Pos[i] + ((Math.Sin(Convert.ToDouble(Roll_Angle[i] * (
      Math.PI / 180))) * PlattformUI.PlattformCC) / 2)));
3
4 // Cylinder 2 and cylinder 3 position
5 Cylinder2_3_Pos.Add(Convert.ToSingle(Heave_Pos[i] - ((Math.Sin(Convert.ToDouble(Roll_Angle[i] *
      (Math.PI / 180))) * PlattformUI.PlattformCC) / 2)));
```

Listing 3: Kodeutsnitt med utregning av sylinder posisjoner i C#

2.7 Beregning av sylinderhastighet

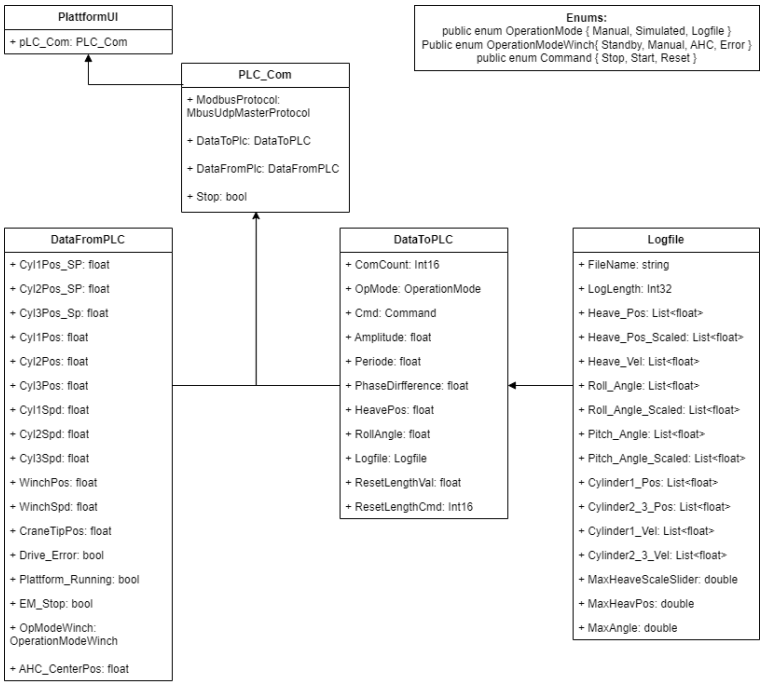
Beregning av sylinderhastighet basert på enkoder tilbakemelding er implementert i PLS ved hjelp av en funksjonsblokk «fbSpeed_V1_02». Den beregnede sylinderhastigheten blir ikke benyttet i reguleringen, men blir vist grafisk i HMIen under «Service» taben. Funksjonsblokken beregner hastigheten ved å telle antall pulser mottatt på en gitt tid, og dermed beregnes *pulses per second* (PPS). Dette er konventert til [m/s], ettersom at vi kjenner antall meter per puls.

2.8 Brukergrensesnitt

Brukergrensesnittet for styring og monitorering av plattform og vinsj er utviklet i «*Visual Studio*» og er programmert i C#. Det er brukt WPF designarkitektur, da dette gir større mulighet for moderne design og skalering av applikasjonen. Dette er nyttig da det er stor sannsynlighet for at datamaskiner med ulik oppløsning kommer til å bruke applikasjonen. Dette ble tatt hensyn til i designprosessen, slik at applikasjonen skalerer seg etter valgt oppløsning. Figur 2.13 viser UML diagram for Plattform UI brukergrensesnittet.

Det ble brukt en rekke biblioteker i dette programmet for design, kommunikasjon og graf plotting. Dette ble gjort for å spare både tid og arbeid, da mange av de nevnte funksjonene allerede finnes i eksisterende biblioteker.

MaterialDesignThemes(4.6.1)
Dette er et *open source* bibliotek som er enkelt å bruke for å designe brukergrensesnitt. Ved å benytte dette biblioteket finnes det ferdige GUI komponenter for alt som er benyttet i dette prosjektet. Dette gjorde det lettere å designe brukergrensesnittet, samtidig som det ferdige produktet fikk et mer profesjonelt utrykk enn om vi skulle ha utarbeidet alt selv. Referanse [4] viser benyttet bibliotek.
FieldTalk.Modbus.Master(2.11.0)
Dette er et bibliotek Scantrol har kjøpt for implementering av Modbus i .Net applikasjoner. Dette ble benyttet for implementering av modbus UDP kommunikasjon mot PLSen. Her er brukergrensesnittet klient og PLS server. Bakgrunnen for valget av Modbus UDP i stedet for TCP er at det gir en mye raskere kommunikasjon. Dette er en fordel når vi skal motta livedata for plotting og sende loggfil data, som ofte kan være store filer. Referanse [3] viser benyttet bibliotek.
ScottPlot.WPF(4.1.61)
Scottplott.WPF er et *open source* bibliotek for plotting av data i .NET applikasjoner. Dette biblioteket ble benyttet i Plattform UI for å plotte live data fra sylindre/vinsj i tillegg til å vise plot av loggfil data. Referanse [5] viser benyttet bibliotek.



Figur 2.13: UML diagram for Plattform UI brukergrensesnitt.

2.9 Kommunikasjon

PLS og HMI kommuniserer over nettverk med modbus UDP-protokoll, hvor PLS ble valgt som server og HMI som klient. Protokolldokument for kommunikasjonen mellom HMI og PLS er beskrevet i vedlegg 11.1. UDP er en raskere og mer uavhengig protokoll enn TCP, ettersom UDP ikke har mekanisme for å sjekke om sendte datapakker er fullstendige. Det ble derfor valgt å benytte UDP-kommunikasjon for å gjøre sendingen av loggfildata fra HMI til PLS raskest mulig.

2.10 Operasjonsmodus plattform

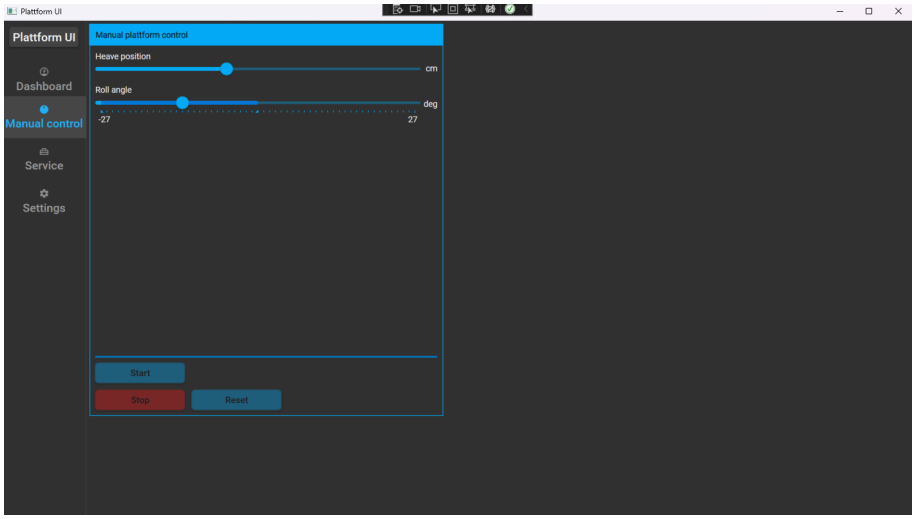
Oppdragsgiver hadde ønske om å få implementert en programvare for å kunne styre bevegelsesplattformen i ulike bevegelser. Som følge av dette ble det utviklet tre forskjellige moduser i HMI-brukergrensesnittet; manuell kjøring, simulerte parametriserte bølger og en loggfil-modus hvor plattformen leser og følger bevegelser fra bølgefiler. Bruksanvisning for betjening av HMI-brukergrensesnittet er dokumentert i vedlegg 13.

2.10.1 Reset-kommando

For å sikre seg at systemet til enhver tid vet korrekt posisjon på hver sylinder, ble det implementert en reset-kommando. Denne kommandoen kan aktiveres ved å trykke på en reset-knapp nederst i HMI, dette er kun mulig dersom plattformen allerede er i stopp-modus. I reset-modus kjøres alle sylindrene ned til bunnposisjon og posisjonsmålingen resettes til 0[m] for alle sylindrene. Det ble implementert en funksjon som gjør at alle sylindrene lander samtidig i bunnposisjon selv om plattformen står i en vinkel før reset-knappen blir aktivert. Dette gjøres ved at den eller de sylindrene som har lengst avstand til bunn kjører med en fast hastighetskommando på ca. 2.3[V] som tilsvarer omtrent $\frac{1}{2}$ makshastighet. Den eller de resterende sylindrene beregner hvor mange [m/s] den/de må kjøre for at alle sylindrene skal nå bunnposisjon samtidig. Denne hastigheten kan sendes inn som argument i feedforward-funksjonen for den gitte sylinderen, for så å gå i ønsket hastighet. Sylindrene kjører ned helt til PLSen merker at nåværende enkoderverdi er lik forrige rundes enkoderverdi, videre slår PLSen automatisk av enable releene og ber HMI om å sette systemet i stopp.

2.10.2 Manuell modus

2.10.2.1 Brukergrensesnitt Figur 2.14 viser skjermbilde av brukergrensesnittet i manuell modus. Her kan bruker betjene og kjøre plattformen til ønsket posisjon og vinkel. *Heave position* og *Roll angle* skalerer seg etter plattformens fysiske begrensninger. Minimum og maksimum verdiene på «*Roll angle*» skaleres automatisk til størst mulig vinkel samtidig som brukeren endrer på verdien for «*heave posisition*». Størst mulig vinkel vil eksistere ved 15[cm] *heave position* (midtposisjon plattform). Det ble valgt å skalere ned min/maks roll angle for å forhindre at brukeren kan sette inn ulovlige verdier som plattformen fysisk ikke kan oppnå.



Figur 2.14: Skjermbilde av UI brukergrensesnitt for manuell modus.

Formel 8 og 9 viser beregninger av min/maks *roll angle* basert på en gitt *heave position* når valgt *heave position* er under 15[cm], ettersom det da er bunnen av sylinderutslaget som er begrensende faktor. Det er benyttet 2° margin for min/maks vinkel, dette gir en maks vinkel på 42° ved 15[cm] *heave position*.

- CC = 0.425[m]
- h – *Heave position* [m].
- Max° – Maksimal *roll angle* [°] ved gitt *heave position*.
- Min° – Minimal *roll angle* [°] ved gitt *heave position*.

$$Max^\circ = \sin^{-1}\left(\frac{h \cdot 2}{CC}\right) - 2 \tag{8}$$

$$Min^\circ = -Max^\circ \tag{9}$$

Dersom *heave position* er over 15[cm] er det fullt sylinderutslag (0.3[m]) som er begrensende faktor for *roll angle*. Dermed blir beregningen for min/max *roll angle* som vist i formel 10 og 11.

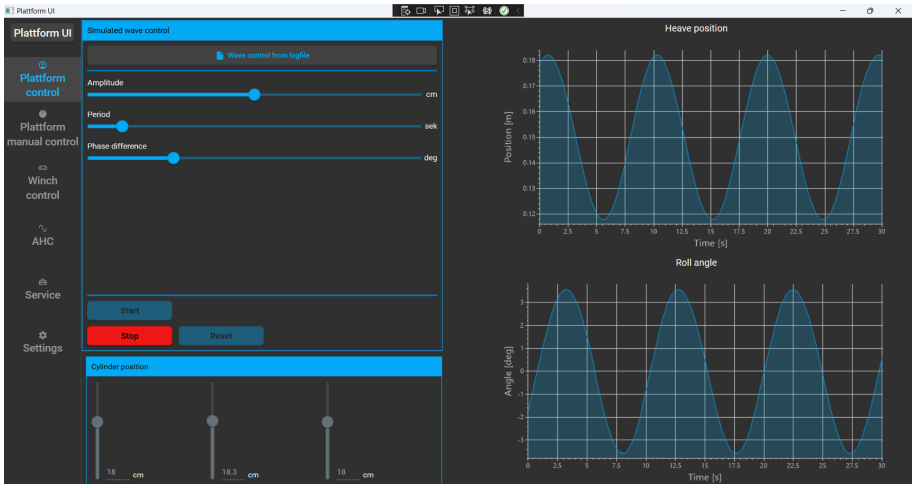
$$Max^\circ = \sin^{-1}\left(\frac{(0.3 - h) \cdot 2}{CC}\right) - 2 \tag{10}$$

$$Min^\circ = -Max^\circ \tag{11}$$

2.10.2.2 Styring I manuell modus benyttes feedforward funksjonene i PLSen for å sørge for at plattformen oppnår ønsket posisjon og vinkel til samme tid. Dette er samme logikk som benyttes for resetting av plattformen hvor den eller de sylindrene som har lengst «kjørevei» beveger seg med en konstant fart, og den/de sylindrene med kortest «kjørevei» benytter en utregnet hastighetskommando. Vedlegg 9 viser spilleliste som inneholder demonstrasjon av manuell modus til bevegelsesplattformen.

2.10.3 Simulert bølgemodus

2.10.3.1 Brukergrensesnitt I denne modusen følger bevegelsesplattformen en simulert sinus. Brukergrensesnittet i figur 2.15 gir bruker mulighet til å endre verdier for amplitude[cm], periode[s] og faseforskyvning[°]. «*Cylinder position*» fanen nede til venstre viser er-verdien[cm] for sylindrene, grafene viser *heave position*[m] og *roll angle*[°] med hensyn på tid[s].



Figur 2.15: Skjermbilde av UI brukergrensesnitt for simulert bølgemodus.

2.10.3.2 Styring Det ble implementert en funksjonsblokk i PLSen «*fbWaveGenerator*» som genererer sinus-posisjonssettpunkt for P-regulatoren og cosinus hastighetssettpunkt for feedforward regulatoren, beregnet fra ønskede inputverdier for amplitude og periode. Formel 12 viser hvordan hastighetssettpunktet beregnes basert på å derivere posisjonssettpunktet.

- A – Amplitude i meter.
- P – Periode i sekunder.
- x – Tidsskritt i sekunder.

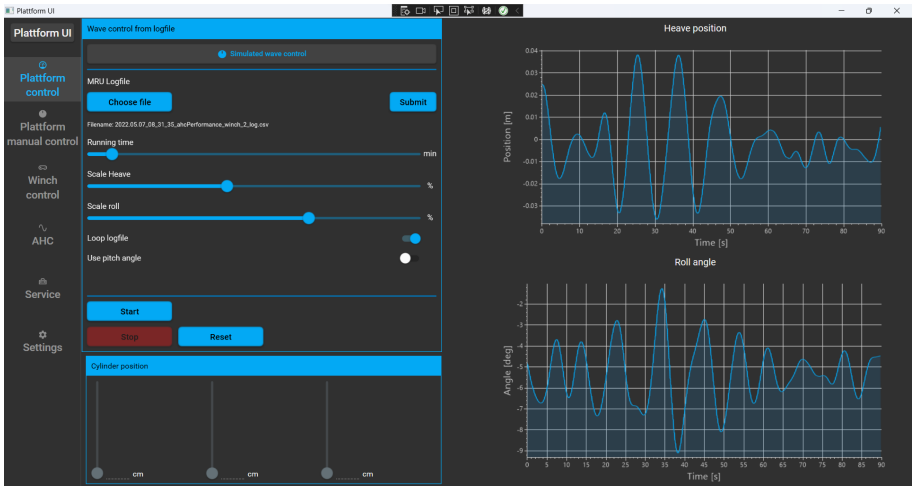
$$\frac{d}{dx}(A \cdot \sin(\frac{2\pi}{P} \cdot x)) = \frac{2\pi \cdot A \cdot \cos(\frac{2\pi}{P} \cdot x)}{P} \tag{12}$$

Før brukeren trykker start må den velge amplitude, periode og en faseforskyvning i grader. Amplituden og perioden er direkte relatert til den simulerte sinus bølgen sin amplitude og periode. Faseforskyvningen sier hvor mange grader settpunktet til sylinder 1 og settpunktet til sylinder 2 og 3 er faseforskyvet i forhold til hverandre. Dersom denne er satt til 0° vil alle sylindrene gå likt, systemet vil da ikke ha *roll angle*, men kun endring i *heave position*. Dersom brukeren velger 180° vil fremre og bakre sylindrene gå motsatt av hverandre og plattformen vil kun ha *roll angle*.

Når brukeren trykker start i simulert bølgemodus kjører plattformen først med jevn hastighet til senterpunkt for bølgen som er 15[cm], før «*fbWaveGenerator*» blokken starter å generere posisjon-/hastighetssettpunkt for alle sylindrene basert på oppgitt amplitude, periode og faseforskyvning. Vedlegg 9 viser spilleliste som inneholder demonstrasjon av simulert bølgemodus til bevegelsesplattformen.

2.10.4 Loggfilmodus

2.10.4.1 Brukergrensesnitt Loggfilmodusen i figur 2.16 gir bruker mulighet til å kjøre plattformen etter bølgefiler i csv-format. Ved hjelp av skyveknappene er det mulig å velge kjøretid i minutt og skalere *heave position* og *roll angle*. Vippebryterne gir bruker mulighet til å *loope* loggfilen, i tillegg til å velge og benytte *pitch angle* fra bølgefilene istedenfor *roll angle*.



Figur 2.16: Skjermbilde av UI brukergrensesnitt for loggfilmodus.

2.10.4.2 Beregninger Beregninger av sylinderposisjons- og hastighetsettpunkt blir utført i HMI før dataene sendes til PLSen. I HMI blir sylinderposisjoner for hvert punkt regnet ut og automatisk skalert ned til plattformens fysiske begrensinger for posisjon og hastighet for sylindrene. Her blir *heave position* skalert mens *roll/pitch* beholder verdiene fra filen.

Hastighetssettpunkt for hver sylinder blir beregnet ved å derivere det allerede beregnede posisjonssettpunktet for hver sylinder. Beregningen utføres ved bruk av formel 13 for numerisk derivasjon i diskret tid. Der x er posisjonssettpunkt [m] og t er tid [s].

$$\dot{x}_t = \frac{x_t - x_{t-1}}{\Delta t} \tag{13}$$

2.10.4.3 Skalering Loggfilene som benyttes i denne modusen er MRU loggfiler som er samplet med 100[ms] intervall i .csv format. Når brukeren trykker «choose file» leses filen og følgende data hentes ut for styring av plattformen:

- *Heave position* [m].
- *Roll angle* [°].
- *Pitch angle* [°].

Skyveknapp «Running time» sin maksverdi er automatisk skalert ned til maks 25 minutter, eller den eventuelle lengden på loggfilen så lenge den er mindre enn 25 minutter. Dette er som følge av begrensningene til hvordan sending av loggfildata til PLS er implementert. Fra HMI sendes sylinderposisjoner og hastigheter for hvert punkt i filen, dette tilsvarer fire verdier per punkt (100[ms]). Modbus har 65535, 16-bit registre som HMI kan plassere data i. For hvert 100[ms] er det benyttet fire 16-bit adresser som gir en maks tid på omtrent 25 minutter. Når brukeren trykker start, skrives alle loggfildataene til modbus sitt «*holding registers*». Tilbakemelding til bruker med status på datasendingen blir gitt i HMI via en *progressbar* ettersom dette er en tidkrevende prosess.

Ved hjelp av skyveknapp «Scale heave» kan bruker skalere ned den allerede nedskalerte *heave position*, og dermed skalere opp *roll/pitch angle* ved hjelp av skyveknapp «Scale roll/pitch». Resultatet av skaleringen kan brukeren se på grafene «*Heave postion*» og «*Roll angle*».

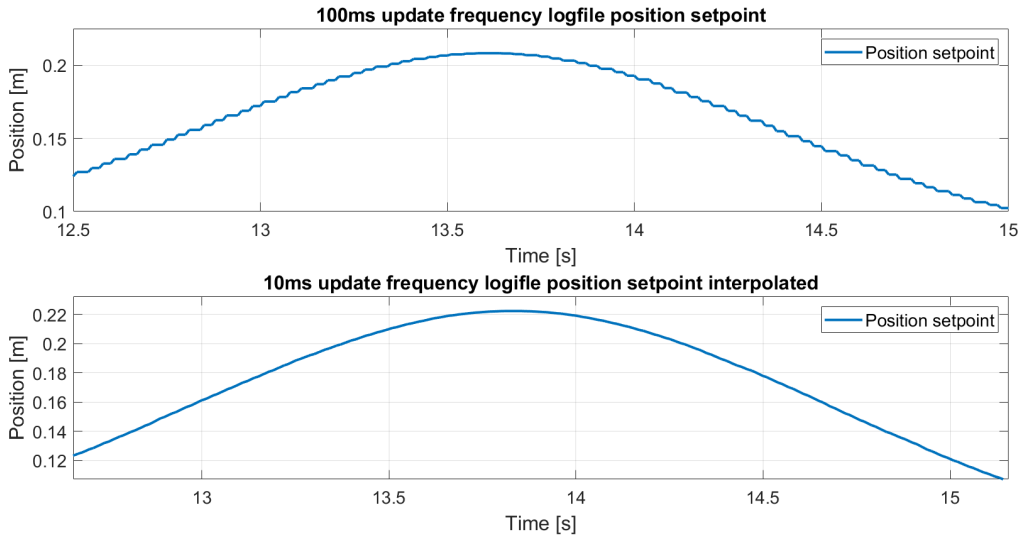
2.10.4.4 Styring Ettersom regulatoren i PLSen kjører med en syklustid på 10[ms] og loggfilene er samplet med 100[ms], er det behov for å interpolere mellom punktene for å unngå støy. Dersom posisjons- og hastighetssettpunktet hadde blitt oppdatert hvert 100[ms], ville regulatoren sjekket og endret pådraget ti ganger mellom hvert oppdaterte settpunkt. Dette ville medført at regulatorer hadde regulert etter et for sjeldent oppdatert settpunkt, som ville skapt støy i pådragssignalet.

For å unngå denne oppførselen i systemet er det valgt å lineært interpolere mellom punktene for å generere posisjon og hastighetssettpunkt hvert 10[ms]. Formel 14 viser utregningen for interpoleringen av loggfildatane.

- Pos_SP – Interpolert posisjonssettpunkt sylinder [m].
- Pos – Posisjonssettpunkt sylinder [m].
- j – Loop count [0-10].

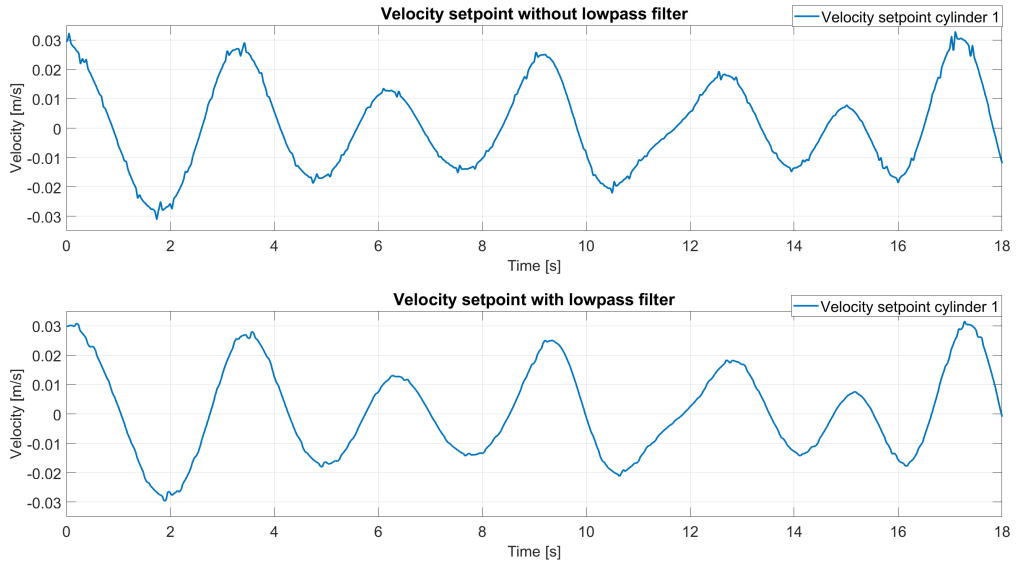
$$Pos\_SP = \frac{(Pos[i+1] + 0.15) - (Pos[i] + 0.15)}{10} \cdot j + Pos[i] + 0.15 \tag{14}$$

Figur 2.17 viser resultatet av interpoleringen av sylinder 1 sitt posisjonssettpunkt. Plottet øverst i figuren viser settpunktet oppdatert med 100[ms] intervall. Plottet nederst i figuren viser settpunktet interpolert og oppdatert med 10[ms] intervall.



Figur 2.17: Plot som viser effekten av interpolerte loggfildata.

Derivasjon av det interpolerte posisjon settpunktet fra loggfildaten forårsaket noe støy i hastighetssettpunktet til feed-forward regulatoren. Det ble dermed implementert ett lavpassfilter i HMI for å redusere støyen. Figur 2.18 viser resultat av hastighetssettpunktet før og etter filtrering av signal.



Figur 2.18: Plot som viser hastighetssettpunkt før og etter lavpassfiltrering i loggfil modus.

Formel 15 er benyttet for å lavpassfiltrere hastighetssettpunktet i HMI. Det er mulig å filtrere ytteligere, men da vil amplituden bli ytterligere dempet i tillegg til at det vil skape større tidsforsinkelser i systemet.

$$x[i] = \frac{x[i] + x[i-1]}{2} \tag{15}$$

Vedlegg 9 viser spilleliste som inneholder demonstrasjon av loggfil modusen til bevegelsesplattformen.

2.11 Begrensninger og tester

2.11.1 Posisjonsbegrensninger

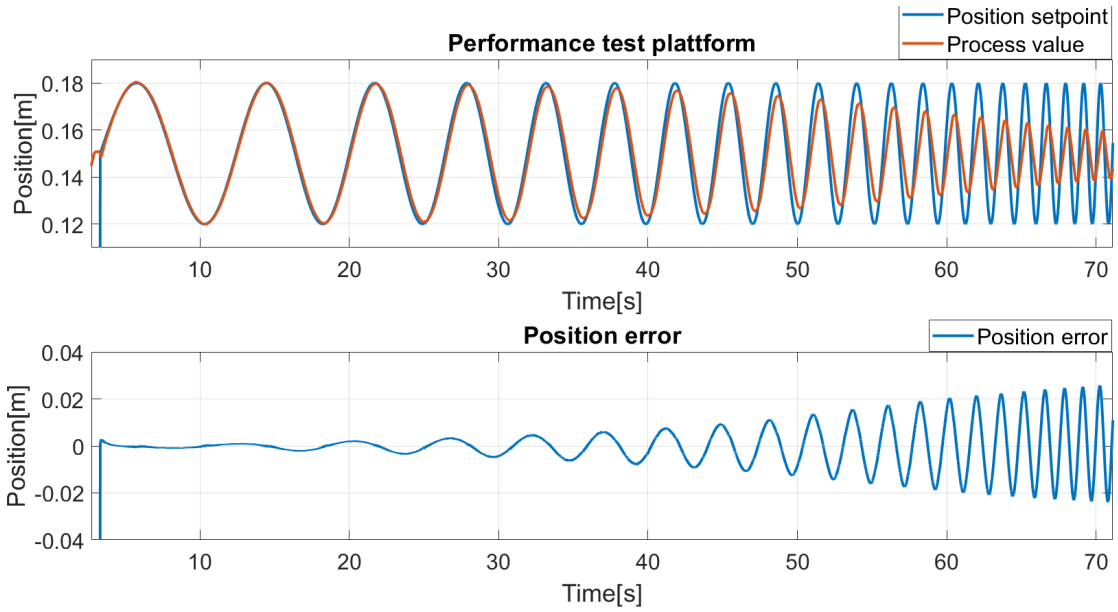
Plattformen har fysiske begrensninger som følge av at sylindrene har et sylinderutslag på 30[cm]. Dette medfører at *heave position* uten *roll/pitch angle* på plattformen vil være mellom 0-30[cm]. Maks vinkel for plattformen kan utføres når plattformen er i sin midtre posisjon som er 15[cm].

2.11.2 Hastighetsbegrensninger

Under hastighetskalibreringen av sylindrene i kapittel 2.4.2.2 ble hastighetene ved maks hastighetskommandoer ±10[V] målt, resultatene viste at sylinder 1, som hadde mest tilført last, maksimalt klarte å kjøre med hastighet -0,0469[m/s] utover. Sylinder 3, med minst tilført last, klarte maksimalt å kjøre med hastighet 0,0490[m/s] innover. Dette er valgt som hastighetsbegrensninger for alle sylindrene i systemet.

2.11.3 Begrensningstest

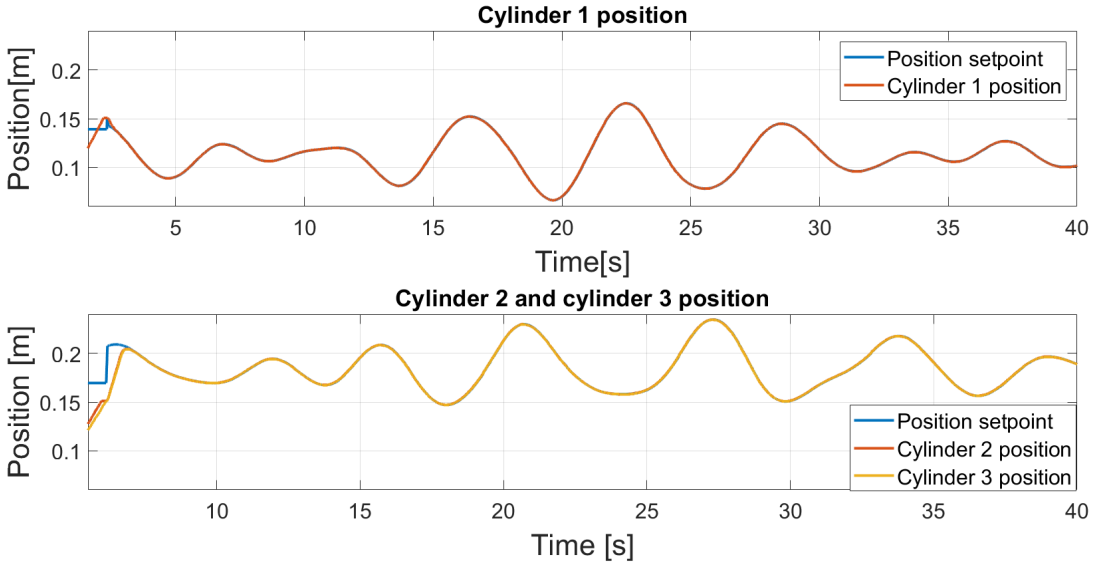
Systemets begrensninger knyttet til hvor godt det følger posisjonssettpunktet til en generert sinus med økende frekvens er testet i figur 2.19. Testen viser hvordan bevegelseplattformen følger posisjonssettpunkt for en generert sinus med amplitude på 0.06[m] og minkende periodetid fra 10[s] til 1[s]. Resultatet av testen viste at sylindrene sliter med å nå topp- og bunnpunkt fra Ca. 4[s] periodetid og lavere.



Figur 2.19: Øverste graf viser hvordan bevegelsesplattformen følger posisjonssettpunkt for en sinus som har økende frekvens fra 10[s] til 1[s] periode. Nederste graf viser hvordan posisjonsfeilen mellom er-verdi og settpunkt øker desto høyere frekvens.

2.11.4 Loggfilmodus test

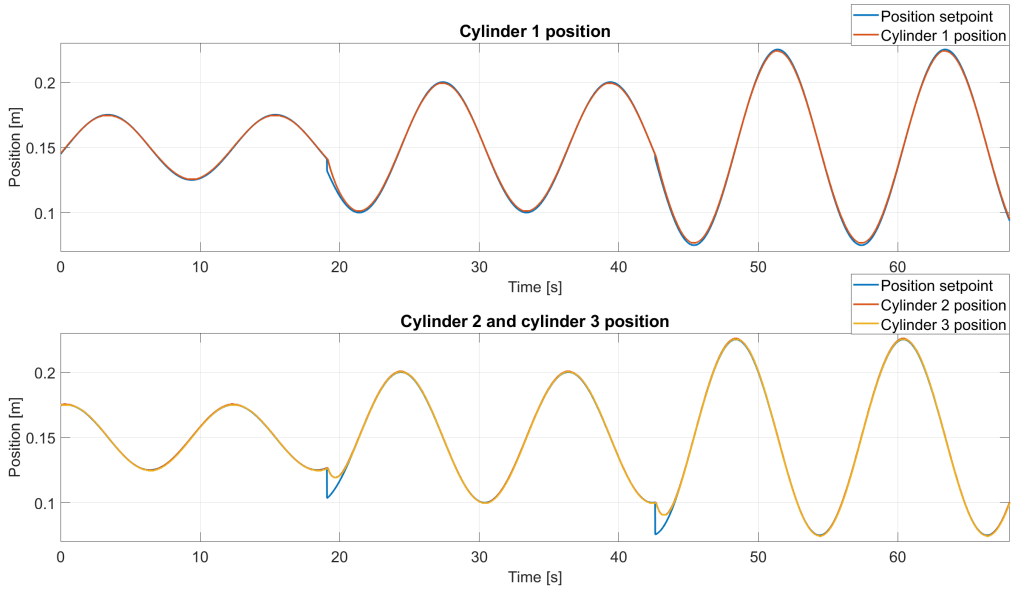
Det er utført en rekke tester på plattformen hvor den følger bevegelsene fra ulike loggfiler. Figur 2.20 viser hvordan sylinder 1, 2 og 3 følger de nedskalerte bølgebevegelsene på en av disse filene etter utført optimalisering.



Figur 2.20: Plot som viser sylinderposisjoner sammenlignet med posisjonssettpunkt for alle tre sylindre i loggfil modus.

2.11.5 Simulert bølgemodus test

Figur 2.21 viser hvordan sylindrene på plattformen følger simulerte sinus bølger med 90°faseforskyvning mellom fremre og bakre sylindre. Periodetiden er fastsatt til 12[s], amplituden varierer med to stk bølgetoppers intervall fra 5,10 og 15 [cm]. Hoppene i posisjonssettpunkt ved omtrent 20 og 45 sekunder er grunnet brå endring av amplitude til sinusgeneratoren i PLSen.

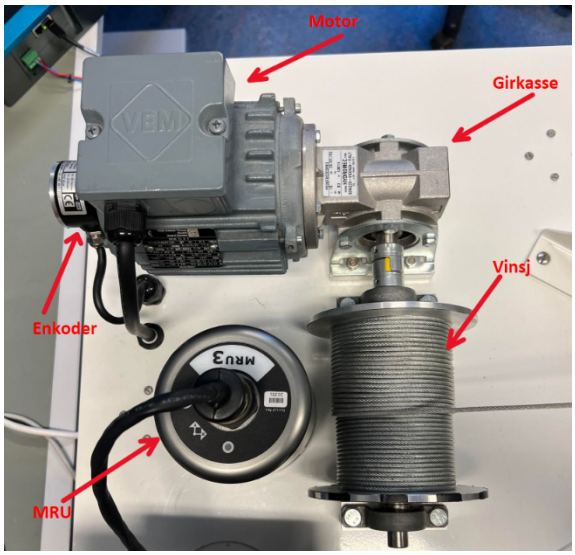


Figur 2.21: Plot som viser posisjon settpunkt og faktisk posisjon for sylinder 1, 2 og 3 ved forskjellige amplituder.

3 Vinsj styring

3.1 Hardware

Del 2 av oppgaven omhandler styring av den fastmonterte vinsjen på bevegelsesplattformen. Figur 3.1 viser bilde av de synlige komponentene tilhørende denne delen. Motoren sin hastighet blir styrt av en frekvensomformer som er vist på samme tilkoblingsbrett som *hardware* komponentene i del 1 av prosjektet 2.11. Enkoderen som vist i figur under, er fra Scancon og har 1024 pulser per rotasjon. MRU-en til venstre på bildet er av type seatex mru5. Den er fastmontert på plattformen, men benyttes ikke i dette prosjektet.



Figur 3.1: Bilde som viser de synlige hardware-komponentene til vinsjstyringen.

3.1.1 Frekvensomformer

Det benyttes en frekvensomformer fra Vacon NXP for styring av hastigheten til vinsjen. Frekvensomformeren har tilkoblet tre utvidelseskort for enkoder, nødstopp og for fieldbus kommunikasjon.

3.1.2 Elektrisk motor

I dette systemet anvendes det en 3-fase asynkron induksjonsmotor fra «VEM» med følgende spesifikasjoner, se tabell 2.

Motor type	3-fase asynkron induksjonsmotor		
IP	55		
Cos φ	0.74		
Effekt	0.37kW		
Koblingsmetode	Δ		
Spenning	230V		
Strøm	1.84A		
Frekvens	50Hz	Hastighet	1345Rpm

Tabell 2: Tabell som viser motorens spesifikasjoner.

3.1.3 Beregning av vinsjposisjon

I dette systemet er det benyttet en girkasse mellom akslingen på motoren og vinsjen. Girforholdet på girkassen er $\frac{1}{10}$, der 10 rotasjoner på motoren tilsvarer 1 rotasjon på vinsjen. Det er ikke tatt hensyn til at vinsjen kjører på flere lag under beregning av omkretsen på trommelen, ettersom at vinsjen kun kjøres på ett lag. Det ville vært nødvendig å ta dette med i beregningene dersom vinsjen hadde hatt en lengre kjørelengde. Kodeutsnitt 4 viser utregning av posisjonen til vinsjen gitt i [m] ved bruk av enkoderpulser, omkretsen til trommelen, kjente pulser per rotasjon og girforholdet:

```
///*************************************************************************
//Code information:
//Function block that calculates winch position in meter, using encoder pulses.
//GearRatio = 10
//iPPR = pulses per rotation
//
//Author: Sander V. Andersen and Lars Askild S. Aarvik
//Date: 13.04.2023
//*************************************************************************/

rWinchPos                       :=  TO_REAL(diEncoderWinch) * ( rCircumference/ (iPPR*
    rGearRatio*4)  ) * -1; // position in [m]
```

Listing 4: Funksjonsblokk som beregner vaierlengden basert på enkoderpulsene fra motorene.

3.1.4 Operatørpanel

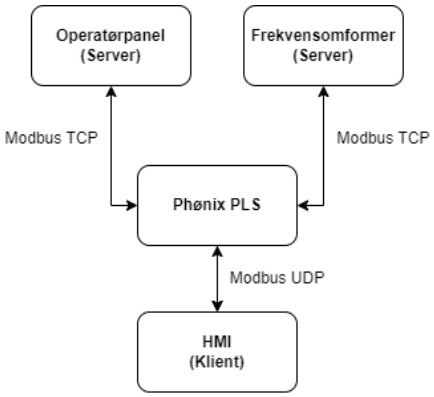
Figur 3.2: Betjeningspanel for styring av vinsj.

Betjeningspanelet vist i figur 3.2 er programmert og implementert i PLS til å styre vinsjen på bevegelsesplattformen. Det består av følgende komponenter:

- **Buzzer:** Benyttes for alarmsignallyd ved feil og under nødstopp.
- **Reset-knapp:** Resetter feilmeldinger på frekvensomformer.
- **Standby-knapp:** Aktiverer standby-modus.
- **InControl-knapp:** Ikke implementert.
- **Manual-knapp:** Aktiverer mulighet for å kjøre vinsj manuelt med potmeter og joystick.
- **AHC-knapp:** Aktiverer AHC-modus for vinsj.
- **Joystick:** Styrer vinsj innover og utover gradvis, hvor maks utslag tilsvarer maks hastighet valgt på potmeter.
- **Potmeter:** Justerer hastigheten til vinsj 0-100[%].

3.2 Kommunikasjon

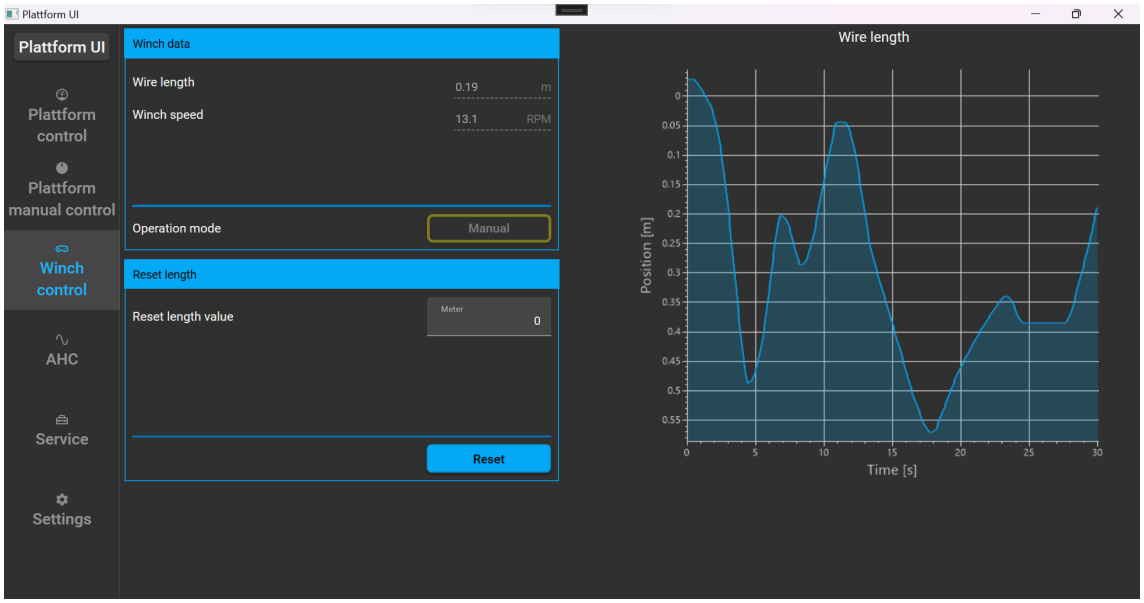
PLS, operatørpanel og Vacon frekvensomformeren kommuniserer over modbus TCP. Figur 3.3 viser oversikttegning over kommunikasjonen mellom de forskjellige enhetene. Vedlegg 11.2 viser implementert kommunikasjonsprotokoll for frekvensomformer. Vedlegg 11.3 viser implementert protokoll for operatørpanelet.



Figur 3.3: Blokkskjema som viser kommunikasjonen mellom operatørpanel, frekvensomformer, PLS og brukergrensesnitt.

3.3 Brukergrensesnitt

For monitorering og resetting av vaierlengde er det implementert en egen tab i HMI for «Winch control». Figur 3.4 viser et skjermbilde av denne siden i HMI.

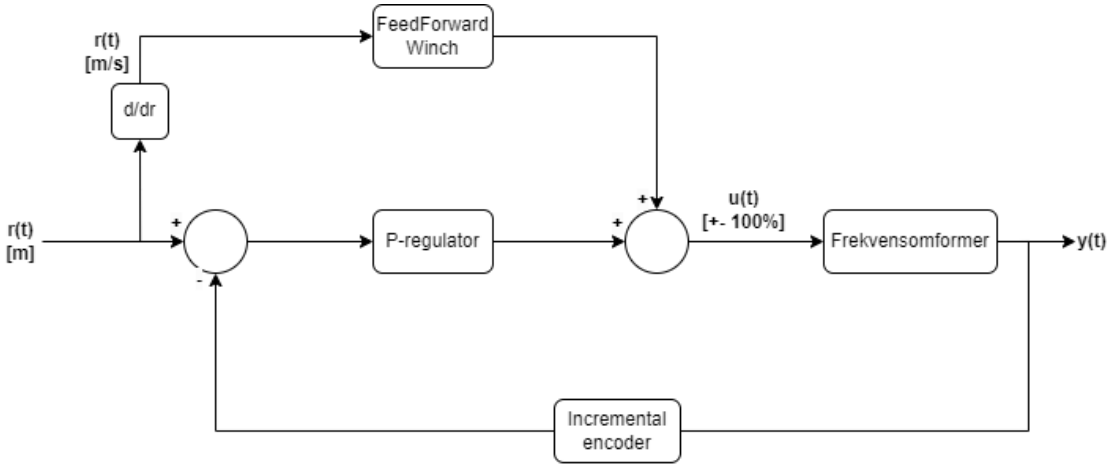


Figur 3.4: Viser skjermbilde av «Winch control» tab i HMI.

- **Wire length:** Viser vaier lengde ute i meter.
- **Winch speed:** Viser hastigheten på vinsjen i RPM.
- **Operation mode:** Viser nåværende operasjonsmodus for vinsjen med tekst og farge.
- **Reset length value:** Lengde brukeren ønsker å resette vaierlengde til.
- **Reset:** Knapp for å resette vaierlengde.
- **Wire lenght graph:** Viser direkte oppdatering av vaierlengde grafisk.

3.4 Posisjons- og hastighetsregulering av vinsj

Vinsjen reguleres etter posisjon[m] og hastighet[m/s] med tilsvarende reguleringssløyfe som for sylindrene vist i figur 2.10, der en P-regulator regulerer etter posisjon og en feedforward-regulator regulerer vinsjen basert på ønsket hastighet. Blokkskjema 3.5 illustrerer reguleringssløyfen som er optimalisert og beskrevet videre i dette kapittelet.



Figur 3.5: Reguleringssløyfe for posisjon og hastighetsregulering av vinsj.

3.4.1 Hastighetskalibrering og tuning av vinsj

Hastighetskalibrering av vinsjen ble gjennomført med utviklet funksjonsblokk «*fbSpeedTestWinch*» i PLSen for å finne ut hvor mange [RPM] vinsjen gir med pådrag fra 0-100% i hver retning. Funksjonsblokken er vist i kodeutklipp 5. Funksjonsblokken kjører vinsjen automatisk i 10 steg. Hastigheten øker fra 0-100% (maks hastighetskommando). På hvert steg kjører den i tre sekund for å få tid til å rampe seg opp, før den *sampler* [RPM] ved denne hastighetskommandoen. Funksjonsblokken returnerer deretter en tabell med hastigheter gitt i [RPM]. Formel 16 ble deretter benyttet for å omregne hastighetene i [RPM] til [m/s].

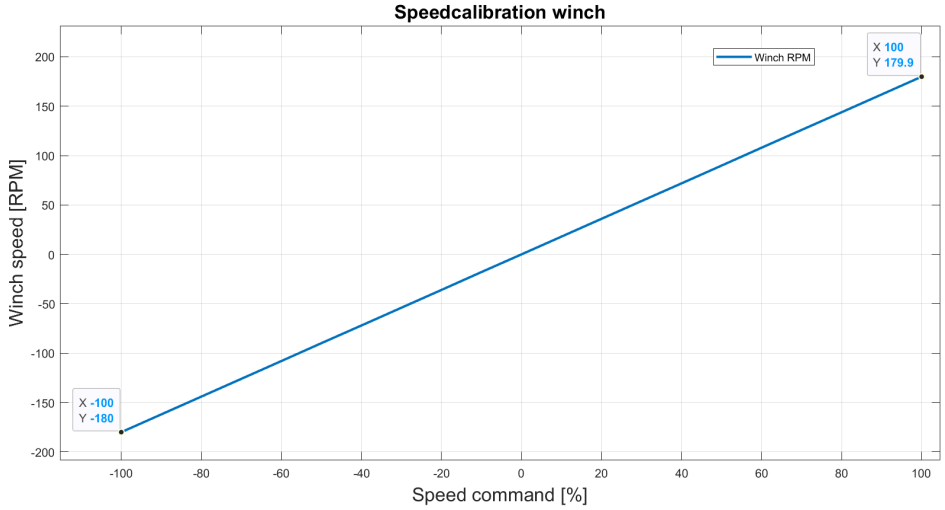
$$V[m/s] = \frac{\text{Omkrets[m]}}{60[s]} \cdot \text{RPM} \tag{16}$$

```
///************************************************************************
//Code information:
//Function block that automatic speedtests the winch, with ramping signals [0-100]%
//Author: Sander V. Andersen and Lars Askild S. Aarvik
//Date: 27.04.2023
//************************************************************************/

IF(xActivate)THEN
    IF(NOT(xTestDone))THEN
        xRun                        := TRUE;
        wTorqueCMD                  := 3000;
        xDir                        := xDirection;

        If(iTimerCount < iTestTime)THEN
            iTimerCount             := iTimerCount + 1;
        ELSE
            IF(wSpeedCMD <> 0)THEN
                rSpeedArray[uiSpdCmdLocal/1000]       := iRPM / rGearRatio;
                END_IF;
            uiSpdCmdLocal           := uiSpdCmdLocal + TO_UINT(rSpdStep);
            wSpeedCMD               := TO_WORD(uiSpdCmdLocal);
            iTimerCount             := 0;
            IF(wSpeedCMD > 10000)THEN
                xTestDone           := TRUE;
                END_IF;
            END_IF;
     ELSE
        xRun                        := FALSE;
        wTorqueCMD                  := 0;
        wSpeedCMD                   := 0;
        END_IF;
ELSE    // Reset all variables

    wSpeedCMD                       := 0;
    wTorqueCMD                      := 0;
```

```
36      xRun                            := FALSE;
37      rSpdStep                        := 10000 / iSteps;
38      iTimerCount                     := iTestTime + 1;
39      uiSpdCmdLocal                   := 0;
40      xTestDone                       := FALSE;
41
42      FOR i := 0 TO iSteps DO
43      rSpeedArray[i]                  := 0;
44      END_FOR;
45      END_IF;
```

Listing 5: Funksjonsblokk som automatisk hastighetskalibrerer vinsjen.

Figur 3.6 viser grafisk fremstilling av hastighetskommando med tilhørende hastighet for vinsjen. Elektriske motorer er veldig lineære, som er tydelig i figur 3.6. Hadde dette vært en hydraulisk vinsj ville det vært mye viktigere med en god kalibrering, gjerne med enda flere punkter i hver retning, da de ikke har like lineær karakteristikk. Figur 3.6 viser at vinsjen har en maks hastighet på 180 [RPM] i «*pay out*» retning og 179.9 [RPM] i «*pay in*» retning.



Figur 3.6: Plot som viser hastighetskalibrering av vinsj, resultatet viser at den elektriske motoren er lineær.

3.4.2 Vektorkontroll

Frekvensomformeren styres med *speed-control* med *torque limitation* og har to forskjellige kontroll metoder; *open-loop* og *closed-loop*.

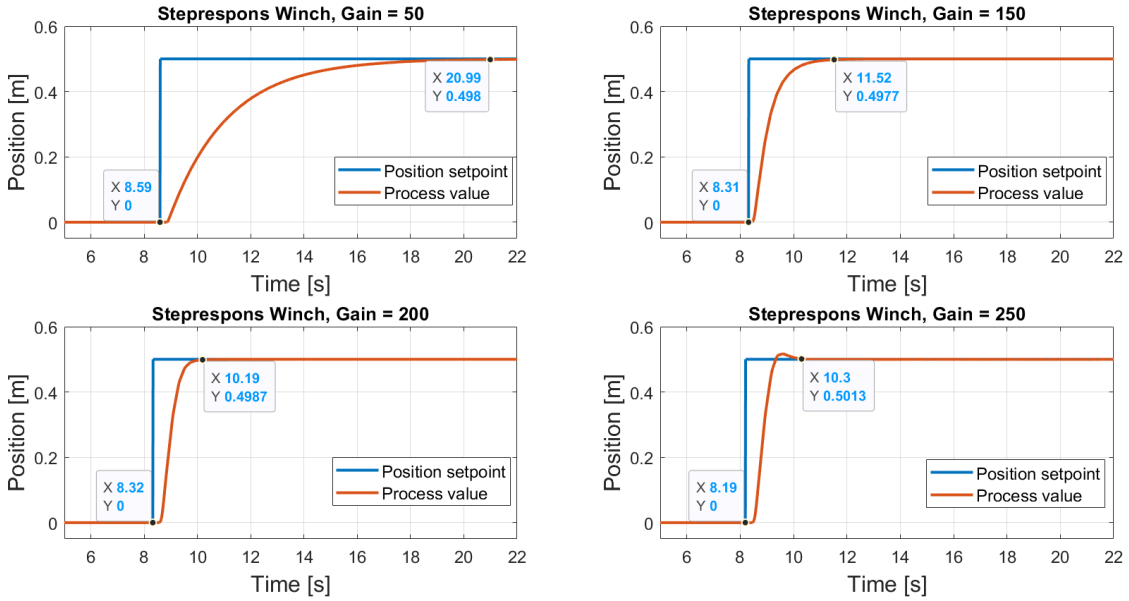
Open-loop baserer seg på å kjøre omformeren uten tilbakemelding fra sensorer. Den styres da med en simulert matematisk modell av systemet internt i omformeren. En nøyaktig regulering med denne metoden forutsetter at den fysiske modellen og den simulerte er tilnærmet like.

Closed-loop kontroll har hele tiden tilbakemelding om posisjonen og hastigheten sin ved å overvåke pulsene fra tilkoblet enkoder[8]. Ettersom denne metoden gir mer nøyaktig hastighet benyttes denne metoden for styring av vinsjen.

Momentgrensen til frekvensomformeren er innstilt til 30%. Denne er begrenset slik at den tillater vekten av loddet på vinsjen, men vil begrense strømmen til motoren ved en høyere tilført belastning.

3.5 Tuning av vinsj

Kalibreringen av vinsjen ble utført ved å kjøre vinsjen med ulike steps over en distanse på 50[cm], med ulike forsterkninger gitt i [%/m]. Plot øverst til venstre i figur 3.7 viser step respons med 50[%/m], resultatet viser at vinsjen klarer å nå settpunkt, men for sakte i forhold til hva som er ønskelig. Plot øverst til høyre viser samme test med tilført forsterkning på 150[%/m], dette ga en raskere stigetid uten oversving. Neste plot nederst til venstre viser step med forsterkning 200[%/m], dette er forsterkningen som har regulert systemet til raskest stigetid uten oversving. Plot nederst til høyre viser at systemet får en liten oversving med forsterkning 250[%/m]. Det ble valgt å benytte 200[%/m] forsterkning da det ga raskest stigetid uten oversving.



Figur 3.7: Plot som viser stepresponser av vinsj med ulike forsterkninger. Stegene går fra 0[m] til 0.5[m].

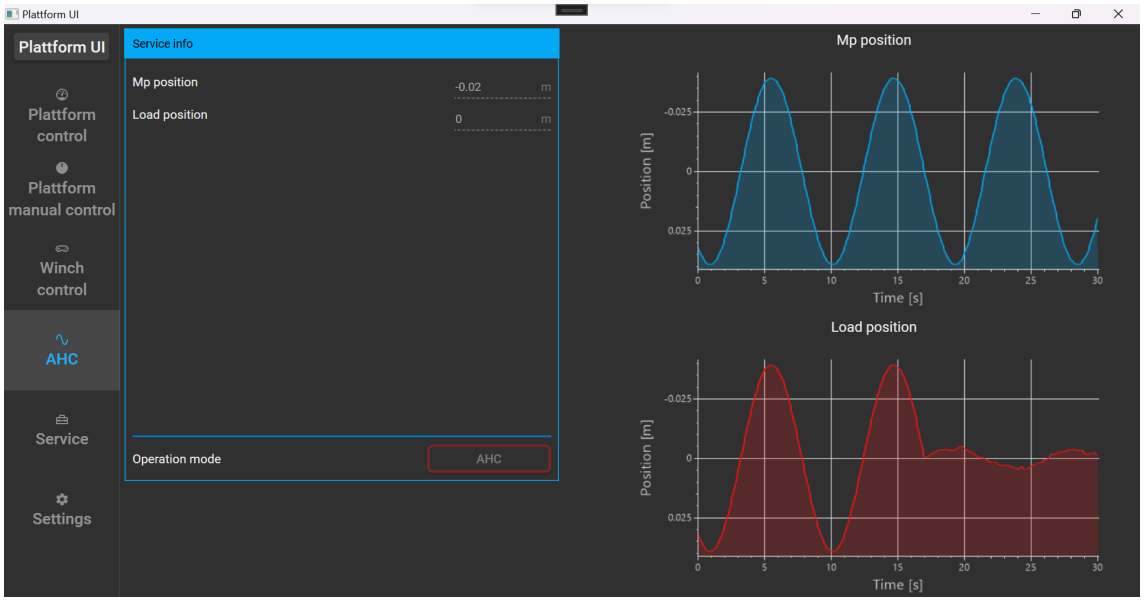
4 Evaluering av metoder for aktiv hiv kompensering

4.1 Ramping av vinsj

For å kunne benytte vinsjen til å kjøre AHC, er det viktig at den er rask og presis. Dette er vanskelig å få til dersom det er konfigurert rampetid på motoren. Rampetiden ble derfor satt til 0[s] i parametrene i frekvensomformeren for å unngå at vinsjen ble for treg til å nå ønsket hastighet. Ved å sette rampetiden til 0[s] i frekvensomformeren blir vinsjen veldig direkte, og kan være farlig å operere i manuell modus. Det ble dermed implementert egen funksjonsblokk i PLS «fbRamping_V1_001» for å håndtere dette. Her benyttes det 2[s] rampetid for at vinsjen skal rampe opp og ned roligere og mer kontrollert under operasjon av vinsj med *joystick*.

4.2 Brukergrensesnitt

For monitorering av ytelsen til AHC systemet er det implementert en egen tab i HMI «AHC». Figur 4.1 viser et skjermbilde av denne siden i HMI.



Figur 4.1: Skjermbilde av «AHC» tab.

- **Mp position:** Viser posisjonsendringen til krantuppen i meter.
- **Load position:** Viser posisjonsendringen til lasten i meter.
- **Operation mode:** Viser nåværende operasjonsmodus for vinsjen med tekst og farge.
- **Mp position plot:** Viser direkte oppdatering av Mp posisjon grafisk.
- **Load position plot:** Viser direkte oppdatering av posisjonensendringen til lasten grafisk. Denne skal optimalt være lik 0[m] ved en ideell AHC.

4.3 Implementasjon av AHC

Ettersom AHC går ut på å motvirke bølgebevegelsene til fartøyet med vinsjen, må systemet vite hvor mye og hvor fort det skal utføre motvirkningen. Ettersom at det er bevegelsen til lasten vi ønsker å motvirke, må systemet vite hvor mye bølgene påvirker akkurat lasten. Denne bevegelsen benevnes som *measuring point* (Mp), og er *heave* bevegelsen for krantuppen. For å benytte AHC, må systemet bergene eller måle denne bevegelsen. I dette prosjektet ble det utført enten ved å beregne Mp posisjonen basert på sylinderposisjoner eller ved å måle bevegelsen ved hjelp av stereokamera.

Ved å invertere Mp posisjonen får systemet et posisjonssettpunkt til vinsjen som benyttes for å motvirke bølgebevegelsen. Det er ikke ønskelig å kun regulere vinsjen basert på et posisjonssettpunkt da vinsjen vil henge etter settpunktet i tillegg til og kunne få et stasjonært avvik. Derfor

benytter vinsjen samme reguleringsmetode som for sylindrene, ved å kombinere posisjonsregulator og feedforward hastighetsregualator. Hastighetssettpunktet beregnes ved å derivere Mp posisjonen, dette gir en hastighet i [m/s] som vinsjen må rotere for å motvirke bølgebevegelsen.

Ettersom systemet beregner Mp posisjonen hvert 10[ms], blir det relativt mye støy i posisjonssettpunktet og enda mer i hastighetssettpunktet. For å unngå for mye støy i pådragsignalet er det benyttet et lavpassfilter for både Mp posisjonen og Mp hastigehten. Her er Mp hastigheten lavpassfiltrert enda mer enn Mp posisjonen ettersom det var mest utsatt for støy. Hvor mye det skulle filteres er basert på prøving og feiling. For mye filtrering skaper for mye tidsforsinkelser i tillegg til demping av amplituden, men for lite vil skape for mye støy i signalene.

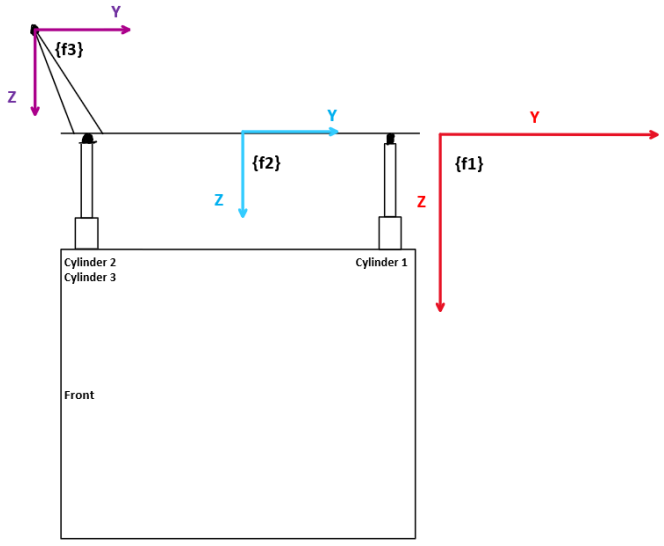
4.4 Teori knyttet til aktuelle metoder for AHC

Dette kapittelet omhandler teorien og beregningene bak de aktuelle metodene for AHC. Opprinnelig i oppgavebeskrivelsen var ønsket fra Scantrol at vi skulle evaluere og benytte mTrack sin egenutviklede AHC-kontroller, AHC ved bruk av radar og AHC ved bruk av stereokamera. Grunnet oppgavens omfang og mangel på tid ble det avklart med Scantrol og veileder Endre Håland at vi heller skulle evaluere og teste metode for «beregnet AHC» og «AHC ved bruk av stereokamera».

4.4.1 Beregnet AHC

Første testmetode for AHC er «beregnet AHC». Siden dette er en bevegelsesplattform og systemet til enhver tid kjenner sylinderposisjonene, er det mulig å beregne hvor mye vinsjen må rotere for å motvirke disse bevegelsene. Denne metoden vil ikke fungere på et ekte fartøy, men det fungerte som et godt referansepunkt for videre tester.

4.4.1.1 Beregning av bevegelse Figur 4.2 illustrerer plattformen med koordinatsystem {f1}, {f2} og {f3} i 2D, der {f1} er referanse koordinatsystem. Ettersom plattformen har to frihetsgrader er det mulig å gjøre en forenkling av utregningene i 2D. I {f2} er heave position og roll angle kjent. For å kunne kjøre beregnet AHC må systemet vite forflytningen til {f3} i Z-retning.



Figur 4.2: Illustrasjon som viser koordinatsystem {f1}, {f2} og {f3} i 2D.

For å beregne denne bevegelsen brukes en 2D transformasjonsmatrise. Ettersom koordinatsystemene ikke blir rotert i forhold til hverandre, ble det ikke behov for å benytte en rotasjonsmatrise. Formel 17, hentet fra [6] viser en 2X2 transformasjonsmatrise for å vite {f3} sin posisjon relativ til {f2} sin posisjon.

$$\begin{bmatrix} Y' \\ Z' \end{bmatrix} = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} Y \\ Z \end{bmatrix} \tag{17}$$

Med bakgrunn i at systemet kun er interessert i bevegelsen til {f3} i Z-retning, brukes kun deler av transformasjonsmatrisen. Formel 18 viser formel for utregning av bevegelsene til {f3} i forhold til {f2}. Her er TipVect forflytningene i Y- og Z-retning fra {f2} til {f3}. TipVect_Z er subtrahert på slutten for å finne forskjell i Z - bevegelse uten offset (TipVect_Z) med i beregningen.

- Mp – Mp posisjon [m].
- TipVect_Y – Lengde i Y-retning fra senter av plattform til krantupp [m].
- TipVect_Z – Lengde i Z-retning fra senter av plattform til krantupp [m].
- α – *Roll angle* [°].

$$Mp = TipVect\_Y \cdot \sin\alpha + TipVect\_Z \cdot \cos\alpha - TipVect\_Z \tag{18}$$

Kodeutsnitt 6 viser implementert funksjon i PLS som beregner Mp posisjon, og definerer hvor mye vinsjen må rotere for å motvirke bevegelsene skapt av sylindrene på plattformen. Mp posisjonen er referansepunktet for AHC i systemet.

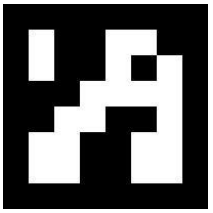
```
1  ///***********************************************************************
2  //Code information:
3  //Function for calculating MpPos.
4  //Takes cylinder 1,2,3 positions as input arguments and returns Mp position with given y and z
       offsets.
5  //
6  //Author: Sander V. Andersen and Lars Askild S. Aarvik
7  //Date: 29.04.2023
8  //***********************************************************************/
9
10 // Calulate heave pos and roll angle for center of plattform
11 rRollAngle                      := ASIN((rCyl1Pos - ((rCyl2Pos + rCyl3Pos) / 2)) / rPlattformCC
       ) * (180 / PI);
12 rHeavePos                       := rCyl1Pos - (((SIN(rRollAngle * (PI / 180))) * rPlattformCC)
       / 2);
13
14 // Calculate crane tip position relativ to plattform and return position relativ to center of
       wave (0.15m)
15 rMpPos                          := (rTipVectY * SIN(rRollAngle * (PI / 180))) + (rTipVectZ *
       COS(rRollAngle * (PI / 180))) - rTipVectZ + rHeavePos - rPlattformCenter;
16
17
18 // Return MpPos
19 fnCalculateMpPos                 := rMpPos;
```

Listing 6: Funksjon som beregner krantupp posisjon basert på sylinderposisjoner i meter.

4.4.2 AHC ved bruk av stereokamera

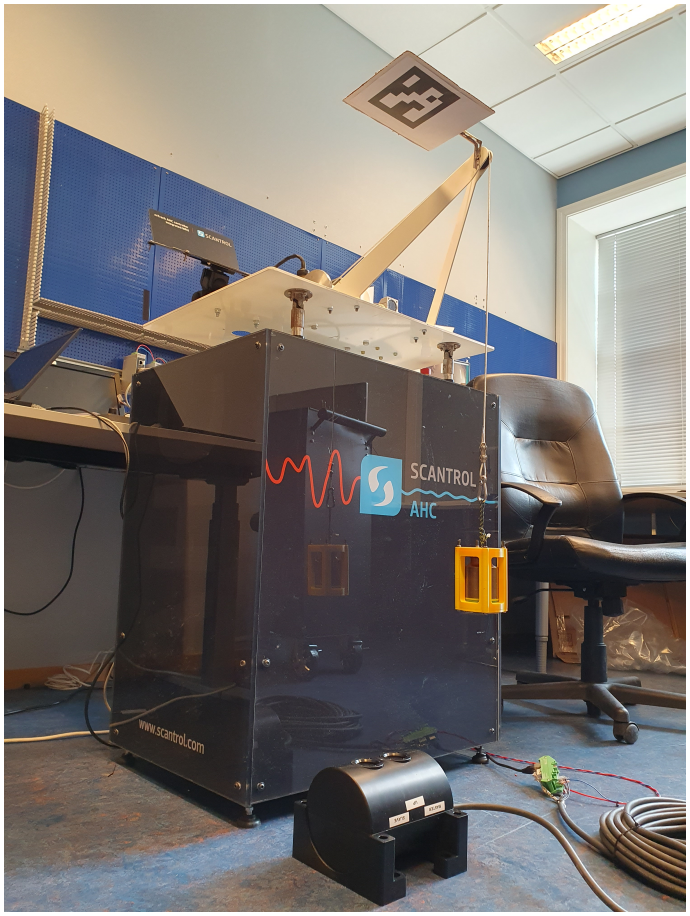
Scantrol har ett egenutviklet stereokamera som er montert i en vanntett kapsling, dette består av to separate objektivlinser og en kraftig liten linux PC som analyserer og sammenligner bildene. Objektivene tar bilde av samme objekt med to forskjellige synsvinkler, resultatet av dette skaper ett stereo bilde. Ved å sammenligne disse to bildene er det mulig å kalkulere informasjon om dybde, rotasjon og bevegelse. Scantrol ønsker å undersøke muligheten for å benytte denne metoden for AHC ved lasting mellom to skip.

4.4.2.1 Aruco-kode Scantrol sitt egenutviklede stereokamera beregner avstand og rotasjon til Aruco-koder. Figur 4.3 viser en Aruco-kode. Aruco-koder er en visuell markør som benyttes til å spore og identifisere objekter, disse kan benyttes til å hente ut informasjon om bildedybde og rotasjon på ønskede objekter[9].



Figur 4.3: Illustrasjon av Aruco-kode.

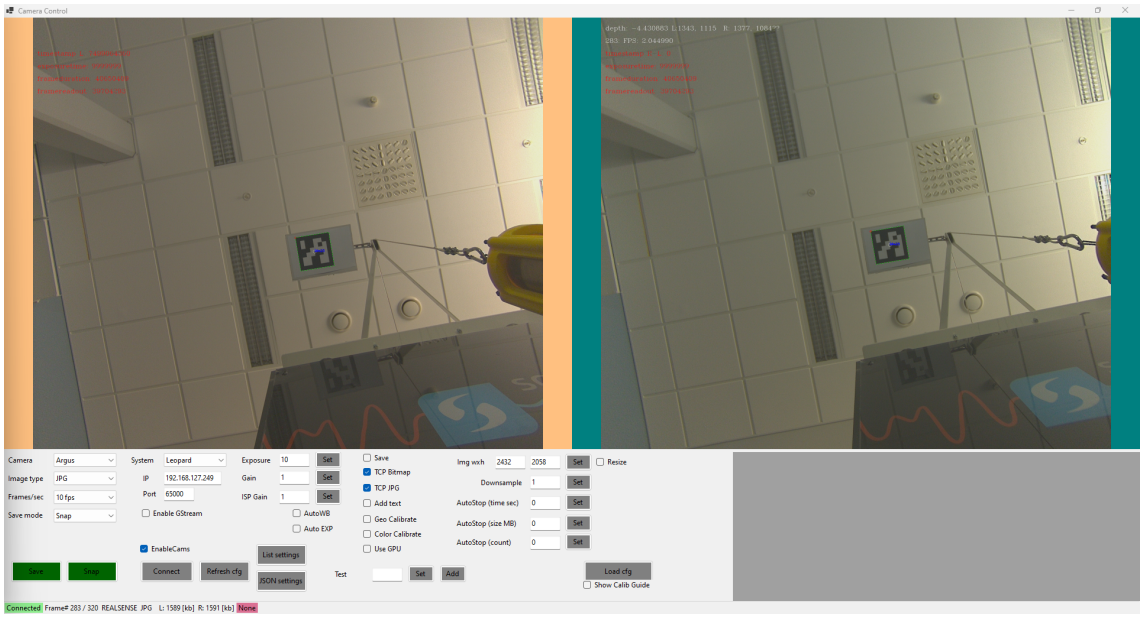
4.4.2.2 Montering For fremtidige systemer ser Scantrol for seg at stereokamera kan være montert i toppen av krantuppen og måle avstanden ned til flaten lasten skal flyttes til. Grunnet vekten av kameraet, er kameraet i dette prosjektet montert på gulvet og Aruco-kode montert i krantupp som gir oss samme dybdeinformasjon. Siden det er krantuppbevegelsen vi ønsker å motvirke i AHC, vil den målte distansen gi oss det vi trenger som referanse til AHC systemet. Figur 4.4 viser bilde av brakett med Arcuo-kode og kamera montert rett under for å måle avstand til koden.



Figur 4.4: Bilde som viser montering av stereokamera og Aruco kode.

4.4.2.3 Kalibrering For at kameraet skal gjenkjenne og bergene korrekt avstand til en Aruco-kodene må det kalibreres. Denne jobben fikk vi hjelp til av en ansatt hos Scantrol. Kalibreringen ble utført ved å ta en serie med bilder av en kalibreringsplate i forskjellige posisjoner og vinkler. Størrelsen og fargen på kalibreringsplaten var kjente faktorer for *softwaren* som ble benyttet til å kalibrere kamera. Ved å kalibrere kamera med hensyn på kalibreringsplaten, skal den etter kalibrering også kunne beregne riktig avstand og rotasjon på Aruco-koder.

4.4.2.4 Tracking Scantrol sitt utviklede stereokamera benytter en algoritme som *tracker* Aruco-koder. Hvis algoritmen oppdager en Aruco-kode i bildet, blir denne detektert og avstand og rotasjon blir beregnet. Etter beregningen er gjort blir X-, Y-, Z-koordinater til Aruco-koden i forhold til kameraet sendt som en UDP-melding til PLSen. For enkelhets skyld er det i dette prosjektet kun brukt Z-komponenten, ettersom det gir oss et godt nok estimat av avstanden fra krantuppen til gulvet. Figur 4.5 viser bilder fra begge kameraene og *tracking* av Aruco-koden montert på krantuppen.

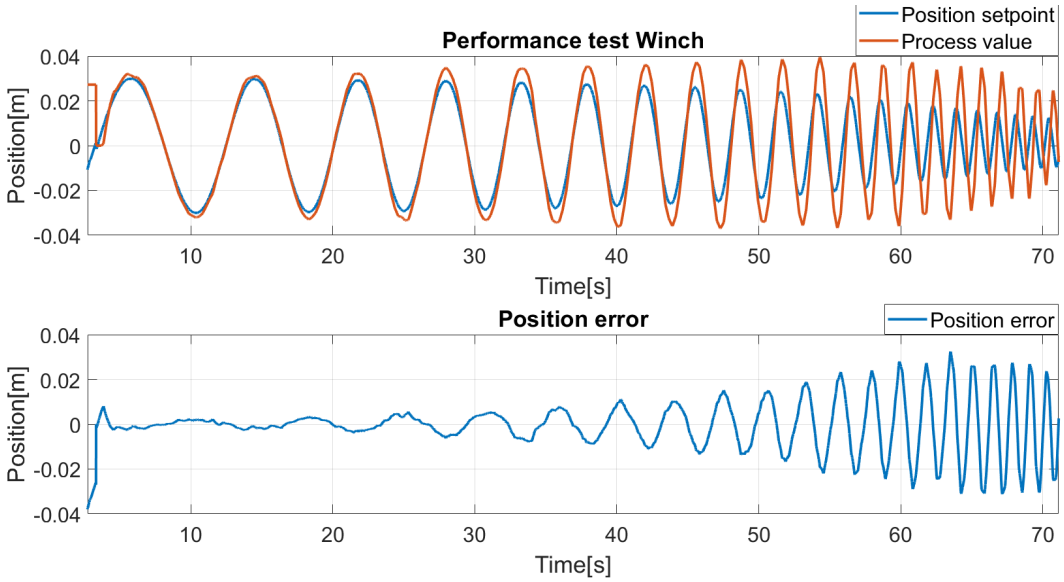


Figur 4.5: Skjermbilde av «*camera control software*», visning av begge kamerabildene med tracking av Aruco-kode montert på krantupp.

4.5 Tester av ulike metoder for AHC

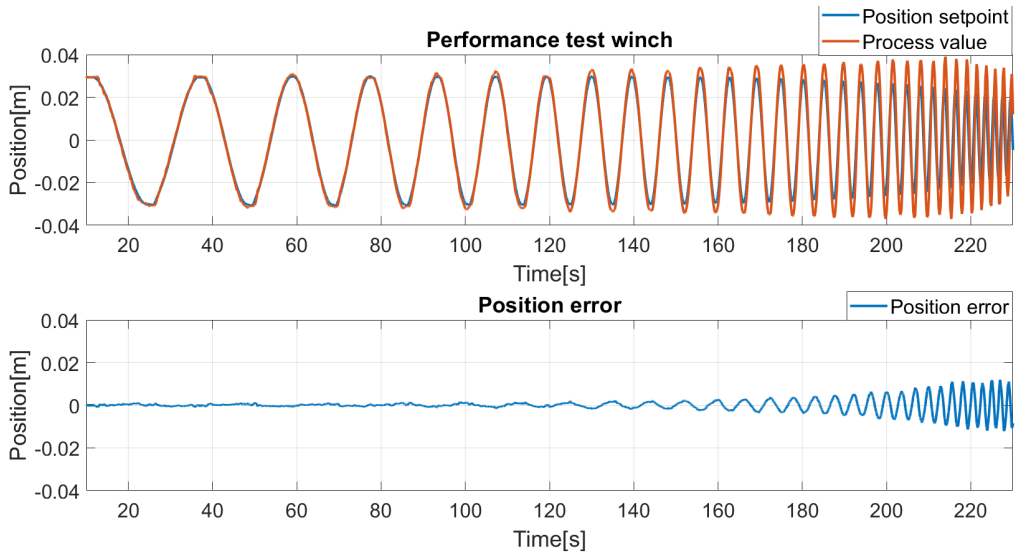
4.5.1 Tuning av AHC

Figur 4.6 viser et plot over Mp- og vinsjposisjonen ved bevegelse på plattformen med fast amplitude på 0.06[m] og økende frekvens ved å minke periodetiden. Vinsjposisjonen er invertert før det er plottet for å tydeliggjøre posisjonsfeilen. Ved optimal AHC er *position setpoint* og *process value* lik, som fører til at *position error* som illustrerer posisjonensendringen til lasten blir tilnærmet lik 0[m]. I figuren vises en av de første innledende testene av beregnet AHC. Det er benyttet en forsterkning på 200[%/m] og en relativt liten filtrering av det beregnede hastighetssettpunktet. Resultatet fra testen viser at lasten relativt tidlig begynner å bevege seg uønsket mye, allerede ved en periodetid på Ca. 10 sekunder begynner vinsjen å slite med å følge med posisjonssettpunktet sitt. Dette er ikke tilfredsstillende AHC-ytelse, ettersom 10 [s] periodetid på bølger er relativt vanlig.



Figur 4.6: Øverste graf i figur viser hvordan vinsjen følger posisjonssettpunkt for en sinus som har økende frekvens fra 10[s] til 1[s] periode under AHC modus. Nederste graf viser hvordan posisjonsfeilen mellom er-verdi og settpunkt øker desto høyere frekvens.

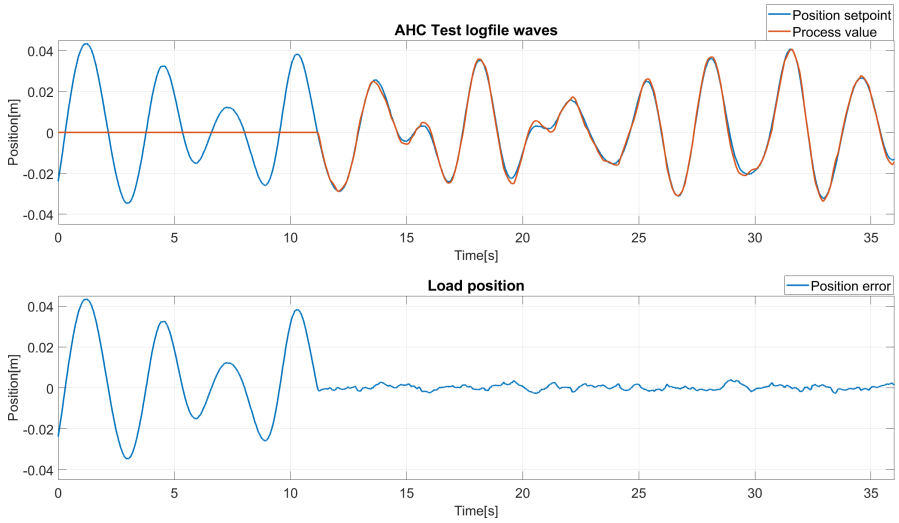
Ved å filtrere hastighetssettpunktet ytterligere, oppnår systemet en betydelig bedre AHC ytelse. Figur 4.7 viser tilsvarende test som i figur 4.6 med samme forsterkning, men med ytterligere lavpassfiltrering av hastighetssettpunktet. I dette plottet er det tydelig bedre AHC-ytelse enn i plottene i figur 4.6. I figur 4.7 begynner ikke lasten å få uønskede bevegelser før plattformen har en periodetid på Ca. 2 sekunder. Ved denne periodetiden klarer ikke sylindrene i plattformen heller å henge med posisjonssettpunktet sitt. Dette betyr at AHCen vil være tilstrekkelig rask nok for denne applikasjonen.



Figur 4.7: Den øverste grafen i figuren viser hvordan vinsjen følger posisjon-settpunkt for en sinus som har økende frekvens fra 10[s] til 1[s] som reguleres med feedforward regulator og P-regulator med i 200 gain. Den nederste grafen viser hvordan posisjonsfeilen mellom er-verdi og settpunkt øker desto høyere frekvens.

4.5.2 Test av metode «Beregnet AHC»

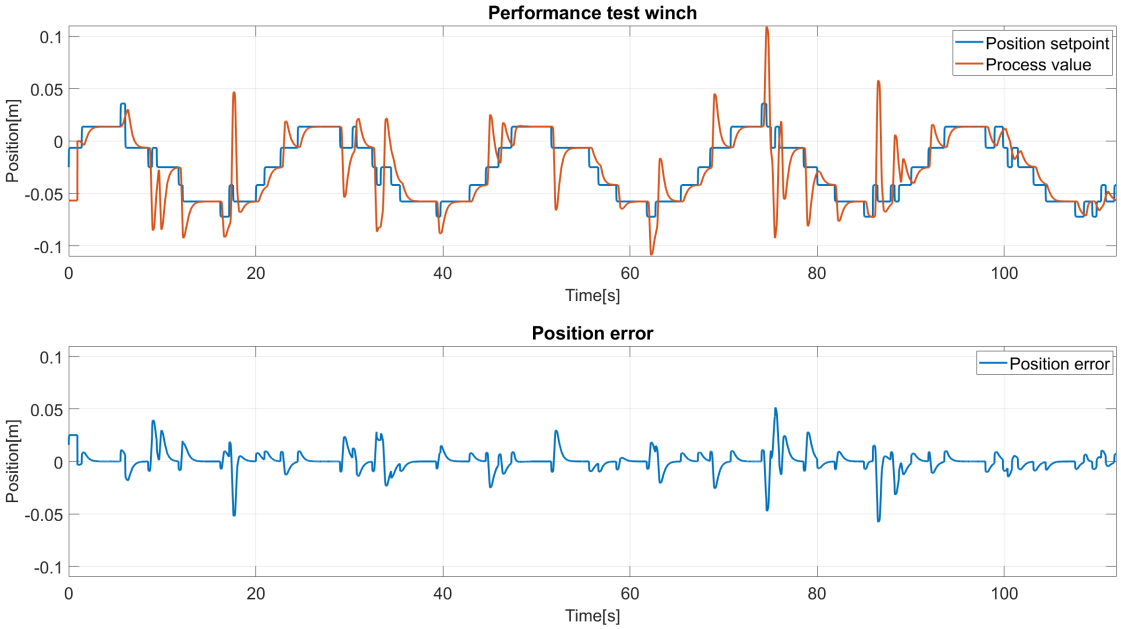
Figur 4.8 viser ytelsen av den beregnede aktiv hiv kompenseringen når plattformen kjører i loggfil modus. De første 12[s] av filen er uten aktivert AHC, som medfører at nederste graf («*Load position*») er lik krantupp-posisjonen grunnet at vinsjen står i ro. Den resterende tiden av testen er med aktivert AHC. Resultatet er da at vinsjen motvirker plattformbevegelsene og regulerer lasten slik at den er tilnærmet i ro. For en optimal AHC vil «*Load position*» være lik 0[m] i AHC-modus. I dette plottet er den tilnærmet lik 0[m] i AHC og tilfredsstiller dermed kravene fra Scantrol. Vedlegg 9 viser spilleliste som inneholder demonstrasjon av «beregnet AHC».



Figur 4.8: Plot som viser ytelsen til den beregnede AHCen når plattformen kjører etter loggfil bølger. Første 12 [s] er uten aktivert AHC og resten er med AHC aktivert. Den blå, øverste grafen er krantupp sin posisjon[m]. Den oransje, øverste grafen viser vinsj bevegelsen, denne er invertert for å tydeliggjøre posisjonsfeil. Den nederste grafen viser den beregnede posisjonen til lasten basert på de to øverste grafene.

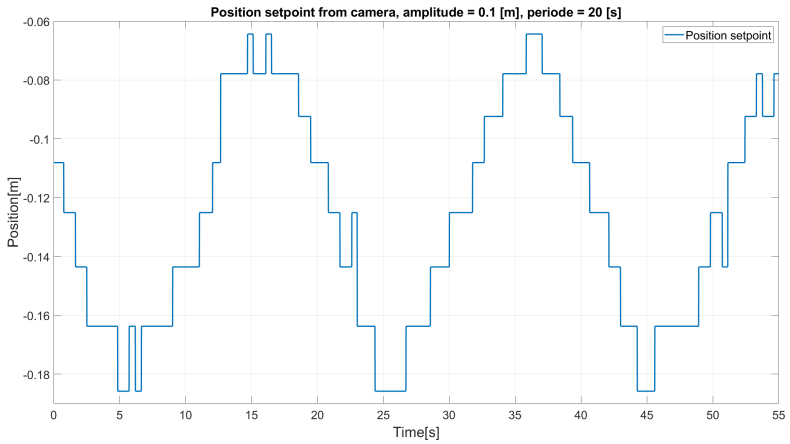
4.5.3 Tester for stereokamera

Siden AHC er tunet i 4.5.1, er det benyttet samme tuning, men ved bruk av et annet referansepunkt for AHC. Når kamera benyttes er det ikke den beregnede krantuppbevegelsen som gir utgangspunkt for AHC, men heller den målte avstanden fra kamera til krantupp. Figur 4.9 viser resultatene av AHC ved bruk av kamera med avstandsmåling til krantupp.



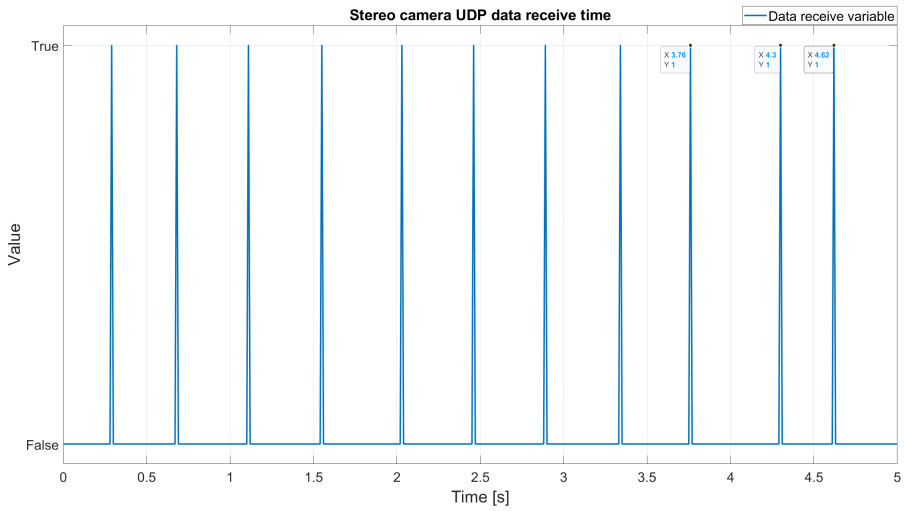
Figur 4.9: Plot som viser ytelsen til AHC ved bruk av stereokamera når plattformen kjører etter simulerte bølger. Den blå, øverste grafen er krantupp sin målte posisjon[m]. Den oransje, øverste grafen viser vinsj bevegelsen, denne er invertert for å tydeliggjøre posisjonsfeil. Den nederste grafen viser den beregnede posisjonen til lasten basert på de to øverste grafene

I figur 4.9 ser man tydelig støy på pådragsignalet som medfører en betydelig *position error* som vist i nederste plot i figuren. Sammenlignet med beregnet AHC er dette resultat betydelig dårligere. Det ble derfor gjort flere tester og undersøkelser for å finne ut av hva som skapte disse problemene. I første undersøkelse sjekket vi posisjonssettpunktet mottatt fra stereokamera. Figur 4.10 viser at dette signalet var svært påvirket av støy.



Figur 4.10: Plot som viser mottatt posisjonssettpunkt fra stereokamera.

Basert på plottet av den beregnede posisjonen til kameraet var det mistanke om at PLSen mottok oppdatert posisjon for sjeldent over UDP. Figur 4.11 viser et plot av hvor ofte data blir mottatt på UDP fra kamera. Her er det tydelig at den sender data uregelmessig og av og til opp mot 0.5 [s] intervall mellom sendingene. Dette kombinert med støyete avstandsmålinger fra kamera, medfører at det er vanskelig å beregne posisjon og hastighet i PLSen. For å redusere støyen er det mulig å lavpassfiltrere den bergede posisjonen og hastigheten, men dette vil skape enda større tidsforsinkelser. Det er problematisk når det allerede er for store tidsforsinkelser i systemet for å kunne benytte AHC ved stereokamera.



Figur 4.11: Plot som viser hvor ofte og med hvilke tidsintervall stereokamera sender beregnede UDP-meldinger til PLS. Resultat viser at data sendes med forskjellig tidsintervall avhengig av analysetiden til datamaskin i stereokamera.

4.6 Evaluering av metoder for AHC

Basert på ytelsestester utført i kapittel 4.5, ble det gjennomført en evaluering for Scantrol om de to aktuelle AHC-metodene.

Beregnet AHC gir som forventet relativt gode resultater ettersom den baserer seg på kjente sylinderposisjoner, det er dermed lite variabler som kan skape støy og målefeil. Dette er derimot ikke en metode som ville fungert på et ekte fartøy, da man ikke kan vite posisjonen til båten på samme måte. På et ekte fartøy må denne bevegelsen måles med en MRU, som ikke er like nøyaktig som den beregnede posisjonen vi oppnår ved bruk av «beregnet AHC».

AHC ved bruk av stereokamera er en metode som fungerer bra i teorien, men som ikke har gitt gode resultater i dette prosjektet. Dette med bakgrunn i testene som er utført i kapittel 4.5.3. Årsaken for at det oppnås dårlig AHC-ytelse med denne metoden er på grunn av for sjelden oppdatering og for mye støy på posisjonsdataene fra kameraet. Dersom kameraet hadde gitt ut posisjoner med høyere frekvens og bedre nøyaktighet, ser vi det mulig å benytte denne metoden for AHC.

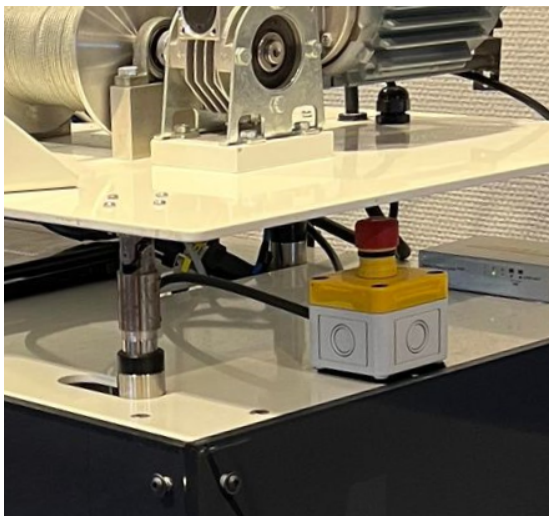
Oppsummert kan man se at beregnet AHC er den foretrukne metoden av de metoder som er evaluert i dette prosjektet. Ettersom kamera gir usikre data med for lav frekvens, blir det for stor tidsforsinkelse og usikkerhet i systemet til at dette er en gunstig metode for AHC. Med forbedringer av kameraet kan det tenkes at dette kan være en god metode for AHC i fremtiden.

5 HMS

Som følge av at dette prosjektet består av flere bevegelige deler med store krefter, har HMS vært høyt prioritert. Det er identifisert flere farer i dette systemet, blant annet, klemfare fra plattform, klemfare i vinsj, ukontrollert vinsj eller plattform og lignende. Derfor er det viktig med gjennomtenkte sikkerhetssystemer og nødstopp. Vedlegg 9 viser spilleliste som inneholder demonstrasjons video av prosedyre ved nødstopp.

5.1 Nødstopp

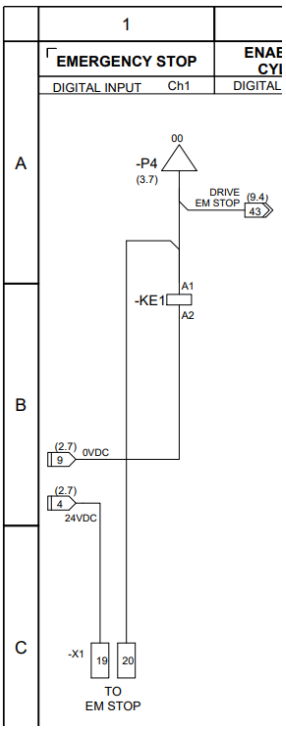
Vi har implementert nødstopp i systemet som er lett tilgjengelig under plattformen, se figur 5.1. Denne har som hovedmål å stoppe systemet på en sikker måte dersom en nødssituasjon skulle oppstå.



Figur 5.1: Bilde av plassering av nødstopp.

5.1.1 Bevegelsesplattform

Nødstoppbryteren er koblet på NC kontakter, som betyr at den åpner kretsen dersom den blir aktivert. Dette medfører at eventuelle feil på kabler eller bryter også vil utløse nødstopp. Nødstoppbryteren er for bevegelsesplattformen koblet direkte på et 3 polt rele som gir strøm til motordriverne for sylindrene. Dette innebærer at dersom nødstopp blir aktivert kuttes strømmen til sylindrene uavhengig av alt annet som skjer i prosessen, og sylindrene stopper. Se figur 5.2 for kobling av nødstopp i dette systemet. Kombinert med å stoppe sylindrene direkte går også nødstoppen inn på en digital inngang i PLSen. Her blir den behandlet og PLSen setter automatisk systemet i stoppmodus. Videre sender PLSen melding til HMI om at den må sette systemet i stopp og gi feilmelding. Ved å utføre disse kommandoene vil systemet være sikkert også når brukeren deaktiverer nødstoppen igjen. Da vil systemet være i stopp-modus og ingenting beveger seg.



Figur 5.2: Elektrisk tegning av nødstopp kobling.

5.1.2 Vinsj

For vinsjen var tanken å gjøre noe tilsvarende som med bevegelsesplattformen. Vårt forslag til bedriften for nødstopp av vinsj, var å benytte en 3-polt kontaktor som strømmen til motoren går gjennom, for så å kutte strømmen til spolen i releet ved en nødstoppsituasjon. Ved å benytte denne metoden vil motoren stanse uavhengig av PLS/frekvensomformer og software.

For enkelhetsskyld kom bedriften med en alternativ løsning der det kun er nødvendig å koble nødstoppen til en digital inngang på frekvensomformeren. Når denne inngangen mister 24[V] (nødstopp aktivert) stopper motoren, og frekvensomformeren går inn i feilmodus. PLSen oppdager også nødstopp og sender alarm i form av lyd i operatørpanelet, samt blinkende rødt

lys på reset-knappen. For å kunne operere vinsjen igjen må reset-knappen trykkes inn, og da resettes frekvensomformer-alarmen og systemet går i standby igjen.

5.2 Andre sikkerhetsfunksjoner

Det er også implementert Torque limit, stopp ved kommunikasjonsfeil, AHC sikkerhet i tillegg til alarmer i HMI som sikkerhetsfunksjoner i systemet.

5.2.1 Torque limit

Det er blant annet lagt til en moment begrensning i frekvensomformeren. Når brukeren opererer vinsjen er det satt en momentbegrensning som skal hindre at personer eller noe henger seg fast i trommelen under drift. Denne begrensningen er satt til 30% av maks moment. På 30% av maksimalt moment er det nok moment til å operere vinsjen tilfredsstillende, men det er samtidig en mulighet for å stoppe den uten for mye kraft.

5.2.2 Kommunikasjonsfeil

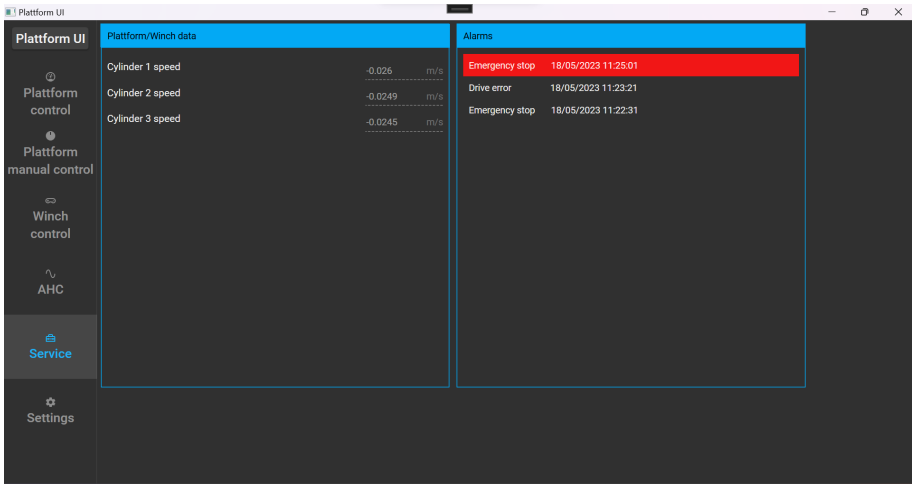
En risiko ved systemet oppstår dersom man mister kommunikasjon mellom HMI, PLS, frekvensomformer og ikke lenger har mulighet til å stoppe eller kontrollere systemet. Dette er løst ved at plattformen stopper automatisk dersom PLS mister kommunikasjon med HMI. Det er implementert en kommunikasjonsteller fra HMI til PLS som teller og sender et oppdatert tall til PLS så lenge HMI og PLS kommuniserer. Når PLS merker at denne er lik flere ganger, betyr det at den har mistet kontakt og systemet går i stopp. Det samme er gjort dersom drive eller operatørpanel mister kontakt med PLS, da går også systemet i stopp.

5.2.3 AHC

Det er implementert sikkerhetsfunksjoner i PLS som må oppfylles før AHC-modus kan bli aktivert. Hvor vinsjen sin posisjon må ligge mellom 0.3 - 0.8 [m], før det er mulig å aktivere modusen. Dette er for å unngå at vinsjen berører gulvet eller krantupp under regulering. Det er også implementert en sikkerhetsfunksjon om at bevegelsesplattformen må være «kjørende» i simulert eller loggfilmodus før AHC kan aktiveres.

5.2.4 Alarmer i HMI

Figur 5.3 viser et skjermbilde av «service» tab i HMI der bruker kan monitorere målte sylinderhastigheter i tillegg til alarmer. Alarmhistorikken vises nedover i listeformat med tidspunkt for hver alarm. Aktive alarmer fremheves med rød bakgrunn, og inaktive alarmer med transparent bakgrunn.



Figur 5.3: Skjermbilde av service tab i HMI.

6 Prosjektutførelse

Prosjektet har bestått av programmering, prosjektering, kobling og testing av ulike løsninger. Det har vært en bratt læringskurve med utfordrende problemstillinger. Den største utfordringen var å utvikle styresystem og optimalisering av reguleringssløyfene. Dette baserte seg på tilegning av mye ny kunnskap innenfor både programmering i «Phoenix PLC next» og reguleringsteknikk teori.

Gjennom prosjektet har vi stort sett fulgt tidsplanen, med unntak av mot slutten, hvor vi hadde satt av for lite tid til del 3 av oppgaven i tillegg til skriving av rapporten. Dette var mer tidkrevende enn forventet, men vi sitter igjen med ett inntrykk over å ha levert ett godt system til Scantrol. Vi har skrevet daglig logg i felles GitHub dokument hvor arbeidet er dokumentert med både tekst og bilder, som er lagt ved i vedlegg 10.5. Det er også ført timelister og møtereferat i felles Microsoft excel-dokument, som er dokumentert i vedlegg 10.3.

Utviklingen av systemet er utført på kontorlokalene til Scantrol i Sandviken, hvor vi har hatt tilgang på nødvendig utstyr og kompetanse ved behov. Muligheten for å ha kunnet arbeide med systemet i praksis har vært svært lærerikt, og har gjort at vi har kunnet testet løsninger og avklart problemstillinger effektivt underveis.

Overgangen fra arbeidet på kontoret med testsylinder og testoppkobling til den fysiske plattformen gikk overraskende effektivt, grunnet resultatet, av grundige og gode forberedelser og kalibreringer gjennomført på testoppsett i forkant. Under arbeidet med testoppsettet ble det lagt til rette for styring av tre sylindre i *software*, som gjorde at vi kunne simulere hele bevegelsessystemet før overgangen til plattformen.

Mot slutten av oppgaven ble det etter møte med veileder Endre Håland og interne veiledere hos Scantrol avklart at vi ikke skulle evaluere metode for AHC med bruk av radarsensor eller teste Scantrol sin egenutviklede mTrack AHC kontroller. Vi ble derimot enig om å teste metode for «Beregnet AHC» og forsøke å benytte «AHC med bruk av stereokamera». Dette ble avgjort grunnet omfanget på oppgaven, og gjorde at vi heller kunne fokusere mer på de to overnevnte metodene. Det var lærerikt å utvikle kontrollsystem for AHC samt benytte stereokamera.

7 Konklusjon

Plattform- og vinsjsystemet som er utviklet i dette prosjektet klarer å simulere bølgebevegelse i tillegg til å benytte AHC for å motvirke bølgebevegelser. Systemet har noen begrensninger med bakgrunn i dimensjonene på sylindrene i tillegg til hastighetsbegrensninger på sylindre og vinsj. Plattformen har mulighet for å bevege seg etter simulerte bølger eller ekte bølger fra loggfil, den kan også kjøre manuelt til en gitt posisjon og vinkel. Vinsjen har mulighet for å opereres i manuell modus med *joystick* eller i AHC hvor den motvirker bølgebevegelsene til plattformen. Det er også utviklet et brukergrensesnitt for monitorering og styring av plattformen samt et PLS program for styring og regulering av kontrollsystemet.

Det er evaluert to ulike metoder for AHC på systemet, «beregnet AHC» og «AHC ved hjelp av stereokamera». Beregnet AHC ga gode resultater, men vil ikke fungere på et ekte fartøy da den baserer seg på å beregne bevegelsen basert på posisjonene til sylindrene i plattformen. AHC ved hjelp av stereokamera ga ikke like gode resultater, grunnet trege og støyete signaldata fra stereokameraet. Vi har gitt tilbakemelding til Scantrol om at dette er en mulig fungerende metode for AHC, dersom stereokameraet blir videreutviklet for å gi raskere og mer troverdige avstander.

7.1 Forslag til forbedringer

7.1.1 Forbedre stereokamera

Vi ser det fullt mulig å benytte stereokamera til AHC, men dette vil kreve at kameraet er utviklet og kalibrert for raskere analyse og dybdeberegning enn hva Scantrol sitt egenutviklede stereokamera gjør i dag. Systemet og PLS-programmet er utviklet slik at andre enkelt skal kunne videreutvikle og teste ulike metoder for AHC.

7.1.2 Forbedre sikkerhet

Sikkerheten kan forbedres med hensyn til vinsjen og nødstopp. Dette er diskutert i kapittel 5.1.2. Det er avtalt med firma at implementert nødstoppløsning på vinsj er tilstrekkelig.

7.2 Videreutvikling

7.2.1 Benytte radarsensor for AHC

På bakgrunn av oppgavens omfang, ble det ikke tid til å evaluere eller teste AHC ved bruk av radarsensor, systemet ligger til rette for at dette kan implementeres ved en senere anledning. Radarsensoren Scantrol ønsket å benytte sender ut 4-20 [mA] basert på avstanden til objektet foran sensoren. Ettersom det ikke er plass til en analog inngangsmodul på PLSen, måtte det blitt brukt en ekstern I/O modul med sensoren koblet til. Tanken var å montere denne sensoren på en metallvinkel ut fra krantuppen slik at sensoren kan måle avstanden ned til gulvet. I og med at denne sensoren er analog vil det i teorien kunne være lettere å lese avstand oftere og unngå for mye tidsforsinkelse i systemet.

Vi ser for oss at AHC med bruk av radarsensor kan fungere tilfredsstillende på plattformen så lenge sensoren gir ut troverdige signaler uten for mye støy. Dette er også noe som må testes nøyere på et ekte fartøy, der det vil være større avstander og andre faktorer som kan påvirke måleresultatet, som for eksempel vær og vibrasjoner.

8 Referanser

Referanser

[1] Scantrol AS. *Active heave compensation(2022)*. [Internett] Available: https://www.scantrol.com/ahc-rov-operations. Funnet: 27.04.2023.

[2] Scantrol AS. *Hjemmeside(2023)*. [Internett] Available: https://www.scantrol.com/. Funnet: 02.05.2023.

[3] Bibliotek. *FieldTalk.Modbus.Master*. [Internett] Available: https://www.modbusdriver.com/shop/product_info.php?products_id=66. Funnet: 08.02.2023.

[4] Bibliotek. *Materialdesign Themes WPF(2023)*. [Internett] Available: http://materialdesigninxaml.net/. Funnet: 27.01.2023.

[5] Bibliotek. *Scottplot WPF(2023)*. [Internett] Available: https://scottplot.net/. Funnet: 16.02.2023.

[6] P. Corke. *Robotics, Vision and Control (2.utgave)*. Springer Tracts in Advanced Robotics, sidetall(22,23,24,25). 2017.

[7] J. Balchen T. Andresen B. Foss. *Reguleringsteknikk (6.utgave)*. ©INSTITUTT FOR TEKNISK KYBERNETIKK, NTNU, TRONDHEIM, sidetall(15,16,17,348,349). 2016.

[8] N.Nise. *Control system engineering (8.utgave)*. Wiley, sidetall(93,94,95). 2019.

[9] OpenCV. *Detection of ArUco Markers(2023))*. [Internett] Available: https://docs.opencv.org/4.x/d5/dae/tutorial_aruco_detection.html. Funnet: 20.05.2023.

9 Appendiks A - Youtube spilleliste

Manuell modus demo video (YouTube)

Simulert bølgemodus demo video (YouTube)

Loggfil bølgemodus demo video (YouTube)

AHC på vinsj i simulert bølgemodus demo video (YouTube)

Nødstopp prosedyre demo video (YouTube)

Manuell kjøring av vinsj med operatørpanel demo video (YouTube)

10 Appendiks B - Prosjektledelse og styring

10.1 Github link

BO23EB-37-Bachelor orginazation (Github)

10.2 Fremdriftsplan

						År	2023																								
						Uke	w6	w7	w8	w9	w10	w11	w12	w13	w14	w15	w16	w17	w18	w19	w20	w21	w22	w23	w24							
						Man	6/2	13/2	20/2	27/2	6/3	13/3	20/3	27/3	3/4	10/4	17/4	24/4	1/5	8/5	15/5	22/5	29/5	5/6	12/6							
						Fre	10/2	17/2	24/2	3/3	10/3	17/3	24/3	31/3	7/4	14/4	21/4	28/4	5/5	12/5	19/5	26/5	2/6	9/6	16/6							
#	Aktivitet	Arbeid [timer]	Start Dato	Slutt Dato	Fram drift	Ansvarlig									Påske																
	Møter med veileder(e)	14	6/2	15/5	0%	Sander, Lars, veileder																									
	Forsprosjekt innlevering	0	6/2	6/2	0%	Sander, Lars	X																								
1	**Styresystem for plattform**	**340**	6/2	17/3	0%	Sander, Lars	V	V	V	V	V	V																			
	Velge drivere for sylindermotorer	8	6/2	6/2	0%	Lars	V																								
	Lese Sylinderpos. Fra enkodere og beregne hastighet og pos.	57	6/2	17/2	0%	Sander	V	V																							
	Sette sammen og koble opp hardware	42	13/2	22/2	0%	Lars		V	V																						
	Lage sw. for styring av plattformen ved Sinus eller loggefil.	105	23/2	17/3	0%	Lars			V	V	V	V																			
	Lage sw. for styring/monitorering fra pc, som skal kommunisere med PLC.	128	20/2	17/3	0%	Sander			V	V	V	V																			
2	**Styring av elektrisk vinsj montert på plattform**	**160**	20/3	21/4	0%	Sander, Lars							V	V		V	V														
	Koble opp frekvensomformer og enkoder til PLC.	20	20/3	24/3	0%	Sander							V																		
	Kommunisere med frekvensomformer over UDP.	30	27/3	12/4	0%	Sander								V		V															
	Styre hastighet og dreiemoment på vinsj	40	20/3	31/3	0%	Lars							V	V																	
	Måle possisjon på vinsj basert på enkoder feedback	40	10/4	21/4	0%	Lars										V	V														
	Koble til operatørpanel for å styre vinsj	30	13/4	21/4	0%	Sander										V	V														
	Midtveis presentasjon forberedelser	32	20/3	24/3	0%	Sander, Lars							V																		
	Midtveispresentasjon	2	31/3	31/3	0%	Sander, Lars								X																	
3	**Teste ulike metoder for AHC av vinsjen**	**160**	24/4	19/5	0%	Sander, Lars												V	V	V	V										
	Benytte og utvikle metode for beregnet AHC	40	24/4	28/4	0%	Sander, Lars												V													
	Benytte stereokamera for å måle avstand fra gulv til aruco	60	1/5	10/5	0%	Sander, Lars													V	V											
	Benytte stereokamera for å måle avstand fra gulv til aruco	60	11/5	19/5	0%	Sander, Lars														V	V										
	Skrive refleksjonsnotat	20	24/4	26/4	0%	Sander, Lars												V													
	Refleksjonsnotat	0	5/5	5/5	0%	Sander, Lars													X												
	Skrive bacheloroppgave	200	10/4	22/5	0%	Sander, Lars							V	V		V	V	V	V	V	V	V									
	Bachelor oppgave innlevering	0	22/5	22/5	0%	Sander, Lars																X									
	Bachelor oppgave muntlig forberedelse	40	22/5	2/6	0%	Sander, Lars																V	V								
	Bachelor oppgave muntlig presentasjon	2	5/6	16/6	0%	Sander, Lars																									
	EXPO23	16	14/6	14/6	0%	Sander, Lars																		V	V						
	Avslutningsfest	0	15/6	15/6	0%	Institutt																			X						
	Totalt antatte timer (hele gruppen):	986																													
	Totalt antatte timer per student:	493																													

Figur 10.1: Fremdriftsplan over bachelorprosjektet.

10.3 Timelister

Navn: Sander Vatne Andersen		
Dato:	**Timer:**	**Aktivitet:**
28.12.2022	8	Begynt på C# program for styring av plattform
29.12.2022	8	Testet biblotek for live charts og design av brukergrensesnitt
30.12.2022	8	Testet biblotek for modbus kommunikasjon mot PLC
09.01.2023	8	Koblet opp PLS/Moduler, kommunikasjon mellom PLS og maskin OK
10.01.2023	4.5	Programmert C#, og koblet/testet encoder mot PLCnext
12.01.2023	7.5	Laget fb for encoder input I PLC og testet modbus kommunikasjon med C# program. Koblet opp sylinder mot PLC.
13.01.2023	7	Lagt til funksjoner I Plattform UI og testet PLC program med PID Posisjonsregulering.
16.01.2023	7.5	Implimentert sinus generator i plc, testet denne. Mottatt amplitude og periodetid over UI. Koblet opp driver til sylinder og testkjørt
17.01.2023	5.5	Koblet enkoder fra sylinder til PLC og fått inn pulser/konvertering til meter. Testet kjøring av sylinder med +-10v fra PLC.
19.01.2023	10	Koblet enable rele og bryter for styring av sylinder. Testet kjøring av sylinder fra PLC, og laget fb for resetting av sylinder til hjem posisjon.
20.01.2023	6	Fått reset sylinder possisjon blokk til å fungere bra. Kjørt step respons og Sinus av sylinder med P - regulator. Logget resultat.
22.01.2023	1	Laget funksjonsblokk for speedtest av sylinder.
23.01.2023	7.5	Laget speedtest funksjonsblokk og kalibrert testsylinder. Laget control funksjonsblokk. Testet og tunet regulering med feed forward og PD-regulator.
24.01.2023	6	Laget funksjonsblokk for sinus bølge simulering med pos setpoint og speed setpoint for alle 3 sylindre. Tunet test sylinder på kontoret og logget alle resultat. Lagt til modbus kode og kjørt sylinder fra Plattform UI.
26.01.2023	7	Møte med alle veiledere. Satt opp kommunikasjon mellom UI og PLC og testet Start/Stop/reset funksjoner fra UI. Testet å kjøre med forkskjellig bølger.
30.01.2023	7	Tatt ny speedtest av sylinder, lagret .csv file for tester av steprespons og diverse tuning paramtre. Laget C# program og matlab kode for å konvertere og plotte .csv filene.
31.01.2023	6	Laget budsjett, tidsplan og blokkdiagram. Satt oss inn i Latex og begynt på forprosjekt dokument.
01.02.2023	1	Programert I PLCnext. Prøvd å legge til ny funksjonalitet I fbOperation.
02.02.2023	8.5	Ferdigsstillt og levert forprosjekt. Laget appendix for sinus test av sylinder for forskjellige tuning parametre og med 5kg last. Speedkalibrert sylinder på nytt med 5kg last og kjørt flere tester. Lagret alt som .csv og lagt I matlab for dokumentasjon.
03.02.2023	8	Faseforskyvet speed sp og tunet sylinder så bra som mulig med last. Laget appendix av alle resultat og logget alt i .csv og plottet i matlab.
06.02.2023	8	Laget appendix for tuning av sylinder. Step respons og ziegler nichols. Testet mange forskjellige parametre og dokumenter. Laget nye funkjsoner og funksjonsblokker I PLC for bruk til alle 3 sylindre. Også endre på hvordan data blir sendt og laget protokoll dok.
07.02.2023	6	Testet ny operation fb som tar hensyn til alle 3 sylindre. Fikset å motta data fra PLC til UI og vise det på skjerm. Lagt til kode for manuel kjøring I UI.
09.02.2023	7	Beregnet Cylinder possisjon fra Hevae/roll pos. Lagt til kode for manual mode I PLC.

10.02.2023	7.5	Laget auto stop funksjon I manual og reset modus. Ferdigstilt manuelt og reset modus for 3 sylindre. Lagt til en del funksjonalitet for loggefiler I UI og innhenting av informasjon fra loggefiler.
13.02.2023	9	Lagt inn funksjon i Plattform UI, for lesing av loggefeil. Henting av data, og sending av data til PLC. Testsending av data over til PLS via MODBUS gikk OK. Laget progressindicators på submit og start fra loggefil siden dette er tidkrevnde.
14.02.2023	4	Laget testprogram i PLC-next som tar imot bølgefildata fra UI via MODBUS. Plottet nedscalerte bølger som vist i bilde under.
16.02.2023	7.5	Testet å kjøre sylinder med MRU loggefiler. Logget resultater og begynt å tune. Laget til rette for 3 sylindre I program for kjøring av loggefil data.
17.02.2023	8	Fikset bugs og små endringer I plattform UI. Sett på mulighet for å velge range av en loggefil og plotte det på grafen før det sendes.
20.02.2023	8	Ordnet reset, home & manual speed kontroll og reached position check i fbOperation. Lagt til 3x av PID og Analog convertert i PLC program, slik at dette er klart for overgang til plattform.Laget klar exeldokument som skal benyttes for SpeedTest kalibrering av alle sylindre på plattform. Bestilt nødvendige komp. Lagt til auto disconnect og andre funksjoner i palttform UI.
21.02.2023	7.5	Koblet opp testsystem for 3 sylindre. Koblet systemet midlertidig over på plattformen. Testet kjøring av plattform i forksjellige moduser
23.02.2023	8	Fikset manual og reset mode for plattformen, og testet mye av disse funksjonene. Fjernet alt av den gamle hardwaren til det gamle systemet. Begynt å koble opp nytt system.
27.02.2023	8	Koblet opp nytt system.
28.02.2023	8	Koblet nytt system til plattformen og festet alle legninger og fikset det mekaniske. Laget skisse av elektriske tegninger for hånd.
01.03.2023	8	Speedkalibrert alle cylinderene nå som de er fastmontert på plattform. Logget resultater og testet kjøring i forskjellige moduser med ny speedkalibrering.Sylinder 1 "henger" litt etter ved kjøring i manual mode.Grunnet mer lastpreg.
06.03.2023	8	Møte med veileder. Rekalibrert sylinder1. Lagt til nødstopp funskjoner og testet og tunet alle funksjoner bedre.
09.03.2023	8	Ferdigstilt nødstopp funksjon I UI og PLC. Rekalbrert alle sylindre med ny sykletid. Testet og logget bølgefiler for tuning.
10.03.2023	8	Møte med veildere I Scantrol. Implementert ny kalibrering på alle sylindre. Jobbet med kjøring av loggefil og fitrering av signaler.
13.03.2023	11	Lagt til flere funksjoner I plattform UI. Service tab, Valg mellom Roll/pitch, Skalering av bølger etter ønsker. Sett på scottplot og mulighet for å implementere denne istdenfor livecharts.
15.03.2023	8	Implementert scottplot istdenfor livecharts. Lagt til og testet plotting I Loggefil og live data fra plattform.

16.03.2023	9	Fikset bugs I UI, Lagt til kode for drive og operatørpanel I plc. Interpolert pos og speed punkter I loggefil data for å få plattformen til å kjøre finere. Gjort endringer i UI for lesing av loggefil og skalering av loggefil data.
17.03.2023	8	Fikset interpolasjon av loggefil data, testet og verifisert mye bedre kjøring av plattform. Fikset skalering av grafer I UI. Funnet feil angående beregning av sylinder possisjon ut fra roll/heave.
18.03.2023	1	Prøvd å fikse problemer med loggefil skalering I UI.
19.03.2023	1.5	Prøvd å fikse problemer med loggefil skalering I UI.
20.03.2023	8	Laget powerpoint til midtveis presentasjon.
21.03.2023	8	Møte med veileder, gjennomgått midtveispresentasjon. Lagt inn skalering for sylinder posisjon og roll/pitch i logfile UI.
23.03.2023	6	Klargjort til midtveispresentasjon. Pakket plattform I bil, og ferdigstilt presentasjon.
24.03.2023	4	Hatt midtveispresentasjon og gjort andre små ting til prosjektet.
27.03.2023	8	Laget rammeverk for Bachelor rapport i Latex. Hatt digitalt møte med veileder for gjennomgang og innspill for rammerverket og rapportens struktur.
28.03.2023	3.5	Endret IP adresse for PLS og Vacon Drive. Testet kommunikasjon over modbus med Drive.
30.03.2023	8	Kommunikasjon mot VaconDrive via modbus, dette fungerer. Men klarer ikke å styre hastigheten riktig. Encoder verdi er oppe å går nå, mottar RPM og retning.
31.03.2023	4	Jobbet med styring av Vacon drive. Styrt vinsj med ulike speed hastigheter nå. Endret til open loop control foreløplig på drive.
03.04.2023	1	Lest meg opp på closed loop vacon motor control.
05.04.2023	1	Fått closed loop vacon motor control til å funke.
11.04.2023	8	Koblet nødstopp til drive, lagt opp DIN skinne med 24V. Ferdigstilt struct for To og From VaconDrive i PLC. Kommunikasjon over TCP er OK. Koblet operatørpanel og laget funksjonsblokk i PLC. Styrt vinsj med operatørpanel joystick og potmeter i begge retninger
13.04.2023	6	Laget logikk for operatørpanel og winch. Testet forskjellige funksjoner.
14.04.2023	9	Lagt til hastighetsberegning, og posisijon for winch. Lagt til resetting og monitorering I UI. Lagt til graf med plotting av winch position i UI. Lagt til andre ui elementer.
17.04.2023	8.5	Jobbet med bachelor rapport.
18.04.2023	2	Møte med veilder, og laget plots til rapport.
20.04.2023	7.5	Jobbet med bachlor rapport.
21.04.2023	5	Jobbet med bachlor rapport.
24.04.2023	7.5	Jobbet med figurer og plots til bachelor rapport. Møte digitalt med veileder.
25.04.2023	7	Jobbet med bachlor rapport.
27.04.2023	10.5	Jobbet med bachlor rapport. Laget funksjonsblokk for automatisk speed kalibrering av vinsj i PLS. Laget funksjon for feedforward for vinsj I PLS.
28.04.2023	8.5	Jobbet bachelor halve dagen, hastighetskalibrert vinsj og implimenter feedforward Jobbet med beregning av crane tip pos
29.04.2023	3	Jobbet med bachelor rapport Laget funksjon I PLS for beregning av Mp position Endret til rett metode for utregning av Heave, Roll

30.04.2023	4.5	Jobbet med bachelor rapport Laget funksjonsblokk I PLS for utregning av MpSpd Laget AHC operation mode I PLS med betingelser for å entre modus
01.05.2023	4	Jobbet med bachelor rapport
02.05.2023	7	Jobbet med bachelor rapport + appendix
03.05.2023	8	Jobbet med bachelorr rapport + appendix Hatt møte med veilder Endre Testet å benytte/kjøre beregnet AHC på vinsj. Testet med P-regulator og feedforward
04.05.2023	2	Fått beregnet AHC med feedforward og p-reg til å fungere.
05.05.2023	8	Jobbet med bachelor. Hatt besøk av veileder, testkjørt plattform og vinsj med AHC til sine begrensninger logget i CSV filer.
07.05.2023	5.5	Jobbet med bachelor rapport. Fikset bugs I UI. Lagt til grafer I AHC tab og utregning av Mp pos I UI.
08.05.2023	8	Kjørt begrensningstester for plattform og vinsj og logget til CSV filer for innhold i rapport. Startet å kalibrere vinsj. Limt inn grafer/plots i rapport.
09.05.2023	10	Jobbet med bachelor rapport Laget bilder og videoer til rapport Tunet vinsj Lagt til utregning og plotting av loadpos I UI. Lagt til sending av AHCCenterPos og OpMode winch fra plc
11.05.2023	8	Koblet kamera til system. Kalibrert og justert fokus på kamera jobbet med bachelor rapport Lagt til kode for å ta I mot data fra kamera I PLS
12.05.2023	8	Fått UDP data fra kamera inn I PLS og tolket x,y,z kordinater til Aruco Kjørt AHC og plottet data med kamera som referanse punkt Jobbet med bachelor rapport
14.05.2023	2	Jobbet med bachelor rapport
15.05.2023	8	Jobbet med bachelor rapport
16.05.2023	8	Jobbet med bachelor rapport + digitalt møte med veileder
18.05.2023	7.5	Jobbet med bachelor rapport
19.05.2023	5.5	Jobbet med bachelor rapport + siste digitale møte med veileder
20.05.2023	9	Jobbet med bachelor rapport, korrekturlest og finpusset, kommentert kode
21.05.2023	10	Ferdigstillt og finpusset bachelor rapport

Navn: Lars Askild Skavhellen Aarvik		
Dato:	**Timer:**	**Aktivitet:**
09/01/2023	8.00	Koblet opp PLS/Moduler, kommunikasjon mellom PLS og maskin OK
10/01/2023	4.5	Testet enkoder mot PLCnext, påbegynt PLC program for testing med PID
12/01/2023	7.5	Laget fb for encoder input I PLC og testet modbus kommunikasjon med C# Koblet sylinder mot PLS
13/01/2023	7	Laget og testet PLC program med PID regulering og testing av posisjon. Laget fb til operasjonsModus for PID regulator. Analog output fb lastet ned.

16/01/2023	7.5	Implimentert sinus generator i plc, testet denne. Mottatt amplitude og periodetid over UI. Koblet opp driver til sylinder og testkjørt
17/01/2023	5.5	Koblet enkoder fra sylinder til PLC og fått inn pulser/konvertering til meter. Testet kjøring av sylinder med +-10v fra PLC.
19/01/2023	8	Koblet opp enable relay og switch-bryter. Testet kjøring av sylinderen fra PLC og laget fb for reset/home pos av sylinder.
20/01/2023	5.5	Fått reset sylinder possisjon blokk til å fungere bra. Kjørt step respons og Sinus av sylinder med P - regulator. Logget resultat.
23/01/2023	7.5	Laget control fb for styring av sylinder med feed forward kombinert med PD-reg. Kjørt speedtest og kalibrert sylinder med bruk av speedtest-fb. Testet og tunet regulering med og uten PD-regulator.
24/01/2023	6	Laget funksjonsblokk for sinus bølge simulering med pos setpoint og speed setpoint for alle 3 sylindre. Tunet test sylinder på kontoret og logget alle resultat. Lagt til modbus kode og kjørt sylinder fra Plattform UI.
26/01/2023	7	Møte med veileder fra 13-14 Lagt til kode for kommunikasjon mellom UI og PLC, testet start/stopp/reset fra UI. Laget struct klasser i PLC for DataToUI og DataFromUI
30/01/2023	7	Laget CSV filer for step respons med P-reg og FeedForward uten P-reg. Laget program i C# for konvertering av data i CSV filene. Kjørt ny speedkalibreringstest av sylinder da denne hadde begynt å drifte.
31/01/2023	6	Jobbet med forprosjekt, satt oss inn i Latex. Laget tidsplan, budsjett og blokkdiagram.
02.02.2023	8.5	Ferdigsstillt og levert forprosjekt. Laget appendix for sinus test av sylinder for forskjellige tuning parametre og med 5kg last. Speedkalibrert sylinder på nytt med 5kg last og kjørt flere tester. Lagret alt som .csv og lagt I matlab for dokumentasjon.
03.02.2023	8	Faseforskyvet speed sp og tunet sylinder så bra som mulig med last. Laget appendix av alle resultat og logget alt i .csv og plottet i matlab.
06/02/2023	8	Laget appendix tuning av cylinder, step respons, zieglerNichols. Testet og tunet sylinder. Gjort om fbFeedForward til en funksjon istedenfor. Startet på ny Operajonsfb. Laget protokoll dokument for datatype og kommunikasjon mellom PLC og UI.
07/02/2023	4	Testet ny operation fb som tar hensyn til alle 3 sylindre. Fikset å motta data fra PLC til UI og vise det på skjerm.
09/02/2023	7	Beregnet Cylinder possisjon fra Hevae/roll pos. Lagt til kode for manual mode I PLC.
10.02.2023	7.5	Laget auto stop funksjon I manual og reset modus. Ferdigstilt manuelt og reset modus for 3 sylindre. Lagt til en del funksjonalitet for loggefiler I UI og innhenting av informasjon fra loggefiler.
13/02/2023	7	Lagt inn funksjon i Plattform UI, for lesing av loggefeil. Henting av data, og sending av data til PLC. Testsending av data over til PLS via MODBUS gikk OK.
14.02.2023	4	Laget testprogram i PLC-next som tar imot bølgefildata fra UI via MODBUS. Plottet nedscalerte bølger som vist i bilde under.

16/02/2023	7.5	Testet å kjøre sylinder med MRU loggefiler. Logget resultater og begynt å tune. Laget til rette for 3 sylindre I program for kjøring av loggefil data.
17/02/2023	0	Syk - egenmelding
20/02/2023	6	Ordnet reset, home & manual speed kontroll og reached position check i fbOperation. Lagt til 3x av PID og Analog convertert i PLC program, slik at dette er klart for overgang til plattform.Laget klar exeldokument som skal benyttes for SpeedTest kalibrering av alle sylindre på plattform. Bestilt nødvendige komp.
21/02/2023	7.5	Koblet opp testsystem for 3 sylindre. Koblet systemet midlertidig over på plattformen. Testet kjøring av plattform i forksjellige moduser
23/02/2023	8	Fikset manual og reset mode for plattformen, og testet mye av disse funksjonene. Fjernet alt av den gamle hardwaren til det gamle systemet. Begynt å koble opp nytt system.
27/02/2023	8	Koblet opp nytt system.
28/02/2023	8	Koblet nytt system til plattformen og festet alle legninger og fikset det mekaniske. Laget skisse av elektriske tegninger for hånd.
01/03/2023	8	Speedkalibrert alle cylinderene nå som de er fastmontert på plattform. Logget resultater og testet kjøring i forskjellige moduser med ny speedkalibrering.Sylinder 1 "henger" litt etter ved kjøring i manual mode.Grunnet mer lastpreg.
06/03/2023	8	Møte med veileder. Rekalibrert sylinder1. Lagt til nødstopp funskjoner og testet og tunet alle funksjoner bedre.
09/03/2023	8	Ferdigstilt nødstopp funksjon I UI og PLC. Rekalbrert alle sylindre med ny sykletid. Testet og logget bølgefiler for tuning.
10/03/2023	8.00	Møte med veildere I Scantrol. Implementert ny kalibrering på alle sylindre. Jobbet med kjøring av loggefil og fitrering av signaler.
13/03/2023	8	Lagt til flere funksjoner I plattform UI. Service tab, Valg mellom Roll/pitch, Skalering av bølger etter ønsker.
15/03/2023	8	Implementert scottplot istdenfor livecharts. Lagt til og testet plotting I Loggefil og live data fra plattform.
16/03/2023	8	Fikset bugs I UI, Lagt til kode for drive og operatørpanel I plc. Interpolert pos og speed punkter I loggefil data for å få plattformen til å kjøre finere.
17/02/2023	8	Fikset interpolasjon av loggefil data, testet og verifisert mye bedre kjøring av plattform. Fikset skalering av grafer I UI. Funnet feil angående beregning av sylinder possisjon ut fra roll/heave.
20.03.2023	8	Laget powerpoint til midtveis presentasjon.
21.03.2023	8	Møte med veileder, gjennomgått midtveispresentasjon. Lagt inn skalering for sylinder posisjon og roll/pitch i logfile UI.
23.03.2023	6	Klargjort til midtveispresentasjon. Pakket plattform I bil, og ferdigstilt presentasjon.
24.03.2023	4	Hatt midtveispresentasjon og gjort andre små ting til prosjektet.
27.03.2023	8	Laget rammeverk for Bachelor rapport i Latex. Hatt digitalt møte med veileder for gjennomgang og innspill for rammerverket og rapportens struktur.
28.03.2023	3.5	Endret IP adresse for PLS og Vacon Drive. Testet kommunikasjon over modbus med Drive.

30.03.2023	8	Kommunikasjon mot VaconDrive via modbus, dette fungerer. Men klarer ikke å styre hastigheten riktig. Encoder verdi er oppe å går nå, mottar RPM og retning.
31.03.2023	4	Jobbet med styring av Vacon drive. Styrt vinsj med ulike speed hastigheter nå. Endret til open loop control foreløplig på drive.
06.04.2023	1	Skrevet på bachelor rapport
07.04.2023	1	Skrevet på bachelor rapport
11.04.2023	8	Koblet nødstopp til drive, lagt opp DIN skinne med 24V. Ferdigstilt struct for To og From VaconDrive i PLC. Kommunikasjon over TCP er OK. Koblet operatørpanel og laget funksjonsblokk i PLC. Styrt vinsj med operatørpanel joystick og potmeter i begge retninger
12.04.2023	2	Jobbet med rapportstruktur, innsatt bilder og overskrifter
13.04.2023	6,5	Laget logikk for operatørpanel og winch. Testet forskjellige funksjoner. Startet på skriving av definisjonen bak elektrisk motor og frekvensomrformer
14/04/2023	9,5	Lagt til hastighetsberegning, og posisijon for winch. Lagt til resetting og monitorering I UI. Lagt til andre ui elementer. Lagt til topptekst på alle funksjoner/fb og program i PLS + ryddet i kode
17/04/2023	8.5	Jobbet med bachelor rapport.
18/04/2023	2	Møte med veilder, og laget plots til rapport.
20/04/2023	7.5	Jobbet med bachlor rapport.
21/04/2023	5	Jobbet med bachlor rapport.
24/04/2023	7.5	Jobbet med figurer og plots til bachelor rapport. Møte digitalt med veileder.
25/04/2023	7	Jobbet med bachlor rapport.
27.04.2023	8	Jobbet med bachlor rapport.
28/04/2023	8.5	Jobbet bachelor halve dagen, hastighetskalibrert vinsj og implimenter feedforward Jobbet med beregning av crane tip pos
29/04/2023	3	Skrevet på bachelor rapport, laget reguleringsløyfe og lagt inn kodeutsnitt for hastighetskalibrering av vinsj
30/04/2023	3.5	Ryddet og skrevet i bachelor, laget illustrasjon i 3D av plattform med kordinatsystem
01/05/2023	4	Jobbet med bachelor rapport.
02.05.2023	8	Jobbet med bachelor rapport + appendix
03.05.2023	8	Jobbet med bachelorr rapport + appendix Hatt møte med veilder Endre Testet å benytte/kjøre beregnet AHC på vinsj. Testet med P-regulator og feedforward
05.05.2023	8	Jobbet med bachelor. Hatt besøk av veileder, testkjørt plattform og vinsj med AHC til sine begrensninger logget i CSV filer.
07/05/2023	5.5	Jobbet med bachelor rapport .
08.05.2023	8	Kjørt begrensningstester for plattform og vinsj og logget til CSV filer for innhold i rapport. Startet å kalibrere vinsj. Limt inn grafer/plots i rapport.
09.05.2023	9	Jobbet med bachelor rapport Laget bilder og videoer til rapport Tunet vinsj Laget youtube spilleliste og lastet opp videoer for ulike demo videor tilknyttet prosjekt
10.05.2023	7.00	Jobbet med bachelor rapport

11.05.2023	8	Koblet kamera til system. Kalibrert og justert fokus på kamera jobbet med bachelor rapport Lagt til kode for å ta I mot data fra kamera I PLS
12.05.2023	8	Fått UDP data fra kamera inn I PLS og tolket x,y,z kordinater til Aruco Kjørt AHC og plottet data med kamera som referanse punkt Jobbet med bachelor rapport
13.05.2023	1	Jobbet med bachelor rapport
15.05.2023	8	Jobbet med bachelor rapport
16.05.2023	8	Jobbet med bachelor rapport + digitalt møte med veileder
18.05.2023	9	Jobbet med bachelor rapport
19.05.2023	7.5	Jobbet med bachelor rapport + siste digitale møte med veileder
20.05.2023	11	Jobbet med bachelor rapport, korrekturlest og finpusset
21.05.2023	11	Ferdigstillt og finpusset bachelor rapport

10.4 Møtereferat

Tidspunkt :	Møte Referat :
16/02/2023	Kort digitalt møte med Endre Håland, fikk god input og tips om lavpassfiltrering av Cylinder speed SP for loggefil. Fikk pirk på å være besvist på aksetekster og knytte terori inn i oppgave/rapport.
06/03/2023	Digitalt møte med veileder, gjennomgått plan mot midtveispresentasjon. Gitt oss tips til å knytte faglig regtek-innhold inn som teori i oppgaven.
21/03/2023	Digital møte med veileder, gått gjennom midtveispresentasjon- Fått input og noen konkrete tilbakemeldinger på forslag til endring. Avtalt møte neste uke planlegging av rapport rammeverk.
27/03/2023	Digital møte med veileder, gått gjennom rammeverk for rapporten i Latex. Fått innspill om å dele rapporten opp i tre deler, slik at
18/04/2023	Digitalt møte med veilder, Snakket om rapport skriving og veien videre.
24/04/2023	Digitalt møte med veilder. Avklart at vi skal droppe og skrive om selvfølgeligheter som for eks. Virkemåte motor/frekvensomformer.
03/05/2023	Digitalt møte med veileder: Tips til rapportskriving - må ha med fler av testene vi har gjort. Begrensninger for systemet (Plattform). Tippset oss også om å lage tabell for hva vi har gjort og hva som hva gjort av arbeid fra tidligere.
05/05/2023	Fysisk møte med veileder på kontoret vårt i Sandviksbodene. Veileder har tipset og vært med å kjøre begrensningtester på plattform og vinsj + logging til CSV filer som skal
12/05/2023	Digitalt møte med veileder, snakket om siste innspurt og diverse ting rundt rapporten.
16/05/2023	Digitalt møte med veileder. Endre har lest gjennom rapport, sa seg fornøyd med alt innhold og mente at vi ikke trengte å legge inn noe mer.
19/05/20023	fra oss. Avklart med Endre at kode skal legges ved som vedlegg I bachelorrapport.

10.5 Arbeidslogg

Wave simulation and evaluation of methods for active heave compensation - Log

Log

21/05/2023

- Ferdigstilt bachelor rapport

20/05/2023

- Jobbet med bachelor rapport

19/05/2023

- Jobbet med bachelor rapport

18/05/2023

- Jobbet med bachelor rapport

16/05/2023

- Jobbet med bachelor rapport
- Hatt digitalt møte med veileder. Siste utkast sendt til han

15/05/2023

- Jobbet med bachelor rapport
- Laget diverse plots til rapport.

14/05/2023

- Jobbet med bachelor rapport

13/05/2023

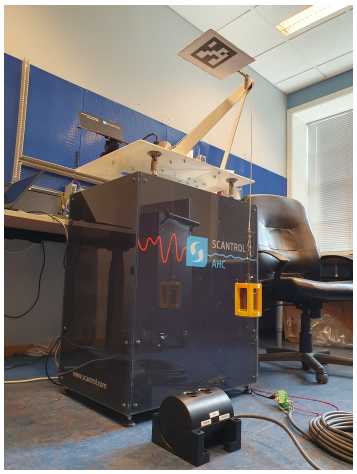
- Jobbet med bachelor rapport

12/05/2023

- Jobbet med bachelor rapport
- Fått inn UDP data fra stereo kamera med xyz kordinater til aruco kode til PLS
- Kjørt AHC med kamerea som referansse punkt, logget resultat til CSV-fil for rapport.

11/05/2023

- Jobbet med bachelor rapport.
- Koblet kamera til system
- kalibrert og justert fokus på kamera
- Montert brakkett med Aruco kode på crane tip
- Lagt til kode i PLS for å ta i mot data fra kamera på UDP.



09/05/2023

- Jobbet med bachelor rapport.
- Jobbet med begrensningstester av plattform og vinsj.
- Tunet winch.
- Laget bilder og videoer til rapport og youtube.
- Lagt til utregning og plotting av load position i UI.
- Lagt til sending av opMode for winch og AHCCenterPos fra PLC.

08/05/2023

- Jobbet med begrensningstester av plattform og vinsj.
- Laget Csv filer og lagt inn plotts i rapport.
- Ordnet bug under reset i UI generert av Scottplot.

07/05/2023

- Jobbet med bachelor rapport.
- Fikset bugs i UI og lagt til graf i AHC av Mp Pos.

05/05/2023

- Jobbet med bachelor rapport.
- Hatt besøk av veileder fra HVL, hvor vi felles har testkjørt plattform og vinsj med beregnet Aktiv hiv kompensering til sine begrensninger med frekvenssweep.

04/05/2023

- Fått AHC med Feedforward og p - regulator til å fungere.
- Filtrert mp_speed og endret måten speedcommand er beregnet.

03/05/2023

- Jobbet med bachelor rapport
- Testet å benytte beregnet AHC på vinsj.
- Testet med P - regualtor og feedforward, fortsatt litt tuning igjen.

02/05/2023

- Jobbet med bachelor rapport
- Laget appendix, bruksanvisning UI + modbus protokoll dokumenter

01/05/2023

- Jobbet med bachelor rapport

30/04/2023

- Jobbet med bachelor rapport
- Laget funksjonsblokk i PLS for utregning av Mp speed
- Laget AHC operation mode i PLS med betingelser for å entre AHC modus.

29/04/2023

- Jobbet med bachelor rapport
- endret utregning av sylinder posisjoner i PLC kode og UI.
- Laget funksjon i PLS for utregning av Mp position.

28/04/2023

- Jobbet med bachelor halve dagen
- Speedkalibrert vinsj
- Jobbet med beregning av crane tip position for AHC.

27/04/2023

- Jobbet med bachelor rapport.
- Laget funksjonsblokk i PLS for automatisk speed kalibrering av vinsj.
- Laget feedforward funksjon for vinsj i PLS.

25/04/2023

- Jobbet med bachelor rapport.

24/04/2023

- Jobbet med figurer og plots til rapport.
- Møte digitalt med veileder, avklart litt som å fks droppe "selvfølgeligheter" som fks prinsipp bak elektrisk motor og frekvensomformer.

21/04/2023

- Jobbet med bachelor rapport.

20/04/2023

- Jobbet med bachelor rapport.

18/04/2023

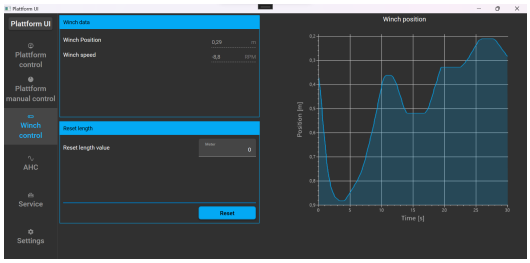
- Hatt møte med veileder.
- Laget plots til rapport.

17/04/2023

- Jobbet med bachelor rapporten.

14/04/2023

- Beregnet hastighet av sylindre og vinsj med bruk av enkoder pulser.
- Beregnet posisjon til vinsj
- Lagt inn tab i UI for reset vinsj lengde og posisjon/hastighet[rpm] til enhver tid.
- Endret UI tabs.
- Lagt til graf med plotting av vinsj posisjon i UI.



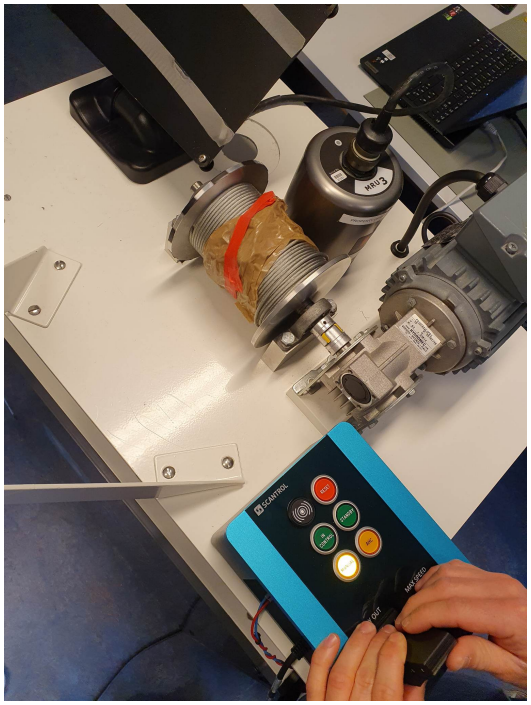
13/04/2023

- Lagt inn endring av hastighet via potmeter på OP-panel.
- Lagt inn nødstopp med buzzer og lys på OP-Panel.
- Sjekket metoder for konvertering av encoderpulser til posisjon og puls/omdreining.

11/04/2023

- Koblet nødstopp til drive - test OK

- Lagt opp DIN - skinne med 24V tilgjengelig på plattform.
- Ferdigstilt struct for To og From VaconDrive i PLC, kommunikasjon over TCP er OK.
- Koblet OperatørPanel og laget funksjonsblokk for denne i PLC.
- Styrt vinsj med joystick fra operatørpanel i begge retninger, endrer torque command via potmeteret.



31/03/2023

- Fått styrt vinsj med ulike hastigheter fra PLC.
- Endret til open loop control forløpig for å få det til å funke.

30/03/2023

- Jobbet med kommunikasjon mot Vacon Drive
- Klarer å styre den med Start/Stopp/reset.
- Mottar encoder verdi og tilbakemelding om RPM.
- Sliter med styring av hastigheten, klarer å endre Torque CMD og Speed CMD.

28/03/2023

- Endret IP adressen for PLS til 192.168.127.213
- Endret IP adressen for Vacon Drive til 192.168.127.212
- Testet å kommunisere mot Vacon vinsj drive i PLS next, ikke fungerende.
- Laget funksjonsblokk fbFromDrive, som leser Modbus Array fra drive.

27/03/2023

- Laget rammeverk for Bachelor rapport i Latex.
- Hatt digitalt møte med veileder for gjennomgang og innspill for rammerverket og rapportens struktur.

24/03/2023

- Midtveispresentasjon hele dagen på HVL.

23/03/2023

- Klargjort siste finish før midtveispresentasjon
- Pakket ned plattform og fått den inn i bilen.

21/03/2023

- Møte med veileder hvor vi har gjennomgått og finpusset på midtveispresentasjon.
- Lagt inn skaleringsmuligheter av heavepos og roll/pitch i logfileUI.

20/03/2023

- Laget powerpoint til midtveis presentasjon.
- Laget reguleringssløyfer og annet materiale til presentasjon.

18/03/2023 - 19/03/2023

- Prøvd å fikse skalerings problemer av loggefil data. Lagt til funksjon for å kun skalere heav pos og beholde riktig roll/pitch angle.

17/03/2023

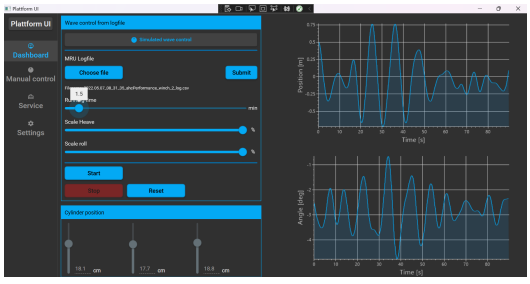
- Jobbet videre med interpolasjon av loggefildata, fått dette til å fungere. Kjørt tester og logget til Csv fil.
- Fikset skalering av UI grafene mh. tid [s].
- Funnet feil i beregning av sylinderposisjon data fra loggefiler, spesielt angle- som ikke blir oppnådd.
    - Må jobbe videre med dette på mandag.

16/03/2023

- Laget struct-klasser for VACON og operatørpanel i PLC next.
- Fikset bugs i UI knyttet mot loggefil-tab og simulering-tab.
- Interpolert posisjon og speed punkter i loggefildata for en finere bevegelse.

15/03/2023

- Byttet om til bruk av Scotplott grafer for UI.
- Laget metoder som beregner roll/pitch vinkel og heave pos som plottes live til grafer i UI.



13/03/2023

- Arbeid med UI, lagt til flere funksjoner:
- Lagt inn mulighet for å velge mellom roll/pitch angle.
- Laget Service tab klar i UI.
- Ordnet lesing av forskjellige typer loggefiler.
- Sett på implementasjon av scottplot isteden for livecharts.

10/03/2023

- Møte med veileder i firma.
- Fikset ferdig og implimentert den nye feed forward kalibreringen fra igår.
- Jobbet og sett mye på filtrering av SpeedSP som er beregnet under lesing fra loggefil.

09/03/2023

- Ferdigstilt nødstopp fix både i UI og PLC.
- Rekalibrert alle sylinderene på plattformen med ny syklustid.
- Testet og logget bølgefiler for tuning.
- Lagt inn mulighet for UI å kjøres på norsk operativ system

06/03/2023

- Møte med veileder
- Rekalibrert sylinder 1
- Lagt til nødstoppfunksjon (UNDER progress)
- Endret sending av loggefildata til 16 bit istedenfor 32 bit.

01/03/2023

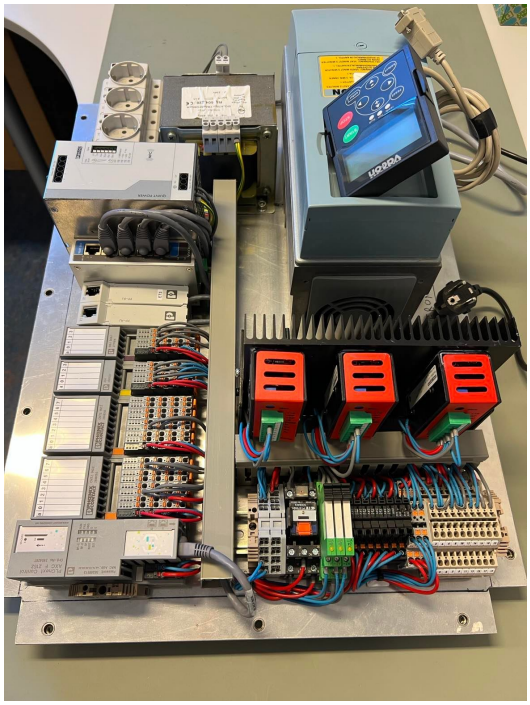
- Speedkalibrert alle cylinderene nå som de er fastmontert på plattform.
- Logget resultater og testet kjøring i forskjellige moduser med ny speedkalibrering.
- Sylinder 1 "henger" litt etter ved kjøring i manual mode.Grunnet mer lastpreg.

28/02/2023

- Koblet ferdig plattform
- Testkjørt plattform - gikk OK
- Ferdigstilt elektriske tegninger

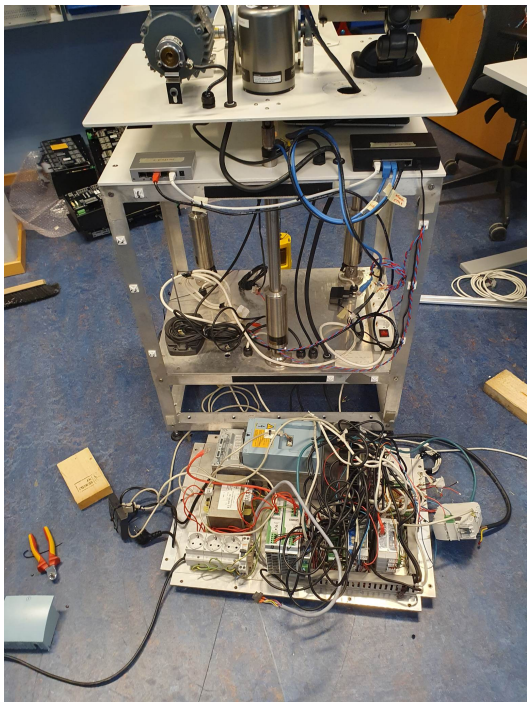
27/02/2023

- Koblet testmodul helt over til plattform.
- Bestilt sikring . 10A - B



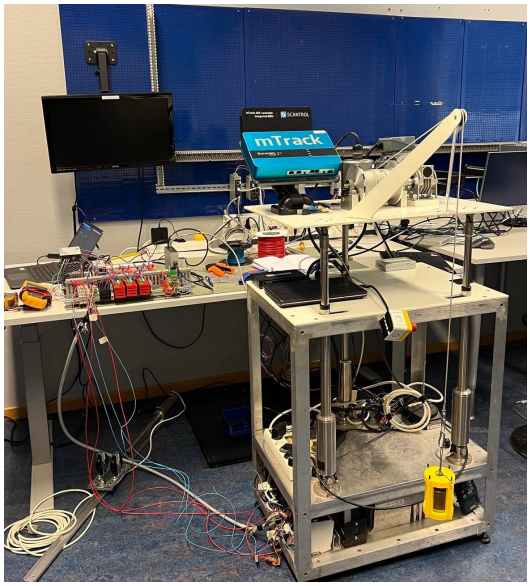
23/02/2023

- Testkjørt plattform i alle moduser.
- Ordnet slik at manual mode og reset fungerer.
- Koblet hele den gamle plattformen helt fra hverandre.
- Montert nye drivere på kjøleribbe
- Målt opp og klargjort overgangen fra testkobling til plattform.



21/02/2023

- Koblet opp testsystem for tre sylindere
- Koblet systemet midlertidlig over til plattformen
- Testet kjøring av plattformen i forskjellige moduser.

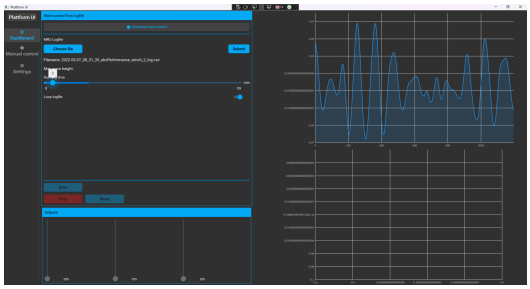


20/02/2023

- Ordnet reset, home & manual speed kontroll og reached position check i fbOperation.
- Lagt til 3x av PID og Analog convertert i PLC program, slik at dette er klart for overgang til plattform.
- Laget klar exeldokument som skal benyttes for SpeedTest kalibrering av alle sylindre på plattform.
- Fått oversikt og bestillt nødvendige komponenter, rekkelemmer, DIN-skinner, sikringholdere(glass).
- Lagt til disconnect og auto disconnect i UI. Samt andre små endringer i UI kode.

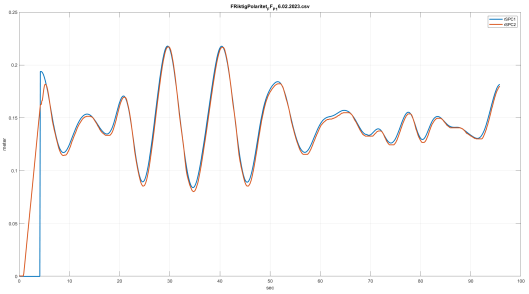
17/02/2023

- Fikset bugs og lagt til små endringer/funksjoner i Plattform UI.
- Sett på muligheten for å velge en valgfri range fra en loggefil og plotte den i UIen før den sendes for å lettere se hvilken type bølge man velger.
- Se bilde under av 2 første minutt valgt av fil og plottet på graf.



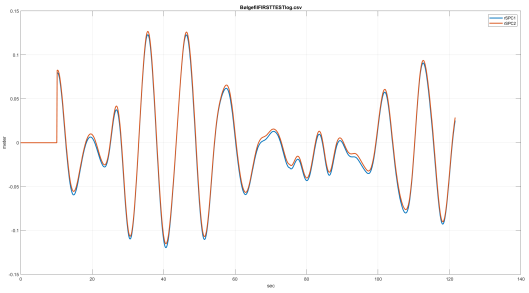
16/02/2023

- Testkjørt sylinder med last - setpunkt fra bølge loggefiler.
- Logget og lagret mange tester til csv loggefiler.
- Startet fintuning, orange funksjon er PV og blå er SP i bildet under:



14/02/2023

- Laget testprogram i PLC-next som tar imot bølgefildata fra UI via MODBUS.
- Plottet nedscalerte bølger som vist i bilde under.



13/02/2023

- Lagt inn funksjon i Plattform UI, for lesing av loggefil. Henting av data og sending av data til PLC.
- Testsending av data over til PLC via MODBUS gikk OK.
- Laget progressindicators på submit og start fra loggefil knapper i UI. Siden dette er tidkrevende oppgaver.

10/02/2023

- Laget auto stop funksjon i manual og reset mode (PLS)
- ferdigstillt manuell og reset mode i OperationMode fb.
- Lagt til funksjonalitet for lesing av Loggefiler og design i UI.
- Laget C# program for konvertering av bølgeloggefiler.

09/02/2023

- Beregnet sylinderposisjon fra heave /roll pos.
- Lagt til kode for manual mode i PLS

07/02/2023

- Testet ny operation blokk som tar hensyn til alle tre sylindrene.
- Sendt posisjonsSetPoint fra PLC til UI og fremvist dette i "slider"/vise på skjerm.
- Lagt til kode for manuel kjøring i UI.
- Ryddet opp i kode i UI og laget diverse metoder.

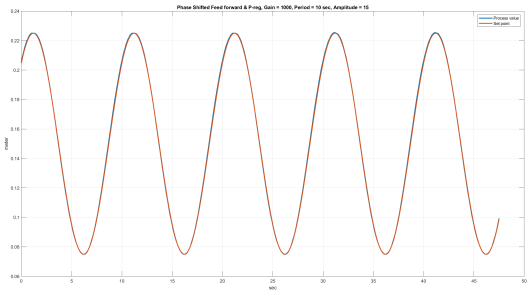
06/02/2023

- Laget appendix for ziegler nichols, step respons og tuning.

- Testet og dokumentert mange tuning parametre.
- Laget nye funksjoner og funksjonsblokker i PLC for bruk til alle 3 sylindre.
- Laget protokoll dokument og endret hvordan data blir sendt i PLC/UI.

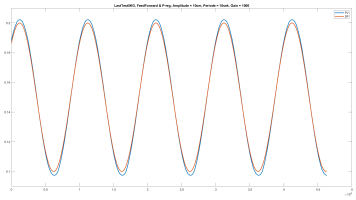
03/02/2023

- Faseforskyvet speed sp og tunet sylinder så bra som mulig med last.
- Laget appendix av alle resultat og dokumentert alt med .csv filer og MATLAB plot.
- Bilde under viser resultater av tuning med periode på 10sek og amplitude på 15cm.



02/02/2023

- Ferdigstilt og levert forprosjekt.
- Laget appendix for sinus test av sylinder for forskjellige tuning parametre og med 5 kg last.
- Speed kalibrert sylinder pånytt med 5 kg last og kjørt flere tester, logget til CSV filer og plottet i MATLAB.



31/01/2023

- Jobbet med forprosjekt.
- Satt oss inn i Latex.
- Laget tidsplan, budsjett og blokkdiagram.

30/01/2023

- Laget CSV filer for step respons med P-regulator.
- Laget CSV filer for FeedForward regulator uten P-reg.
- Laget program i C# og matlab for konvertering av data i CSV filene, slik at disse kan plottes.
- Kjørt ny speedkalibrering av sylinder da denne hadde begynt å drifte.

26/01/2023

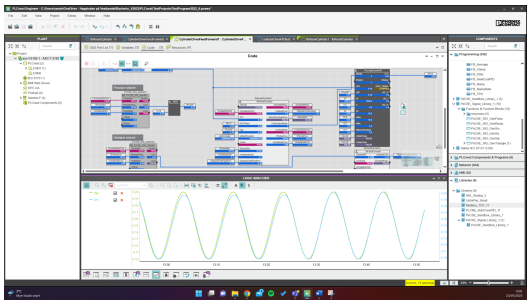
- Møte med veileder Endre fra 13-14.
- Lagt til kode for kommunikasjon mellom UI og PLC. Testet start/stopp/reset funksjoner fra UI.
- Laget struct klasser i PLC for DataToUI og DataFromUI

24/01/2023

- Laget funksjonsblokk for sinus bølgesimullering som gir ut posisjon-setpoint og speed-setpoint for alle 3 sylindre. Tar inn periode, amplitude og phase diffference.
- Tunet testsylinder og logget alle resultat. lagret .csv filer av tester.
- Implementert Modbus kode i UI og kjørt sylinder (Simulert bølge) fra Plattform UI.

23/01/2023

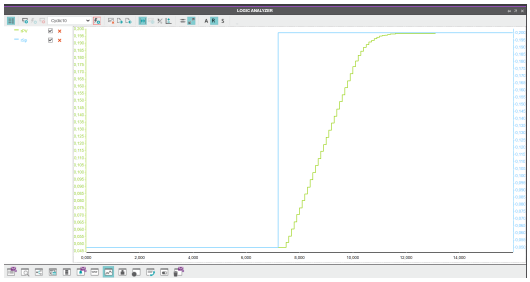
- Laget speedtest funksjonsblokk for kalibrering av feed forward loop.
- Kjørt speedtest og kalibrert test sylinder.
- Laget control funksjonsblokk for styring av sylinder med feed forward kombinert med PD-regulator.
- Testet og tunet regulering med og uten PD-regulator.

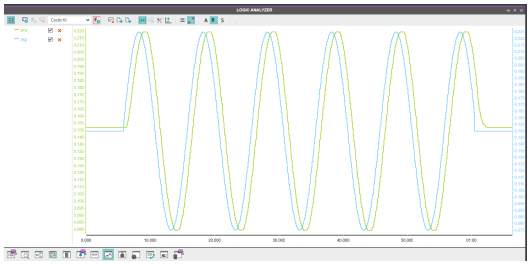


- Bildet over viser første halvdel uten PD-regulator (kun feed forward), andre del med PD-regulator innkoblet.

20/01/2023

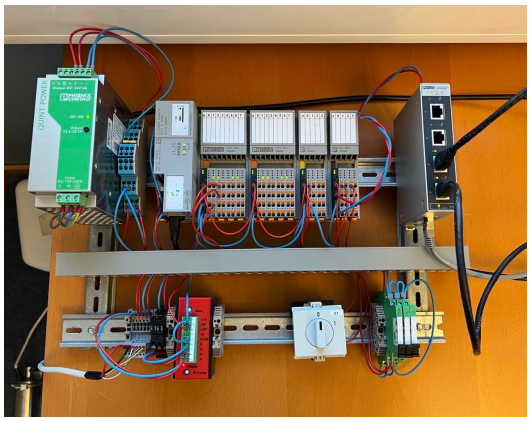
- Fått reset funksjonsblokk for å kjøre sylinder til hjem pos til å fungere.
- Kjørt step respons og sinus av sylinder med P-regulator. Logget resultat.





19/01/2023

- Koblet enable rele og bryter for styring av sylinder.
- Test kjørt sylinder fra PLC med p - regulator.
- Laget fb for resetting av sylinder til hjem possisjon.

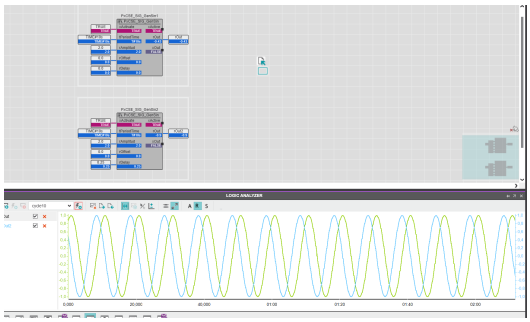


17/01/2023

- Koblet enkoder fra sylinder til PLC og fått inn pulser og konvertering til meter.
- Testet kjøring av sylinder med +-10v fra PLC.

16/01/2023

- Implimenterte sinus generator til PLC next og testet
- Mottatt amplitude, faseforskyvning og periodetid fra UI via MODBUS
- Koblet opp motordriver til sylinder og testkjørt.

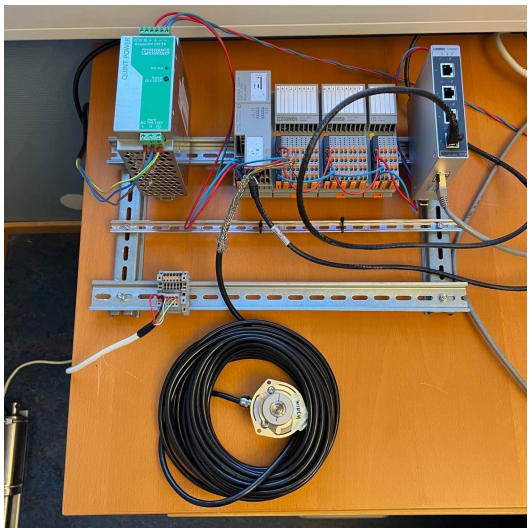


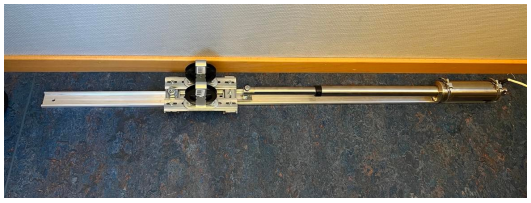
13/01/2023

- Lagt til x, y, z crane tip settings i plattform UI, og info til disse.
- Testet PID posisjonsregulering med encoder input og analog output.

12/01/2023

- Laget funksjonsblokk for encoder input scalering til meter i PLCnext.
- Testet modbus kommunikasjon med C# og PLCnext
- Koblet sylinder mot PLC





10/01/2023

- Programmert C# (Plattform UI)
- Koblet opp og fått inn encoder signal i PLCnext

09/01/2023

- Koblet opp PLS/Moduler, kommunikasjon mellom PLS og maskin OK. Satt oss inn i PLCnext Engineer software.

28/12/2022 - 30/12/2022

- Begynt på C# program for strying av plattform, og testet biblotek for "Live Chart" design og modbus kommunikasjon mot PLC.

21/12/22

- Oppstartsmøte for bachelor prosjekt på kontorlokalet.

10.6 Materialliste

Tabell 3 viser en oversikt over valgt materiell med referanse til nettbutikk.

Materialliste BO23EB-37			
Material	**Antall**	**Beskrivelse**	**Referanse**
Sylinder motor driver	3	3A DC motor driver	Link
PLCnext kontroller	1	PLCnext f 2152	Link
DI8/DI8	1	Digital I/O modul	Link
AO4	1	Analog utgangsmodul	Link
CNT2/INC2	2	Encoder inngangs modul	Link
3P rele	1	nødstopp rele	Link
solid state rele	3	sylinder enable rele	Link
strømforsyning	1	24V/20A strømforsyning	Link
switch	1	5port switch	Link

Tabell 3: Materialliste tabell

11 Appendiks C - Kommunikasjonsprotokoller

11.1 Kommunikasjonsprotokoll Plattform UI

Project

SCANTROL

PLATTFORM UI PROTOCOL DESCRIPTION MODBUS UDP

Document title:

PROTOCOL 01 MODBUS UDP

Customer:	**Supplier:**
Scantrol AS	**Lars Askild S. Aarvik** **Sander V. Andersen**
Customer project no: **BACHELOR_B023EB-37**	**Supplier project no:**

Code	**Reason for issue**	**Approved by client**
01A	ISSUED FOR INTERNAL DOCUMENT CONTROL	Date: (dd.mm.yy) _____ Sign: _____
02A	ISSUED FOR EXTERNAL DOCUMENT CONTROL	
03A	APPROVED FOR CONSTRUCTION	
04A	AS-DELIVERED	
05A	AS-BUILT	

01A	02.05.2023	ISSUED FOR INTERNAL DOCUMENT CONTROL	LA, SA		
Rev.	**Date(dd.mm.yy)**	**Reason for issue**	**Auth.**	**Checked**	**Appr.**

			No of attachments: 0	**No of pages: 5**

CONTENTS

1 HMI SETTINGS

To activate this protocol, these settings must be applied to HMI:

- Correct IP address to PLC – 192.168.127.213

2 COMMUNICATION PROTOCOL

2.1 PHYSICAL TRANSMISSION

Signal transmission is done by Ethernet network to PLC. Medium can be anything that supports UDP/IP transmissions.

2.2 MESSAGE PROTOCOL

Messages transfer is done on a non-complete implementation of Modbus UDP with HMI as communication client (master). Standard network port for Modbus UDP is 502.

2.2.1 Function codes

HMI use function codes 3, read holding registers and 16, write multiple registers, reading 28 words (56 bytes) of data from device using one read messages, and writing variable amount of words depending length of logfile data.

2.2.2 Endianness

The protocol is big-endian for bytes within words and little-endian for words within double-word values.
A 32-bit integer value 168496141 ($0A0B0C0D_h$) is transferred like this, with leftmost byte transferred first:

LSW/MSB	LSW/LSB	MSW/MSB	MSW/LSB
0C	0D	0A	0B

3 SIGNAL LISTS

3.1 TRANSMITTED DATA FROM HMI

Starting address is default as 100.

Word Address	Signal	Format	Value	Unit	Comment
100	Communication_counter	16bit	0-65 535		Counter for detecting com error
101	Operation Mode	16bit			0 = Manual 1 = Simulated 2 = Log file
102	Command	16bit			0 = Stop 1 = Start 2 = Reset
103	Amplitude	32bit	0-250	1/1000 meter	0 = 0 meter 250 = 0.25 meter
105	Period	32bit	0-60 000	ms	0 = 0 sec 60 000 = 60 sec
107	Phase difference	32bit	0-180	DEG	Phase difference between front and rear cylinders, when in simulated mode
109	Heave Position	32bit	0-300	1/1000 Meter	Heave position when in MANUAL mode.
111	Roll angle	32bit	+- 42	DEG	Roll platform angle in MANUAL mode.
113	Reset_Wire_LengthValue	32bit		1/1000 meter	Desired wire length reset value
115.01	Reset Wire Length command	16bit	0-1	BOOL	BOOL for resetting wire length
116…	C1_HeavePos	16bit		1/100000 meter	Heave position rear cylinder 1, when in Logfile mode.
117…	C1_Velocity	16bit		1/100000 meter	Velocity rear cylinder 1, when in Logfile mode.
118…	C2_C3_HeavePos	16bit		1/100000 [m/s]	Heave position front cylinder 2 & 3, when in Logfile mode.
119…	C2_C3_Velocity	16bit		1/100000 [m/s]	Velocity rear cylinder 2 & 3, when in Logfile mode.

TRANSMITTED DATA TO HMI

Starting address is default as 0.

Word Address	Signal	Format	Value	Unit	Comment
0	Position setpoint cylinder 1	32bit	0-300	1/1000 meter	Position setpoint for cylinder 1
2	Position setpoint cylinder 2	32bit	0-300	1/1000 meter	Position setpoint for cylinder 2
4	Position setpoint cylinder 3	32bit	0-300	1/1000 meter	Position setpoint for cylinder 3
6	Process value cylinder 1	32bit	0-300	1/1000 meter	Process value for cylinder 1
8	Process value cylinder 2	32bit	0-300	1/1000 meter	Process value for cylinder 2
10	Process value cylinder 3	32bit	0-300	1/1000 meter	Process value for cylinder 3
12.01	Running signal	16bit	0-1	BOOL	On off signal for platform running.
13.01	EM stop	16bit	0-1	BOOL	Emergency stop

Word Address	Signal	Format	Value	Unit	Comment
14.01	Drive error	16bit	0-1	BOOL	Error on drive
15	Cylinder 1 speed	32bit		1/100000 [m/s]	Speed for cylinder 1
17	Cylinder 2 speed	32bit		1/100000 [m/s]	Speed for cylinder 2
19	Cylinder 3 speed	32bit		1/100000 [m/s]	Speed for cylinder 3
21	Wire length	32bit		1/1000 meter	Winch wire length
23	Winch speed	32bit		1/1000 [RPM]	Winch speed in RPM
25	AHC center position	32bit		1/1000[m]	Center position for AHC
27	Operation mode winch	16bit	0-3	INT	0 = Standby winch 1 = Manual mode winch 2 = AHC mode winch 3 = Error

11.2 Kommunikasjonsprotokoll Vacon NXP drive

Project

SCANTROL

VACON NXP PROTOCOL DECRIPTION
MODBUS TCP

Document title:

PROTOCOL 02 MODBUS TCP

Customer:	**Supplier:**
Scantrol AS	**Lars Askild S. Aarvik** **Sander V. Andersen**
Customer project no: **BACHELOR_B023EB-37**	**Supplier project no:**

Code	**Reason for issue**	**Approved by client**
01A	ISSUED FOR INTERNAL DOCUMENT CONTROL	Date: (dd.mm.yy) _____ Sign: _____
02A	ISSUED FOR EXTERNAL DOCUMENT CONTROL	
03A	APPROVED FOR CONSTRUCTION	
04A	AS-DELIVERED	
05A	AS-BUILT	

01A	02.05.2023	ISSUED FOR INTERNAL DOCUMENT CONTROL	LA, SA		
Rev.	**Date(dd.mm.yy)**	**Reason for issue**	**Auth.**	**Checked**	**Appr.**

			No of attachments: 0	**No of pages: 4**

CONTENTS

1 VACON NXP DRIVE SETTINGS

To activate this protocol, these settings must be applied to Vacon NXP drive:
Drive is setup to “Closed loop speed control with torque limitation”:

Parameter	ID		Setting
P2.6.1		Motor Control Mode	Set to CL SpeedCtrl
P2.9.11		FB1 control parameter	Set to 609, Torque limit

2 COMMUNICATION PROTOCOL

2.1 PHYSICAL TRANSMISSION

Signal transmission is done by Ethernet network to PLC. Medium can be anything that supports TCP/IP transmissions.

2.2 MESSAGE PROTOCOL

Messages transfer is done on a non-complete implementation of Modbus TCP with drive as communication client (master). Standard network port for Modbus TCP is 502.

2.2.1 Function codes

PLC use function codes 23, read/write multiple holding registers, reading 8 words (16 bytes) of data from drive using one read messages, and writing 4 words(8 bytes) of data to drive.

3 SIGNAL LISTS

3.1 TRANSMITTED DATA FROM VACON NXP DRIVE

Starting address is default as 2100.

Word Address	Signal	Format	Value	Unit	Comment
2100.01	Ready	1 bit	0 - 1	BOOL	Drive is ready to run
2100.02	Run	1 bit	0 - 1	BOOL	Drive is running
2100.03	Counter clockwise	1 bit	0 - 1	BOOL	Drive motor running counter clockwise
2100.04	Fault	1 bit	0 - 1	BOOL	Drive fault
2100.05	Warning	1 bit	0 - 1	BOOL	Drive warning
2100.06	At reference speed	1 bit	0 - 1	BOOL	Motor is at reference speed
2100.07	At zero speed	1 bit	0 - 1	BOOL	Motor is at zero speed
2100.08	Flux ready	1 bit	0 - 1	BOOL	Drive flux is ready
2101	SPARE				
2102	SPARE				
2103	SPARE				
2104	Motor RPM	16 bit	+- 32767	INT	Motor RPM
2105	SPARE				
2106	SPARE				
2107	SPARE				

TRANSMITTED DATA TO VACON NXP DRIVE

Starting address is default as 2000.

Word Address	Signal	Format	Value	Unit	Comment
2000.01	Run command	1 bit	0 - 1	BOOL	Run command to drive
2000.02	Direction command	1 bit	0 - 1	BOOL	Direction command to drive
2000.03	Reset error command	1 bit	0 - 1	BOOL	Reset error command to drive
2001	SPARE				
2002	Speed command	16 bit	0-10000	1/100%	Speed command to drive
2003	Torque command	16 bit	0-10000	1/100%	Torque limit command to drive

11.3 Kommunikasjonsprotokoll Operatørpanel

Project

SCANTROL

OPERATORPANEL PROTOCOL DECRIPTION MODBUS TCP

Document title:

PROTOCOL 03 MODBUS TCP

Customer:	**Supplier:**
Scantrol AS	**Lars Askild S. Aarvik** **Sander V. Andersen**
Customer project no: **BACHELOR_B023EB-37**	**Supplier project no:**

Code	**Reason for issue**	**Approved by client**
01A	ISSUED FOR INTERNAL DOCUMENT CONTROL	Date: (dd.mm.yy) _____ Sign: _____
02A	ISSUED FOR EXTERNAL DOCUMENT CONTROL	
03A	APPROVED FOR CONSTRUCTION	
04A	AS-DELIVERED	
05A	AS-BUILT	

01A	02.05.2023	ISSUED FOR INTERNAL DOCUMENT CONTROL	LA, SA		
Rev.	**Date(dd.mm.yy)**	**Reason for issue**	**Auth.**	**Checked**	**Appr.**

			No of attachments: 0	**No of pages: 4**

CONTENTS

1 OPERATORPANEL SETTINGS

To activate this protocol, these settings must be applied to operatorpanel:
Operator panel must be on the same subnet as the PLC:

2 COMMUNICATION PROTOCOL

2.1 PHYSICAL TRANSMISSION

Signal transmission is done by Ethernet network to PLC. Medium can be anything that supports TCP/IP transmissions.

2.2 MESSAGE PROTOCOL

Messages transfer is done on a non-complete implementation of Modbus TCP with drive as communication client (master). Standard network port for Modbus TCP is 502.

2.2.1 Function codes

PLC use function codes 23, read/write multiple holding registers, reading 6 words (12 bytes) of data from drive using one read messages, and writing 1 words(2 bytes) of data to drive.

3 SIGNAL LISTS

3.1 TRANSMITTED DATA FROM OPERATORPANEL

Starting address is default as 0.

Word Address	Signal	Format	Value	Unit	Comment
0.B0	Communication counter	8 bit	0 - 255	Byte	Communication counter
01.15	EM stop	1 bit	0 - 1	BOOL	Emergency stop
01.14	Joystick error	1 bit	0 - 1	BOOL	True if joystick switch has error
01.06	Reset button	1 bit	0 - 1	BOOL	True if reset button is active
01.07	Standby button	1 bit	0 - 1	BOOL	True if standby button is active
01.08	AHC button	1 bit	0 - 1	BOOL	True if AHC button is active
01.10	InControl button	1 bit	0 - 1	BOOL	True if InControl button is active
01.11	Manual button	1 bit	0 - 1	BOOL	True if manual button is active
02	Joystick switch	16 bit	0 – 2	INT	0 = center position 1 = Pay IN 2 = Pay OUT
05	Potmeter 1	16 bit	0 – 1000	1/10%	Max speed potmeter [0 – 100%]
07	Temperatur	16 bit	+- 32767	INT	Operator panel temp
08	Joystick 1	16 bit	+- 1000	1/10%	Control joystick

TRANSMITTED DATA TO OPERATORPANEL

Starting address is default as 20.

Word Address	Signal	Format	Value	Unit	Comment
20.06	Reset button light	1 bit	0 - 1	BOOL	Activates light if true
20.07	Standby button light	1 bit	0 - 1	BOOL	Activates light if true
20.08	AHC button light	1 bit	0 - 1	BOOL	Activates light if true
20.10	InControl button light	1 bit	0 – 1	BOOL	Activates light if true
20.11	Manual button light	1 bit	0 - 1	BOOL	Activates light if true
20.15	Buzzer	1 bit	0 - 1	BOOL	Alarm buzzer

12 Appendiks D - Tuning

12.1 Lastavhengighet

Tidlig i arbeidet undersøkte vi lastavhengigheten til sylindrene. Dette ble gjort ved å feste ett lodd på 5kg på sylinderen, for så å sjekke nøyaktigheten og forholdet mellom settpunkt og er-verdi. På figur1(bilde - sylinder uten last. Samme G,A,P)

12.2 Hastighetskalibrering sylindre

Hastighets-kalibrerings testen ble gjennomført ved ta tiden sylindrene bruker på å kjøre 20 cm for så å beregne hastigheten i [m/s]. [+V] kjører sylindrene innover og [-V] kjører sylindrene utover.

Speed calibration cylinder 1			Speed calibration cylinder 2			Speed calibration cylinder 3		
	SpdCMD +- 10 V	Speed m/s		SpdCMD +- 10 V	Speed m/s		SpdCMD +- 10 V	Speed m/s
	-0.5	0.000000		-0.5	-0.001950		-0.5	-0.002350
	-1	-0.006040		-1	-0.008680		-1	-0.008760
	-1.5	-0.012820		-1.5	-0.015440		-1.5	-0.015660
	-2	-0.019690		-2	-0.022230		-2	-0.022450
	-2.5	-0.026450		-2.5	-0.029020		-2.5	-0.029190
	-3	-0.033220		-3	-0.035760		-3	-0.035950
	-3.5	-0.039810		-3.5	-0.042520		-3.5	-0.042780
	-4	-0.044300		-4	-0.048370		-4	-0.049080
	-4.5	-0.046680		-4.5	-0.050310		-4.5	-0.050700
	-5	-0.046790		-5	-0.050790		-5	-0.050820
	-5.5	-0.046880		-5.5	-0.050790		-5.5	-0.050920
	-6	-0.046900		-6	-0.050820		-6	-0.050950
	-6.5	-0.046900		-6.5	-0.050820		-6.5	-0.050950
	-7	-0.046900		-7	-0.050820		-7	-0.050950
	-7.5	-0.046900		-7.5	-0.050820		-7.5	-0.050950
	-8	-0.046900		-8	-0.050820		-8	-0.050950
	-8.5	-0.046900		-8.5	-0.050820		-8.5	-0.050950
	-9	-0.046900		-9	-0.050820		-9	-0.050950
	-9.5	-0.046900		-9.5	-0.050820		-9.5	-0.050950
	-10	-0.046900		-10	-0.050820		-10	-0.050950
	0.5	0.007520		0.5	0.007160		0.5	0.006760
	1	0.014010		1	0.013670		1	0.013190
	1.5	0.020410		1.5	0.020110		1.5	0.019500
	2	0.026740		2	0.026420		2	0.025670
	2.5	0.032940		2.5	0.032770		2.5	0.031740
	3	0.039040		3	0.039110		3	0.037640
	3.5	0.045100		3.5	0.045200		3.5	0.043630
	4	0.049290		4	0.048720		4	0.047230
	4.5	0.050660		4.5	0.050060		4.5	0.048140
	5	0.051080		5	0.050540		5	0.048600
	5.5	0.051210		5.5	0.050820		5.5	0.048720
	6	0.051480		6	0.050950		6	0.048960
	6.5	0.051570		6.5	0.051080		6.5	0.048960
	7	0.051610		7	0.051080		7	0.049080
	7.5	0.051610		7.5	0.051080		7.5	0.049080
	8	0.051610		8	0.051080		8	0.049080
	8.5	0.051610		8.5	0.051080		8.5	0.049080
	9	0.051610		9	0.051080		9	0.049080
	9.5	0.051610		9.5	0.051080		9.5	0.049080
	10	0.051610		10	0.051080		10	0.049080

Figur 12.1: Utklipp av hastighets kalibreringtabell som viser hastighet i [m/s] med hastighetskommandoer mellom +-10[V]

12.3 Hastighetskalibrering vinsj

SpeedCMD (%)	Pay in [rpm]	[m/s]
0	0	0
10	18	0.0834
20	36	0.1668
30	54	0.2502
40	71.9	0.33313667
50	89.9	0.41653667
60	108	0.5004
70	126	0.5838
80	143.9	0.66673667
90	161.9	0.75013667
100	179.9	0.83353667

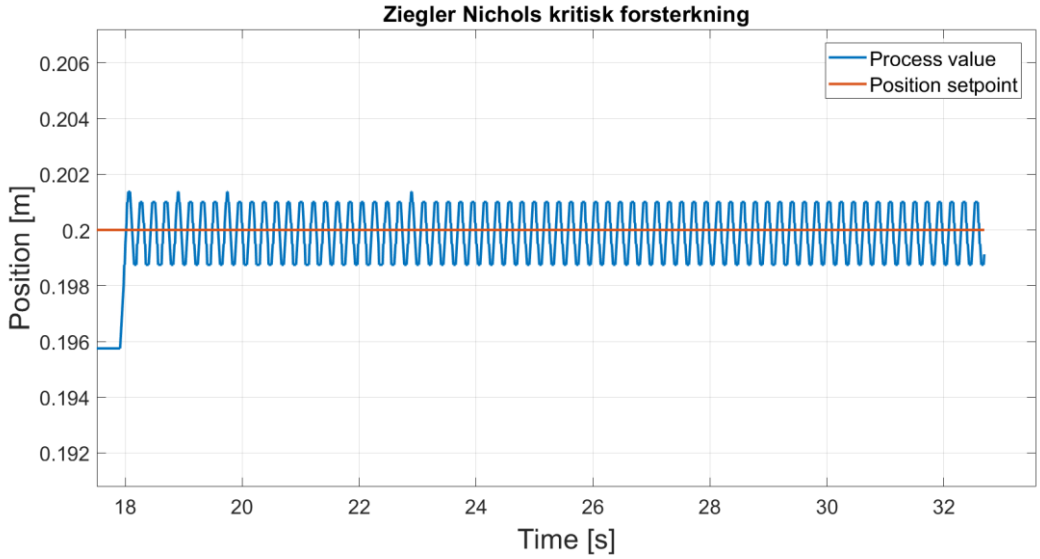
SpeedCMD (%)	Pay out [rpm]	[m/s]
0	0	0
10	-17.9	-0.08293667
20	-36	-0.1668
30	-54	-0.2502
40	-72	-0.3336
50	-89.9	-0.41653667
60	-108	-0.5004
70	-126	-0.5838
80	-144	-0.6672
90	-162	-0.7506
100	-180	-0.834

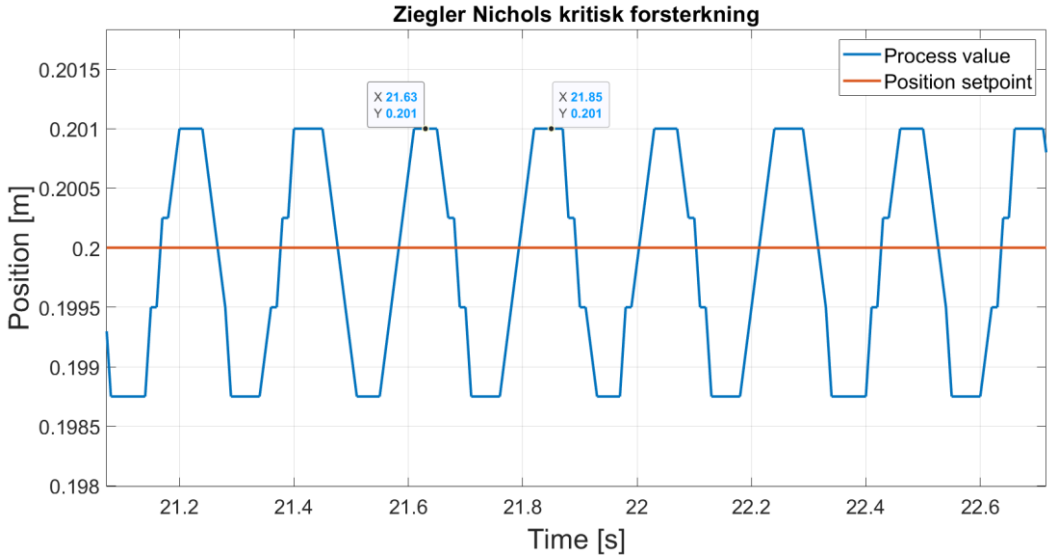
Figur 12.2: Utklipp av hastighets kalibreringtabell som viser hastighet til vinsj i [m/s] og [RPM] med pådrag på frekvensomformer mellom [0-100%]

12.4 Ziegler Nichols første metode

Gjennomførte Ziegler Nichols første metode for å finne ett utgangspunkt for regulator parameterne.

Vi testkjørte sylinderen med gain på 50 000[V/m], som skapte stabile oscillasjoner.





Måler periodetiden til 0,22 sekund. Som blir brukt til videre utregninger under.

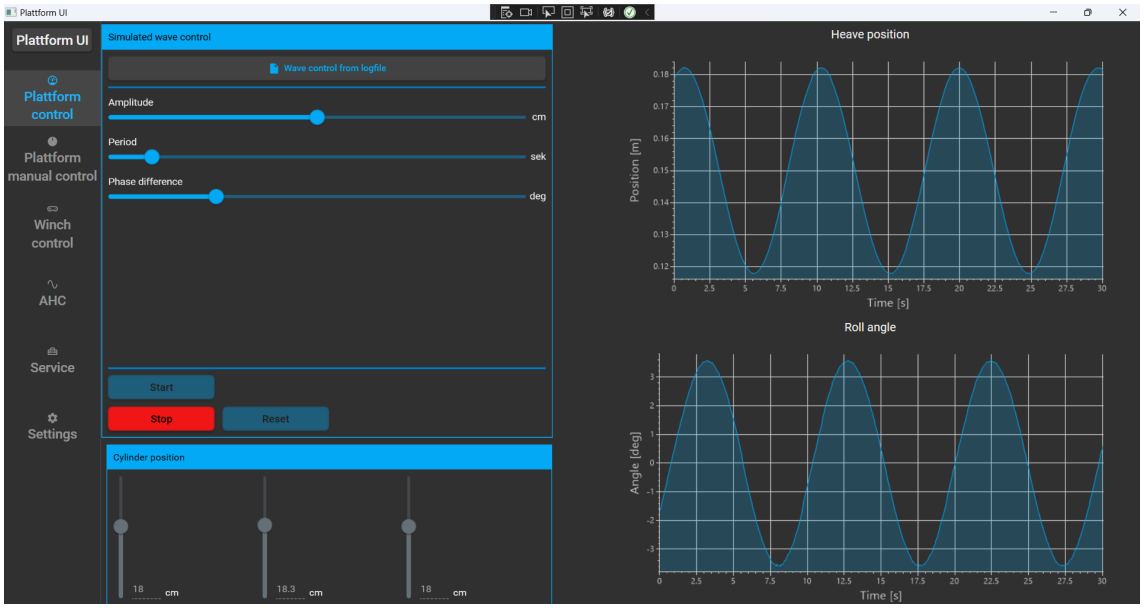
<u>P – regulator:</u>

P = 0.5 * KK => 0.5 * 50 000[V/m] = <u>25 000[V/m].</u>

<u>Bruker 25 000[V/m] i gain som utgangspunkt for videre step-responstester.</u>

13 Appendiks E - Bruksanvisning Plattform UI

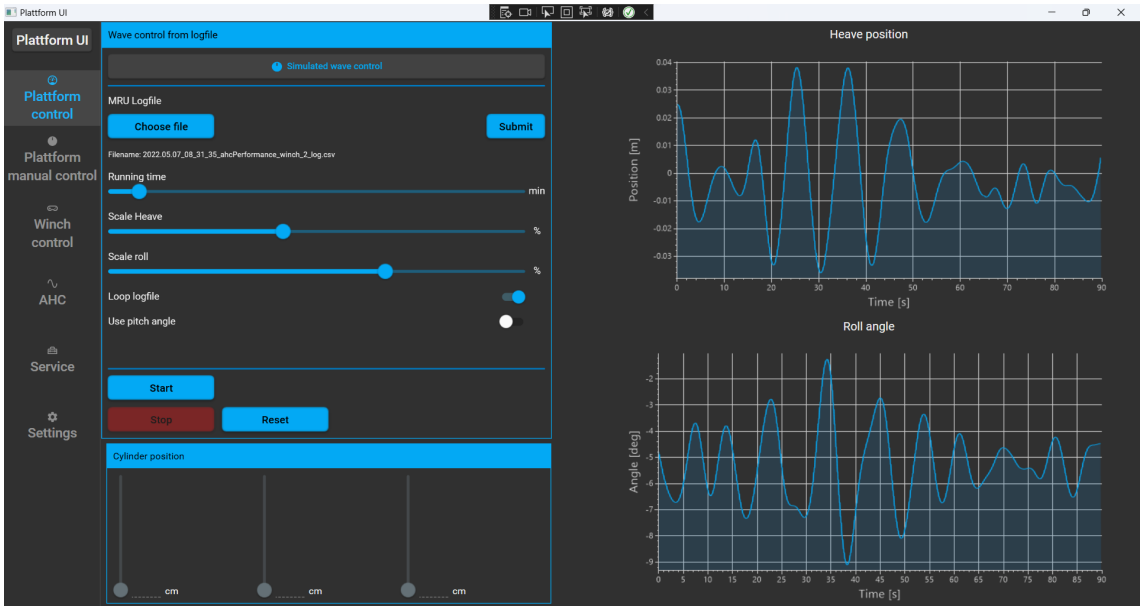
Bruksanvisning av plattform UI inneholder en kort beskrivelse av hvordan brukergrensesnittet fungerer og anvendes.



Figur 13.1: Skjermbilde av «Plattform control» tab i simulert bølge modus

Figur 13.1 viser er skjermbilde av «Plattform control» tab med simulert bølge modus valgt. Her er det mulig å kjøre plattform etter simulerte sinus bølger.

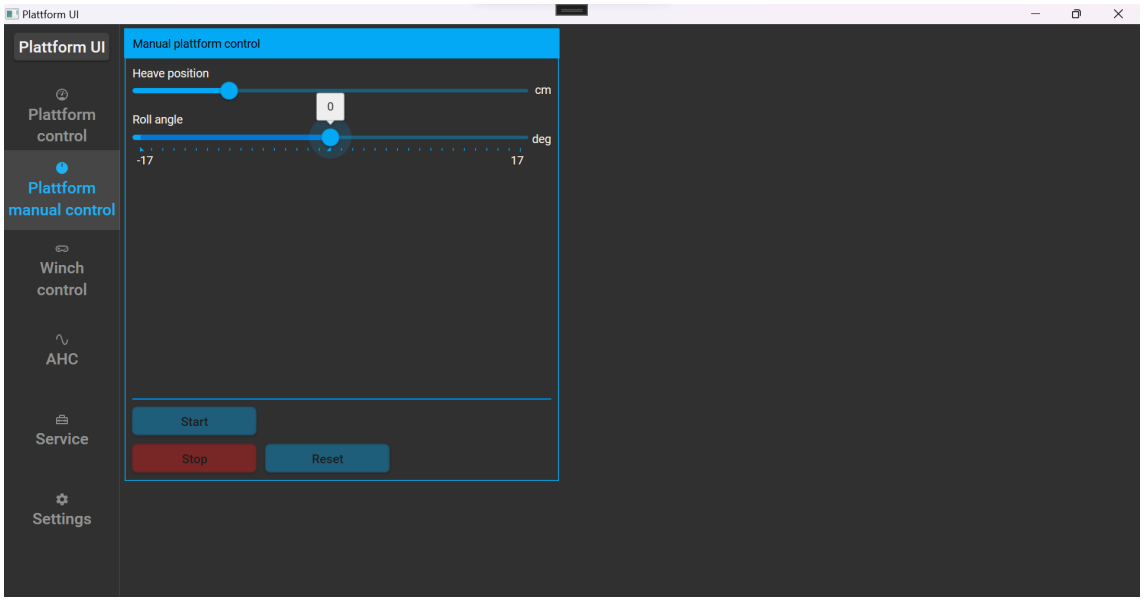
- **Wave control form logfile**: Endrer brukergrensesnitt og modus til plattform kontroll fra MRU loggfil.
- **Amplitude**: velg amplitude på sinus bølge (0 - 30cm).
- **Period**: Velg periode på sinus bølge (0 - 60sek).
- **Phase difference**: Velg faseforskjell for sinus på fremre og bakre sylindre. Ved 0° kjører alle sylindre likt, og ved 180° kjører de med motsatt fase (0 - 180°).
- **Cylinder position**: Viser posisjonene for hver sylinder til en hver tid.
- **Heave posistion**: Graf som viser heave position til plattformen.
- **Roll angle**: Graf som viser roll angle til palttformen.
- **Start**: Knapp som starter å kjøre plattformen med ønskede parametre.
- **Stop**: Knapp som stopper plattformen.
- **Reset**: Knapp som resetter plattformen. Da kjører alle sylindre til 0cm og enkoder telling resettes til 0.



Figur 13.2: Skjermbilde av «Plattform control» tab i loggfil bølge modus

Figur 13.2 viser er skjermbilde av «Plattform control» tab med loggfil modus valgt. Her er det mulig å kjøre plattform etter ekte MRU loggfil data.

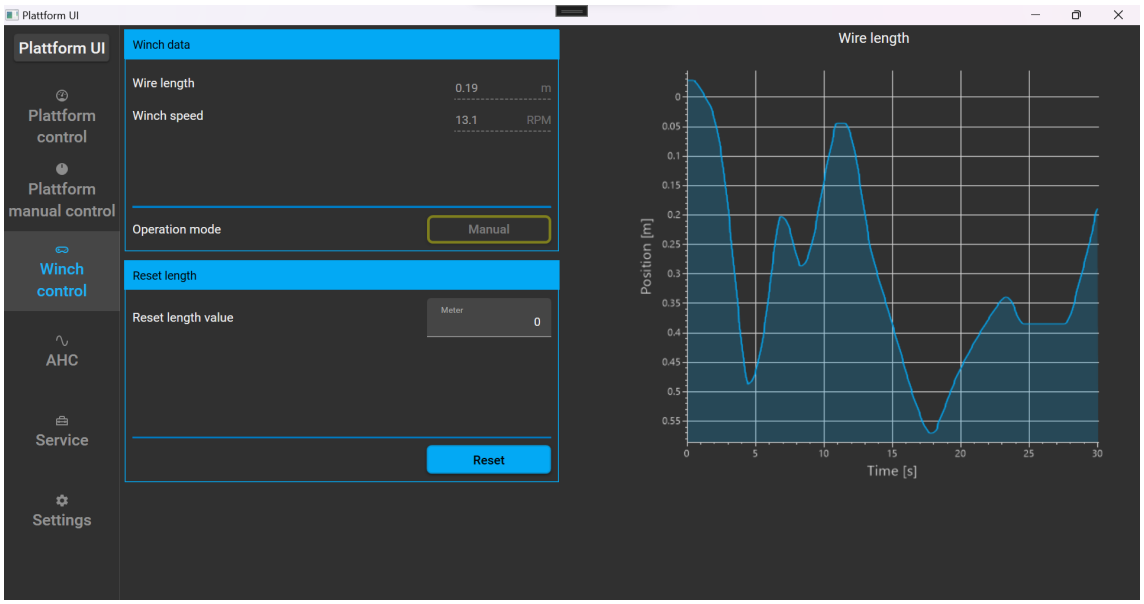
- **Simulated wave control**: Endrer brukergrensesnitt og modus til plattform kontroll med simulerte sinus bølger.
- **Choose file**: Knapp som åpner filutforsker og mulighet for å velge loggfile med .csv format.
- **Submit**: Knapp for å bekrefte valg av loggfil parametre før sylinder poisjoner beregnes og sendes til PLS.
- **Running time**: Mulighet for å velge hvor mye av filen som skal kjøres. Skalaen går fra 0 - 25min eller max lenged på loggfil dersom den er mindre enn 25min.
- **Scale heave**: Denne settes automatisk til bergenet max etter fil er valgt. Slideren viser skalert heave position i prosent i forhold til ekte bølger i loggfil.
- **Scale roll**: Mulighet for å skalere opp/ned roll angle avhenging av hvor mye heave er skalert ned. Mulig å skalere mer opp jo mer heave er skalert ned pga. posisjon/hastighets begrensninger.
- **Loop logfile**: Toogle bryter for å velge hva som skjer når valgt lengde av loggfil er kjørt. Dersom denne ikke er valgt, stopper plattformen og resettes automatisk. Dersom den er valgt fortsetter den å kjøre fra begynnelsen av filen igjen.
- **Use pitch angle**: Toggle knapp for å velge om roll eller pitch skal brukes som plattform vinkel.
- **Cylinder position**: Viser posisjonene for hver sylinder til en hver tid.
- **Heave posistion**: Graf som viser heave position til plattformen. Før brukeren trykker start viser den også heave position fra valgt loggfil, som oppdateres ettersom brukeren endrer parametre.
- **Roll angle**: Graf som viser roll angle til plattformen. Før brukeren trykker start viser den også roll/pitch fra valgt loggfil, som oppdateres ettersom brukeren endrer parametre.
- **Start**: Knapp som starter å kjøre plattformen med ønskede parametre. Kanppen har en prograssbar implementer for å vise status for sending av loggfil data til PLS. Når den er ferdig å sende alt, starter plattformen å kjøre.
- **Stop**: Knapp som stopper plattformen.
- **Reset**: Knapp som resetter plattformen. Da kjører alle sylindre til 0cm og enkoder telling resettes til 0.



Figur 13.3: Skjermbilde av «Plattform manual control» tab

Figur 13.3 viser er skjermbilde av «Plattform manual control» tab med manuell modus valgt. Her er det mulig å kjøre plattformen til ønsket posisjon.

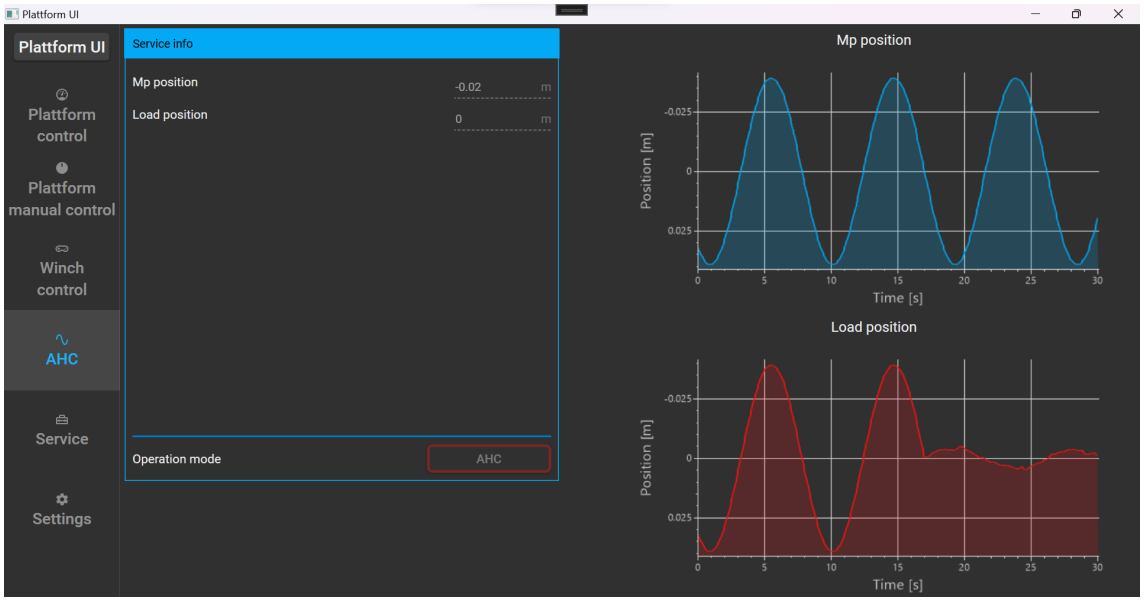
- **Heave position**: Her velger man hvilke heave posisjon [0-30cm] man ønsker å kjøre plattformen til.
- **Roll angle**: Her velger man hvilke vinkel man ønsker å kjøre plattformen i, størst lovlig vinkel er når plattformen har heave posisjon 15[cm].
- **Start**: Knapp som kjører plattformen til ønskede parametre.
- **Stopp**: Knapp som stopper plattformen
- **Reset**: Knapp som resetter plattformen. Da kjører alle sylindre til 0cm og enkoder telling resettes til 0.



Figur 13.4: Skjermbilde av «Winch control» tab

Figur 13.4 viser skjermbilde av «Winch control» tab. Her kan man resette vinsj lengden til ønsket verdi og avlese «live» posisjon og hastighet på vinsjen.

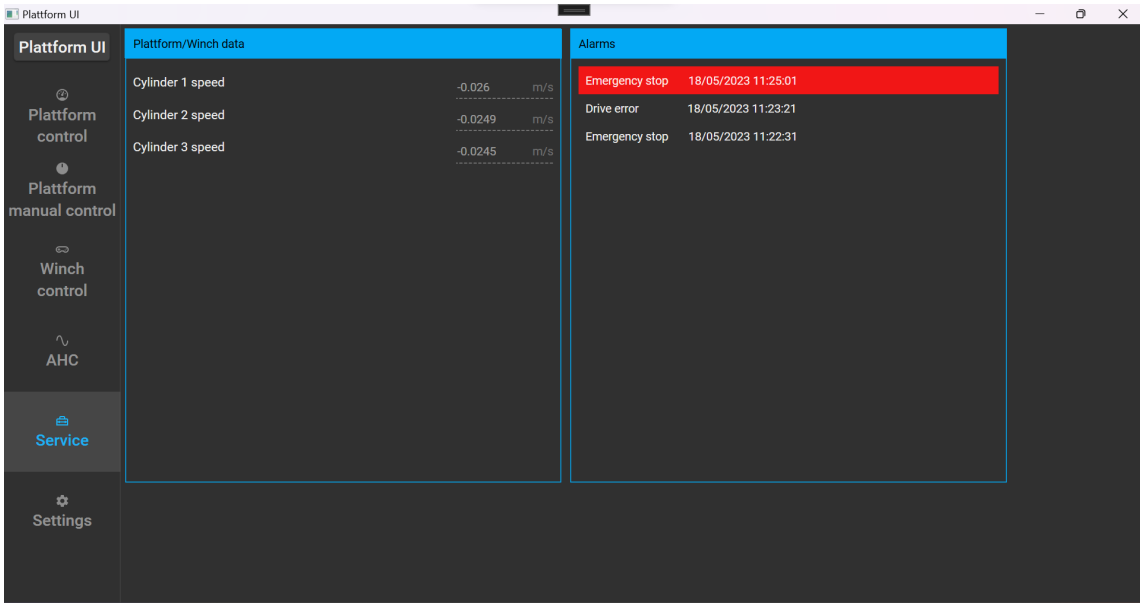
- **Wire length**: Her vises er-verdi posisjonen til vaier lengden i [m].
- **Wire speed**: Her vises hastigheten til vinsjen gitt i [RPM].
- **Operation mode**: Viser hvilket operasjonsmodus vinsjen er i med tekst kombinert med farge.
- **Reset wire length**: Her velger man ønsket verdi i [m] for hva vaierlengden skal re-settes til.
- **Wire length plot**: Graf viser vaier lengden til en hver tid.



Figur 13.5: Skjermbilde av «AHC» tab

Figur 13.5 viser skjermbilde av «AHC» tab. Her kan man se bevegelsene på krantuppen samt lasten og se ytelsen til AHC sysytemet.

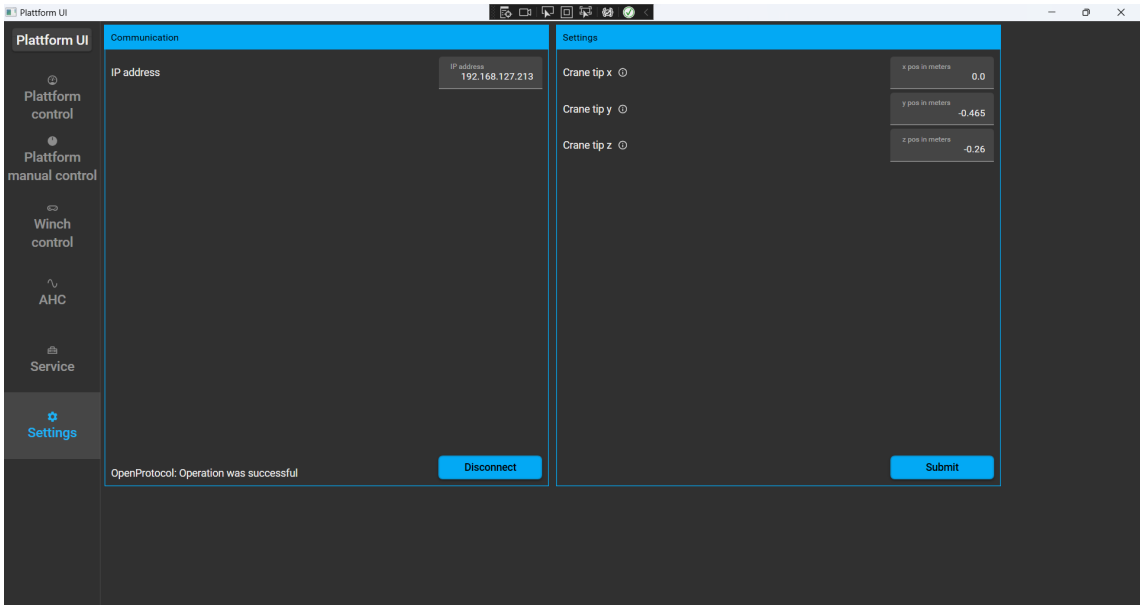
- **Mp position**: Viser posisjonsendringen til krantuppen i meter.
- **Load position**: Viser posisjonsendringen til lasten i meter.
- **Operation mode**: Viser hvilket operasjonsmodus vinsjen er i med tekst kombinert med farge.
- **Mp position plot**: Plot som viser bevegelsen til krantuppen (målepunktet) grafisk.
- **Load position plot**: Plot som viser bevegelsen til lasten koblet til vinsjen grafisk. Ved optimal AHC står den tilnærmet lit 0[m].



Figur 13.6: Skjermbilde av «Service» tab

Figur 13.6 viser skjermbilde av «Service» tab. Her vises diverse plattform data og alarmer.

- **Cylinder 1 Speed**: Viser hastigheten til sylinder 1 i [m/s].
- **Cylinder 2 Speed**: Viser hastigheten til sylinder 2 i [m/s].
- **Cylinder 3 Speed**: Viser hastigheten til sylinder 3 i [m/s].
- **Alarms**: Viser en liste over alarmhistorikk med tidspunkt samt fremheving av aktive alarmer med rød bakgrunn.



Figur 13.7: Skjermbilde av «Settings» tab

Figur 13.7 viser skjermbilde av «Settings» tab. Denne delen av brukergrensesnittet inneholder innstillinger for systemet og kommunikasjonen. Den ene delen av skjermen inneholder innstillinger for kommunikasjonen og den andre for generelle innstillinger.

- **IP adress**: Her skriver man inn IP-adressen til PLC (initialverdi: 192.168.127.213).

- **Connect**: Knapp som starter kommunikasjon over modbus på valgt IP-nettverk. Tekste endres til «Disconnect» dersom HMI er koblet til PLS, og det er da mulig å koble fra PLS med samme knapp.
- **Crane tip x**: Her er det mulig å sette inn lengder til ønsket krantip dersom brukeren ønsker å se bevegelsene i krantippen med større/mindre avstand enn på plattformen. Initital verdiene er satt til avstandene på plattformen. Lengde til krantip i x - retning (initialverdi: 0.0m).
- **Crane tip y**: Lengde til krantip i y - retning (initialverdi: -0.465m).
- **Crane tip z**: Lengde til krantip i z - retning (initialverdi: -0.26m).
- **Submit**: Knapp som setter valgte data i «crane tip» posisjon x,y,z.

14 Appendiks F - Elektriske tegninger

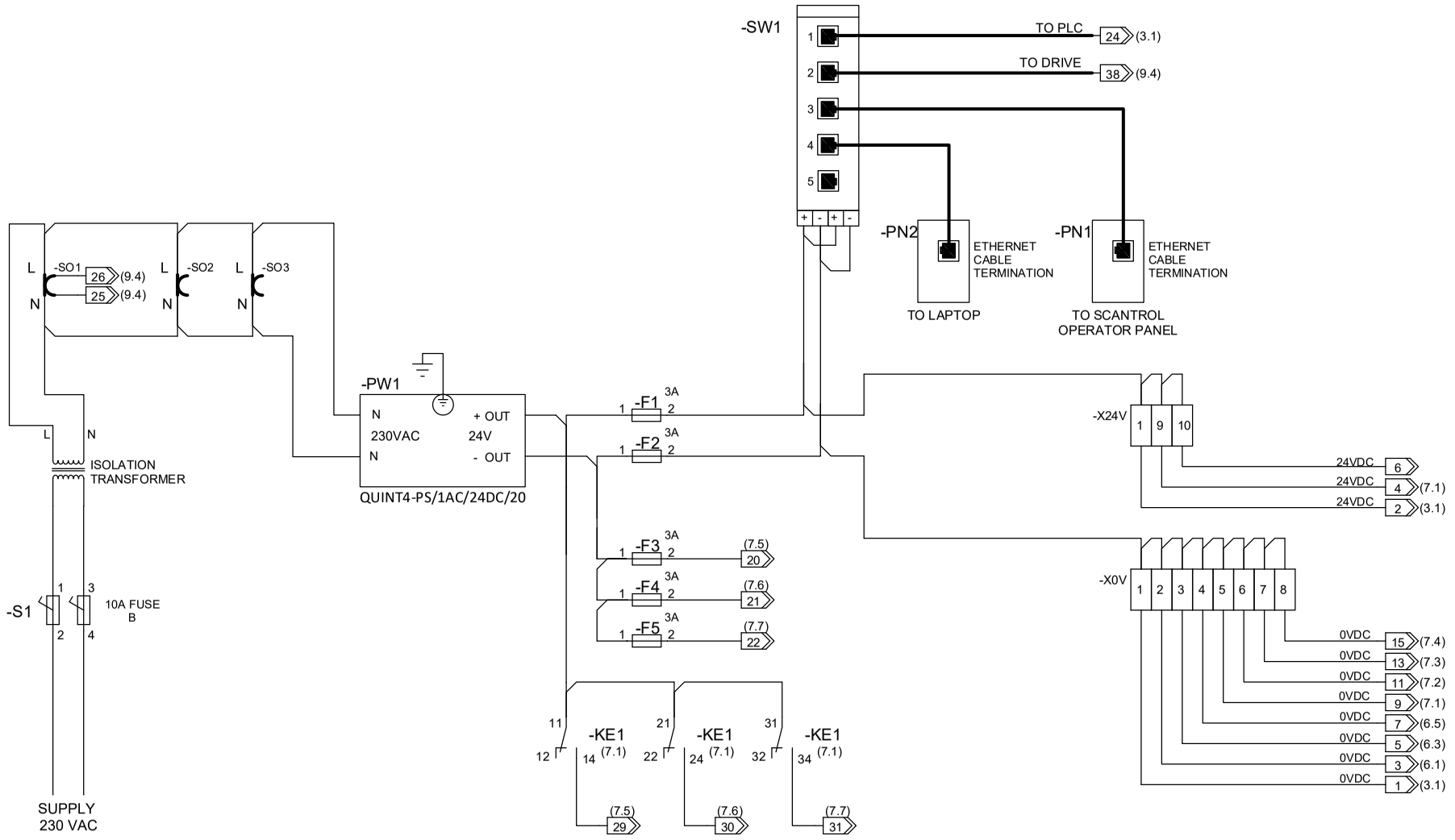
PROJECT: **Wave Simulation Platform**	CUSTOMER:
DOCUMENT TITLE: **Internal Wiring** B023EB-37-Bachelor	SUPPLIER: **Scantrol AS** **Sandviksbodene 1C, 5035 Bergen** **NORWAY**
CUSTOMER PROJECT #:	SUPPLIER PROJECT #:

Code	Reason for issue	Approved by client
01A	ISSUED FOR INTERNAL DOCUMENT CONTROL	**Date:** (dd.mm.yy) _____
02A	ISSUED FOR EXTERNAL DOCUMENT CONTROL	
03A	APPROVED FOR CONSTRUCTION	**Sign:** _____
04A	AS-DELIVERED	
05A	AS-BUILT	

Rev.	Date (dd.mm.yy)	Reason for issue	Auth.	Checked	Appr.
05A	14.03.2023	AS-BUILT	SA, LA	SA, LA	SA, LA

Ver.	Revision	Date	Sign.

Date	14.03.2023	Scantrol Wave Simulation Plattform	SCANTROL	INTERNAL WIRING	Project no. **B023EB** / Ver.
Perf. by	SA				
Appr. by					Scale 1:1 A3 / Drawing no. **B023EB-37-Bachelor** / Page 1
Norm		Original / Repl. for / Repl. of			N.pg. 2



				Date	14.03.2023	Scantrol Wave Simulation Plattform		SCANTROL	POWER DISTRIBUTION	Project no.	Ver.
				Perf. by	SA				INTERNAL WIRING	B023EB	
				Appr. by						Scale / Drawing no.	Page 2
Ver.	Revision	Date	Sign.	Norm		Original	Repl. for	Repl. of		1:1 A3 / B023EB-37-Bachelor	N.pg. 3

PLC IP: 192.168.127.20

-PWR1

(2.6) 24

X1

X2

Term.	X-ref
a1	(3.1)
a2	(3.2)
b1	(3.2)
b2	(3.2)

AXC F 2152

2 Ch. Counter Inputs

CNT2 -P1

Term. a	X-ref	Term. 0	X-ref	Term. 1	X-ref	Term. 2	X-ref	Term. 3	X-ref	Term. 4	X-ref	Term. 5	X-ref	Term. 6	X-ref	Term. 7	X-ref
a1	(3.3)	00	(6.1)	01	(6.3)	02	(4.3)	03		04		05		06	(4.5)	07	
a2	(3.4)	10		11		12	(4.4)	13		14		15		16	(4.6)	17	
b1	(3.3)	20	(6.2)	21	(4.1)	22	(6.6)	23		24		25		26		27	
b2	(3.4)	30		31	(4.2)	32		33		34		35		36		37	

AXL F CNT2 INC2 1F

2 Ch. Counter Inputs

CNT2 -P2

Term. a	X-ref	Term. 0	X-ref	Term. 1	X-ref	Term. 2	X-ref	Term. 3	X-ref	Term. 4	X-ref	Term. 5	X-ref	Term. 6	X-ref	Term. 7	X-ref
a1	(3.5)	00		01		02	(5.1)	03		04		05		06	(5.5)	07	(5.7)
a2	(3.5)	10		11		12	(5.2)	13		14		15		16	(5.6)	17	(5.8)
b1	(3.5)	20		21		22		23		24		25	(5.3)	26		27	
b2	(3.5)	30		31		32		33		34		35	(5.4)	36		37	

AXL F CNT2 INC2 1F

4 CH. AN. OUTPUT

AO4 -P3

Term. a	X-ref	Term. 0	X-ref	Term. 1	X-ref	Term. 2	X-ref	Term. 3	X-ref
a1	(3.6)	00		01		02		03	
a2	(3.6)	10		11		12		13	
b1	(3.6)	20		21		22		23	
b2	(3.6)	30		31		32		33	

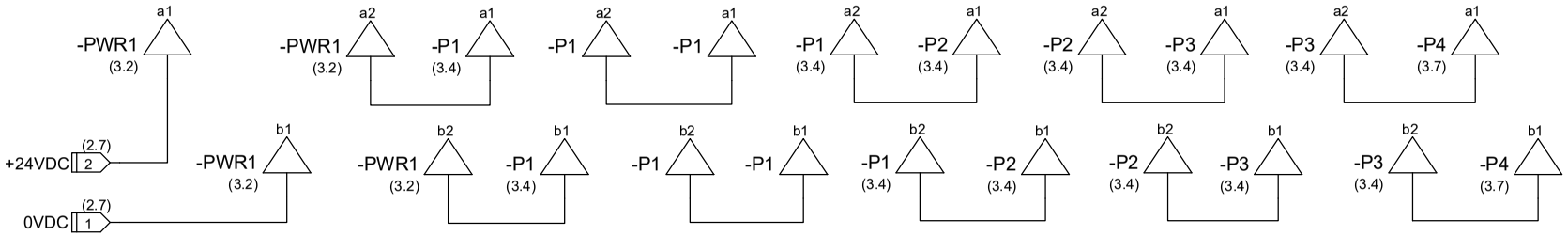
AXL F AO4 1H

8 CH. DIG. INPUT / 8 CH. DIG. OUTPUT

DI8/DO8 -P4

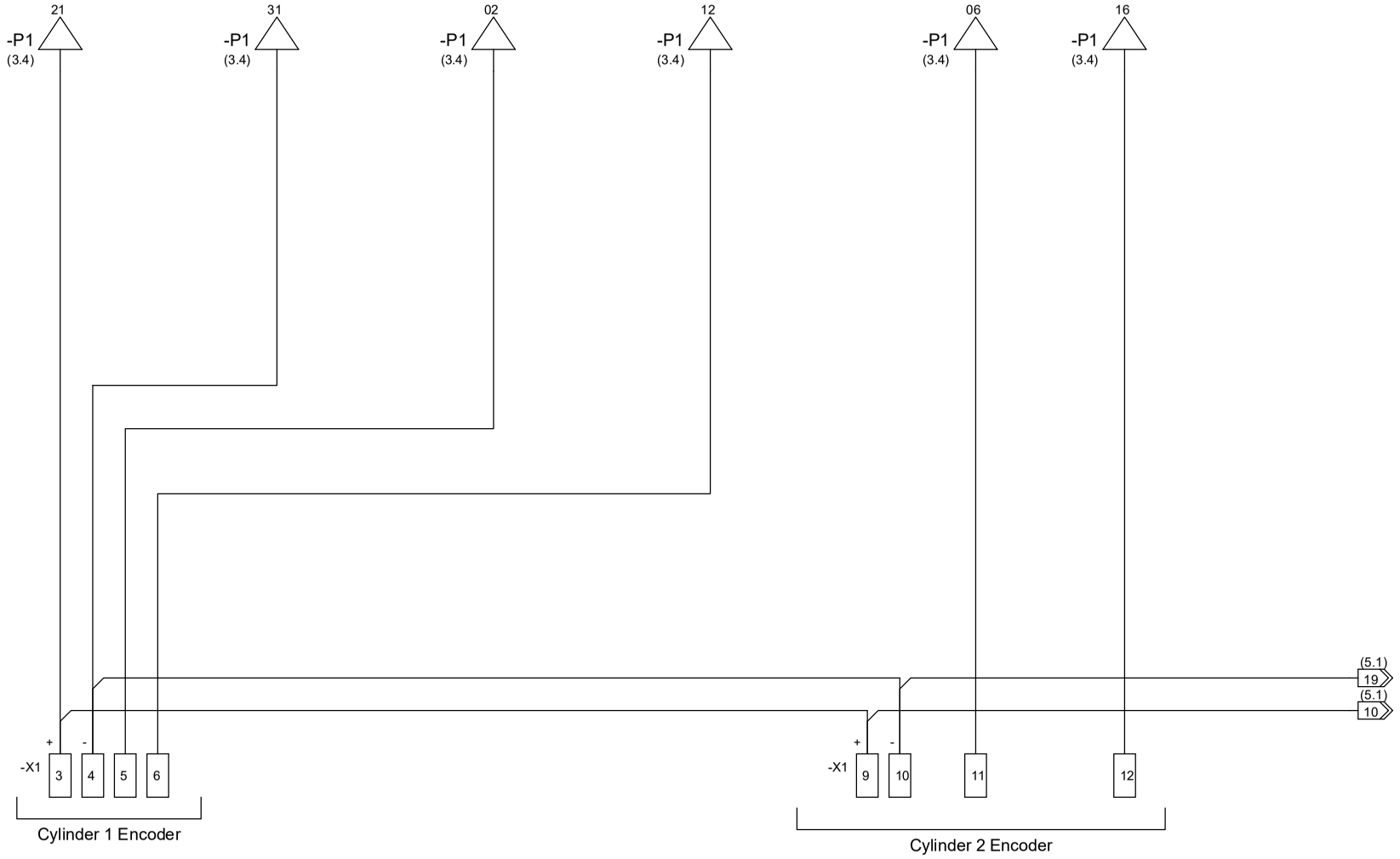
Term. a	X-ref	Term. 0	X-ref	Term. 1	X-ref	Term. 2	X-ref	Term. 3	X-ref
a1	(3.7)	00	(7.1)	01		02	(7.2)	03	
a2		10		11		12		13	
b1	(3.7)	20		21		22		23	
b2		30		31		32		33	

AXL F DI8/1 DO8/1 1H



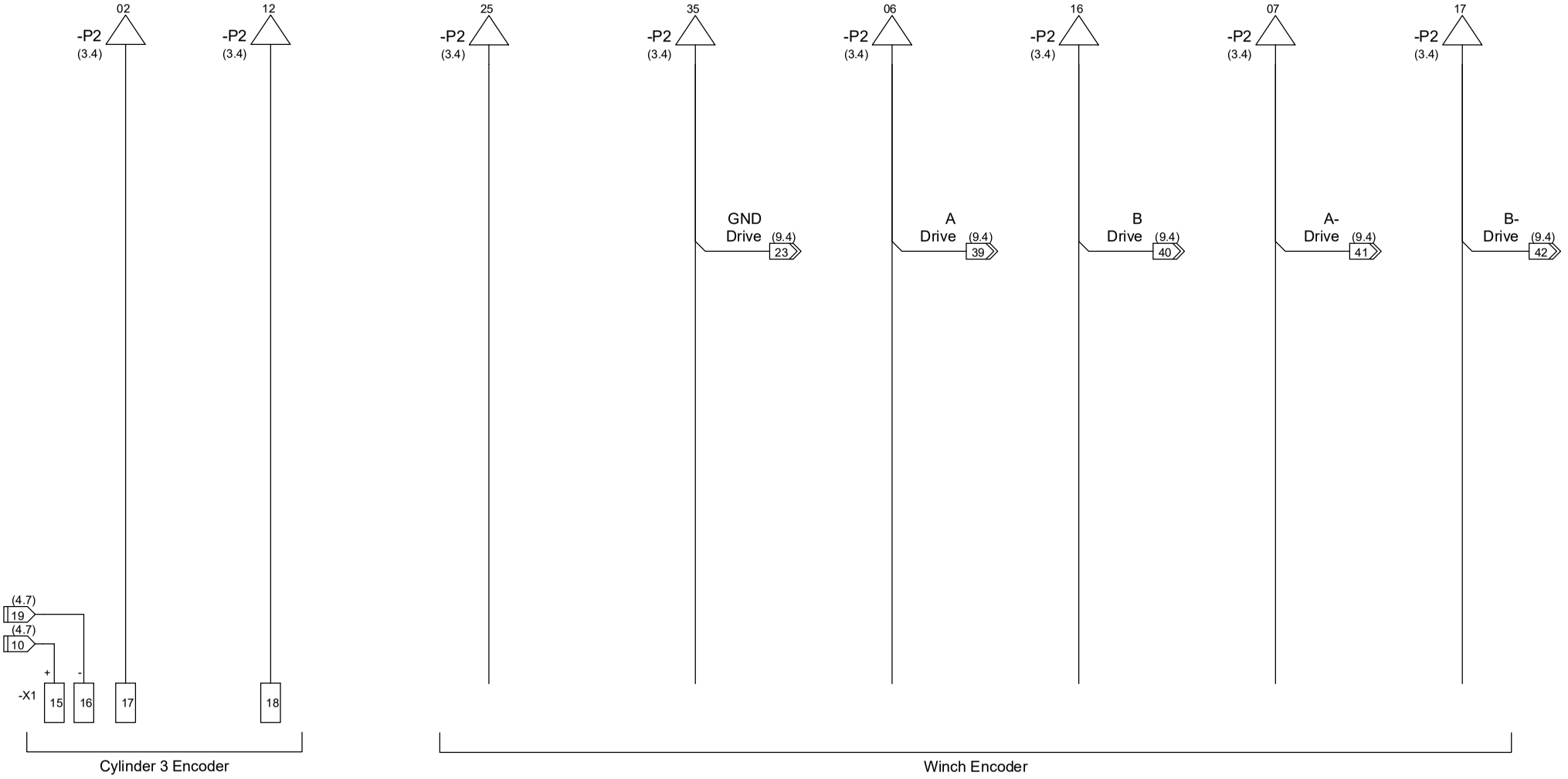
				Date	14.03.2023	Scantrol Wave Simulation Plattform		PLC REFERENCE	Project no.	Ver.
				Perf. by	SA		SCANTROL	INTERNAL WIRING	B023EB	
				Appr. by					Scale	Drawing no.
Ver.	Revision	Date	Sign.	Norm		Original	Repl. for	Repl. of	1:1 A3	B023EB-37-Bachelor

US1	GND	A1	INC1 - B1	INC2 - A2	INC2 - B2		
COUNTER 1	COUNTER 1	COUNTER 1	COUNTER 1	COUNTER 2	COUNTER 2		



				Date	14.03.2023	Scantrol Wave Simulation Plattform			SCANTROL	COUNTER INPUTS INTERNAL WIRING	Project no. B023EB / Ver.
				Perf. by	SA						
				Appr. by							Scale 1:1 A3 / Drawing no. B023EB-37-Bachelor / Page 4
Ver.	Revision	Date	Sign.	Norm		Original	Repl. for	Repl. of			N.pg. 5

INC1 - A1	INC1 - B1	US2	GND	INC2 - A2	INC2 - B2	INC2 - /A2	INC2 - /B2
COUNTER 1	COUNTER 1	COUNTER 2	COUNTER 2	COUNTER 2	COUNTER 2	COUNTER 2	COUNTER 2



Ver.	Revision	Date	Sign.

Date	14.03.2023
Perf. by	SA
Appr. by	
Norm	

Scantrol Wave Simulation Plattform

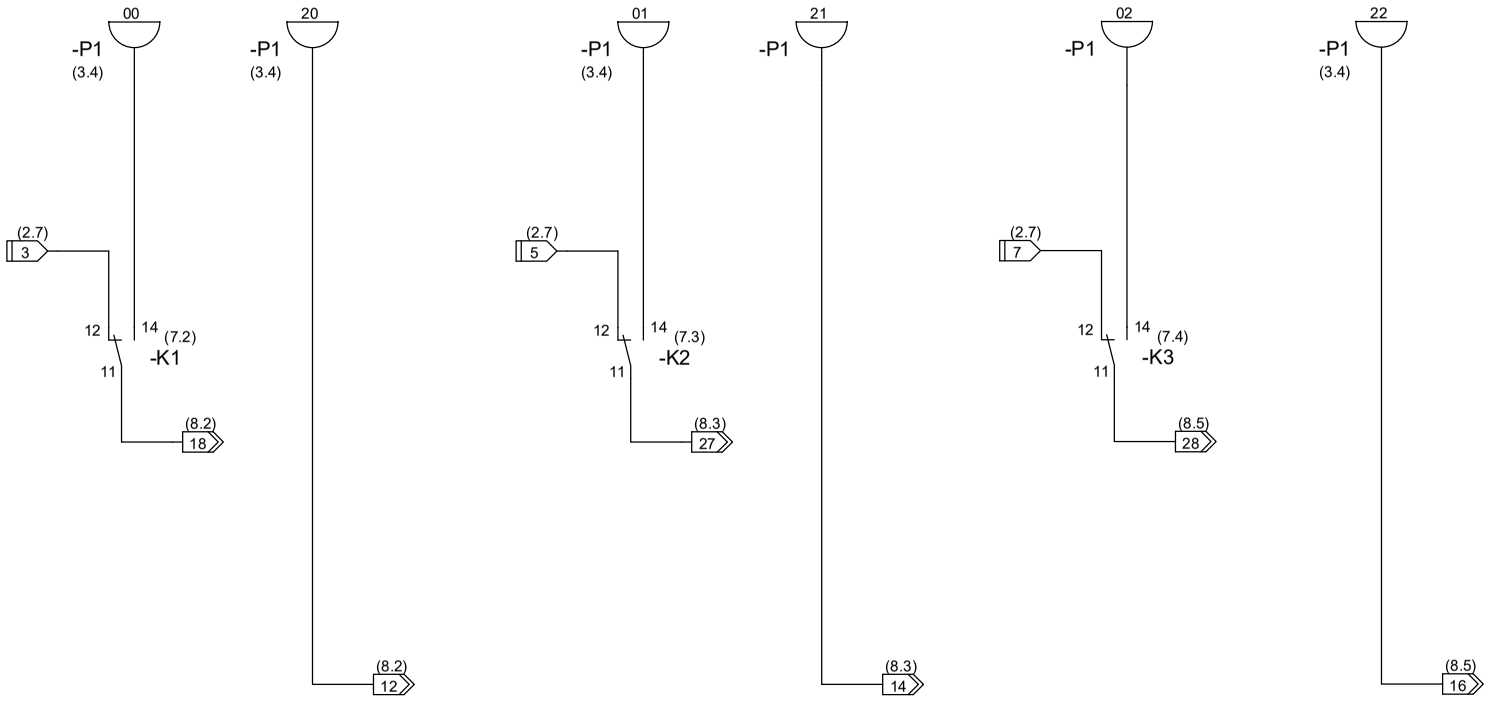
Original | Repl. for | Repl. of

SCANTROL

COUNTER INPUTS
INTERNAL WIRING

Project no.	Ver.
B023EB	
Scale: 1:1 A3	Drawing no.: B023EB-37-Bachelor
Page 5	N.pg. 6

SPEED REF CYLINDER 1	SPEED REF CYLINDER 2	SPEED REF CYLINDER 3
+/- 10V Output CH1	+/- 10V Output CH2	+/- 10V Output CH3



Ver.	Revision	Date	Sign.

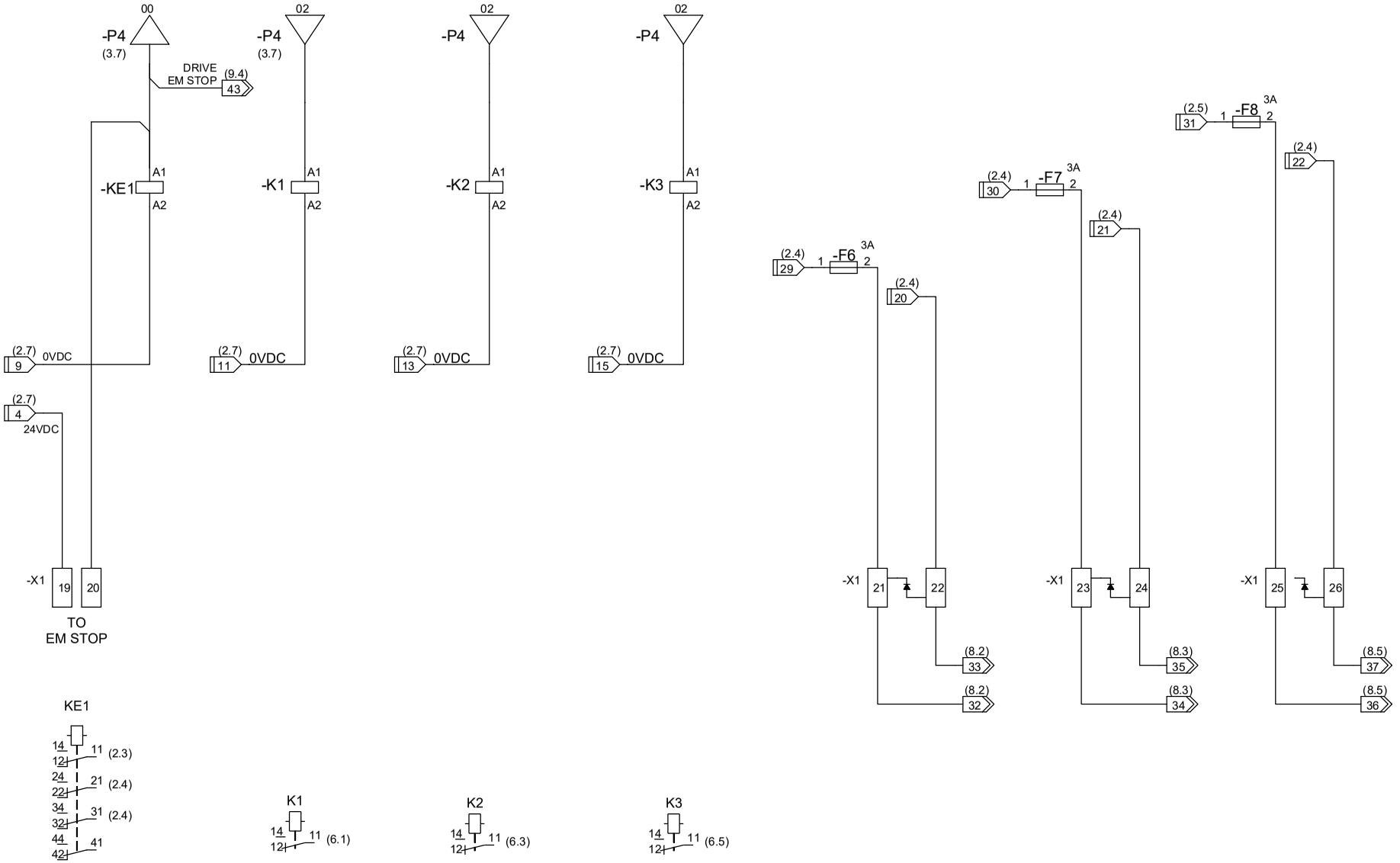
Date	14.03.2023	
Perf. by	SA	Scantrol Wave Simulation Plattform
Appr. by		
Norm		Original / Repl. for / Repl. of

SCANTROL

INTERNAL WIRING

Project no.	Ver.
B023EB	
Scale: 1:1 A3	Drawing no.: B023EB-37-Bachelor
Page 6	N.pg. 7

1	2	3	4	5	6	7	8
EMERGENCY STOP	**ENABLE RELAY CYLINDER 1**	**ENABLE RELAY CYLINDER 2**	**ENABLE RELAY CYLINDER 3**				
DIGITAL INPUT Ch1	DIGITAL OUTPUT Ch1	DIGITAL OUTPUT Ch2	DIGITAL OUTPUT Ch3				



Ver.	Revision	Date	Sign.

Date	14.03.2023
Perf. by	SA
Appr. by	
Norm	

Scantrol Wave Simulation Plattform

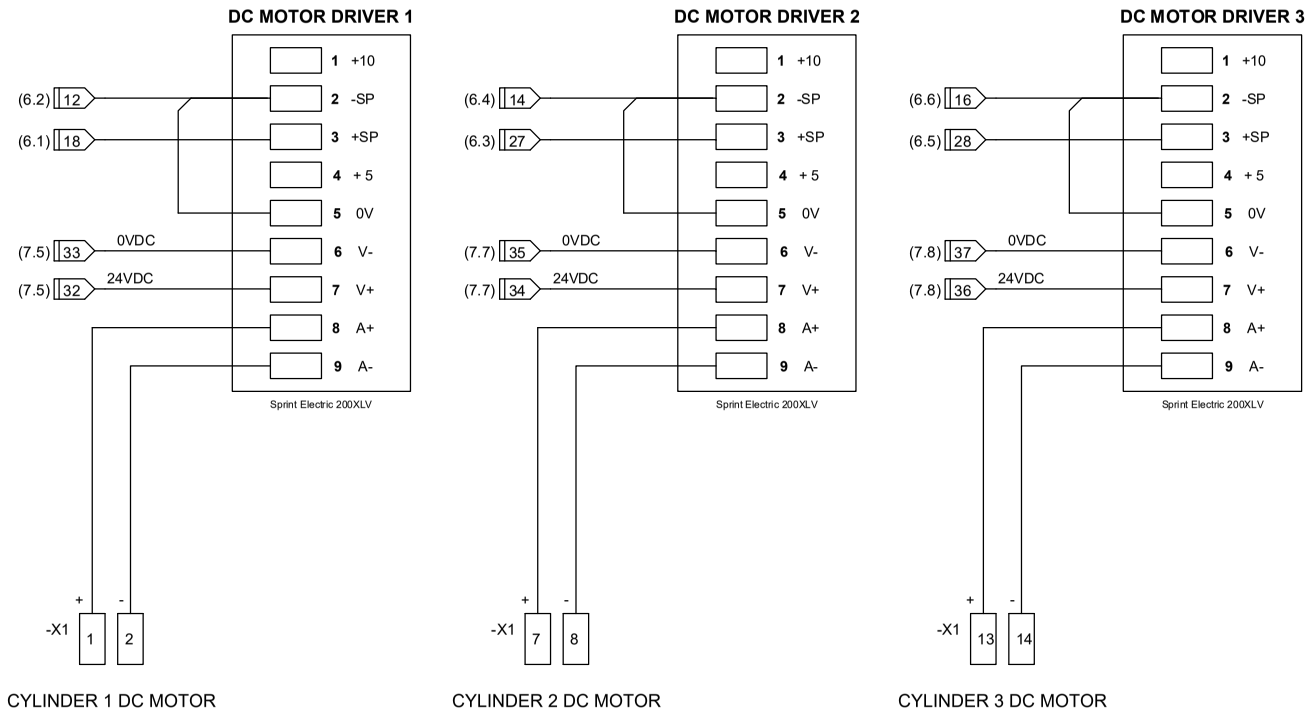
Original | Repl. for | Repl. of

SCANTROL

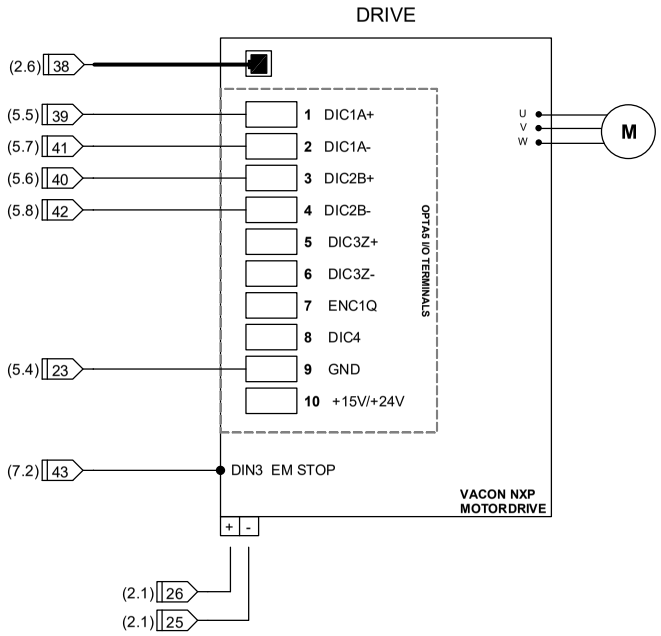
DIGITAL OUTPUT
INTERNAL WIRING

Project no.	Ver.
B023EB	
Scale	Drawing no.
1:1 A3	**B023EB-37-Bachelor**

Page 7
N.pg. 8



				Date	14.03.2023	Scantrol Wave Simulation Plattform			SCANTROL	INTERNAL WIRING	Project no. B023EB / Ver.
				Perf. by	SA						
				Appr. by							Scale 1:1 A3 / Drawing no. B023EB-37-Bachelor / Page 8
Ver.	Revision	Date	Sign.	Norm		Original	Repl. for	Repl. of			N.pg. 9



				Date	14.03.2023	Scantrol Wave Simulation Plattform		SCANTROL	INTERNAL WIRING	Project no. B023EB	Ver.	
				Perf. by	SA							
				Appr. by						Scale 1:1 A3	Drawing no. B023EB-37-Bachelor	Page 9
Ver.	Revision	Date	Sign.	Norm		Original	Repl. for	Repl. of				N.pg.

15 Appendiks G - Kildekode HMI(C#)

Dette vedlegget inneholder kildekoden til HMI. Den er programmert i C# WPF, og inneholder følgende klasser.

- PlattformUI.xaml.cs
- PLC_Com.cs
- DataFromPLC.cs
- DataToPLC.cs
- Logfile.cs

15.1 PlattformUI.xaml.cs

```
using System.Windows;
using System.IO;
using Microsoft.Win32;
using System.Collections.Generic;
using System;
using FieldTalk.Modbus.Master;
using System.Windows.Controls;
using MaterialDesignThemes.Wpf;
using System.Threading;
using System.Transactions;
using System.Windows.Media;
using MaterialDesignColors;
using System.Drawing;
using LiveCharts;
using LiveCharts.Definitions.Series;
using LiveCharts.Wpf;
using LiveCharts.Helpers;
using System.Collections.ObjectModel;
using LiveCharts.Defaults;
using ScottPlot;
using System.Linq;

namespace Plattform_UI
{

    /**************************************************************************
    Code information:
    PlattformUI.xaml.cs class

    All code linked to GUI operations is defined here.

    Author: Sander V. Andersen and Lars Askild S. Aarvik
    Date: 19.05.2023
    **************************************************************************/



    // Delegates
    public delegate void UpdateGUIDelegate();
    public delegate void StopPlattformDelegate(OperationMode OpMode);
    public delegate void AnimateProgressSubmitDelegate(double value);
    public delegate void UpdateProgressLogFileStartDelegate(double value, bool disableBtn);
    public delegate void CloseMbusProtocolDelegate();
    public delegate void UpdatedLivePlotsDelegate();
    public delegate void ClearAllPlotsDelegate();

    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class PlattformUI : Window
    {
        // Variables for scotplots
        const double Plt_SampleRate_Read_Sim = 17; // Samples per second
        const double Plt_SamlpeRate_Read_Log = 6.7;
        const double Plt_SampleRate_Logfile = 10; // Sample per second
        const int MaxDataPoints_Sim = 512;//3000;
        const int MaxDataPoints_Log = 202;
        int MaxDataPoints = 0;
        int datapoints = 0;
        public float[] Plt_Roll_pitch_Pos_data;
        public float[] Plt_Heave_Pos_data;
        public float[] Plt_Winch_Pos_data;
        public float[] Plt_Mp_Pos_data;
        public float[] Plt_Load_Pos_data;
        public ScottPlot.Plottable.SignalPlotGeneric<float> Signal_Plt_Heave;
        public ScottPlot.Plottable.SignalPlotGeneric<float> Signal_Plt_Roll_Pitch;
        public ScottPlot.Plottable.SignalPlotGeneric<float> Signal_Plt_Winch_Pos;
```

```
68          public ScottPlot.Plottable.SignalPlotGeneric<float> Signal_Plt_Mp_Pos;
69          public ScottPlot.Plottable.SignalPlotGeneric<float> Signal_Plt_Load_Pos;
70
71          //Defining colors
72          const string BackgroundColor_hex = "#FF303030";
73          const string LineColor_hex = "#FF03A9F4";
74          System.Drawing.Color Line_color_Blue;
75          System.Drawing.Color Line_color_Red;
76          System.Drawing.Color Plt_BackgroundColor;
77
78
79
80          // Public variables and constants
81          PLC_Com pLC_Com;
82          OperationMode OpMode;
83          Command command;
84          public static double PlattformCC = 0.425; // Center - Center distance between front/
        rear cylinders. (m)
85          public static double MaxCylinderHeight = 0.30; // Max cylinder height (m)
86          public static double MaxSpeedCylinder = 0.0469; // Max speed of the slowest cylinder (1
        ) m/s
87          const double MAXLOGFILELENGTH = 13; // Max logfile length possible to send
88          float TipVectX;
89          float TipVectY;
90          float TipVectZ;
91
92
93          public PlattformUI()
94          {
95              InitializeComponent();
96
97              // Initializing tipvectors
98              TipVectX = Convert.ToSingle(tb_CraneTip_xPos.Text);
99              TipVectY = Convert.ToSingle(tb_CraneTip_yPos.Text);
100             TipVectZ = Convert.ToSingle(tb_CraneTip_zPos.Text);
101
102             // Initialize scotplots
103             // Background color
104             Line_color_Blue = System.Drawing.ColorTranslator.FromHtml(LineColor_hex);
105             Line_color_Red = System.Drawing.ColorTranslator.FromHtml("#FFF11616");
106             Plt_BackgroundColor = System.Drawing.ColorTranslator.FromHtml(BackgroundColor_hex);
107
108             // Scotplot design style
109
110             // Disable zoom and pan
111             SctPlt_HeavePos.Configuration.Zoom = false;
112             SctPlt_Roll_Pitch_Pos.Configuration.Zoom = false;
113             SctPlt_WinchPos.Configuration.Zoom = false;
114             SctPlt_MpPos.Configuration.Zoom = false;
115             SctPlt_LoadPos.Configuration.Zoom = false;
116             SctPlt_HeavePos.Configuration.Pan = false;
117             SctPlt_Roll_Pitch_Pos.Configuration.Pan = false;
118             SctPlt_WinchPos.Configuration.Pan = false;
119             SctPlt_MpPos.Configuration.Pan = false;
120             SctPlt_LoadPos.Configuration.Pan = false;
121
122             // Design heave
123             SctPlt_HeavePos.Plot.XAxis.TickDensity(1);
124             SctPlt_HeavePos.Plot.Style(figureBackground: Plt_BackgroundColor, dataBackground:
        Plt_BackgroundColor, axisLabel: System.Drawing.Color.LightGray, tick: System.Drawing.Color.
        LightGray);//System.Drawing.Color.LightGray);
125             SctPlt_HeavePos.Plot.Grid(color: System.Drawing.Color.LightGray, lineStyle:
        LineStyle.Solid);
126             SctPlt_HeavePos.Plot.XLabel("Time [s]");
127             SctPlt_HeavePos.Plot.YLabel("Position [m]");
128             SctPlt_HeavePos.Plot.YAxis2.Line(false);
129             SctPlt_HeavePos.Plot.XAxis2.Line(false);
130
131             // Design roll/pitch
132             SctPlt_Roll_Pitch_Pos.Plot.XAxis.TickDensity(1);
133             SctPlt_Roll_Pitch_Pos.Plot.Style(figureBackground: Plt_BackgroundColor,
        dataBackground: Plt_BackgroundColor, axisLabel: System.Drawing.Color.LightGray, tick:
        System.Drawing.Color.LightGray);//System.Drawing.Color.LightGray);
134             SctPlt_Roll_Pitch_Pos.Plot.Grid(color: System.Drawing.Color.LightGray, lineStyle:
        LineStyle.Solid);
135             SctPlt_Roll_Pitch_Pos.Plot.XLabel("Time [s]");
136             SctPlt_Roll_Pitch_Pos.Plot.YLabel("Angle [deg]");
137             SctPlt_Roll_Pitch_Pos.Plot.YAxis2.Line(false);
138             SctPlt_Roll_Pitch_Pos.Plot.XAxis2.Line(false);
139
140             // Design Winch pos
141             SctPlt_WinchPos.Plot.XAxis.TickDensity(1);
142             SctPlt_WinchPos.Plot.Style(figureBackground: Plt_BackgroundColor, dataBackground:
        Plt_BackgroundColor, axisLabel: System.Drawing.Color.LightGray, tick: System.Drawing.Color.
        LightGray);//System.Drawing.Color.LightGray);
143             SctPlt_WinchPos.Plot.Grid(color: System.Drawing.Color.LightGray, lineStyle:
        LineStyle.Solid);
144             SctPlt_WinchPos.Plot.XLabel("Time [s]");
145             SctPlt_WinchPos.Plot.YLabel("Position [m]");
146             SctPlt_WinchPos.Plot.YAxis2.Line(false);
147             SctPlt_WinchPos.Plot.XAxis2.Line(false);
```

```
            SctPlt_WinchPos.Plot.YAxis.TickLabelNotation(invertSign: true);

            // Design Mp pos
            SctPlt_MpPos.Plot.XAxis.TickDensity(1);
            SctPlt_MpPos.Plot.Style(figureBackground: Plt_BackgroundColor, dataBackground:
    Plt_BackgroundColor, axisLabel: System.Drawing.Color.LightGray, tick: System.Drawing.Color.
    LightGray);//System.Drawing.Color.LightGray);
            SctPlt_MpPos.Plot.Grid(color: System.Drawing.Color.LightGray, lineStyle: LineStyle.
    Solid);
            SctPlt_MpPos.Plot.XLabel("Time [s]");
            SctPlt_MpPos.Plot.YLabel("Position [m]");
            SctPlt_MpPos.Plot.YAxis2.Line(false);
            SctPlt_MpPos.Plot.XAxis2.Line(false);
            SctPlt_MpPos.Plot.YAxis.TickLabelNotation(invertSign: true);

            // Design Load pos
            SctPlt_LoadPos.Plot.XAxis.TickDensity(1);
            SctPlt_LoadPos.Plot.Style(figureBackground: Plt_BackgroundColor, dataBackground:
    Plt_BackgroundColor, axisLabel: System.Drawing.Color.LightGray, tick: System.Drawing.Color.
    LightGray);//System.Drawing.Color.LightGray);
            SctPlt_LoadPos.Plot.Grid(color: System.Drawing.Color.LightGray, lineStyle:
    LineStyle.Solid);
            SctPlt_LoadPos.Plot.XLabel("Time [s]");
            SctPlt_LoadPos.Plot.YLabel("Position [m]");
            SctPlt_LoadPos.Plot.YAxis2.Line(false);
            SctPlt_LoadPos.Plot.XAxis2.Line(false);
            SctPlt_LoadPos.Plot.YAxis.TickLabelNotation(invertSign: true);

            // Initialize variables
            pLC_Com = new PLC_Com();
            OpMode = OperationMode.Manual;
            command = Command.Stop;

            // Initialize winch plot
            Plt_Winch_Pos_data = new float[MaxDataPoints_Sim];
            Signal_Plt_Winch_Pos = SctPlt_WinchPos.Plot.AddSignal(Plt_Winch_Pos_data,
    Plt_SampleRate_Read_Sim);
            SctPlt_WinchPos.Refresh();

            // Initialize Mp Pos plot
            Plt_Mp_Pos_data = new float[MaxDataPoints_Sim];
            Signal_Plt_Mp_Pos = SctPlt_MpPos.Plot.AddSignal(Plt_Mp_Pos_data,
    Plt_SampleRate_Read_Sim);
            SctPlt_MpPos.Refresh();

            // Initialize Load Pos plot
            Plt_Load_Pos_data = new float[MaxDataPoints_Sim];
            Signal_Plt_Load_Pos = SctPlt_LoadPos.Plot.AddSignal(Plt_Load_Pos_data,
    Plt_SampleRate_Read_Sim);
            SctPlt_LoadPos.Refresh();
        }

        private void Window_Closing(object sender, System.ComponentModel.CancelEventArgs e)
        {
            // Close modbus protocol when closing window
            CloseMbusProtocol();
        }

        private void btn_chooseFile_Click(object sender, RoutedEventArgs e)
        {
            // select logefile and read data from file
            OpenFileDialog ofd = new OpenFileDialog
            {
                Title = "Browse for MRU log file",
                Filter = "csv files (*.csv)|*.csv|All files (*.*)|*.*",
                FilterIndex = 1,
                RestoreDirectory = true,
                CheckFileExists = true,
                CheckPathExists = true
            };

            if(ofd.ShowDialog() == true)
            {
                tb_NoFileChoosen.Text = "";
                tb_Filename.Text = "Filename: " + ofd.SafeFileName;

                pLC_Com.DataToPlc.Logfile.FileName = ofd.FileName;
                if (pLC_Com.DataToPlc.Logfile.GetLogfileLength() < 26) sl_length_LogFile.
    Maximum = pLC_Com.DataToPlc.Logfile.GetLogfileLength() - 1;
                else sl_length_LogFile.Maximum = 25;

                // Read data from file
                pLC_Com.DataToPlc.Logfile.UpdateLogLength(sl_length_LogFile.Maximum + 1);
                pLC_Com.DataToPlc.Logfile.ReadDataFromFile();

                Grid_LogfileInfo.Visibility = Visibility.Visible;
                btn_Logfile_submit.IsEnabled = true;
            }
        }

        public void btn_Connect_Click(object sender, RoutedEventArgs e)
```

```
229         {
230             // Connect/disconnet modbus com to PLC
231 
232             int result = 0;
233 
234             // Connect to PLC
235             if (!pLC_Com.ModbusProtocol.isOpen())
236             {
237                 bool error = CheckValidIP(tb_IPAddress.Text);
238 
239                 if (error) MessageBox.Show("Error!", "Not a valid IP address. Try again!");
240                 else
241                 {
242                     result = pLC_Com.OpenProtocol(tb_IPAddress.Text);
243                     tb_Com_Status.Text = "OpenProtocol: " + BusProtocolErrors.
    getBusProtocolErrorText(result);
244 
245                     if (result == BusProtocolErrors.FTALK_SUCCESS)
246                     {
247                         // Enable buttons
248                         btn_Reset_Manual.IsEnabled = true;
249                         btn_Reset_Sim.IsEnabled = true;
250                         btn_Reset_Log.IsEnabled = true;
251                         btn_Reset_Length.IsEnabled = true;
252 
253                         btn_Connect.Content = "Disconnect";
254 
255                         //Reset alla alarms
256                         lv_Alarms.Items.Clear();
257 
258                         // Uncomment to startread from PLC
259                         pLC_Com.StartDataReciver(this);
260                     }
261                     else
262                     {
263                         MessageBox.Show(BusProtocolErrors.getBusProtocolErrorText(result), "
    Error!");
264                     }
265                 }
266             }
267             // Disconnect from PLC
268             else
269             {
270                 // Close protocol
271                 CloseMbusProtocol();
272             }
273         }
274 
275         public void CloseMbusProtocol()
276         {
277             // Close protocol
278             pLC_Com.CloseProtocol();
279 
280             // Disable/enable buttons
281             DisableAllButtons();
282 
283             btn_Connect.Content = "Connect";
284             tb_Com_Status.Text = "Dicsonnected";
285         }
286 
287         void DisableAllButtons()
288         {
289             btn_Start_Sim.IsEnabled = false;
290             btn_Reset_Sim.IsEnabled = false;
291             btn_Stop_Sim.IsEnabled = false;
292             btn_Start_Log.IsEnabled = false;
293             btn_Reset_Log.IsEnabled = false;
294             btn_Stop_Log.IsEnabled = false;
295             btn_Stop_Manual.IsEnabled = false;
296             btn_Reset_Manual.IsEnabled = false;
297             btn_Start_Manual.IsEnabled = false;
298             btn_Reset_Length.IsEnabled = false;
299         }
300 
301         bool CheckValidIP(string text)
302         {
303             // Cheks for valid IP adress, returns false if not valid
304             if (String.IsNullOrWhiteSpace(text))
305             {
306                 return true;
307             }
308 
309             string[] splitValues = text.Split('.');
310             if (splitValues.Length != 4)
311             {
312                 return true;
313             }
314 
315             return false;
316         }
317 
```

```
318         private void btn_info_xPos_Click(object sender, RoutedEventArgs e)
319         {
320             // Show infocard
321             infoCardVisibility(Crd_info_xPos);
322         }
323 
324         private void btn_info_yPos_Click(object sender, RoutedEventArgs e)
325         {
326             // Show infocard
327             infoCardVisibility(Crd_info_yPos);
328         }
329 
330         private void btn_info_zPos_Click(object sender, RoutedEventArgs e)
331         {
332             // Show infocard
333             infoCardVisibility(Crd_info_zPos);
334         }
335 
336         void infoCardVisibility(MaterialDesignThemes.Wpf.Card infoCard)
337         {
338             // toggle infocard visibility
339             if (infoCard.Visibility == Visibility.Collapsed) infoCard.Visibility = Visibility.
    Visible;
340             else infoCard.Visibility = Visibility.Collapsed;
341         }
342 
343         private void btn_Start_Sim_Click(object sender, RoutedEventArgs e)
344         {
345             // Set data and send to PLC
346             int result = pLC_Com.SendData(OperationMode.Simulated, Command.Start, (float)
    sl_Amplitude_sim.Value / 100,
347                 (float)sl_Period_sim.Value, (float)sl_PhaseDiff_sim.Value, 0, 0, 0, 0);
348 
349             // Check if data is sent successfully and diable/enable buttons if sucess.
350             if(result == BusProtocolErrors.FTALK_SUCCESS)
351             {
352                 Btn_Start_Stop_Reset_click(Command.Start);
353             }
354         }
355 
356         private void btn_Stop_Sim_Click(object sender, RoutedEventArgs e)
357         {
358             // Set data and send to PLC
359             int result = pLC_Com.SendData(OperationMode.Simulated, Command.Stop, 0, 0, 0, 0, 0,
     0, 0);
360 
361             // Check if data is sent successfully and diable/enable buttons if sucess.
362             if (result == BusProtocolErrors.FTALK_SUCCESS)
363             {
364                 Btn_Start_Stop_Reset_click(Command.Stop);
365             }
366         }
367 
368         private void btn_Reset_Sim_Click(object sender, RoutedEventArgs e)
369         {
370             // Set data and send to PLC
371             int result = pLC_Com.SendData(OperationMode.Simulated, Command.Reset, 0, 0, 0, 0, 0
    , 0, 0);
372 
373             // Check if data is sent successfully and diable/enable buttons if sucess.
374             if (result == BusProtocolErrors.FTALK_SUCCESS)
375             {
376                 Btn_Start_Stop_Reset_click(Command.Reset);
377             }
378         }
379 
380         private void btn_Start_Manual_Click(object sender, RoutedEventArgs e)
381         {
382             // Set data and send to PLC
383             int result = pLC_Com.SendData(OperationMode.Manual, Command.Start, 0, 0, 0,
384                 (float)sl_HeavePos_Manual.Value / 100, (float)sl_RollAngle_Manual.Value, 0, 0);
385 
386             // Check if data is sent successfully and diable/enable buttons if sucess.
387             if (result == BusProtocolErrors.FTALK_SUCCESS)
388             {
389                 Btn_Start_Stop_Reset_click(Command.Start);
390             }
391         }
392 
393         private void btn_Start_Log_Click(object sender, RoutedEventArgs e)
394         {
395             // Clear all plots on start
396             ClearAllPlots();
397             // Set data and send to PLC
398             int result = pLC_Com.SendData(OperationMode.Logfile, Command.Stop, 0, 0, 0, 0, 0, 0
    , 0);
399 
400             // Check if data is sent successfully and diable/enable buttons if sucess.
401             if (result == BusProtocolErrors.FTALK_SUCCESS)
402             {
403                 Btn_Start_Stop_Reset_click(Command.Start);
```

```
            }
        }

        private void btn_Stop_Manual_Click(object sender, RoutedEventArgs e)
        {
            // Set data and send to PLC
            int result = pLC_Com.SendData(OperationMode.Manual, Command.Stop, 0, 0, 0, 0, 0, 0,
    0);

            // Check if data is sent successfully and diable/enable buttons if sucess.
            if (result == BusProtocolErrors.FTALK_SUCCESS)
            {
                Btn_Start_Stop_Reset_click(Command.Stop);
            }
        }

        private void btn_Stop_Log_Click(object sender, RoutedEventArgs e)
        {
            // Set data and send to PLC
            int result = pLC_Com.SendData(OperationMode.Manual, Command.Stop, 0, 0, 0, 0, 0, 0,
    0);

            // Check if data is sent successfully and diable/enable buttons if sucess.
            if (result == BusProtocolErrors.FTALK_SUCCESS)
            {
                Btn_Start_Stop_Reset_click(Command.Stop);
            }
        }

        private void btn_Reset_Manual_Click(object sender, RoutedEventArgs e)
        {
            // Set data and send to PLC
            int result = pLC_Com.SendData(OperationMode.Manual, Command.Reset, 0, 0, 0, 0, 0, 0
    , 0);

            // Check if data is sent successfully and diable/enable buttons if sucess.
            if (result == BusProtocolErrors.FTALK_SUCCESS)
            {
                Btn_Start_Stop_Reset_click(Command.Reset);
            }
        }

        private void btn_Reset_Log_Click(object sender, RoutedEventArgs e)
        {
            // Set data and send to PLC
            int result = pLC_Com.SendData(OperationMode.Manual, Command.Reset, 0, 0, 0, 0, 0, 0
    , 0);

            // Check if data is sent successfully and diable/enable buttons if sucess.
            if (result == BusProtocolErrors.FTALK_SUCCESS)
            {
                Btn_Start_Stop_Reset_click(Command.Reset);
            }
        }

        void Btn_Start_Stop_Reset_click(Command cmd)
        {
            // Enable/Disable buttons based on button click
            switch (cmd)
            {
                case Command.Stop:
                    btn_Stop_Manual.IsEnabled = false;
                    btn_Stop_Log.IsEnabled = false;
                    btn_Stop_Sim.IsEnabled = false;
                    btn_Start_Manual.IsEnabled = true;
                    btn_Start_Sim.IsEnabled = true;
                    btn_Start_Log.IsEnabled = true;
                    btn_Reset_Manual.IsEnabled = true;
                    btn_Reset_Sim.IsEnabled = true;
                    btn_Reset_Log.IsEnabled = true;
                    break;
                case Command.Start:
                    btn_Start_Manual.IsEnabled = false;
                    btn_Start_Log.IsEnabled = false;
                    btn_Reset_Manual.IsEnabled = false;
                    btn_Start_Sim.IsEnabled = false;
                    btn_Reset_Log.IsEnabled = false;
                    btn_Reset_Sim.IsEnabled = false;
                    btn_Stop_Manual.IsEnabled = true;
                    btn_Stop_Log.IsEnabled = true;
                    btn_Stop_Sim.IsEnabled = true;
                    break;
                case Command.Reset:
                    btn_Reset_Sim.IsEnabled = false;
                    btn_Reset_Log.IsEnabled = false;
                    btn_Reset_Manual.IsEnabled = false;
                    btn_Start_Log.IsEnabled = false;
                    btn_Start_Sim.IsEnabled = false;
                    btn_Start_Log.IsEnabled = false;
                    btn_Stop_Manual.IsEnabled = true;
                    btn_Stop_Log.IsEnabled = true;
```

```
                    btn_Stop_Sim.IsEnabled = true;
                    break;
                default:
                    break;
            }
        }



        public void UpdateGUI()
        {
            // Updates GUI with data from PLC

            // Update sliders with cylinder positions
            sl_PV_Cyl1.Value = pLC_Com.DataFromPlc.Cyl1Pos * 100;
            tb_PV_Cyl1.Text = (pLC_Com.DataFromPlc.Cyl1Pos * 100).ToString("#.#");
            sl_PV_Cyl2.Value = pLC_Com.DataFromPlc.Cyl2Pos * 100;
            tb_PV_Cyl2.Text = (pLC_Com.DataFromPlc.Cyl2Pos * 100).ToString("#.#");
            sl_PV_Cyl3.Value = pLC_Com.DataFromPlc.Cyl3Pos * 100;
            tb_PV_Cyl3.Text = (pLC_Com.DataFromPlc.Cyl3Pos * 100).ToString("#.#");

            // Update service tab with plattform/winch data
            tb_Cyl1Spd.Text = pLC_Com.DataFromPlc.Cyl1Spd.ToString("0.####");
            tb_Cyl2Spd.Text = pLC_Com.DataFromPlc.Cyl2Spd.ToString("0.####");
            tb_Cyl3Spd.Text = pLC_Com.DataFromPlc.Cyl3Spd.ToString("0.####");
            tb_Wire_Length.Text = pLC_Com.DataFromPlc.WinchPos.ToString("0.##");
            tb_Winch_Spd.Text = pLC_Com.DataFromPlc.WinchSpd.ToString("0.##");

            // Update WinchOpMode
            switch (pLC_Com.DataFromPlc.OpModeWinch)
            {
                case OperationModeWinch.Standby:
                    btn_WinchOpMode_AHCTab.BorderBrush = new SolidColorBrush(Colors.DarkGreen);
                    btn_WinchOpMode_AHCTab.Content = "Standby";
                    btn_WinchOpMode_ManualTab.BorderBrush = new SolidColorBrush(Colors.
DarkGreen);
                    btn_WinchOpMode_ManualTab.Content = "Standby";
                    break;
                case OperationModeWinch.Manual:
                    btn_WinchOpMode_AHCTab.BorderBrush = new SolidColorBrush(Colors.Yellow);
                    btn_WinchOpMode_AHCTab.Content = "Manual";
                    btn_WinchOpMode_ManualTab.BorderBrush = new SolidColorBrush(Colors.Yellow);
                    btn_WinchOpMode_ManualTab.Content = "Manual";
                    break;
                case OperationModeWinch.AHC:
                    btn_WinchOpMode_AHCTab.BorderBrush = new SolidColorBrush(System.Windows.
Media.Color.FromArgb(0xFF, 0xF1, 0x16, 0x16));
                    btn_WinchOpMode_AHCTab.Content = "AHC";
                    btn_WinchOpMode_ManualTab.BorderBrush = new SolidColorBrush(System.Windows.
Media.Color.FromArgb(0xFF, 0xF1, 0x16, 0x16));
                    btn_WinchOpMode_ManualTab.Content = "AHC";
                    break;
                case OperationModeWinch.Error:
                    btn_WinchOpMode_AHCTab.BorderBrush = new SolidColorBrush(System.Windows.
Media.Color.FromArgb(0xFF, 0xF1, 0x16, 0x16));
                    btn_WinchOpMode_AHCTab.Content = "Error";
                    btn_WinchOpMode_ManualTab.BorderBrush = new SolidColorBrush(System.Windows.
Media.Color.FromArgb(0xFF, 0xF1, 0x16, 0x16));
                    btn_WinchOpMode_ManualTab.Content = "Error";
                    break;
                default:
                    break;
            }

            // Update winch position live plot
            if (Plt_Winch_Pos_data != null && Plt_Winch_Pos_data.Length == MaxDataPoints_Sim)
            {
                if ((Signal_Plt_Winch_Pos != null) && !(float.IsNaN(pLC_Com.DataFromPlc.
WinchPos)) && (pLC_Com.DataFromPlc.WinchPos != float.NegativeInfinity) && (pLC_Com.
DataFromPlc.WinchPos != float.PositiveInfinity))
                {
                    Array.Copy(Plt_Winch_Pos_data, 1, Plt_Winch_Pos_data, 0, MaxDataPoints_Sim
- 1); // shift the data buffer

                    Plt_Winch_Pos_data[MaxDataPoints_Sim - 1] = -pLC_Com.DataFromPlc.WinchPos;
// add the new data point at the end


                    // FLytte dette til initializing p   toppen???
                    SctPlt_WinchPos.Plot.AxisAuto(0, 0.05);
                    Signal_Plt_Winch_Pos.LineStyle = LineStyle.Solid;
                    Signal_Plt_Winch_Pos.LineWidth = 1.5;
                    Signal_Plt_Winch_Pos.FillBelow();
                    Signal_Plt_Winch_Pos.MarkerSize = 0;
                    Signal_Plt_Winch_Pos.LineColor = Line_color_Blue;

                    SctPlt_WinchPos.Refresh();
                }
            }

            // Update Mp position live plot
```

```
573             float MpPos = GetMpPos(pLC_Com.DataFromPlc.Cyl1Pos, pLC_Com.DataFromPlc.Cyl2Pos,
    pLC_Com.DataFromPlc.Cyl3Pos);
574             if (Plt_Mp_Pos_data != null && Plt_Mp_Pos_data.Length == MaxDataPoints_Sim)
575             {
576                 if ((Signal_Plt_Mp_Pos != null) && !(float.IsNaN(MpPos)) && (MpPos != float.
    NegativeInfinity) && (MpPos != float.PositiveInfinity))
577                 {
578                     Array.Copy(Plt_Mp_Pos_data, 1, Plt_Mp_Pos_data, 0, MaxDataPoints_Sim - 1);
    // shift the data buffer
579                     Plt_Mp_Pos_data[MaxDataPoints_Sim - 1] = MpPos; // add the new data point
    at the end
580                     //Update textbox
581                     tb_Mp_Pos.Text = MpPos.ToString("0.##");
582 
583                     // FLytte dette til initializing p   toppen???
584                     SctPlt_MpPos.Plot.AxisAuto(0, 0.05);
585                     Signal_Plt_Mp_Pos.LineStyle = LineStyle.Solid;
586                     Signal_Plt_Mp_Pos.LineWidth = 1.5;
587                     Signal_Plt_Mp_Pos.FillBelow();
588                     Signal_Plt_Mp_Pos.MarkerSize = 0;
589                     Signal_Plt_Mp_Pos.LineColor = Line_color_Blue;
590 
591                     SctPlt_MpPos.Refresh();
592                 }
593             }
594 
595             // Update Load position live plot
596             if (Plt_Load_Pos_data != null && Plt_Load_Pos_data.Length == MaxDataPoints_Sim)
597             {
598                 float LoadPos = GetLoadPos(pLC_Com.DataFromPlc.AHC_CenterPos, pLC_Com.
    DataFromPlc.WinchPos, MpPos);
599                 if ((Signal_Plt_Load_Pos != null) && !(float.IsNaN(LoadPos)) && (LoadPos !=
    float.NegativeInfinity) && (LoadPos != float.PositiveInfinity))
600                 {
601                     Array.Copy(Plt_Load_Pos_data, 1, Plt_Load_Pos_data, 0, MaxDataPoints_Sim -
    1); // shift the data buffer
602                     Plt_Load_Pos_data[MaxDataPoints_Sim - 1] = LoadPos; // add the new data
    point at the end
603                     //Update textbox
604                     tb_Load_Pos.Text = LoadPos.ToString("0.##");
605 
606                     // FLytte dette til initializing p   toppen???
607 
608                     Signal_Plt_Load_Pos.LineStyle = LineStyle.Solid;
609                     Signal_Plt_Load_Pos.LineWidth = 1.5;
610                     Signal_Plt_Load_Pos.FillBelow();
611                     Signal_Plt_Load_Pos.MarkerSize = 0;
612                     if (pLC_Com.DataFromPlc.OpModeWinch == OperationModeWinch.AHC)
613                     {
614                         Signal_Plt_Load_Pos.LineColor = Line_color_Red;
615                         SctPlt_LoadPos.Plot.SetAxisLimits(SctPlt_MpPos.Plot.GetAxisLimits());
616                     }
617                     else
618                     {
619                         Signal_Plt_Load_Pos.LineColor = Line_color_Blue;
620                         SctPlt_LoadPos.Plot.AxisAuto(0, 0.05);
621                     }
622 
623                     SctPlt_LoadPos.Refresh();
624                 }
625             }
626         }
627 
628             public void StopPlattform(OperationMode OpMode)
629         {
630             // Sets plattform in stop mode if Em stop is pressed.
631             switch (OpMode)
632             {
633                 case OperationMode.Manual:
634                     btn_Stop_Manual_Click(new object(), new RoutedEventArgs());
635                     break;
636                 case OperationMode.Simulated:
637                     btn_Stop_Sim_Click(new object(), new RoutedEventArgs());
638                     break;
639                 case OperationMode.Logfile:
640                     btn_Stop_Log_Click(new object(), new RoutedEventArgs());
641                     break;
642                 default:
643                     break;
644             }
645             // Add alarm to alarm history
646             lv_Alarms.Items.Add("Emergency stop     " + DateTime.Now);
647         }
648 
649         private void sl_HeavePos_Manual_ValueChanged(object sender,
    RoutedPropertyChangedEventArgs<double> e)
650         {
651             // Set limits to roll angle based on heave pos selection
652             //Check if value heavepos value is at max/min value. Then it needs to be manually
    set to 0 because of 2degree margin that has been added to roll angle
653             if ((sl_HeavePos_Manual.Value != 0) && (sl_HeavePos_Manual.Value != 30))
```

```
654             {
655                 if (sl_HeavePos_Manual.Value < 15)
656                 {
657                     sl_RollAngle_Manual.Maximum = Math.Floor((180 / Math.PI) * Math.Asin((2 * (
    sl_HeavePos_Manual.Value / 100)) / PlattformCC)) - 2;
658                     sl_RollAngle_Manual.Minimum = -sl_RollAngle_Manual.Maximum;
659                 }
660                 else
661                 {
662                     sl_RollAngle_Manual.Maximum = Math.Floor((180 / Math.PI) * Math.Asin(((2 *
    MaxCylinderHeight) - (2 * (sl_HeavePos_Manual.Value / 100))) / PlattformCC)) - 2;
663                     sl_RollAngle_Manual.Minimum = -sl_RollAngle_Manual.Maximum;
664                 }
665             }
666             else
667             {
668                 sl_RollAngle_Manual.Minimum = 0;
669                 sl_RollAngle_Manual.Maximum = 0;
670             }
671             tb_min_RollAngle.Text = sl_RollAngle_Manual.Minimum.ToString();
672             tb_max_RollAngle.Text = sl_RollAngle_Manual.Maximum.ToString();
673         }
674 
675       private void sl_Amplitude_sim_ValueChanged(object sender, RoutedPropertyChangedEventArgs
    <double> e)
676         {
677             // Set limits for period based on heave pos selction
678             sl_Period_sim.Minimum = Math.Ceiling((sl_Amplitude_sim.Value * 0.5 * 0.01 * 2 *
    Math.PI) / MaxSpeedCylinder);
679         }
680 
681         private void sl_Period_sim_ValueChanged(object sender, RoutedPropertyChangedEventArgs<
    double> e)
682         {
683             // Set limits to amplitude based on periode selection
684             if ((Math.Floor(((sl_Period_sim.Value * MaxSpeedCylinder * 2) / (2 * Math.PI)) * 10
    0) > 25) || (sl_Period_sim.Value == sl_Period_sim.Minimum)) sl_Amplitude_sim.Maximum = 25;
685             else sl_Amplitude_sim.Maximum = Math.Floor(((sl_Period_sim.Value * MaxSpeedCylinder
     * 2) / (2 * Math.PI)) * 100);
686         }
687 
688 
689         private void sl_Scale_RollPitch_Log_ValueChanged(object sender,
    RoutedPropertyChangedEventArgs<double> e)
690         {
691             // Plot logifle data when slider is moved
692             if (sl_Scale_RollPitch_Log.Value != sl_Scale_RollPitch_Log.Maximum)
693             {
694                 Calculate_Plot_Heave_Roll_Pitch((float)sl_Scale_Heave_Log.Value, (float)
    sl_Scale_RollPitch_Log.Value);
695             }
696         }
697 
698 
699         private void sl_Scale_Heave_Log_ValueChanged(object sender,
    RoutedPropertyChangedEventArgs<double> e)
700         {
701             // Plot logifle data when slider is moved
702             if (sl_Scale_Heave_Log.Value != sl_Scale_Heave_Log.Maximum)
703             {
704                 Calculate_Plot_Heave_Roll_Pitch((float)sl_Scale_Heave_Log.Value, (float)
    sl_Scale_RollPitch_Log.Value);
705 
706                 sl_Scale_RollPitch_Log.Maximum = Math.Floor( (((Math.Abs( (180 / Math.PI) *
    Math.Asin(((0.15 - pLC_Com.DataToPlc.Logfile.MaxHeavPos) * 2) / PlattformCC ) ) - 2) / Math
    .Abs(pLC_Com.DataToPlc.Logfile.MaxAngle)) * 100) * 0.3);
707                 if (sl_Scale_RollPitch_Log.Maximum < 100) sl_Scale_RollPitch_Log.Maximum = 100;
708             }
709         }
710 
711         private void btn_Logfile_submit_Click(object sender, RoutedEventArgs e)
712         {
713             if (sl_length_LogFile.Value <= 0) MessageBox.Show("Error", "Select valid logfile
    length (>0)!");
714             else
715             {
716                 Calculate_Plot_Heave_Roll_Pitch((float)sl_Scale_Heave_Log.Value, (float)
    sl_Scale_RollPitch_Log.Value);
717                 ThreadPool.QueueUserWorkItem(Submit_Progress_Animation, null);
718             }
719         }
720 
721         void Submit_Progress_Animation(object o)
722         {
723             // Animation for progres feedback
724             AnimateProgressSubmitDelegate APSD = new AnimateProgressSubmitDelegate(
    UpdateProgressSubmit);
725             for (int i = 0; i < 101; i++)
726             {
727                 Dispatcher.Invoke(APSD, i);
728                 Thread.Sleep(10);
```

```
            }
            Thread.Sleep(1000);
            Dispatcher.Invoke(APSD, -1);
        }

        void UpdateProgressSubmit(double value)
        {
            // Animation for progres feedback
            PrBar_Submit.Value = value;
        }

        public void UpdateProgressStartLogfile(double value, bool finished)
        {
            // Progres feedback on logfile start click... shows progres while data is sent to PLC
            ButtonProgressAssist.SetValue(btn_Start_Log, value);
            prBar_LogfileSend.Value = value;
            if (finished)
            {
                prBar_LogfileSend.Visibility = Visibility.Hidden;
                // Set data and send to PLC
                pLC_Com.SendData(OperationMode.Logfile, Command.Start, 0, 0, 0, 0, 0, 0, 0);
            }
            else prBar_LogfileSend.Visibility = Visibility.Visible;
        }



        public void Calculate_Plot_Heave_Roll_Pitch(float HeaveScaleVal, float RollScaleVal)
        {
            // Calculates cylinder positions and velocities and plots heave and roll to HMI
            pLC_Com.DataToPlc.Logfile.UpdateLogLength(sl_length_LogFile.Value);
            //pLC_Com.DataToPlc.Logfile.ReadDataFromFile();
            pLC_Com.DataToPlc.Logfile.Calculate_Scale_CylinderPosVel(tglbtn_UsePitch.IsChecked.Value, HeaveScaleVal, RollScaleVal);


            // Plot Logefile data heave pos
            SctPlt_HeavePos.Plot.Clear();
            Plt_Heave_Pos_data = pLC_Com.DataToPlc.Logfile.Heave_Pos_Scaled.ToArray();
            Signal_Plt_Heave = SctPlt_HeavePos.Plot.AddSignal(Plt_Heave_Pos_data, Plt_SampleRate_Logfile);
            Signal_Plt_Heave.LineStyle = LineStyle.Solid;
            Signal_Plt_Heave.LineWidth = 1.5;
            Signal_Plt_Heave.FillBelow();
            Signal_Plt_Heave.MarkerSize = 0;
            Signal_Plt_Heave.LineColor = Line_color_Blue;
            SctPlt_HeavePos.Plot.Style();
            SctPlt_HeavePos.Refresh();



            // Plot Logefile data roll/pitch
            SctPlt_Roll_Pitch_Pos.Plot.Clear();
            if (tglbtn_UsePitch.IsChecked.Value) Plt_Roll_pitch_Pos_data = pLC_Com.DataToPlc.Logfile.Pitch_Angle_Scaled.ToArray();
            else Plt_Roll_pitch_Pos_data = pLC_Com.DataToPlc.Logfile.Roll_Angle_Scaled.ToArray();
            Signal_Plt_Roll_Pitch = SctPlt_Roll_Pitch_Pos.Plot.AddSignal(Plt_Roll_pitch_Pos_data, Plt_SampleRate_Logfile);
            Signal_Plt_Roll_Pitch.LineStyle = LineStyle.Solid;
            Signal_Plt_Roll_Pitch.LineWidth = 1.5;
            Signal_Plt_Roll_Pitch.FillBelow();
            Signal_Plt_Roll_Pitch.MarkerSize = 0;
            Signal_Plt_Roll_Pitch.LineColor = Line_color_Blue;
            SctPlt_Roll_Pitch_Pos.Refresh();

        }

        private void tglbtn_UsePitch_Click(object sender, RoutedEventArgs e)
        {
            if (tglbtn_UsePitch.IsChecked.Value) tb_ScaleRollPitch.Text = "Scale pitch";
            else tb_ScaleRollPitch.Text = "Scale roll";
            Calculate_Plot_Heave_Roll_Pitch(Convert.ToSingle(sl_Scale_Heave_Log.Value), Convert.ToSingle(sl_Scale_RollPitch_Log.Value));
        }


        private void sl_length_LogFile_ValueChanged(object sender, RoutedPropertyChangedEventArgs<double> e)
        {
            // Calculate sclaefactor and plot data based on lgofile length selction
            if (sl_length_LogFile.Value == 0)
            {
                SctPlt_HeavePos.Plot.Clear();
                SctPlt_Roll_Pitch_Pos.Plot.Clear();
            }
            else
            {
                Calculate_Plot_Heave_Roll_Pitch(1, 1);
                sl_Scale_Heave_Log.Maximum = Math.Floor(pLC_Com.DataToPlc.Logfile.
```

```
    MaxHeaveScaleSlider);
812             sl_Scale_Heave_Log.Value = sl_Scale_Heave_Log.Maximum;
813             sl_Scale_RollPitch_Log.Value = sl_Scale_RollPitch_Log.Maximum;
814             sl_Scale_Heave_Log.IsEnabled = true;
815             sl_Scale_RollPitch_Log.IsEnabled = true;
816         }
817     }
818
819     public void UpdateLivePlots()
820     {
821         // Update heave pos and roll angle plots with live data from PLC.
822         if (Plt_Heave_Pos_data != null && Plt_Roll_pitch_Pos_data != null &&
    Plt_Heave_Pos_data.Length == MaxDataPoints && Plt_Roll_pitch_Pos_data.Length ==
    MaxDataPoints)
823         {
824             float Heavpos = GetHeavePos(pLC_Com.DataFromPlc.Cyl1Pos, pLC_Com.DataFromPlc.
    Cyl2Pos, pLC_Com.DataFromPlc.Cyl3Pos);
825             if ((Signal_Plt_Heave != null) && !(float.IsNaN(Heavpos)) && (Heavpos != float.
    NegativeInfinity) && (Heavpos != float.PositiveInfinity))
826             {
827                 Array.Copy(Plt_Heave_Pos_data, 1, Plt_Heave_Pos_data, 0, MaxDataPoints - 1)
    ; // shift the data buffer
828                 Plt_Heave_Pos_data[MaxDataPoints - 1] = Heavpos;// add the new data point
    at the end
829             }
                                            //}
830
831             float rollangle = GetRollAngle(pLC_Com.DataFromPlc.Cyl1Pos, pLC_Com.DataFromPlc
    .Cyl2Pos, pLC_Com.DataFromPlc.Cyl3Pos);
832             if ((Signal_Plt_Roll_Pitch != null) && !(float.IsNaN(rollangle)) && (rollangle
    != float.NegativeInfinity) && (rollangle != float.PositiveInfinity))
833             {
834                 Array.Copy(Plt_Roll_pitch_Pos_data, 1, Plt_Roll_pitch_Pos_data, 0,
    MaxDataPoints - 1); // shift the data buffer
835                 Plt_Roll_pitch_Pos_data[MaxDataPoints - 1] = rollangle; // add the new data
     point at the end
836             }
                                                                    //}
837
838
839             SctPlt_HeavePos.Plot.AxisAuto(0, 0.05);
840             Signal_Plt_Heave.LineStyle = LineStyle.Solid;
841             Signal_Plt_Heave.LineWidth = 1.5;
842             Signal_Plt_Heave.FillBelow();
843             Signal_Plt_Heave.MarkerSize = 0;
844             Signal_Plt_Heave.LineColor = Line_color_Blue;
845
846
847             SctPlt_Roll_Pitch_Pos.Plot.AxisAuto(0, 0.05);
848             Signal_Plt_Roll_Pitch.LineStyle = LineStyle.Solid;
849             Signal_Plt_Roll_Pitch.LineWidth = 1.5;
850             Signal_Plt_Roll_Pitch.FillBelow();
851             Signal_Plt_Roll_Pitch.MarkerSize = 0;
852             Signal_Plt_Roll_Pitch.LineColor = Line_color_Blue;
853
854             SctPlt_HeavePos.Refresh();
855             SctPlt_Roll_Pitch_Pos.Refresh();
856         }
857     }
858
859     public void ClearAllPlots()
860     {
861         // Clears heave and roll plots
862         SctPlt_HeavePos.Plot.Clear();
863         SctPlt_Roll_Pitch_Pos.Plot.Clear();
864         if(pLC_Com.DataToPlc.OpMode == OperationMode.Simulated)
865         {
866             MaxDataPoints = MaxDataPoints_Sim;
867             Plt_Heave_Pos_data = new float[MaxDataPoints_Sim];
868             Plt_Roll_pitch_Pos_data = new float[MaxDataPoints_Sim];
869             Signal_Plt_Heave = SctPlt_HeavePos.Plot.AddSignal(Plt_Heave_Pos_data,
    Plt_SampleRate_Read_Sim);
870             Signal_Plt_Roll_Pitch = SctPlt_Roll_Pitch_Pos.Plot.AddSignal(
    Plt_Roll_pitch_Pos_data, Plt_SampleRate_Read_Sim);
871         }
872         else
873         {
874             MaxDataPoints = MaxDataPoints_Log;
875             Plt_Heave_Pos_data = new float[MaxDataPoints_Log];
876             Plt_Roll_pitch_Pos_data = new float[MaxDataPoints_Log];
877             Signal_Plt_Heave = SctPlt_HeavePos.Plot.AddSignal(Plt_Heave_Pos_data,
    Plt_SamlpeRate_Read_Log);
878             Signal_Plt_Roll_Pitch = SctPlt_Roll_Pitch_Pos.Plot.AddSignal(
    Plt_Roll_pitch_Pos_data, Plt_SamlpeRate_Read_Log);
879         }
880
881     }
882
883     public float GetRollAngle(float cyl1Pos, float cyl2Pos, float cyl3Pos)
884     {
885         // Returns roll angle from given cylinder positions
```

```
886             float result = Convert.ToSingle(Math.Asin((cyl1Pos - ((cyl2Pos + cyl3Pos) / 2)) /
    PlattformCC) * (180 / Math.PI));
887             if (result < 0.6 && result > -0.6 && sl_PhaseDiff_sim.Value == 0 && pLC_Com.
    DataToPlc.OpMode == OperationMode.Simulated) result = 0; // Deadband +-0.4deg
888             return result;
889 
890         }
891 
892         public float GetHeavePos(float cyl1Pos, float cyl2Pos, float cyl3Pos)
893         {
894             // Returns Heave position from given cylinder positions
895             float result = Convert.ToSingle(cyl1Pos - ((Math.Sin(GetRollAngle(cyl1Pos, cyl2Pos,
     cyl3Pos) * (Math.PI / 180)) * PlattformCC) / 2));
896             if (result < 0.152 && result > 0.148 && sl_PhaseDiff_sim.Value == 180 && pLC_Com.
    DataToPlc.OpMode == OperationMode.Simulated) result = (float)0.15;// Deadband +-0.001m
897             return result;
898         }
899 
900         public float GetMpPos(float cyl1Pos, float cyl2Pos, float cyl3Pos)
901         {
902             // Returns Mp position from given cylinder positions
903             float result = Convert.ToSingle((TipVectY * Math.Sin(GetRollAngle(cyl1Pos, cyl2Pos,
     cyl3Pos) * (Math.PI/180))) + (TipVectZ * Math.Cos(GetRollAngle(cyl1Pos, cyl2Pos, cyl3Pos)
    * (Math.PI/180))) - TipVectZ + GetHeavePos(cyl1Pos, cyl2Pos, cyl3Pos) - 0.15);
904             return result;
905         }
906 
907         public float GetLoadPos(float AHCCenterPos, float WireLength, float MpPos)
908         {
909             // Returns Load position from given AHCCenterPos, Wirelength and MpPos
910             if (AHCCenterPos == 0) AHCCenterPos = WireLength;
911             float result = MpPos - (WireLength - AHCCenterPos);
912             return result;
913         }
914 
915         private void btn_Reset_Length_Click(object sender, RoutedEventArgs e)
916         {
917             // Reset wire length click
918             try
919             {
920                 float resetlengthVal = 0;
921                 resetlengthVal = - Convert.ToSingle(tb_ResetLengthValue.Text);
922                 // Set data and send to PLC
923                 int result = pLC_Com.SendData(pLC_Com.DataToPlc.OpMode, pLC_Com.DataToPlc.Cmd,
    pLC_Com.DataToPlc.Amplitude, pLC_Com.DataToPlc.Periode, pLC_Com.DataToPlc.PhaseDifference
924                                               , pLC_Com.DataToPlc.HeavePos, pLC_Com.DataToPlc
    .RollAngle, resetlengthVal, 1);
925 
926                 // Check if data is sent successfully and diable/enable buttons if sucess.
927                 if (result == BusProtocolErrors.FTALK_SUCCESS)
928                 {
929                     Btn_Start_Stop_Reset_click(Command.Reset);
930                 }
931             }
932             catch (Exception ex)
933             {
934                 MessageBox.Show(ex.Message);
935             }
936 
937         }
938 
939         private void btn_Submit_Settings_Click(object sender, RoutedEventArgs e)
940         {
941             // Submit tipvector settings
942             try
943             {
944                 TipVectX = Convert.ToSingle(tb_CraneTip_xPos.Text);
945                 TipVectY = Convert.ToSingle(tb_CraneTip_yPos.Text);
946                 TipVectZ = Convert.ToSingle(tb_CraneTip_zPos.Text);
947             }
948             catch (Exception ex)
949             {
950                 MessageBox.Show(ex.Message, "Error!");
951             }
952         }
953     }
954 }
```

Listing 7: PlattformUI.xaml.cs klasse i C#

15.2 PLC_Com.cs

```
using FieldTalk.Modbus.Master;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Threading;

namespace Plattform_UI
{
    /*************************************************************************
    Code information:
    PLC_Com class

    All comuniaction with PLC is done here.
    Data sent and received is also done here and saved in DataToPLc andDataFromPLc variables.

    Author: Sander V. Andersen and Lars Askild S. Aarvik
    Date: 19.05.2023
    *************************************************************************/


    // Defining usefull enums
    public enum OperationModeWinch { Standby, Manual, AHC, Error}
    public enum OperationMode { Manual, Simulated, Logfile }
    public enum Command { Stop, Start, Reset }

    internal class PLC_Com
    {
        // Defining variables
        public MbusUdpMasterProtocol ModbusProtocol { get; set; }
        public DataToPLC DataToPlc { get; set; }
        public DataFromPLC DataFromPlc { get; set; }
        public bool Stop { get; set; }
        const int SLAVEADDR = 1;
        const int STARTREFWRITE = 101;
        const int STARTREFREAD = 1;
        static object ReadWriteSync;
        PlattformUI plattform_ui;
        bool LastPlattformRunning;
        bool LastEmStop;
        Command LastStart;

        public PLC_Com()
        {
            // Initializing variables
            DataToPlc = new DataToPLC();
            DataFromPlc = new DataFromPLC();
            ModbusProtocol = new MbusUdpMasterProtocol();
            Stop = false;
            ReadWriteSync = new object();
            LastPlattformRunning = false;
            LastEmStop = true;
            LastStart = Command.Stop;
        }

        public int OpenProtocol(string ip)
        {
            // Open modbus protocoll and connect to PLC using modbus udp
            Stop = false;
            int result = 0;
            result =  ModbusProtocol.openProtocol(ip);
            // If success, send stop command to plattform
            if(result == BusProtocolErrors.FTALK_SUCCESS)
            {
                result = SendData(OperationMode.Manual, Command.Stop, 0, 0, 0, 0, 0, 0, 0);
            }
            return result;
        }

        public void CloseProtocol()
        {
            // Close modbus protocol
            Stop = true;
            Thread.Sleep(100); // Delay for recivetread to stop before closing mbus protocol
            ModbusProtocol.closeProtocol();
        }

        public int SendData(OperationMode OpMode, Command Cmd, float Amplitude, float Periode,
            float PhaseDiff, float HeavePos, float RollAngle, float ResetLengthVal, Int16
    ResetLengthCmd)
        {
            // Send selected data to PLC

            int result = 0;

```

```
88              // Set data
89              DataToPlc.OpMode = OpMode;
90              DataToPlc.Cmd = Cmd;
91              DataToPlc.Amplitude = Amplitude;
92              DataToPlc.Periode = Periode;
93              DataToPlc.PhaseDifference = PhaseDiff;
94              DataToPlc.HeavePos = HeavePos;
95              DataToPlc.RollAngle = RollAngle;
96              DataToPlc.ResetLengthVal = ResetLengthVal;
97              DataToPlc.ResetLengthCmd = ResetLengthCmd;
98
99              // Send to PLC
100             switch (DataToPlc.OpMode)
101             {
102                 case OperationMode.Manual:
103                 case OperationMode.Simulated:
104                     Monitor.Enter(ReadWriteSync);
105                     result = ModbusProtocol.writeMultipleRegisters(SLAVEADDR, STARTREFWRITE,
    DataToPlc.GetDataArr());
106                     Monitor.Exit(ReadWriteSync);
107                     break;
108                 case OperationMode.Logfile:
109                     switch (Cmd)
110                     {
111                         case Command.Stop:
112                             ThreadPool.QueueUserWorkItem(SendLogFileData, null);
113                             break;
114                         case Command.Start:
115                             Monitor.Enter(ReadWriteSync);
116                             result = ModbusProtocol.writeMultipleRegisters(SLAVEADDR,
    STARTREFWRITE, DataToPlc.GetDataArr());
117                             Monitor.Exit(ReadWriteSync);
118                             break;
119                         case Command.Reset:
120                             break;
121                         default:
122                             break;
123                     }
124                     break;
125                 default:
126                     break;
127             }
128
129             //Reset reset length command
130             DataToPlc.ResetLengthCmd = 0;
131
132             // Check for Modbus errors
133             checkModbusResult(result);
134             return result;
135         }
136
137         void SendLogFileData(object o)
138         {
139             // Send logifle data to PLC
140
141             UpdateProgressLogFileStartDelegate UPLSD = new UpdateProgressLogFileStartDelegate(
    plattform_ui.UpdateProgressStartLogfile);
142             int result = 0;
143             int count = 0;
144             bool finished = false;
145             Int16[] dataArr = DataToPlc.GetDataArrLogFile();
146             Int16[] dataArr_temp = new Int16[122];
147
148             for (int i = 0; i < dataArr.Length; i++)
149             {
150                 dataArr_temp[count] = dataArr[i];
151
152                 // Send 121 registers at the time.. max possible
153                 if (count >= 121)
154                 {
155                     Monitor.Enter(ReadWriteSync);
156                     result = ModbusProtocol.writeMultipleRegisters(SLAVEADDR, i - 20,
    dataArr_temp);
157                     Monitor.Exit(ReadWriteSync);
158
159                     checkModbusResult(result);
160
161                     count = -1;
162                     for (int j = 0; j < dataArr_temp.Length; j++)
163                     {
164                         dataArr_temp[j] = 0;
165                     }
166                 }
167                 count++;
168                 plattform_ui.Dispatcher.Invoke(UPLSD, (((double)i / (double)dataArr.Length) * 9
    9), finished);
169             }
170             Monitor.Enter(ReadWriteSync);
171             result = ModbusProtocol.writeMultipleRegisters(SLAVEADDR, dataArr.Length - count +
    STARTREFWRITE, dataArr_temp);
172             Monitor.Exit(ReadWriteSync);
```

```
            checkModbusResult(result);

            plattform_ui.Dispatcher.Invoke(UPLSD, 100, finished);
            Thread.Sleep(1000);
            finished = true;
            plattform_ui.Dispatcher.Invoke(UPLSD, -1, finished);
        }

        void checkModbusResult(int result)
        {
            // Check for errors in modbus transmissions
            if (result != BusProtocolErrors.FTALK_SUCCESS)
            {
                MessageBox.Show(BusProtocolErrors.getBusProtocolErrorText(result), "Modbus error");
            }
        }

        public void StartDataReciver(PlattformUI Form)
        {
            // Start thread for receiving data from PLC

            plattform_ui = Form;
            ThreadStart ts = new ThreadStart(ReceiveData);
            Thread receiverThread = new Thread(ts);
            receiverThread.IsBackground = true;
            receiverThread.Start();

            // Start comcount thread
            ThreadStart ts2 = new ThreadStart(SendUpdatedComCount);
            Thread ComCountThread = new Thread(ts2);
            ComCountThread.IsBackground = true;
            ComCountThread.Start();
        }

        void ReceiveData()
        {
            // Thread for receiving data from PLC
            // Reads every 20ms

            Int16[] data_arr = new Int16[28];
            while(!Stop)
            {
                Monitor.Enter(ReadWriteSync);
                int result = ModbusProtocol.readMultipleRegisters(SLAVEADDR, STARTREFREAD, data_arr);
                Monitor.Exit(ReadWriteSync);
                DataFromPlc.UpdateValues(data_arr);

                if ((DataToPlc.OpMode == OperationMode.Manual || DataToPlc.Cmd == Command.Reset) && !(DataToPlc.Cmd == Command.Stop))
                {
                    // set plattform in stop mode when resetting is finished
                    if (!DataFromPlc.Plattform_Running && LastPlattformRunning)
                    {
                        // Small delay for reset pos in PLC to work
                        Thread.Sleep(50);
                        StopPlattformDelegate SPD = new StopPlattformDelegate(plattform_ui.StopPlattform);
                        plattform_ui.Dispatcher.Invoke(SPD, DataToPlc.OpMode);

                        // Clear all plots when finished reset
                        ClearAllPlotsDelegate CAPD = new ClearAllPlotsDelegate(plattform_ui.ClearAllPlots);
                        plattform_ui.Dispatcher.Invoke(CAPD);
                    }
                }

                if(DataToPlc.Cmd == Command.Start || DataToPlc.Cmd == Command.Reset)
                {
                    // Update live plots
                    UpdatedLivePlotsDelegate ULPD = new UpdatedLivePlotsDelegate(plattform_ui.UpdateLivePlots);
                    plattform_ui.Dispatcher.Invoke(ULPD);
                }

                if ((DataFromPlc.EM_Stop == false) && (LastEmStop == true))
                {
                    // Set plattform in stop mode if Em stop is pressed
                    StopPlattformDelegate SPD = new StopPlattformDelegate(plattform_ui.StopPlattform);
                    plattform_ui.Dispatcher.Invoke(SPD, DataToPlc.OpMode);
                }

                // Update last values
                LastStart = DataToPlc.Cmd;
                LastEmStop = DataFromPlc.EM_Stop;
                LastPlattformRunning = DataFromPlc.Plattform_Running;

                // Update GUI
                UpdateGUIDelegate UGD = new UpdateGUIDelegate(plattform_ui.UpdateGUI);
```

```
257                 plattform_ui.Dispatcher.Invoke(UGD);
258
259
260                 // Lost connection.. closing protocol
261                 if(result != BusProtocolErrors.FTALK_SUCCESS)
262                 {
263                     // Disconnecting
264                     CloseMbusProtocolDelegate CMPD = new CloseMbusProtocolDelegate(plattform_ui
    .CloseMbusProtocol);
265                     plattform_ui.Dispatcher.Invoke(CMPD);
266                 }
267
268                 Thread.Sleep(20);
269             }
270         }
271
272         void SendUpdatedComCount()
273         {
274             // Send com count to PLC
275             // This is for PLC to check for com errors with HMI and stop automatically if com
    error appears
276             int result = 0;
277
278             while(!Stop)
279             {
280                 if (DataToPlc.ComCount >= 32000) DataToPlc.ComCount = 0;
281                 else DataToPlc.ComCount++;
282
283                 Monitor.Enter(ReadWriteSync);
284                 result = ModbusProtocol.writeMultipleRegisters(SLAVEADDR, STARTREFWRITE,
    DataToPlc.GetDataArr());
285                 Monitor.Exit(ReadWriteSync);
286
287                 if (result != BusProtocolErrors.FTALK_SUCCESS)
288                 {
289                     // Disconnecting
290                     CloseMbusProtocolDelegate CMPD = new CloseMbusProtocolDelegate(plattform_ui
    .CloseMbusProtocol);
291                     plattform_ui.Dispatcher.Invoke(CMPD);
292                 }
293
294                 Thread.Sleep(200);
295             }
296         }
297
298     }
299 }
```

Listing 8: PLC$_{C}om.csklasseiC\#$

15.3 DataFromPLC.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Plattform_UI
{
    internal class DataFromPLC
    {

        /*************************************************************************
        Code information:
        DataFromPLC class

        Datatype containing all data from PLC
        UpdateValues method updates all variables with data from PLC with given INT16 array

        Author: Sander V. Andersen and Lars Askild S. Aarvik
        Date: 19.05.2023
        *************************************************************************/

        // Defining variables
        public float Cyl1Pos_SP { get; set; }
        public float Cyl2Pos_SP { get; set; }
        public float Cyl3Pos_SP { get; set; }
        public float Cyl1Pos { get; set; }
        public float Cyl2Pos { get; set; }
        public float Cyl3Pos { get; set; }
        public float Cyl1Spd { get; set; }
        public float Cyl2Spd { get; set; }
        public float Cyl3Spd { get; set; }
        public float WinchPos { get; set; }
        public float WinchSpd { get; set; }
        public bool Drive_Error { get; set; }
        public bool Plattform_Running { get; set; }
        public bool EM_Stop { get; set; }
        public OperationModeWinch OpModeWinch { get; set; }
        public float AHC_CenterPos { get; set; }

        public void UpdateValues(Int16[] data_arr)
        {
            // Updates all variables with data from PLC with given INT16 array

            List<byte> byte_list = new List<byte>();

            for (int i = 0; i < data_arr.Length; i++)
            {
                byte[] temp = BitConverter.GetBytes(data_arr[i]);
                for (int j = 0; j < temp.Length; j++)
                {
                    byte_list.Add(temp[j]);
                }
            }

            byte[] byte_arr = new byte[data_arr.Length * 2];
            for (int i = 0; i < byte_list.Count; i++)
            {
                byte_arr[i] = byte_list[i];
            }

            Cyl1Pos_SP = BitConverter.ToSingle(byte_arr, 0) / 1000;
            Cyl2Pos_SP = BitConverter.ToSingle(byte_arr, 4) / 1000;
            Cyl3Pos_SP = BitConverter.ToSingle(byte_arr, 8) / 1000;
            Cyl1Pos = BitConverter.ToSingle(byte_arr, 12) / 1000;
            Cyl2Pos = BitConverter.ToSingle(byte_arr, 16) / 1000;
            Cyl3Pos = BitConverter.ToSingle(byte_arr, 20) / 1000;
            Plattform_Running = ((byte_arr[24] >> 0) & 1) != 0;
            EM_Stop = ((byte_arr[26] >> 0) & 1) != 0;
            Cyl1Spd = BitConverter.ToSingle(byte_arr, 28) / 100000;
            Cyl2Spd = BitConverter.ToSingle(byte_arr, 32) / 100000;
            Cyl3Spd = BitConverter.ToSingle(byte_arr, 36) / 100000;
            WinchPos = BitConverter.ToSingle(byte_arr, 40) / 1000;
            WinchSpd = BitConverter.ToSingle(byte_arr, 44) / 1000;
            AHC_CenterPos = BitConverter.ToSingle(byte_arr, 48) / 1000;
            if (BitConverter.ToInt16(byte_arr, 52) == 3) OpModeWinch = OperationModeWinch.Error
    ;
            else if (BitConverter.ToInt16(byte_arr, 52) == 2) OpModeWinch = OperationModeWinch.
    AHC;
            else if (BitConverter.ToInt16(byte_arr, 52) == 1) OpModeWinch = OperationModeWinch.
    Manual;
            else OpModeWinch = OperationModeWinch.Standby;
        }
    }
}
```

Listing 9: DataFromPLC.cs klasse i C#

15.4 DataToPLC.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Media.TextFormatting;

namespace Plattform_UI
{
    internal class DataToPLC
    {

        /*************************************************************************
        Code information:
        DataToPLC class

        Datatype containing all data to PLC
        Containing methods for returning INT16 array that can be sent with modbus to PLC

        Author: Sander V. Andersen and Lars Askild S. Aarvik
        Date: 19.05.2023
        *************************************************************************/

        // Defining variables
        public Int16 ComCount { get; set; }
        public OperationMode OpMode { get; set; }
        public Command Cmd { get; set; }
        public float Amplitude { get; set; }
        public float Periode { get; set; }
        public float PhaseDifference { get; set; }
        public float HeavePos { get; set; }
        public float RollAngle { get; set; }
        public Logfile Logfile { get; set; }
        public float ResetLengthVal { get; set; }
        public Int16 ResetLengthCmd { get; set; }

        public DataToPLC()
        {
            // Initializing variables
            ComCount = 0;
            OpMode = OperationMode.Manual;
            Cmd = Command.Stop;
            Amplitude = 0;
            Periode = 0;
            PhaseDifference = 0;
            HeavePos = 0;
            RollAngle = 0;
            Logfile = new Logfile();
            ResetLengthVal = 0;
            ResetLengthCmd = 0;
        }

        public Int16[] GetDataArr()
        {
            // Returns INT16 array with all data to PLC without logfile data

            Int16[] dataArr = new Int16[16];
            dataArr[0] = ComCount;
            dataArr[1] = (Int16)OpMode;
            dataArr[2] = (Int16)Cmd;
            dataArr[3] = ConvertToWordArray(Amplitude * 1000)[0];
            dataArr[4] = ConvertToWordArray(Amplitude * 1000)[1];
            dataArr[5] = ConvertToWordArray(Periode * 1000)[0];
            dataArr[6] = ConvertToWordArray(Periode * 1000)[1];
            dataArr[7] = ConvertToWordArray(PhaseDifference)[0];
            dataArr[8] = ConvertToWordArray(PhaseDifference)[1];
            dataArr[9] = ConvertToWordArray(HeavePos * 1000)[0];
            dataArr[10] = ConvertToWordArray(HeavePos * 1000)[1];
            dataArr[11] = ConvertToWordArray(RollAngle)[0];
            dataArr[12] = ConvertToWordArray(RollAngle)[1];
            dataArr[13] = ConvertToWordArray(ResetLengthVal * 1000)[0];
            dataArr[14] = ConvertToWordArray(ResetLengthVal * 1000)[1];
            dataArr[15] = ResetLengthCmd;
            return dataArr;
        }

        public Int16[] GetDataArrLogFile()
        {
            // Returns INT16 array with all data to PLC including logfile data when sending
    logfile data to plc

            Int16[] dataArr = new Int16[16 + (Logfile.Cylinder1_Pos.Count * 4)];
            dataArr[0] = ComCount;
            dataArr[1] = (Int16)OpMode;
            dataArr[2] = (Int16)Cmd;
            dataArr[3] = ConvertToWordArray(Amplitude * 1000)[0];
```

```
            dataArr[4] = ConvertToWordArray(Amplitude * 1000)[1];
            dataArr[5] = ConvertToWordArray(Periode * 1000)[0];
            dataArr[6] = ConvertToWordArray(Periode * 1000)[1];
            dataArr[7] = ConvertToWordArray(PhaseDifference)[0];
            dataArr[8] = ConvertToWordArray(PhaseDifference)[1];
            dataArr[9] = ConvertToWordArray(HeavePos * 1000)[0];
            dataArr[10] = ConvertToWordArray(HeavePos * 1000)[1];
            dataArr[11] = ConvertToWordArray(RollAngle)[0];
            dataArr[12] = ConvertToWordArray(RollAngle)[1];
            dataArr[13] = ConvertToWordArray(ResetLengthVal * 1000)[0];
            dataArr[14] = ConvertToWordArray(ResetLengthVal * 1000)[1];
            dataArr[15] = ResetLengthCmd;

            int count = 16;
            for (int i = 0; i < Logfile.Cylinder1_Pos.Count; i++)
            {
                dataArr[count] = (Int16)(Logfile.Cylinder1_Pos[i] * 100000);
                dataArr[count + 1] = (Int16)(Logfile.Cylinder1_Vel[i] * 100000);
                dataArr[count + 2] = (Int16)(Logfile.Cylinder2_3_Pos[i] * 100000);
                dataArr[count + 3] = (Int16)(Logfile.Cylinder2_3_Vel[i] * 100000);

                count = count + 4;
            }

            return dataArr;
        }

        Int16[] ConvertToWordArray(float data)
        {
            // Converts one float value to two INT16 values.. Since float is 32 bit

            Int16[] data_arr = new Int16[2];
            byte[] b_data = BitConverter.GetBytes(data);
            data_arr[0] = BitConverter.ToInt16(b_data, 0);
            data_arr[1] = BitConverter.ToInt16(b_data, 2);
            return data_arr;
        }

    }
}
```

Listing 10: DataToPLC.cs klasse i C#

15.5 Logfile.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;
using System.Windows;
using FieldTalk.Modbus.Master;
using System.Windows.Markup;
using System.Windows.Shapes;
using System.Security.Permissions;

namespace Plattform_UI
{
    public class Logfile
    {

        /*************************************************************************
        Code information:
        Logfile class

        All calculations and filehandling is done in this class
        Reading data from file
        Scaling waves
        Calculating cylinder positions and velocities
        Lowpass filtering calculated cylinder velocities

        Author: Sander V. Andersen and Lars Askild S. Aarvik
        Date: 19.05.2023
        *************************************************************************/


        // Defining variables
        public string FileName { get; set; }
        public Int32 LogLength { get; set; } //Number of lines to read from logfile based on selected slider length (min)
        public List<float> Heave_Pos { get; set; }
        public List<float> Heave_Pos_Scaled { get; set; }
        public List<float> Heave_Vel { get; set; }
        public List<float> Roll_Angle { get; set; }
        public List<float> Roll_Angle_Scaled { get; set; }
        public List<float> Pitch_Angle { get; set; }
        public List<float> Pitch_Angle_Scaled { get; set; }
        public List<float> Cylinder1_Pos { get; set; }
        public List<float> Cylinder2_3_Pos { get; set; }
        public List<float> Cylinder1_Vel { get; set; }
        public List<float> Cylinder2_3_Vel { get; set; }
        public double MaxHeaveScaleSlider { get; set; }
        public double MaxHeavPos { get; set; }
        public double MaxAngle { get; set; }
        int pos_index;
        int vel_index;
        int roll_index;
        int pitch_index;
        int TextStartCount;
        double MaxMovement;
        double MaxCylSpeed;
        double maxPossibleHeavPos;
        float ScaleFactorPos;
        float ScaleFactorVel;
        List<string> Lines = new List<string>();
        const int SAMPLETIME = 100; // ms data is sampled with
        const int FREQUENCY = 10; // Frequency data is sample with
        public static double MAXCYLINDERMOVE = 0.125; // m max movement in each direction on cylinder
        const string MRU_pos = "mru_1_mruHeavePos";
        const string MRU_vel = "mru_1_mruHeaveVel";
        const string MRU_roll = "mru_1_roll";
        const string MRU_pitch = "mru_1_pitch";
        const string StartText = "Time";

        public Logfile()
        {
            // Initializing variables
            FileName = "";
            LogLength = 0;
            Heave_Pos = new List<float>();
            Heave_Pos_Scaled = new List<float>();
            Heave_Vel = new List<float>();
            Roll_Angle = new List<float>();
            Roll_Angle_Scaled = new List<float>();
            Pitch_Angle = new List<float>();
            Pitch_Angle_Scaled = new List<float>();
            Cylinder1_Pos = new List<float>();
            Cylinder2_3_Pos = new List<float>();
            Cylinder1_Vel = new List<float>();
            Cylinder2_3_Vel = new List<float>();
            pos_index = 0;
```

```
            vel_index = 0;
            roll_index = 0;
            pitch_index = 0;
            MaxMovement = 0;
            ScaleFactorPos = 0;
            ScaleFactorVel = 0;
            TextStartCount = 0;
            MaxCylSpeed = 0;
            MaxAngle = 0;
            MaxHeavPos = 0;
            maxPossibleHeavPos = 0;
            MaxHeaveScaleSlider = 0;
        }



        public double GetLogfileLength()
        {
            // Returning logfile length in minutes.

            double result = 0;
            int lineCount = 0;
            try
            {
                lineCount = File.ReadLines(FileName).Count();
            }
            catch (Exception ex)
            {
                MessageBox.Show("Error", "Error: " + ex.Message);
            }
            result = ((lineCount - 6) / FREQUENCY) / 60;
            return result;
        }

        public void UpdateLogLength(double sl_Value)
        {
            // Updated loglength variable based on slider value

            try
            {
                LogLength = Convert.ToInt32(Math.Floor(sl_Value * 60 * 10));
            }
            catch (Exception ex)
            {
                MessageBox.Show("Error", "cannot convert slider value to integer.\n" + ex.
    Message);
            }
        }



        public void ReadDataFromFile()
        {
            // Read data from file and save to lists

            // Deleting all elemnts from lists.
            Heave_Vel.Clear();
            Roll_Angle.Clear();
            Heave_Pos.Clear();
            Pitch_Angle.Clear();

            try
            {
                Lines = File.ReadLines(FileName).ToList();

                // Deletes unnecessary lines from logfile.csv
                for (int i = 0; i < Lines.Count; i++)
                {
                    string[] str = Lines[i].Split(';');
                    if (str[0] == StartText)
                        TextStartCount = i;
                }

                Lines.RemoveRange(0, TextStartCount);

                // Finding data positions
                string[] dataName = Lines[0].Split(';');
                for (int i = 0; i < dataName.Length; i++)
                {
                    if (dataName[i] == MRU_pos) pos_index = i;
                    if (dataName[i] == MRU_vel) vel_index = i;
                    if (dataName[i] == MRU_roll) roll_index = i;
                    if (dataName[i] == MRU_pitch) pitch_index = i;
                }
                Lines.RemoveAt(0);

                // Updating values from logfile
                for (int i = 0; i < LogLength - 1; i++)
                {
                    string[] data = Lines[i].Split(";");
                    Heave_Pos.Add(float.Parse(data[pos_index], System.Globalization.CultureInfo
```

```
.InvariantCulture));
177                 Heave_Vel.Add(float.Parse(data[vel_index], System.Globalization.CultureInfo
.InvariantCulture));
178                 Roll_Angle.Add(float.Parse(data[roll_index], System.Globalization.
CultureInfo.InvariantCulture));
179                 Pitch_Angle.Add(float.Parse(data[pitch_index], System.Globalization.
CultureInfo.InvariantCulture));
180             }
181
182         }
183         catch (Exception ex)
184         {
185             MessageBox.Show("Error", "Error: " + ex.Message);
186         }
187     }
188
189
190
191     public void Calculate_Scale_CylinderPosVel(bool Pitch_Checked, float ScaleHeave_Value,
float ScaleRollPitch_Value)
192     {
193         // Calculating and scaling sylinder positions and velocities
194
195         MaxCylSpeed = 0;
196         MaxMovement = 0;
197
198         // Deleting all elements from lists.
199         Cylinder1_Pos.Clear();
200         Cylinder1_Vel.Clear();
201         Cylinder2_3_Pos.Clear();
202         Cylinder2_3_Vel.Clear();
203         Heave_Pos_Scaled.Clear();
204         Roll_Angle_Scaled.Clear();
205         Pitch_Angle_Scaled.Clear();
206
207
208
209
210         if (ScaleHeave_Value == 1 && ScaleRollPitch_Value == 1)
211         {
212
213             // Scale for postion limits
214             // Finidng max MP with max angle
215             if (Pitch_Checked) // Use pitch value
216             {
217                 for (int i = 0; i < LogLength; i++)
218                 {
219                     Cylinder1_Pos.Add(Convert.ToSingle(Heave_Pos[i] + ((Math.Sin(Convert.
ToDouble(Pitch_Angle[i] * (Math.PI / 180))) * PlattformUI.PlattformCC) / 2)));
220                     Cylinder2_3_Pos.Add(Convert.ToSingle(Heave_Pos[i] - ((Math.Sin(Convert.
ToDouble(Pitch_Angle[i] * (Math.PI / 180))) * PlattformUI.PlattformCC) / 2)));
221                     if (Math.Abs(Cylinder1_Pos[i]) > MaxMovement)
222                     {
223                         MaxHeavPos = Heave_Pos[i];
224                         MaxAngle = Pitch_Angle[i];
225                         MaxMovement = Math.Abs(Cylinder1_Pos[i]);
226
227                     }
228                     if (Math.Abs(Cylinder2_3_Pos[i]) > MaxMovement)
229                     {
230                         MaxHeavPos = Heave_Pos[i];
231                         MaxAngle = Pitch_Angle[i];
232                         MaxMovement = Math.Abs(Cylinder2_3_Pos[i]);
233                     }
234                 }
235             }
236             else // Use Roll Angle
237             {
238                 for (int i = 0; i < LogLength; i++)
239                 {
240                     Cylinder1_Pos.Add(Convert.ToSingle(Heave_Pos[i] + ((Math.Sin(Convert.
ToDouble(Roll_Angle[i] * (Math.PI / 180))) * PlattformUI.PlattformCC) / 2)));
241                     Cylinder2_3_Pos.Add(Convert.ToSingle(Heave_Pos[i] - ((Math.Sin(Convert.
ToDouble(Roll_Angle[i] * (Math.PI / 180))) * PlattformUI.PlattformCC) / 2)));
242                     if (Math.Abs(Cylinder1_Pos[i]) > MaxMovement)
243                     {
244                         MaxHeavPos = Heave_Pos[i];
245                         MaxAngle = Roll_Angle[i];
246                         MaxMovement = Math.Abs(Cylinder1_Pos[i]);
247
248                     }
249                     if (Math.Abs(Cylinder2_3_Pos[i]) > MaxMovement)
250                     {
251                         MaxHeavPos = Heave_Pos[i];
252                         MaxAngle = Roll_Angle[i];
253                         MaxMovement = Math.Abs(Cylinder2_3_Pos[i]);
254                     }
255                 }
256             }
257
258
```

```
259             maxPossibleHeavPos = MAXCYLINDERMOVE - ((Math.Sin(MaxAngle * (Math.PI / 180)) *
     PlattformUI.PlattformCC) / 2);
260             if (Math.Abs(maxPossibleHeavPos) > MAXCYLINDERMOVE + ((Math.Sin(MaxAngle * (
    Math.PI / 180)) * PlattformUI.PlattformCC) / 2)) maxPossibleHeavPos = MAXCYLINDERMOVE + ((
    Math.Sin(MaxAngle * (Math.PI / 180)) * PlattformUI.PlattformCC) / 2);
261
262             ScaleFactorPos =  (float)(MaxHeavPos / maxPossibleHeavPos);
263
264             //Scale heavpos
265             for (int i = 0; i < LogLength; i++)
266             {
267                 Heave_Pos_Scaled.Add(Heave_Pos[i] / ScaleFactorPos);
268             }
269
270             Cylinder1_Pos.Clear();
271             Cylinder2_3_Pos.Clear();
272
273             if (Pitch_Checked) // Use pitch value
274             {
275                 for (int i = 0; i < LogLength; i++)
276                 {
277                     Cylinder1_Pos.Add(Convert.ToSingle(Heave_Pos_Scaled[i] + ((Math.Sin(
    Convert.ToDouble(Pitch_Angle[i] * (Math.PI / 180))) * PlattformUI.PlattformCC) / 2)));
278                     Cylinder2_3_Pos.Add(Convert.ToSingle(Heave_Pos_Scaled[i] - ((Math.Sin(
    Convert.ToDouble(Pitch_Angle[i] * (Math.PI / 180))) * PlattformUI.PlattformCC) / 2)));
279                 }
280             }
281             else // Use Roll Angle
282             {
283                 for (int i = 0; i < LogLength; i++)
284                 {
285                     Cylinder1_Pos.Add(Convert.ToSingle(Heave_Pos_Scaled[i] + ((Math.Sin(
    Convert.ToDouble(Roll_Angle[i] * (Math.PI / 180))) * PlattformUI.PlattformCC) / 2)));
286                     Cylinder2_3_Pos.Add(Convert.ToSingle(Heave_Pos_Scaled[i] - ((Math.Sin(
    Convert.ToDouble(Roll_Angle[i] * (Math.PI / 180))) * PlattformUI.PlattformCC) / 2)));
287                 }
288             }
289
290
291             // Scale for position done
292
293             // Scale for velocity limits
294             // Scales down and calculate maxspeed for each cyl
295             for (int i = 1; i < Cylinder1_Pos.Count; i++)
296             {
297                 Cylinder1_Vel.Add((Cylinder1_Pos[i] - Cylinder1_Pos[i - 1]) / ((float)
    SAMPLETIME / 1000));
298                 Cylinder2_3_Vel.Add((Cylinder2_3_Pos[i] - Cylinder2_3_Pos[i - 1]) / ((float
    )SAMPLETIME / 1000));
299                 if (Math.Abs(Cylinder1_Vel[i - 1]) > MaxCylSpeed) MaxCylSpeed = Math.Abs(
    Cylinder1_Vel[i - 1]);
300                 if (Math.Abs(Cylinder2_3_Vel[i - 1]) > MaxCylSpeed) MaxCylSpeed = Math.Abs(
    Cylinder2_3_Vel[i - 1]);
301
302             }
303
304             // finds the scalefactor for velocity
305             if (MaxCylSpeed > PlattformUI.MaxSpeedCylinder)
306             {
307                 ScaleFactorVel = (float)(MaxCylSpeed / PlattformUI.MaxSpeedCylinder);
308             }
309             else ScaleFactorVel = 1;
310
311
312             Cylinder1_Pos.Clear();
313             Cylinder2_3_Pos.Clear();
314             Cylinder1_Vel.Clear();
315             Cylinder2_3_Vel.Clear();
316
317             //Scale heavpos
318             for (int i = 0; i < LogLength; i++)
319             {
320                 Heave_Pos_Scaled[i] = Heave_Pos_Scaled[i] / ScaleFactorVel;
321             }
322
323             // Calculate evryting again with correct scaling.
324             if (Pitch_Checked) // Use pitch value
325             {
326                 for (int i = 0; i < LogLength; i++)
327                 {
328                     Cylinder1_Pos.Add(Convert.ToSingle(Heave_Pos_Scaled[i] + ((Math.Sin(
    Convert.ToDouble(Pitch_Angle[i] * (Math.PI / 180))) * PlattformUI.PlattformCC) / 2)));
329                     Cylinder2_3_Pos.Add(Convert.ToSingle(Heave_Pos_Scaled[i] - ((Math.Sin(
    Convert.ToDouble(Pitch_Angle[i] * (Math.PI / 180))) * PlattformUI.PlattformCC) / 2)));
330                 }
331             }
332             else // Use Roll Angle
333             {
334                 for (int i = 0; i < LogLength; i++)
335                 {
336                     Cylinder1_Pos.Add(Convert.ToSingle(Heave_Pos_Scaled[i] + ((Math.Sin(
```

```
    Convert.ToDouble(Roll_Angle[i] * (Math.PI / 180))) * PlattformUI.PlattformCC) / 2)));
337                     Cylinder2_3_Pos.Add(Convert.ToSingle(Heave_Pos_Scaled[i] - ((Math.Sin(
    Convert.ToDouble(Roll_Angle[i] * (Math.PI / 180))) * PlattformUI.PlattformCC) / 2)));
338                 }
339             }
340 
341 
342             for (int i = 1; i < Cylinder1_Pos.Count; i++)
343             {
344                 Cylinder1_Vel.Add((Cylinder1_Pos[i] - Cylinder1_Pos[i - 1]) / ((float)
    SAMPLETIME / 1000));
345                 Cylinder2_3_Vel.Add((Cylinder2_3_Pos[i] - Cylinder2_3_Pos[i - 1]) / ((float
    )SAMPLETIME / 1000));
346             }
347 
348             Cylinder1_Pos.RemoveAt(Cylinder1_Pos.Count - 1);
349             Cylinder2_3_Pos.RemoveAt(Cylinder2_3_Pos.Count - 1);
350 
351             MaxHeaveScaleSlider = 100 / (ScaleFactorPos + ScaleFactorVel);
352 
353             // Updated scaled list
354             for (int i = 0; i < LogLength; i++)
355             {
356                 Roll_Angle_Scaled.Add(Roll_Angle[i]);
357                 Pitch_Angle_Scaled.Add(Pitch_Angle[i]);
358             }
359         }
360 
361         // Scaling from sliders
362         else
363         {
364             MaxMovement = 0;
365 
366             // Scale all values
367             for (int i = 0; i < LogLength; i++)
368             {
369                 Heave_Pos_Scaled.Add(Heave_Pos[i] * (ScaleHeave_Value / 100));
370                 Pitch_Angle_Scaled.Add(Pitch_Angle[i] * (ScaleRollPitch_Value / 100));
371                 Roll_Angle_Scaled.Add(Roll_Angle[i] * (ScaleRollPitch_Value / 100));
372             }
373 
374 
375             if (Pitch_Checked) // Use pitch value
376             {
377                 for (int i = 0; i < LogLength; i++)
378                 {
379                     Cylinder1_Pos.Add(Convert.ToSingle(Heave_Pos_Scaled[i] + ((Math.Sin(
    Convert.ToDouble(Pitch_Angle_Scaled[i] * (Math.PI / 180))) * PlattformUI.PlattformCC) / 2))
    );
380                     Cylinder2_3_Pos.Add(Convert.ToSingle(Heave_Pos_Scaled[i] - ((Math.Sin(
    Convert.ToDouble(Pitch_Angle_Scaled[i] * (Math.PI / 180))) * PlattformUI.PlattformCC) / 2))
    );
381                     if (Math.Abs(Cylinder1_Pos[i]) > MaxMovement)
382                     {
383                         MaxHeavPos = Heave_Pos_Scaled[i];
384                         MaxMovement = Math.Abs(Cylinder1_Pos[i]);
385 
386                     }
387                     if (Math.Abs(Cylinder2_3_Pos[i]) > MaxMovement)
388                     {
389                         MaxHeavPos = Heave_Pos_Scaled[i];
390                         MaxMovement = Math.Abs(Cylinder2_3_Pos[i]);
391                     }
392                 }
393 
394             }
395             else // Use Roll Angle
396             {
397                 for (int i = 0; i < LogLength; i++)
398                 {
399                     Cylinder1_Pos.Add(Convert.ToSingle(Heave_Pos_Scaled[i] + ((Math.Sin(
    Convert.ToDouble(Roll_Angle_Scaled[i] * (Math.PI / 180))) * PlattformUI.PlattformCC) / 2)))
    ;
400                     Cylinder2_3_Pos.Add(Convert.ToSingle(Heave_Pos_Scaled[i] - ((Math.Sin(
    Convert.ToDouble(Roll_Angle_Scaled[i] * (Math.PI / 180))) * PlattformUI.PlattformCC) / 2)))
    ;
401                     if (Math.Abs(Cylinder1_Pos[i]) > MaxMovement)
402                     {
403                         MaxHeavPos = Heave_Pos_Scaled[i];
404                         MaxMovement = Math.Abs(Cylinder1_Pos[i]);
405 
406                     }
407                     if (Math.Abs(Cylinder2_3_Pos[i]) > MaxMovement)
408                     {
409                         MaxHeavPos = Heave_Pos_Scaled[i];
410                         MaxMovement = Math.Abs(Cylinder2_3_Pos[i]);
411                     }
412                 }
413             }
414 
415 
```

```
                for (int i = 1; i < Cylinder1_Pos.Count; i++)
                {
                    Cylinder1_Vel.Add((Cylinder1_Pos[i] - Cylinder1_Pos[i - 1]) / ((float)
    SAMPLETIME / 1000));
                    Cylinder2_3_Vel.Add((Cylinder2_3_Pos[i] - Cylinder2_3_Pos[i - 1]) / ((float
    )SAMPLETIME / 1000));
                }

                Cylinder1_Pos.RemoveAt(Cylinder1_Pos.Count - 1);
                Cylinder2_3_Pos.RemoveAt(Cylinder2_3_Pos.Count - 1);



            }


            //Lowpass filtering cylinder velocities
            for (int i = 0; i < Cylinder1_Vel.Count - 1; i++)
            {
                Cylinder1_Vel[i] = (Cylinder1_Vel[i] + Cylinder1_Vel[i + 1]) / 2;
                Cylinder2_3_Vel[i] = (Cylinder2_3_Vel[i] + Cylinder2_3_Vel[i + 1]) / 2;
            }


        }

    }


}
```

Listing 11: Logfile.cs klasse i C#

16 Appendiks H - Kildekode PLS

ESM1

cycle10

Task Type	Event Name	Interval	Priority	Threshold	Watchdog	Comment
Cyclic task		10	0	0	100	

Program Name	Component Name	Program Type	Comment
PlattformControl	Arp.Plc.Eclr	PlattformControl	
Camera_Com1	Arp.Plc.Eclr	Camera_Com	

cycle2

Task Type	Event Name	Interval	Priority	Threshold	Watchdog	Comment
Cyclic task		2	0	0	100	

Program Name	Component Name	Program Type	Comment
ModBus	Arp.Plc.Eclr	ModBus	

ESM2

cycle10_Logfile

Task Type	Event Name		Interval	Priority	Threshold	Watchdog	Comment
Cyclic task			10	0	0	100	
Program Name		**Component Name**	**Program Type**	**Comment**			
LogFile1		Arp.Plc.Eclr	LogFile				

cycle10_WinchControl

Task Type	Event Name		Interval	Priority	Threshold	Watchdog	Comment
Cyclic task			10	0	0	100	
Program Name		**Component Name**	**Program Type**	**Comment**			
WinchControl1		Arp.Plc.Eclr	WinchControl				

Identity

Network name	network01
Top-level domain	

IP range

Start IP address	192.168.127.2
End IP address	192.168.127.254
Subnet mask	255.255.255.0
Default gateway	

SNMP

Version	SNMP v2c

SNMP v1 / v2c

Read community	public
Write community	private

SNMP v3

Username	
Authentication	MD5
Authentication passphrase	
Privacy protocol	DES
Privacy passphrase	

Components / Camera / fbScaleZPos / Code

```
/*******************************************************************************
Code information:
Function block that scales down data z position-length given from camerea to 0.0 - 0.3 [m].


Author: Sander V. Andersen and Lars Askild S. Aarvik
Date: 09.05.2023
*******************************************************************************/

if(LastMpPos = 0)THEN
    rOutput                         := TO_REAL((((rInput - rOffset) / rGain) * -1) - 0.15);
ELSE
    rOutput                         := (TO_REAL((((rInput - rOffset) / rGain) * -1) - 0.15) + LastMpPos) / 2;
    END_IF;

LastMpPos                           := rOutput;

```

Input

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
rInput	LREAL	Input		Z - position data from camera								

Output

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
rOutput	REAL	Output		Scaled Z -pos from camera								

Local

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
rGain	REAL	Local		DownScale factor used to scale Z pos	REAL#14.76666667	Private						
rOffset	REAL	Local		Offsett	REAL#-4.43	Private						
LastValues	LReal_0_10	Local		Local var for calcualtions		Private						
LastMpPos	REAL	Local		Local var for calcualtions		Private						

Default

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
rOut	REAL	External										

```
/*****************************************************************************
Code information:
Datatype for camera data

Author: Sander V. Andersen and Lars Askild S. Aarvik
Date: 08.05.2023
*****************************************************************************/

TYPE
    fromCamera                  : STRUCT
    rX                          : REAL;     // X-vector Aruco
    rY                          : REAL;     // Y-vector Aruco
    rZ                          : REAL;     // Z-vector Aruco
    END_STRUCT;

    STRING_10                   : STRING[10];
    BYTE_0_255                  : ARRAY[0..255] OF BYTE;
    arrString                   : ARRAY[0..99] OF STRING_10;
    LReal_0_10                  : ARRAY[0..10] of LREAL;

END_TYPE
```

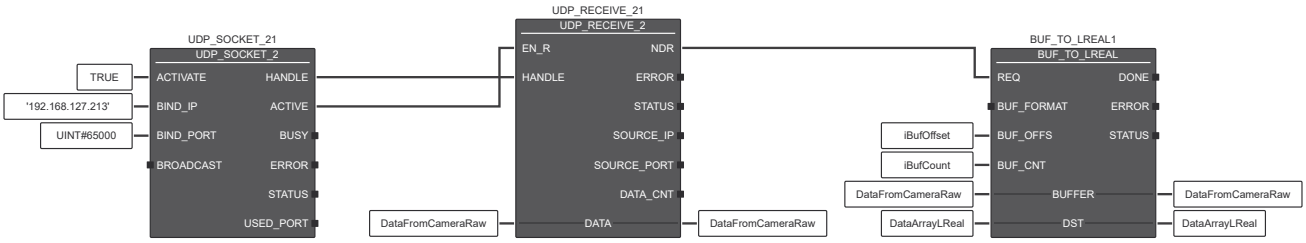
Local

Name	Type	Usage	Translate	Comment	Init	Retain	Constant	OPC	HMI	Proficloud	I/Q
DataFromCameraRaw	BYTE_0_255	Local									

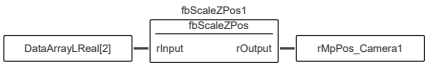
Default

Name	Type	Usage	Translate	Comment	Init	Retain	Constant	OPC	HMI	Proficloud	I/Q
UDP_SOCKET_21	UDP_SOCKET_2	Local									
UDP_RECEIVE_21	UDP_RECEIVE_2	Local									
BUF_TO_REAL1	BUF_TO_REAL	Local									
DataArray	REAL_0_10	Local									
iBufOffset	INT	Local			INT#12						
iBufCount	INT	Local			INT#24						
BUF_TO_LREAL1	BUF_TO_LREAL	Local									
DataArrayLReal	LReal_0_10	Local									
fbScaleZPos1	fbScaleZPos	Local									
xREQTest	BOOL	Local									
rMpPos_Camera1	REAL	External		Mp position from camera							

Receive UDP messages from stereocamera



Scale Aruco distance from cam…



```
/*******************************************************************************
Code information:
Datatype for cylinder data

Author: Sander V. Andersen and Lars Askild S. Aarvik
Date: 20.02.2023
*******************************************************************************/

TYPE

    // Data to cylinder
    sToCylinder :   STRUCT
    xEnableRelay        : BOOL;
    rSpeedCMD           : REAL;
    xResetPos           : BOOL;
    END_STRUCT

    // Data from cylinder
    sFromCylinder :   STRUCT
    rSpeedSP            : REAL;
    rPosSP              : REAL;
    rPV                 : REAL;
    dwPulses            : DWORD;
    rRCV                : REAL;
    END_STRUCT

END_TYPE
```

```
1  /*****************************************************************************
2  Code information:
3  General usefull datatypes
4
5  Author: Sander V. Andersen and Lars Askild S. Aarvik
6  Date: 20.02.2023
7  *****************************************************************************/
8
9  TYPE
10 REAL_0_10                    :   ARRAY[0..10]OF REAL;
11 BYTE_0_36                    :   ARRAY[0..36]OF BYTE;
12 END_TYPE
```

```
/*******************************************************************************
Code information:
Datatype for HMI data

Author: Sander V. Andersen and Lars Askild S. Aarvik
Date: 12.02.2023
*******************************************************************************/

TYPE

    // Operation mode
    sMode :   STRUCT
    iMode               :   INT;
    xManual             :   BOOL;
    xSimulated          :   BOOL;
    xLogFile            :   BOOL;
    END_STRUCT

    // Command
    sCMD :   STRUCT
    iCMD                :   INT;
    xStart              :   BOOL;
    xStopp              :   BOOL;
    xReset              :   BOOL;
    END_STRUCT

    // From UI
    sFromUI :   STRUCT
    iComCount           :   WORD;   // Communication counter
    sOpMode             :   sMode;  // Operation mode
    sCommand            :   sCMD;   // Command
    rAmplitude          :   REAL;   // Amplitude [m]
    tPeriod             :   TIME;   // Periode [s]
    rPhaseDiff          :   REAL;   // Phase difference [deg]
    rHeavePos           :   REAL;   // Heave position for manual mode
    rRollAngle          :   REAL;   // Roll angle for manual mode
    rResetLength        :   REAL;   // Winch reset length
    xResetWinch         :   BOOL;   // Winch reset command
    rC1HeavePosLogF     :   ARRAY[0..16355] OF REAL;    // Cylinder 1 logfile position setpoint array
    rC2_C3HeavePosLogF  :   ARRAY[0..16355] OF REAL;    // Cylinder 2, 3 logfile position setpoint array
    rC1_Velocity        :   ARRAY[0..16355] OF REAL;    // Cylinder 1 logfile velocity setpoint array
    rC2_C3_Velocity     :   ARRAY[0..16355] OF REAL;    // Cylinder 2, 3 logfile velocity setpoint array

    END_STRUCT

    // To UI
    sToUI   :   STRUCT
    rCyl1SP_Pos         :   REAL;    // Position setpoint cylinder 1 [m]
    rCyl2SP_Pos         :   REAL;    // Position setpoint cylinder 2 [m]
    rCyl3SP_Pos         :   REAL;    // Position setpoint cylinder 3 [m]
    rCyl1PV_Pos         :   REAL;    // Position cylinder 1 [m]
    rCyl2PV_Pos         :   REAL;    // Position cylinder 2 [m]
    rCyl3PV_Pos         :   REAL;    // Position cylinder 3 [m]
    rCyl1_Speed         :   REAL;    // Velocity cylinder 1 [m/s]
    rCyl2_Speed         :   REAL;    // Velocity cylinder 2 [m/s]
    rCyl3_Speed         :   REAL;    // Velocity cylinder 3 [m/s]
    rWinch_Pos          :   REAL;    // Wire length winch [m]
    rWinch_Speed        :   REAL;    // Velocity winch [RPM]
    xRunningStatus      :   BOOL;    // TRUE = Running, FALSE = Not Running.
    xEM_STOP            :   BOOL;    // Emergency stop active
    rAHC_CenterPos      :   REAL;    // AHC center position
    wWinchOpMode        :   WORD;    // Operationmode winch
    END_STRUCT

END_TYPE
```

Inputs

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
rawData	MB_TCP_ARR_W_0_65535	Input		Rawdata from HMI								
xSTOP_EM	BOOL	Input		Emergency stop input from HMI								

Outputs

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
Data	sFromUI	Output										

Local

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
rPlattformCC	REAL	Local		Center-center between cylinder front/rear in meter	REAL# 0.425	Private						
xConection	BOOL	Local		True if conection OK, False if not		Private						
PI	REAL	Local		Local var for PI	REAL# 3.141592653 59	Private						
j	DINT	Local		Used for for-loop		Private						
i	DINT	Local		Used for for-loop		Private						
emtyWord	WORD	Local				Private						
rTEST	REAL	Local				Private						
dwTest	DWORD	Local				Private						
iCounterCom	INT	Local		Communication counter		Private						
iLastComCounter	WORD	Local		Used variable under communication check		Private						

External

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
rC1PosSP_Manual	REAL	External		External cylinder 1 position setpoint in manual mode								
rC2_C3PosSP_Manual	REAL	External		External cylinder 2 and 3 position setpoint in manual mode								

Components / HMI / fbFromHMI / Code

```
/*******************************************************************************
Code information:
Function block that maps data from user interface to PLC.


Author: Sander V. Andersen and Lars Askild S. Aarvik
Date: 05.03.2023
*******************************************************************************/


// Mapping from UI
// Checking com counter
if(iCounterCom > 150)THEN
    iCounterCom             :=  0;
    if(iLastComCounter = Data.iComCount)THEN
        xConection          :=  FALSE;
        ELSE
        xConection          :=  TRUE;
        END_IF;
    iLastComCounter     :=  Data.iComCount;

    ELSE
    iCounterCom             :=  iCounterCom + 1;
END_IF;

 Data.iComCount              := rawData[100];

// Checking safety functions
IF(NOT(xSTOP_EM) OR NOT(xConection))THEN

Data.sOpMode.iMode              := 0;
Data.sCommand.iCMD              := 0;
Data.sOpMode.xManual        :=  TRUE;
Data.sOpMode.xSimulated     :=  FALSE;
Data.sOpMode.xLogFile       :=  FALSE;
Data.sCommand.xStopp        :=  TRUE;
Data.sCommand.xStart        :=  FALSE;
Data.sCommand.xReset        :=  FALSE;

ELSE

If(rawData[101] = 0)THEN
    Data.sOpMode.xManual        :=  TRUE;
    Data.sOpMode.xSimulated     :=  FALSE;
    Data.sOpMode.xLogFile       :=  FALSE;
    ELSIF (rawData[101] = 1)THEN
    Data.sOpMode.xSimulated     :=  TRUE;
    Data.sOpMode.xManual        :=  FALSE;
    Data.sOpMode.xLogFile       :=  FALSE;
    ELSIF (rawData[101] = 2)THEN
    Data.sOpMode.xLogFile       :=  TRUE;
    Data.sOpMode.xManual        :=  FALSE;
    Data.sOpMode.xSimulated     :=  FALSE;
    END_IF;

IF(rawData[102] = 0)THEN
    Data.sCommand.xStopp        :=  TRUE;
    Data.sCommand.xStart        :=  FALSE;
    Data.sCommand.xReset        :=  FALSE;
    ELSIF (rawData[102] = 1)THEN
    Data.sCommand.xStart        :=  TRUE;
    Data.sCommand.xStopp        :=  FALSE;
    Data.sCommand.xReset        :=  FALSE;
    ELSIF (rawData[102] = 2)THEN
    Data.sCommand.xReset        :=  TRUE;
    Data.sCommand.xStopp        :=  FALSE;
    Data.sCommand.xStart        :=  FALSE;
    END_IF;

// Operation mode and command mapping
Data.sOpMode.iMode              :=  TO_INT(rawData[101]);
Data.sCommand.iCMD              :=  TO_INT(rawData[102]);

// Simulated Wave commands
Data.rAmplitude             :=  fnWordsToReal(FALSE,FALSE,rawData[103], rawData[104])/1000.0;
Data.tPeriod                :=  TO_TIME(TO_UINT (fnWordsToReal(FALSE,FALSE,rawData[105],rawData[106])));
Data.rPhaseDiff             :=  fnWordsToReal(FALSE,FALSE,rawData[107],rawData[108]);

// Manual mode
Data.rHeavePos              :=  fnWordsToReal(FALSE,FALSE, rawData[109],rawData[110])/1000.0;
Data.rRollAngle             :=  fnWordsToReal(FALSE,FALSE, rawData[111],rawData[112]);
```

```
82
83  IF(Data.sOpMode.xManual)THEN
84      // Calculate manual pos.
85      rC1PosSP_Manual         :=  Data.rHeavePos + ((SIN(Data.rRollAngle * (PI/180.0)) *  rPlattformCC))/2;
86      rC2_C3PosSP_Manual      :=  Data.rHeavePos - ((SIN(Data.rRollAngle * (PI/180.0)) *  rPlattformCC))/2;
87      END_IF;
88
89
90  // Reset winch pos data
91  Data.rResetLength           := fnWordsToReal(FALSE,FALSE, rawData[113],rawData[114])/1000.0;
92  IF(rawData[115] = 0)THEN
93      Data.xResetWinch        := FALSE;
94  ELSE
95      Data.xResetWinch        := TRUE;
96      END_IF;
97
98   // Logfile data
99   IF(Data.sOpMode.xLogFile)THEN
100      j  := 0;
101
102      FOR i := 116 TO 65532 BY 4 DO
103          data.rC1HeavePosLogF[j]       := TO_REAL(TO_INT(rawData[i]))/100000;
104          Data.rC1_Velocity[j]          := TO_REAL(TO_INT(rawData[i+1]))/100000;
105          Data.rC2_C3HeavePosLogF[j]    := TO_REAL(TO_INT(rawData[i+2]))/100000;
106          Data.rC2_C3_Velocity[j]       := TO_REAL(TO_INT(rawData[i+3]))/100000;
107
108          j                             := j+1;
109          END_FOR;
110
111      END_IF;
112
113 END_IF;
```

Inputs

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
DataFromPLC	sToUI	Input		Struct - data to HMI from PLC								

Outputs

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
Rdata	MB_TCP_ARR_W_0_65535	Output		Mapped data to HMI								

Default

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
DataFromHMI	sFromUI	External		Struct for all data from HMI								
arrHoldingRegisters	MB_TCP_ARR_W_0_65535	External										
fbRealToWords1	fbRealToWords	Local				Private						
DataToHMI	sToUI	External		Struct for all data to HMI								
xEM_STOP	BOOL	External		Emergency stop								

Components / HMI / fbToHMI / Code

```
1  /*************************************************************************
2  Code information:
3  Function block that maps data from PLC to user interface.
4
5
6  Author: Sander V. Andersen and Lars Askild S. Aarvik
7  Date: 03.03.2023
8  *************************************************************************/
9
10
11 // Mapping data to UI
12 fbRealToWords1(xSwapBytes := FALSE, xSwapWords := FALSE, rIn := DataFromPLC.rCyl1SP_Pos*1000.0, w0 => Rdata[0], w1 => Rdata
   [1]);
13 fbRealToWords1(xSwapBytes := FALSE, xSwapWords := FALSE, rIn := DataFromPLC.rCyl2SP_Pos*1000.0, w0 => Rdata[2], w1 => Rdata
   [3]);
14 fbRealToWords1(xSwapBytes := FALSE, xSwapWords := FALSE, rIn := DataFromPLC.rCyl3SP_Pos*1000.0, w0 => Rdata[4], w1 => Rdata
   [5]);
15 fbRealToWords1(xSwapBytes := FALSE, xSwapWords := FALSE, rIn := DataFromPLC.rCyl1PV_Pos*1000.0, w0 => Rdata[6], w1 => Rdata
   [7]);
16 fbRealToWords1(xSwapBytes := FALSE, xSwapWords := FALSE, rIn := DataFromPLC.rCyl2PV_Pos*1000.0, w0 => Rdata[8], w1 => Rdata
   [9]);
17 fbRealToWords1(xSwapBytes := FALSE, xSwapWords := FALSE, rIn := DataFromPLC.rCyl3PV_Pos*1000.0, w0 => Rdata[10], w1 =>
   Rdata[11]);
18 Rdata[12].X0              := DataToHMI.xRunningStatus;
19 Rdata[13].X0              := DataToHMI.xEM_STOP;
20 fbRealToWords1(xSwapBytes := FALSE, xSwapWords := FALSE, rIn := DataFromPLC.rCyl1_Speed *100000.0, w0 => Rdata[14], w1 =>
   Rdata[15]);
21 fbRealToWords1(xSwapBytes := FALSE, xSwapWords := FALSE, rIn := DataFromPLC.rCyl2_Speed *100000.0, w0 => Rdata[16], w1 =>
   Rdata[17]);
22 fbRealToWords1(xSwapBytes := FALSE, xSwapWords := FALSE, rIn := DataFromPLC.rCyl3_Speed *100000.0, w0 => Rdata[18], w1 =>
   Rdata[19]);
23 fbRealToWords1(xSwapBytes := FALSE, xSwapWords := FALSE, rIn := DataFromPLC.rWinch_Pos *1000.0, w0 => Rdata[20], w1 =>
   Rdata[21]);
24 fbRealToWords1(xSwapBytes := FALSE, xSwapWords := FALSE, rIn := DataFromPLC.rWinch_Speed *100.0, w0 => Rdata[22], w1 =>
   Rdata[23]);
25 fbRealToWords1(xSwapBytes := FALSE, xSwapWords := FALSE, rIn := DataFromPLC.rAHC_CenterPos * 1000.0, w0 => Rdata[24], w1 =>
   Rdata[25]);
26 Rdata[26]                 := DataFromPLC.wWinchOpMode;
```

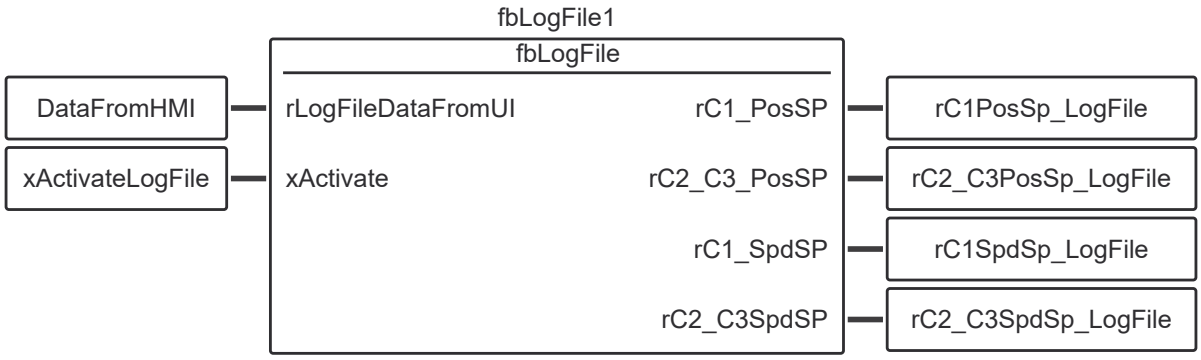
External

Name	Type	Usage	Translate	Comment	Init	Retain	Constant	OPC	HMI	Proficloud	I/Q
rC1PosSp_LogFile	REAL	External		PositionSetpoint to cylinder 1 calculated from logfile							
rC1SpdSp_LogFile	REAL	External		SpeedSetpoint to cylinder 1 calculated from logfile							
rC2_C3PosSp_LogFile	REAL	External		PositionSetpoint to cylinder 2 and 3 calculated from logfile							
rC2_C3SpdSp_LogFile	REAL	External		SpeedSetpoint to cylinder 2 and 3 calculated from logfile							

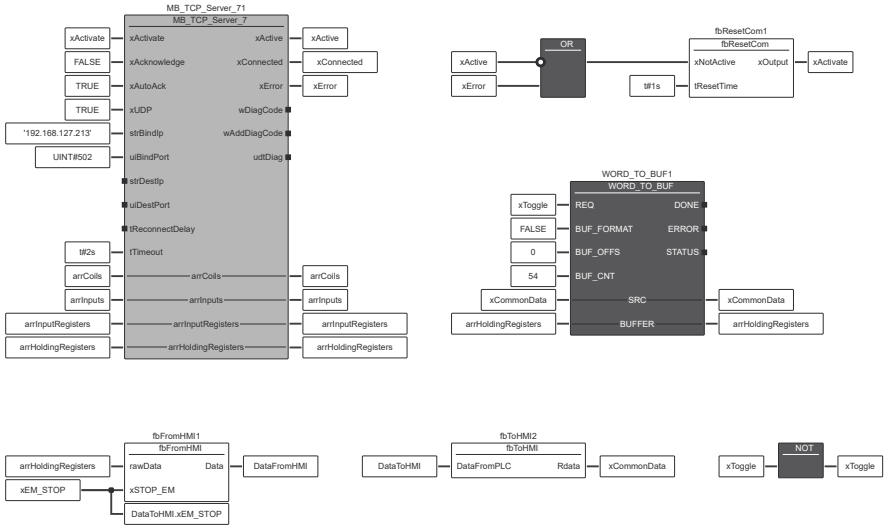
Default

Name	Type	Usage	Translate	Comment	Init	Retain	Constant	OPC	HMI	Proficloud	I/Q
DataFromHMI	sFromUI	External		Struct for all data from HMI							
xActivateLogFile	BOOL	External		Bool used to activate logfile mode							
fbLogFile1	fbLogFile	Local									

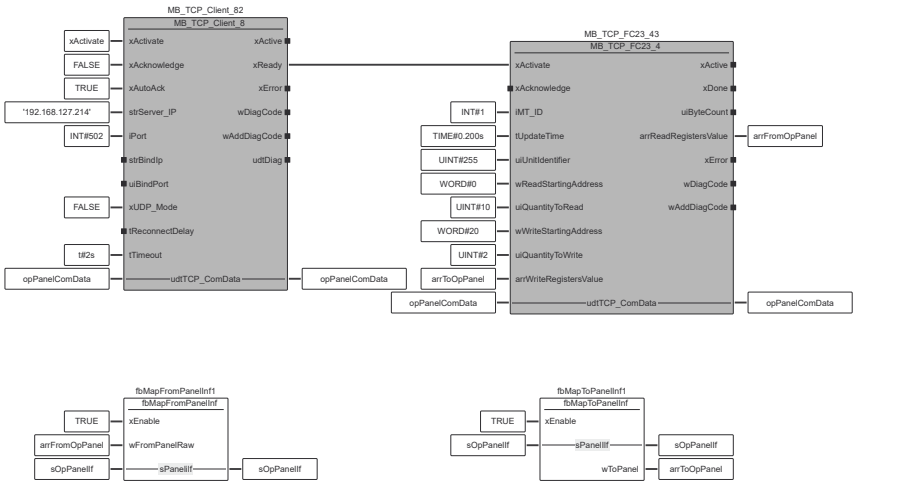
Logfile plattform control



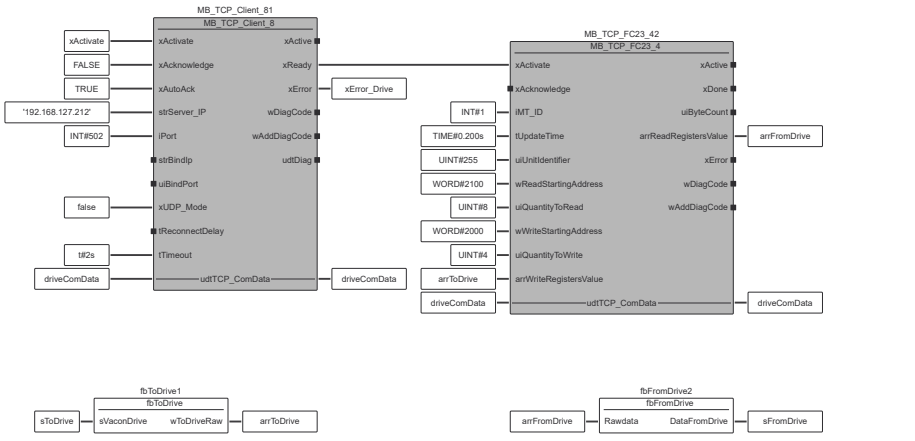
MODBUS UDP communication with User Interface



MODBUS TCP communication with Operation Panel



MODBUS TCP communication with VaconDrive



Local

Name	Type	Usage	Translate	Comment	Init	Retain	Constant	OPC	HMI	Proficloud	I/Q
xActivate	BOOL	Local									
xActive	BOOL	Local									
xConnected	BOOL	Local									
xError	BOOL	Local									
xToggle	BOOL	Local									
xCommonData	MB_TCP_ARR_W_0_65535	Local									
fbResetCom1	fbResetCom	Local									
fbFromHMI1	fbFromHMI	Local									
fbToHMI2	fbToHMI	Local									
MB_TCP_Server_71	MB_TCP_Server_7	Local									
WORD_TO_BUF1	WORD_TO_BUF	Local									
xError_Drive	BOOL	Local									
driveComData	MB_TCP_UDT_COMMUNICATION	Local									
opPanelComData	MB_TCP_UDT_COMMUNICATION	Local									
arrFromDrive	MB_TCP_ARR_W_1_125	Local									
arrToDrive	MB_TCP_ARR_W_1_125	Local									
arrFromOpPanel	MB_TCP_ARR_W_1_125	Local									
arrToOpPanel	MB_TCP_ARR_W_1_125	Local									

External

Name	Type	Usage	Translate	Comment	Init	Retain	Constant	OPC	HMI	Proficloud	I/Q
sOpPanelIf	sPanelInterface	External		Struct with data to/from op panel							
sToDrive	ToVaconDriveEng	External		Struct with data to drive							
sFromDrive	FromVaconDriveEng	External		Struct with data from drive							
DataToHMI	sToUI	External		Struct for all data to HMI							
DataFromHMI	sFromUI	External		Struct for all data from HMI							
arrHoldingRegisters	MB_TCP_ARR_W_0_65535	External									
arrCoils	MB_TCP_ARR_X_0_65535	External									
arrInputs	MB_TCP_ARR_X_0_65535	External									
arrInputRegisters	MB_TCP_ARR_W_0_65535	External									
xEM_STOP	BOOL	External		Emergency stop							

Default

Name	Type	Usage	Translate	Comment	Init	Retain	Constant	OPC	HMI	Proficloud	I/Q
MB_TCP_Client_81	MB_TCP_Client_8	Local									
MB_TCP_FC23_42	MB_TCP_FC23_4	Local									
MB_TCP_Client_82	MB_TCP_Client_8	Local									
MB_TCP_FC23_43	MB_TCP_FC23_4	Local									
fbMapFromPanelInf1	fbMapFromPanelInf	Local									
fbMapToPanelInf1	fbMapToPanelInf	Local									
fbFromDrive2	fbFromDrive	Local									
fbToDrive1	fbToDrive	Local									

Components / Panelinterface / fbMapFromPanelInf / Code

```
/*************************************************************************
Code information:
Function block that maps data from Operator panel to PLC

Author: Sander V. Andersen and Lars Askild S. Aarvik
Date: 11.04.2023
*************************************************************************/

IF xEnable THEN
    sPanelilf.bCounter          :=  wFromPanelRaw[1].B0;
    sPanelilf.xEmStop           :=  wFromPanelRaw[2].X15;
    sPanelilf.xJoyErr           :=  wFromPanelRaw[2].X14;
    sPanelilf.wButtons          :=  wFromPanelRaw[2];
    sPanelilf.wJoySw            :=  wFromPanelRaw[3];
    sPanelilf.rJoy1             :=  TO_REAL(TO_UINT(wFromPanelRaw[9])) / 10.0;
    sPanelilf.rJoy2             :=  TO_REAL(TO_UINT(wFromPanelRaw[10])) / 10.0;
    sPanelilf.rPot1             :=  TO_REAL(TO_UINT(wFromPanelRaw[6])) / 10.0;
    sPanelilf.rPot2             :=  TO_REAL(TO_UINT(wFromPanelRaw[7])) / 10.0;
    sPanelilf.iTemperature      :=  WORD_TO_INT(wFromPanelRaw[8]);
    sPanelilf.wJoy1Raw          :=  wFromPanelRaw[9];
    sPanelilf.wJoy2Raw          :=  wFromPanelRaw[10];
    sPanelilf.wPot1Raw          :=  wFromPanelRaw[6];
    sPanelilf.wPot2Raw          :=  wFromPanelRaw[7];


    sPanelilf.xReset            :=  wFromPanelRaw[2].X6;
    sPanelilf.xStandBy          :=  wFromPanelRaw[2].X7;
    sPanelilf.xAHC              :=  wFromPanelRaw[2].X8;
    sPanelilf.xInControl        :=  wFromPanelRaw[2].X10;
    sPanelilf.xManual           :=  wFromPanelRaw[2].X11;

END_IF;
```

Inputs

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
xEnable	BOOL	Input		Enable block								
wFromPanelRaw	MB_TCP_ARR_W_1_125	Input		Modbus array from panel								
sPanelIf	sPanelInterface	InOut		Panel interface structure								

Components / Panelinterface / fbMapToPanelInf / Code

```
/*************************************************************************
Code information:
Function block that maps data from PLC to Operator panel.

Author: Sander V. Andersen and Lars Askild S. Aarvik
Date: 11.04.2023
*************************************************************************/
IF xEnable THEN

    sPanelIlf.wBtnLightA.X6       :=    sPanelIlf.xResetLight;
    sPanelIlf.wBtnLightA.X7       :=    sPanelIlf.xStandbyLight;
    sPanelIlf.wBtnLightA.X8       :=    sPanelIlf.xAHCLight;
    sPanelIlf.wBtnLightA.X10      :=    sPanelIlf.xControlLight;
    sPanelIlf.wBtnLightA.X11      :=    sPanelIlf.xManualLight;

    wToPanel[1]     := sPanelIlf.wBtnLightA;
    wToPanel[2]     := sPanelIlf.wBtnLightB;
    wToPanel[1].X15 := sPanelIlf.xBuzzer;


END_IF;
```

Input

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
xEnable	BOOL	Input		Enable function block								
sPanelIIf	sPanelInterface	InOut		Struct data to Operator panel								

Output

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
wToPanel	MB_TCP_ARR_W_1_125	Output		Word array to panel								

```
/*******************************************************************************
Code information:
Datatype for panelinterface data

Author: Sander V. Andersen and Lars Askild S. Aarvik
Date: 28.03.2023
*******************************************************************************/

TYPE

    WORD_0_9                    :   ARRAY[0..9] OF WORD;
    WORD_0_1                    :   ARRAY[0..1] OF WORD;

    // Struct for interface
    sPanelInterface             :   STRUCT
        xConnected              :   BOOL;    (* True if connected and bCounter gets updated *)
        bCounter                :   BYTE;    (* Counter increments every 50ms by panel *)
        xEmStop                 :   BOOL;    (* True if emergency stop is pressed *)
        xJoyErr                 :   BOOL;    (* True if joystick switch error *)
        wJoySw                  :   WORD;    (* Joystick switches: X0=Joy1 SwNeg, X1=Joy1 SwPos, X2=Joy2 SwNeg, X3 = Joy2
        SwPos *)
        wButtons                :   WORD;    (* Button states: X0-X11 *)
        rJoy1                   :   REAL;    (* Joystick +/-100% *)
        rJoy2                   :   REAL;    (* Joystick +/-100% *)
        wJoy1Raw                :   WORD;    (* Joystick 0-1000 without direction *)
        wJoy2Raw                :   WORD;    (* Joystick 0-1000 without direction *)
        rPot1                   :   REAL;    (* Potmeter 0-100%*)
        rPot2                   :   REAL;    (* Potmeter 0-100%*)
        wPot1Raw                :   WORD;    (* Potmeter 0-1000 *)
        wPot2Raw                :   WORD;    (* Potmeter 0-1000 *)
        iTemperature            :   INT;     (* Panel termperature *C *)
        wBtnLightA              :   WORD;    (* Button lights channel A, X0-X11 *)
        wBtnLightB              :   WORD;    (* Button lights channel B, X0-X11 *)
        xBuzzer                 :   BOOL;    (* Buzzer output ON *)
        xReset                  :   BOOL;
        xStandBy                :   BOOL;
        xAHC                    :   BOOL;
        xInControl              :   BOOL;
        xManual                 :   BOOL;
        xResetLight             :   BOOL;
        xStandbyLight           :   BOOL;
        xAHCLight               :   BOOL;
        xControlLight           :   BOOL;
        xManualLight            :   BOOL;
    END_STRUCT;
END_TYPE
```

```
/*******************************************************************************
Code information:
Datatype for PID data

Author: Sander V. Andersen and Lars Askild S. Aarvik
Date: 05.02.2023
*******************************************************************************/

TYPE

    // Operation mode
    sPID :   STRUCT
    iMode               :   INT;    // PID mode
    rPosSP              :   REAL;   // Position setpoint
    END_STRUCT

END_TYPE
```

Input

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
dwRawDataEncoder	DWORD	Input		Raw encoder value								

Output

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
rOutValueInMeter	REAL	Output		OutValue in meter 0.1 = 10 cm								

Local

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
rPulses	REAL	Local		Pulses encoder has	REAL#800.0	Private						
rDistance	REAL	Local		Driving distance for cylinder in meter	REAL#0.30	Private						
udiRawDataUDINT	UDINT	Local				Private						

Components / Plattform / fbEncConverter / Code

```
/*************************************************************************
Code information:
Function block that scales pulses given from encoder to position in meter.

Author: Sander V. Andersen and Lars Askild S. Aarvik
Date: 14.02.2023
*************************************************************************/


udiRawDataUDINT            :=      TO_UDINT(dwRawDataEncoder); // For calulation

rOutValueInMeter         :=      TO_REAL(udiRawDataUDINT) * (rDistance/rPulses); // Outvalue - position[m]
```

INPUT

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
rLogFileDataFromUI	sFromUI	Input		Struct with logfila data from HMI								
xActivate	BOOL	Input		Actiivates function block								

OUTPUT

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
rC1_PosSP	REAL	Output		Output interpolated position setpoint Cylinder 1								
rC2_C3_PosSP	REAL	Output		Output interpolated position setpoint Cylinder 2 and 3								
rC1_SpdSP	REAL	Output		Output interpolated velocity setpoint Cylinder 1								
rC2_C3SpdSP	REAL	Output		Output interpolated velocity setpoint Cylinder 2 and 3								

LOCAL

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
iArrayCount	INT	Local		Local array used to interpolate data		Private						
iLoopCount	INT	Local		Loop counter used in for loop - which interpolate data		Private						

```
/*************************************************************************
Code information:
Function block that interpolates wavefile position setpoint & speed setpoint.
Returns Position setpoint and speed point.
Author: Sander V. Andersen and Lars Askild S. Aarvik
Date: 17.03.2023
*************************************************************************/


IF(xActivate)THEN
IF(rLogFileDataFromUI.sOpMode.xLogFile)THEN

    IF(rLogFileDataFromUI.sCommand.xStart)THEN




        // Interpolation
        If(iLoopCount >= 10)THEN
            iLoopCount                  := 0;
            iArrayCount                 := iArrayCount + 1;
            rC1_PosSP                   := rLogFileDataFromUI.rC1HeavePosLogF[iArrayCount] + 0.15;
            rC1_SpdSP                   := rLogFileDataFromUI.rC1_Velocity[iArrayCount];
            rC2_C3_PosSP                := rLogFileDataFromUI.rC2_C3HeavePosLogF[iArrayCount] + 0.15;
            rC2_C3SpdSP                 := rLogFileDataFromUI.rC2_C3_Velocity[iArrayCount];

            ELSE


            // Interpolation
            rC1_PosSP                   := (
                                            ((((rLogFileDataFromUI.rC1HeavePosLogF[iArrayCount+1] + 0.15) -
                                            (rLogFileDataFromUI.rC1HeavePosLogF[iArrayCount] + 0.15)
                                            ) / 10) * iLoopCount) + rLogFileDataFromUI.rC1HeavePosLogF[iArrayCount] + 0.15
                                            );


            rC1_SpdSP                    := (
                                            ((((rLogFileDataFromUI.rC1_Velocity[iArrayCount+1]) -
                                            (rLogFileDataFromUI.rC1_Velocity[iArrayCount])
                                            ) / 10) * iLoopCount) + rLogFileDataFromUI.rC1_Velocity[iArrayCount]
                                            );


            rC2_C3_PosSP                    := (
                                            ((((rLogFileDataFromUI.rC2_C3HeavePosLogF[iArrayCount+1] + 0.15) -
                                            (rLogFileDataFromUI.rC2_C3HeavePosLogF[iArrayCount] + 0.15)
                                            ) / 10) * iLoopCount) + rLogFileDataFromUI.rC2_C3HeavePosLogF[iArrayCount] +
                                            0.15
                                            );


            rC2_C3SpdSP                    := (
                                            ((((rLogFileDataFromUI.rC2_C3_Velocity[iArrayCount+1]) -
                                            (rLogFileDataFromUI.rC2_C3_Velocity[iArrayCount])
                                            ) / 10) * iLoopCount) + rLogFileDataFromUI.rC2_C3_Velocity[iArrayCount]
                                            );



            iLoopCount                  := iLoopCount + 1;
            END_IF;



        if(iArrayCount >= 16355)THEN
            iArrayCount    := 0;
            END_IF;

        END_IF;

    END_IF;
    ELSE
        iArrayCount       :=  0;
    END_IF;
```

Inputs

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
sFromCylinder1	sFromCylinder	Input		Struct with data from cylinder 1								
sFromCylinder2	sFromCylinder	Input		Struct with data from cylinder 2								
sFromCylinder3	sFromCylinder	Input		Struct with data from cylinder 3								
sDataFromHMI	sFromUI	Input		Struct with datacommands from HMI								

Outputs

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
sToCylinder1	sToCylinder	Output		Struct with data to cylinder 1								
sToCylinder2	sToCylinder	Output		Struct with data to cylinder 2								
sToCylinder3	sToCylinder	Output		Struct with data to cylinder 3								
xActivateSinGen	BOOL	Output		Activate Sinus wave generator								
xRunning	BOOL	Output		TRUE if plattfrom is running, FALSE if not								
xActivateLogFile	BOOL	Output		Bool that activates logfile mode								
sPID1	sPID	Output		Struct with PID data to cylinder 1								
sPID2	sPID	Output		Struct with PID data to cylinder 2								
sPID3	sPID	Output		Struct with PID data to cylinder 3								

Local

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
xResetFinished	BOOL	Local				Private						
iWaitCounter	INT	Local				Private						
xStartPosReached	BOOL	Local				Private						
rRCVC1	REAL	Local				Private						
rRCVC2	REAL	Local				Private						
rRCVC3	REAL	Local				Private						
rTempTime	REAL	Local				Private						
xC1RunningOut	BOOL	Local			TRUE	Private						
xC1RunningIN	BOOL	Local			TRUE	Private						
xC2_C3RunningOut	BOOL	Local			TRUE	Private						
xC2_C3RunningIN	BOOL	Local			TRUE	Private						
xC1PosReached	BOOL	Local				Private						
xC2PosReached	BOOL	Local				Private						
xC3PosReached	BOOL	Local				Private						
rTempSPD_C1	REAL	Local				Private						
rTempSPD_C2	REAL	Local				Private						
rTempSPD_C3	REAL	Local				Private						
uLastPulse1	UDINT	Local			UDINT# 1000	Private						
uLastPulse2	UDINT	Local				Private						
uLastPulse3	UDINT	Local				Private						
rStartPosition	REAL	Local			REAL# 0.15	Private						
xTempTimeCalculated	BOOL	Local				Private						
rDefaultSpd	REAL	Local			REAL# 0.025	Private						
xtravelCalculated	BOOL	Local				Private						
xC2C3_LongestWay	BOOL	Local				Private						
rDeadBand	REAL	Local			REAL# 0.002	Private						
iResetCounter	INT	Local				Private						

```
/*************************************************************************
Code information:
Function block for operating the plattform in different operation modes (Manual-control, Simulated and Logfile) with
different commands(Start, stopp, reset).

Author: Sander V. Andersen and Lars Askild S. Aarvik
Date: 13.04.2023
*************************************************************************/


CASE sDataFromHMI.sCommand.iCMD OF
    //
    // Stop
     //
    0:
    xC1RunningIN                                := TRUE;
    xC1RunningOut                               := TRUE;
    xC2_C3RunningIN                             := TRUE;
    xC2_C3RunningOut                            := TRUE;
    xTempTimeCalculated                         := FALSE;
    xtravelCalculated                           := FALSE;
    xC1PosReached                               := FALSE;
    xC2PosReached                               := FALSE;
    xC3PosReached                               := FALSE;

    xRunning                                    := FALSE;
    xResetFinished                              := FALSE;
    sToCylinder1.xResetPos                      := FALSE;
    sToCylinder2.xResetPos                      := FALSE;
    sToCylinder3.xResetPos                      := FALSE;
    sTOCylinder1.xEnableRelay                   := FALSE;
    sTOCylinder2.xEnableRelay                   := FALSE;
    sTOCylinder3.xEnableRelay                   := FALSE;
    xActivateSinGen                             := FALSE;
    xActivateLogFile                            := FALSE;
    xStartPosReached                            := FALSE;
    sTOCylinder1.rSpeedCMD                      := 0;
    sTOCylinder2.rSpeedCMD                      := 0;
    sTOCylinder3.rSpeedCMD                      := 0;
    iWaitCounter                                := 0;
    //
    // Start
    //
    1:
    CASE sDataFromHMI.sOpMode.iMode OF
        //
        // Manual
        //
        0:

        if(iWaitCounter < 2) THEN
            iWaitCounter                        := iWaitCounter + 1;
        ELSE


        // Check wich cylinder has the longest way to travel
        IF((ABS(sFromCylinder1.rPosSP - sFromCylinder1.rPV) <= ABS(sFromCylinder2.rPosSP - sFromCylinder2.rPV)) AND NOT
        (xtravelCalculated))THEN
            xC2C3_LongestWay                    := TRUE;
            xtravelCalculated                   := TRUE;
        ELSIF((ABS(sFromCylinder1.rPosSP - sFromCylinder1.rPV) > ABS(sFromCylinder2.rPosSP - sFromCylinder2.rPV)) AND NOT
        (xtravelCalculated))THEN
            xC2C3_LongestWay                    := FALSE;
            xtravelCalculated                   := TRUE;
            END_IF;


        // Cylinder2 and cylinder3 longest travel way
        IF(xC2C3_LongestWay)THEN

            //Calculating cylinder2 and 3 traveltime
            IF(NOT(xTempTimeCalculated))THEN
                rTempTime                       := ABS(sFromCylinder2.rPV-sFromCylinder2.rPosSP)/rDefaultSpd;
                IF(rTempTime > 0) THEN
                    // Calculating speed for cylinder1
                    rTempSPD_C1                     := fnFeedForwardC1((sFromCylinder1.rPV - sFromCylinder1.rPosSP)/
                    rTempTime);
                    END_IF;
                xTempTimeCalculated             := TRUE;
                END_IF;

```

```
78
79              // Check if cylinder2 and cylinder3 should run out or in
80              // Cylinder2 should run out
81              If((sFromCylinder2.rPV < (sFromCylinder2.rPosSP - rDeadBand)) AND NOT(xC2PosReached))THEN
82                  sToCylinder2.rSpeedCMD          := fnFeedForwardC2(-rDefaultSpd);
83                  sToCylinder2.xEnableRelay       := TRUE;
84              // Cylinder2 should run in
85              ELSIF((sFromCylinder2.rPV > (sFromCylinder2.rPosSP + rDeadBand)) AND NOT(xC2PosReached))THEN
86                  sToCylinder2.rSpeedCMD          := fnFeedForwardC2(rDefaultSpd);
87                  sToCylinder2.xEnableRelay       := TRUE;
88              // Stop Cylinder2
89              ELSE
90                  sToCylinder2.rSpeedCMD          := 0;
91                  sToCylinder2.xEnableRelay       := FALSE;
92                  xC2PosReached                   := TRUE;
93                  END_IF;
94
95              // Cylinder3 should run out
96              If((sFromCylinder3.rPV < (sFromCylinder3.rPosSP - rDeadBand)) AND NOT(xC3PosReached))THEN
97                  sToCylinder3.rSpeedCMD          := fnFeedForwardC3(-rDefaultSpd);
98                  sToCylinder3.xEnableRelay       := TRUE;
99                  // Cylinder3 should run in
100             ELSIF((sFromCylinder3.rPV > (sFromCylinder3.rPosSP + rDeadBand)) AND NOT(xC3PosReached))THEN
101                 sToCylinder3.rSpeedCMD          := fnFeedForwardC3(rDefaultSpd);
102                 sToCylinder3.xEnableRelay       := TRUE;
103             // Stop Cylinder3
104             ELSE
105                 sToCylinder3.rSpeedCMD          := 0;
106                 sToCylinder3.xEnableRelay       := FALSE;
107                 xC3PosReached                   := TRUE;
108                 END_IF;
109
110             // Running cylinder1 with calculated speed
111             // Cylinder1 should run out
112             If((sFromCylinder1.rPV < (sFromCylinder1.rPosSP - rDeadBand)) AND NOT(xC1PosReached))THEN
113                 sToCylinder1.rSpeedCMD          := rTempSPD_C1;
114                 sToCylinder1.xEnableRelay       := TRUE;
115                 // Cylinder1 should run in
116             ELSIF((sFromCylinder1.rPV > (sFromCylinder1.rPosSP + rDeadBand)) AND NOT(xC1PosReached))THEN
117                 sToCylinder1.rSpeedCMD          := rTempSPD_C1;
118                 sToCylinder1.xEnableRelay       := TRUE;
119             // Stop Cylinder1
120             ELSE
121                 sToCylinder1.rSpeedCMD          := 0;
122                 sToCylinder1.xEnableRelay       := FALSE;
123                 xC1PosReached                   := TRUE;
124                 END_IF;
125
126
127         // Cylinder1 longest travel way
128         ELSE
129
130              //Calculating cylinder1
131             IF(NOT(xTempTimeCalculated))THEN
132                 rTempTime                       := ABS(sFromCylinder1.rPV-sFromCylinder1.rPosSP)/rDefaultSpd;
133                 IF(rTempTime > 0) THEN
134                     // Calculating speed for cylinder2 and cylinder 3
135                     rTempSPD_C2                     := fnFeedForwardC2((sFromCylinder2.rPV - sFromCylinder2.rPosSP)/
                        rTempTime);
136                     rTempSPD_C3                     := fnFeedForwardC3((sFromCylinder3.rPV - sFromCylinder3.rPosSP)/
                        rTempTime);
137                     END_IF;
138                 xTempTimeCalculated             := TRUE;
139                 END_IF;
140
141
142             // Check if cylinder1 should run out or in
143             // Cylinder1 should run out
144             If((sFromCylinder1.rPV < (sFromCylinder1.rPosSP - rDeadBand)) AND NOT(xC1PosReached))THEN
145                 sToCylinder1.rSpeedCMD          := fnFeedForwardC1(-rDefaultSpd);
146                 sToCylinder1.xEnableRelay       := TRUE;
147             // Cylinder1 should run in
148             ELSIF((sFromCylinder1.rPV > (sFromCylinder1.rPosSP + rDeadBand)) AND NOT(xC1PosReached))THEN
149                 sToCylinder1.rSpeedCMD          := fnFeedForwardC1(rDefaultSpd);
150                 sToCylinder1.xEnableRelay       := TRUE;
151             // Stop Cylinder1
152             ELSE
153                 sToCylinder1.rSpeedCMD          := 0;
154                 sToCylinder1.xEnableRelay       := FALSE;
155                 xC1PosReached                   := TRUE;
156                 END_IF;
```

```
157
158                 // Running cylinder2 and cylinder3 with calculated speed
159                 // Cylinder2 should run out
160                 If((sFromCylinder2.rPV < (sFromCylinder2.rPosSP - rDeadBand)) AND NOT(xC2PosReached))THEN
161                     sToCylinder2.rSpeedCMD          := rTempSPD_C2;
162                     sToCylinder2.xEnableRelay       := TRUE;
163                     // Cylinder2 should run in
164                 ELSIF((sFromCylinder2.rPV > (sFromCylinder2.rPosSP + rDeadBand)) AND NOT(xC2PosReached))THEN
165                     sToCylinder2.rSpeedCMD          := rTempSPD_C2;
166                     sToCylinder2.xEnableRelay       := TRUE;
167                 // Stop Cylinder2
168                 ELSE
169                     sToCylinder2.rSpeedCMD          := 0;
170                     sToCylinder2.xEnableRelay       := FALSE;
171                     xC2PosReached                   := TRUE;
172                     END_IF;
173
174                 // Cylinder3 should run out
175                 If((sFromCylinder3.rPV < (sFromCylinder3.rPosSP - rDeadBand)) AND NOT(xC3PosReached))THEN
176                     sToCylinder3.rSpeedCMD          := rTempSPD_C3;
177                     sToCylinder3.xEnableRelay       := TRUE;
178                     // Cylinder3 should run in
179                 ELSIF((sFromCylinder3.rPV > (sFromCylinder3.rPosSP + rDeadBand)) AND NOT(xC3PosReached))THEN
180                     sToCylinder3.rSpeedCMD          := rTempSPD_C3;
181                     sToCylinder3.xEnableRelay       := TRUE;
182                 // Stop Cylinder3
183                 ELSE
184                     sToCylinder3.rSpeedCMD          := 0;
185                     sToCylinder3.xEnableRelay       := FALSE;
186                     xC3PosReached                   := TRUE;
187                     END_IF;
188
189
190
191
192                 END_IF;
193
194
195             // Check if cylinders are moving
196                  If(xC1PosReached AND xC2PosReached AND xC3PosReached)THEN
197                      xRunning                       := FALSE;
198                      ELSE
199                      xRunning                       := TRUE;
200                      END_IF;
201
202
203             END_IF;
204
205
206             //
207             // Simulated
208             //
209             1:
210             // Enabling Enable relays
211             sToCylinder1.xEnableRelay               := TRUE;
212             sToCylinder2.xEnableRelay               := TRUE;
213             sToCylinder3.xEnableRelay               := TRUE;
214
215             // Checking if all cylinders have reached start positon
216             // if they have, the system will start running in simulated mode
217             IF(xC1PosReached AND xC2PosReached AND xC3PosReached)THEN
218                 xActivateSinGen                     := TRUE;
219             // Cylinder1 PID OpMode
220             IF(sFromCylinder1.rPV - sFromCylinder1.rPosSP > 0) THEN
221                 sPID1.iMode                         := 3;
222                 rRCVC1                              :=  sFromCylinder1.rRCV;
223                 ELSE
224                 sPID1.iMode                         :=  2;
225                 rRCVC1                              := sFromCylinder1.rRCV * -1.0;
226                 END_IF;
227              IF(sFromCylinder2.rPV - sFromCylinder2.rPosSP > 0) THEN
228                 sPID2.iMode                         :=  3;
229                 rRCVC2                              :=  sFromCylinder2.rRCV;
230                 ELSE
231                 sPID2.iMode                         :=  2;
232                 rRCVC2                              := sFromCylinder2.rRCV * -1.0;
233                 END_IF;
234               IF(sFromCylinder3.rPV - sFromCylinder3.rPosSP > 0) THEN
235                 sPID3.iMode                         :=  3;
236                 rRCVC3                              :=  sFromCylinder3.rRCV;
237                 ELSE
```

```
238                 sPID3.iMode                         :=  2;
239                 rRCVC3                              := sFromCylinder3.rRCV * -1.0;
240                 END_IF;
241
242               sPID1.rPosSP                          := sFromCylinder1.rPosSP;
243               sPID2.rPosSP                          := sFromCylinder2.rPosSP;
244               sPID3.rPosSP                          := sFromCylinder3.rPosSP;
245               sToCylinder1.rSpeedCMD                := fnFeedForwardC1(sFromCylinder1.rSpeedSP) + rRCVC1;//
246               sToCylinder2.rSpeedCMD                := fnFeedForwardC2(sFromCylinder2.rSpeedSP) + rRCVC2;//
247               sToCylinder3.rSpeedCMD                := fnFeedForwardC3(sFromCylinder3.rSpeedSP) + rRCVC3;//
248             ELSE
249             // Running to start position
250             IF(sFromCylinder1.rPV < rStartPosition AND NOT(xC1PosReached)) THEN
251                 sToCylinder1.rSpeedCMD              := fnFeedForwardC1(-rDefaultSpd);
252
253             ELSE
254                 sToCylinder1.rSpeedCMD              := 0;
255                 xC1PosReached                       := TRUE;
256                 END_IF;
257             IF(sFromCylinder2.rPV < rStartPosition AND NOT(xC2PosReached)) THEN
258                 sToCylinder2.rSpeedCMD              := fnFeedForwardC2(-rDefaultSpd);
259             ELSE
260                 sToCylinder2.rSpeedCMD              := 0;
261                 xC2PosReached                       := TRUE;
262                 END_IF;
263             IF(sFromCylinder3.rPV < rStartPosition AND NOT(xC3PosReached)) THEN
264                 sToCylinder3.rSpeedCMD              := fnFeedForwardC3(-rDefaultSpd);
265             ELSE
266                 sToCylinder3.rSpeedCMD              := 0;
267                 xC3PosReached                       := TRUE;
268                 END_IF;
269             END_IF;
270
271
272             //
273             // Logfile
274             //
275             2:
276             // Enabling Enable relays
277             sToCylinder1.xEnableRelay               := TRUE;
278             sToCylinder2.xEnableRelay               := TRUE;
279             sToCylinder3.xEnableRelay               := TRUE;
280
281             // Checking if all cylinders have reached start positon
282             // if they have, the system will start running in simulated mode
283             IF(xC1PosReached AND xC2PosReached AND xC3PosReached)THEN
284                 xActivateLogFile                    := TRUE;
285
286             // Cylinder1 PID OpMode
287             IF(sFromCylinder1.rPV - sFromCylinder1.rPosSP > 0) THEN
288                 sPID1.iMode                         := 3;
289                 rRCVC1                              :=  (sFromCylinder1.rRCV);// + rLastRCV1)/2;
290                 ELSE
291                 sPID1.iMode                         :=  2;
292                 rRCVC1                              := ((sFromCylinder1.rRCV * -1.0)); // + rLastRCV1) * -1.0)/2;
293                 END_IF;
294              IF(sFromCylinder2.rPV - sFromCylinder2.rPosSP > 0) THEN
295                 sPID2.iMode                         :=  3;
296                 rRCVC2                              :=  (sFromCylinder2.rRCV);// + rLastRCV2)/2;
297                 ELSE
298                 sPID2.iMode                         :=  2;
299                 rRCVC2                              := ((sFromCylinder2.rRCV * -1.0)); // + rLastRCV2) * -1.0)/2;
300                 END_IF;
301               IF(sFromCylinder3.rPV - sFromCylinder3.rPosSP > 0) THEN
302                 sPID3.iMode                         :=  3;
303                 rRCVC3                              :=  (sFromCylinder3.rRCV); // + rLastRCV3)/2;
304                 ELSE
305                 sPID3.iMode                         :=  2;
306                 rRCVC3                              := ((sFromCylinder3.rRCV * -1.0)); // + rLastRCV3) * -1.0)/2;
307                 END_IF;
308
309               sPID1.rPosSP                          := sFromCylinder1.rPosSP;
310               sPID2.rPosSP                          := sFromCylinder2.rPosSP;
311               sPID3.rPosSP                          := sFromCylinder3.rPosSP;
312               sToCylinder1.rSpeedCMD                := fnFeedForwardC1(-sFromCylinder1.rSpeedSP) + rRCVC1;
313               sToCylinder2.rSpeedCMD                := fnFeedForwardC2(-sFromCylinder2.rSpeedSP) + rRCVC2;
314               sToCylinder3.rSpeedCMD                := fnFeedForwardC3(-sFromCylinder3.rSpeedSP) + rRCVC3;
315
316             ELSE
317
318             // Running to start position
```

```
319             IF(sFromCylinder1.rPV < (sDataFromHMI.rC1HeavePosLogF[0] + rStartPosition) AND NOT(xC1PosReached)) THEN
320                 sToCylinder1.rSpeedCMD              := fnFeedForwardC1(-rDefaultSpd);
321             ELSE
322                 sToCylinder1.rSpeedCMD              := 0;
323                 xC1PosReached                       := TRUE;
324                 END_IF;
325             IF(sFromCylinder2.rPV < (sDataFromHMI.rC2_C3HeavePosLogF[0] + rStartPosition) AND NOT(xC2PosReached)) THEN
326                 sToCylinder2.rSpeedCMD              := fnFeedForwardC2(-rDefaultSpd);
327             ELSE
328                 sToCylinder2.rSpeedCMD              := 0;
329                 xC2PosReached                       := TRUE;
330                 END_IF;
331             IF(sFromCylinder3.rPV < (sDataFromHMI.rC2_C3HeavePosLogF[0] + rStartPosition) AND NOT(xC3PosReached)) THEN
332                 sToCylinder3.rSpeedCMD              := fnFeedForwardC3(-rDefaultSpd);
333             ELSE
334                 sToCylinder3.rSpeedCMD              := 0;
335                 xC3PosReached                       := TRUE;
336                 END_IF;
337             END_IF;
338
339             ELSE
340             END_CASE;
341
342
343         //
344         // Reset
345         //
346         2:
347          // Check wich cylinder has the longest way to travel
348             IF((sFromCylinder1.rPV <= sFromCylinder2.rPV) AND NOT(xtravelCalculated))THEN
349                 xC2C3_LongestWay                    := TRUE;
350                 xtravelCalculated                   := TRUE;
351             ELSIF((sFromCylinder1.rPV > sFromCylinder2.rPV) AND NOT(xtravelCalculated))THEN
352                 xC2C3_LongestWay                    := FALSE;
353                 xtravelCalculated                   := TRUE;
354                 END_IF;
355
356              // Cylinder2 and cylinder3 longest travel way
357             IF(xC2C3_LongestWay)THEN
358
359                  //Calculating cylinder2 and 3 traveltime
360                 IF(NOT(xTempTimeCalculated))THEN
361                     rTempTime                       := ABS(sFromCylinder2.rPV)/rDefaultSpd;
362
363                     // Check if valid temp time.
364                     IF((rTempTime > 0) AND (rTempTime < 20)) THEN
365                         // Calculating speed for cylinder1
366                         rTempSPD_C1                     := fnFeedForwardC1((sFromCylinder1.rPV)/rTempTime);
367                     ELSE
368                         rTempSPD_C1                     := fnFeedForwardC1(rDefaultSpd);
369                         END_IF;
370                     xTempTimeCalculated             := TRUE;
371                     END_IF;
372
373
374                     // Check if cylinders has reached pos and not moving
375          if(iWaitCounter > 40) THEN
376
377             If(iResetCounter > 5)THEN
378                 iResetCounter                       := 0;
379             // Cylinder 1 resetting
380             if((uLastPulse1 <> TO_UDINT(sFromCylinder1.dwPulses)) AND NOT(xC1PosReached)) THEN
381                 uLastPulse1                         := TO_UDINT(sFromCylinder1.dwPulses);
382                 sToCylinder1.rSpeedCMD              := rTempSPD_C1;
383                 sToCylinder1.xEnableRelay           := TRUE;
384             ELSE
385                 sToCylinder1.rSpeedCMD              := 0;
386                 sToCylinder1.xEnableRelay           := FALSE;
387                 sToCylinder1.xResetPos              := TRUE;
388                 xC1PosReached                       := TRUE;
389                 END_IF;
390
391             // Cyllinder 2 resetting
392             if((uLastPulse2 <> TO_UDINT(sFromCylinder2.dwPulses)) AND NOT(xC2PosReached)) THEN
393                 uLastPulse2                          := TO_UDINT(sFromCylinder2.dwPulses);
394                 sToCylinder2.rSpeedCMD              := fnFeedForwardC2(rDefaultSpd);
395                 sToCylinder2.xEnableRelay           := TRUE;
396             ELSE
397                 sToCylinder2.rSpeedCMD              := 0;
398                 sToCylinder2.xEnableRelay           := FALSE;
399                 sToCylinder2.xResetPos              := TRUE;
```

```
400             xC2PosReached                       := TRUE;
401             END_IF;
402
403         // Cylinder 3 resetting
404         if((uLastPulse3 <> TO_UDINT(sFromCylinder3.dwPulses)) AND NOT(xC3PosReached)) THEN
405             uLastPulse3                          := TO_UDINT(sFromCylinder3.dwPulses);
406             sToCylinder3.rSpeedCMD              := fnFeedForwardC3(rDefaultSpd);
407             sToCylinder3.xEnableRelay           := TRUE;
408         ELSE
409             sToCylinder3.rSpeedCMD              := 0;
410             sToCylinder3.xEnableRelay           := FALSE;
411             sToCylinder3.xResetPos              := TRUE;
412             xC3PosReached                       := TRUE;
413             END_IF;
414         ELSE
415             iResetCounter                       := iResetCounter + 1;
416             END_IF; // End reset counter if.
417     ELSE
418         iWaitCounter                            :=  iWaitCounter + 1;
419         sToCylinder1.rSpeedCMD                  :=  rTempSPD_C1;
420         sToCylinder2.rSpeedCMD                  := fnFeedForwardC2(rDefaultSpd);
421         sToCylinder3.rSpeedCMD                  := fnFeedForwardC3(rDefaultSpd);
422         sToCylinder1.xEnableRelay               := TRUE;
423         sTOCylinder2.xEnableRelay               := TRUE;
424         sTOCylinder3.xEnableRelay               := TRUE;
425
426     END_IF;
427
428         ELSE // Cylinder 1 longest travel way
429
430              //Calculating cylinder1
431             IF(NOT(xTempTimeCalculated))THEN
432                 rTempTime                       := ABS(sFromCylinder1.rPV)/rDefaultSpd;
433
434                 // Check if valid temp time.
435                 IF((rTempTime > 0) AND (rTempTime < 20)) THEN
436                     // Calculating speed for cylinder2 and cylinder 3
437                     rTempSPD_C2                     := fnFeedForwardC2((sFromCylinder2.rPV)/rTempTime);
438                     rTempSPD_C3                     := fnFeedForwardC3((sFromCylinder3.rPV)/rTempTime);
439                 ELSE
440                     rTempSPD_C2                     := fnFeedForwardC2(rDefaultSpd);
441                     rTempSPD_C3                     := fnFeedForwardC3(rDefaultSpd);
442                     END_IF;
443                 xTempTimeCalculated             := TRUE;
444                 END_IF;
445
446          // Check if cylinders has reached pos and not moving
447      if(iWaitCounter > 40) THEN
448
449         If(iResetCounter > 5)THEN
450             iResetCounter                       := 0;
451         // Cylinder 1 resetting
452         if((uLastPulse1 <> TO_UDINT(sFromCylinder1.dwPulses)) AND NOT(xC1PosReached)) THEN
453             uLastPulse1                         := TO_UDINT(sFromCylinder1.dwPulses);
454             sToCylinder1.rSpeedCMD              := fnFeedForwardC1(rDefaultSpd);
455             sToCylinder1.xEnableRelay           := TRUE;
456         ELSE
457             sToCylinder1.rSpeedCMD              := 0;
458             sToCylinder1.xEnableRelay           := FALSE;
459             sToCylinder1.xResetPos              := TRUE;
460             xC1PosReached                       := TRUE;
461             END_IF;
462
463         // Cyllinder 2 resetting
464         if((uLastPulse2 <> TO_UDINT(sFromCylinder2.dwPulses)) AND NOT(xC2PosReached)) THEN
465             uLastPulse2                          := TO_UDINT(sFromCylinder2.dwPulses);
466             sToCylinder2.rSpeedCMD              := rTempSPD_C2;
467             sToCylinder2.xEnableRelay           := TRUE;
468         ELSE
469             sToCylinder2.rSpeedCMD              := 0;
470             sToCylinder2.xEnableRelay           := FALSE;
471             sToCylinder2.xResetPos              := TRUE;
472             xC2PosReached                       := TRUE;
473             END_IF;
474
475         // Cylinder 3 resetting
476         if((uLastPulse3 <> TO_UDINT(sFromCylinder3.dwPulses)) AND NOT(xC3PosReached)) THEN
477             uLastPulse3                          := TO_UDINT(sFromCylinder3.dwPulses);
478             sToCylinder3.rSpeedCMD              := rTempSPD_C3;
479             sToCylinder3.xEnableRelay           := TRUE;
480         ELSE
```

```
481                 sToCylinder3.rSpeedCMD              := 0;
482                 sToCylinder3.xEnableRelay           := FALSE;
483                 sToCylinder3.xResetPos              := TRUE;
484                 xC3PosReached                       := TRUE;
485                 END_IF;
486             ELSE
487                 iResetCounter                       := iResetCounter + 1;
488                 END_IF; // end resetcounter if
489
490         ELSE
491             iWaitCounter                            :=  iWaitCounter + 1;
492             sToCylinder1.rSpeedCMD                  := fnFeedForwardC1(rDefaultSpd);
493             sToCylinder2.rSpeedCMD                  :=  rTempSPD_C2;
494             sToCylinder3.rSpeedCMD                  :=  rTempSPD_C3;
495             sToCylinder1.xEnableRelay               := TRUE;
496             sTOCylinder2.xEnableRelay               := TRUE;
497             sTOCylinder3.xEnableRelay               := TRUE;
498
499         END_IF;
500
501                 END_IF;
502
503
504          // Check if cylinders are moving
505                  If(xC1PosReached AND xC2PosReached AND xC3PosReached)THEN
506                      xRunning                        := FALSE;
507                      ELSE
508                      xRunning                        := TRUE;
509                      END_IF;
510
511
512         ELSE
513         END_CASE;
514
515
```

Input

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
rPhaseDiff	REAL	Input		Phase diff from UI in deg								

Output

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
rSinDelay	REAL	Output		Sinus delay								
rCosDelay	REAL	Output		Cosinus delay								

Local

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
rCos90Delay	REAL	Local		Constant	REAL#0.25	Private						

Components / Plattform / fbPhaseConversion / Code

```
1  /*************************************************************************
2  Code information:
3  Functionblock that calculates sinus and cosinus delay based on phase difference from UI.
4
5  Author: Sander V. Andersen and Lars Askild S. Aarvik
6  Date: 02.02.2023
7  *************************************************************************/
8
9  rSinDelay               := rPhaseDiff/360;
10 rCosDelay               := rSinDelay - rCos90Delay;
```

Input

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
tResetTime	TIME	Input										
xNotActive	BOOL	Input										

Output

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
xOutput	BOOL	Output										

Local

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
TON1	TON	Local				Private						
TON2	TON	Local				Private						
iState	INT	Local				Private						

Components / Plattform / fbResetCom / Code

```
1   // Used to restart the communication if there is a timeout
2
3   IF xNotActive THEN
4
5       CASE iState OF
6           0 : xOutput := FALSE;
7               iState := 10;
8
9           10 : TON1(IN := NOT(TON1.Q), PT := tResetTime);
10
11                IF TON1.Q THEN
12                   iState := 20;
13                END_IF;
14
15          20 : xOutput := TRUE;
16                iState := 30;
17
18          30 : TON2(IN := NOT(TON2.Q), PT := tResetTime);
19
20                IF TON2.Q THEN
21                   iState := 0;
22                END_IF;
23      END_CASE
24
25  ELSE
26      xOutput := TRUE;
27  END_IF;
```

Input

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
tPeriodeSin	TIME	Input		Periode from UI								
rAmplitudeSin	REAL	Input		Amplitude from UI								

Output

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
tPeriodeCos	TIME	Output		Calculated periode cos								
rAmplitudeCos	REAL	Output		Calculated amplitude cos								

Local

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
PI	REAL	Local		constant	REAL#3.14159265359	Private						

Components / Plattform / fbSinCosConversion / Code

```
/*******************************************************************************
Code information:
Function block that convert periode and amplitude for sinus to period and amplitud for cos (sin').
finding the amplitude and periode for the derivative of sin.

Author: Sander V. Andersen and Lars Askild S. Aarvik
Date: 02.02.2023
*******************************************************************************/

IF(NOT(TO_REAL(TO_DINT(tPeriodeSin))*0.001 = 0))THEN
    tPeriodeCos                 := tPeriodeSin;
    rAmplitudeCos               := rAmplitudeSin * ((2*PI)/(TO_REAL(TO_DINT(tPeriodeSin))*0.001));
    END_IF;
```

Input

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
xActivate	BOOL	Input		Activate test								
rSpeedCmd	REAL	Input		Speed cmd for test								
rCylinderPos	REAL	Input		Cylinder position [m]								
rTestRunLength	REAL	Input		Test run length								

Output

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
xEnableRelay	BOOL	Output		Cylinder enable relay								
rSpeedCmdOut	REAL	Output		Speed command cylinder								
iTimeElapsed	INT	Output		Time elapsed								
rAvgVelocity	REAL	Output		Calculated velocity [m/s]								

Local

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
iTimerCount	INT	Local				Private						
rStartPos	REAL	Local				Private						

```
/*******************************************************************************
Code information:
Function block which is used to speedtest the cylinders.
For feedforward calibration for cylinders.

Author: Sander V. Andersen and Lars Askild S. Aarvik
Date: 18.02.2023
*******************************************************************************/



IF(xActivate) THEN
    // Run speedtest
    IF(ABS(rCylinderPos - rStartPos) < rTestRunLength) THEN
        rSpeedCmdOut            := rSpeedCmd;
        xEnableRelay            := TRUE;
        iTimerCount             := iTimerCount + 1;
        iTimeElapsed            := iTimerCount * 100;
    ELSE
        rSpeedCmdOut            := 0;
        xEnableRelay            := FALSE;
        iTimeElapsed            := iTimerCount * 100;
        rAvgVelocity            := 1000.0* ABS(rCylinderPos - rStartPos)/(iTimeElapsed*0.001);
        END_IF;


ELSE
    // Reset variables
    iTimerCount                 := 0;
    rStartPos                   := rCylinderPos;
    rAvgVelocity                := 0;
    iTimeElapsed                := 0;
    rSpeedCmdOut                := 0;
    xEnableRelay                := FALSE;
    END_IF;
```

Input

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
iOperationMode	INT	Input		Switch OpMode								
rSinC1_PosSP	REAL	Input		PosSP_C1_Simulated								
rSinC2_C3_PosSP	REAL	Input		PosSP_C2_C3_Simulated								
rSinC1_SpdSP	REAL	Input		VelSP_C1_Simulated								
rSinC2_C3_SpdSP	REAL	Input		VelSP_C2_C3_Simulated								
rLogFileC1_PosSP	REAL	Input		PosSP_C1_Logfile								
rLogFileC2_C3_PosSP	REAL	Input		PosSP_C2_C3_Logfile								
rLogFileC1_SpdSP	REAL	Input		VelSP_C1_Logfile								
rLogFileC2_C3_SpdSP	REAL	Input		VelSP_C2_C3_Logfile								
rManualC1_PosSP	REAL	Input		PosSP_C1_Manual								
rManualC2_C3_PosSP	REAL	Input		PosSP_C2_C3_Manual								

Output

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
rC1_PosSP	REAL	Output		PosSP_C1_Output								
rC2_C3_PosSP	REAL	Output		PosSP_C2_C3_Output								
rC1_SpdSP	REAL	Output		VelSP_C1_Output								
rC2_C3_SpdSP	REAL	Output		VelSP_C2_C3_Output								

Components / Plattform / fbSwitchOPmode / Code

```
/*******************************************************************************
Code information:
Functionblock that switches pos/vel setpoints between differents operation modes.
Manual control, Simulated control and Logfile mode.

Author: Sander V. Andersen and Lars Askild S. Aarvik
Date: 10.03.2023
*******************************************************************************/

CASE iOperationMode OF
    ///
    // Manual
    ///
    0:
    rC1_PosSP                       := rManualC1_PosSP;
    rC1_SpdSP                       := 0;

    rC2_C3_PosSP                    := rManualC2_C3_PosSP;
    rC2_C3_SpdSP                    := 0;


    ///
    // Simulated
    ///
    1:
    rC1_PosSP                       :=  rSinC1_PosSP;
    rC1_SpdSP                       :=  rSinC1_SpdSP;

    rC2_C3_PosSP                    :=  rSinC2_C3_PosSP;
    rC2_C3_SpdSP                    :=  rSinC2_C3_SpdSP;


    ///
    // Logfile
    ///
    2:

    rC1_PosSP                       :=  rLogFileC1_PosSP;
    rC1_SpdSP                       :=  rLogFileC1_SpdSP;

    rC2_C3_PosSP                    :=  rLogFileC2_C3_PosSP;
    rC2_C3_SpdSP                    :=  rLogFileC2_C3_SpdSP;


    ELSE
    END_CASE;
```

Input

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
xActivate	BOOL	Input		Activate wave generator								
tPeriode	TIME	Input		Periode [s]								
rAmplitude	REAL	Input		Amplitude [m]								
rPhaseDiff	REAL	Input		Phase shift [deg]								

Output

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
rPosSP_Front	REAL	Output		Positionsetpoint C2_C3								
rPosSP_Rear	REAL	Output		Positionsetpoint C1								
rSpeedSP_Front	REAL	Output		Velocitysetpoint C2_C3								
rSpeddSP_Rear	REAL	Output		Velocitysetpoint C1								

Local

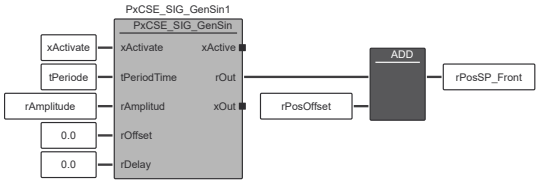
Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
tPeriodeCos	TIME	Local				Private						
rAmplitudeCos	REAL	Local				Private						
rCos90Delay	REAL	Local			REAL#-0.234	Private						
rSinDelayRear	REAL	Local				Private						
rCosDelayRear	REAL	Local				Private						
rPosOffset	REAL	Local			REAL#0.15	Private						

Default

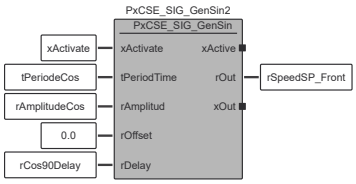
Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
PxCSE_SIG_GenSin1	PxCSE_SIG_GenSin	Local				Private						
PxCSE_SIG_GenSin2	PxCSE_SIG_GenSin	Local				Private						
fbSinCosConversion1	fbSinCosConversion	Local				Private						
PxCSE_SIG_GenSin3	PxCSE_SIG_GenSin	Local				Private						
PxCSE_SIG_GenSin4	PxCSE_SIG_GenSin	Local				Private						
fbPhaseConversion1	fbPhaseConversion	Local				Private						

Front Cylinder

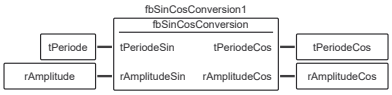
Position SP



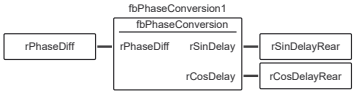
Speed SP



Find Cos periode and Amplitude

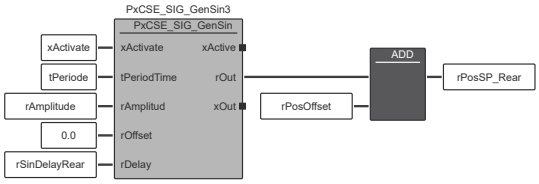


Find sin/cos delay based on phaseshift

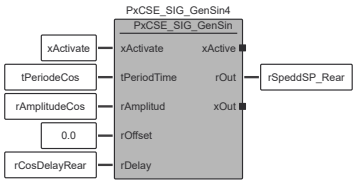


Rear Cylinder

Position SP



Speed SP



Components / Plattform / fnFeedForwardC1 / Signature

Access Specifier: Public

Return Type: REAL

```
/*******************************************************************************
Code information:
Feed-forward function that returns speedcommand for the cylinder1.
Input-argument: Velocity setpoint [m/s]
Output: Speed command [+-10V]

Author: Sander V. Andersen and Lars Askild S. Aarvik
Date: 08.03.2023
*******************************************************************************/

// Only for calculation reasons
    IF(rSpeedSP = 1000.0)THEN
        rSpeedCmdLocal                  := p10v;

    ELSIF(rSpeedSP = -1000.0)THEN
        rSpeedCmdLocal                  := n10v;


     // Positiv speed
    ELSIF(rSpeedSP >= p10v) THEN
        rSpeedCmdLocal                  := 10.0;

    ELSIF(rSpeedSP < p10v AND rSpeedSP >= p6_5v)THEN
        rSpeedCmdLocal                   := 10.0;

    ELSIF(rSpeedSP < p6_5v AND rSpeedSP >= p6v)THEN
        rSpeedCmdLocal                  := ((6.5 - 6.0)/(p6_5v - p6v))*(rSpeedSP - p6_5v) + 6.5;

    ELSIF(rSpeedSP < p6v AND rSpeedSP >= p5_5v)THEN
        rSpeedCmdLocal                  := ((6.0 - 5.5)/(p6v - p5_5v))*(rSpeedSP - p6v) + 6.0;

    ELSIF(rSpeedSP < p5_5v AND rSpeedSP >= p5v)THEN
        rSpeedCmdLocal                  := ((5.5 - 5.0)/(p5_5v - p5v))*(rSpeedSP - p5_5v) + 5.5;

    ELSIF(rSpeedSP < p5v AND rSpeedSP >= p4_5v)THEN
        rSpeedCmdLocal                  := ((5.0 - 4.5)/(p5v - p4_5v))*(rSpeedSP - p5v) + 5.0;

    ELSIF(rSpeedSP < p4_5v AND rSpeedSP >= p4v)THEN
        rSpeedCmdLocal                  := ((4.5 - 4.0)/(p4_5v - p4v))*(rSpeedSP - p4_5v) + 4.5;

    ELSIF(rSpeedSP < p4v AND rSpeedSP >= p3_5v)THEN
        rSpeedCmdLocal                  := ((4.0 - 3.5)/(p4v - p3_5v))*(rSpeedSP - p4v) + 4.0;

    ELSIF(rSpeedSP < p3_5v AND rSpeedSP >= p3v)THEN
        rSpeedCmdLocal                  := ((3.5 - 3.0)/(p3_5v - p3v))*(rSpeedSP - p3_5v) + 3.5;

    ELSIF(rSpeedSP < p3v AND rSpeedSP >= p2_5v)THEN
        rSpeedCmdLocal                  := ((3.0 - 2.5)/(p3v - p2_5v))*(rSpeedSP - p3v) + 3.0;

    ELSIF(rSpeedSP < p2_5v AND rSpeedSP >= p2v)THEN
        rSpeedCmdLocal                  := ((2.5 - 2.0)/(p2_5v - p2v))*(rSpeedSP - p2_5v) + 2.5;

    ELSIF(rSpeedSP < p2v AND rSpeedSP >= p1_5v)THEN
        rSpeedCmdLocal                  := ((2.0 - 1.5)/(p2v - p1_5v))*(rSpeedSP - p2v) + 2.0;

    ELSIF(rSpeedSP < p1_5v AND rSpeedSP >= p1v)THEN
        rSpeedCmdLocal                  := ((1.5 - 1.0)/(p1_5v - p1v))*(rSpeedSP - p1_5v) + 1.5;

    ELSIF(rSpeedSP < p1v AND rSpeedSP >= p0_5v)THEN
        rSpeedCmdLocal                  := ((1.0 - 0.5)/(p1v - p0_5v))*(rSpeedSP - p1v) + 1.0;

    ELSIF(rSpeedSP < p0_5v AND rSpeedSP > 0.0)THEN
        rSpeedCmdLocal                  := ((0.5 - 0.0)/(p0_5v - 0.0))*(rSpeedSP - p0_5v) + 0.5;

    ELSIF(rSpeedSP = 0.0)THEN
        rSpeedCmdLocal                  := 0.0;



    // Negativ speed

    ELSIF(rSpeedSP < n0_5v AND rSpeedSP > n1v)THEN
        rSpeedCmdLocal                  := ((-0.5 - (-1.0))/(n0_5v - n1v))*(rSpeedSP - n0_5v) - 0.5;

    ELSIF(rSpeedSP <= n1v AND rSpeedSP > n1_5v)THEN
        rSpeedCmdLocal                  := ((-1.0 - (-1.5))/(n1v - n1_5v))*(rSpeedSP - n1v) - 1.0;

    ELSIF(rSpeedSP <= n1_5v AND rSpeedSP > n2v)THEN
        rSpeedCmdLocal                  := ((-1.5 - (-2.0))/(n1_5v - n2v))*(rSpeedSP - n1_5v) - 1.5;

    ELSIF(rSpeedSP <= n2v AND rSpeedSP > n2_5v)THEN
```

```
82            rSpeedCmdLocal                    := ((-2.0 - (-2.5))/(n2v - n2_5v))*(rSpeedSP - n2v) - 2.0;
83
84        ELSIF(rSpeedSP <= n2_5v AND rSpeedSP > n3v)THEN
85            rSpeedCmdLocal                    := ((-2.5 - (-3.0))/(n2_5v - n3v))*(rSpeedSP - n2_5v) - 2.5;
86
87        ELSIF(rSpeedSP <= n3v AND rSpeedSP > n3_5v)THEN
88            rSpeedCmdLocal                    := ((-3.0 - (-3.5))/(n3v - n3_5v))*(rSpeedSP - n3v) - 3.0;
89
90        ELSIF(rSpeedSP <= n3_5v AND rSpeedSP > n4v)THEN
91            rSpeedCmdLocal                    := ((-3.5 - (-4.0))/(n3_5v - n4v))*(rSpeedSP - n3_5v) - 3.5;
92
93        ELSIF(rSpeedSP <= n4v AND rSpeedSP > n4_5v)THEN
94            rSpeedCmdLocal                    := ((-4.0 - (-4.5))/(n4v - n4_5v))*(rSpeedSP - n4v) - 4.0;
95
96        ELSIF(rSpeedSP <= n4_5v AND rSpeedSP > n5v)THEN
97            rSpeedCmdLocal                    := ((-4.5 - (-5.0))/(n4_5v - n5v))*(rSpeedSP - n4_5v) - 4.5;
98
99        ELSIF(rSpeedSP <= n5v AND rSpeedSP > n5_5v)THEN
100           rSpeedCmdLocal                    := ((-5.0 - (-5.5))/(n5v - n5_5v))*(rSpeedSP - n5v) - 5.0;
101
102       ELSIF(rSpeedSP <= n5_5v AND rSpeedSP > n6v)THEN
103           rSpeedCmdLocal                    := ((-5.5 - (-6.0))/(n5_5v - n6v))*(rSpeedSP - n5_5v) - 5.5;
104
105       ELSIF(rSpeedSP <= n6v AND rSpeedSP > n10v)THEN
106           rSpeedCmdLocal                    := -10.0;
107       ELSIF(rSpeedSP <= n10v)THEN
108
109           rSpeedCmdLocal                    := -10.0;
110
111
112        END_IF;
113
114
115
116
117
118   // Return SpeedCommandvalue
119   rSpeedCmd       :=  rSpeedCmdLocal;
120   fnFeedForwardC1 :=  rSpeedCmd;
```

Input

Name	Type	Usage	Comment	Init	Constant
rSpeedSP	REAL	Input	Velocity setpoint [m/s]		

Output

Name	Type	Usage	Comment	Init	Constant
rSpeedCmd	REAL	Local	Speed command [+-10V]		

Local

Name	Type	Usage	Comment	Init	Constant
n0_5v	REAL	Local			
n1v	REAL	Local		REAL#-0.00604	
n1_5v	REAL	Local		REAL#-0.01282	
n2v	REAL	Local		REAL#-0.01969	
n2_5v	REAL	Local		REAL#-0.02645	
n3v	REAL	Local		REAL#-0.03322	
n3_5v	REAL	Local		REAL#-0.03981	
n4v	REAL	Local		REAL#-0.0443	
n4_5v	REAL	Local		REAL#-0.04668	
n5v	REAL	Local		REAL#-0.04679	
n5_5v	REAL	Local		REAL#-0.04688	
n6v	REAL	Local		REAL#-0.0469	
n10v	REAL	Local		REAL#-0.0469	
p0_5v	REAL	Local		REAL#0.00752	
p1v	REAL	Local		REAL#0.01401	
p1_5v	REAL	Local		REAL#0.02041	
p2v	REAL	Local		REAL#0.02674	
p2_5v	REAL	Local		REAL#0.03294	
p3v	REAL	Local		REAL#0.03904	
p3_5v	REAL	Local		REAL#0.0451	
p4v	REAL	Local		REAL#0.04929	
p4_5v	REAL	Local		REAL#0.05066	
p5v	REAL	Local		REAL#0.05108	
p5_5v	REAL	Local		REAL#0.05121	
p6v	REAL	Local		REAL#0.05148	
p6_5v	REAL	Local		REAL#0.05157	
p10v	REAL	Local		REAL#0.05157	
rSpeedCmdLocal	REAL	Local			

Components / Plattform / fnFeedForwardC2 / Signature

Access Specifier: Public

Return Type: REAL

```
/*************************************************************************
Code information:
Feed-forward function that returns speedcommand for the cylinder2.
Input-argument: Velocity setpoint [m/s]
Output: Speed command [+-10V]

Author: Sander V. Andersen and Lars Askild S. Aarvik
Date: 08.03.2023
*************************************************************************/

// Only for calculation reasons
    IF(rSpeedSP = 1000.0)THEN
        rSpeedCmdLocal                  := p10v;

    ELSIF(rSpeedSP = -1000.0)THEN
        rSpeedCmdLocal                  := n10v;


     // Positiv speed
    ELSIF(rSpeedSP >= p10v) THEN
        rSpeedCmdLocal                  := 10.0;

    ELSIF(rSpeedSP < p10v AND rSpeedSP >= p6_5v)THEN
        rSpeedCmdLocal                   := 10.0;

    ELSIF(rSpeedSP < p6_5v AND rSpeedSP >= p6v)THEN
        rSpeedCmdLocal                  := ((6.5 - 6.0)/(p6_5v - p6v))*(rSpeedSP - p6_5v) + 6.5;

    ELSIF(rSpeedSP < p6v AND rSpeedSP >= p5_5v)THEN
        rSpeedCmdLocal                  := ((6.0 - 5.5)/(p6v - p5_5v))*(rSpeedSP - p6v) + 6.0;

    ELSIF(rSpeedSP < p5_5v AND rSpeedSP >= p5v)THEN
        rSpeedCmdLocal                  := ((5.5 - 5.0)/(p5_5v - p5v))*(rSpeedSP - p5_5v) + 5.5;

    ELSIF(rSpeedSP < p5v AND rSpeedSP >= p4_5v)THEN
        rSpeedCmdLocal                  := ((5.0 - 4.5)/(p5v - p4_5v))*(rSpeedSP - p5v) + 5.0;

    ELSIF(rSpeedSP < p4_5v AND rSpeedSP >= p4v)THEN
        rSpeedCmdLocal                  := ((4.5 - 4.0)/(p4_5v - p4v))*(rSpeedSP - p4_5v) + 4.5;

    ELSIF(rSpeedSP < p4v AND rSpeedSP >= p3_5v)THEN
        rSpeedCmdLocal                  := ((4.0 - 3.5)/(p4v - p3_5v))*(rSpeedSP - p4v) + 4.0;

    ELSIF(rSpeedSP < p3_5v AND rSpeedSP >= p3v)THEN
        rSpeedCmdLocal                  := ((3.5 - 3.0)/(p3_5v - p3v))*(rSpeedSP - p3_5v) + 3.5;

    ELSIF(rSpeedSP < p3v AND rSpeedSP >= p2_5v)THEN
        rSpeedCmdLocal                  := ((3.0 - 2.5)/(p3v - p2_5v))*(rSpeedSP - p3v) + 3.0;

    ELSIF(rSpeedSP < p2_5v AND rSpeedSP >= p2v)THEN
        rSpeedCmdLocal                  := ((2.5 - 2.0)/(p2_5v - p2v))*(rSpeedSP - p2_5v) + 2.5;

    ELSIF(rSpeedSP < p2v AND rSpeedSP >= p1_5v)THEN
        rSpeedCmdLocal                  := ((2.0 - 1.5)/(p2v - p1_5v))*(rSpeedSP - p2v) + 2.0;

    ELSIF(rSpeedSP < p1_5v AND rSpeedSP >= p1v)THEN
        rSpeedCmdLocal                  := ((1.5 - 1.0)/(p1_5v - p1v))*(rSpeedSP - p1_5v) + 1.5;

    ELSIF(rSpeedSP < p1v AND rSpeedSP >= p0_5v)THEN
        rSpeedCmdLocal                  := ((1.0 - 0.5)/(p1v - p0_5v))*(rSpeedSP - p1v) + 1.0;

    ELSIF(rSpeedSP < p0_5v AND rSpeedSP > 0.0)THEN
        rSpeedCmdLocal                  := ((0.5 - 0.0)/(p0_5v - 0.0))*(rSpeedSP - p0_5v) + 0.5;

    ELSIF(rSpeedSP = 0.0)THEN
        rSpeedCmdLocal                  := 0.0;

    // Negativ speed

    ELSIF(rSpeedSP < 0.0 AND rSpeedSP > n0_5v)THEN
        rSpeedCmdLocal                  := ((0.0 - (-0.5))/(0.0 - n0_5v))*(rSpeedSP - 0.0) + 0.0;

    ELSIF(rSpeedSP <= n0_5v AND rSpeedSP > n1v)THEN
        rSpeedCmdLocal                  := ((-0.5 - (-1.0))/(n0_5v - n1v))*(rSpeedSP - n0_5v) - 0.5;

    ELSIF(rSpeedSP <= n1v AND rSpeedSP > n1_5v)THEN
        rSpeedCmdLocal                  := ((-1.0 - (-1.5))/(n1v - n1_5v))*(rSpeedSP - n1v) - 1.0;

    ELSIF(rSpeedSP <= n1_5v AND rSpeedSP > n2v)THEN
        rSpeedCmdLocal                  := ((-1.5 - (-2.0))/(n1_5v - n2v))*(rSpeedSP - n1_5v) - 1.5;

```

```
82          ELSIF(rSpeedSP <= n2v AND rSpeedSP > n2_5v)THEN
83              rSpeedCmdLocal                  := ((-2.0 - (-2.5))/(n2v - n2_5v))*(rSpeedSP - n2v) - 2.0;
84
85          ELSIF(rSpeedSP <= n2_5v AND rSpeedSP > n3v)THEN
86              rSpeedCmdLocal                  := ((-2.5 - (-3.0))/(n2_5v - n3v))*(rSpeedSP - n2_5v) - 2.5;
87
88          ELSIF(rSpeedSP <= n3v AND rSpeedSP > n3_5v)THEN
89              rSpeedCmdLocal                  := ((-3.0 - (-3.5))/(n3v - n3_5v))*(rSpeedSP - n3v) - 3.0;
90
91          ELSIF(rSpeedSP <= n3_5v AND rSpeedSP > n4v)THEN
92              rSpeedCmdLocal                  := ((-3.5 - (-4.0))/(n3_5v - n4v))*(rSpeedSP - n3_5v) - 3.5;
93
94          ELSIF(rSpeedSP <= n4v AND rSpeedSP > n4_5v)THEN
95              rSpeedCmdLocal                  := ((-4.0 - (-4.5))/(n4v - n4_5v))*(rSpeedSP - n4v) - 4.0;
96
97          ELSIF(rSpeedSP <= n4_5v AND rSpeedSP > n5v)THEN
98              rSpeedCmdLocal                  := ((-4.5 - (-5.0))/(n4_5v - n5v))*(rSpeedSP - n4_5v) - 4.5;
99
100         ELSIF(rSpeedSP <= n5v AND rSpeedSP > n5_5v)THEN
101             rSpeedCmdLocal                  := ((-5.0 - (-5.5))/(n5v - n5_5v))*(rSpeedSP - n5v) - 5.0;
102
103         ELSIF(rSpeedSP <= n5_5v AND rSpeedSP > n6v)THEN
104             rSpeedCmdLocal                  := ((-5.5 - (-6.0))/(n5_5v - n6v))*(rSpeedSP - n5_5v) - 5.5;
105
106         ELSIF(rSpeedSP <= n6v AND rSpeedSP > n10v)THEN
107             rSpeedCmdLocal                  := -10.0;
108
109         ELSIF(rSpeedSP <= n10v)THEN
110             rSpeedCmdLocal                  := -10.0;
111
112
113          END_IF;
114
115
116     // Return SpeedCommandvalue
117     rSpeedCmd       :=  rSpeedCmdLocal;
118     fnFeedForwardC2 :=  rSpeedCmd;
```

Input

Name	Type	Usage	Comment	Init	Constant
rSpeedSP	REAL	Input	Velocity setpoint [m/s]		

Output

Name	Type	Usage	Comment	Init	Constant
rSpeedCmd	REAL	Local	Speed command [+-10V]		

Local

Name	Type	Usage	Comment	Init	Constant
rSpeedCmdLocal	REAL	Local			
n0_5v	REAL	Local		REAL#-0.00195	
n1v	REAL	Local		REAL#-0.00868	
n1_5v	REAL	Local		REAL#-0.01544	
n2v	REAL	Local		REAL#-0.02223	
n2_5v	REAL	Local		REAL#-0.02902	
n3v	REAL	Local		REAL#-0.03576	
n3_5v	REAL	Local		REAL#-0.04252	
n4v	REAL	Local		REAL#-0.04837	
n4_5v	REAL	Local		REAL#-0.05031	
n5v	REAL	Local		REAL#-0.05079	
n5_5v	REAL	Local		REAL#-0.05079	
n6v	REAL	Local		REAL#-0.05082	
n10v	REAL	Local		REAL#-0.05082	
p0_5v	REAL	Local		REAL#0.00716	
p1v	REAL	Local		REAL#0.01367	
p1_5v	REAL	Local		REAL#0.02011	
p2v	REAL	Local		REAL#0.02642	
p2_5v	REAL	Local		REAL#0.03277	
p3v	REAL	Local		REAL#0.03911	
p3_5v	REAL	Local		REAL#0.0452	
p4v	REAL	Local		REAL#0.04872	
p4_5v	REAL	Local		REAL#0.05006	
p5v	REAL	Local		REAL#0.05054	
p5_5v	REAL	Local		REAL#0.05082	
p6v	REAL	Local		REAL#0.05095	
p6_5v	REAL	Local		REAL#0.05108	
p10v	REAL	Local		REAL#0.05108	

Components / Plattform / fnFeedForwardC3 / Signature

Access Specifier: Public

Return Type: REAL

```
/*******************************************************************************
Code information:
Feed-forward function that returns speedcommand for the cylinder3.
Input-argument: Velocity setpoint [m/s]
Output: Speed command [+-10V]

Author: Sander V. Andersen and Lars Askild S. Aarvik
Date: 08.03.2023
*******************************************************************************/

// Only for calculation reasons
    IF(rSpeedSP = 1000.0)THEN
        rSpeedCmdLocal                  := p10v;

    ELSIF(rSpeedSP = -1000.0)THEN
        rSpeedCmdLocal                  := n10v;


     // Positiv speed
    ELSIF(rSpeedSP >= p10v) THEN
        rSpeedCmdLocal                  := 10.0;

    ELSIF(rSpeedSP < p10v AND rSpeedSP >= p7v)THEN
        rSpeedCmdLocal                   := 10.0;

    ELSIF(rSpeedSP < p7v AND rSpeedSP >= p6_5v)THEN
        rSpeedCmdLocal                  := ((7.0 - 6.5)/(p7v - p6_5v))*(rSpeedSP - p7v) + 7.0;

    ELSIF(rSpeedSP < p6_5v AND rSpeedSP >= p6v)THEN
        rSpeedCmdLocal                  := ((6.5 - 6.0)/(p6_5v - p6v))*(rSpeedSP - p6_5v) + 6.5;

    ELSIF(rSpeedSP < p6v AND rSpeedSP >= p5_5v)THEN
        rSpeedCmdLocal                  := ((6.0 - 5.5)/(p6v - p5_5v))*(rSpeedSP - p6v) + 6.0;

    ELSIF(rSpeedSP < p5_5v AND rSpeedSP >= p5v)THEN
        rSpeedCmdLocal                  := ((5.5 - 5.0)/(p5_5v - p5v))*(rSpeedSP - p5_5v) + 5.5;

    ELSIF(rSpeedSP < p5v AND rSpeedSP >= p4_5v)THEN
        rSpeedCmdLocal                  := ((5.0 - 4.5)/(p5v - p4_5v))*(rSpeedSP - p5v) + 5.0;

    ELSIF(rSpeedSP < p4_5v AND rSpeedSP >= p4v)THEN
        rSpeedCmdLocal                  := ((4.5 - 4.0)/(p4_5v - p4v))*(rSpeedSP - p4_5v) + 4.5;

    ELSIF(rSpeedSP < p4v AND rSpeedSP >= p3_5v)THEN
        rSpeedCmdLocal                  := ((4.0 - 3.5)/(p4v - p3_5v))*(rSpeedSP - p4v) + 4.0;

    ELSIF(rSpeedSP < p3_5v AND rSpeedSP >= p3v)THEN
        rSpeedCmdLocal                  := ((3.5 - 3.0)/(p3_5v - p3v))*(rSpeedSP - p3_5v) + 3.5;

    ELSIF(rSpeedSP < p3v AND rSpeedSP >= p2_5v)THEN
        rSpeedCmdLocal                  := ((3.0 - 2.5)/(p3v - p2_5v))*(rSpeedSP - p3v) + 3.0;

    ELSIF(rSpeedSP < p2_5v AND rSpeedSP >= p2v)THEN
        rSpeedCmdLocal                  := ((2.5 - 2.0)/(p2_5v - p2v))*(rSpeedSP - p2_5v) + 2.5;

    ELSIF(rSpeedSP < p2v AND rSpeedSP >= p1_5v)THEN
        rSpeedCmdLocal                  := ((2.0 - 1.5)/(p2v - p1_5v))*(rSpeedSP - p2v) + 2.0;

    ELSIF(rSpeedSP < p1_5v AND rSpeedSP >= p1v)THEN
        rSpeedCmdLocal                  := ((1.5 - 1.0)/(p1_5v - p1v))*(rSpeedSP - p1_5v) + 1.5;

    ELSIF(rSpeedSP < p1v AND rSpeedSP >= p0_5v)THEN
        rSpeedCmdLocal                  := ((1.0 - 0.5)/(p1v - p0_5v))*(rSpeedSP - p1v) + 1.0;

    ELSIF(rSpeedSP < p0_5v AND rSpeedSP > 0.0)THEN
        rSpeedCmdLocal                  := ((0.5 - 0.0)/(p0_5v - 0.0))*(rSpeedSP - p0_5v) + 0.5;

    ELSIF(rSpeedSP = 0.0)THEN
        rSpeedCmdLocal                  := 0.0;

    // Negativ speed

    ELSIF(rSpeedSP < 0.0 AND rSpeedSP > n0_5v)THEN
        rSpeedCmdLocal                  := ((0.0 - (-0.5))/(0.0 - n0_5v))*(rSpeedSP - 0.0) + 0.0;

    ELSIF(rSpeedSP <= n0_5v AND rSpeedSP > n1v)THEN
        rSpeedCmdLocal                  := ((-0.5 - (-1.0))/(n0_5v - n1v))*(rSpeedSP - n0_5v) - 0.5;

    ELSIF(rSpeedSP <= n1v AND rSpeedSP > n1_5v)THEN
        rSpeedCmdLocal                  := ((-1.0 - (-1.5))/(n1v - n1_5v))*(rSpeedSP - n1v) - 1.0;

```

```
82          ELSIF(rSpeedSP <= n1_5v AND rSpeedSP > n2v)THEN
83              rSpeedCmdLocal                  := ((-1.5 - (-2.0))/(n1_5v - n2v))*(rSpeedSP - n1_5v) - 1.5;
84
85          ELSIF(rSpeedSP <= n2v AND rSpeedSP > n2_5v)THEN
86              rSpeedCmdLocal                  := ((-2.0 - (-2.5))/(n2v - n2_5v))*(rSpeedSP - n2v) - 2.0;
87
88          ELSIF(rSpeedSP <= n2_5v AND rSpeedSP > n3v)THEN
89              rSpeedCmdLocal                  := ((-2.5 - (-3.0))/(n2_5v - n3v))*(rSpeedSP - n2_5v) - 2.5;
90
91          ELSIF(rSpeedSP <= n3v AND rSpeedSP > n3_5v)THEN
92              rSpeedCmdLocal                  := ((-3.0 - (-3.5))/(n3v - n3_5v))*(rSpeedSP - n3v) - 3.0;
93
94          ELSIF(rSpeedSP <= n3_5v AND rSpeedSP > n4v)THEN
95              rSpeedCmdLocal                  := ((-3.5 - (-4.0))/(n3_5v - n4v))*(rSpeedSP - n3_5v) - 3.5;
96
97          ELSIF(rSpeedSP <= n4v AND rSpeedSP > n4_5v)THEN
98              rSpeedCmdLocal                  := ((-4.0 - (-4.5))/(n4v - n4_5v))*(rSpeedSP - n4v) - 4.0;
99
100         ELSIF(rSpeedSP <= n4_5v AND rSpeedSP > n5v)THEN
101             rSpeedCmdLocal                  := ((-4.5 - (-5.0))/(n4_5v - n5v))*(rSpeedSP - n4_5v) - 4.5;
102
103         ELSIF(rSpeedSP <= n5v AND rSpeedSP > n5_5v)THEN
104             rSpeedCmdLocal                  := ((-5.0 - (-5.5))/(n5v - n5_5v))*(rSpeedSP - n5v) - 5.0;
105
106         ELSIF(rSpeedSP <= n5_5v AND rSpeedSP > n6v)THEN
107             rSpeedCmdLocal                  := ((-5.5 - (-6.0))/(n5_5v - n6v))*(rSpeedSP - n5_5v) - 5.5;
108
109         ELSIF(rSpeedSP <= n6v AND rSpeedSP > n10v)THEN
110             rSpeedCmdLocal                  := -10.0;
111
112         ELSIF(rSpeedSP <= n10v)THEN
113             rSpeedCmdLocal                  := -10.0;
114
115
116          END_IF;
117
118
119
120     // Return SpeedCommandvalue
121     rSpeedCmd       :=  rSpeedCmdLocal;
122     fnFeedForwardC3:=  rSpeedCmd;
```

Input

Name	Type	Usage	Comment	Init	Constant
rSpeedSP	REAL	Input	Velocity setpoint [m/s]		

Output

Name	Type	Usage	Comment	Init	Constant
rSpeedCmd	REAL	Local	Speed command [+-10V]		

Local

Name	Type	Usage	Comment	Init	Constant
rSpeedCmdLocal	REAL	Local			
n0_5v	REAL	Local		REAL#-0.00235	
n1v	REAL	Local		REAL#-0.00876	
n1_5v	REAL	Local		REAL#-0.01566	
n2v	REAL	Local		REAL#-0.02245	
n2_5v	REAL	Local		REAL#-0.02919	
n3v	REAL	Local		REAL#-0.03595	
n3_5v	REAL	Local		REAL#-0.04278	
n4v	REAL	Local		REAL#-0.04908	
n4_5v	REAL	Local		REAL#-0.0507	
n5v	REAL	Local		REAL#-0.05082	
n5_5v	REAL	Local		REAL#-0.05092	
n6v	REAL	Local		REAL#-0.05095	
n10v	REAL	Local		REAL#-0.05095	
p0_5v	REAL	Local		REAL#0.00676	
p1v	REAL	Local		REAL#0.01319	
p1_5v	REAL	Local		REAL#0.0195	
p2v	REAL	Local		REAL#0.02567	
p2_5v	REAL	Local		REAL#0.03174	
p3v	REAL	Local		REAL#0.03764	
p3_5v	REAL	Local		REAL#0.04363	
p4v	REAL	Local		REAL#0.04723	
p4_5v	REAL	Local		REAL#0.04814	
p5v	REAL	Local		REAL#0.0486	
p5_5v	REAL	Local		REAL#0.04872	
p6v	REAL	Local		REAL#0.04896	
p6_5v	REAL	Local		REAL#0.04896	
p7v	REAL	Local		REAL#0.04908	
p10v	REAL	Local		REAL#0.04908	

Components / Plattform / fbPerformanceTest / Code

```
/*************************************************************************
Code information:
Functionblock that is used to performance test plattform and winch.
Calculates a sinus position setpoint with wanted startperiod which is decreasing each cycle.
Outputs sinus setpoint with static amplitude and decreasing periode.

Author: Sander V. Andersen and Lars Askild S. Aarvik
Date: 02.02.2023
*************************************************************************/


IF(xActivateP_test)THEN
    rB                  :=  (2*rPI)/rPeriode;

    rPosSP_out          :=  rAmplitude * SIN(rB * rX) + 0.15;
    rVelSP_out          :=  ((rAmplitude  * ((2*rPI)/rPeriode)) * COS(rB * (rX-0.5))) * -1;

    rX                  :=  rX + 0.01;

    rPeriode            :=  rStartPeriod - rCycleCount;
    rCycleCount         :=  rCycleCount + 0.001;
    ELSE
    rPosSP_out          :=  0;
    rVelSP_out          :=  0;
    rCycleCount         :=  0;
    rX                  :=  0;
    rPeriode            :=  rStartPeriod;
    END_IF;
```

Input

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
xActivateP_test	BOOL	Input		Activates function block								
rStartPeriod	REAL	Input		Chosen startPeriod for test								
rAmplitude	REAL	Input		Chosen amplitde for test								

Output

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
rPosSP_out	REAL	Output		Calculated position setpoint								
rVelSP_out	REAL	Output		Calculated velocity setpoint								

Local

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
rCycleCount	REAL	Local		Cycle counter variable		Private						
rB	REAL	Local		Variable B for calculate sinus and cosinus		Private						
rPI	REAL	Local		Constant PI	REAL#3.14	Private						
rX	REAL	Local		Variable X for calculate sinus and cosinus		Private						
rPeriode	REAL	Local		Period variable used to calculate sin and cos		Private						

Input

Name	Type	Usage	Translate	Comment	Init	Retain	Constant	OPC	HMI	Proficloud	I/Q
dwEncoder_Cyl1	DWORD	InputPort		Raw data value from Encoder connected on cylinder 1 [0 - ...]							
dwEncoder_Cyl2	DWORD	InputPort		Raw data value from Encoder connected on cylinder 2 [0 - ...]							
dwEncoder_Cyl3	DWORD	InputPort		Raw data value from Encoder connected on cylinder 3 [0 - ...]							

Output

Name	Type	Usage	Translate	Comment	Init	Retain	Constant	OPC	HMI	Proficloud	I/Q
wAnalogOut_Cyl1	WORD	OutputPort									
wAnalogOut_Cyl2	WORD	OutputPort									
wAnalogOut_Cyl3	WORD	OutputPort									
xEnableRelay_Cyl1	BOOL	OutputPort									
xEnableRelay_Cyl2	BOOL	OutputPort									
xEnableRelay_Cyl3	BOOL	OutputPort									
xResetInc_Cyl1	BOOL	OutputPort		Bool to reset incremental encoder cylinder 1							
xResetInc_Cyl2	BOOL	OutputPort		Bool to reset incremental encoder cylinder 2							
xResetInc_Cyl3	BOOL	OutputPort		Bool to reset incremental encoder cylinder 3							

Local

Name	Type	Usage	Translate	Comment	Init	Retain	Constant	OPC	HMI	Proficloud	I/Q
xActivateSinGen	BOOL	Local									
rP	REAL	Local		Gain for P regulator	REAL# 1500.0						
sToC1	sToCylinder	Local		Struct for data to cylinder 1							
sToC2	sToCylinder	Local		Struct for data to cylinder 2							
sToC3	sToCylinder	Local		Struct for data to cylinder 3							
sPID1	sPID	Local		Struct for PID regulator cylinder 1							
sPID2	sPID	Local		Struct for PID regulator cylinder 2							
sPID3	sPID	Local		Struct for PID regulator cylinder 3							
tVariabelPeriode	TIME	Local			TIME#1 0s						
AmplitudeVar	REAL	Local									

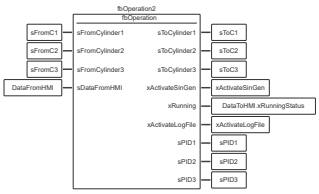
External

Name	Type	Usage	Translate	Comment	Init	Retain	Constant	OPC	HMI	Proficloud	I/Q
sFromC1	sFromCylinder	External		Struct for data from cylinder 1							
sFromC2	sFromCylinder	External		Struct for data from cylinder 2							
sFromC3	sFromCylinder	External		Struct for data from cylinder 3							
DataFromHMI	sFromUI	External		Struct for all data from HMI							
DataToHMI	sToUI	External		Struct for all data to HMI							
rC1SpdSp_LogFile	REAL	External		SpeedSetpoint to cylinder 1 calculated from logfile							
rC1PosSp_LogFile	REAL	External		PositionSetpoint to cylinder 1 calculated from logfile							
rC2_C3PosSp_LogFile	REAL	External		PositionSetpoint to cylinder 2 and 3 calculated from logfile							
rC2_C3SpdSp_LogFile	REAL	External		SpeedSetpoint to cylinder 2 and 3 calculated from logfile							
xActivateLogFile	BOOL	External		Bool used to activate logfile mode							

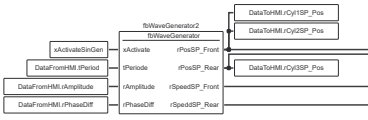
Default

Name	Type	Usage	Translate	Comment	Init	Retain	Constant	OPC	HMI	Proficloud	I/Q
FB_AutoTunePID1	FB_AutoTunePID	Local									
AXL_ANALOG_OUT_31	AXL_ANALOG_OUT_3	Local									
fbWaveGenerator2	fbWaveGenerator	Local									
fbOperation2	fbOperation	Local									
fbSwitchOPmode1	fbSwitchOPmode	Local									
FB_AutoTunePID2	FB_AutoTunePID	Local									
FB_AutoTunePID3	FB_AutoTunePID	Local									
AXL_ANALOG_OUT_1	AXL_ANALOG_OUT_3	Local									
AXL_ANALOG_OUT_2	AXL_ANALOG_OUT_3	Local									
rC1PosSP_Manual	REAL	External		External cylinder 1 position setpoint in manual mode							
rC2_C3PosSP_Manual	REAL	External		External cylinder 2 and 3 position setpoint in manual mode							
fbSpeed_V1_021	fbSpeed_V1_02	Local									
fbPulseToSpeed2	fbPulseToSpeed	Local									
fbSpeed_V1_1	fbSpeed_V1_02	Local									
fbPulseToSpeed3	fbPulseToSpeed	Local									
fbSpeed_V1_2	fbSpeed_V1_02	Local									
fbPulseToSpeed4	fbPulseToSpeed	Local									
fbEncConverter1	fbEncConverter	Local									
fbEncConverter2	fbEncConverter	Local									
fbEncConverter4	fbEncConverter	Local									
fbPerformanceTest1	fbPerformanceTest	Local									

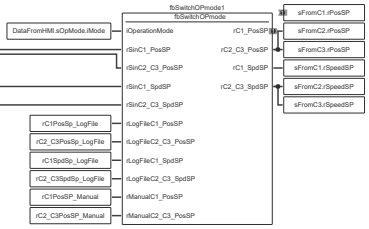
Plattform control function block



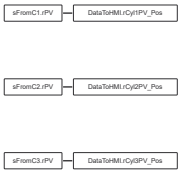
Sinus/Cosinus generator



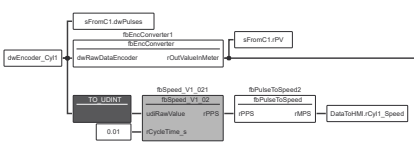
Switch Pos/Spd setpoint from operation mode



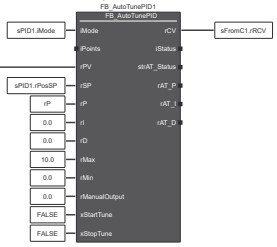
Mapping data to UI



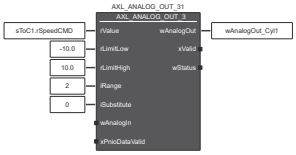
Scaleblock of data from cyl1-encoder



PID Cylinder 1



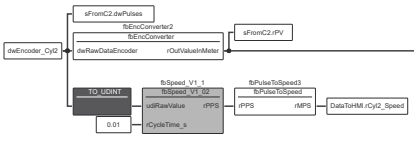
Analog out converter cylinder 1



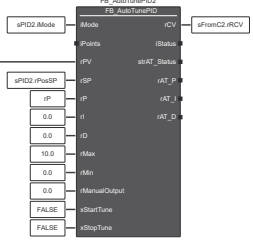
Activates cyl1 relay

Reset cyl1 encoder

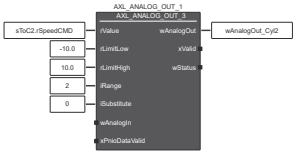
Scaleblock of data from cyl2-encoder



PID Cylinder 2



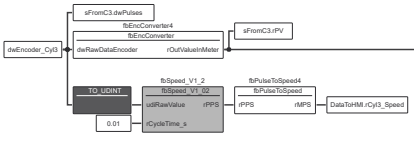
Analog out converter cylinder 2



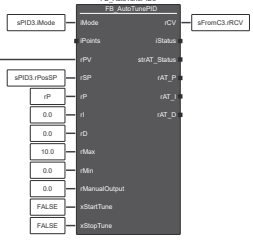
Activates cyl2 relay

Reset cyl2 encoder

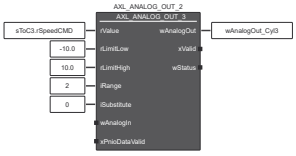
Scaleblock of data from cyl3-encoder



PID Cylinder 3



Analog out converter cylinder 3



Activates cyl3 relay

Reset cyl3 encoder

Components / Vacon / fbFromDrive / Code

```
/*************************************************************************
Code information:
Function block that maps data from Vacon drive to PLC.


Author: Sander V. Andersen and Lars Askild S. Aarvik
Date: 05.04.2023
*************************************************************************/


// Mapping from Drive
DataFromDrive.xReady                := Rawdata[1].X0;
DataFromDrive.xRun                  := Rawdata[1].X1;
DataFromDrive.xCCW                  := Rawdata[1].X2;
DataFromDrive.xFault                := Rawdata[1].X3;
DataFromDrive.xWarning              := Rawdata[1].X4;
DataFromDrive.xAtRefSpd             := Rawdata[1].X5;
DataFromDrive.xZeroSpd              := Rawdata[1].X6;
DataFromDrive.xFluxRdy              := Rawdata[1].X7;

DataFromDrive.iPar3                 := fnWordToInt(FALSE, Rawdata[4]);
DataFromDrive.iMotorRPM             := fnWordToInt(FALSE, Rawdata[5]);
DataFromDrive.iPar6                 := fnWordToInt(FALSE, Rawdata[7]);


```

Input

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
Rawdata	MB_TCP_ARR_W_1_125	Input		Rawdata modbus registers								

Output

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
DataFromDrive	FromVaconDriveEng	Output		Struct with data from drive								

Components / Vacon / fbToDrive / Code

```
/*******************************************************************************
Code information:
Function block that maps data to Vacon drive from PLC.


Author: Sander V. Andersen and Lars Askild S. Aarvik
Date: 05.04.2023
*******************************************************************************/


// Mapping to drive
wToDriveRaw[1].X0               :=  sVaconDrive.xRun;
wToDriveRaw[1].X1               :=  sVaconDrive.xDir;
wToDriveRaw[1].X2               :=  sVaconDrive.xRstErr;
wToDriveRaw[3]                  :=  sVaconDrive.wSpeedCMD;
wToDriveRaw[4]                  :=  sVaconDrive.wTorqueCMD;

```

Input

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
sVaconDrive	ToVaconDriveEng	Input		Struct with data to drive								

Output

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
wToDriveRaw	MB_TCP_ARR_W_1_125	Output		modbus registers to drive								

Default

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
sOpPanelIf	sPanelInterface	External		Struct with data to/from op panel								
sToDrive	ToVaconDriveEng	External		Struct with data to drive								

```
/*****************************************************************************
Code information:
Datatype for drive data

Author: Sander V. Andersen and Lars Askild S. Aarvik
Date: 02.04.2023
*****************************************************************************/

TYPE
WORD_0_3                        :   ARRAY[0..3]OF WORD;
WORD_0_7                        :   ARRAY[0..7]OF WORD;
END_TYPE

TYPE
    // Structs to VACON:
    ToVaconDriveRaw             :   STRUCT
       wCtrl                    :   WORD;   (* X0: Run, X1: Dir, X2: RstErr*)
       wSpare                   :   WORD;
       wSpeed                   :   WORD;
       wTorque                  :   WORD;
       END_STRUCT

    ToVaconDriveEng             :   STRUCT
        xRun                    :   BOOL;   // Run command
        xDir                    :   BOOL;   // Direction command
        xRstErr                 :   BOOL;   // Reset drive errors
        wSpeedCMD               :   WORD;   // Speed command [0 - 10000]
        wTorqueCMD              :   WORD;   // Torque limitation command [0 - 10000]
        END_STRUCT


     // Structs From Vacon:
     FromVaconDriveRaw          :   STRUCT
        wStatus                 :   WORD;
        wSpare1                 :   WORD;
        wSpare2                 :   WORD;
        wSpare3                 :   WORD;
        wSpare4                 :   WORD;
        wSpare5                 :   WORD;
        wSpare6                 :   WORD;
        wSpare7                 :   WORD;
        END_STRUCT

     FromVaconDriveEng          :   STRUCT
        xReady                  :   BOOL;   // Drive ready
        xRun                    :   BOOL;   // Motor running
        xCCW                    :   BOOL;   // Motor running counter clockwise
        xFault                  :   BOOL;   // Drive fault
        xWarning                :   BOOL;   // Drive warning
        xAtRefSpd               :   BOOL;   // motor at reference speed
        xZeroSpd                :   BOOL;   // Motor at zero speed
        xFluxRdy                :   BOOL;   // Flux ready
        iPar3                   :   INT;    // Par3
        iMotorRPM               :   INT;    // Motor RPM
        iPar6                   :   INT;    // Par6
        END_STRUCT

END_TYPE
```

Components / Winch / fbPulseToSpeed / Code

```
/*************************************************************************
Code information:
Function block that calculates pulses per second to meter/second for winch.

Author: Sander V. Andersen and Lars Askild S. Aarvik
Date: 14.04.2023
*************************************************************************/

rMPS                := rPPS    *   (  rDistance/iPulses  );
```

Input

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
rPPS	REAL	Input		Pulses per second from encoder								

Output

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
rMPS	REAL	Output		Speed per second for cylinder								

Local

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
rDistance	REAL	Local		TravelDistance cylinder	REAL#0.3	Private						
iPulses	INT	Local		Number of pulses encoder has	INT#800	Private						

Components / Winch / fbWinchScale / Code

```
1  /*************************************************************************
2  Code information:
3  Function block that calculates winch position in meter, using encoder pulses.
4  GearRatio = 1/10
5  iPPR = pulses per rotation
6
7  Author: Sander V. Andersen and Lars Askild S. Aarvik
8  Date: 13.04.2023
9  *************************************************************************/
10
11
12 rWinchPos                      :=  TO_REAL(diEncoderWinch) * ( rCircumference/ (iPPR*rGearRatio*4)  ) * -1; // position in
   [m]
13
```

Input

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
diEncoderWinch	DINT	Input		Encoder pulses								

Output

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
rWinchPos	REAL	Output		Wire length								

Local

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
rGearRatio	REAL	Local			REAL#10.0	Private						
rCircumference	REAL	Local			REAL#0.278	Private						
iPPR	INT	Local		Pulses per revolution	INT#1024	Private						

Components / Winch / fnPosToPulse / Signature

Access Specifier: Public

Return Type: UDINT

```
/*************************************************************************
Code information:
Function that converts a given position in meter to pulses.
Returns UDINT variable.


Author: Sander V. Andersen and Lars Askild S. Aarvik
Date: 13.04.2023
*************************************************************************/

udiPulse                        :=  TO_UDINT(rPosition * ((iPPR * rGearRatio * 4) / Omkrets));

// Returns pulse
fnPosToPulse                    :=  udiPulse;
```

Input

Name	Type	Usage	Comment	Init	Constant
rPosition	REAL	Input	Position in meter		

Local

Name	Type	Usage	Comment	Init	Constant
udiPulse	UDINT	Local	Pulses		
rGearRatio	REAL	Local		REAL#10.0	
Omkrets	REAL	Local	Circumference	REAL#0.279	
iPPR	INT	Local	Pulses per revolution	INT#1024	

```
/*******************************************************************************
Code information:
Function block that automatic speedtests the winch, with ramping signals [0-100]% in each direction
Returns array with RPM for each speedcommand.

Author: Sander V. Andersen and Lars Askild S. Aarvik
Date: 27.04.2023
*******************************************************************************/


IF(xActivate)THEN

    IF(NOT(xTestDone))THEN
        xRun                        := TRUE;
        wTorqueCMD                  := 3000;
        xDir                        := xDirection;

        If(iTimerCount < iTestTime)THEN

            iTimerCount             := iTimerCount + 1;
        ELSE
            IF(wSpeedCMD <> 0)THEN
                rSpeedArray[uiSpdCmdLocal/1000]       := iRPM / rGearRatio;
                END_IF;
            uiSpdCmdLocal           := uiSpdCmdLocal + TO_UINT(rSpdStep);
            wSpeedCMD               := TO_WORD(uiSpdCmdLocal);
            iTimerCount             := 0;
            IF(wSpeedCMD > 10000)THEN
                xTestDone           := TRUE;
                END_IF;
            END_IF;

     ELSE
        xRun                        := FALSE;
        wTorqueCMD                  := 0;
        wSpeedCMD                   := 0;
        END_IF;


ELSE    // Reset all variables

    wSpeedCMD                       := 0;
    wTorqueCMD                      := 0;
    xRun                            := FALSE;
    rSpdStep                        := 10000 / iSteps;
    iTimerCount                     := iTestTime + 1;
    uiSpdCmdLocal                   := 0;
    xTestDone                       := FALSE;

    FOR i := 0 TO iSteps DO
    rSpeedArray[i]                  := 0;
    END_FOR;

    END_IF;
```

Input

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
xActivate	BOOL	Input		Activate speedtest								
xDirection	BOOL	Input		Set direction								
rGearRatio	REAL	Input		Set gear ratio								
iRPM	INT	Input		Measurres RPM								

Output

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
rSpeedArray	REAL_0_10	Output		Array with RPM for each step								
wSpeedCMD	WORD	Output		Speed command to drive								
wTorqueCMD	WORD	Output		Torque command to drive								
xDir	BOOL	Output		Driection to drive								
xRun	BOOL	Output		Run signal to drive								
xTestDone	BOOL	Output		Bool test done								

Local

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
iSteps	INT	Local		Number of steps	INT#10	Private						
i	INT	Local				Private						
iTimerCount	INT	Local				Private						
rSpdStep	REAL	Local				Private						
iTestTime	INT	Local			INT#300	Private						
uiSpdCmdLocal	UINT	Local				Private						

Default

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
sFromDrive	FromVaconDriveEng	External		Struct with data from drive								
sToDrive	ToVaconDriveEng	External		Struct with data to drive								

Components / Winch / fnCalculateMpPos / Signature

Access Specifier: Public

Return Type: REAL

```
1  /*************************************************************************
2  Code information:
3  Function for calculating MpPos.
4  Takes cylinder 1,2,3 positions as input arguments and returns Mp position with given y and z offsets.
5
6  Author: Sander V. Andersen and Lars Askild S. Aarvik
7  Date: 29.04.2023
8  *************************************************************************/
9
10 // Calulate heave pos and roll angle for center of plattform
11 rRollAngle                    := ASIN((rCyl1Pos - ((rCyl2Pos + rCyl3Pos) / 2)) / rPlattformCC) * (180 / PI);
12 rHeavePos                     := rCyl1Pos - (((SIN(rRollAngle * (PI / 180))) * rPlattformCC) / 2);
13
14 // Calculate crane tip position relativ to plattform and return position relativ to center of wave (0.15m)
15 rMpPos                        := (rTipVectY * SIN(rRollAngle * (PI / 180))) + (rTipVectZ * COS(rRollAngle * (PI / 180)))
   - rTipVectZ + rHeavePos - rPlattformCenter;
16
17
18 // Return MpPos
19 fnCalculateMpPos              := rMpPos;
```

Input

Name	Type	Usage	Comment	Init	Constant
rCyl1Pos	REAL	Input	Cyl1 position [m]		
rCyl2Pos	REAL	Input	Cyl2 position [m]		
rCyl3Pos	REAL	Input	Cyl3 position [m]		

Output

Name	Type	Usage	Comment	Init	Constant
rMpPos	REAL	Local	Mp position [m]		

Local

Name	Type	Usage	Comment	Init	Constant
rPlattformCC	REAL	Local	CC disctance front-rear cylinders	REAL#0.425	
rTipVectX	REAL	Local	From center of plattform to cranetip		
rTipVectY	REAL	Local	From center of plattform to cranetip	REAL#-0.465	
rTipVectZ	REAL	Local	From center of plattform to cranetip	REAL#-0.26	
PI	REAL	Local		REAL#3.14159265359	
rPlattformCenter	REAL	Local	Cente point for plattform	REAL#0.15	
rHeavePos	REAL	Local			
rRollAngle	REAL	Local			

Components / Winch / fbCalculateMpSpd / Code

```
/*************************************************************************
Code information:
Functionblock for calculating Mp Speed.
Takes MpPos as input and finds the derivative and returns the Mp speed as output.
Lowpass filters Mp speed based on ICounterconstant filtertime

Author: Sander V. Andersen and Lars Askild S. Aarvik
Date: 30.04.2023
*************************************************************************/

IF(xActivate)THEN
    IF(iCount >= iCountconstant)THEN
        rMpSpd                  := (rMpPos - rLastMpPos) / (rCycleTime * iCountconstant);
        rLastMpPos              := rMpPos;
        iCount                  := 0;
    ELSE
        iCount                  := iCount + 1;
        END_IF;

ELSE
    rLastMpPos                 := 0;
    END_IF;
```

Input

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
xActivate	BOOL	Input		Activate fb								
rMpPos	REAL	Input		Mp Position								

Output

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
rMpSpd	REAL	Output		Filtered Mp velocity								

Local

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
rCycleTime	REAL	Local			REAL#0.01	Private						
rLastMpPos	REAL	Local				Private						
rLastMpSpd	REAL	Local				Private						
rMpSpdTemp	REAL	Local				Private						
iCount	INT	Local				Private						
iCountconstant	INT	Local		Filter time	INT#8	Private						

Components / Winch / fnFeedForwardWinch / Signature

Access Specifier: Public

Return Type: REAL

```
/*************************************************************************
Code information:
Input: wanted speed [m/s]
Output: +-100% speedCMD
Feed-forward function that returns speedcommand for the winch.

Author: Sander V. Andersen and Lars Askild S. Aarvik
Date: 27.04.2023
*************************************************************************/


// Positiv speed
IF(rSpeedSP >= p100) THEN
    rSpeedCmdLocal                  := 100;

ELSIF(rSpeedSP < p100 AND rSpeedSP >= p90)THEN
    rSpeedCmdLocal                   := (10 / (p100 - p90)) * (rSpeedSP - p100) + 100;

ELSIF(rSpeedSP < p90 AND rSpeedSP >= p80)THEN
    rSpeedCmdLocal                  := (10 / (p90 - p80)) * (rSpeedSP - p90) + 90;

ELSIF(rSpeedSP < p80 AND rSpeedSP >= p70)THEN
    rSpeedCmdLocal                  := (10 / (p80 - p70)) * (rSpeedSP - p80) + 80;

ELSIF(rSpeedSP < p70 AND rSpeedSP >= p60)THEN
    rSpeedCmdLocal                  := (10 / (p70 - p60)) * (rSpeedSP - p70) + 70;

ELSIF(rSpeedSP < p60 AND rSpeedSP >= p50)THEN

    rSpeedCmdLocal                  := (10 / (p60 - p50)) * (rSpeedSP - p60) + 60;

ELSIF(rSpeedSP < p50 AND rSpeedSP >= p40)THEN

    rSpeedCmdLocal                  := (10 / (p50 - p40)) * (rSpeedSP - p50) + 50;

ELSIF(rSpeedSP < p40 AND rSpeedSP >= p30)THEN
    rSpeedCmdLocal                  := (10 / (p40 - p30)) * (rSpeedSP - p40) + 40;

ELSIF(rSpeedSP < p30 AND rSpeedSP >= p20)THEN
    rSpeedCmdLocal                  := (10 / (p30 - p20)) * (rSpeedSP - p30) + 30;

ELSIF(rSpeedSP < p20 AND rSpeedSP >= p10)THEN
    rSpeedCmdLocal                  := (10 / (p20 - p10)) * (rSpeedSP - p20) + 20;

ELSIF(rSpeedSP < p10 AND rSpeedSP >= 0.0)THEN
    rSpeedCmdLocal                  := (10 / (p10)) * (rSpeedSP - p10) + 10;

ELSIF(rSpeedSP = 0.0)THEN
    rSpeedCmdLocal                  := 0;


// Negativ speed

ELSIF(rSpeedSP < 0.0 AND rSpeedSP > n10)THEN
    rSpeedCmdLocal                  := ((10 / (-n10)) * (rSpeedSP));

ELSIF(rSpeedSP <= n10 AND rSpeedSP > n20)THEN
    rSpeedCmdLocal                  := ((10 / (n10 - n20)) * (rSpeedSP - n10) - 10);

ELSIF(rSpeedSP <= n20 AND rSpeedSP > n30)THEN
    rSpeedCmdLocal                  := ((10 / (n20 - n30)) * (rSpeedSP - n20) - 20);

ELSIF(rSpeedSP <= n30 AND rSpeedSP > n40)THEN
    rSpeedCmdLocal                  := ((10 / (n30 - n40)) * (rSpeedSP - n30) - 30);

ELSIF(rSpeedSP <= n40 AND rSpeedSP > n50)THEN
    rSpeedCmdLocal                  := ((10 / (n40 - n50)) * (rSpeedSP - n40) - 40);

ELSIF(rSpeedSP <= n50 AND rSpeedSP > n60)THEN
    rSpeedCmdLocal                  := ((10 / (n50 - n60)) * (rSpeedSP - n50) - 50);

ELSIF(rSpeedSP <= n60 AND rSpeedSP > n70)THEN
    rSpeedCmdLocal                  := ((10 / (n60 - n70)) * (rSpeedSP - n60) - 60);

ELSIF(rSpeedSP <= n70 AND rSpeedSP > n80)THEN

    rSpeedCmdLocal                  := ((10 / (n70 - n80)) * (rSpeedSP - n70) - 70);

ELSIF(rSpeedSP <= n80 AND rSpeedSP > n90)THEN
    rSpeedCmdLocal                  := ((10 / (n80 - n90)) * (rSpeedSP - n80) - 80);

```

```
82    ELSIF(rSpeedSP <= n90 AND rSpeedSP > n100)THEN
83        rSpeedCmdLocal                    := ((10 / (n90 - n100)) * (rSpeedSP - n90) - 90);
84
85    ELSIF(rSpeedSP <= n100)THEN
86
87        rSpeedCmdLocal                    := -100;
88        END_IF;
89
90
91
92
93
94    // Return SpeedCommandvalue
95    rSpeedCmd                              :=  rSpeedCmdLocal;
96    fnFeedForwardWinch                     :=  rSpeedCmd;
```

Input

Name	Type	Usage	Comment	Init	Constant
rSpeedSP	REAL	Input	Speed setpoint [m/s]		

Output

Name	Type	Usage	Comment	Init	Constant
rSpeedCmd	REAL	Local	Speed command +-100%		

Local

Name	Type	Usage	Comment	Init	Constant
n10	REAL	Local		REAL#-0.08293667	
n20	REAL	Local		REAL#-0.1668	
n30	REAL	Local		REAL#-0.2502	
n40	REAL	Local		REAL#-0.3336	
n50	REAL	Local		REAL#-0.41653667	
n60	REAL	Local		REAL#-0.5004	
n70	REAL	Local		REAL#-0.5838	
n80	REAL	Local		REAL#-0.6672	
n90	REAL	Local		REAL#-0.7506	
n100	REAL	Local		REAL#-0.834	
p10	REAL	Local		REAL#0.0834	
p20	REAL	Local		REAL#0.1668	
p30	REAL	Local		REAL#0.2502	
p40	REAL	Local		REAL#0.33313667	
p50	REAL	Local		REAL#0.41653667	
p60	REAL	Local		REAL#0.5004	
p70	REAL	Local		REAL#0.5838	
p80	REAL	Local		REAL#0.66673667	
p90	REAL	Local		REAL#0.75013667	
p100	REAL	Local		REAL#0.83353667	
rSpeedCmdLocal	REAL	Local			

Input

Name	Type	Usage	Translate	Comment	Init	Retain	Constant	OPC	HMI	Proficloud	I/Q
dwRawEncoderWinch	DWORD	InputPort		Encoder pulses winch							

Local

Name	Type	Usage	Translate	Comment	Init	Retain	Constant	OPC	HMI	Proficloud	I/Q
iOperationMode	INT	Local		Operation mode winch							
rWinchPosition	REAL	Local		WIre length winch							
xActivateTOF	BOOL	Local			TRUE						
tBuzzerTime	TIME	Local			TIME#5s						
iLightCounter	INT	Local									
xFirstCycle	BOOL	Local			TRUE						
udiEncoderWinch	UDINT	Local									
diLastEncWinchVal	DINT	Local									
diWinchPulsCount	DINT	Local									
udiResetLenght	UDINT	Local									
rMinWirelLengthAHC	REAL	Local			REAL#0.30						
rMaxWireLengthAHC	REAL	Local			REAL#0.90						
rRampTime	REAL	Local		Ramp time for joystick control	REAL#2.0						
rSpdCmdPercentage	REAL	Local									
xEnableRamping	BOOL	Local									
rMP_Pos	REAL	Local		Mp position							
rMP_Spd	REAL	Local		Mp velocity							
iModeAHC	INT	Local									
rP_AHC	REAL	Local		Gain for p regulator in AHC mode	REAL#200.0						
rRCV_AHC	REAL	Local		Output P-controller							
rAHC_CenterPos	REAL	Local									
rSpdCmd	REAL	Local									
rOffsetVar	REAL	Local									
rLastMP_Pos	REAL	Local									
rMP_PosNow	REAL	Local									
rLastLastMP_Pos	REAL	Local									
xAlarmDone	BOOL	Local									
xLastBuzzer	BOOL	Local									

External

Name	Type	Usage	Translate	Comment	Init	Retain	Constant	OPC	HMI	Proficloud	I/Q
sFromDrive	FromVaconDriveEng	External		Struct with data from drive							
DataToHMI	sToUI	External		Struct for all data to HMI							
rMpPos_Camera1	REAL	External		Mp position from camera							

Default

Name	Type	Usage	Translate	Comment	Init	Retain	Constant	OPC	HMI	Proficloud	I/Q
sToDrive	ToVaconDriveEng	External		Struct with data to drive							
sOpPanelIf	sPanelInterface	External		Struct with data to/from op panel							
TOF1	TOF	Local									
fbEncoderCount_V1_001	fbEncoderCount_V1_00	Local									
fbWinchScale2	fbWinchScale	Local									
DataFromHMI	sFromUI	External		Struct for all data from HMI							
fbSpeedTestWinch1	fbSpeedTestWinch	Local									
fbSpeedTestWinch2	fbSpeedTestWinch	Local									
fbRamping_V1_001	fbRamping_V1_00	Local									
fbCalculateMpSpd1	fbCalculateMpSpd	Local									
sFromC1	sFromCylinder	External		Struct for data from cylinder 1							
sFromC2	sFromCylinder	External		Struct for data from cylinder 2							
sFromC3	sFromCylinder	External		Struct for data from cylinder 3							
fbCalculateMpSpd2	fbCalculateMpSpd	Local									
PID1	PID	Local									
FB_AutoTunePID1	FB_AutoTunePID	Local									
rMpPos_Camera	LREAL	External									

```
/*************************************************************************
Code information:
Program that runs the plattform winch via operator panel commands and in AHC.
4 different modes; manual, Fault, AHC and standby.


Author: Sander V. Andersen and Lars Askild S. Aarvik
Date: 13.04.2023
*************************************************************************/

// Timer for panelinterface alarm buzzer
TOF1(IN := xActivateTOF, PT := tBuzzerTime, Q => sOpPanelIf.xBuzzer, ET => );

// Calculate and filter Mp Position
rMP_PosNow  := fnCalculateMpPos(sFromC1.rPV (* rCyl1Pos *), sFromC2.rPV (* rCyl2Pos *), sFromC3.rPV (* rCyl3Pos *));
rMP_Pos     :=  (rMP_PosNow + rLastMP_Pos + rLastLastMP_Pos)/3;
rLastLastMP_Pos := rLastMP_Pos;
rLastMP_Pos :=  rMP_PosNow;


// Calculate Mp velocity
fbCalculateMpSpd2(xActivate := TRUE, rMpPos :=rMP_Pos, rMpSpd => rMP_Spd);

// Set operationmode based on button inputs
IF(sOpPanelIf.xStandBy)THEN
    iOperationMode              :=  0;
    END_IF;

IF(sOpPanelIf.xManual)THEN
    iOperationMode              :=  1;
    END_IF;

IF(sOpPanelIf.xAHC)THEN
    // Only enter AHC if its safe
    // Wire length is beteween safety margins
    // Plattform is moving in simulated og logifle mode
    IF((rWinchPosition < rMaxWireLengthAHC) AND (rWinchPosition > rMinWirelLengthAHC)
        AND (DataFromHMI.sOpMode.xSimulated OR DataFromHMI.sOpMode.xLogFile) AND DataFromHMI.sCommand.xStart)THEN
        iOperationMode              :=  2;
        END_IF;
    END_IF;

IF(sFromDrive.xFault OR sFromDrive.xWarning)THEN
    iOperationMode              :=  3;
    END_IF;

IF(sOpPanelIf.xReset)THEN
    sToDrive.xRstErr            :=  TRUE;
ELSE
    sToDrive.xRstErr            := FALSE;
    END_IF;



// Switch between operation modes for winch
CASE iOperationMode OF
    // Standby
    0:
    sToDrive.xRun               :=  FALSE;
    sOpPanelIf.xManualLight     :=  FALSE;
    sOpPanelIf.xStandbyLight    :=  TRUE;
    sOpPanelIf.xAHCLight        :=  FALSE;
    sOpPanelIf.xResetLight      :=  FALSE;
    sOpPanelIf.xBuzzer          :=  FALSE;
    xActivateTOF                :=  FALSE;
    xAlarmDone                  :=  FALSE;
    iLightCounter               :=  -1;
    rAHC_CenterPos              :=  0.0;
    DataToHMI.wWinchOpMode      := 0;
    rOffsetVar                  := 0;

    // Manual control
    1:
    sToDrive.xRun               :=  TRUE;
    sOpPanelIf.xManualLight     :=  TRUE;
    sOpPanelIf.xStandbyLight    :=  FALSE;
    sOpPanelIf.xAHCLight        :=  FALSE;
    sOpPanelIf.xResetLight      :=  FALSE;
    sOpPanelIf.xBuzzer          :=  FALSE;
    rAHC_CenterPos              :=  0.0;
    DataToHMI.wWinchOpMode      := 1;
```

```
82
83
84          IF(sOpPanelIf.wJoySw = 1)THEN
85              // Pos direction
86              sToDrive.xDir                   :=  FALSE;
87          ELSE // neg direction
88              sToDrive.xDir                   :=  TRUE;
89              END_IF;
90
91          // Manual control with joystick and potmeter
92          IF(sOpPanelIf.wJoySw = 1 OR sOpPanelIf.wJoySw = 2)THEN
93              xEnableRamping                  :=  TRUE;
94              sToDrive.wSpeedCMD              :=  TO_WORD(TO_UINT(rSpdCmdPercentage * 100.0 * (sOpPanelIf.rPot1/100)));
95              sToDrive.wTorqueCMD             :=  3000;
96              ELSE
97              sToDrive.wSpeedCMD              :=  0;
98              sToDrive.wTorqueCMD             :=  0;
99              xEnableRamping                  :=  FALSE;
100             rSpdCmdPercentage               :=  0;
101             END_IF;
102
103
104         // AHC
105         2:
106         sToDrive.xRun               :=  TRUE;
107         sOpPanelIf.xManualLight     :=  FALSE;
108         sOpPanelIf.xStandbyLight    :=  FALSE;
109         sOpPanelIf.xAHCLight        :=  TRUE;
110         sOpPanelIf.xResetLight      :=  FALSE;
111         sOpPanelIf.xBuzzer          :=  FALSE;
112         sToDrive.wTorqueCMD         :=  3000;
113         DataToHMI.wWinchOpMode      := 2;
114
115         //AHC
116         IF(rAHC_CenterPos = 0.0)THEN
117             rAHC_CenterPos            :=  rWinchPosition;
118             END_IF;
119
120          rOffsetVar                 :=  (rWinchPosition-rAHC_CenterPos);//For test and plotting
121
122          // P-controller output
123          rRCV_AHC                   := (rMP_Pos - (rWinchPosition-rAHC_CenterPos)) * rP_AHC;
124
125          // Max limits from P-controller
126          IF(rRCV_AHC < -100.0)THEN
127              rRCV_AHC               := -100.0;
128          ELSIF(rRCV_AHC > 100.0)THEN
129             rRCV_AHC                := 100.0;
130             END_IF;
131
132          // Joystick control in AHC
133          IF(sOpPanelIf.wJoySw = 1 OR sOpPanelIf.wJoySw = 2)THEN
134             xEnableRamping          := TRUE;
135             rAHC_CenterPos          := 0;
136
137             // If joystick is active, Winch only runs feedforward + joystick
138             // Sets new AHC center pos when joystick is done and puts P-controller back in control
139             IF(sOpPanelIf.wJoySw = 1)THEN
140                 rSpdCmd             := fnFeedForwardWinch(rMP_Spd) + (rSpdCmdPercentage * (sOpPanelIf.rPot1/100) * -1);
141             ELSE
142                 rSpdCmd             := fnFeedForwardWinch(rMP_Spd) + (rSpdCmdPercentage * (sOpPanelIf.rPot1/100));
143                 END_IF;
144
145          ELSE
146             xEnableRamping          := FALSE;
147             // combination of P-controller and feedforward control
148             rSpdCmd                 := rRCV_AHC + fnFeedForwardWinch(rMP_Spd);
149             END_IF;
150
151          // Speed command to drive [0 - 10000]
152          sToDrive.wSpeedCMD         := TO_WORD(TO_UINT(ABS(rSpdCmd * 100)));
153
154          // Direction command to drive
155          if(rSpdCmd < 0)THEN
156              sToDrive.xDir          := FALSE;
157          ELSE
158              sToDrive.xDir          := TRUE;
159              END_IF;
160
161
162         // Fault
```

```
163         3:
164         sToDrive.xRun               :=  FALSE;
165         sOpPanelIf.xManualLight     :=  FALSE;
166         sOpPanelIf.xStandbyLight    :=  FALSE;
167         sOpPanelIf.xAHCLight        :=  FALSE;
168         DataToHMI.wWinchOpMode      := 3;
169
170         // Alarm light blink
171         if(iLightCounter < 50)THEN
172         sOpPanelIf.xResetLight      :=  TRUE;
173         ELSIF (iLightCounter >= 50 AND iLightCounter < 100)THEN
174         sOpPanelIf.xResetLight      :=  FALSE;
175         ELSIF (iLightCounter >= 100)THEN
176         iLightCounter               :=  -1;
177         END_IF;
178         iLightCounter               :=  iLightCounter + 1;
179
180         IF(xLastBuzzer AND NOT(sOpPanelIf.xBuzzer))THEN
181             xAlarmDone              := TRUE;
182             END_IF;
183         IF(NOT(xActivateTOF) AND NOT(xAlarmDone) AND NOT(sOpPanelIf.xBuzzer))THEN
184             xActivateTOF            := TRUE;
185         ELSE
186             xActivateTOF            := FALSE;
187             END_IF;
188         sOpPanelIf.xManualLight     :=  FALSE;
189         sOpPanelIf.xStandbyLight    :=  FALSE;
190
191         // If no drive alarm, go to standby
192         IF(NOT(sFromDrive.xFault OR sFromDrive.xWarning))THEN
193             iOperationMode          :=  0;
194             END_IF;
195
196         xLastBuzzer                 := sOpPanelIf.xBuzzer;
197
198
199         END_CASE;
200
201
202         // Reset winch encoder with given ResetLengthcommand from UI
203         udiEncoderWinch             :=  TO_UDINT(dwRawEncoderWinch);
204         udiResetLenght              :=  fnPosToPulse(DataFromHMI.rResetLength);
205         fbEncoderCount_V1_001(xFirstCycle := xFirstCycle, udiEncoderValue := udiEncoderWinch, xResetLengthCmd :=
            DataFromHMI.xResetWinch, udiResetLengthValue := udiResetLenght, diLastValue := diLastEncWinchVal, xInitDone =>,
            diEncoderValueOut => diWinchPulsCount);
206         diLastEncWinchVal           :=  diWinchPulsCount;
207         fbWinchScale2(diEncoderWinch := diWinchPulsCount, rWinchPos => rWinchPosition);
208
209         // Mapping winch speed and position to UI.
210         DataToHMI.rWinch_Pos         := rWinchPosition;
211         DataToHMI.rWinch_Speed       := sFromDrive.iMotorRPM;
212         DataToHMI.rAHC_CenterPos     := rAHC_CenterPos;
213         xFirstCycle                  := FALSE;
214
215
216         // Ramping joystick in manual mode
217         fbRamping_V1_001(xEnable := xEnableRamping, rInput := sOpPanelIf.rJoy1, rRamptime_sec := rRampTime , uiMaxValue :=
```