Universitetet
i Stavanger

DET TEKNISK-NATURVITENSKAPELIGE FAKULTET

BACHELOROPPGAVE

Studieprogram/spesialisering: Bachelor i ingeniørfag / Automatisering og elektronikkdesign	Vårsemesteret 2021 Åpen
Forfatter(e): Vetle Normann, Ørjan Vier, Emil Wiik Larsen	
Fagansvarlig: Tormod Drengstig Veileder(e): Tormod Drengstig	
Tittel på bacheloroppgaven: Bygging av heismodell og utvikling av tilhørende laboratorieoppgaver Engelsk tittel: Building of elevator model and developing laboratory tasks	
Studiepoeng: 20	
Emneord: Sysmac Studio PLS Heismodell Servomotor Styringsteknikk	Sidetall: 133 (+ 61 i PDF vedlegg) + vedlegg/annet: 22 vedlegg (13 PDF-filer og 9 Sysmac Studio filer) Stavanger 15. mai 2021

Sammendrag

I denne bacheloroppgaven har det blitt bygget en heismodell som skal brukes i undervisning i faget ELE310 Styringsteknikk ved Universitetet i Stavanger. Heismodellen består av en skinneprofil på 2,2 m med en påmontert tannreim fra topp til bunn som drives av en servomotor. På skinneprofilen er det påmontert en vogn som flyttes opp og ned av tannreimen. På vognen er det montert en avstandsensor for måling av høyde, og en servomotor som roterer en utstikkende plastplate. Det er utviklet fire forskjellige prosjekter i Sysmac Studio som benytter seg av heismodellen. Det er også laget forslag til laboppgaver til hvert av de fire prosjektene.

- Det første prosjektet tar for seg hvordan servomotorer kan konfigureres til å gå kontinuerlig mellom to punkter. Hensikten med prosjektet er å vise funksjonaliteten til en servodriver, samt å demonstrere effekten av å begrense hastighet, akselerasjon og/ eller *jerk*.
- Det andre prosjektet tar for seg hvordan en heis med fire etasjer programmeres. Hovedstrukturen av programmet omhandler bruk av ett *Array*, og manipulering av det. *Arrayet* brukes som en etasjeprioriteringsliste som prioriterer etasje-forespørsler fra knapper på *HMI*-skjerm.
- Det tredje prosjektet tar for seg hvordan servomotorer kan styres ved bruk av momentregulering. For å vise hvordan momentregulering fungerer i praksis har det blitt utviklet ett program som måler høyden til personer. Ved å stille en person inntil heismodellen og kjøre servomotoren slik at den utstikkende plastplaten treffer toppen av hodet og stopper når momentet er likt ett referansemoment. Deretter måles høyden til personen ved hjelp av en avstandssensor.
- Det fjerde prosjektet tar utgangspunkt i en 2 ordens sprangrespons. I dette prosjektet er det mulig å spesifisere egenskapene til en overdempet, underdempet eller udempet respons i posisjonen til heisvognen. Hensikten med prosjektet er å vise ett praktisk eksempel knyttet til disse typer responser som er lett å forstå. Det vises også hvordan servomotorene kan styres til å bevege seg etter ett refereansessignal.

Innhold

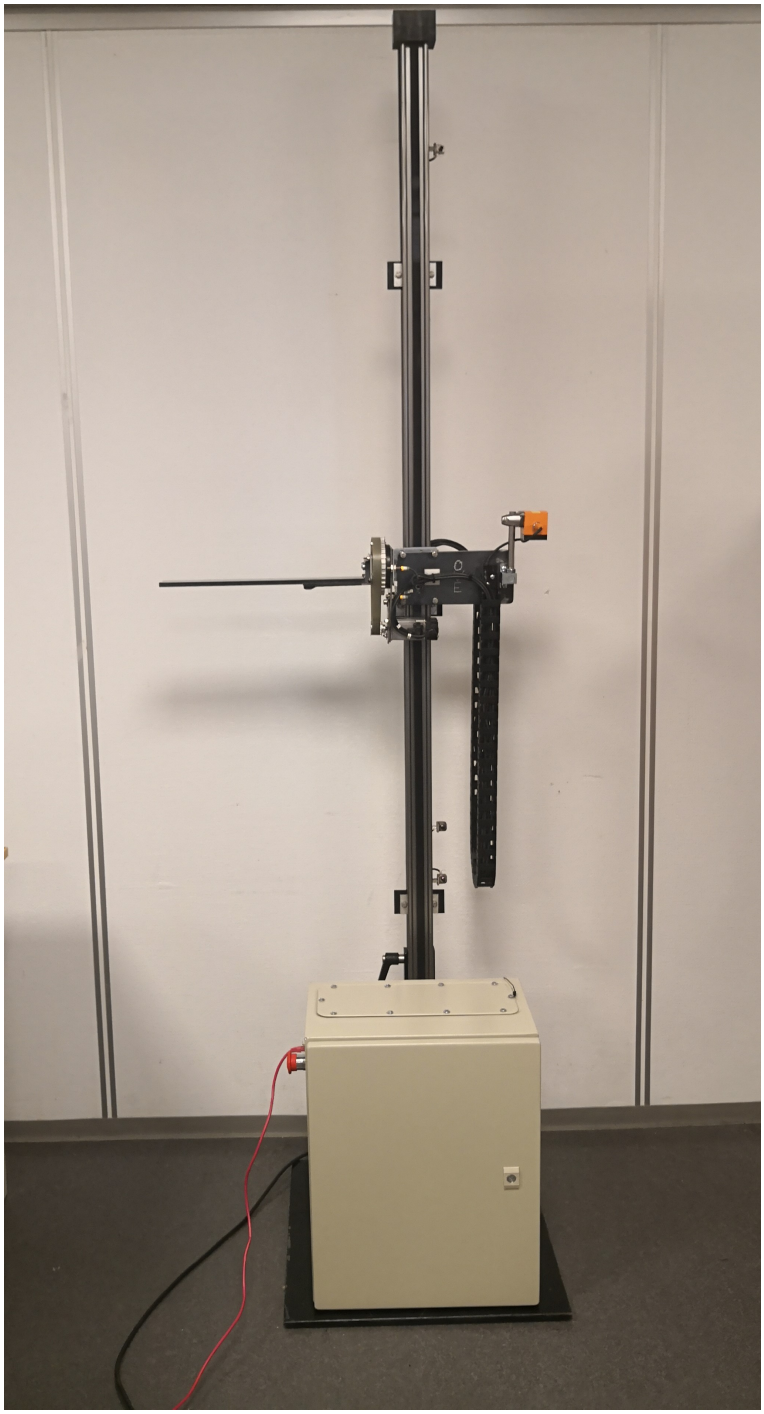
Kapittel 1

Innledning

I faget ELE310 Styringsteknikk ved Universitetet i Stavanger har det til nå vært brukt PLS av typen CJ1M CPU12-ETN fra produsenten Omron. Programvaren som brukes til å programmere denne typen PLS er CX-Programmer. Til høstsemesteret 2021 skal disse bli erstattet av typen NX102-1000 som er en nyere serie fra Omron. Med disse kommer også en ny programvare kalt Sysmac Studio.

Siden Sysmac Studio er nytt på UiS har vi fått i oppgave å sette oss inn i hvordan dette programmet fungerer, og komme med eksempler på hvordan dette kan bli brukt i undervisning. Det er da ønskelig med oppgaver som engasjerer studenten samtidig som det gir en innføring i PLS programmering. Tidligere i faget ELE310 Styringsteknikk har det vært programert med utgangspunkt i tilstandsdiagram hvor hver tilstand ble representert med en *Keep*-blokk. I Sysmac Studio er *Keep*-blokkene fjernet, og dermed vil det være vanskelig å programmere tilstander i ladder-kode.

Med oppaven ble det levert en heismodell og et koblingsbrett med forskjellige komponenter. I figur 1.1 vises den ferdigbygde heismodellen som har blitt bygget i denne bacheloroppgaven. Systemet skulle utbedres slik at det kan brukes på laboratoriet av fremtidige studenter. Hovedkomponentene i heismodellen er to servomotorer med hver sin servodriver. Det har også blitt utarbeidet prosedyrer på hvordan disse konfigureres, implementeres og programmeres i Sysmac Studio.



Figur 1.1: Oversiktsbilde som viser ferdigbygd heismodell med påmontert styreskap. Detaljer om komponenter vises i kapittel 2.

1.1 Sysmac Studio

Sysmac Studio er den nye programvaren som skal brukes i undervisning i faget ELE310 Styringsteknikk. I Sysmac Studio er det mulig å programmere i ladder og i strukturert tekst. Det grafiske programmeringspråket SFC (*Sequential Function Chart*) fra CX-Programmer er fjernet i Sysmac Studio.

På grunn av at *Keep*-blokkene ikke lenger finnes i Sysmac Studio, så har vi valgt å programmere tilstander ved hjelp av *case*-struktur i strukturert tekst. Da blir strukturen i programmet lik som med *Keep*-blokker i CX-programmer, og det er mulig å tegne tilstandsdiagram slik at strukturen blir oversiktlig.

I Sysmac Studio brukes det *motion control* funksjonsblokker for å kommunisere med servodriverne. *Motion control* funksjonsblokkene får instruksjoner fra tilstander programmert i *case*-struktur. *Motion control* funksjonsblokkene brukt i bacheloroppgaven blir beskrevet i vedlegg D.2.

PLS bruker *EtherCat*-protokoll til å kommunisere med servodriverne og *EtherCat*-modul i styreskap. *EtherCat*-nettverket kobles opp som en seriekoblet buss med *Ethernet*-kabler. Datamaskin, PLS og *HMI*-skjerm kommuniserer med *Ethernet*-protokoll, og er koblet sammen med *Ethernet*-kabler via en *Ethernet-svitsj*. Hvordan *Ethernet*- og *EtherCat*-nettverket er koblet vises med koblingskjema i vedlegg C.3.

Det er laget en prosedyre for oppkobling og konfigurering av heismodellen i Sysmac Studio (se vedlegg D.1). Prosedyren inneholder detaljer om hvordan følgende gjøres i Sysmac Studio:

- Hvordan koble til datamaskin og opprette ett nytt prosjekt.
- Hvordan *EtherCat*-nettverket konfigureres.
- Hvordan *EtherCat*-modul med påmontert AD-kort, servodrivere og servomotorer konfigureres.
- Eksempel på hvordan det kan lages ett enkelt program for å styre servomotor 1 og servomotor 2.

1.2 Oppgavebeskrivelse

IDE har en 2 meter høy heismodell som styres av en kraftig servomotor. Selve heisen består av en liten stålplate som kan roteres individuelt av en liten servomotor. Du kan studere modellen inne på rom E-457. Modellen er pr. idag ikke i drift.

Oppgaven består i å montere riggen sammen og gjøre nødvendig sammenkopling av alt medfølgende utstyr. Dette inkluderer også å montere alt det elektriske utstyret i et styreskap. Deretter skal det utvikles flere alternative lab-oppgaver som kan brukes i faget ELE310 Styringsteknikk. Disse oppgavene skal blant annet gjøre studentene kjent med funksjonaliteten til motordrivere. PLS-programmeringen skal foregå i Sysmac Studio (og ikke i CX-Programmer). En HMI-skjerm skal programmeres slik at heisen kan opereres fra denne. Dersom det blir tid skal det også undersøkes hvorvidt heisen kan opereres i vannrett posisjon. Dette vil i så fall åpne opp for alternative anvendelser.

Studenter med fagbrev vil bli prioritert.

1.3 Rapportens struktur

Rapporten er strukturert på følgende måte:

1. I kapittel 2 presenteres utstyret som er brukt i oppgaven.
2. I kapittel 3, 4, 5 og 6 dokumenteres prosjektene som er laget til heismodellen.
3. I kapittel 7 er konklusjonen av bacheloroppgaven. Kapittelet inneholder også forslag til forbedring og videreutvikling av heismodellen og tilhørende programmer.
4. Vedlegg A inneholder en timeliste.
5. Vedlegg B inneholder programfilene utarbeidet i prosjektene.
6. Vedlegg C inneholder teknisk dokumentasjon knyttet til heismodellen.
7. Vedlegg D inneholder prosedyrer og bruksanvisninger.
8. Vedlegg E inneholder programfiler til laboppgaver utarbeidet i prosjektene.

Kapittel 2

Utstyr

Dette kapittelet presenterer arbeidet som har blitt gjort i bygging av heismodellen. Det inneholder også en oversikt over alle komponenter som er en del av heismodellen med beskrivelse av hva de brukes til. Utstyret deles inn i to hovedgrupper som vist i listen under.

1. Utstyret montert på PLS-stasjon, som brukes i undervisning ved UiS som vist i kapittel 2.1.
2. Utstyret montert på heismodellen som vist i kapittel 2.2.

Tabell 2.1 gir en oversikt over hovedkomponentene som brukes i heismodellen. Det vises også til referanser hvor datablader og annen informasjon tilhørende komponentene kan finnes.

Tabell 2.1: Oversikt over utstyr brukt i prosjektetene med referanse til datablad og annen informasjon tilhørende komponentene.

Beskrivelse	Komponentnavn	Referanse
PLS	NX102-1000	[1]
HMI-skjerm	NX102-1000	[2]
Strømforsyning	S8VS-06024B	[3]
Ethernet-svitsj	W4S1-05B	[4]
EtherCat-modul	NX-ECC203	[5]
AD-kort	NX-AD3604	[6]
Servodriver 100W	R88D-KN01H-ECT	[7]
Servodriver 400W	R88D-KN04H-ECT	[8]
Servomotor 100W	R88M-K10030H-S2	[9]
Servomotor 400W	R88M-K40030H-S2	[10]
Avstandssensor	O1D15	[11]
Nærhets-sensorer	E2A-S	[12]

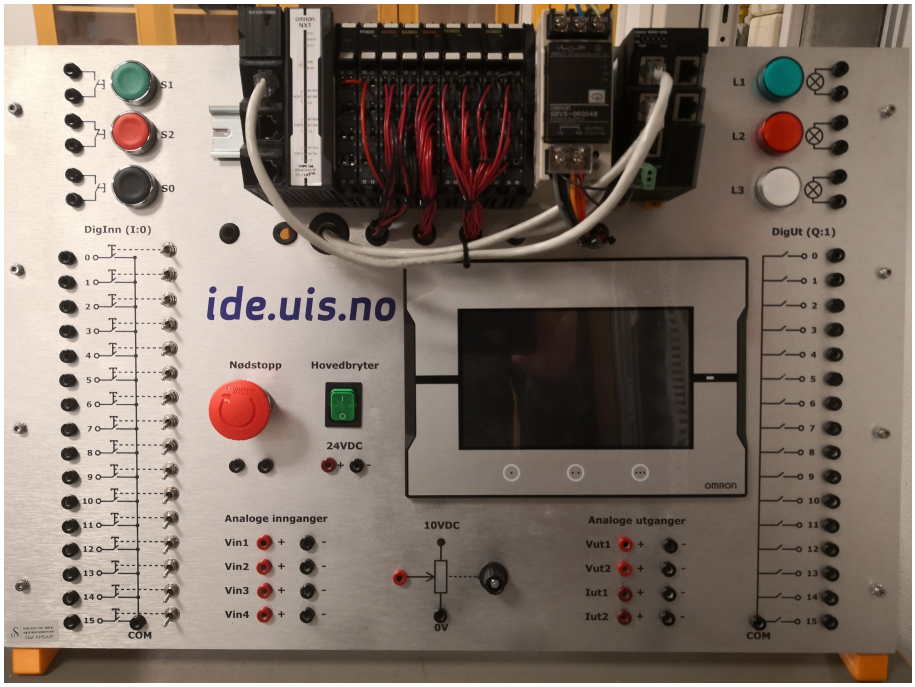
2.1 PLS-stasjon

I figur 2.1 vises PLS-stasjon som skal brukes på lab i faget ELE310 Styringsteknikk. Utstyret som er montert på stasjonen er vist i punktlisten under.

- PLS med *IO*-kort.
- *HMI*-skjerm.
- *Ethernet*-svitsj.
- Strømforsyning.
- Brytere, tilkoblingspunkter, knapper, lys, nødstopp og potmeter.

I prosjektene videre i rapporten er det kun PLS, *HMI*-skjerm, *Ethernet*-svitsj og strømforsyning som blir brukt.

2.1 PLS-stasjon

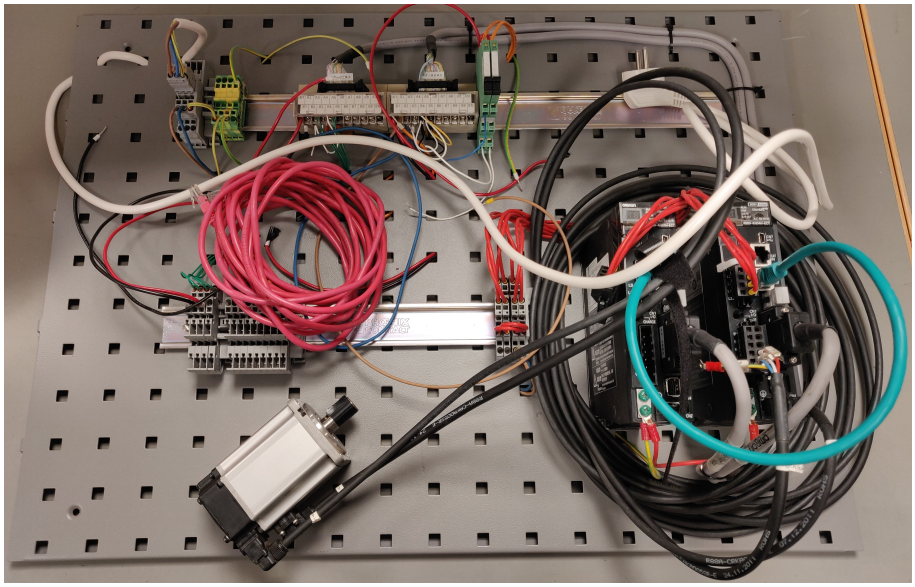


Figur 2.1: Oversiktsbilde av PLS-stasjonen som skal brukes i faget ELE310 Styringsteknikk ved UiS.

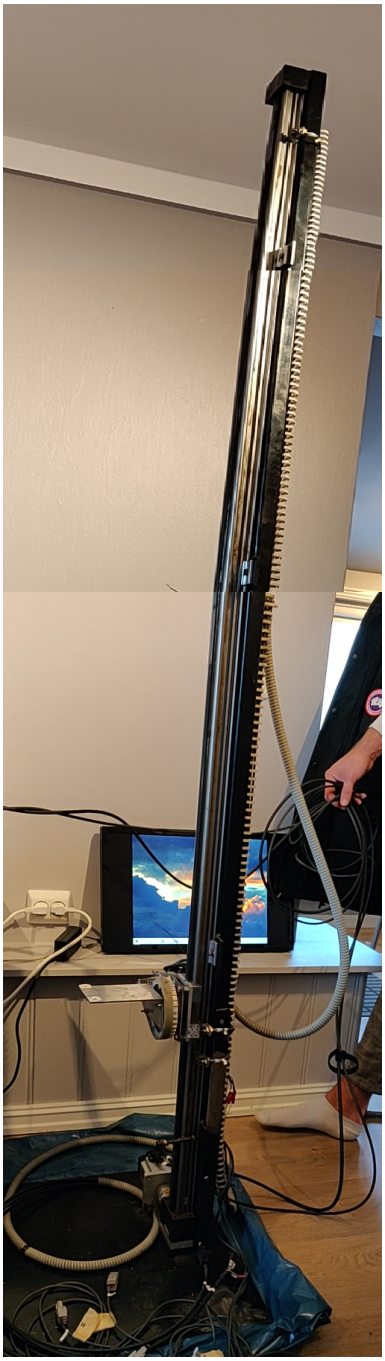
2.2 Heismodell

2.2.1 Utgangspunkt

I figur 2.2 og figur 2.3 vises utstyret slik det fremsto når vi startet prosjektet. Heismodellen skal brukes til undervisning i faget ELE310 Styringsteknikk. Det er derfor viktig å ha et system som er sikkerhetsmessig bedre enn den åpne installasjonen vist i figur 2.2. Videre i rapporten vises det at komponentene ble bygget inn i ett styreskap montert på heismodellen.



Figur 2.2: Utstyret slik det fremsto når vi startet prosjektet.



Figur 2.3: Heismodellen slik vi fikk den levert.

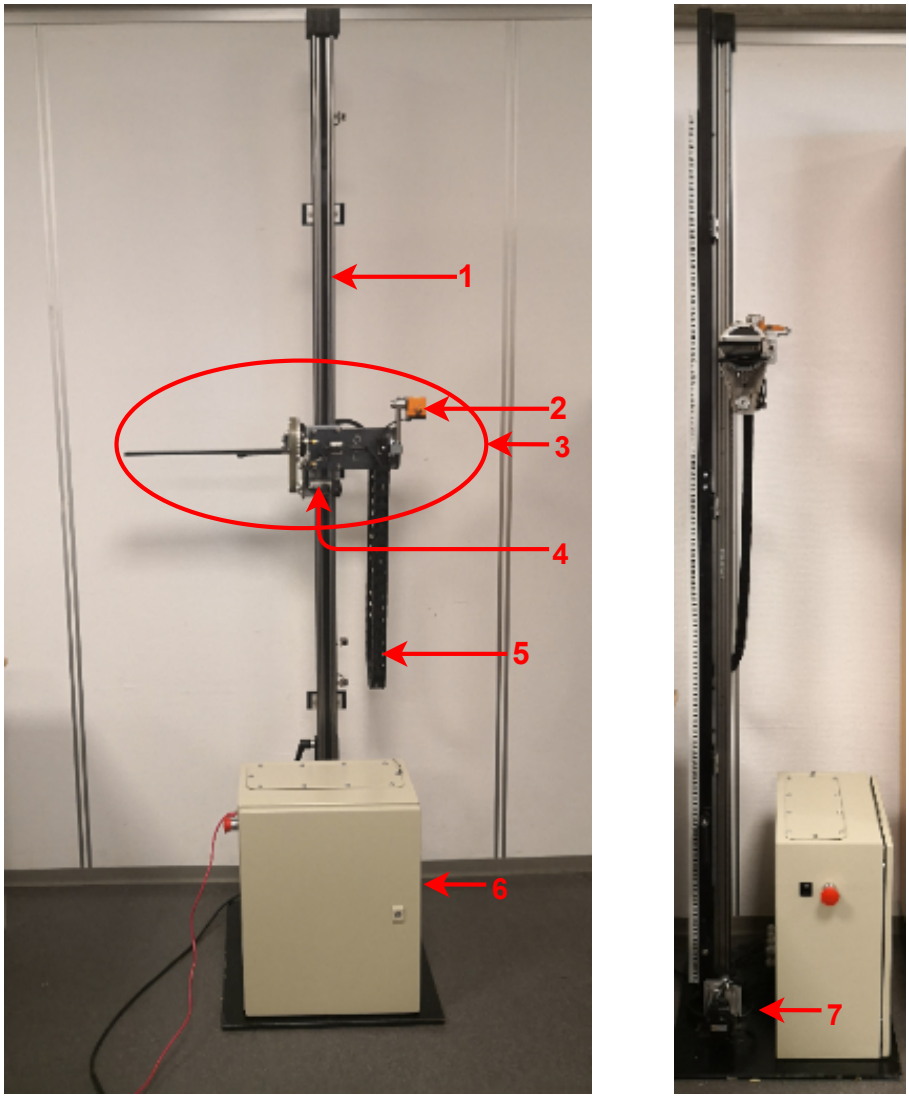
2.2.2 Ferdigstilt heismodell

Følgende endringer har blitt gjort på heismodellen iløpet av bacheloroppgaven (markeringene som refereres til i punktlisten under vises i figur 2.4):

- Kabelføring for kablene som går fra styreskap til den bevegelige heisvognen har blitt byttet ut. Det har blitt montert ny fast kabelbane på baksiden av skinneprofilen. Det har også blitt montert en bevegelig kabelbane som er festet til heisvognen som vist i markering 5.
- Heisvognen har blitt bygget om slik at den roterende utstikkende plastplaten stikker ut til siden som vist i figur 2.4. Det har også blitt montert en større metall brakett som en del av heisvognen slik at avstandssenoren (vist i markering 2) og den bevegelige kabelbanen (vist i markering 5) kunne festes.
- Det har blitt bygget et styreskap (vist i markering 6) som har blitt festet til heismodellens bunnplate. Styreskapet inneholder alle nødvendige komponenter for å styre servomotorene.

Listen under beskriver markeringene vist i figur 2.4.

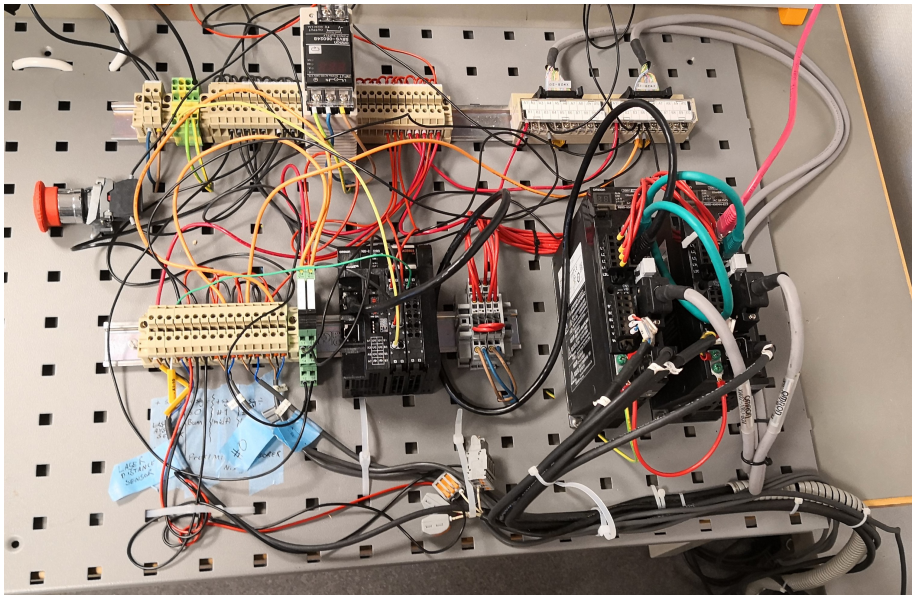
1. Markeringen viser skinneprofil og tannreim.
2. Avstandsensor.
3. Viser til den røde sirkelen som inneholder heisvognen.
4. Servomotor 2.
5. Kabelføring for bevegelige kabler som går fra styreskap til heisvogn.
6. Styreskap.
7. Servomotor 1.



Figur 2.4: Bildet viser ferdigstilt heismodell med styreskap. Markeringene på bildet beskrives i punktlisten over.

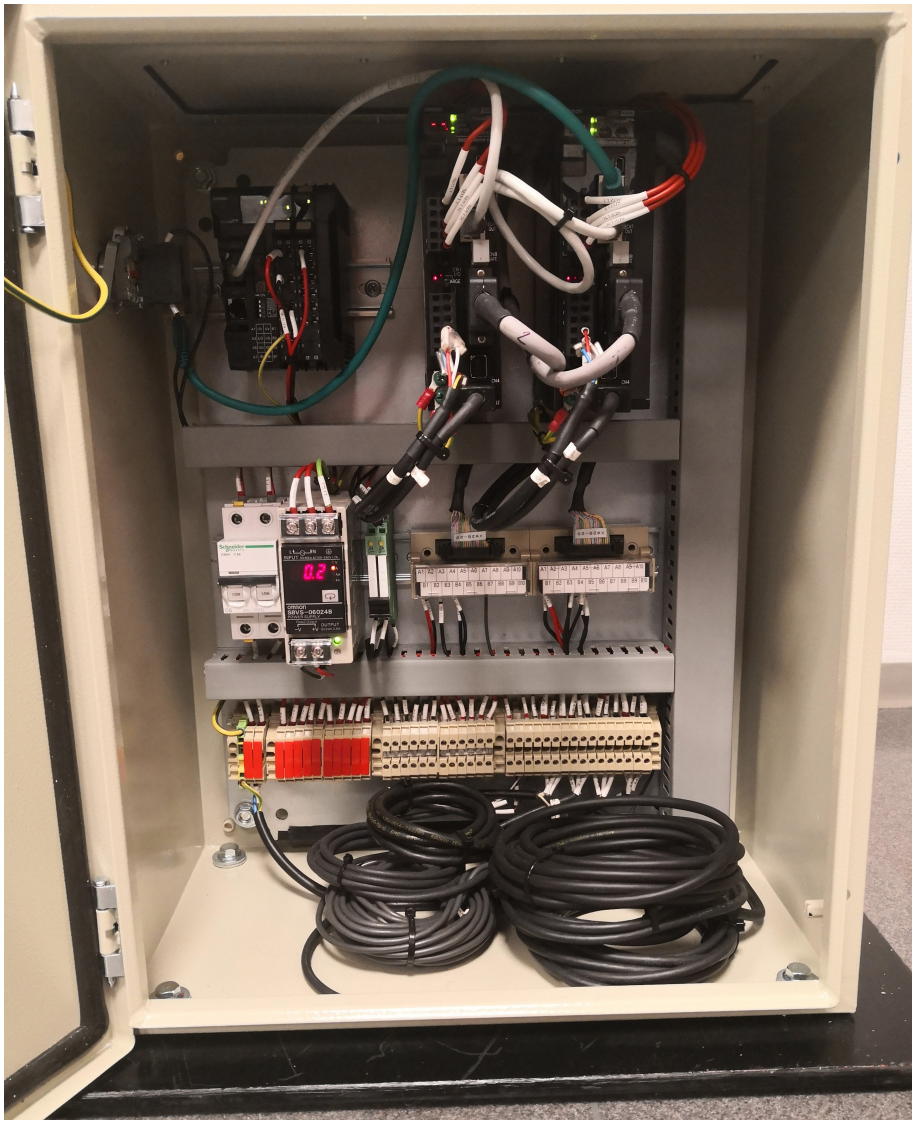
2.2.3 Styreskap

I figur 2.5 vises det hvordan komponentene var koblet opp før de ble montert i styreskapet. Grunnen til at det ble koblet opp slik var for at det da kunne gjøres modifikasjoner relativt raskt og enkelt.



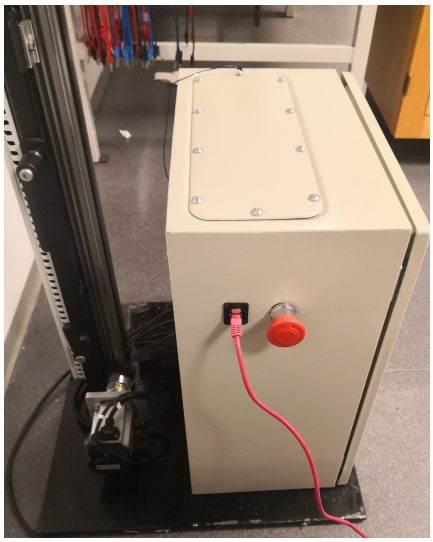
Figur 2.5: Komponentene ble først koblet opp på et åpent brett for å lettere kunne gjøre modifikasjoner.

For å få et mer oversiktlig og ryddig system har alle komponenter blitt montert i ett styreskap som vist i figur 2.6. I styreskapet er det montert DIN-skinner for å feste komponentene. DIN-skinnene er festet med selvborende skruer i bakplata til styreskapet. Servodriverne er montert direkte på bakplata. Styreskapet er montert fast i heismodellens bunnplate med 4 bolter. Beskrivelse av alle komponentene vist i figuren av det ferdig bygde styreskapet gis videre i kapittelet. Vedlegg C inneholder teknisk dokumentasjon av styreskapet. Vedlegg C består av følgende dokumentasjon: Hovedstrømskjema styreskap (230 VAC) C.1, styrestrømskjema styreskap (24 VDC) C.2, *Ethernet/ EtherCat* oversikt C.3, komponent oppsett styreskap C.4, rekkeklemmetabbell styreskap C.5 og komponentnavn forkortelser styreskap C.6.



Figur 2.6: Bildet viser det ferdigstilte styreskapet som er montert på som en del av heismodellen.

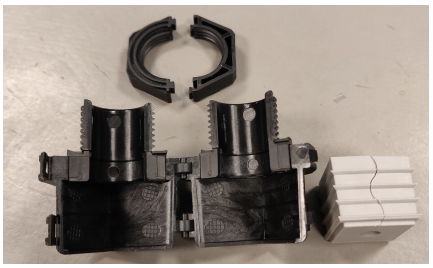
Som vist i figur 2.7a er det montert enn nødstopp-bryter og en RJ-45 kontakt som brukes for kommunikasjon mellom PLS-stasjonen og heismodellen. I figur 2.7b vises kabelgjennomføringene fra styreskapet. Alle kablene med unntak av strømtilførsels-kabelen går inn i den grå kabelføringen på baksiden av skinneprofilen som vist til høyre i figur 2.7b. To av kabelføringene er av en type som kan deles i to som vist i figur 2.7c og figur 2.7d. Disse gjennomføringene er brukt fordi tilførselskablene som går fra styreskapet til begge servomotorene er støpt i endepunktene.



(a) Nødstopp-bryter og RJ-45 kontakt for *EtherCat*-tilkobling.

(b) Kabelgjennomføring til styreskap.

(c) Gjennomføringen brukt til servomotorenes tilførselskabel.

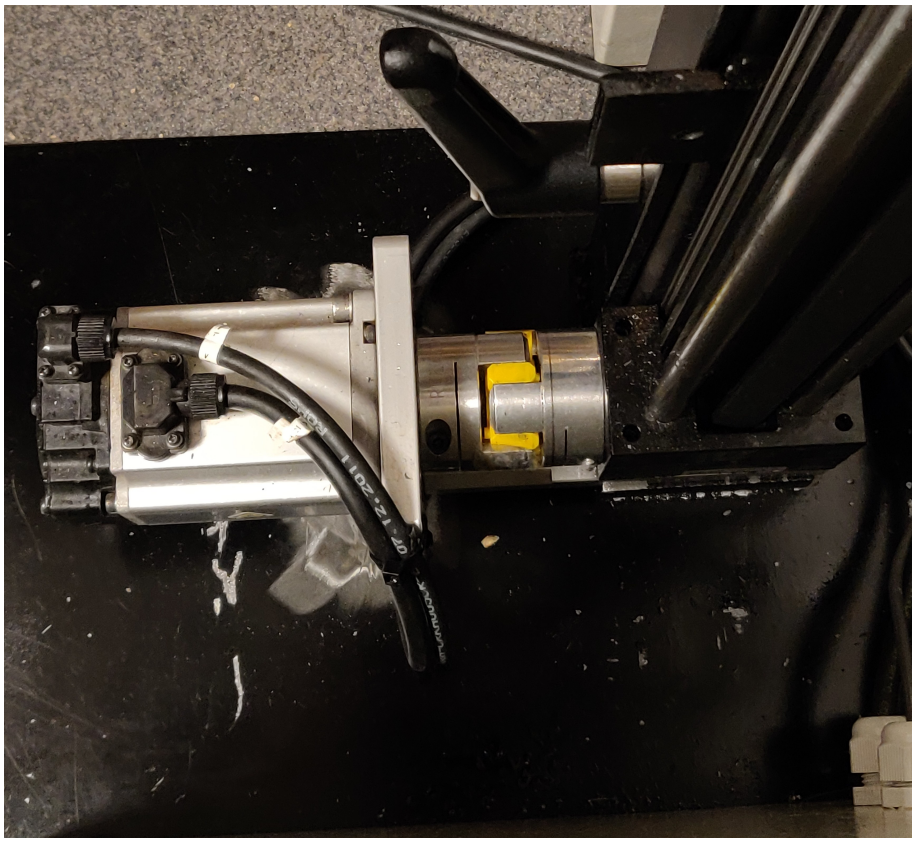


(d) Gjennomføringen vist i figur 2.7c delt i to.

Figur 2.7: Figuren viser flere forskjellige detaljer på heismodellen.

2.2.4 Servomotor 1

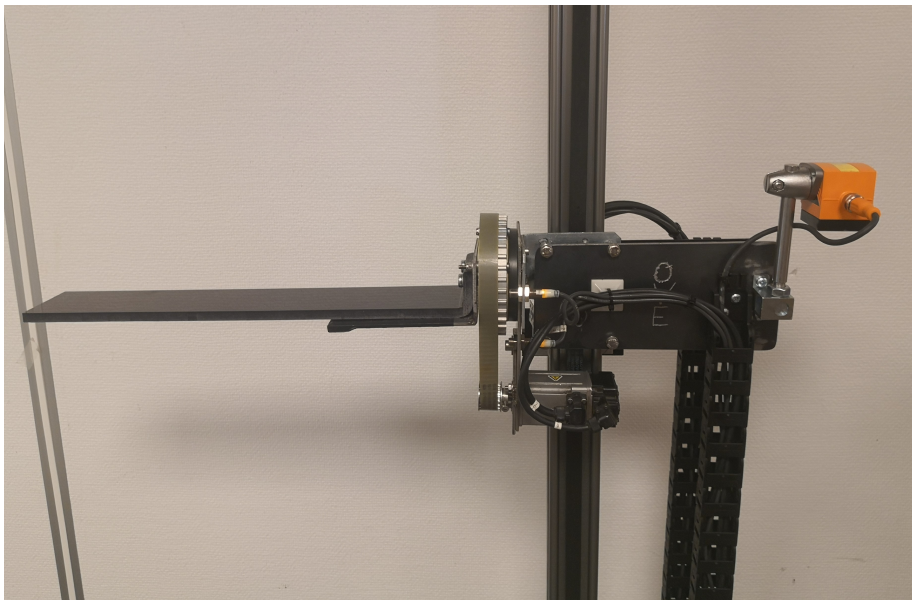
Servomotoren vist i figur 2.8 er produsert av *Omron*, og modellen heter *R88M-K40030H-S2* [10]. Servomotoren drives av en effekt på opp til 400W. Denne servomotoren blir videre omtalt som servomotor 1 i rapporten. Funksjonen til servomotor 1 er å drive tannreimen som beveger heisvognen i vertikal retning. Det svarte huset som er montert på servomotoren helt til venstre vist i figur 2.8 er servomotorens enkoder. Enkoderen er på 20 bit som tilsvarer en oppløsning på $2^{20\,bit} = 1048576$ pulser per omdreining. Hvordan servomotoren konfigureres i Sysmac Studio vises i vedlegg D.1.



Figur 2.8: Oversiktsbilde av servomotor 1 med enkoder.

2.2.5 Servomotor 2 og heisvogn

Servomotoren som er montert på selve heisvognen vist i figur 2.9 er produsert av *Omron*, og modellen heter *R88M-K10030H-S2* [9]. Servomotoren drives av en effekt på opp til 100W. Servomotoren blir videre omtalt som servomotor 2 i rapporten. Funksjonen til servomotor 2 er å rotere den utstikkende plastplaten som vises til venstre i figuren under. Slik som servomotor 1 har også servomotor 2 en tilsvarende enkoder på 20bit. Hvordan servomotoren konfigureres i Sysmac Studio vises i vedlegg D.1.

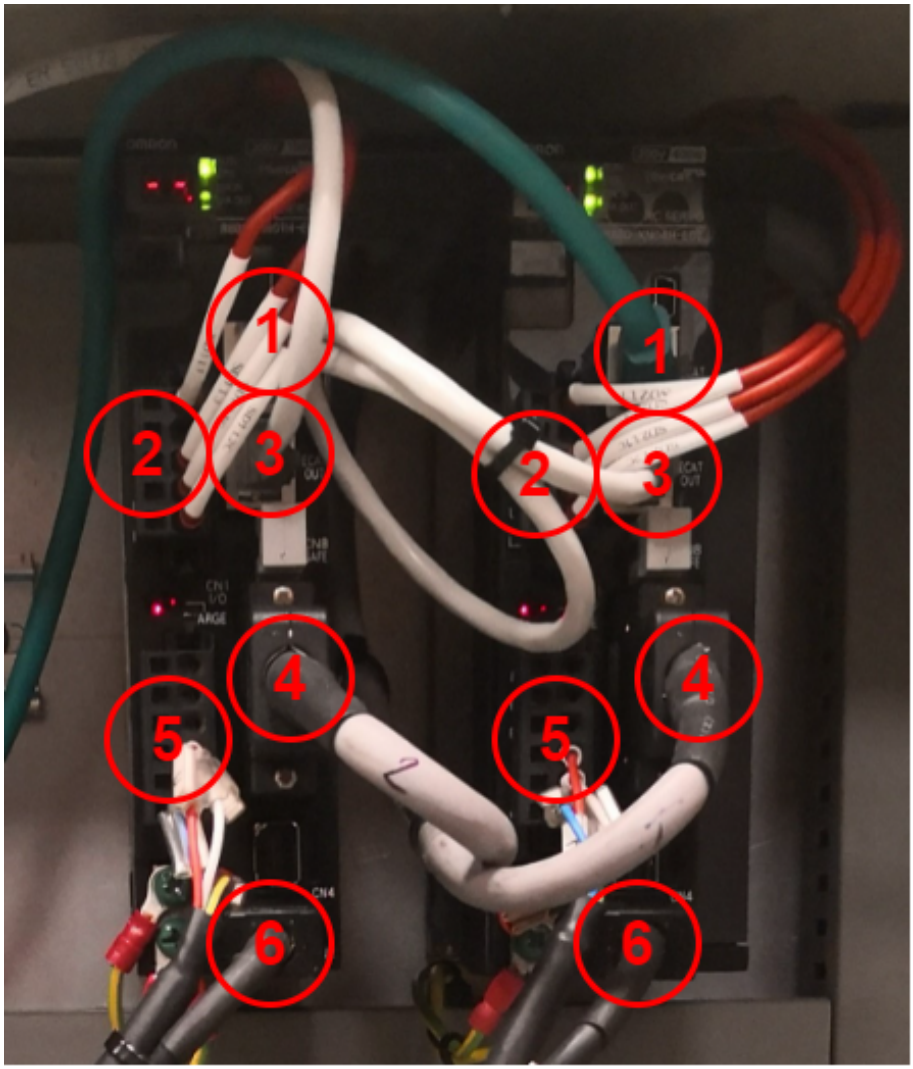


Figur 2.9: I figuren vises heisvognen som beveges av servomotor 1. På selve heisvognen er servomotor 2 og avstandssensoren montert.

2.2.6 Servodriver 1 og servodriver 2

I figur 2.10 viser heismodellens to servodrivere som regulerer og styrer servomotorene. Servodriverne kommuniserer med PLS via *EtherCat*-nettverket. Videre i rapporten blir servodriverne omtalt som servodriver 1 (til høyre i figuren) og servodriver 2 (til venstre i figuren), henholdsvis er de tilknyttet servomotor 1 og servomotor 2. Tilkoblingene til servodriverne er oppgitt i listen under. Nummereringen i listen samsvarer med nummereringen i figur 2.10.

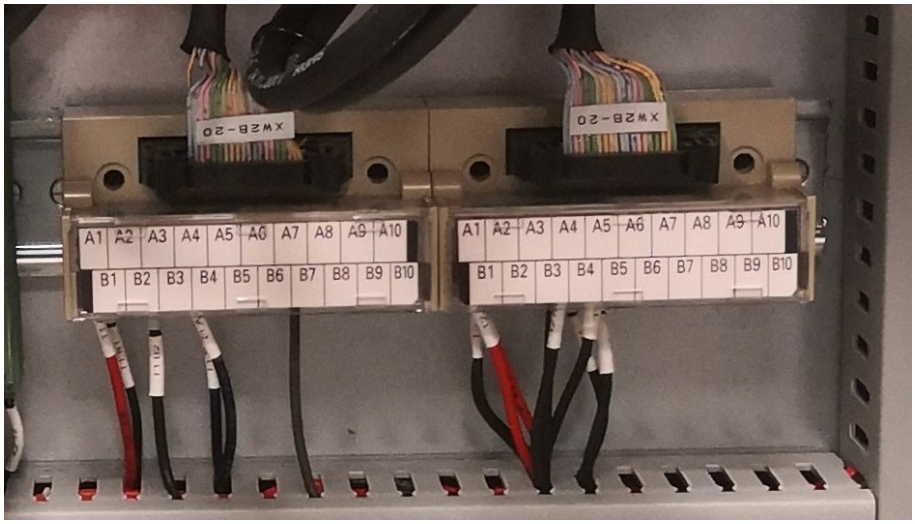
1. *EtherCat-In*: RJ-45 port for tilkobling av *EtherCat*-nettverk.
2. 230 V driftspenning tilført servodriverne.
3. *EtherCat-Out*: RJ-45 port for tilkobling av *EtherCat*-nettverk.
4. Spenning fra servodriver til servomotor.
5. Tilkobling til eksterne inn- og utganger (terminalene for tilkobling av de eksterne inn- og utgangene blir beskrevet i kapittel 2.2.7).
6. Tilkobling av enkoder som er montert på servomotor.



Figur 2.10: I figuren vises servodriver 1 (til høyre i figuren) og servodriver 2 (til venstre i figuren). Servodriver 1 er tilknyttet servomotor 1, og servodriver 2 er tilknyttet servomotor 2. Nummereringen i figuren samsvarer med punktlisten vist på foregående side.

2.2.7 Terminaler til servodrivere

I figur 2.11 vises terminalene som er koblet til servodriverne via markering 4 i figur 2.10. Terminalen til høyre i figuren er tilkoblet servodriver 1 og terminalen til venstre i figuren er tilkoblet servodriver 2. Terminalene er tilkoblingspunktet for forskjellige inn- og utganger til servodriverne. Nødstopp og nærhets-sensorer er koblet til terminalene.



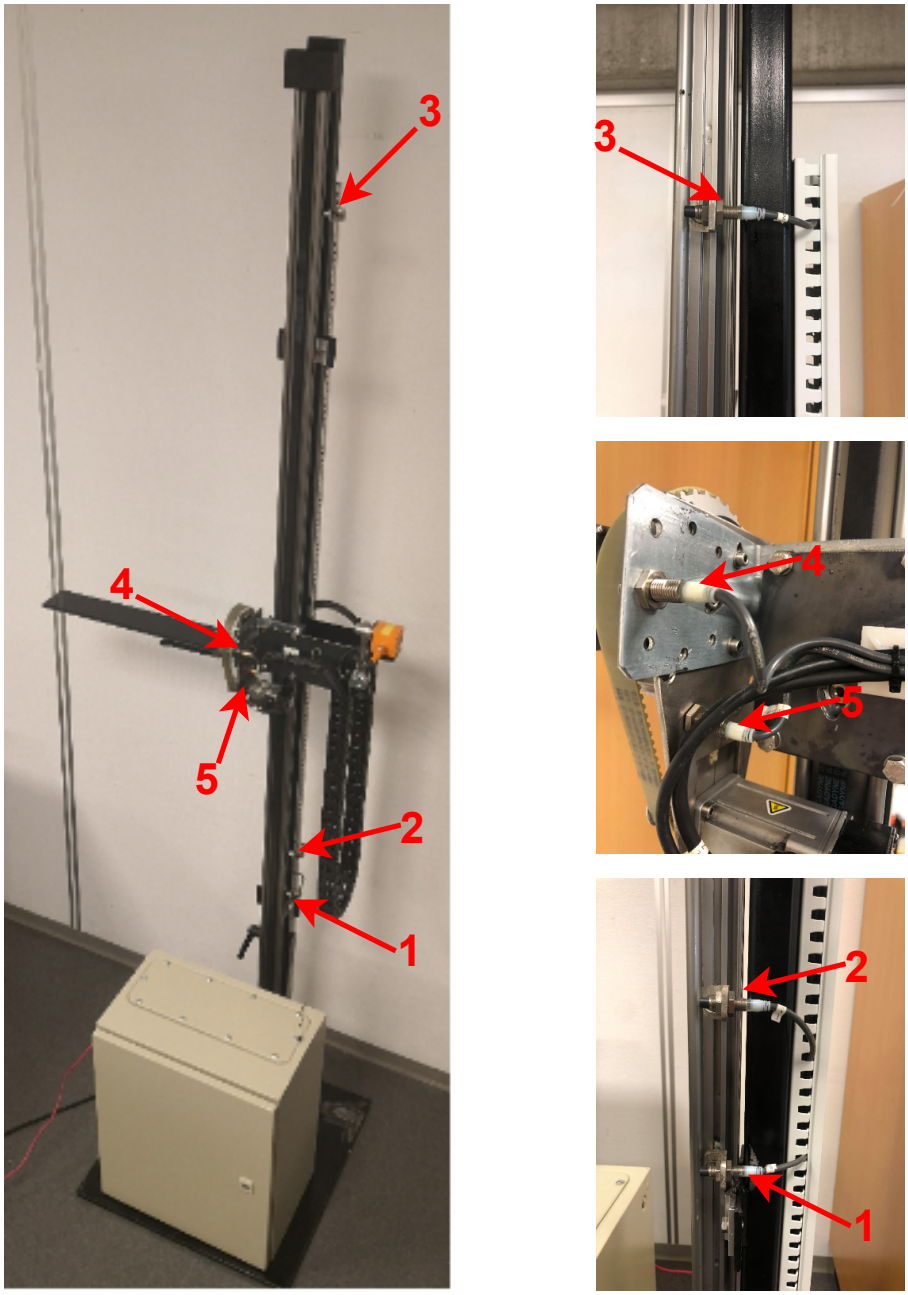
Figur 2.11: Oversiktsbilde av terminalene knyttet til servodriverne.

2.2.8 Nærhets-sensorer

Som vist i figur 2.12 er det montert fem nærhets-sensorer [12] på heismodellen. Markeringene i figuren tilsvarer nærhets-sensorenes nummer. Nærhetssensor 1 og 3 (som er konfigurert som endestopper for heisvognen drevet av servomotor 1) er koblet via kontaktor 1 og kontaktor 2 for at kretsen skal brytes når heisvognen detekteres. Kontaktorene og de resterende sensorene er koblet til terminalene beskrevet i kapittel 2.2.7.

Tabell 2.2: Tabellen gir en oversikt over nærhets-sensorene montert på heismodellen (nærhets-sesnor 2 og 5 som ikke er i bruk kan brukes ved alternativt oppsett av kalibreringsprosedyre).

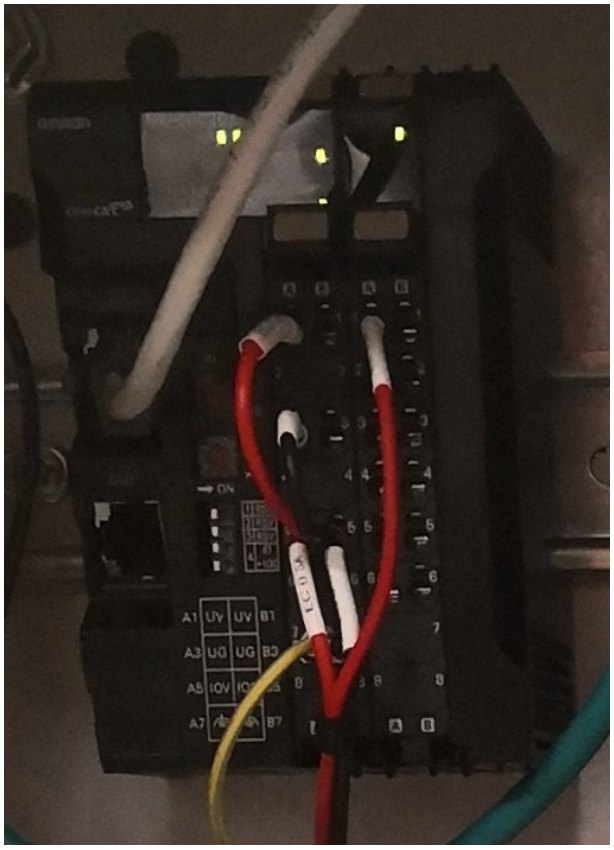
Navn	**Brukes til**
Nærhets-sensor 1	Endestopp/ kalibrerings-referanse
Nærhets-sensor 2	Ikke i bruk
Nærhets-sensor 3	Endestopp/ kalibrerings-referanse
Nærhets-sensor 4	Kalibreringsreferanse
Nærhets-sensor 5	Ikke i bruk



Figur 2.12: Figuren viser hvor på heismodellen nærhets-sensorerene er montert. De nummererte pilene referer til nærhets-sesnorenes nummerering.

2.2.9 *EtherCat*-modul med AD-kort

EtherCat-modulen [5] med påmontert AD-kort [6] er vist i figur 2.13. *EtherCat*-modulen konverterer ett 12-bit digitalsignal med oppløsning i området $[0, 4096]$ fra AD-kortet til *EtherCat*-protokoll. AD-kortet tar i mot en analogspenning i området $[0, 10]V$ fra avstandssensoren. Hvordan *EtherCat*-modulen og AD-kortet konfigureres i Sysmac Studio vises i vedlegg D.1.



Figur 2.13: Bildet viser *EtherCat*-modulen med påmontert AD-kort. AD-kortet mottar en analogtspenning i området $[0, 10]V$ fra avstandssensoren og konverterer det til ett 12-bit digitalt signal med oppløsning $[0, 4096]$. *EtherCat*-Modulen kommuniserer data mellom AD-kortet og PLS.

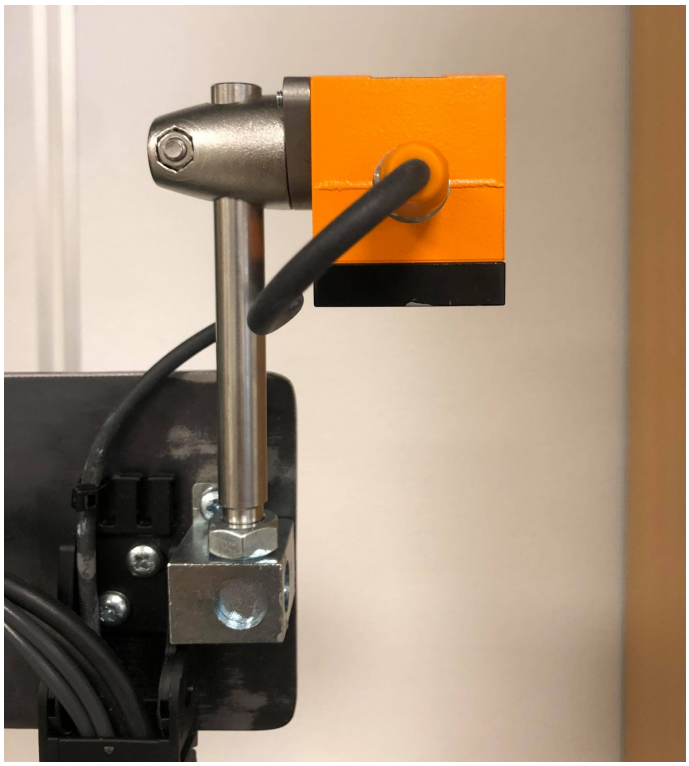
2.2.10 Sikring, strømforsyning og kontaktorer

Styreskapet har en hovedsikring på 6 ampere. Hovedsikringen brukes som en *service*-bryter for heismodellen. Styreskapet har en 24 Volt strømforsyning som supplerer *EtherCat*-Modulen, eksterne sensorer og brytere med spenning. For mer informasjon om oppkobling se hovestrømskjema C.1 og styrestrømskjema C.2. To kontaktorer endrer logikken på nærhets/ endestopp sensorene fra normalt åpen til normalt lukket. Dette er for å øke sikkerheten hvis det skulle oppstå brudd i sensorkretsen.

Figur 2.14: Fra venstre: Hovedsikring for styreskapet, strømforsyning og kontaktorer.

2.2.11 Avstandssensor

Avstandssensoren vist i figur 2.15 er produsert av IFM, og modellen heter *O1D15*[11]. Sensoren er montert på heisvognen. En skjerm på avstandssensoren viser målt avstand i $[mm]$. Avstandssensoren gir ut tilsvarende et spenningssignal i området $[0, 10]V$ som representerer avstander i området $[0, 2500]mm$. Det har blitt valgt å konfigurere avstandsensoren til å måle avstander i området $[0, 2500]mm$ fordi heismodellen er i underkant av 2500 mm høy. Servomotorenes enkodere kan også måle avstand, men vi har valgt å implementere avstandsensoren i heismodellen for å kunne vise hvordan en analog avstandsensor blir implementert via en ekstern *EtherCat*-modul tilkoblet ett AD-kort. I kapittel 5.2 beskrives det mer utfyllende hvordan sensoren er konfigurert.



Figur 2.15: I figuren vises det bilde av avstandssensoren, som er montert på heisvognen.

Kapittel 3

Bevegelses-begrensning av servomotorer

I denne videoen blir dette prosjektet demonstrert.

3.1 Motivasjon for prosjekt

I dette kapittelet vises effekten av å begrense hastighet, akselerasjon og *jerk* når servomotorene er i bevegelse. Heretter refereres begrensning av hastighet, akselerasjon og *jerk* til som bevegelses-begrensning. Ved å la servomotoren gå kontinuerlig frem og tilbake mellom to posisjoner, så vil effekten av bevegelses-begrensning kunne observeres visuelt og grafisk. Motivasjonen for prosjektet i dette kapittelet er å vise studentene hvordan det kan lages forskjellige posisjonprofiler for servomotorene ved forskjellige bevegelses-begrensninger.

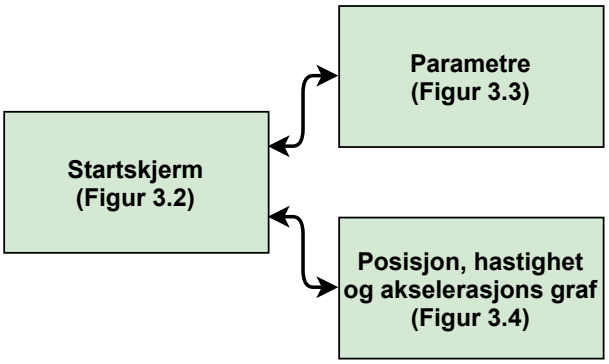
I presentasjonen av prosjektet har vi delt det opp i følgende deler:

1. Presentasjon av hvordan programmet brukes, som vist i kapittel 3.2
2. Tilstandsdiagram for programmet, som vist i kapittel 3.3
3. Presentasjon av sammenhengen mellom posisjon, hastighet, akselerasjon og *jerk*, og metode som er brukt til å beregne heisvognens akselerasjon og *jerk*, som vist i kapittel 3.4.
4. Grafisk fremstilling av forskjellige posisjonsprofiler, med forskjellig bevegelses-begrensninger, som vist kapittel 3.5
5. Forslag til laboppgave ved bruk av programmet vist i gjeldende kapittel, som vist i kapittel 3.6

3.2 Programfunksjonalitet

Hovedfunksjonaliteten til programmet som presenteres i dette kapittelet er at servomotor 1 og servomotor 2 kontinuerlig beveges mellom posisjon = 0 og posisjon = 1000, henholdsvis i enhetene $[mm]$ og $[grader]$. Intervallet $[0, 1000]$ $[mm]/$ $[grader]$ ble valgt fordi 1000 mm er en fornuftig lengde i forhold til heismodellens fysiske størrelse, og 1000 *grader* ble valgt for at servomotorene skal beveges i samme intervall. For å realisere ønsket oppførsel brukes `MC_Move`-funksjonsblokker som vist i vedlegg D.2 for å styre servomotorene. Ved å kun spesifisere begrensning for hastighet vil bevegelsene til servomotorene være relativt brå når servomotorenes dreieretning endres. Dette er fordi det ikke er spesifisert begrensning på akselerasjon eller *jerk*, og dermed endres servomotorens hastighet seg svært hurtig. Ved å spesifisere begrensning for hastighet, akselerasjon og/ eller *jerk* vil servomotorenes bevegelser være mye jevnere når dreieretningen endres. Dette er fordi endringsraten til posisjon, hastighet og/ eller akselerasjon er begrenset.

I figur 3.1 vises det hvordan skjermene i *HMI* er koblet sammen. *HMI*-skjermen gjør det enklere og mer oversiktlig for bruker å betjene programmet.

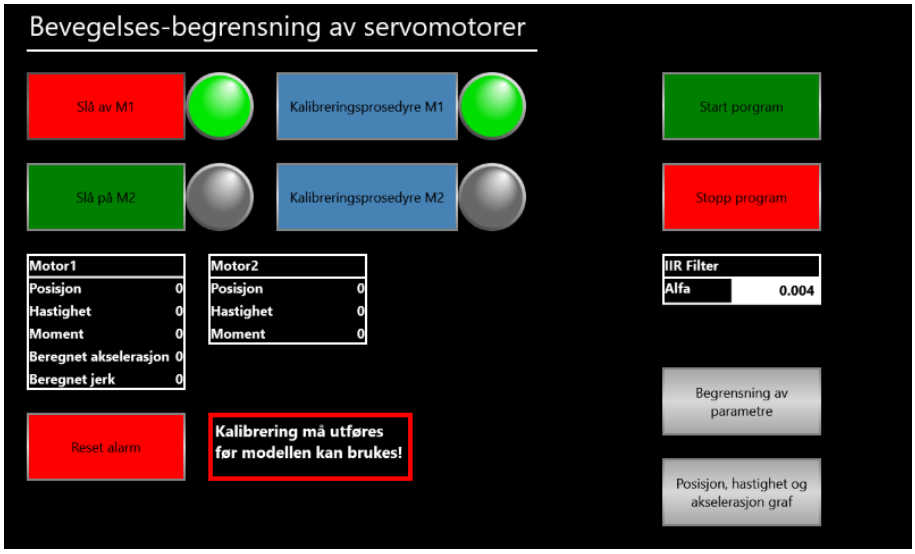


Figur 3.1: Oversiktsbilde viser hvordan skjermene i *HMI* er koblet sammen.

I listen under forklares det hvordan programmet i PLS og *HMI* brukes:

1. I figur 3.2 vises hjemskjermen til programmet. Den røde knappen merket **Slå av M1** slår av/ på servodriver 1, og den grønne lampen til høyre for knappen viser om servodriveren er av/ på (her er servodriver 1 påslått). Den grønne knappen merket **Slå på M2** slår på/ av servodriver 2, og den grå lampen til høyre for knappen viser om servodriveren er av/ på (her er servodriver 2 avslått). Av/ på-knappene endrer farge og tekst ettersom servodriverne og servomotorene er avslått/ påslått.

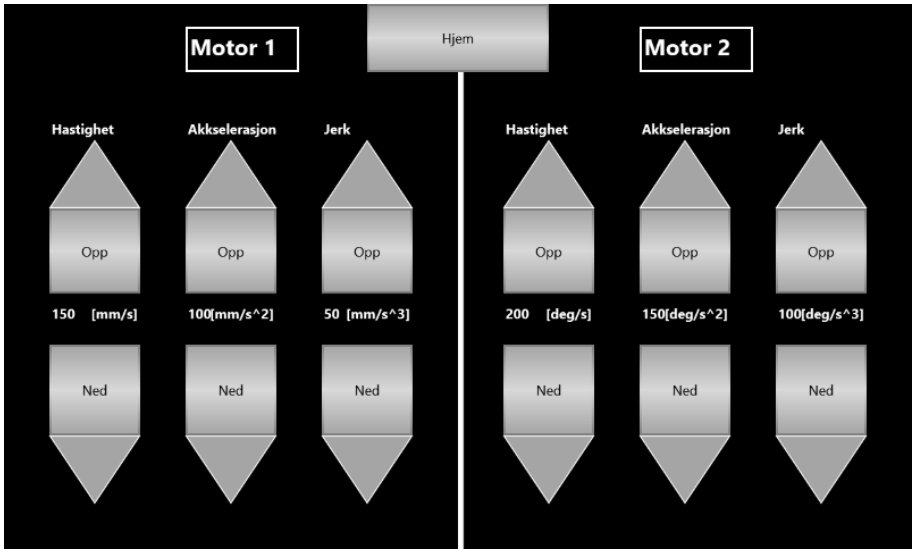
2. De blå knappene i figur 3.2 merket **Kalibreringsprosedyre M1** og **Kalibreringsprosedyre M2** utfører kalibreringsprosedyre som beskrevet i vedlegg D.5 for den aktuelle servomotoren. Ved fullført kalibreringsprosedyre tennes lampene til høyre for kalibrerings-knappene.



Figur 3.2: I figuren vises startskjermen til programmet. Knappene **Slå av M1** og **Slå på M2** betjenes for å slå av/ på servodriverne og servomotorene, og lampene til høyre for knappene indikerer om servomotorene er påslått/ avslått. De blå knappene **Kalibreringsprosedyre M1** og **Kalibreringsprosedyre M2** utfører kalibreringsprosedyre for servomotorene, og lampene til høyre for disse to knappene viser kalibreringsstatus til servomotorene. De hvite tabellene viser data tilhørende servomotorene. Den rød knappen **Reset alarm** resetter eventuelle alarmer. Knappene **Start program** og **Stopp program**, starter og stopper programmet. I den hvite boksen **IIR Filter** velges ønsket alfa verdi som brukes for å filtrere beregnet akselerasjon og *jerk*. De grå knappene **Begrensning av parametre** og **Posisjon, hastighet og akselerasjon graf** fører til andre skjermer.

3. Kalibreringsprosedyre må være utført for begge servomotorene før programmet kan kjøres. Ved å betjene knappen **Start program** og **Stopp program** som vist i figur 3.2 startes og stoppes programmet. Når programmet kjører vil servomotor 1 og servomotor 2 beveges kontinuerlig mellom posisjon $= 0$ og posisjon $= 1000$, henholdsvis i enhetene $[mm]$ og $[grader]$ helt til programmet blir stoppet, som da resulterer i at servomotorene returnerer til posisjon $= 0$.

4. Ved å betjene knappen **Parametre** som vist i figur 3.2, åpnes skjermen vist i figur 3.3. For å endre på bevegelses-begrensningene til servomotorene betjenes knappene **Opp** og **Ned**.

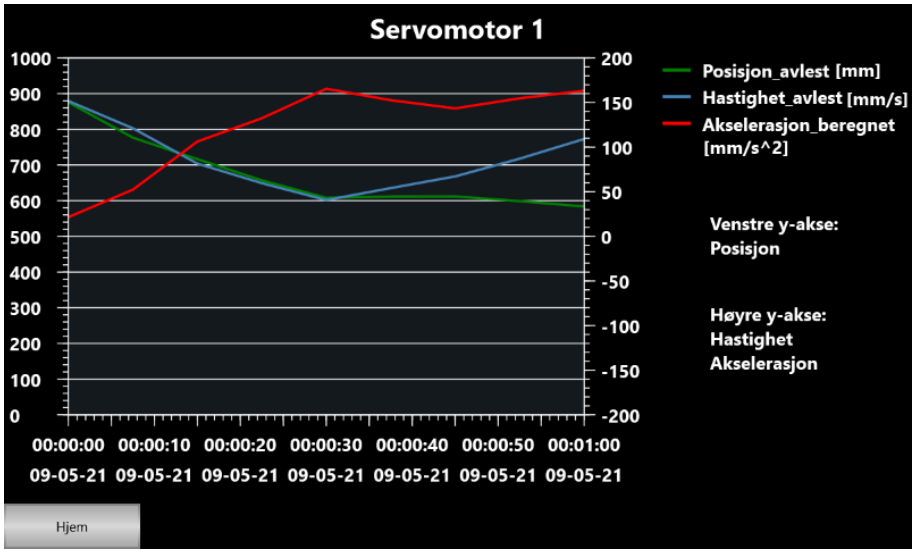


Figur 3.3: Skjermen i figuren åpnes ved å betjene knappen **Parametre** som vist i figur 3.2. Ved å betjene knappene **Opp** eller **Ned**, inkrementeres/dekrementere parametrene beskrevet på skjermen. Knappen **Hjem** åpner skjermen i figur 3.2.

5. Ved å betjene knappen **Posisjon, hastighet og akselerasjons graf** som vist i figur 3.2, åpnes skjermen vist i figur 3.4.

6. På skjermen vist i figur 3.4 vises variabler knyttet til servomotor 1. I punktlisten under presenteres disse variablene.

    - **Posisjon:** Leser posisjonen $[mm]$ til servomotor 1 fra servodriveren. Servodriveren beregner posisjon ved bruk av enkoderen.
    - **Hastighet:** Leser hastigheten $[mm/s]$ til servomotor 1 fra fra servodriveren. Servodriveren beregner hastigheten ved bruk av posisjon, som beregnes ved bruk av enkoderen.
    - **Akselerasjon:** Viser beregnet akselerasjonen $[mm/s^2]$ til servomotor 1. Akselerasjonen er funnet ved å derivere og filtrere hastigheten. I kapittel 3.4 vises det hvordan derivering og filtrering av hastighet har blitt gjort for å finne servomotoren akselrasjon.

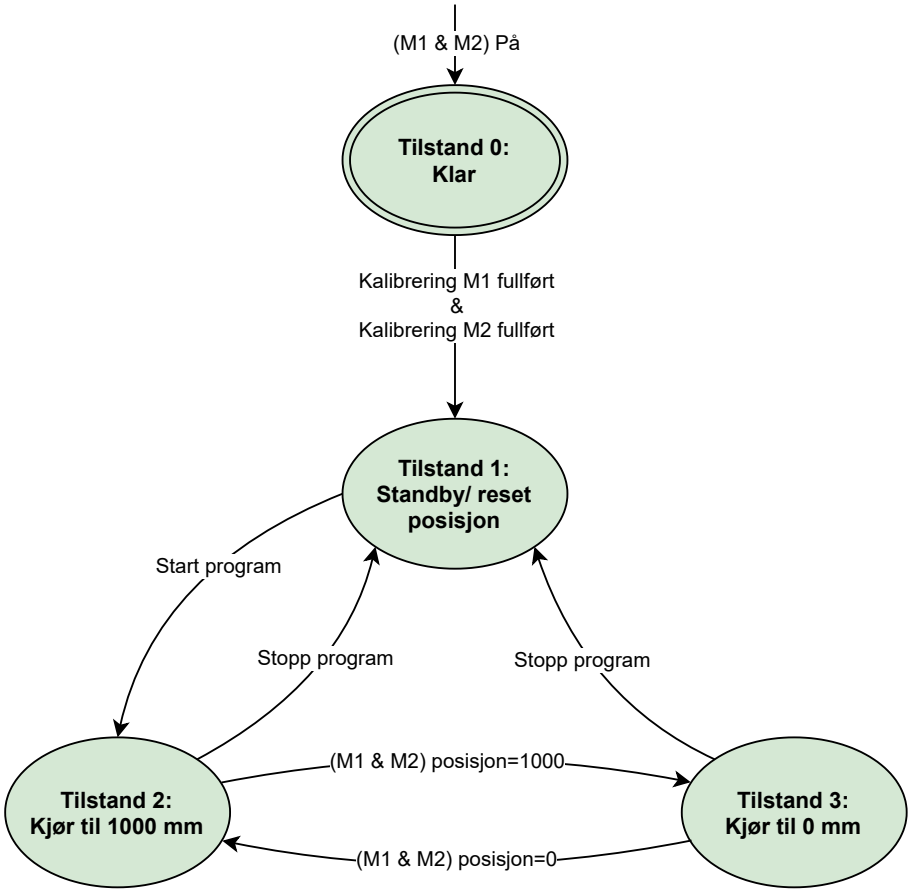


Figur 3.4: Graf som viser avlest posisjon, avlest hastighet og beregnet akselerasjon til servomotor 1. Y-aksene er knyttet til variablene som beskrevet i figuren. X-aksen er knyttet opp mot tid, og grafen viser data i ett minutters periode. Ved bruk av plot-funksjonen i *HMI*-skjermer så knyttes både klokketid og datomerking opp til x-aksen, og det er ikke mulighet til å ta bort datomerking selv om tidsintervallet som brukes er ett minutt. Grafen er hentet fra Sysmac Studio, og viser tilfeldige data.

I Sysmac Studio kan servomotorens posisjon og hastighet leses av fra servodriveren, men det er ikke mulighet for å lese av akselerasjon og *jerk*. Dette til tross for at det er mulig å begrense både hastighet, akselerasjon og *jerk*. Dette er grunnen til at akselerasjon og *jerk* har blitt beregnet manuelt i programmet for servomotor 1.

3.3 Tilstandsdiagram og programstruktur

For å vise hvordan programmet er bygd opp har vi i figur 3.5 tegnet et tilstandsdiagram som viser hovedstrukturen til programmet i dette kapittelet.



Figur 3.5: Tilstandsdiagram for programmet. Beskrivelse av tilstandene og transisjonene blir gitt videre i delkapittelet.

Videre forklares det hva som utføres i tilstandene, betingelse for transisjon mellom tilstandene og tilhørende kodeutklipp til tilstandene.

Tilstand 0: Dette er starttilstanden til programmet. Når servomotorene slås på går programmet til tilstand 0. Kravet for transisjon til tilstand 1 er oppfylt hvis både servomotor 1 og servomotor 2 har utført kalibreringsprosedyre som vist i vedlegg D.5. På kodelinje 24-29 i kodeutklipp 3.1 settes holde-variablene `kalibrering_1_utfort` og `kalibrering_2_utfort` høye dersom tilhørende `MC_Home`-funksjonsblokk (som vist i vedlegg D.2) gir ut høyt flankesignal om at kalibreringsprosedyre er utført. Dersom servomotorene slås av settes holde-variablene `kalibrering_1_utfort` og `kalibrering_2_utfort` lave, og kalibreringsprosedyre må utføres på ny før bruk av programmet.

Kode 3.1: Kodeutklipp for tilstand 0

```
19      // Servomotor 1 og 2 maa vaere aktivert
20      IF servo1_power_status AND servo2_power_status THEN
21          CASE case_program OF
22
23              0: // Klar
24              IF servo1_home_done THEN
25                  kalibrering_1_utfort:=TRUE;
26              END_IF;
27              IF servo2_home_done THEN
28                  kalibrering_2_utfort:=TRUE;
29              END_IF;
30              IF kalibrering_1_utfort AND ...
                    kalibrering_2_utfort THEN
31                  case_program:=1;
32              END_IF;
```

Tilstand 1: Denne tilstanden fungerer som en ventetilstand etter kalibreringsprosedyre har blitt utført. Tilstanden aktiverer `MC_MoveZeroPosition`-funksjonsblokken vist i vedlegg D.2, som gjør at servomotorene returnerer til kalibrert startposisjon = 0. Kravene for transisjon til tilstand 2 er oppfylt dersom begge servomotorene er i startposisjon, og bruker betjener knappen **Start program** som vist i figur 3.2.

Kode 3.2: Kodeutklipp for tilstand 1

```
34      1: // Standby/ reset posisjon
35      servo1_position0_execute:=TRUE;
36      servo2_position0_execute:=TRUE;
37      IF start_porgram AND servo1_position0_done AND ...
            servo2_position0_done THEN
38          servo1_position0_execute:=FALSE;
39          servo2_position0_execute:=FALSE;
40          case_program:=2;
41      END_IF;
```

Tilstand 2: I tilstanden aktiveres `MC_Move`-funksjonsblokker som vist i vedlegg D.2 med instruksjon om å bevege servomotorene til posisjon= 1000, og dette vises i kodeutklipp 3.3 på kodelinje 44-47. Kravene for transisjon til tilstand 3 er oppfylt når begge `MC_Move`-funksjonsblokkene gir signal om at instruksjonen er fullført. Dersom knappen **Stopp program** vist i figur 3.2 betjenes oppfylles kravet om transisjon til tilstand 1.

Kode 3.3: Kodeutklipp for tilstand 2

```
43      2: // Kjor mot 1000 [mm]/ [grader]
44      servo1_move_position:=1000;
45      servo2_move_position:=1000;
46      servo1_move_execute:=TRUE;
47      servo2_move_execute:=TRUE;
48      IF servo1_move_done AND servo2_move_done THEN
49          servo1_move_execute:=FALSE;
50          servo2_move_execute:=FALSE;
51          case_program:=3;
52      ELSIF stopp_program THEN
53          servo1_move_execute:=FALSE;
54          servo2_move_execute:=FALSE;
55          case_program:=1;
56      END_IF;
```

Tilstand 3: I tilstanden aktiveres `MC_Move`-funksjonsblokker som vist i vedlegg D.2 med instruksjon om å bevege servomotorene til posisjon= 0, og dette vises i kodeutklipp 3.4 på kodelinje 59-62. Kravene for transisjon til tilstand 2 er oppfylt når begge `MC_Move`-funksjonsblokkene gir signal om at instruksjonen er fullført. Dersom knappen **Stopp program** vist i figur 3.2 betjenes oppfylles kravet om transisjon til tilstand 1.

Kode 3.4: Kodeutklipp for tilstand 3

```
58      3: // Kjor mot 0 [mm]/ [grader]
59      servo1_move_position:=0;
60      servo2_move_position:=0;
61      servo1_move_execute:=TRUE;
62      servo2_move_execute:=TRUE;
63      IF servo1_move_done AND servo2_move_done THEN
64          servo1_move_execute:=FALSE;
65          servo2_move_execute:=FALSE;
66          case_program:=2;
67      ELSIF stopp_program THEN
68          servo1_move_execute:=FALSE;
69          servo2_move_execute:=FALSE;
70          case_program:=1;
71      END_IF;
```

3.4 Beregning av akselerasjon og *jerk*

I Sysmac Studio er det mulig å lese servomotorenes posisjon, moment og hastighet fra tilhørende servodriver, men det er ikke mulig å lese av servomotorens akselerasjon og *jerk*. Fordi dette kapittelet handler om å begrense posisjon, hastighet, akselerasjon og *jerk* så er det også ønskelig å kunne observere servomotorens akselerasjon og *jerk*. Videre vises det derfor hvordan akselerasjon og *jerk* beregnes ut fra servomotorenes avleste hastighet.

Sammenhengen mellom posisjon, hastighet, akselerasjon og *jerk* er vist i ligning (3.1) og ligning (3.2).

$$akselerasjon = \frac{d}{dt} hastighet = \frac{d^2}{dt^2} posisjon \tag{3.1}$$

$$jerk = \frac{d}{dt} akselerasjon \tag{3.2}$$

3.4 Beregning av akselerasjon og *jerk*

For å utføre derivasjon i Sysmac Studio har numerisk derivasjon blitt implementert. I ligning (3.3) vises formelen som er implementert i Sysmac Studio for å utføre numerisk derivasjon.

$$Sekant = \frac{Avlest\ verdi - Gammel\ avlest\ verdi}{Tidsskritt} \tag{3.3}$$

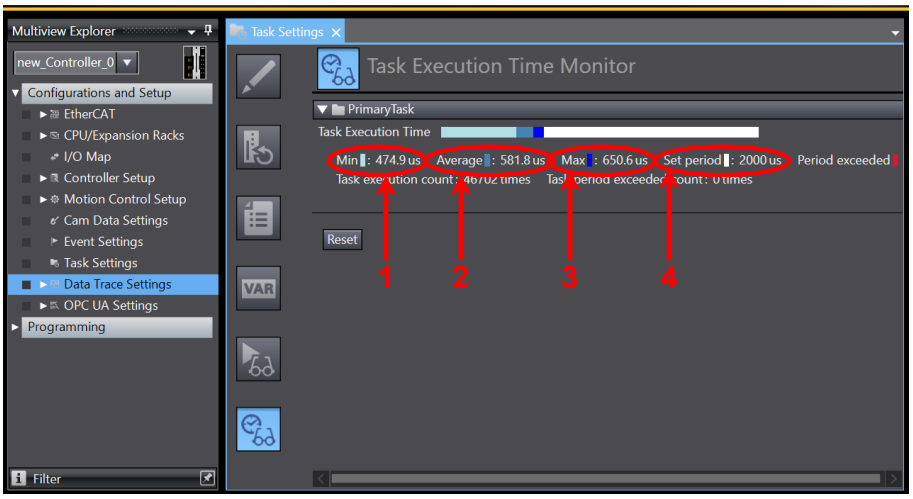
For å utføre numerisk derivasjon må det anvendes tidsskritt. I Sysmac Studio har vi valgt å sette opp PLS med syklisk eksekvering [13], og med en fast eksekveringshastighet på 2 ms. Den faste eksekveringshastighet lagres som `tidsskritt` i programmet som vist i kodeutklipp 3.5 på kodelinje 3.

Kode 3.5: Kodeutklippet er hentet fra dokumentasjonen i Sysmac Studio. I kodeutklippet vises det hvordan den faste eksekveringshastigheten til PLS lagres i programmet.

```
1 // Lagrer eksekveringshastigheten til programmet
2 IF P_First_RunMode = TRUE THEN
3    tidsskritt := ...
         LINT_TO_LREAL(TimeToNanoSec(GetMyTaskInterval())) ...
         /1000000000;
4 END_IF;
```

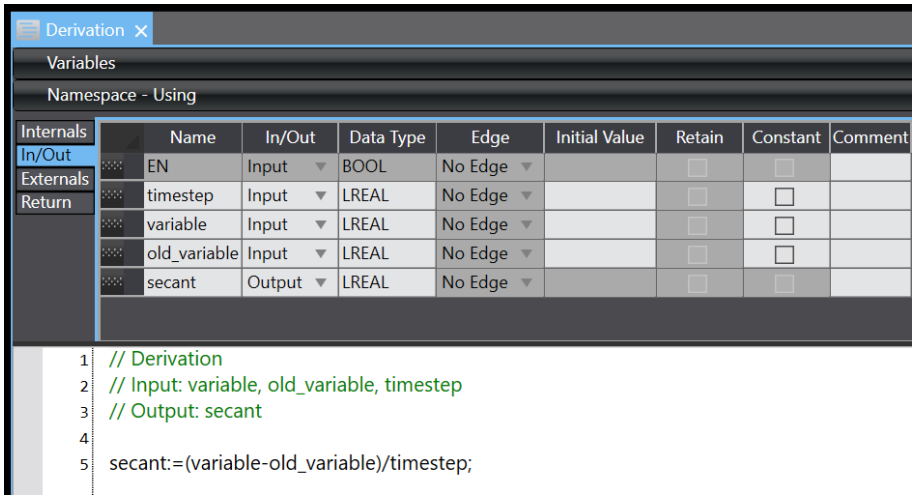
Den valgte eksekveringshastigheten må være større enn syklustiden til PLS, slik at PLS rekker å kjøre igjennom programmet før ny syklus starter. I figur 3.6 kan valgt eksekveringshastighet, minste syklustid, gjennomsnittlig syklustid og største syklustid leses av. I figuren har følgende blitt markert og lest av:

1. Minste syklustid $\approx 0,47ms$.
2. Gjennomsnittlig syklustid $\approx 0,58ms$.
3. Største syklustid $\approx 0,65ms$.
4. Valgt eksekveringshastighet $= 2ms$. Dette er valgt for at det skal være tilstrekkelig tid før ny syklus starter i PLS.



Figur 3.6: Figuren viser utklipp fra Sysmac Studio hvor det er mulig å lese av minste syklustid, gjennomsnittlig syklustid, største syklustid og valgt eksekveringshastighet.

I figur 3.7 vises funksjonen som har blitt opprettet for å utføre numerisk derivasjon. På kodelinje 5 i figuren er ligning (3.3) implementert. Derivasjons funksjonen har blitt opprettet som beskrevet i vedlegg D.4.



Figur 3.7: På kodelinje 5 i figuren vises koden for numerisk derivasjon. Funksjonen brukes til å derivere servomotorens hastighet for å finne akselerasjon, og til å derivere beregnet akselerasjon for å finne servomotorens *jerk*.

På kodelinje 31-36 i kodeutklipp 3.6 vises det hvordan derivasjonsfunksjonen fra figur 3.7 brukes til å beregne akselerasjon ut fra servomotorens hastighet som blir avlest fra servodriveren. Ved numerisk derivasjon av støybefengte målinger sammen med bruk av ett lite tidsskritt vil det deriverte signalet fremstå som ubrukelig. Derfor filtreres den numerisk deriverte akselerasjonen ved hjelp av ett enkelt første ordens *IIR*-filter (*Infinite impulse response filter*). På kodelinje 38-43 i kodeutklipp 3.6 vises det hvordan *IIR*-filter funksjonen brukes til å filtrere den beregnede akselerasjonen (`beregnet_akselerasjon` i kodeutklippet). Den gamle verdien (`gml_avlest_hastighet`) som brukes til å derivere hastigheten lagres i programmet som vist på kodelinje 45 i kodeutklipp 3.6. Hvordan *IIR*-filterert har blitt opprettet, og hvordan det fungerer vises i vedlegg D.4.

Kode 3.6: I kodeutklippet vises det hvordan funksjonene for derivasjon og *IIR*-filtrering brukes. Her er det hastighet som deriveres for å finne akselerasjon, og *IIR*-filteret brukes til å filtrere resultatet fra derivasjons funksjonen.

```
30      // Beregner akselerasjon og filtrerer
31      Derivation(
32          EN:=TRUE,
33          timestep:=tidsskritt,
34          variable:=avlest_hastighet,
35          old_variable:=gml_avlest_hastighet,
36          secant=>beregnet_akselerasjon   );
37
38      IIR_Filter(
39          EN:=TRUE,
40          alpha:=alfa,
41          variable:=beregnet_akselerasjon,
42          old_filtered_variable:=gml_IIR_beregnet_akselerasjon,
43          filtered_variable=>IIR_beregnet_akselerasjon    );
44
45      gml_avlest_hastighet:=avlest_hastighet;
```

Servomotorens *jerk* beregnes på samme vis som servomotorens akselerasjon som vist i kodeutklippet over. Ved beregning av *jerk* velges følgende:

- `variable` = `IIR_beregnet_akselerasjon`
- `old_variable` = `IIR_beregnet_akselerasjon` fra forrige syklus
- `timestep` = `tidsskritt`

3.5 Analysering av resultat i Matlab

Formålet med å begrense hastighet, akselerasjon og *jerk* er å oppnå jevnere/ roligere bevegelser av servomotorene. For vise forskjellen på servomotorens bevegelser med og uten begrensning av akselerasjon og/ eller *jerk* vises det videre til tre forskjellige tilfeller.

- I kapittel 3.5.1 vises servomotorens bevegelser der kun hastigheten har blitt begrenset.
- I kapittel 3.5.2 vises servomotorens bevegelser der hastighet og akselerasjon har blitt begrenset.
- I kapittel 3.5.3 vises servomotorens bevegelser der hastighet, akselerasjon og *jerk* har blitt begrenset.

For å grafisk presentere effekten av å begrense hastighet, akselerasjon og *jerk* så logges avlest posisjon, avlest hastighet, bergenet akselerasjon og beregnet *jerk* i Sysmac Studio ved hjelp av *data trace* som vist i vedlegg D.6. Etter kjøring av heismodellen blir opprettet *data trace* eksportert som en `.csv` fil fra Sysmac Studio, og `.csv` filen blir lest inn i Matlab som beskrevet i vedlegg D.7. Oppførselen til servomotor 1 og servomotor 2 er helt like, derfor er det valgt å kun vise frem resultatene til servomotor 1 i påfølgende delkapitler.

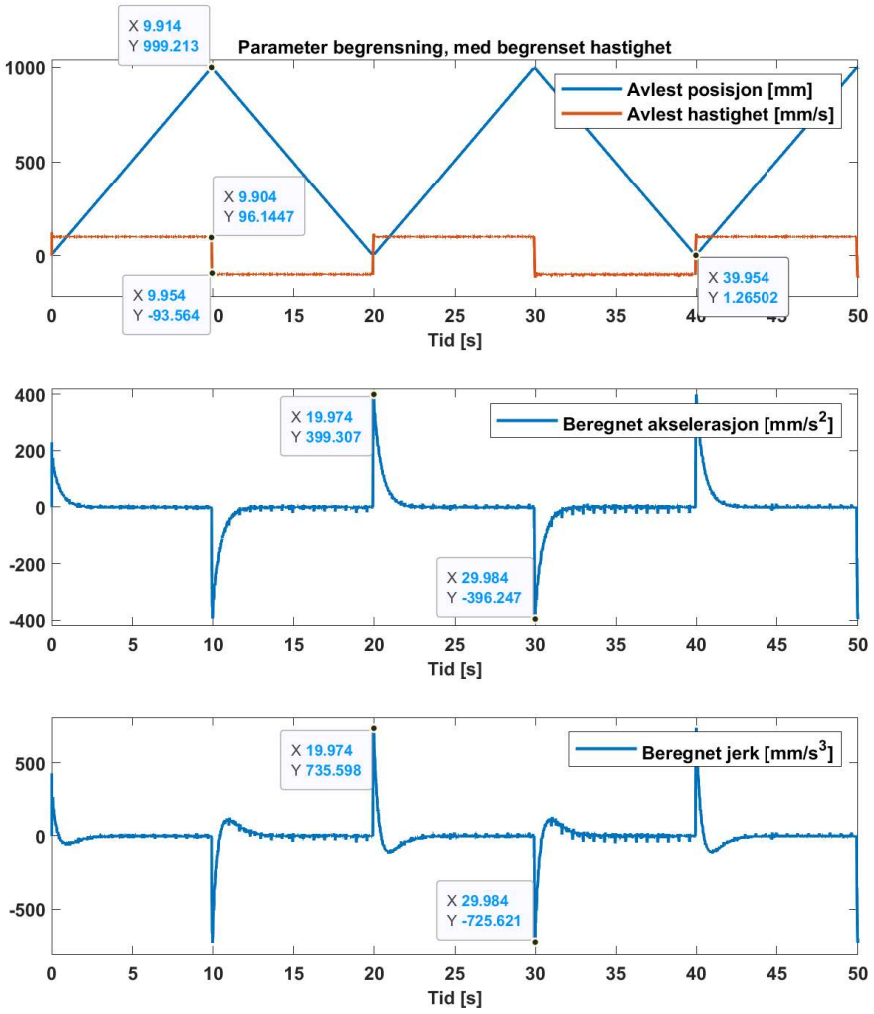
3.5.1 Med hastighets begrensning

I figur 3.8 er kun servomotorens hastighet begrenset. Hastigheten er begrenset til ± 100 $[\frac{mm}{s}]$. I figuren observeres det at servomotorens bevegelser er raske i endepunktene, som resulterer i relativt brå bevegelser.

I figur 3.8 kan følgende data leses av markeringer:

- Markeringene til venstre i øverste delplott viser at tiden servomotoren bruker på å endre hastigheten fra ≈ 100 $[\frac{mm}{s}]$ til ≈ -100 $[\frac{mm}{s}]$ er $\Delta Tid = 9,954 - 9,904 = 50$ $[ms]$.

- Det observeres i øverste delplott i figuren at hastigheten er tilnærmet konstant 100 $[\frac{mm}{s}]$ når servomotorens posisjon går fra $0 \rightarrow 1000$ $[mm]$.
- Det er ikke valgt begrensninger for akselerasjon og *jerk*. Det observeres i de to nederste delplottene i figuren at servomotorens akselerasjon er i intervallet $[-400, 400]$ $[\frac{mm}{s^2}]$ og at servomotorens *jerk* er i intervallet $[-750, 750]$ $[\frac{mm}{s^3}]$.



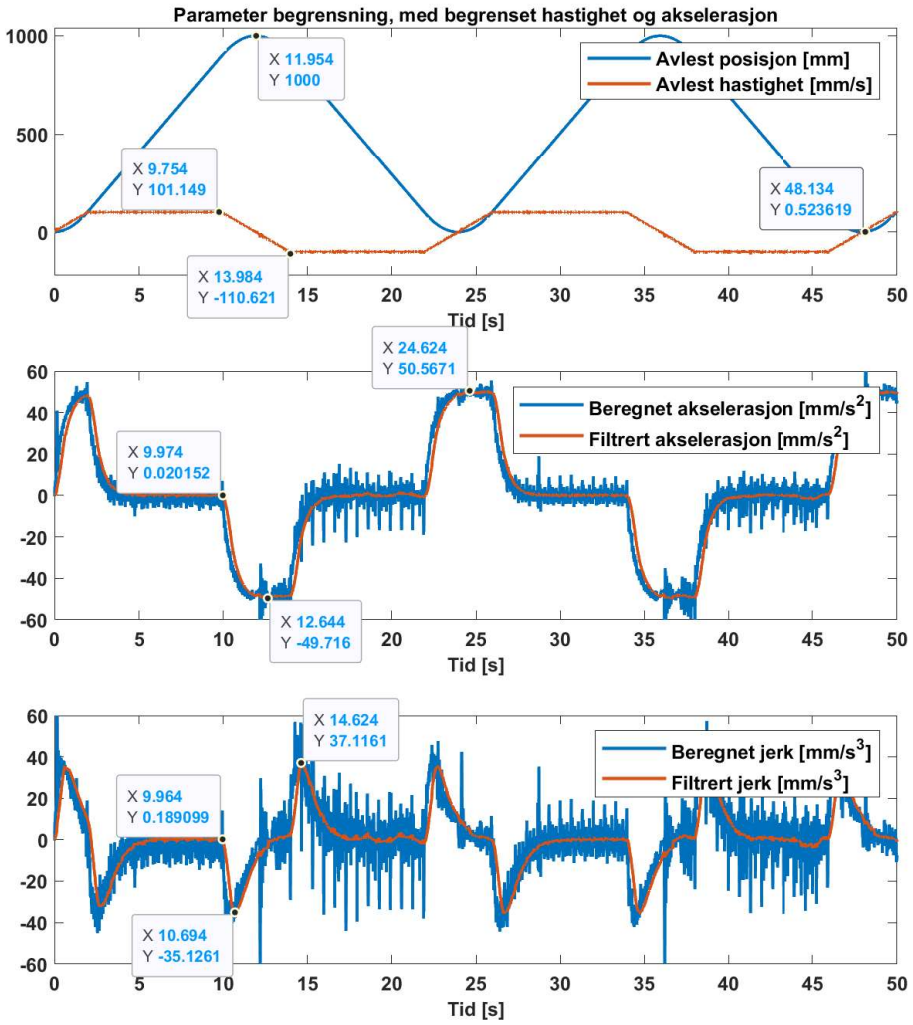
Figur 3.8: I figuren er servomotorens hastighet begrenset til ± 100 $[\frac{mm}{s}]$. Akselerasjon og *jerk* har ikke blitt begrenset. Merk at delplottene har forskjellig område på y-aksene.

3.5.2 Med hastighet og akselerasjon begrensning

I figur 3.9 er servomotorens hastighet og akselerasjon begrenset, men ikke *jerk*. Servomotorens hastighet er fortsatt begrenset til ± 100 $[\frac{mm}{s}]$, og servomotorens akselerasjonen har blitt begrenset til ± 50 $[\frac{mm}{s^2}]$. Det kommer tydelig frem i figuren at servomotorens bevegelser er jevnere/ roligere rundt endepunktene når både hastighet og akselerasjon blir begrenset.

I figur 3.9 kan følgende data leses av markeringer:

- Markeringene til venstre i øverste delplott viser at tiden servomotoren bruker på å endre hastigheten fra ≈ 100 $[\frac{mm}{s}]$ til ≈ -100 $[\frac{mm}{s}]$ er $\Delta Tid = 13,984 - 9,754 = 4,23$ $[s]$.
- Det observeres fra det midterste delplottet i figuren at servomotorens akselerasjon er $\pm 50[\frac{mm}{s^2}]$ når hastigheten endres.
- Det observeres fra det nederste delplottet i figuren at servomotorens *jerk* er i intervallet
  $[-60, 60]$ $[\frac{mm}{s^3}]$.



Figur 3.9: I figuren er servomotorens hastighet begrenset til ± 100 $[\frac{mm}{s}]$, akselerasjon er begrenset til $= \pm 50$ $[\frac{mm}{s^2}]$ og *jerk* har ikke blitt begrenset. Merk at delplottene har forskjellig område på y-aksene.

Akselerasjon og *jerk* beregningene har først blitt filtrert i Sysmac Studio ved hjelp av *IIR*-filteret med en alfa verdi $= 0,004$. Etter importering til Matlab ble det fortsatt observert en del støy, som kan observeres i de to nederste delplottene i figur 3.9. I kodeutklipp 3.7 vises det hvordan Matlabs `filter`-funksjon har blitt brukt til å filtrere akselerasjon og *jerk* beregningene. I de to nederste delplottene vist i figur 3.9 er beregnet akselerasjon og beregnet *jerk* plottet før og etter bruk av Matlabs `filter`-funksjon.

Kode 3.7: Bruk av `filter` funksjonen i Matlab for å filtrere beregnet akselerasjon og beregnet *jerk* ytterligere.

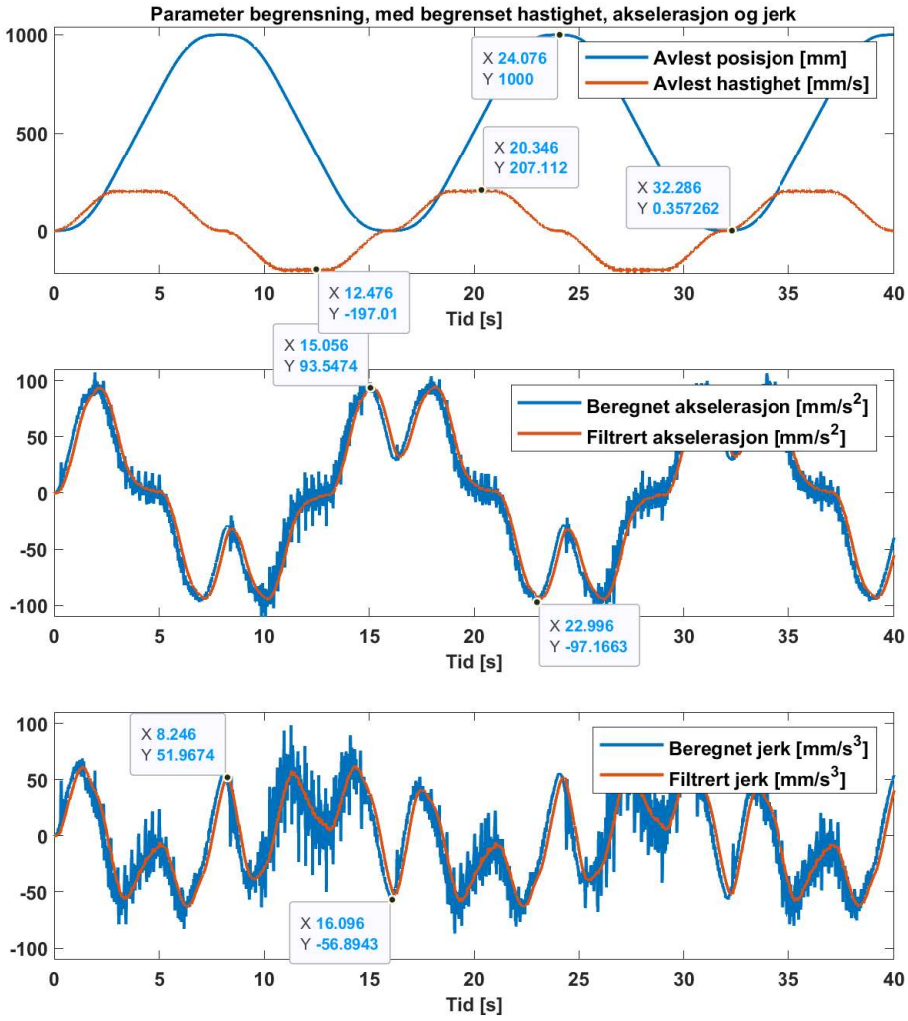
```
1 fil = funksjon_les_fil( ...
      'a_og_j_begrensning2_20210409100020_00001.csv');
2
3 x_lengde = 50;
4 a =1;
5 b =(1/x_lengde)*ones(1,x_lengde);
6
7 filtrert_jerk = filter(b,a,fil.IIR_beregnet_jerkLREAL);
8 filtrert_akselerasjon = ...
      filter(b,a,fil.IIR_beregnet_akselerasjonLREAL);
```

3.5.3 Med hastighet, akselerasjon og *jerk* begrensning

I figur 3.10 er servomotorens hastighet, akselerasjon og *jerk* begrenset. Servomotorens hastighets begrensning er endret til ± 200 $[\frac{mm}{s}]$, servomotorens akselerasjons begrensning har blitt endret til ± 100 $[\frac{mm}{s^2}]$ og servomotorens *jerk* begrensning er satt til ± 50 $[\frac{mm}{s^3}]$. Det kommer tydelig frem i figuren at servomotorens bevegelser er enda jevnere/ roligere rundt endepunktene når hastighet, akselerasjon og *jerk* blir begrenset.

I figur 3.10 kan følgende data observeres:

- Posisjon er i området $[0,\ 1000]$ $[mm]$ (øverste delplott).
- Hastighet er i området $[-200,\ 200]$ $[\frac{mm}{s}]$ (øverste delplott).
- Akselerasjon er i området $[-100,\ 100]$ $[\frac{mm}{s^2}]$ (midterste delplott).
- *Jerk* er omtrent i området $[-50,\ 50]$ $[\frac{mm}{s^3}]$ (nederste delplott).



Figur 3.10: I figuren er servomotorens hastighet begrenset til ± 200 $[\frac{mm}{s}]$, akselerasjon er begrenset til $= \pm 100$ $[\frac{mm}{s^2}]$ og *jerk* er begrenset til $= \pm 50$ $[\frac{mm}{s^3}]$. Merk at delplottene har forskjellig område på y-aksene.

3.6 Forslag til laboppgave

For at studentene skal få en god forståelse av effekten av å begrense hastighet, akselerasjon og *jerk* så vil det være fordel med ett labopplegg som består både av oppgaver før og på lab. Utgangspunktet for laboppgaven vil være at studenten skal lage ett program som får servomotor 1 til å gå kontinuerlig mellom to punkter, og deretter teste effekten av å begrense parametre på labben.

I vedlegg E.1 er det ett program som er konfigurert til heismodellen, men mangler kode som studenten skal implementere. Programmet i vedlegget inneholder:

- Ferdigprogrammert *HMI*.
- Ferdigprogrammerte blokker for aktivering, kalibrering og styring av servomotor 1.

3.6.1 Før lab

1. Lag tilstandsdiagram for ett program som instruerer servomotor 1 til å gå kontinuerlig mellom to punkter, der tilstandsdiagram vist i figur 3.5 kan brukes som løsningsforslag.
2. Opprett en funksjon for numerisk derivasjon og en for *IIR*-filter i Sysmac Studio. Hvordan en funksjon opprettes vises i vedlegg D.4, og programmering av derivasjons funksjon vises i kapittel 3.4.

3.6.2 På lab

1. Programmer tilstandsdiagrammet som ble tegnet før lab i strukturert tekst. I kapittel 3.5 vises det kodeutdrag på hvordan tilstandene i strukturert tekst kan programmeres.
2. Test derivasjonsfunksjonen ved beregne akselerasjon, og plott resultatet i *HMI* som vist i figur 3.4. Hvordan en graf konfigureres i *HMI* vises i vedlegg D.3.

Kapittel 4

Heisstyring med prioritering

I denne Videoen blir dette prosjektet demonstrert

4.1 Motivasjon for prosjekt

I faget ELE310 Styringsteknikkstyringsteknikk lærer studentene seg grunnleggende måter å bruke en PLS på. Et bruksområde for PLS i dag er heisstyringer. Heiser har forskjellig funksjonalitet alt etter hva bruksområdet til heisen er. En heis i en boligblokk har for eksempel andre prioriteringer enn en heis i et kjøpesenter. I en boligblokk hvor beboerne i de fleste tilfeller tar heisen fra sin leilighet og ned til utgangsetasjen er det ofte kun nødvendig med en utvendig knapp for hver av etasjene. I motsetning til et kjøpesenter hvor det vil være nødvendig med utvendige knapper som spesifiserer hvilken retning kunden ønsker å ta heisen for og kunne optimalisere effektiviteten til heisen, fordi det er like sannsynlig at kunden skal opp som ned. Systemet som skal beskrives i dette kapittelet er en heis i et kjøpesenter hvor det er flere brukere av heisen samtidig. En heis som bare går etter hvilken etasje som ble først aktivert vil da være lite effektiv, fordi heisen bare kan oppfølge ønsket til en passasjer av gangen. Derfor er det nødvendig med en heis som kan prioritere og sortere etasje-forespørsler slik at heisturen blir mest mulig effektiv for passasjerene.

4.1 Motivasjon for prosjekt

I presentasjonen av prosjektet har vi delt det opp i følgende deler:

1. Beskrivelse av heisens virkemåte, som vist i kapittel 4.2.
    - Beskrivelse av hvordan etasjene er fordelt på heismodellen, som vist i kapittel 4.2.1
2. Beskrivelse av etasjeprioriteringslisten som brukes når heisen er stasjonær og hvordan denne manipuleres, som vist i kapittel 4.3.
    - Oppretting av etasjeprioriteringslisten, som vist i kapittel 4.3.1.
    - Hvordan man legger til verdier i etasjeprioriteringslisten ved bruk av `StackPush`-funksjonblokken i Sysmac Studio, som vist i kapittel 4.3.2.
    - Hvordan man sorterer verdier i etasjeprioriteringslisten ved bruk av `RecSort`-funksjonblokken i Sysmac Studio, som vist i kapittel 4.3.3.
    - Hvordan man henter ut verdier fra etasjeprioriteringslisten ved bruk av `StackFIFO`-funksjonblokken i Sysmac Studio, som vist i kapittel 4.3.4.
    - Beskrivelse av hvordan etasjeprioriteringslisten blir sortert og hvordan en ny etasje blir satt, som vist i kappittel 4.3.5.
    - Eksempel på hvordan verdiene i etasjeprioriteringslisten blir behandlet, som i kapittel 4.3.6
3. Beskrivelse av hvordan etasjeprioriteringslisten blir overstyrt når heisen er i bevegelse, som vist i kapittel 4.4. løst
4. Beskrivelse av hvordan og servodriverene servodriverene er knyttet sammen for å lage funksjonen til heisdøra, som vist i kapittel 4.5.
5. Forslag til laboppgave ved programmering av *HMI*-skjerm, som vist i kapittel 4.6

4.2 Heisens virkemåte

Heisvognen som beveges i vertikal retning av servomotor 1 tilsvarer heiskupeen i en vanlig heis. Servomotor 2 som roterer den utstikkende plastplaten på heisvognen tilsvarer døren til en vanlig heis.

Heisen er programmert til å prioritere etasje-forespørsler ut fra hvilken retning den beveger seg i. Hvis heisen er på vei oppover vil den slippe av passasjerer som skal av underveis, og plukke opp passasjerer som skal være med opp. Hvis heisen er på vei nedover vil den slippe av og plukke opp passasjerer på samme vis. I *HMI*-skjermen er det lagt til betjeningsknapper for å styre heisen. Heisen har fire innvendige knapper, en for hver etasje, som vist i markert område nr. 4 i figur 4.1.

Utenfor heisen er det seks knapper som vist i markert område nr. 2 i figur 4.1. Øverste og nederste etasje har kun en utvendig knapp, mens de to midterste etasjene har to knapper hver. En knapp for passasjerer som skal opp og en for passasjerer som skal ned.

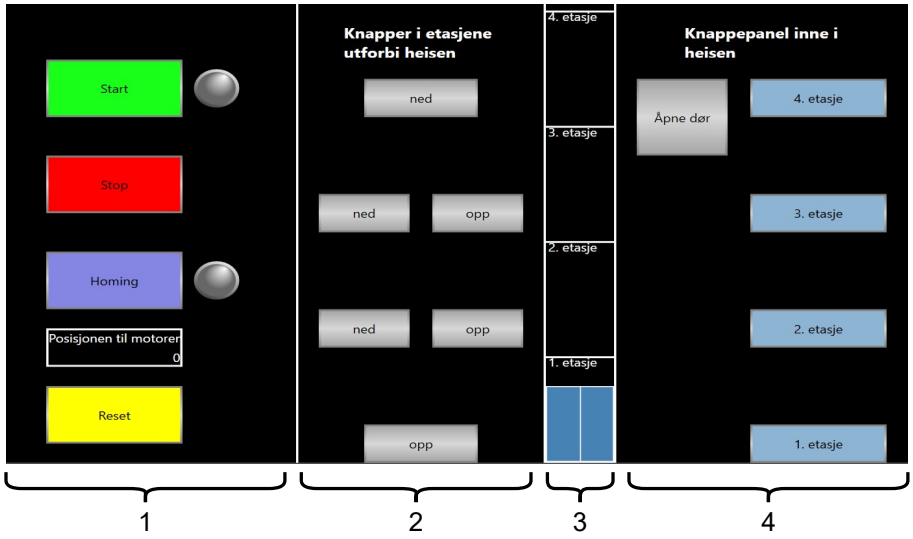
Er heisen stasjonær uten aktive etasje-forespørseler og en utvendig knapp blir aktivert, så vil heisen gå til den etasjen passasjeren befinner seg i uavhengig om passasjeren skal opp eller ned. Befinner passasjeren seg i samme etasje som heisen, og utvendig knapp blir aktivert, vil heisen åpne døren.

Når heisen kommer til en forespurt etasje vil dørene åpnes og lukkes automatisk. Det er også en knapp for å åpne dørene inne i heisen. I markert område nr. 3 i figur 4.1 viser en animasjon av posisjonen til heisen. Animasjonen følger heismodellens bevegelser i sanntid.

I markert område nr. 1 i figur 4.1 har vi følgende knapper:

1. **Start**: Aktiverer servodriver 1 og servodriver 2. Lampen til høyre for knappen indikerer om servodriverne er aktivert/ deaktivert.
2. **Stop**: Deaktiverer servodriver 1 og servodriver 2.
3. **Homing**: Utfører kalibreringsprosedyre som beskrevet i vedlegg D.5. Når kalibreringsprosedyre er fullført vil heisen stå i 1. etasje med startposisjon = 0 mm og lampen til høyre for knappen vil bli tent.

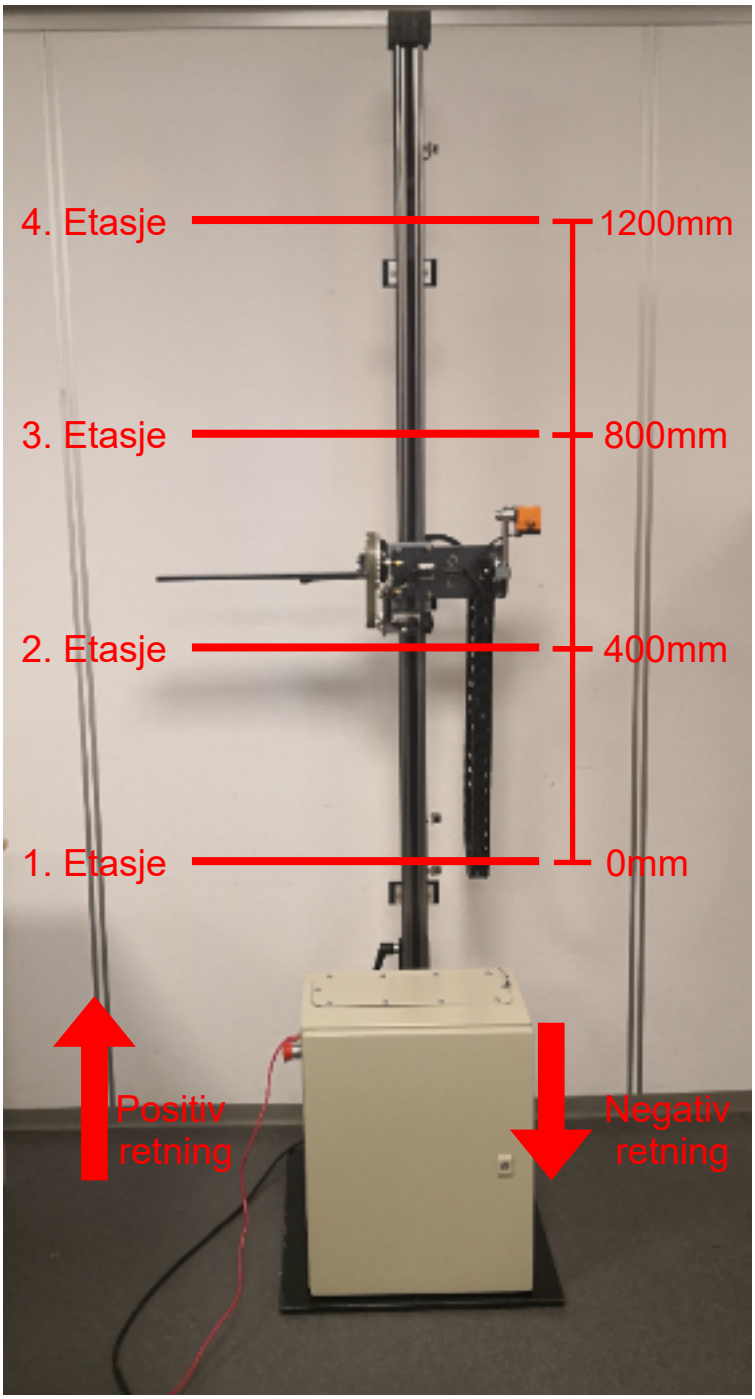
4. **Reset**: Resetter aktive feilmeldinger i PLS og servodrivere. Dette kan for eksempel være feilmeldinger som kommer av at en knapp er trykket før servodriverene er aktive. Mer informasjon i vedlegg D.2.



Figur 4.1: Figuren viser *HMI*-skjermen i dette prosjektet. Markeringene i figuren forklares ovenfor.

4.2.1 Etasjene på heismodellen

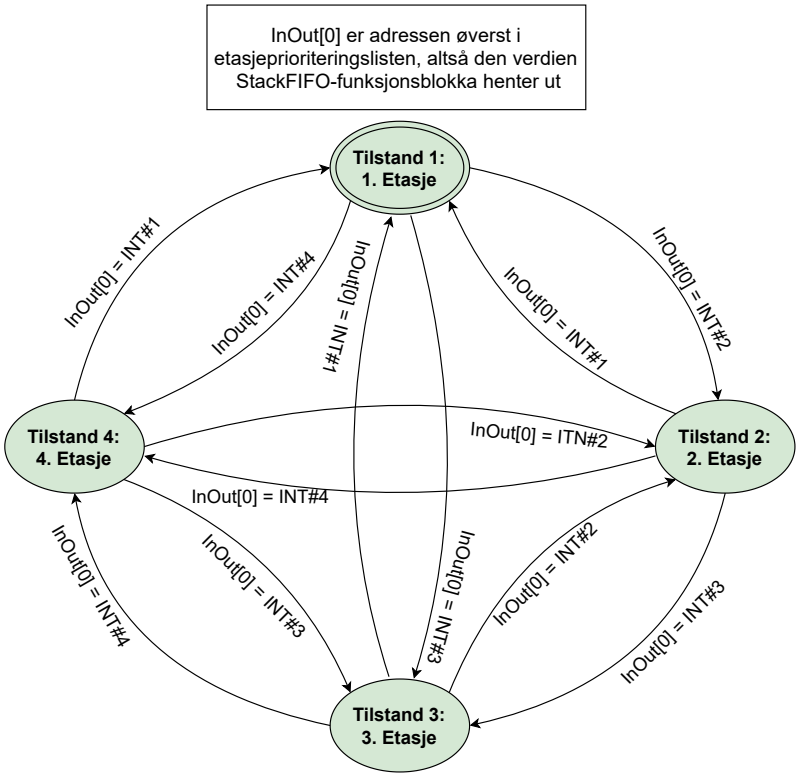
I dette prosjektet brukes servomotor 1 sin avleste enkoder posisjon til registrere hvilken etasje heisvognen befinner seg i. Heisvognens posisjon hentes ut med følgende kode: `MC_Axis000.Act.Pos`. De forskjellige etasjene er definert ved posisjonene beskrevet vist i figur 4.2. Ett annet alternativ er å bruke nærhets-sensorer for å indikere de forskjellige etasjene. Vi har valgt å bruke den avleste enkoderposisjonen siden den er nøyaktig og pålitelig. Det er også enklere å tilpasse antall etasjer uten behov for å montere flere nærhets-senorer på heismodellen. Etter fullført kalibreringsprosedyre står heisen i første etasje, og servomotor 1 sin enkoder avleser posisjon = 0 mm. Posisjonene er valgt ut fra lengden på skinneprofilen til heismodellen slik at etasjene er fordelt på lengden markert i figur 4.2.



Figur 4.2: Figuren viser de forskjellige etasjene definert i prosjektet med tilhørende posisjon som leses ved hjelp av enkoderen til servomotor 1.

4.3 Prioritering av etasje-forespørsler ved bruk av etasjeprioriteringsliste når heisen er stasjonær

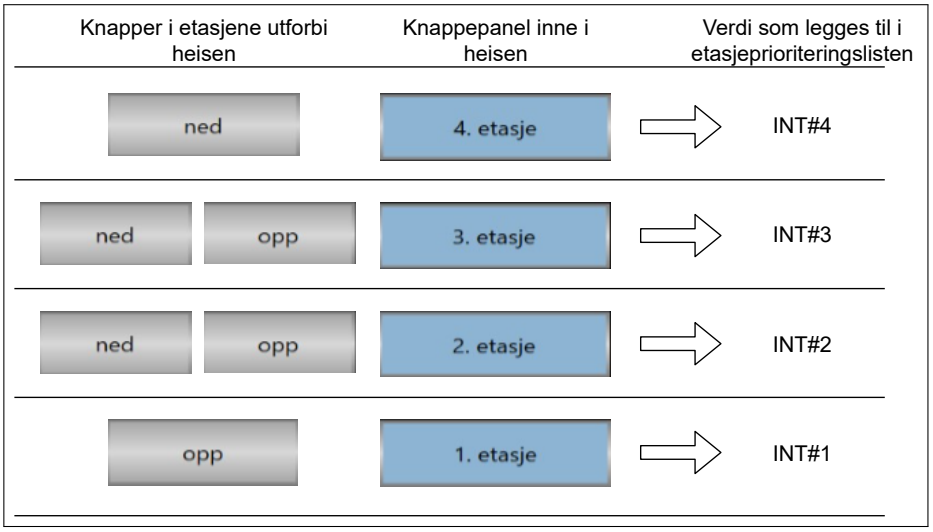
Heisen kan gå fritt mellom alle etasjene og ved flere etasje-forespørsler fra passasjerer vil heisen lagre forespørselene i ett *array*. Videre omtales *array*-et som brukes til å lagre etasje-forespørslene som etasjeprioriteringslisten. Ved flere etasje-forespørsler i etasjeprioriteringslisten vil heisen først gå til etasjen øverst i listen, og deretter vil heisen gå videre til neste i listen. Senere i dette kapitellet forklares det hvordan etasjeprioriteringslisten manipuleres for å oppnå en mest mulig effektiv prioritering. Funksjonaliteten til programmet når heisen er stasjonær er illustrert i figur 4.3 (i kapittel 4.4 vises det hvordan etasje-forespørsler som mottas når heisen er i bevegelse håndteres).



Figur 4.3: Tilstandsdiagram som funksjonaliteten til programmet når heisen er stasjonær. I figur 4.4 forklares det hva som menes med `INT#1-INT#4` i transisjonene.

4.3.1 Etasjeprioriteringsliste

For å få til funksjonaliteten vist i tilstandsdiagrammet i figur 4.3 gis hver etasje en `INT#` verdi fra 1 til 4 som vist i figur 4.4. Ved å betjene en av knappene vist i figur 4.4 lagres verdien til knappen i etasjeprioriteringslisten. I etasjeprioriteringslisten sorteres `INT#` verdiene i stigende eller synkende rekkefølge før de hentes ut og bestemmer hvilken etasje heisen skal gå til. Blir for eksempel en `INT#2` hentet ut av etasjeprioriteringslisten, så vil heisen gå til andre etasje.



Figur 4.4: I figuren vises verdier som de forskjellige knappene legger til i etasjeprioriteringslisten når heisen er stasjonær. Knappene er de samme som vist i avmerket område 2 og 4 i figur 4.1.

Etasjeprioriteringslisten er kalt `InOut` i programmet, og er spesifisert til å ha plass til ni `INT#` verdier som vist i figur 4.5. Dersom en av knappene tilhørende den etasjen heisen står i blir betjent vil verdien ikke legges inn i etasjeprioriteringslisten. Det vil si at det maksimalt vil kunne være åtte verdier i etasjeprioriteringslisten (eksempel: dersom heisen er i første etasje, og alle knapper i 2, 3 og 4 etasje blir betjent vil det bli lagt til åtte verdier). Derfor er det viktig at etasjeprioriteringslisten har plass til mer enn åtte verdier, derfor er størrelsen på etasjeprioriteringslisten valgt til å kunne inneholde ni verdier.

4.3 Prioritering av etasje-forespørsler ved bruk av etasjeprioriteringsliste når heisen er stasjonær

Figur 4.5: Figuren viser utklipp av etasjeprioriteringslisten som er implementert i programmet

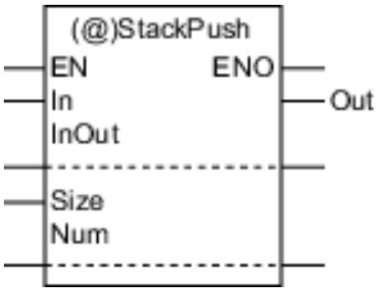
`Arrayet` som er opprettet som vist i figur 4.5 er illustrert i figur 4.6. I figur 4.6 er det illustrert at hver celle i etasjeprioriteringslisten tar inn datatypen `INT#`, men i programmet vil tomme celler ha verdien null.

INT#	InOut[0]
INT#	InOut[1]
INT#	InOut[2]
INT#	InOut[3]
INT#	InOut[4]
INT#	InOut[5]
INT#	InOut[6]
INT#	InOut[7]
INT#	InOut[8]

Figur 4.6: Array med navn `InOut` som består av 9 `INT#` verdier der InOut[0] - InOut[8] er adressen til `INT#` verdiene i etasjeprioriteringslisten

4.3.2 `StackPush`-funksjonsblokk

For å legge til `INT#` verdier i etasjeprioriteringslisten brukes det en `StackPush`-funksjonsblokk, som vist i figur 4.7. `INT#` verdiene som blir lagt til i etasjeprioriteringslisten representerer etasjene og legges inn i etasjeprioriteringslisten slik at det er mulig å sortere etasjer om det er flere etasjeønsker samtidig.



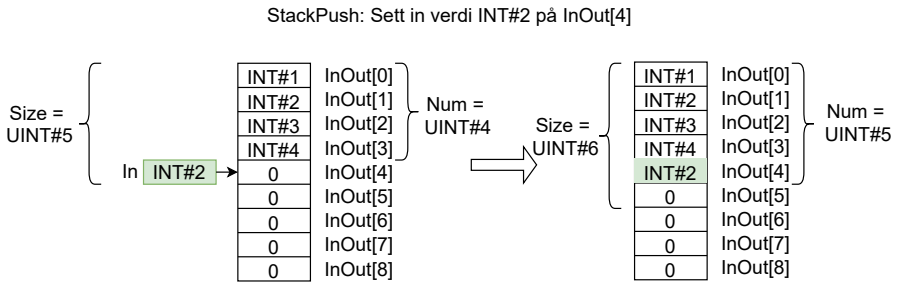
Figur 4.7: `StackPush`-funksjonsblokk fra Sysmac Studio

Under er de forskjellige inngangene og utgangene på `StackPush`-funksjonsblokken i figur 4.7 beskrevet:

- `EN`: Ved høyt signal legges `INT#` verdien som ligger på inngangen `In` til i etasjeprioriteringslisten som ligger på inngangen `InOut`.
- `In`: `INT#` verdien på `In` blir lagt til i etasjeprioriteringslisten som ligger på inngangen `InOut` ved høyt signal på inngangen `EN`.
- `InOut`: Her knytter man opp minnet som man vil legge til `INT#` verdien i (i denne oppgaven er det etasjeprioriteringslisten `InOut` som heter det samme som inngangen på funksjonsblokken).
- `Size`: Her spesifiseres hvor stor del av etasjeprioriteringslisten som skal brukes ved høyt signal på `EN`.
- `Num`: Her vises hvor mange `INT#` verdier som er i etasjeprioriteringslisten til en hver tid
- `ENO`: Gir signal hver gang funksjonsblokken har utført en instruksjon.

4.3 Prioritering av etasje-forespørsler ved bruk av etasjeprioriteringsliste når heisen er stasjonær

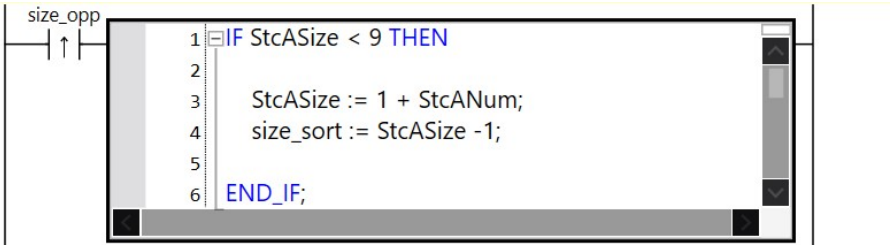
For å legge til en ny verdi må `StackPush`-funksjonsblokken få et signal på `EN`. På `InOut`-inngangen legger vi til etasjeprioriteringslisten `InOut`, som heter det samme som inngangen til funksjonsblokken på grunn av god oversikt i programmet. Adressen i etasjeprioriteringslisten hvor den første `INT#` verdiene skal legges til spesifiseres ved å legge til en `[0]` etter navnet på etasjeprioriteringslisten(`InOut[0]`). Da vil den første `INT#` verdien legge seg øverst i etasjeprioriteringslisten. Om det legges til enda en ny `INT#` verdi vil denne verdien legge seg på `InOut[1]` og den forrige `INT#` verdien vil beholde plassen sin på `InOut[0]`. På figur 4.8 er det vist et eksempel der det er 4 verdier i etasjeprioriteringslisten og `Stackpush`-funksjonsblokken legger til en ny `INT#` verdi.



Figur 4.8: `StackPush`-funksjonsblokken legger til en `INT#2` verdi på adressen `InOut[4]`. Utgangspunktet til etasjeprioriteringslisten er på venstre side og prioriteringsliste med ny `INT#` verdi på høyre side

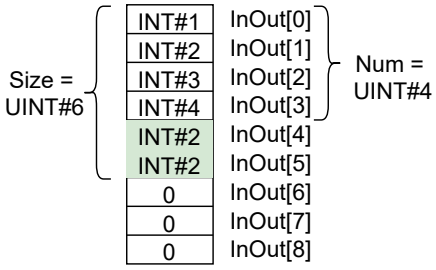
Det må spesifiseres hvor mange av plassene i etasjeprioriteringslisten som skal få den nye `INT#` verdien, slik at ikke `INT#` verdien blir lagt inn flere plasser i etasjeprioriteringslisten. Dette realiseres ved å sette variabelen `StcASize` som er knyttet til `Size`-inngangen på `StackPush`-funksjonsblokken en større enn antall verdier i etasjeprioriteringslisten (`Num`). Da forteller vi til `StackPush`-funksjonsblokken at det er en ledig plass å legge til `INT#` verdien på, som vist i figur 4.8. I figur 4.9 på kodelinje 3 kan man se hvordan dette er løst med strukturert tekst i Sysmac Studio.

4.3 Prioritering av etasje-forespørsler ved bruk av etasjeprioriteringsliste når heisen er stasjonær



Figur 4.9: Hver gang det blir lagt til en ny `INT#` verdi vil utgangen `ENO`-utgangen på `StackPush`-funksjonsblokken gi et signal til `Size_opp`-bryteren på figuren slik at koden i figuren kjøres en gang. `UINT#` verdien på `StcASize` som er knyttet til `Size`-inngangen på Stackpush-funksjonsblokken blir da en høyere enn `StcANum` som er knyttet til `Num` på `StackPush`-funksjonsblokken

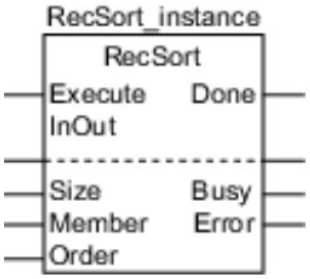
Om man derimot setter `Size` til å være to større enn `Num` vil to `INT#` verdier bli lagt til, som vist i figur 4.10.



Figur 4.10: `Size` = 6 og `Num` = 4. Grønne celler er nye verdier.

4.3.3 `RecSort`-funksjonsblokk

For å sortere etasjene i etasjeprioriteringslisten slik at heisen opererer på en effektiv måte brukes to `RecSort`-funksjonsblokker. Grunnen til at vi bruker to `RecSort`-funksjonsblokker er fordi at vi kan ha en som sorterer `INT#` verdiene i etasjeprioriteringslisten i stigende rekkefølge og en som sorterer i synkende rekkefølge. Da vil heisen etter hvilken vei den går ta med seg etasjene i den rekkfølgen som passer og ikke etter hvilke etasje-forespørsler som ble valgt først. `StackPush`-funksjonsblokken er vist i figur 4.11



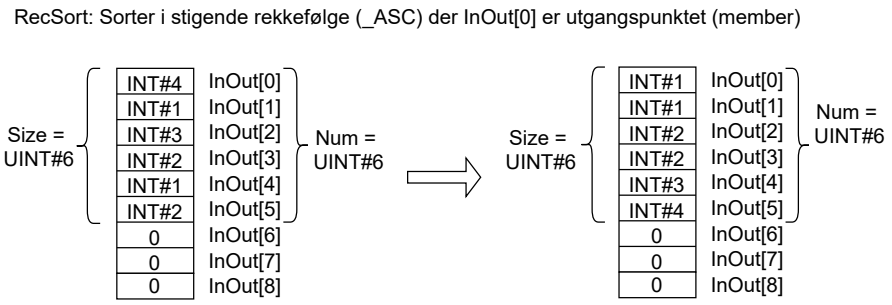
Figur 4.11: `RecSort`-funksjonsblokk fra Sysmac Studio

Under er de viktigste inngangene på `RecSort`-funksjonsblokken i figur 4.11 beskrevet:

- `Execute`: Ved høyt signal sorteres verdiene i etasjeprioriteringslisten
- `InOut`: Her knytter man opp minnet som man vil sortere (i denne oppgaven etasjeprioriteringslisten)
- `Size`: Her spesifiseres hvor mange verdier som skal sorteres ved høyt signal på `Execute`.
- `Member`: Hvilket Member i `STRUCT[]`-gruppen man skal bruke.
- `Order`: Rekkefølgen man vil sortere `INT#` verdiene i etasjeprioriteringslisten på der `_ASC` er stigende rekkefølge og `_DESC` er synkende rekkefølge
- `ENO`: Gir signal hver gang blokka har utført en instruksjon.

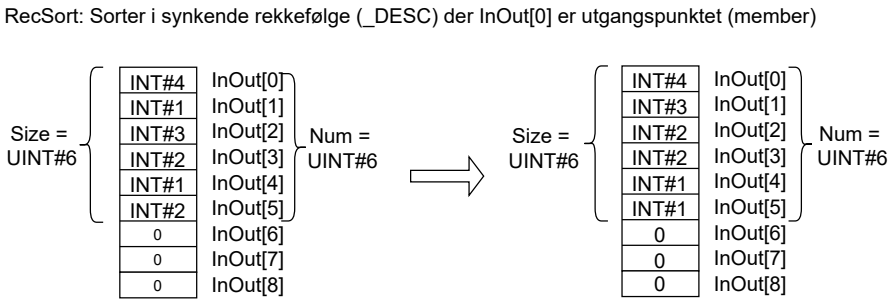
4.3 Prioritering av etasje-forespørsler ved bruk av etasjeprioriteringsliste når heisen er stasjonær

På `InOut`-inngangen på `RecSort`-funksjonsblokken knytter vi opp etasjeprioriteringslisten. Det spesifiseres hvor mange verdier av etasjeprioriteringslisten som skal sorteres ved å sette `Size` lik `Num` slik at alle `INT#` verdiene i etasjeprioriteringslisten blir sortert. Dette er vist på figur 4.9 i linje 4. Adressen i etasjeprioriteringslisten som skal være utgangspunkt for sorteringen settes til `InOut[0]` slik at `INT#` verdien som blir tatt ut enten er den høyeste eller den laveste `INT#` verdien i etasjeprioriteringslisten. En av de to `RecSort`-funksjonsblokkene brukes til å sortere `INT#` verdiene i etasjeprioriteringslisten i stigende rekkefølge der `InOut[0]` blir den laveste verdien, som vist i figur 4.12. På denne blokken settes `_ASC` på `Order`-inngangen. Den andre `RecSort`-funksjonsblokken brukes til å sortere `INT#` verdiene i etasjeprioriteringslisten i synkende rekkefølge der `InOut[0]` blir den høyeste verdien, som vist i figur 4.13. På denne blokken settes `_DESC` på `Order`-inngangen.



Figur 4.12: `_ASC` på inngangen `Order` gjør at `RecSort`-funksjonsblokken sorterer `INT#` verdiene i etasjeprioriteringslisten i stigende rekkefølge der `InOut[0]` er den laveste `INT#` verdien. Usortert etasjeprioriteringsliste på venstre side og sortert etasjeprioriteringsliste på høyre side.

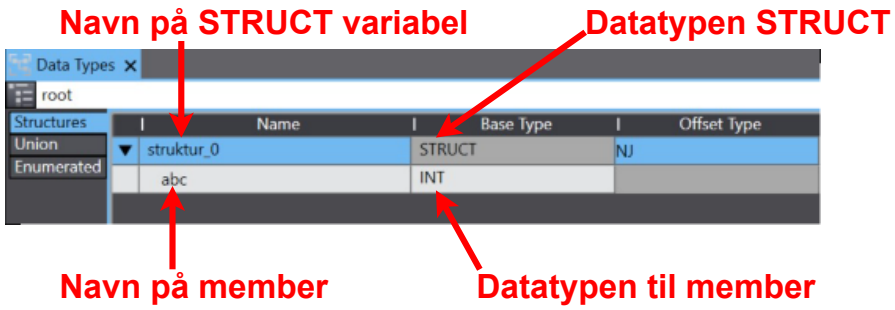
4.3 Prioritering av etasje-forespørsler ved bruk av etasjeprioriteringsliste når heisen er stasjonær



Figur 4.13: `_DESC` på inngangen `Order` gjør at `RecSort`-funksjonsblokken sorterer `INT#` verdiene i etasjeprioriteringslisten i synkende rekkefølge der `InOut[0]` er den høyeste `INT#` verdien. Usortert etasjeprioriteringsliste på venstre side og sortert etasjeprioriteringsliste på høyre side.

`RecSort`-funksjonsblokken tar kun inn datatypen `STRUCT[]` på inngangen `InOut`. Dette er en datatype der flere variabler kan legges til og hentes ut. Derfor må vi lage en variabel med Datatypen `STRUCT[]`. Måten dette gjøres på er ved å lage en ny variabel med datatypen `STRUCT[]` i multiview Explorer på Sysmac Studio under Programing/Data/Data types. Siden etasjeverdiene i koden har datatype `INT#` lager vi en `Member` som vi kaller `abc` med datatype `INT#`. Siden det nå jobbes med datatypen-`STRUCT[]` så må denne legges inn i etasjeprioriteringslisten ved hjelp av `StackPush`-funksjonsblokken.

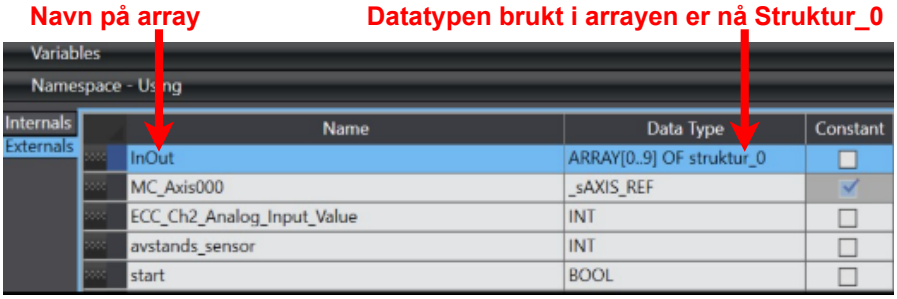
Figur 4.14 viser `Struktur_0` med `Member abc`, som er `STRUCT[]`-variabelen brukt i dette prosjektet.



Figur 4.14: Figuren viser hvordan en variabel av typen `STRUCT` er konfigurert til å ta inn en `INT` verdi.

4.3 Prioritering av etasje-forespørsler ved bruk av etasjeprioriteringsliste når heisen er stasjonær

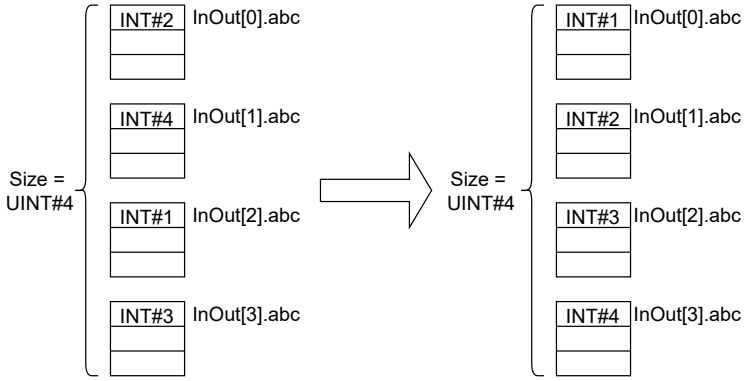
`STRUCT[]`-variabelen som ble laget må så legges inn i Variable table som datatypen `ARRAY[0..9] OF struktur_0` der `struktur_0` er navnet på `STRUCT[]`-variabelen som vi laget i Data types. I figur 4.15 kan man se at den nye variabelen til etasjeprioriteringslisten blir kalt `InOut` som tidligere.



Figur 4.15: I figuren vises det at `STRUCT` variabelen er lagt til i de globale variablene i Sysmac Studio.

For å hente etasjeverdiene i etasjeprioriteringslisten så må man i Sysmac Studio skrive: `navn_på_variabel[plass_til_verdi].navn_på_Member`. I tilfellet der man vil hente ut verdien i `Member abc` i adresse `InOut[0]` må man skrive `InOut[0].abc` i Sysmac Studio.

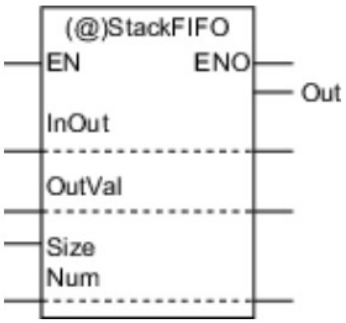
Som man kan se på figur 4.16 vil etasjeprioriteringslisten nå bestå av grupper med flere celler som i Sysmac Studio kalles for `Member`. Dermed kan hver gruppe inneholde flere variabler. Om det kun brukes en `Member` i hver gruppe vil etasjeprioriteringslisten fungere på samme måte som tidligere. Derfor fortsetter vi å beskrive etasjeprioriteringslisten på samme måte som tidligere.



Figur 4.16: Etasjeprioriteringslisten med datatypen `STRUCT[]` med fire `INT#` verdier.

4.3.4 `StackFIFO`-funksjonsblokk

For å hente ut verdier fra etasjeprioriteringslisten slik at heisen kan utføre etasjeforespørselen brukes en `StackFIFO`-funksjonsblokk (*First In First Out*) som vist i figur 4.17. Grunnen til at `StackFIFO` blir brukt er at vi vil hente ut `INT#` verdien på adressen `InOut[0]` i etasjeprioriteringslisten.



Figur 4.17: `StackFIFO`-funksjonsblokk i Sysmac Studio

4.3 Prioritering av etasje-forespørsler ved bruk av etasjeprioriteringsliste når heisen er stasjonær

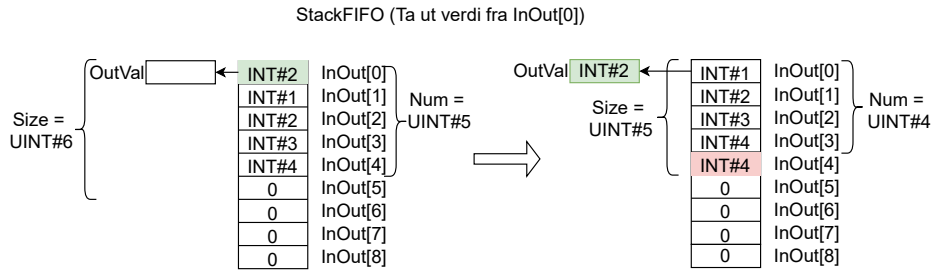
Under er de forskjellige inngangene på `StackPush`-funksjonsblokken i figur 4.17 beskrevet:

- `EN`: Ved høyt signal legges `INT#` verdien på adressen `InOut[0]` til i `OutVal` på funksjonsblokken.
- `InOut`: Her knytter man opp minnet som man vil hente verdien fra (i denne oppgaven etasjeprioriteringslisten).
- `OutVal`: Her legges verdien som blir hentet ut fra etasjeprioriteringslisten.
- `Size`: Her spesifiseres hvor stor del av etasjeprioriteringslisten som skal hentes ut ved høyt signal på `EN`-inngangen.
- `Num`: Her vises hvor mange verdier som er i etasjeprioriteringslisten til en hver tid.
- `ENO`: Gir signal hver gang blokken har utført en instruksjon.

`StackFIFO`-funksjonsblokken er spesifisert til å hente ut verdien på `InOut[0]` fra etasjeprioriteringslisten hver gang blokken får et positivt signal på `EN`. Denne verdien legges inn i variabelen `tilstand_out` som er knyttet til `OutVal` på `StackFIFO`-funksjonsblokken. `INT#` verdien som blir tatt ut av etasjeprioriteringslisten blir brukt til å bestemme hvilken tilstand programmet skal gå til.

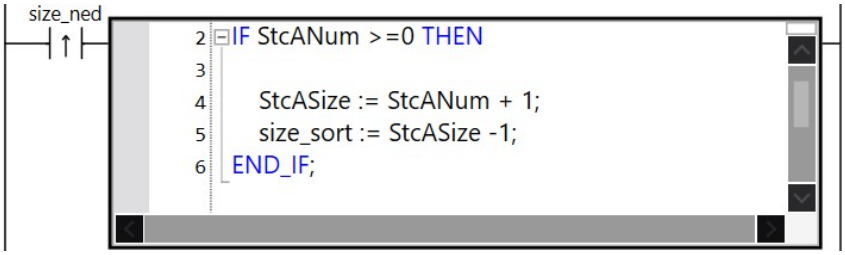
I figur 4.18 vises det at ved et positivt signal på `EN` på funksjonsblokken vil `INT#` verdien i `InOut[0]` legges inn i `OutVal`. `INT#` verdien som var på `InOut[0]` vil bli tatt ut og `INT#` verdien som var på `InOut[1]` tar plassen. `INT#` verdien som var på `InOut[4]` blir flyttet til `InOut[3]` men vil også stå igjen på `InOut[4]`, som vist i rød celle i figur 4.18. Denne `INT#` verdien vil ikke være en gyldig verdi i etasjeprioriteringslisten lengre siden `Num` har gått fra 5 til 4.

4.3 Prioritering av etasje-forespørsler ved bruk av etasjeprioriteringsliste når heisen er stasjonær



Figur 4.18: `StackFIFO`-funksjonsblokken tar ut verdi fra `InOut[0]`. På venstre side er etasjeprioriteringslisten før `INT#` verdien er tatt ut av `StackFIFO`-funksjonsblokken og på høyre side er `INT#` verdien tatt ut av `StackFIFO`-funksjonsblokken. Den røde cellen i figuren indikerer at verdien står igjen, men den er ikke gyldig da hele listen har blitt skrevet over en plass opp.

På `ENO` utgangen til `StackFIFO`-funksjonsblokken knyttes det til kode som legger til en ny verdi på `StcASize`, slik at verdien på `Size` er en større enn verdien på `Num` som vist i figur 4.19. Da vil `Size` være oppdatert til neste gang man vil legge til en `INT#` verdi i etasjeprioriteringslisten



Figur 4.19: Strukturert tekst kode for `StcASize`

4.3.5 Transisjon og sorteringslogikk når heisen er stasjonær. Markeringene i figuren forklares i listen under.

Kodeutklippet under er et utdrag fra koden som sorterer og henter ut ny `INT#` verdi fra etasjeprioriteringslisten slik at heisen går videre til et nytt etasjeønske når heisen er stasjonær.

Kode 4.0: Transisjon og sorteringslogikk når heisen er stasjonær

```
29              // Transisjon; Henter ny etasje forespørsel fra array
30              og går til tilstanden, inneholder også
31              prioriteringslogikk
32              IF m1_inaktiv AND MC_Axis000.Act.Pos < posisjon +5 AND
33              MC_Axis000.Act.Pos > posisjon -5 THEN
34          1       timer_start := TRUE;
35                  etasje_1_innvendig := FALSE;
36                  etasje_1_utvendig_opp := FALSE;
37                  servodriver_m1_utfor := FALSE;
38                  IF dor_lukket AND NOT dor_open and
39                  (etasje_2_innvendig OR etasje_3_innvendig OR
40                  etasje_4_innvendig OR etasje_2_utvendig_ned OR
41                  etasje_2_utvendig_opp OR etasje_3_utvendig_ned OR
42              2   etasje_3_utvendig_opp OR etasje_4_utvendig_ned)
43                  and timer_ferdig THEN
44                       sorter_stigende := TRUE;
45                       timer_start := FALSE;
46                       timer_reset := TRUE;
47                       IF  sorter_stigende_ferdig THEN
48                  3         ny_tilstand := TRUE;
49                            tilstand := tilstand_out.abc;
50                       END_IF;
51                  END_IF;
52              END_IF;
```

Under er de markerte områdene i kodeutklipp over beskrevet.

1. Når heisen står i ro (`m1_innaktiv` i kodelinje 32) og posisjonen til etasjen er nådd (`MC_Axis000.Act.Pos < posisjon + 5 AND MC_Axis000.Act.Pos > posisjon - 5` i kodelinje 32 og 33) starter en timer på ett sekund slik at passasjerer kan velge etasjer fra knappepanelet inne i heisen før den kjører videre. Tiden på timeren kan godt være mer enn ett sekund, men her er ett sekund valgt for å få en effektiv heis. Knappene til etasjen heisen befinner seg i blir satt lave slik at de er klare til å bli aktivert igjen, og `execute` signalet

på `MC_move`-funksjonsblokken til `servodriver_m1` (som vist i kapittel 4.5) blir satt lav slik at blokka er klar for ny instruksjon.

2. Når døren er lukket, timeren er ferdig og en eller flere av knappene til de etasjene som heisen ikke befinner seg i er høye vil heisen sortere etasjene i etasjeprioriteringslisten og resette timeren. Denne `IF`-løkka gjør at heisen kjører automatisk, der både de utvendige og innvendige knappene fungerer som et startsignal for å sende heisen til neste etasje.

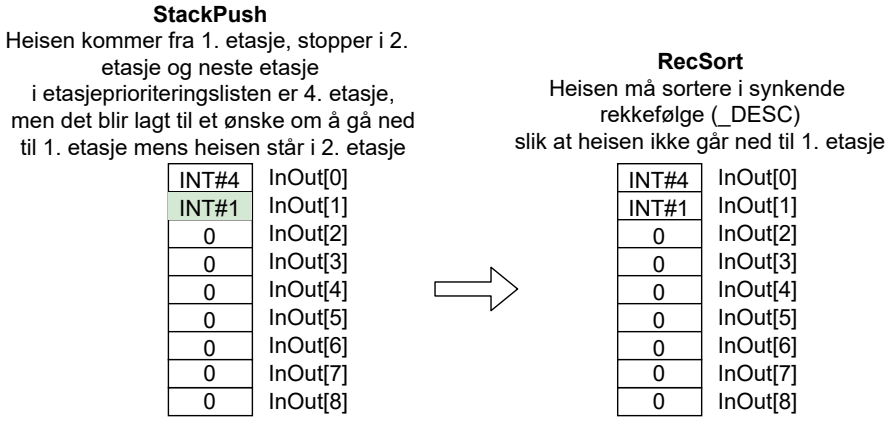
3. Når sorteringa av etasjene er gjort sendes et høyt signal til `StackFIFO`-funksjonsblokken (vist i figur4.17) for å hente en ny verdi som legges i `tilstand_out` variabelen på kodelinje 49, som igjen setter en ny tilstand til case-strukturen. Da vil en ny tilstand begynne heisen går til en ny etasje.

I kodeutklipp 4.0 på kodelinje 42 vises det at programmet sorterer etasjeprioriteringslisten i stigende rekkefølge når heisen er i 1. etasje. Det er tilsvarende sorteringslogikk i alle etasjer. I tabell 4.1 vises det hvilken rekkefølge etasjeprioriteringslisten sorteres i de forskjellige etasjene. Det er brukt to `RecSort`-funksjonsblokker, der den ene sorterer i stigende rekkefølge og den andre sorterer i synkende rekkefølge.

Tabell 4.1: Tabell som viser sorteringslogikk i etasjene.

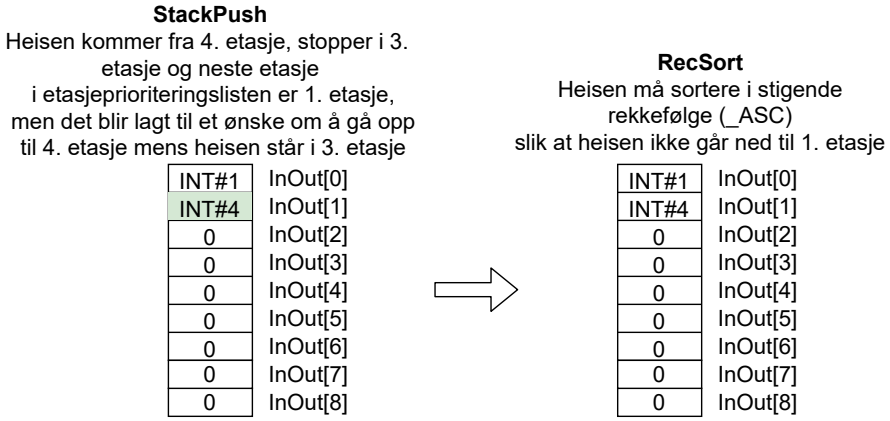
	Heis på vei opp	**Heis på vei ned**	**Heis står i ro**
1. Etasje	Stigende (`_ASC`)	-	Stigende (`_ASC`)
2. Etasje	Stigende (`_ASC`)	Stigende (`_ASC`)	Synkende (`_DESC`)
3. Etasje	Synkende (`_DESC`)	Synkende (`_DESC`)	Stigende (`_ASC`)
4. Etasje	-	Synkende (`_DESC`)	Synkende (`_DESC`)

Spesialtilfelle i 2. etasje: Når heisen stopper i 2. etasje før den skal opp til 4. etasje og en bruker legger til et ønske om 1. etasje, så vil 1. etasje bli utført før 4. etasje om heisen sorterer etasjene i stigende rekkefølge som i tabell 4.1. Dette er ikke en effektiv løsning. Derfor legges det til en betingelse i tillegg til betingelsen *Heis på vei opp* fra tabell 4.1 som gjør at heisen ikke sorterer i stigende, men synkende rekkefølge om etasje 1 blir aktivert i denne situasjonen. Dette er illustrert i figur 4.20.



Figur 4.20: Spesialtilfelle i 2. etasje. Grønn celle er ny verdi lagt til av `StackPush`-funksjonsblokken

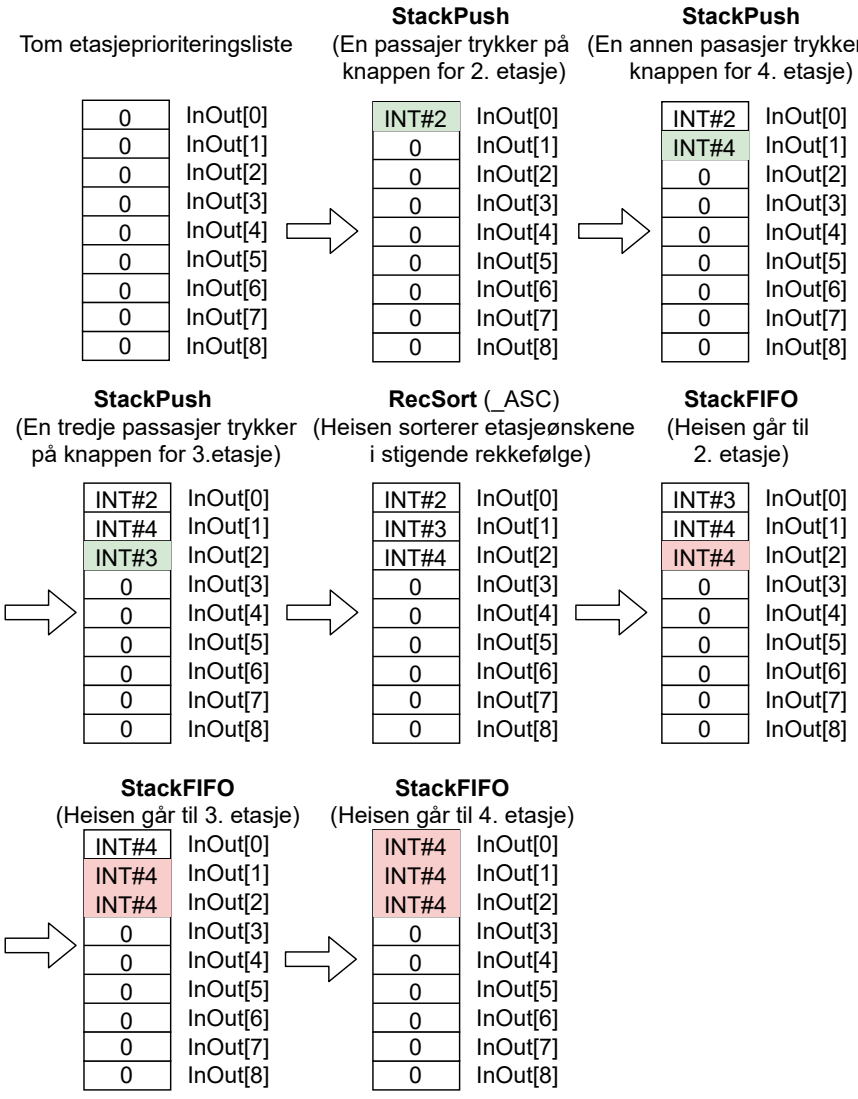
Spesialtilfelle i 3. etasje: Når heisen stopper i 3. etasje før den skal ned til 1. etasje og en bruker legger til et ønske om 4. etasje, så vil 4. etasje bli utført før 1. etasje om heisen sorterer etasjene i synkende rekkefølge som i tabell 4.1. Dette er ikke en effektiv løsning. Derfor legges det til en betingelse i tillegg til betingelsen *Heis på vei ned* fra tabell 4.1 som gjør at heisen ikke sorterer i synkende, men stigende rekkefølge om etasje 1 blir aktivert i denne situasjonen. Dette er illustrert i figur 4.21



Figur 4.21: Spesialtilfelle i 3. etasje. Grønn celle er ny verdi lagt til av `StackPush`-funksjonsblokken

4.3.6 Eksempel ved bruk av etasjeprioriteringslisten

I figur 4.22 er et eksempel på hvordan heisen prioriterer etasjer når den står i 1. etasje og tre passasjerer skal til forskjellige etasjer.

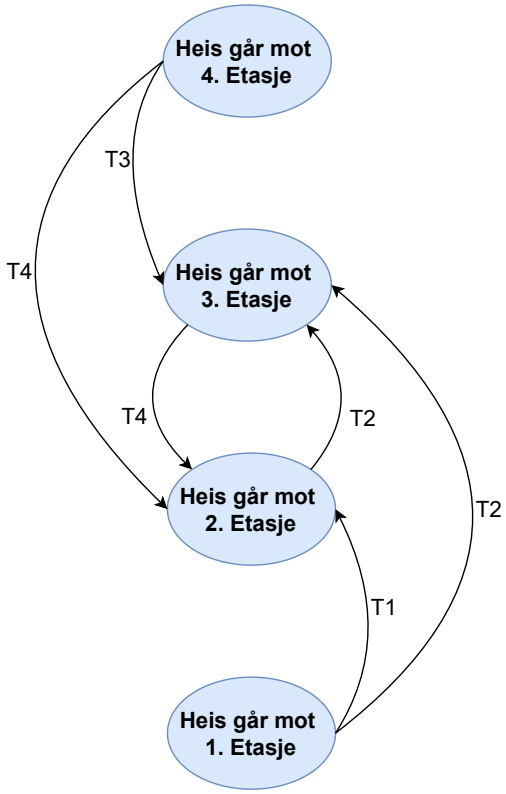


Figur 4.22: Eksempel på hvordan heisen legger til, sorterer og tar ut `INT#` verdier fra etasjeprioriteringslisten. De grønne cellene er nye verdier og de røde er ugyldige verdier som blir overskrevet om man legger inn nye verdier.

4.4 Overstyring av etasjeprioriteringslisten når heisen er i bevegelse

Når heisen er stasjonær vil etasjeprioriteringslisten beskrevet i kapittel 4.3 bli brukt til å styre heisen. Derimot når heisen er i bevegelse vil ikke et ny etasje-forespørsel bli lagt til før heisen er stasjonær igjen ved bruk av etasjeprioriteringslisten. Det er derfor nødvendig å overstyre etasjeprioriteringslisten når heisen er i bevegelse, slik at det er mulig å velge etasjer selv om heisen er i bevegelse.

For å vise hvordan dette er løst har vi i figur 4.23 laget et flytskjema som viser hvordan etasjeprioriteringslisten overstyres når heisen er i bevegelse.

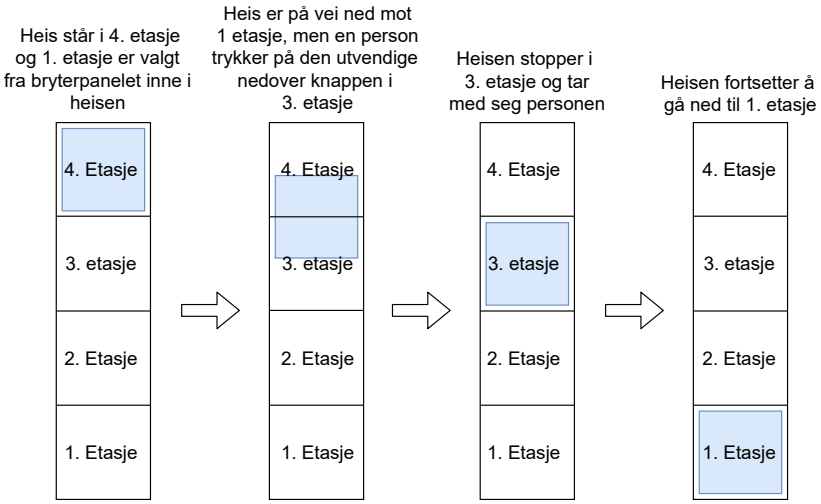


Figur 4.23: Flytskjema med transisjoner vist i tabell 4.2

Tabell 4.2: Transisjoner til flytskjema i figur 4.23

Transisjon	Krav til transisjon
T1	Positiv retning & heis i bevegelse & faktisk posisjon <400 & (etasje 2 utvendig opp knapp / etasje 2 innvendig knapp)
T2	Positiv retning & heis i bevegelse & faktisk posisjon <800 & (etasje 3 utvendig knapp opp / etasje 3 innvendig knapp)
T3	Negativ retning & heis i bevegelse & faktisk posisjon >800 & (etasje 3 utvendig ned knapp / etasje 3 innvendig knapp)
T4	Negativ retning & heis i bevegelse & faktisk posisjon >400 & (etasje 2 utvendig ned knapp / etasje 2 innvendig knapp)

Ut fra flytskjema vist i figur 4.23 og tabell 4.2 kan man se at dersom heisen er på vei mot en etasje og betingelsene gitt i tabellen er oppfylt så vil heisen stoppe innom aktuell etasje. For eksempel om heisen går ned mot 1. etasje fra 4. etasje og heisen har en posisjon som er mindre enn 800 (der den faktiske posisjonen til heisen blir hentet fra servodriveren med koden `MC_Axis000.Act.Pos` som vist i kodelinje 147 i kodeutklipp 4.1). Om den utvendige nedoverknappen i etasje 3 blir aktivert, så vil heisen stoppe og ta med seg passasjeren i 3. etasje før den fortsetter til 1. etasje som vist i figur 4.24.



Figur 4.24: Eksempel på hvordan utvendige knapper overstyrer etasjeprioriteringslisten når heisen er i bevegelse. Heisvognen representeres av den blå firkanten.

4.4 Overstyring av etasjeprioriteringslisten når heisen er i bevegelse

Denne funksjonen er realisert ved bruk av en `IF`-løkke som vist i kodeutklipp 4.1. `INT#` verdiene til etasjene blir da ikke lagt inn i etasjeprioriteringslisten, men knappen til den ønskede etasjen vil ligge høy til `IF`-løkkens krav tilfredstilles og dermed skiftes tilstand.

Kode 4.1: Kode i 1. etasje: Kodelinje 15 - 19 sjekker om nedover knappen eller innvendig knapp til 3. etasje er aktiv og sender heisen innom 3. etasje om en av knappene er aktiv og heisen er over posisjonen (800) til 3 etasje og heisen går nedover. Sjekker også det samme for 2. etasje i kodelinje 20 - 24

```
14      // Heisen sjekker om betjeningsknapper blir trykket ...
            naar heisen er i bevegelse
15              IF MC_Axis000.Dir.Nega AND MC_Axis000.Act.Pos ...
                    > 800 AND (etasje_3_utvendig_ned OR ...
                    etasje_3_innvendig) THEN
16                  servodriver_m1_utfor := FALSE;
17                  timer_start:=FALSE;
18                  timer_reset:=TRUE;
19                  tilstand := 3;
20              ELSIF MC_Axis000.Dir.Nega AND ...
                    MC_Axis000.Act.Pos > 400 AND ...
                    (etasje_2_utvendig_ned OR ...
                    etasje_2_innvendig) THEN
21                  servodriver_m1_utfor := FALSE;
22                  timer_start:=FALSE;
23                  timer_reset:=TRUE;
24                  tilstand := 2;
25              END_IF;
```

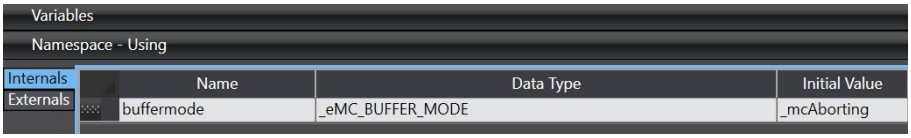
I kodeutklipp 4.1 vises det hvordan overstyringen av etasjebrioriteringslisten er løst når heisen går mot 1. etasje. Det vil væretilsvarende logikk i de andre etasjene, som vist i tabell 4.3.

4.4 Overstyring av etasjeprioriteringslisten når heisen er i bevegelse

Tabell 4.3: Hvordan etasjeprioriteringslisten blir overstyrt når heisen går mot forskjellige etasjer. I parantesene er koden som blir brukt i Sysmac Studio for tillhørende beskrivelse.

Heis går mot	Retning til heis	Posisjonbegrensning	Knapper	Stopp innom
1. Etasje	Nedover (`MC_Axis000.Dir.Nega`)	Over 800mm (`MC_Axis000.Act.Pos >800`)	Utvendig nedoverknapp i 3. etasje eller knapp for 3. etasje inne i heis (`etasje_3_utvendig_ned OR etasje_3_innvendig`)	3. Etasje
1. Etasje	Nedover (`MC_Axis000.Dir.Nega`)	Over 400mm (`MC_Axis000.Act.Pos >400`)	Utvendig nedoverknapp i 2 etasje eller knapp for 2, etasje inne i heis (`etasje_2_utvendig_ned OR etasje_2_innvendig`)	2. Etasje
2. Etasje	Nedover (`MC_Axis000.Dir.Nega`)	Over 800mm (`MC_Axis000.Act.Pos >800`)	Utvendig nedoverknapp i 3. etasje eller knapp for 3. etasje inne i heis (`etasje_3_utvendig_ned OR etasje_3_innvendig`)	3. Etasje
3. Etasje	Oppover (`MC_Axis000.Dir.Posi`)	Under 400mm (`MC_Axis000.Act.Pos <400`)	Utvendig oppoverknapp i 2 etasje eller knapp for 2. etasje inne i heis (`etasje_2_utvendig_opp OR etasje_2_innvendig`)	2. Etasje
4. Etasje	Oppover (`MC_Axis000.Dir.Posi`)	Under 400mm (`MC_Axis000.Act.Pos <400`)	Utvendig oppoverknapp i 2 etasje eller knapp for 2. etasje inne i heis (`etasje_2_utvendig_opp OR etasje_2_innvendig`)	2. Etasje
4. Etasje	Oppover (`MC_Axis000.Dir.Posi`)	Under 800mm (`MC_Axis000.Act.Pos <800`)	Utvendig oppoverknapp i 3. etasje eller knapp for 3. etasje inne i heis (`etasje_3_utvendig_opp OR etasje_3_innvendig`)	3. Etasje

For å overstyre prioriteringslisten er det brukt en funksjon som gjør at `MC_move`-funksjonsblokken avbryter instruksjoner ved en ny instruksjon for så å fulføre den gamle instruksen etterpå. Denne funksjonen heter `_mcAborting` og legges inn som en *internal value* i en variabel i Sysmac Studio med datatype `_eMC_BUFFER_MODE` som i figur som vist i figur 4.25.



Figur 4.25: legge til `_mcAborting` i variabelen kalt `buffermode`

Variabelen som blir generert i figur 4.25 knyttets til `BufferMode`-Inngangen på `MC_move`-funksjonsblokken fra vedlegg D.2.

4.5 Styring av servodrivere og servomotorer for heis

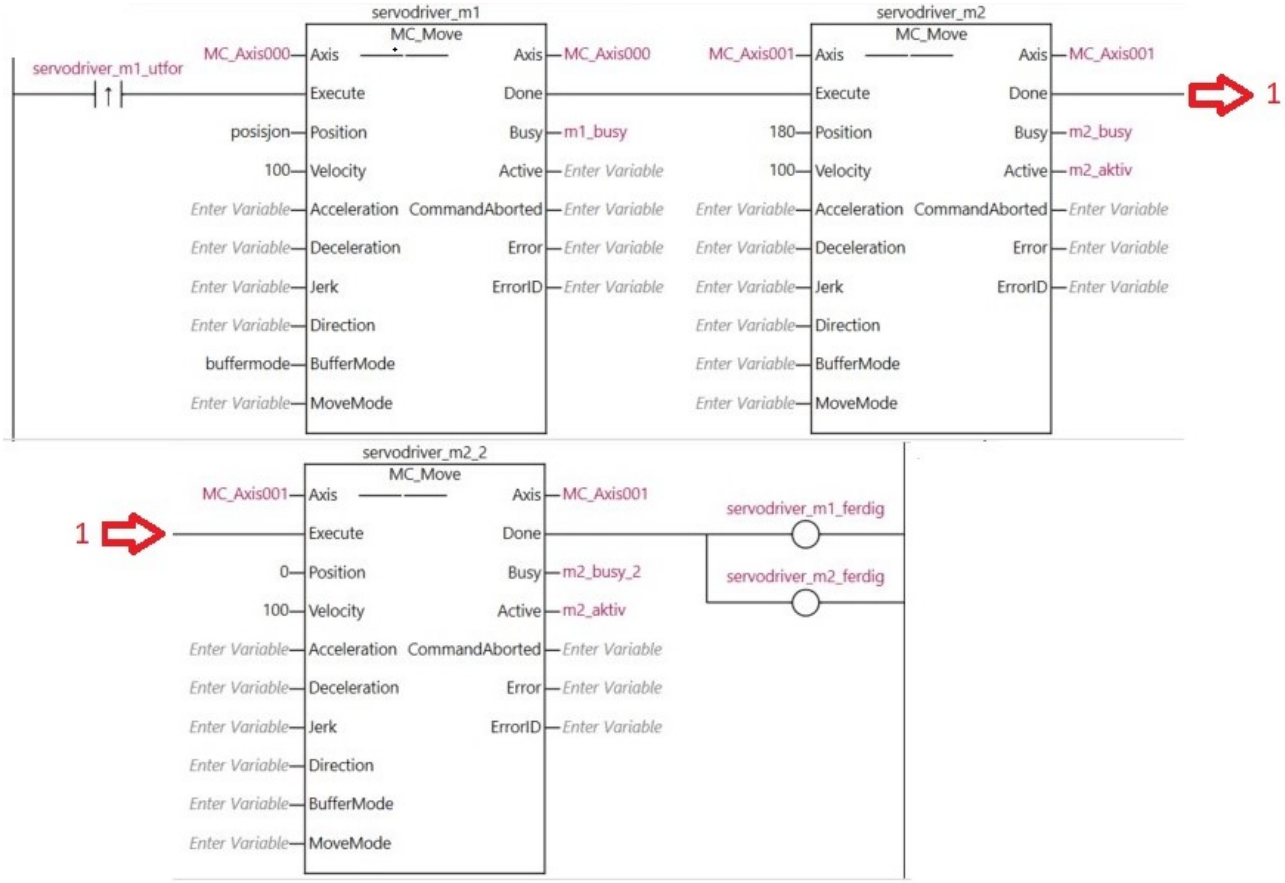
I begynnelsen av hver tilstand blir det satt en posisjon, som vist i kodelinje 139 i kodeutklipp 4.2 og et *enable*-signal, som vist i kodelinje 140. *Enable*-signalet kjører heisen til gitt posisjon. Instruksjonen blir sendt til `MC_MOVE`-funksjonsblokken til `servodriver_m1`, som er vist i figur 4.26. Mer om `MC_Move`-funksjonsblokken vise i vedlegg D.2.

Kode 4.2: I kodeutklippet vises det hva som utføres i tilstand 4. Alle tilstander ser helt like ut med unntak av forskjellige posisjoner og sorterings betingelser.

```
136    4: // Tilstand: 4. etasje (heisen gaar mot/ er i 4 etasje)
137
138               // Heisen sendes til 4. etasje
139               posisjon := 1200;
140               servodriver_m1_utfor := TRUE;
141               sorter_stigende := FALSE;
142               sorter_synkende := FALSE;
143               ny_tilstand := FALSE;
144               timer_reset:=FALSE;
```

Døren blir styrt av ladderkoden på figur 4.26. Når en ny tilstand blir satt i hovedkoden sendes en posisjon til `Position`-inngangen på `MC_move`-funksjonsblokken og et høyt signal til `Execute`-inngangen på `MC_Move`-funksjonsblokken til `servodriver_m1`. Da kjøres heisen til posisjonen som ble sendt. Når denne instruksen er fullført vil den samme blokka gi ut et høyt signal på `Done`-utgangen som er knyttet til `Execute` på `MC_Move`-funksjonsblokken til `servodriver_m2`. Da kjører servomotor 2 180 grader, som simulerer at døra åpner. Når denne blokka har fulført instruksen vil det gå et høyt signal fra `Done` på `MC_Move`-funksjonsblokken til servodriver 2 til `Execute` på `servodriver_m2_2` som kjører servomotor 2 tilbake til utgangspunktet. Dette simulerer at døren lukkes.

Her er det også mulig å legge til en timer som holder døra oppe i en satt tid, men vi har her valgt å ikke ta med en forsinkelse på døra. Da dette gjør systemet en god del tregere.



Figur 4.26: Kode for å kjøre heis og dør. Pilene markert med 1 viser hvor rungen fortsetter på figuren.

4.6 Forslag til lab

For at studentene i faget ELE310 Styringsteknikk skal bli kjent med *HMI*-delen av Sysmac Studio har vi laget et forslag til en laboppgave hvor studentene selv skal opprette og knytte knapper, lamper og andre komponenter til heisstyrings-koden fra dette kapittelet. Oppgaven kan løses hjemme eller på lab.

1. Opprett en *HMI*-skjerm i Sysmac Studio ved å følge veiledningen i vedlegg D.3.
2. Legg til de innvendige knappene som høre til heisen. Variablene som er knyttet til de forskjellige knappene vises i tabell 4.4.

Tabell 4.4: Innvendige knapper i *HMI*-skjerm.

Navn på knapp	**Type knapp**	**Variabel**
1. Etasje	Set button	PLS_etasje_1_innvendig
2. Etasje	Set button	PLS_etasje_2_innvendig
3. Etasje	Set button	PLS_etasje_3_innvendig
4. Etasje	Set button	PLS_etasje_4_innvendig
åpne dør	Momentary button	PLS_apne_dor

3. Legg til de utvendige knappene som er i etasjene utenfor heisen. Variablene som er knyttet til de forskjellige knappene vises i tabell 4.5.

Tabell 4.5: Utvendige knapper i *HMI*-skjerm.

Navn på knapp	Type knapp	Variabel
1. Etasje utvendig opp	Set button	PLS_etasje_1_utvendig_opp
2. Etasje utvendig opp	Set button	PLS_etasje_2_utvendig_opp
2. Etasje utvendig ned	Set button	PLS_etasje_2_utvendig_ned
3. Etasje utvendig opp	Set button	PLS_etasje_3_utvendig_opp
3. Etasje utvendig ned	Set button	PLS_etasje_3_utvendig_ned
4. Etasje utvendig opp	Set button	PLS_etasje_4_utvendig_ned

4. Legg til knapper som styrer aktiverer/ deaktiverer, og lampene tilknyttet. Variablene som er knyttet til de forskjellige komponentene i *HMI*-skjermen vises i tabell 4.6.

Tabell 4.6: Variabler knyttet til servodriver 1 og servodriver 2.

Navn på knapp	Type knapp	Variabel
Start	MomentaryButton	PLS_start_1
Stop	MomentaryButton	PLS_ stop_1
Homing	MomentaryButton	PLS_homing
Reset	MomentaryButton	PLS_reset_servodriver
Start lampe	BitLamp	PLS_MC_Axis000.DrvStatus.ServoOn
Homing lampe	BitLamp	PLS_MC_Axis000.Details.Homed
Posisjon til heis	DataDisplay	PLS_MC_Axis000.Act.Pos

5. Legg til sanntidsanimasjon av heismodellen. Hvordan sanntidsmodellen opprettes er beskrevet i vedlegg D.3.

6. Last opp *HMI*-skjermen og heisprogrammet og test programmet.

Kapittel 5

Momentstyring av servomotorer

I denne videoen blir dette prosjektet demonstrert

5.1 Motivasjon for prosjekt

Dette prosjektet demonstrerer hvordan servomotorer kan styres ved hjelp av momentregulering. Det er valgt en praktisk tilnærming for å demonstrere dette. Oppgaven beskriver hvordan høyden til en person måles ved å styre servomotorens moment samtidig som det blir tatt avstandsmåling med hjelp avstandssensoren. Formålet med prosjektet er å vise studentene hvordan momentstyring av en servomotor gjøres og å vise hvordan avstandssenoren har blitt implementert i heismodellen ved hjelp av en *EtherCat*-modul med påmontert AD-kort i styreskapet. I dette prosjektet skal studentene oppgi et referansemoment servomotoren skal oppnå ved å presse den utstikkende plastplaten (som er en del av heisvognen) ned mot hodet til en person som skal måles. Videre måles avstanden fra heisvognen og ned til gulvet ved hjelp av avstandssensoren, og denne avstanden skaleres slik at den tilsvarer personens høyde. I dette prosjektet omtales prosessen fra bruker starter målingen og til den er fullført til som målesekvensen.

5.1 Motivasjon for prosjekt

Figur 5.1 viser et bilde av en person som måler høyden sin ved hjelp av heismodellen.

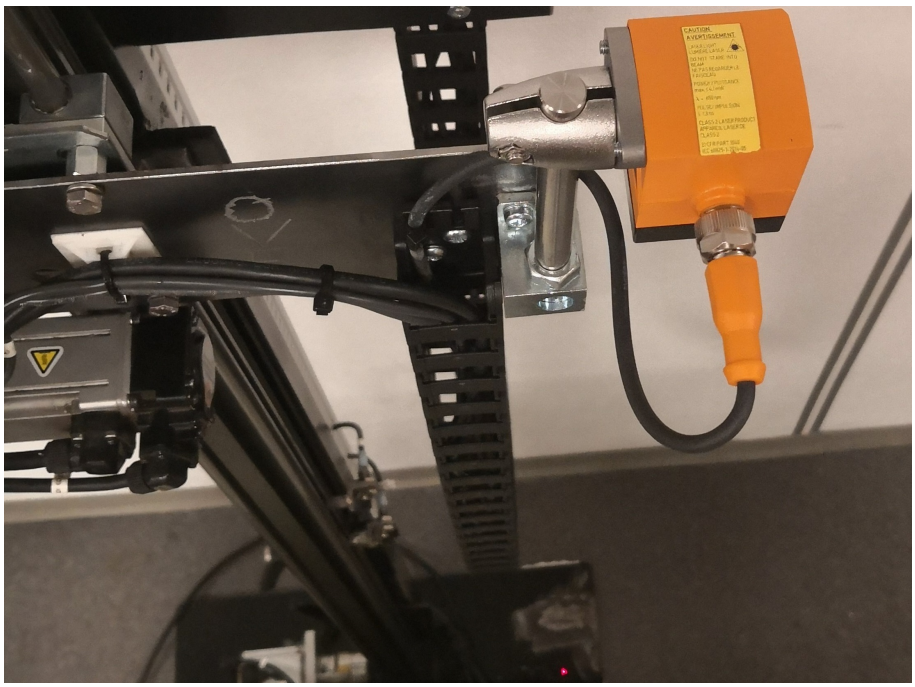
Figur 5.1: I figuren vises heismodellen når en person bruker den for å måle høyde.

Presentasjonen av prosjektet er delt det opp i følgende deler:

1. Presentasjon av hvordan avstandssensoren er implementert, som vist i kapittel 5.2.
2. Hvordan programmet brukes og hvordan *HMI*-skjermene er koblet sammen, som vist i kapittel 5.3.
3. Tilstandsdiagram, som vist i kapittel 5.4.
4. Grafisk fremstilling av en målesekvens, som vist i kapittel 5.5.
5. Forslag til laboppgave, som vist i kapittel 5.6.

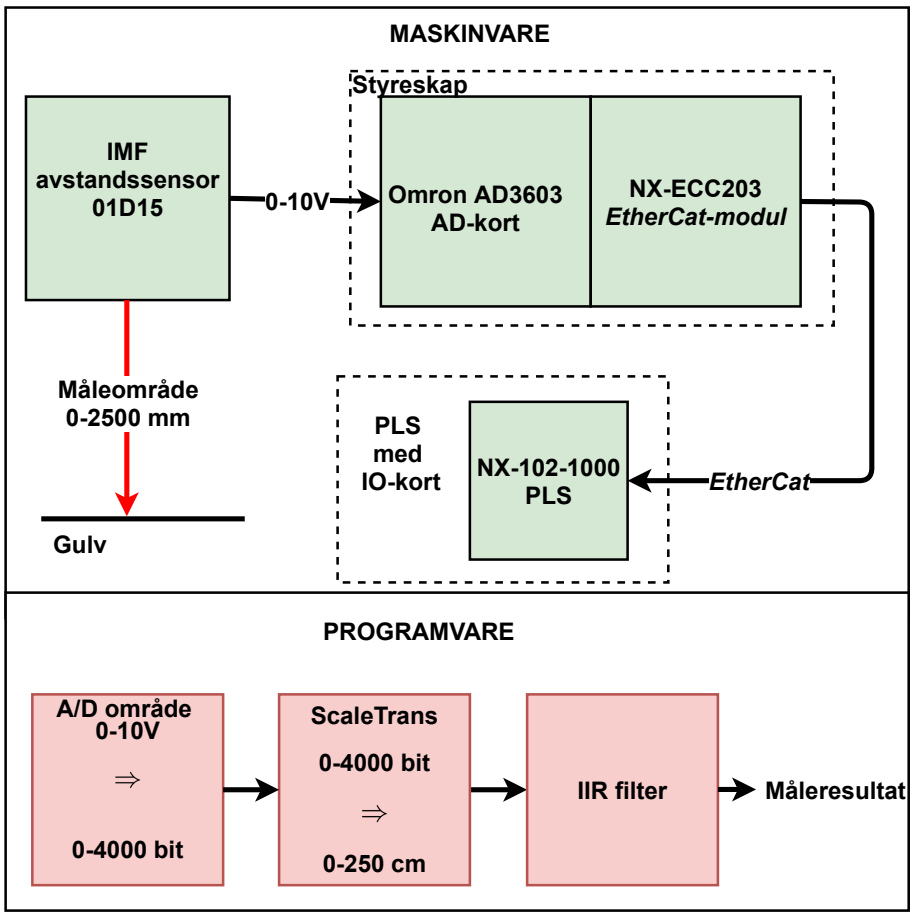
5.2 Implementering av avstandssensor

Dette prosjektet benytter seg av avstandssensoren 01D15 fra IFM oppført i tabell 2.1 for måling av avstand. Det er mulig å måle avstand ved å lese posisjonen gitt av enkoderen til servomotor 1, , men det ble valgt å implementere avstandssensoren i heismodellen for å vise hvordan ett analogsignal (avstandsmåling fra avstandssensor) ved hjelp av AD-kort montert på *EtherCat*-modul kobles til PLS for å så skalere signal slik at det kan avlses i programmet som en høyde. Avstandssensoren er montert på heisvognen og måler avstanden fra heisvognen og ned til gulvet som vist i figur 5.2.



Figur 5.2: I figuren vises avstandssensoren montert på heisvognen

Sensoren er konfigurert slik at den leverer et analogt signal i området $[0, 10]V$ til AD-kortet. Dette spenningsområdet tilsvarer en høyde i området $[0, 2500]mm$. Dette signalet konverteres i AD-kortet til et digitalt signal med oppløsning i området $[0, 4000]$. I programmet skaleres signalet fra AD-kortet ved hjelp av `ScaleTrans`-funksjonen (vist i kodeutklipp 5.1) om til en høyde oppgitt i $[cm]$. På grunn av støy i spenningssignalet fra avstandssensoren så anvendes ett *IIR*-filter (gitt som eksempel i vedlegg D.4) for å redusere støykomponentene i signalet. Støyet oppstår på grunn av at avstandssensorens kabel er lagt sammen med servomotorenes strømkabler i kabelføringene i heismodellen.



Figur 5.3: Figuren viser en oversikt over signalgangen i maskin- og programvare for IFM sensoren.

Kodeutklipp 5.1 viser hvordan signalet fra AD-kortet blir skalert. Signalet skaleres ved hjelp av `ScaleTrans`-funksjonen fra Sysmac Studio biblioteket. På kodelinje 4 og 6 gis størrelsen på innsignalet som her er digitalverdiene i 0 og 4000. På kodelinje 5 og 7 gis størrelsen på utsignalet som her er personhøyden som er valgt til å være melllom 1 og 250 cm. På kodelinje 8 vises det at signalet blir forskøvet med 3 cm, fordi dette er den høydedifferansen mellom den utstikkende plastplaten og avstandssensoren.

Kode 5.1: Kodeutklippet viser bruk av `ScaleTrans` funksjonen for å skalere data. Funksjonen finnes i Sysmac Studio sitt funksjons bibliotek.

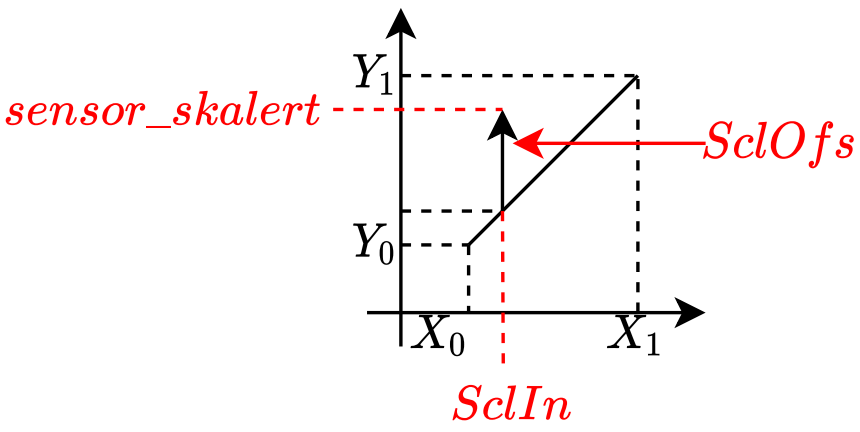
```
1     // Laser sensor, skalering og filtrering
2  sensor_skalert:= ScaleTrans(
3                               SclIn:=N7_Ch1_Analog_Input_Value,
4                               X0:=0,
5                               Y0:=0,
6                               X1:=4000,
7                               Y1:=250,
8                               SclOfs:=-3);
```

`ScaleTrans`-funksjonsblokken tar inn verdiene som vist i kodeutklipp 5.1 og setter disse inn i ligning

I ligning (5.1) vises det hvordan `ScaleTrans`-funksjonen skalerer det digitale signalet.

$$sensor\_skalert = \frac{Y1 - Y0}{X1 - X0} \cdot (sclIn - X0) + Y0 + Scl\,Ofs \qquad (5.1)$$

I figur 5.4 vises det i graf hvordan ligning (5.1) skalere det digitale inngangs-signalet fra AD-kortet for å gi ut personhøyde i cm.



Figur 5.4: I figuren vises det hvordan skaleringsfunksjonen `ScaleTrans` skalerer det digitale signalet fra AD-kortet. Utsignalet fra `ScaleTrans`-funksjonen er personens høyde oppgitt i cm.

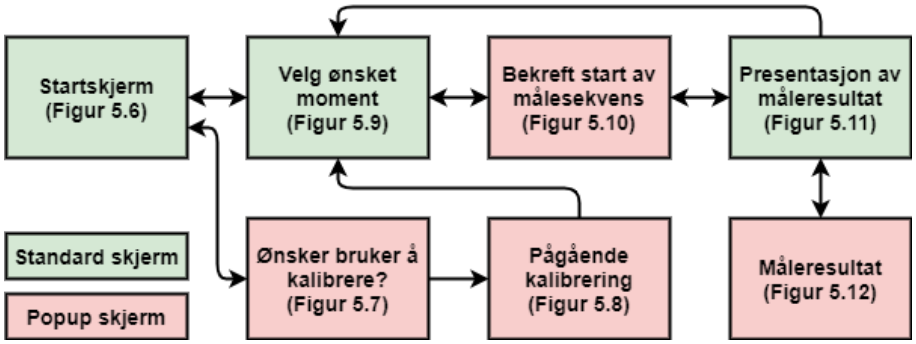
5.3 Programfunksjonalitet og bruk av *HMI*-skjerm

For å presentere hvordan programmet fungerer og hvordan det betjenes gis det videre en beskrivelse av programflyten, og de forskjellige *HMI*-skjermene presenteres i videre delkapitler.

Brukeren navigerer mellom de ulike *HMI*-skjermene for å betjene programmet. Via *HMI*-skjermene er det mulig å aktivere og kalibrere systemet, spesifisere referansemoment, starte målesekvensen og lese av resultatet av målesekvensen. Når målesekvensen starter vil brukeren være plassert rett under den bevegelige plastplaten slik at den treffer på toppen av hodet. Momentet servomotoren utøver på personens hode vil øke helt til det når det spesifiserte referansemomentet. Det er mulig å spesifisere referansemomentet i området $[10, 30]\%$. Momentet er oppgitt i % og er basert på den maksimale effekten servomotoren kan få tilført fra servodriveren (for servomotor 1 er dette 400 W). Det er valgt at det kun kan spesifiseres referansemoment $\leq 30\%$ fordi høyere enn dette vil være ubehagelig for personen som blir målt, og referansemomentet må være $\geq 10\%$ fordi dette er momentet som kreves for at servomotoren klare å bevege heisvognen. Når servomotoren har holdt samme moment som det spesifiserte referansemomentet i fire sekunder vil målingen fra avstandssensoren bli lagret som en variabel i programmet, og heisvognen går tilbake til samme posisjon som den var i før målingen startet. Det er valgt en tid på 4 sekunder slik at personen som måler høyden sin får tid til å kjenne på momentet som presser mot hodet. Måleresultatet blir presentert for brukeren på *HMI*-skjermen vist i figur 5.12.

5.3 Programfunksjonalitet og bruk av *HMI*-skjerm

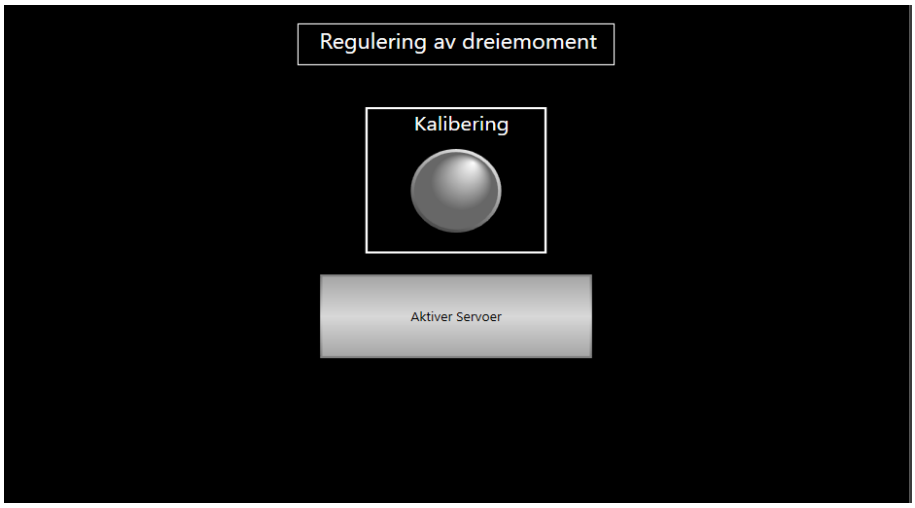
I figur 5.5 vises det hvordan de forskjellige *HMI*-skjermene er koblet sammen. *HMI*-skjermene gjør det enklere og mer oversiktlig for bruker å betjene programmet.



Figur 5.5: Figuren viser hvordan de ulike skjermene er koblet sammen. Grønn rektangel representerer en standard skjerm. Rød skjerm representerer en *popup*-skjerm. Denne legger seg over standard skjermen.

5.3.1 Startskjerm

Ved oppstart av programmet vises skjermen i figur 5.6. Lampen markert med **Kalibrering** indikerer om kalibreringsprosedyren er utført (kalibreringsprosedyren er konfigurert som beskrevet i vedlegg D.5). Ved å betjene knappen **Aktiver Servoer** aktiverer man servodriverne til servomotor 1 og servodriver 2 ved hjelp av `MC_Power`-funksjonsblokken som vist i vedlegg D.2. *Popup*-skjermen "Ønsker bruker å kalibrere" hvor brukeren starter kalibreringsprosedyren (vist i kapittel 5.3.2) vises etter at servodriverne er aktivert.



Figur 5.6: I figuren vises oppstarts skjermen til programmet. På skjermen aktiveres servodriverene av bruker.

5.3.2 Ønsker bruker å kalibrere?

Skjermen vist i figur 5.7 åpnes etter at servodriverne er aktivert. I tekstfeltet får brukeren informasjon om at alle hindringer må fjernes før kalibreringsprosedyren kjøres. Brukeren får også spørmål om å bekrefte start av kalibreringsprosedyren. Ved å betjene knappen **Ja** aktiveres `MC_Home`-funksjonsblokken som starter kalibreringsprosedyren. Ved å trykke **Nei** lukkes skjermen og "startskjermen" i kapittel 5.3.1 vises. Det er valgt å implementere denne skjermen for å gi brukeren informasjon om hva som utføres når brukeren trykker **Ja** og hvilke forhåndsregler som tas siden studenten i dette prosjektet skal ha fysisk kontakt med de bevegelige delene i heismodellen.

Figur 5.7: I figuren vises skjermen hvor bruker kan starte kalibreringsprosedyren for servomotor 1 og servomotor 2. Merk at dette er en *popup*-skjerm.

5.3.3 Pågående kalibrering

Når kalibreringsprosedyren pågår vises skjermen i figur 5.8. Dette er for å informere brukeren om at servomotorene kalibreres. Skjermen lukkes når kalibreringsprosedyren er fullført. Skjermen "Velg ønsket moment" fra kapittel 5.3.4 blir så åpnet.

Figur 5.8: Når kalibreringsprosedyren pågår vil brukeren bli informert via skjermen vist i denne figuren. Merk at dette er en *popup*-skjerm.

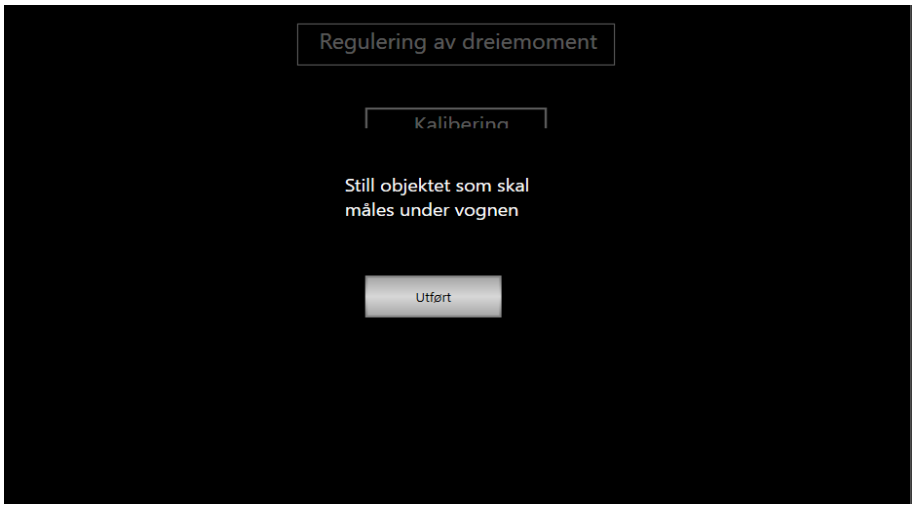
5.3.4 Velg ønsket moment

I det hvite feltet på skjermen vist i figur 5.9 spesifiserer bruker ønsket moment (videre omtalt som referansemoment) under måling av høyde. Momentet blir oppgitt i prosent % av maksimalt moment. Det er kun verdier i området $[10, 30]$ % som blir akseptert som en gyldig verdi. Dette er på grunn av at servomotoren trenger ett moment $\geq 10\%$ for å kunne flytte heisvognen, og for at momentet ikke skal bli så stort at det er ubehagelig for personen som blir målt. Ved å variere verdien på ønsket referansemoment kan brukeren observere/føle servodriverens evne til å regulere ulike moment. Ved å trykke på Knappen **Klar for måling** vises skjermen "Bekreft start av målesekvens" fra kapittel 5.3.5. Ved å trykke knappen **Lukk** i nederste venstre hjørne vises "Startskjermen" fra kapittel 5.3.1.

Figur 5.9: På skjermen vist i figuren spesifiserer bruker referansemoment som servomotor 1 skal bruke til å utføre høydemåling. Det er mulig å spesifisere moment i området $[10, 30]$ % av maksimalt moment.

5.3.5 Bekreft start av målesekvens

Formålet med denne skjermen er å informere brukeren om at personen som skal måles må nå stilles under den bevegelige plastplaten før målesekvensen starter. Knappen **Utfør** starter målesekvensen og åpner "Presentasjon av måleresultater" skjermen i kapittel 5.3.6.

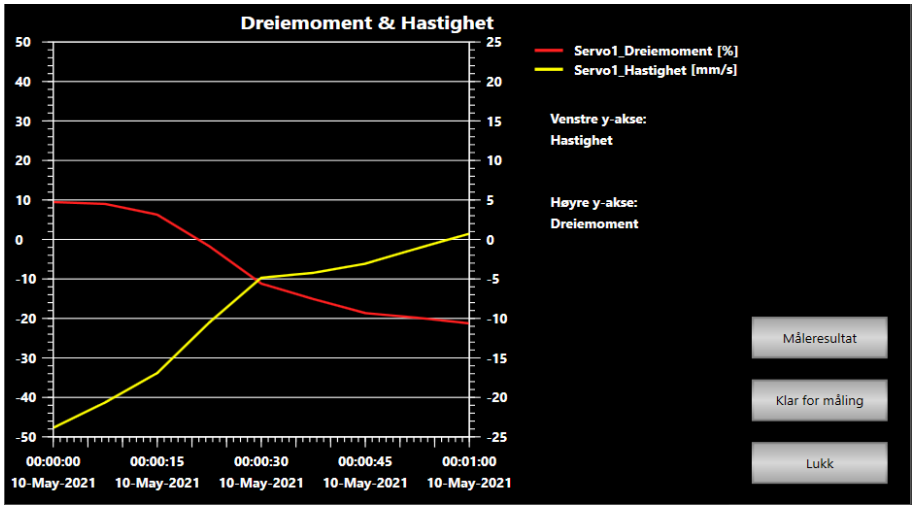


Figur 5.10: I denne figuren vises skjermen hvor brukeren bekrefter start av målesekvens. Merk at dette er en *popup*-skjerm.

5.3.6 Presentasjon av måleresultater

Når målesekvensen har startet presenteres skjermen vist i figur 5.11. Momentet og hastigheten til servomotor 1 vises i grafen på venstre side av figur 5.11. Grafen er opprettet som vist i vedlegg D.3. Dette gjør at brukeren kan observere momentet og hastigheten til servomotor 1 under utførelse av målesekvensen og etter at målingen er utført. Forklaring til hvilke verdier som er plottet mot y-aksen vises i figurteksten til høyre for plottet. Skjermen har også tre knapper nede i høyre hjørne forklart i listen under.

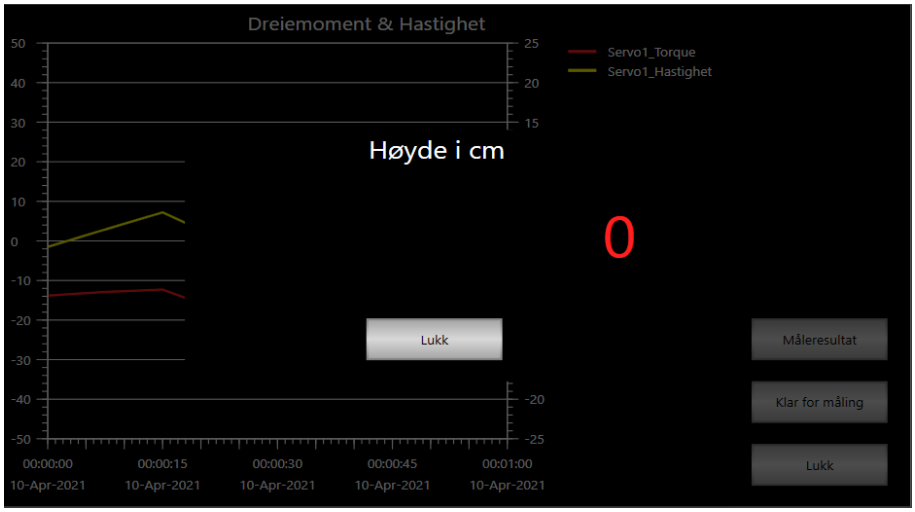
1. **Måleresultat** viser skjermen "Måleresultat" fra kapittel 5.3.7 som presenterer siste registrerte måleresultat.
2. **Klar for måling** viser skjermen "Bekreft start av målesekvens" fra kapittel 5.3.5 hvor man starter målesekvensen.
3. **Lukk** åpner skjermen "Velg ønsket moment" som vist i kapittel 5.3.4 hvor man spesifisere hvilket moment servomotor 1 skal kjøre med.



Figur 5.11: Under målesekvensen blir moment og hastighet logget i sanntid og presentert grafisk i *HMI*. *HMI*-skjermen vist i figuren er hentet fra Sysmac Studio og viser tilfeldige data.

5.3.7 Måleresultat

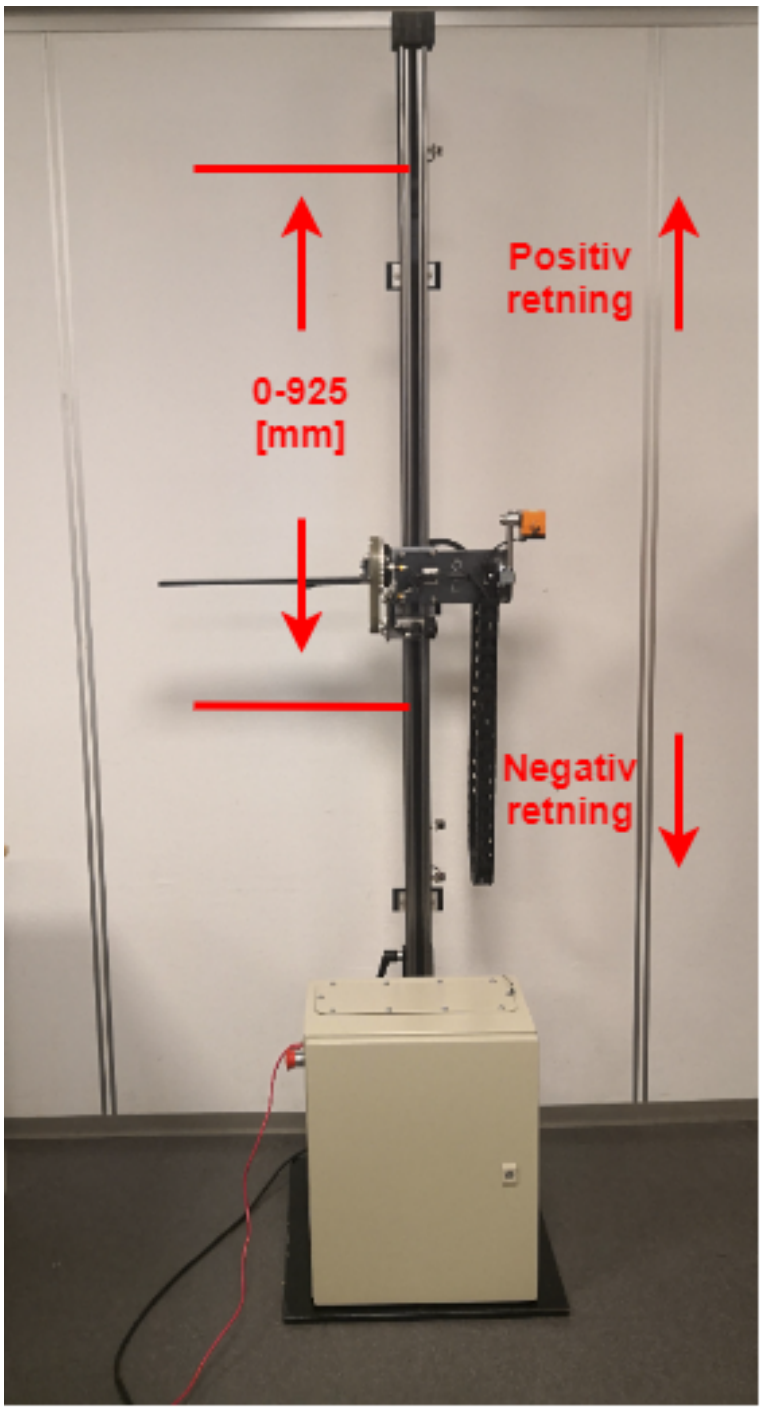
Når målesekvensen er fullført presenteres måleresultatet som vist på skjermen i figur 5.12. Dette er den skalerte verdien fra avstandssensoren beskrevet i kapittel 5.2 og representerer høyden til personen som har målt seg i cm.



Figur 5.12: Skjermen i figuren åpnes når målesekvensen er ferdig. På skjermen presenteres måleresultatet. Merk at dette er en *popup*-skjerm.

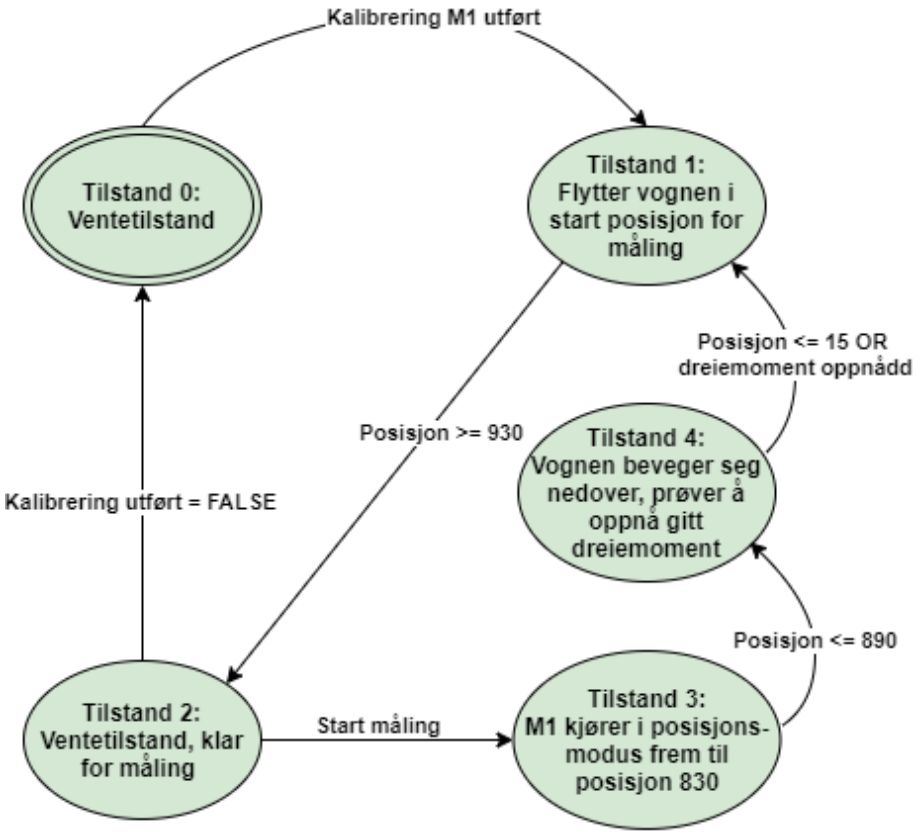
5.4 Tilstandsdiagram og programstruktur

I tilstandsdiagrammet og programkoden brukes det flere forskjellige avstander. For å vise hva de forskjellige tallene brukes til presenteres det først en figur 5.13 med markeringer og tilhørende beskrivelse. Figuren 5.13 viser at området på heismodellen som tilsvarer posisjon $0 \rightarrow 925$ mm i tilstandsdiagrammet i figur 5.14 og videre i kodeutklipp. Under målesekvensen kan heisvognen bevege seg i området $[15, 925]$ mm. Bruksområdet er å måle høyden til en student, derfor er det valgt å begrense bevegelsen til heisvognen til dette området. Det vil da være mulig å gjøre målinger i verdiområdet $[203, 112]$ cm som er tilstrekkelig. Dreieretningen til servomotor 1 er indikert med pilene markert med positiv og negativ retning. Momentet vil også følge disse retningene slik at bevegelse i negativ retning fører til negativt moment.



Figur 5.13: Bildet viser plassering av posisjon på relativt til heismodellen og negativ/positiv dreieretning.

For å vise hvordan programmet er bygd opp har vi i figur 5.14 tegnet et tilstandsdiagram. Videre vises hva som utføres i hver tilstand, og hva som er krav for transisjon mellom tilstandene.



Figur 5.14: Tilstandsdiagram for programmet. Beskrivelse av tilstandene og transisjonen blir gitt i teksten nedenfor.

Videre forklares det hva som utføres i tilstandene, betingelse for transisjon mellom tilstandene og tilhørende kodeutklipp.

Tilstand 0: Denne tilstanden kontrollerer om servomotorene har utført kalibrering i henhold til kalibreringsprosedyren som beskrevet i vedlegg D.5. Hvis kalibreringsprosedyren har blitt utført oppfylles kravet for transisjon til tilstand 1. Kravet for transisjon vises på linje 50 i kodeutklippet 5.2.

Kode 5.2: Kodeutklipp for tilstand 0

```
48      0: //Ventetilstand
49
50          IF Kalibrering_utfort THEN
51              trqStyring := 1;
52          END_IF;
```

Tilstand 1: I denne tilstanden settes servomotor 1 til å bevege heisvognen oppover med maksimalt moment på 100 % og maksimal hastighet på 100 $\frac{mm}{s}$, dette kan ses på kodelinje 57 og 58 i kodeutklipp 5.3. Når servomotor 1 når posisjon $\geq 925\ mm$, oppfylles kravet for transisjon til ventetilstanden, tilstand 2. Dette vises på kodelinje 61 - 63. For bevegelse av heisvognen er `MC_TorqueControl`-funksjonsblokken benyttet, som er vist i vedlegg D.2.

Kode 5.3: Kodeutklipp for tilstand 1

```
54     1: //Flytter vognen til startposisjon
55
56          reset_acctime := TRUE;
57          TrqSetValue_m1 := 100;    // Moment i prosent
58          TrqVelocity_m1 := 100;
59          Exectute_TrqCtrl_m1 := TRUE;
60
61          IF MC_servo1.Act.Pos ≥ 925 THEN
62              Exectute_TrqCtrl_m1 := FALSE;
63              trqStyring := 2;
64          END_IF;
```

Tilstand 2: Hvis servodriverene blir deaktivert vil kalibreringen være ugyldig. Dette er for å skape en ekstra sikkerhet for personen som skal måle seg. Denne tilstanden verifiserer at kalibreringen fortsatt er gyldig før målingen begynner. Dersom kalibrering er ugyldig oppfylles kravene for transisjon tilbake til tilstand 0, her må kalibreringsprosedyren utføres igjen før videre bruk av programmet. Dersom knappen **Klar for måling** betjenes (som vist i figur 5.11 og figur 5.9) oppfylles kravet for transisjon til tilstand 3. Funksjonsblokken `MC_TorqueControl` er benyttet for holde heismodellen stasjonær vist på linje 68-70 i kodeutklipp 5.4 (ved ett moment = 5% vil heisvognen forbli stasjonær).

Kode 5.4: Kodeutklipp for tilstand 2

```
66    2: //Venteposisjon, klar for maaling
67
68          TrqSetValue_m1 := 5;    // Moment i prosent
69          TrqVelocity_m1 := 0;
70          Exectute_TrqCtrl_m1 := TRUE;
71          IF NOT Kalibrering_utfort THEN
72              Exectute_TrqCtrl_m1 := FALSE;
73              trqStyring :=0;
74          ELSIF klar_for_maaling THEN
75              Exectute_TrqCtrl_m1 := FALSE;
76              trqStyring := 3;
77          END_IF;
```

Tilstand 3: Momentet som kreves for at servomotor 1 skal starte bevegelse er større enn momentet som kreves for å vedlikeholde den. Vi ønsker å kunne kjøre på et lavt moment under målesekvensen, derfor er tilstand 3 lagt inn for å starte bevegelser fra stasjonær tilstand for å overvinne startmomentet. Det benyttes en `MC_Move`-funksjonsblokk, som vist i vedlegg D.2 for å oppnå dette. Når heisvognen har beveget seg med $\Delta L = 35mm$ ($\Delta L = 925 - 890 = 35mm$) oppfylles kravet for transisjon til tilstand 4, som fortsetter bevegelsen i negativ retning ved hjelp av momentstyring.

Kode 5.5: Kodeutklipp for tilstand 3

```
79      3: //Servo1 kjorer i negativ retning ved bruk av ...
            posisjonering
80
81          Servo1_Move_Position := 0;
82          Servo1_Move_Velocity := 60;
83          Servo1_Move_Execute := TRUE;
84          IF MC_servo1.Act.Pos ≤ 890 THEN
85              Servo1_Move_Execute := FALSE;
86              trqStyring := 4;
87          END_IF;
```

Tilstand 4: Fra tilstand 3 til tilstand 4 endres styringen av servomotor 1 fra posisjonstyring til momentstyring. Hensikten med å bytte fra posisjonsstyring til momentstyring er at i tilstand 3 skal heisvognen oppnå en fart for å kunne overvinne den stasjonære friksjonen, og i tilstand 4 skal servomotoren forsøke og oppnå et referansemoment. Momentstyring av servomotoren gjøres ved bruk av `MC_TorqueControl`-funksjonsblokken som vist i kodeutklipp 5.6 på linje 93-95. Maksimal hastighet er satt til 30$[\frac{mm}{s}]$ for å begrense momentet som servomotoren klarer å oppnå før den bevegelige plastplaten treffer motstand. Referansemomentet er lagret i variablen `TrqBrukerInput`. Brukeren endrer denne variablen på *HMI*-skjermen i kapittel 5.3.4 når programmet er i tilstand 2. Servomotor 1 oppnår referansemomentet når den bevegelige plastplaten treffer hodet til personen som måles. Når servomoteren har holdt dette momentet i totalt 4 sekunder er kravet for transisjon tilbake til tilstand 1 oppfylt. Som vist på linje 99 i kodeutklipp 5.6 lagres den skalerte måleverdien fra avstandssensoren kalt, `sensor_filtrert`, til variabelen `maalt_hoyde`. Den lagrede måleverdien vises på skjermen i kapittel 5.3.7.

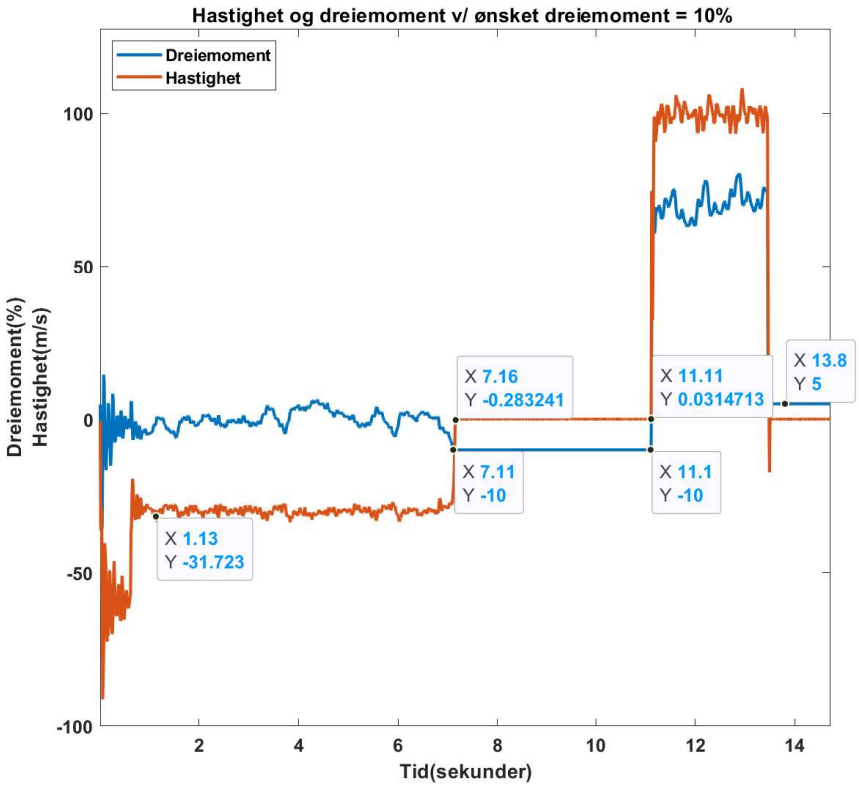
Kode 5.6: Kodeutklipp for tilstand 4

```
90  4: //Maaling startet, leser av maaleverdi fra sensor ...
        naar TrqBrukerInput har vaert hoy i 4s
91
92          reset_acctime := FALSE;
93          TrqVelocity_m1_neg:= 30;
94          TrqSetValue_m1_neg := TrqBrukerInput;
95          Exectute_TrqCtrl_m1_neg := TRUE;
96              IF MC_servo1.Act.Pos ≤ 15 OR Q_0 THEN
97                  Exectute_TrqCtrl_m1_neg := FALSE;
98                  klar_for_maaling := FALSE;
99                  maalt_hoyde := sensor_filtrert;
100                 Vishoyde:= TRUE;
101                 trqStyring := 1;
102             END_IF;
103 END_CASE;
```

5.5 Plot av målesekvens

I dette kapitellet presenterer vi måledata fra en målesekvens i et plot i Matlab for å kunne vise hva som skjer grafisk. Måledataen i dette kapittelet er logget ved og lage et *data trace* i Sysmac Studio som lagres i en `.csv` fil som vist i vedlegg D.6. Måledataen presenteres som et plot i Matlab ved å importere `.csv` filen som vist i vedlegg D.7.

Figur 5.15 viser et plot hvor moment og hastighet er plottet som en funksjon av tid under kjøring av en målesekvens. I dette plottet er referansemomentet spesifisert til 10 %. Datapunktene markert i figuren forklares

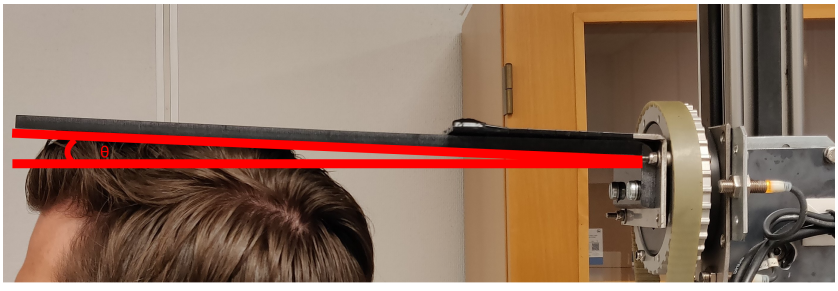


Figur 5.15: I figuren er hastighet og moment plottet som funksjon av tid. Data presentert i plottet representerer data fra en målesekvens.

Punktlisten under utdyper de markerte datapunktene i figur 5.15

- **Tid = 0:** Programmet er i tilstand 3 (se kodeutklipp 5.5). Servomotor 1 opererer i posisjonsmodus med ønsket hastighet = 60 $\frac{mm}{s}$ frem til betingelsen for transisjon er oppfylt.
- **Tid ≈ 1:** Programmet er i tilstand 4 (se kodeutklipp 5.6) hvor hastighetsbegrensningen er 30$\frac{mm}{s}$ og referansemomentet er -10%. Med en hastighet på -30 $\frac{mm}{s}$ vil servomotoren hovedsakelig klare å oppnå et moment i området [5, -5]% før den bevegelige plastplaten treffer hodet til personen som skal måles.
- **Tid ≈ 7:** Momentet er nå lik -10% som er det samme som det spesifiserte referansemomentet. Grunnen til at servomotoren nå leverer et høyere moment er at heisvognen presser mot hodet til personen som skal måles. På grunn av vandring i innfestningen/ bøyelighet i den svarte platen som vist i figur 5.16 og figur 5.17 vil momentet bygge seg opp før heisvognen står helt i ro.

Figur 5.16: Bildet viser vinklingen til den svarte platen rett før den presser mot hodet til personen som skal måles.

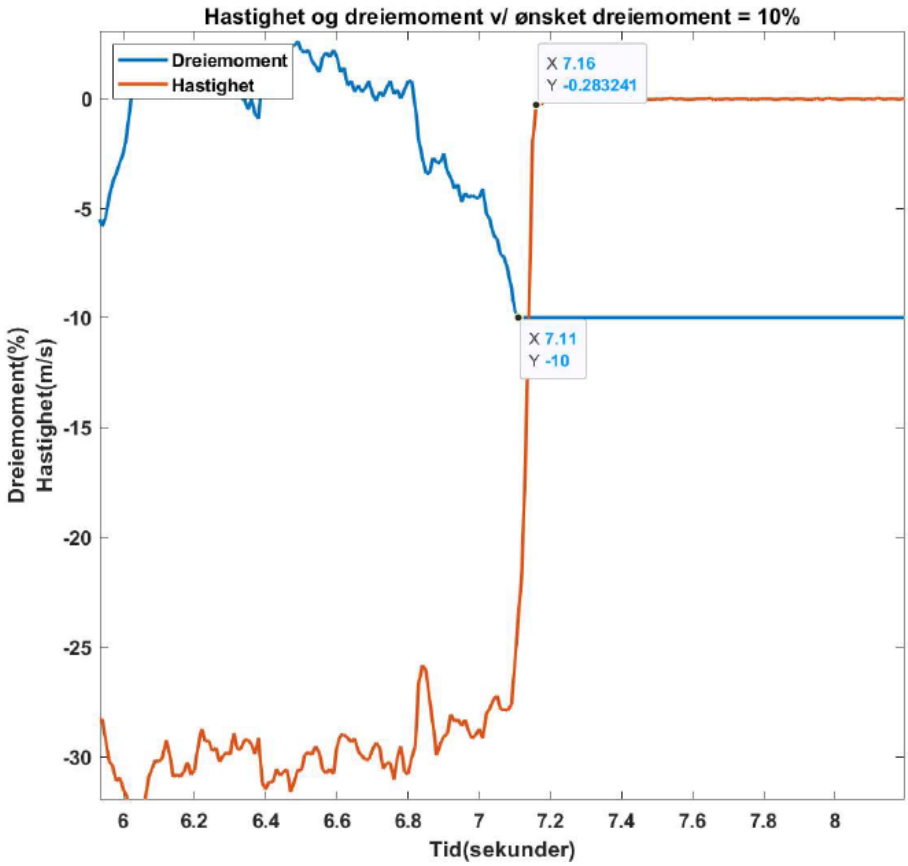


Figur 5.17: Bildet viser platens vinkling etter at servomotor 1 har oppnådd et moment på -10%.

- **Tid $\approx$ 11:** Når servomotorens moment har vært likt referansemomentet i 4 sekunder vil programmet gå fra tilstand 4 til tilstand 1. I tilstand 1 er hastighets bregrensingen satt til $100\frac{mm}{s}$ og referansemoment = 100% i positiv retning slik at heisvognen beveger seg mot toppunktet til heismodellen.

- **Tid $\approx$ 14:** Programmet er nå i tilstand 2 som er en ventetilstand. Servomotoren har nå en hastighetsbegrensing på 0 $\frac{mm}{s}$ og et referansemoment på 5%, heisvognen står stasjonært på toppunktet til heismodellen og er klar for ny måling.

Plottet i figur 5.18 viser ett forstørret utklipp fra figur 5.15. Når tid $\approx$ 6.8 sekunder treffer den bevegelige plastplaten hodet til personen som måles (som vist i figur 5.16). Dette leses fra plottet ved at kurven til momentet har en nedadgående trend og hastigheten synker. Servomotor 1 oppnår det spesifiserte referansemomentet (som vist i figur 5.17) i løpet av ca. 0,4 sekunder på grunn av vandring i innfestningen/ bøyelighet i den bevegelige plastplaten. Det ble valgt å ikke utbedre den mekaniske unøyaktigheten i heisvognen siden målet med prosjektet er ikke å få nøyaktige målinger, men å engasjere studenten.



Figur 5.18: I figuren vises ett forstørret utklipp fra figur 5.15.

5.6 Forslag til laboppgave

Målet med denne laboppgaven er at studentene skal kunne observere/føle servodriverens evne til å regulere momentet til servomotoren. Dette oppnås ved at studentene skal bruke heismodellen til måle høyden sin med forskjellige spesifiserte referansemoment. Studentene skal også skalere analogsignalet fra avstandssensoren som vist i kapittel 5.2.

I vedlegg E.3 er programmet studentene skal benytte. Vedlegget inneholder følgende:

- Ferdig programert *HMI*.
- Ferdig program som utfører målesekvensen.
- *IIR*-Filterfunksjon.
- Ett program for skalering av signalet fra avstandssensoren hvor studentene selv må sette opp skaleringen.

5.6.1 Før lab

1. Identifiser alle verdier som må tas med for å realisere ligning (5.1).
2. Skaler inngangsignalet fra AD-kortet ved bruk av `ScaleTrans`-funksjonen. Bruk informasjon gitt i kapittel 5.2 og figur 5.3 som en veiledning for hvordan signalet skaleres.

5.6.2 På lab

1. Gjør følgende:
   (a) Mål avstanden fra avstandsensoren og ned til gulvet.
   (b) Mål avstanden fra den bevegelige plastplaten og ned til gulvet.
   (c) Finn differanseavstansen mellom punktene funnet i 1a og 1b. Dette vil være *offset* verdien i `ScaleTrans`-funksjonen. (*offset*= 1b-1a).
2. Implementer nødvendige verdier i `ScaleTrans`-funksjonen.
3. Kjør programmet å observer om måleresultatet er korrekt.
4. Endre referansemomentet og observer hvordan dette påvirker servomotoren.

Kapittel 6

Demonstrasjon av 2-ordens respons

I denne videoen blir dette prosjektet demonstrert

6.1 Motivasjon for prosjekt

Elektrostudentene introduseres for andre ordens systemer i faget ELE200 Elektroteknikk 2 som kommer til å gå parallelt med faget ELE310 Styringsteknikk fra høsten 2021. I faget ELE200 Elektroteknikk 2 benyttes RLC-kretser for å demonstrere hvordan andre ordens prosesser ser ut. Siden endring/variasjon i strøm/spenning kun kan observeres gjennom oscilloscope/kurver, har vi i dette kapittelet laget en oppgave som fysisk visualiserer responsen til et andre ordens system. Måten dette blir gjort på er at brukeren spesifiserer egenskapene til den ønskede responsen via skjermen, og PLS-programmet genererer deretter den tilhørende tidsfunksjonen som sendes til servodriverne og videre til servomotorene. Vi tror at studentenes forståelse for dynamikk generelt kan bli ytterligere forsterket med en slik laboppgave, samtidig som det gir en nyttig innføring i posisjonsregulering i servodriveren.

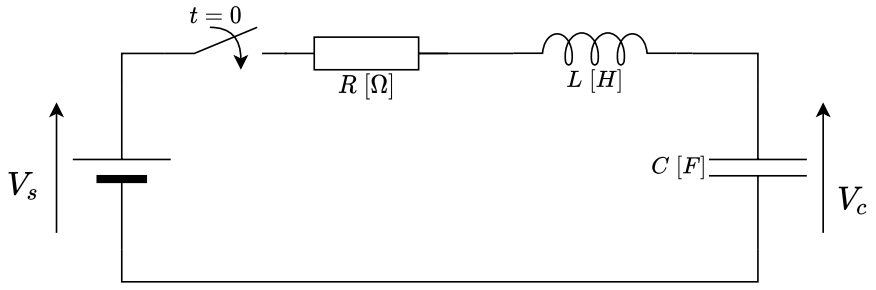
I presentasjon av prosjektet, har vi delt det opp i følgende deler:

1. Eksempel på en RLC-krets som utgangspunkt for prosjektet, pensum fra faget ELE200 Elektroteknikk 2, vist i kapittel 6.2.
2. Generering av referansefunksjon til servodriverne, vist i kapittel 6.3.
3. Presentasjon av hvordan programmet i dette kapittelet brukes, og hva det inneholder, vist i kapittel 6.4.
4. Presentasjon av tilstandsdiagram og forklaring på tilstandene sammen med tilhørende kodeutklipp, som vist i kapittel 6.5.
5. Verifisering av resultat opp mot ideell sprangrespons generert i Matlab, som vist i kapittel 6.6.
6. Forslag til laboppgave ved bruk av programmet vist i kapittelet, som vist i kapittel 6.7.

6.2 Seriekoblet *RLC*-krets, grunnlag for prosjektet

I faget ELE200 Elektroteknikk 2 benyttes RLC-kretser for å demonstrere oppførselen til andre ordens prosesser. Ved å variere størrelsene til R, L og C på lab så kan spenningen $V_c(t)$ over kondensatoren oppføre seg på mange måter. På samme vis som på lab skal det spesifiseres forskjellige verdier av ω_0 og ζ til heismodellen for å få servomotorene/ heisvognen til å bevege seg på samme vis.

I figur 6.1 vises en seriekoblet RLC-krets.



Figur 6.1: Seriekoblet RLC-krets, kretsen påtrykkes ett sprang ved $t = 0$.

Matematisk kan denne RLC-kretsen beskrives som vist i ligning (6.1).

$$V_s(t) = LC\,\frac{d^2\,V_c(t)}{dt^2} + RC\,\frac{d\,V_c(t)}{dt} + V_c(t) \tag{6.1}$$

RLC-kretsen skrives på standardform, som vist i ligning (6.2).

$$V_s(t) = \frac{1}{\omega_0^2}\,\frac{d^2\,V_c(t)}{dt^2} + 2\zeta\,\frac{1}{\omega_0}\,\frac{d\,V_c(t)}{dt} + V_c(t) \tag{6.2}$$

Ved å sammenligne ligning (6.1) og ligning (6.2) så kan sammenhengen mellom R, L, C, ω_0 og ζ beskrives som i ligning (6.3).

$$\omega_0 = \frac{1}{\sqrt{LC}} \quad \text{og} \quad \zeta = \frac{RC}{2\sqrt{LC}} \tag{6.3}$$

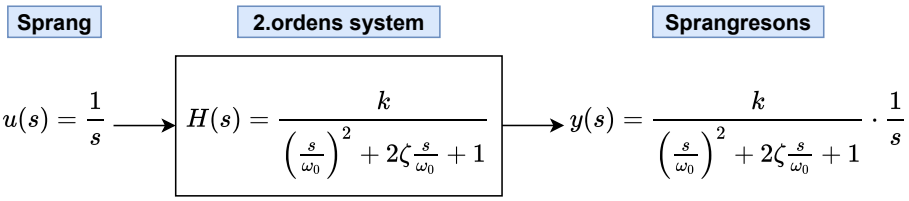
Ved å Laplacetransformere ligning (6.2) finner vi transferfunksjonen fra $V_s(s)$ og $V_c(s)$ som vist i ligning (6.4).

$$\frac{V_c(s)}{V_s(s)} = H(s) = \frac{1}{\left(\frac{s}{\omega_0}\right)^2 + \frac{s}{\omega_0}2\zeta + 1} \tag{6.4}$$

Transferfunksjonen vist i ligning (6.4) brukes både til å beskrive RLC-kretsen og heismodellen. Videre i oppgaven baseres servomotorens bevegleser på denne ligningen slik at elektrostudentene kan se en direkte kobling mellom RLC-kretsen og heismodellen.

6.3 Referansefunksjon til servodriverne

I figur 6.2 vises det ett andre ordens system $H(s)$. Systemet påtrykkes ett sprang $u(s)$, og det resulterer i en sprangrespons $y(s)$ på utgangen. Heismodellen følger sprangresponsen som en referansefunksjon. Videre i dette delkapittelet vises det hvilke ligninger som er implementert som referansefunksjoner i programmet i Sysmac Studio.



Figur 6.2: Systemet $H(s)$ påtrykkes ett sprang $u(s)$, som resulterer i sprangrespons $y(s)$.

For å finne en tidsavhengig funksjon i 2 orden utføres det invers Laplacetransformasjon på sprangrespons funksjonen $y(s)$ vist i figur 6.2 før den blir implementert i Sysmac Studio. De invers Laplacetransformerte sprangresponsene omtales som $r(t)$. Heismodellen skal følge sprangresponsen som en refereansefunksjon, derfor brukes $r(t)$ fremfor $y(t)$ videre i kapittelet for å beskrive referansefunksjonen. Det er implementert to referansefunksjoner $r(t)$ i Sysmac Studio. En for overdempet system, og en for underdempet system og udempet system.

Ligningen for $y(s)$ vises i ligning (6.5). På grunn av at overdempet system kun har reelle røtter/poler ble det valgt å bruke T_1 og T_2 fremfor ζ og ω_0 i dette tilfellet. For underdempet system ble formen med ζ og ω_0 brukt på grunn av at her vil det være komplekse røtter/poler. Ligningene i dette delkapittelet vises også i boken "Dynamiske systemer" [14].

$$y(s) = \frac{k}{\left(\frac{s}{\omega_0}\right)^2 + 2\zeta\frac{s}{\omega_0} + 1} \cdot \frac{1}{s} = \frac{k}{(T_1 s + 1)(T_2 s + 1)} \cdot \frac{1}{s} \tag{6.5}$$

Referansefunksjonen $r(t)$ som er implementert i Sysmac Studio for overdempet system vises i ligning (6.6).

$$r(t) = k \cdot \left[1 + \frac{1}{T_2 - T_1} \cdot \left(T_1 e^{-t/T_1} - T_2 e^{-t/T_2}\right)\right] \tag{6.6}$$

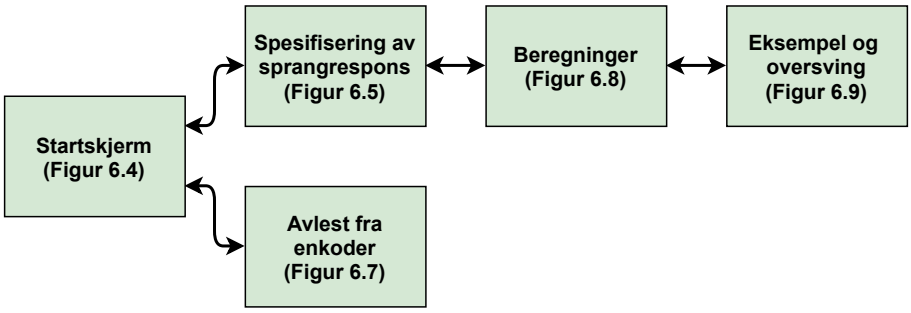
Referansefunksjonen $r(t)$ som er implementert i Sysmac Studio for underdempet system og udempet system vises i ligning (6.7).

$$r(t) = k \cdot \left[1 - \frac{1}{\sqrt{1-\zeta^2}} \cdot e^{-\zeta\omega_0 t} \cdot cos\left(\sqrt{1-\zeta^2} \cdot \omega_0 t - sin^{-1}(\zeta)\right)\right] \tag{6.7}$$

6.4 Programfunksjonalitet

Hovedfunksjonaliteten til programmet som presenteres i dette kapittelet er at servomotor 1 og servomotor 2 blir satt til å følge enten en overdempet eller en underdempet referansefunksjon. For at servomotorene skal følge gitt referansefunksjon brukes en `MC_SyncMoveAbsolute`-funksjonsblokk som vist i vedlegg D.2.

I figur 6.3 vises det hvordan skjermene i *HMI* er koblet sammen. *HMI*-skjermen gjør det enklere og mer oversiktlig for bruker å betjene programmet.

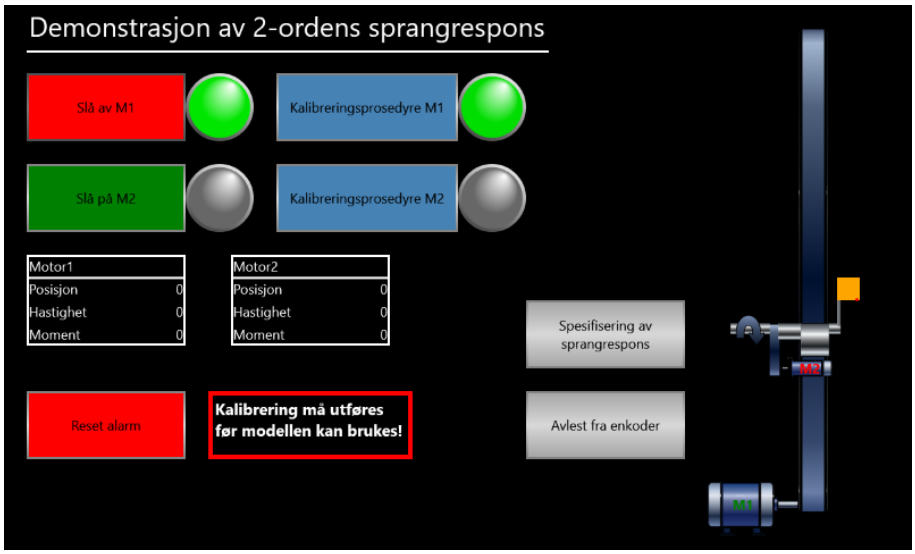


Figur 6.3: Oversiktsbilde viser hvordan skjermene i *HMI* er koblet sammen.

I listen under forklares det hvordan programmet brukes:

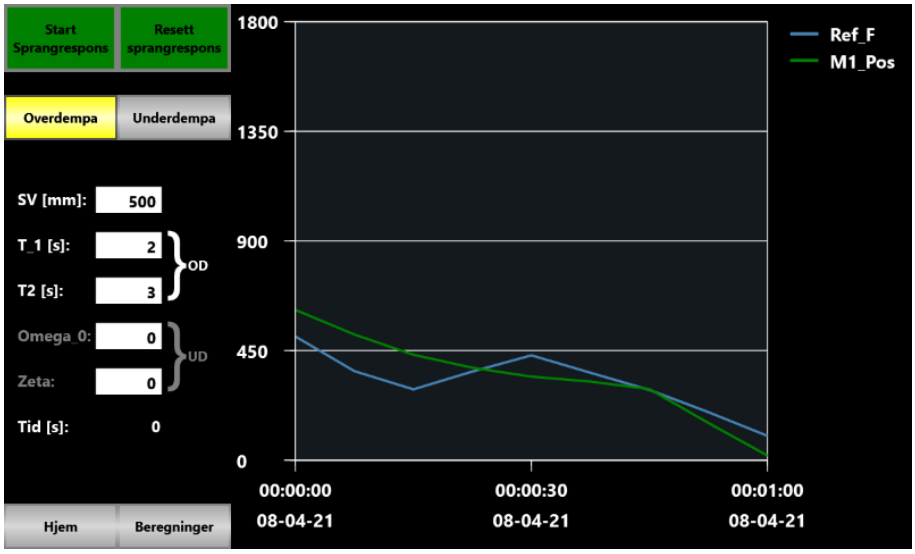
1. I figur 6.4 vises hjemskjermen til programmet. Den røde knappen merket **Slå av M1** slår av/ på servodriver 1, og den grønne lampen til høyre for knappen viser om servodriveren er av/ på (her er servodriver 1 påslått). Den grønne knappen merket **Slå på M2** slår på/ av servodriver 2, og den grå lampen til høyre for knappen viser om servodriveren er på/ av (her er servodriver 2 avslått). Av/ på-knappene endrer farge og tekst ettersom servodriverne og servomotorene er avslått/ påslått.

2. De blå knappene i figur 6.4 merket **Kalibreringsprosedyre M1** og **Kalibreringsprosedyre M2** utfører kalibreringsprosedyre som beskrevet i vedlegg D.5 for den aktuelle servomotoren. Ved fullført kalibreringsprosedyre tennes lampene til høyre for kalibrerings-knappene.



Figur 6.4: I figuren vises hjemskjermen til programmet. Knappene **Slå av M1** og **Slå på M2** betjenes for å slå på/ av servodriverne og servomotorene. De blå knappene **Kalibreringsprosedyre M1** og **Kalibreringsprosedyre M2** utfører kalibreringsprosedyre for servomotorene, og lampene viser kalibreringsstatus til servomotorene. De hvite tabellene viser data hentet fra servomotorenes enkodere. Den rød knappen **Reset alarm** resetter eventuelle alarmer. De grå knappene **Spesifisering av sprangrespons** og **Avlest fra enkoder** fører til andre skjermer. Figuren til høyre viser heismodellens bevegelser, animasjonen er knyttet opp mot avlest enkoder posisjon for servomotorene.

3. Ved å betjene de grå knappene **Spesifisering av sprangrespons** og **Avlest fra enkoder** i figur 6.4 åpnes nye sider i *HMI*. Ved å betjene den grå knappen **Spesifisering av sprangrespons**, åpnes skjermen vist figur 6.5, og ved å betjene den grå knappen **Avlest fra enkoder**, åpnes skjermen vist figur 6.7.

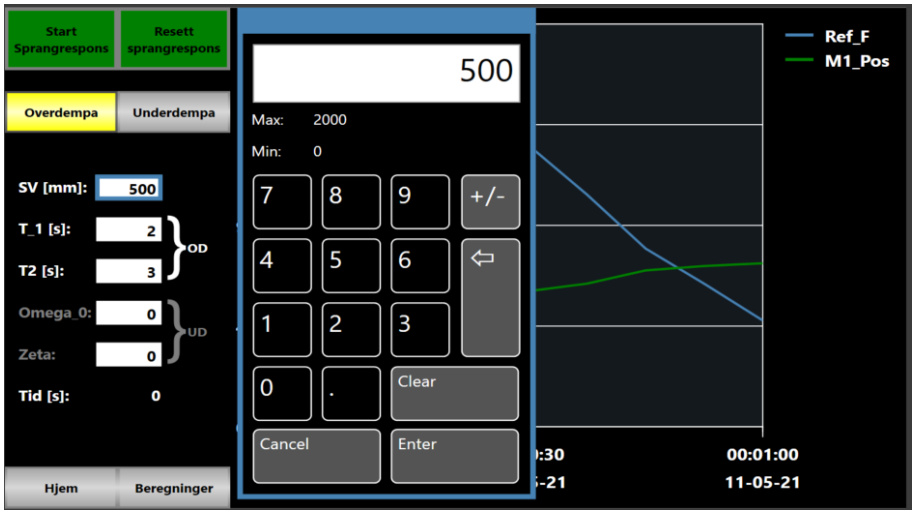


Figur 6.5: Skjerm for spesifisering av ønsket respons. Respons startes og stoppes med de grønne knappene **Start sprangrespons** og **Resett sprangrespons**. Ønsket respons velges ved å spesifisere **overdempet** eller **underdempet** system, ut fra hva som er valgt indikeres variablene som må spesifiseres i hvit skrift. De grå knappene **Hjem** og **Beregninger**, nederst på siden fører til andre skjermer. Grafen viser beregnet referansefunksjon sammen med posisjon avlest fra servomotorens enkoder. I grafen viser y-aksen høyde i mm og x-aksen tiden i sekunder. Grafen i figuren viser tilfeldige data, da det er ett utklipp fra Sysmac Studio.

4. På skjermen vist i figur 6.5 spesifiserer bruker variablene vist i tabell 6.1 for å få ønskede egenskaper for responsen. Ved å spesifisere overdempet/ underdempet system, vil tilhørende variabler som må spesifiseres for responsen være i uthevet hvit skrift. Ved å trykke på de hvite boksene som inneholder variabler så åpnes ett talltastatur som vist i figur 6.6 hvor variabel verdi kan spesifiseres.

Tabell 6.1: Tabellen viser variablene som kan spesifiseres i figur 6.5.

Navn	**Symbol**	**Typisk verdi**
Overdempet/ underdempet system	Knapp	Overdempa/underdempa
Set verdi	SV	$[500, 1000]$ mm
Tidskonstant 1	T_1	$[2, 10]$ s
Tidskonstant 2	T_2	$[2, 10]$ s
Naturlig frekvens	ω_0	$[\frac{\pi}{4}, \frac{\pi}{8}]$ rad/s
Dempningsfaktor	ζ	$[0, 1]$

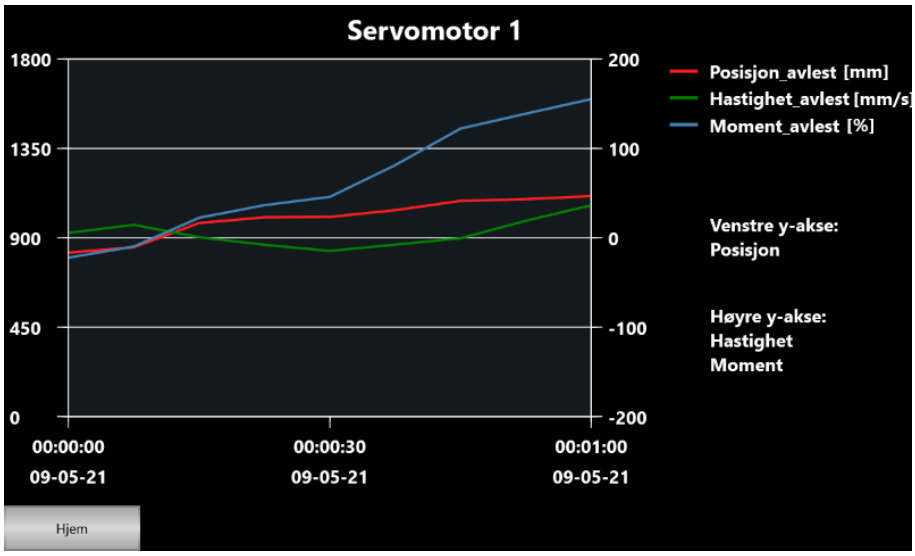


Figur 6.6: I figuren vises samme *HMI*-skjerm som vist i figur 6.6. I denne figuren vises også talltastatur som brukes for å spesifisere variabler som brukes for å lage responsen til heismodellen. Talltastaturet åpnes ved å klikke på hvit boks til variabel som skal spesifiseres.

5. Når ønsket respons er spesifisert betjenes knappen **Start sprangrespons** som vist i figur 6.5, heismodellen vil følge angitt respons som en referanse.

6. For å stoppe heismodellen under kjøring og returnere den tilbake til den kalibrerte startposisjonen betjenes knappen **Resett sprangrespons** som vist i figur 6.5.

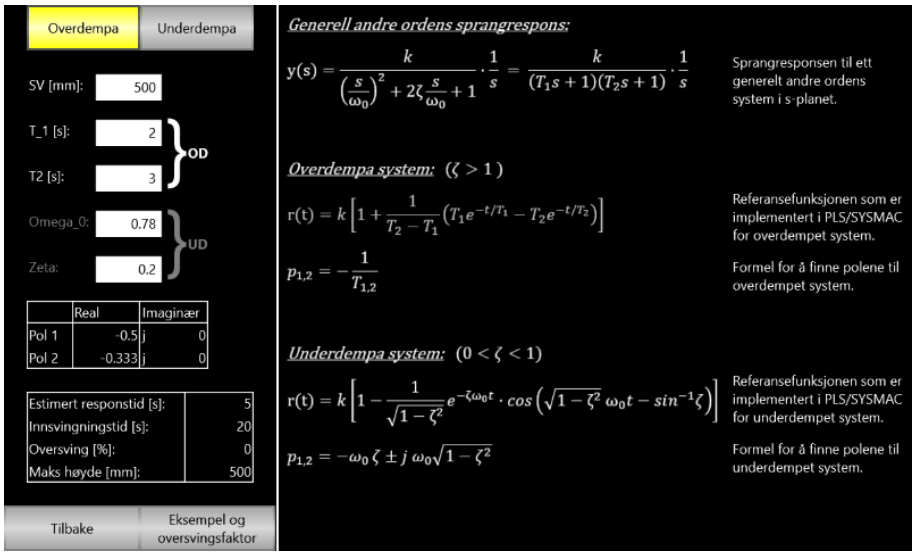
Utover hovedfunksjonaliteten til programmet som er vist i punklisten over, så er det flere skjermer som viser mer informasjon om systemet. Ved å betjene knappen **Avlest fra enkoder** på hjemskjermen, som vist i figur 6.4, åpnes skjermen vist i figur 6.7. Her vises informasjon som leses fra enkoderen som hører til servomotor 1. Informasjonen som hentes ut er; posisjon [mm], hastighet [mm/s] og moment [%].



Figur 6.7: Graf som viser posisjon, hastighet og moment til servomotor 1. Venstre y-akse er knyttet opp mot posisjon [mm]. Høyre y-akse er knyttet opp mot hastighet [$\frac{mm}{s}$] og moment [%]. X-aksen er knyttet opp mot tid, grafen viser data i ett minutters periode. Grafen er hentet fra Sysmac Studio, og viser tilfeldige data.

Ved å betjene knappen **Beregninger** på skjermen vist i figur 6.5 åpnes skjermen vist i figur 6.8. Skjermen er lagt til i *HMI* for å gi brukeren en oversikt over hvilke formler som brukes for å generere referansefunksjonen og for å vise frem data om den spesifiserte responsen. På venstre side kan bruker spesifisere ønsket respons, på samme vis som skjermen vist i figur 6.5. Muligheten for å spesifisere respons er også lagt til på denne skjermen fordi beregningene som vises i tabellene lenger nede på skjermen oppdateres når de valgte variablene endres. I tabellene videre nedover på den venstre siden vises polene til systemet, estimert responstid, innsvingningstid, oversving og maksimum høyde. På høyre side av figur 6.8 vises først sprangresponsen $y(s)$, som heismodellen er basert på. Videre i figuren vi-

ses referansefunksjonene $r(t)$ som heismodellen bruker for overdempet og underdempet sprangrespons, formlene for beregning av poler er også vist.



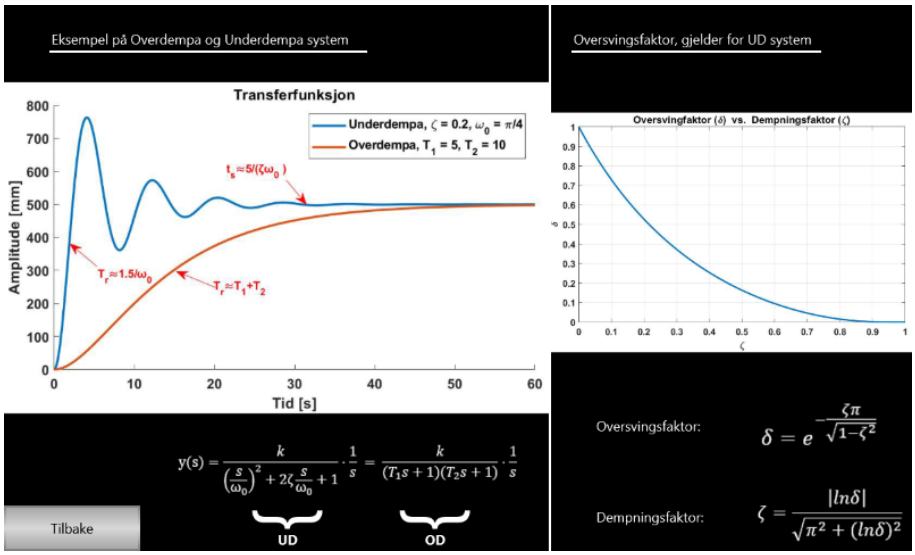
Figur 6.8: På venstre side kan respons spesifiseres, og diverse beregnede variabler kan leses. Til høyre vises generell funksjon for andre ordens system, og referansefunksjonen som er implementert for overdempet og underdempet system.

I tabell 6.2 vises ligningene som er lagt inn i programmet for å beregne variablene som er presentert nede på venstre side i figur 6.8.

Tabell 6.2: Ligninger som er lagt inn for å beregne verdier som er vist på venstre side i figur 6.8

	Overdempet system	**Underdempet system**
Estimert responstid	$T_r = T_1 + T_2$	$T_r = \frac{1{,}5}{\omega_0}$
Innsvingningstid	$T_s = 4 \cdot T_r$	$T_s = \frac{4}{\zeta\,\omega_0}$
Oversving	$\delta = 0$	$\delta = e^{-\frac{\zeta\pi}{\sqrt{1-\zeta^2}}}$
Maksimum høyde	$y_{maks} = SV$	$y_{maks} = SV \cdot (\delta + 1)$

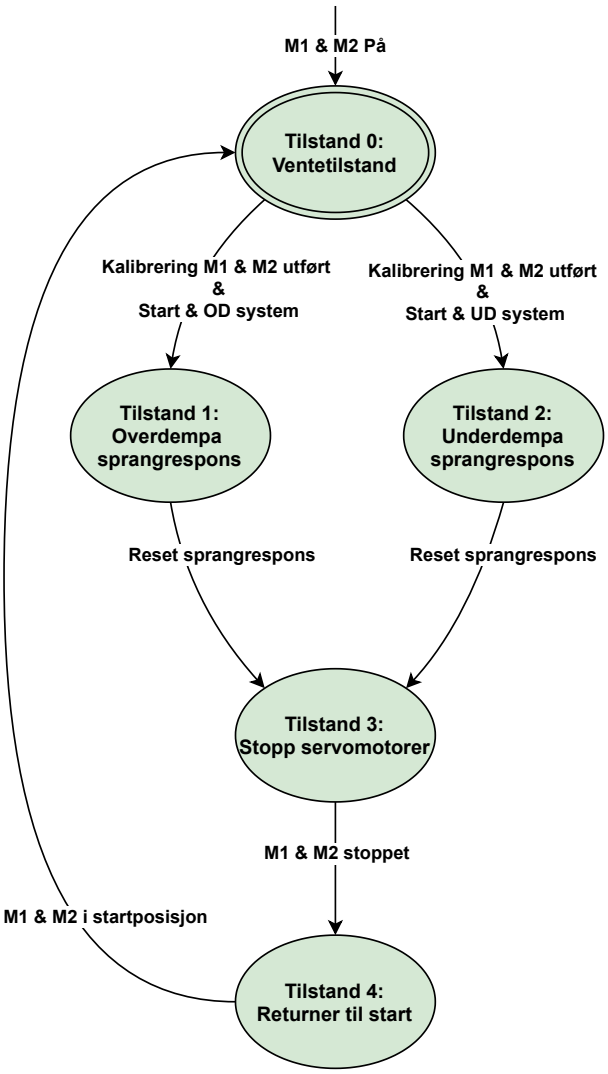
Ved å betjene knappen **Eksempel og oversvingsfaktor** på skjermen vist i figur 6.8 åpnes skjermen vist under i figur 6.9. Skjermen er lagt til i *HMI* for å gi brukeren innblikk i to typiske responser for 2 ordens system, med tilhørende verdier som fungerer bra for heismodellen, og for å vise brukeren oversvinget for forskjellige verdier av ζ. På venstre side i figur 6.9 vises sprangresponsen for ett typisk underdempet og overdempet system. Grafen på høyre side brukes til å estimere oversving til ett underdempet system. Oversvinget estimeres ut fra spesifisert ζ. Sammenhengen mellom dempningsfaktor ζ og oversvingsfaktor δ er gitt med ligninger under tilhørende graf på høyre side i figur 6.9.



Figur 6.9: Til venstre i figuren vises graf med ett typisk over- og under-dempa system. Under grafen på venstre side er ligningene som gir tilhørende sprangrespons. Til høyre i figuren vises sammenhengen mellom dempningsfaktor ζ og oversvingsfaktor δ. Under grafen på høyre side vises sammenheng mellom dempningsfaktor ζ og oversvingsfaktor δ ved hjelp av ligninger.

6.5 Tilstandsdiagram og programstruktur

For å vise hvordan programmet er bygd opp har vi i figur 6.10 tegnet et tilstandsdiagram som viser hovedstrukturen til programmet i dette kapittelet. Videre i delkapittelet gis det forklaring på hva som utføres i hver tilstand og hva som er krav for transisjon mellom tilstandene.



Figur 6.10: Tilstandsdiagram for programmet. Beskrivelse av tilstandene og transisjonene blir gitt videre i gjeldende delkapittelet.

Videre forklares det hva som utføres i tilstandene, betingelse for transisjon mellom tilstandene og tilhørende kodeutklipp til tilstandene.

Tilstand 0: Dette er starttilstanden til programmet. Når servomotorene slås på går programmet til tilstand 0. På kodelinje 15-20 i kodeutklipp 6.1 settes holde-variablene `kalibrering_1_utfort` og `kalibrering_2_utfort` høye dersom tilhørende `MC_Home`-funksjonsblokk gir ut høyt flankesignal om at kalibreringsprosedyre er utført som vist i vedlegg D.5. Dersom servomotorene slås av settes holde-variablene `kalibrering_1_utfort` og `kalibrering_2_utfort` lave, og kalibreringsprosedyre må utføres på ny før bruk av programmet. Kravet for transisjon til neste tilstand er oppfylt dersom: 1) kalibreringsprosedyre for servomotorene er utført, 2) egenskapene til ønsket respons er spesifisert og 3) Bruker betjener knappen **Start sprangrespons** som vist i figur 6.5.

Kode 6.1: Kodeutklipp for tilstand 0

```
8  // Servomotor 1 og 2 maa vaere aktivert
9  IF servo1_power_paa AND servo2_power_paa THEN
10
11 // Tilstandsprogram
12     CASE case_program OF
13
14         0: // Ventetilstand, venter paa start signal
15         IF servo1_home_done THEN
16             kalibrering_1_utfort:=TRUE;
17         END_IF;
18         IF servo2_home_done THEN
19             kalibrering_2_utfort:=TRUE;
20         END_IF;
21         IF kalibrering_1_utfort AND ...
               kalibrering_2_utfort THEN
22             IF start_program AND overdempa_system THEN
23                 case_program:=1;
24             ELSIF start_program AND Underdempa_system ...
                   THEN
25                 case_program:=2;
26             END_IF;
27         END_IF;
```

Tilstand 1: Ved spesifisert overdempet system går transisjonen fra tilstand 0 til tilstand 1. Heismodellen følger referansefunksjonen vist i ligning (6.6), referansefunksjonen beregnes ut i fra spesifisert T_1, T_2, setverdi, og tiden som vist på kodelinje 33 i kodeutklipp 6.2. `MC_SyncMoveAbsolute`-funksjonsblokken som vist i vedlegg D.2 brukes i programmet for å gi signal til servodriveren om at servomotoren skal følge referansefunksjonen som posisjonsreferanse. Funksjonsblokken tilhørende servomotor 1 og servomotor 2 blir aktivert på henholdsvis kodelinje 30 og 31 som vist i kodeutklipp 6.2. Hvis bruker betjener knappen **Resett sprangrespons** som vist i figur 6.5 oppfylles kravet for transisjon til tilstand 3.

Kode 6.2: Kodeutklipp for tilstand 1.

```
29      1: // Overdempet system
30      mc_sprangrespons_1_execute:=TRUE;
31      mc_sprangrespons_2_execute:=TRUE;
32      tid:=tid+syklus_tid;
33      referansefunksjon:= k*(1+(1/(T2-T1))* ...
            (T1*exp(-tid/T1)-T2*exp(-tid/T2)));
34      IF reset_posisjon THEN
35         referansefunksjon:=0;
36         mc_sprangrespons_1_execute:=FALSE;
37         mc_sprangrespons_2_execute:=FALSE;
38         case_program:=3;
39      END_IF;
```

Tilstand 2: Ved spesifisert underdempet system går transisjonen fra tilstand 0 til tilstand 2. Heismodellen følger referansefunksjonen vist i ligning (6.7). Referansefunksjonen beregnes ut i fra spesifisert ω_0, ζ, setverdi og tiden som vist på kodelinje 45 i kodeutklipp 6.3. `MC_SyncMoveAbsolute`-funksjonsblokken som vist i vedlegg D.2 brukes i programmet for å gi signal til servodriveren om at servomotoren skal følge referansefunksjonen som posisjonsreferanse. Funksjonsblokken tilhørende servomotor 1 og servomotor 2 blir aktivert på henholdsvis linje 42 og 43 som vist i kodeutklipp 6.3. Hvis knappen **Resett sprangrespons** blir betjent (som vist i figur 6.5) oppfylles kravet for transisjon til tilstand 3.

Kode 6.3: Kodeutklipp for tilstand 2.

```
41        2: // Underdempet system
42        mc_sprangrespons_1_execute:=TRUE;
43        mc_sprangrespons_2_execute:=TRUE;
44        tid:=tid+syklus_tid;
45        referansefunksjon:=k*(1-((exp(-zeta*omega_0*tid)* ...
              sin(sqrt(1-EXPT(zeta,2))*omega_0* ...
              tid+acos(zeta)))/(sqrt(1-EXPT(zeta,2)))));
46      IF reset_posisjon THEN
47          referansefunksjon:=0;
48          mc_sprangrespons_1_execute:=FALSE;
49          mc_sprangrespons_2_execute:=FALSE;
50          case_program:=3;
51      END_IF;
```

Tilstand 3: I denne tilstanden gis det signal til servodriverne om å slutte å følge referansefunksjonen. `MC_Stop`-funksjonsblokken som vist i vedlegg D.2 brukes for å fortelle servodriveren at servomotoren skal slutte å følge `MC_MoveSyncAbsolute`-funksjonsblokken som posisjonsreferanse. `MC_Stop`-funksjonsblokken tilhørende servomotor 1 og servomotor 2 blir aktivert på henholdsvis linje 54 og 55 som vist i kodeutklipp 6.4. Transisjon til tilstand 4 er oppfylt når `MC_Stop`-funksjonsblokkene knyttet til servomotorene gir høyt signal om at instruksjon er fullført.

Kode 6.4: Kodeutklipp for tilstand 3.

```
53      3: // Stopper servomotorene
54      servo1_stopp_execute:=TRUE;
55      servo2_stopp_execute:=TRUE;
56      IF servo1_stopp_done AND servo2_stopp_done THEN
57          servo1_stopp_execute:=FALSE;
58          servo2_stopp_execute:=FALSE;
59          case_program:=4;
60      END_IF;
```

Tilstand 4: I denne tilstanden gis signal til servodriverne om å returnere servomotorene til den kalibrerte startposisjonen, som vist i vedlegg D.5. `MC_MoveZeroPosition`-funksjonsblokken som vist i vedlegg D.2 brukes for å fortelle servodriveren at servomotoren skal returnere til den kalibrerte startposisjonen. `MC_MoveZeroPosition`-funksjonsblokken tilhørende servomotor 1 og servomotor 2 blir aktivert på henholdsvis linje 63 og 64 som vist i kodeutklipp 6.5. Transisjon til tilstand 1 er oppfylt når `MC_MoveZeroPosition`-funksjonsblokkene knyttet til servomotorene gir høyt signal om at instruksjon er fullført.

Kode 6.5: Kodeutklipp for tilstand 4.

```
62      4: // Returnerer servomotorene til start
63      servo1_posisjon_0_execute:=TRUE;
64      servo2_posisjon_0_execute:=TRUE;
65      IF servo1_posisjon_0_done AND ...
            servo2_posisjon_0_done THEN
66          servo1_posisjon_0_execute:=FALSE;
67          servo2_posisjon_0_execute:=FALSE;
68          tid:=0;
69          case_program:=0;
70      END_IF;
```

6.6 Verifisering av resultat i Matlab

For å verifisere at servomotorene følger referansefunksjonen $r(t)$, så plottes den avleste enkoderposisjonen til servomotor 1 sammen med teoretisk fremstilling av sprangresponsen i Matlab. Enkoderposisjonen til servomotor 1 logges ved hjelp av *data trace*, som vist i vedlegg D.6. Etter kjøring av heismodellen blir *data trace* eksportert som en `.csv` fil fra Sysmac Studio, og `.csv` filen blir lest inn i Matlab som vist i vedlegg D.6. I kodeutklipp 6.6 vises det ett eksempel på hvordan enkoderposisjonen til servomotor 1 og teoretisk sprangrespons generert i Matlab har blitt plottet sammen.

Kode 6.6: Kodeutklippet viser eksempel på hvordan logget *data trace* fra Sysmac Studio leses inn i Matlab. På kodelinje 2 leses *data trace* filen inn. På kodelinje 5 genereres teoretisk sprangrespons. På kodelinje 8-13 plottes servomotorens posisjon opp mot den genererte teoretiske sprangresponsen.

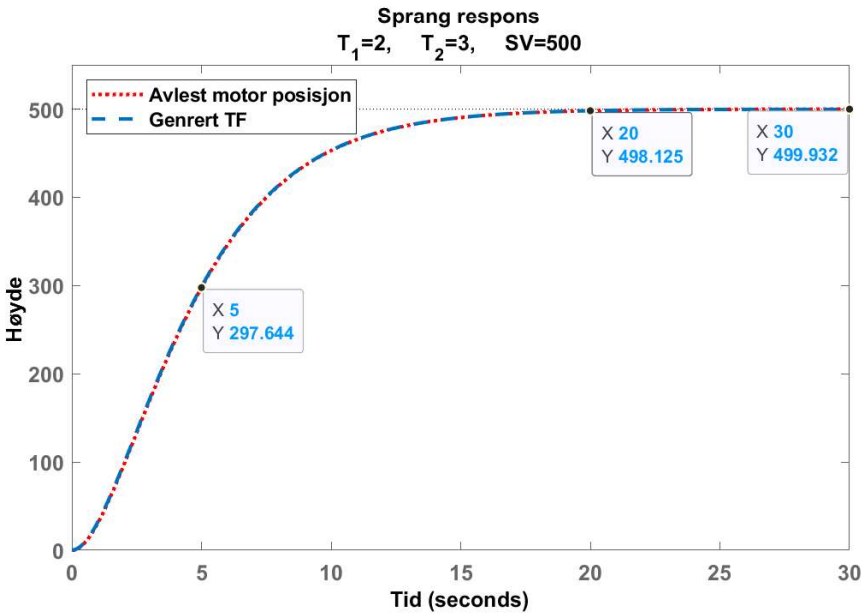
```
1  % Leser angitt fil
2  fil = funksjon_les_sprangrespons('6_udempa_202102.csv');
3
4  % Bruker matlabs innebygde transferfunksjon generator, med ...
       parametre lest fra program
5  system = tf(fil.kLREAL(1), [1/(fil.omega_0LREAL(1)^2) ...
       (2*fil.zetaLREAL(1))/fil.omega_0LREAL(1), 1]);
6
7  % Plotter logga motor posisjon og beregninger gjort i PLS
8  hold on
9  plot(fil.tidLREAL, fil.MC_Axis000ActPosLREAL)
10 stepplot(system)
11 legend('Avlest motor posisjon', 'Genrert TF ')
12 title('Sprangrespons')
13 hold off
```

6.6.1 Overdempet system

I figur 6.11 har avlest enkoder posisjon til servomotor 1 blitt plottet sammen med sprangrespons generert i Matlab. Det kommer tydelig frem ved å se på figuren at systemet er overdempet, fordi systemet ikke har oversving. Servomotoren har fulgt ligning (6.6) som referansefunksjon med $T_1 = 2$ og $T_2 = 3$. Sprangresponsen som er generert i Matlab er også beregnet med $T_1 = 2$ og $T_2 = 3$.

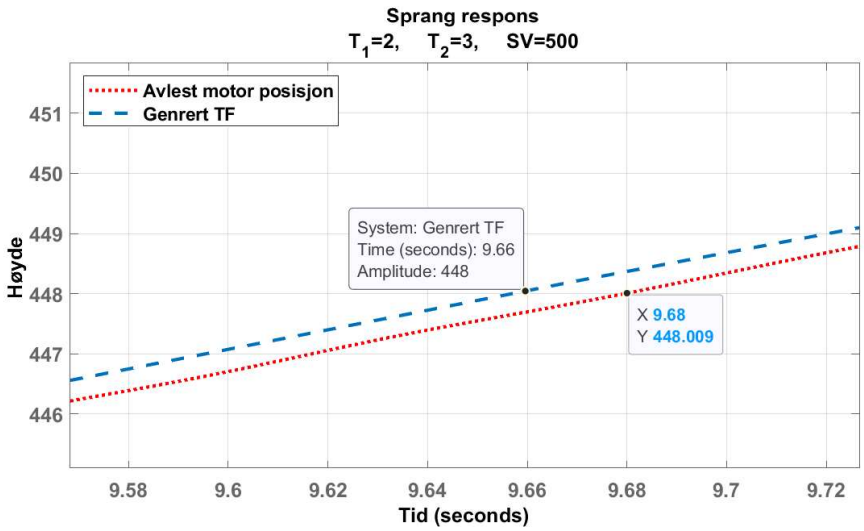
I figur 6.11 har følgende data blitt markert:

- Estimert responstid: $T_r = T_1 + T_2 = 2 + 3 = 5$ *sekund*
- Estimert innsvingningstid: $T_s = 4 \cdot T_r = 4 \cdot 5 = 20$ *sekund*
- Maksverdi: ≈ 500



Figur 6.11: Overdempet system, systemet svinger seg inn til ønsket verdi uten oversving. Den avleste enkoder posisjonen til servomotor 1 er plottet sammen med teoretisk fremstilling av tilsvarende andre ordens sprangrespons.

I figur 6.12 vises ett forstørret utklipp av figur 6.11. Her vises det at servomotoren følger den teoretiske fremstillingen svært bra. Markeringene i figuren viser at ved høyde ≈ 448 mm, så er tidsdifferansen mellom teoretisk fremstilling og avlest posisjon $\Delta\text{Tid} \approx 9,68 - 9,66 = 0,02\ s = 20\ ms$. Eksekveringshastigheten til PLS er satt til 2 ms. Det betyr at tiden som systemet bruker fra PLS beregner referansefunksjon og til enkoder leser at servomotoren er i angitt posisjon så utføres det 10 eksekveringer i PLS.



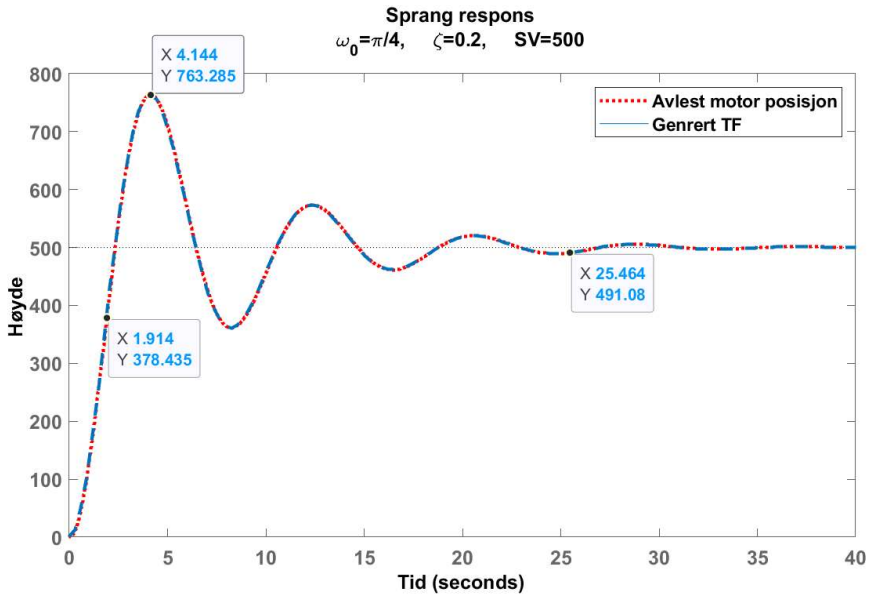
Figur 6.12: Forstørret utklipp av figur 6.11. For å vise tidsdifferansen mellom teoretisk fremstilling og servomotorens posisjon. Det observeres en tidsdifferanse på omtrent 20 ms, som tilsvarer 10 eksekveringer i PLS.

6.6.2 Underdempet system

I figur 6.13 har avlest enkoder posisjon til servomotor 1 blitt plottet sammen med sprangrespons generert i Matlab. Det kommer tydelig frem ved å se på figuren at systemet er underdempet, fordi systemet har oversving. Servomotoren har fulgt ligning (6.7) som referansefunksjon med $\omega_0 = \pi/4$ og $\zeta = 0,2$. Sprangresponsen som er generert i Matlab er også beregnet med $\omega_0 = \pi/4$ og $\zeta = 0,2$.

I figur 6.13 har følgende data blitt markert:

- Estimert responstid: $T_r = \dfrac{1,5}{\omega_0} = \dfrac{1,5}{\pi/4} \approx 1,91\ sekund$
- Estimert innsvingningstid: $T_s = \dfrac{4}{\zeta\omega_0} = \dfrac{4}{0,2 \cdot \pi/4} \approx 25,46\ sekund$
- Estimert oversving: $\delta = e^{-\frac{\zeta\pi}{\sqrt{1-\zeta^2}}} = e^{-\frac{0,2\pi}{\sqrt{1-0,2^2}}} \approx 53\ \%$



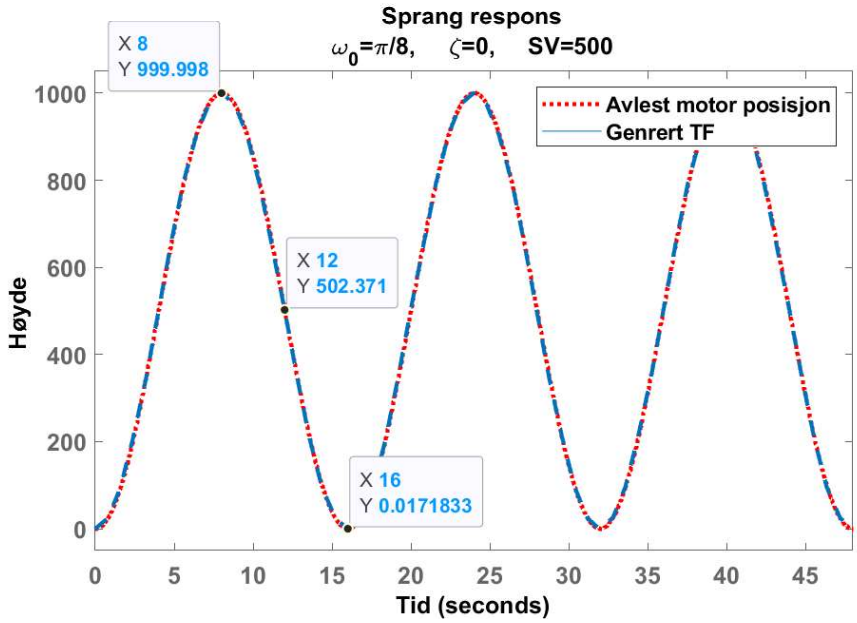
Figur 6.13: Underdempet system, systemet svinger seg inn til ønsket verdi med oversving. Den avleste enkoder posisjonen til servomotor 1 er plottet sammen med teoretisk fremstilling av tilsvarende andre ordens sprangrespons.

6.6.3 Udempet system

I figur 6.14 har avlest enkoder posisjon til servomotor 1 blitt plottet sammen med sprangrespons generert i Matlab. Det kommer tydelig frem ved å se på figuren at systemet er udempet, fordi det observeres kontinuerlig oscillasjon. Servomotoren har fulgt ligning (6.7) som referansefunksjon med $\omega_0 = \pi/8$ og $\zeta = 0$. Sprangresponsen som er generert i Matlab er også beregnet med $\omega_0 = \pi/8$ og $\zeta = 0$.

I figur 6.14 har følgende data blitt markert:

- Maksverdi: ≈ 1000, ved $Tid = 8$ *sekund*
- Ønsket verdi: ≈ 500, ved $Tid = 12$ *sekund*
- Minimumverdi: ≈ 0, ved $Tid = 16$ *sekund*



Figur 6.14: Udempet system, systemet vil aldri svinge seg inn til ønsket verdi. Den avleste enkoder posisjonen til servomotor 1 er plottet sammen med teoretisk fremstilling av tilsvarende andre ordens sprangrespons.

6.7 Forslag til laboppgave

For at studenten skal få en fullstendig og god forståelse av sprangresponsen til andre ordens systemer så vil det være fordel med ett labopplegg som består av oppgaver før, på og etter lab. Ved laboppgave knyttet til sprangrespons referanse anbefales det at oppgaven blir gjennomført som en demonstrasjonslab. Laboppgaven tar utgangspunkt i at studenten skal gjøre det samme som ble vist i kapittel 6.6.

I vedlegg E.4 er det ett program som er tilpasset til laboppgaven, kode som skal implementeres av studenten er klippet vekk, og det har blitt lagt til hjelpekommentarer.

6.7.1 Før lab

1. Skisser sprangresponsen til følgende andre ordens system:
   (a) Overempet system basert på ligning (6.5), der $T_1 = 2$ og $T_2 = 3$.
   (b) Underdempet system basert på ligning (6.5), der $\zeta = 0.2$ og $\omega_0 = \frac{\pi}{4}$.
   (c) Udempet system basert på ligning (6.5), der $\zeta = 0$ og $\omega_0 = \frac{\pi}{8}$.
2. Implementer ligning (6.6) og ligning (6.7) i tilstand 1 og tilstand 2 i programmet gitt i vedlegg E.4.
3. Studer koden i programmet, og tegn tilhørende tilstandsdiagram.

6.7.2 På lab

1. Koble opp datamaskin til PLS og heismodell, og last opp programmet.
2. Opprett *datatrace* med følgende innhold: Posisjon for servomotor 1, referansefunksjon og tidsvariabelen som brukes (som vist i vedlegg D.6).
3. Kjør heismodellen med følgende responser spesifisert i *HMI*-skjerm, og logg kjøringen med *datatrace*:
   (a) Overempet system med $T_1 = 2$ og $T_2 = 3$.
   (b) Underdempet system med $\zeta = 0.2$ og $\omega_0 = \frac{\pi}{4}$.
   (c) Udempet system med $\zeta = 0$ og $\omega_0 = \frac{\pi}{8}$.

6.7.3 Etter lab

1. Opprett importeringscript i Matlab for importering av *datatrace* (som vist i vedlegg D.7).
2. Importer *datatrace* som ble logget på lab til Matlab, og sammenlign med Matlabs innebygde funksjon for generering av sprangresponser (som vist i kapittel 6.6).

Kapittel 7

Konklusjon

Resultatet av denne bacheloroppgaven er en heismodell med styreskap og fire prosjekter med forslag til laboppgaver i faget ELE310 Styringsteknikk.

Bachelorpppgaven har i hovedsak bestått av å lage fire prosjekter som kan brukes til labopplegget i faget ELE310 Styringsteknikk, bygge ett styreskap med alle nødvendige komponenter til heismodellen og utbedre heisvognen som er montert på heismodellens skinneprofil. Det har blitt lagt mye arbeid ned i å planlegge samt å utføre byggingen av styreskapet og utbedring av heismodellen.

De fire prosjektene som har blitt utviklet i arbeidet med oppgaven viser forskjellige måter servomotorer kan styres på ved hjelp av PLS i Sysmac Studio. Det første prosjektet demonstrerer hvordan det kan settes begrensninger for å oppnå jevnere bevegelser for servomotoren. Det andre prosjektet viser hvordan servomotorer kan brukes til å programmere en personheis som prioriterer etasje-forespørsler fra passasjerer. Det tredje prosjektet demonstrerer hvordan servomotorer kan styres med momentregulering for å måle høyden til personer. Det fjerde prosjektet viser hvordan servomotorer kan bli satt til å følge ett referansesignal.

7.1 Forslag til forbedringer

I kapittel 4 hvor det har blitt laget ett program for en personheis vil det være mulig utvide programmet til flere etasjer. Utfordringen her blir at ved flere etasjer så blir også logikken som trengs for prioritering av etasjeforespørsler mer avansert. I tillegg er det mulig å legge til begrensninger for heisens akselerasjon og *jerk* som beskrevet i kapittel 3 slik at heisen oppnår jevnere bevegelser.

I kapittel 5 hvor det har blitt laget ett program der servomotorene styres med momentregulering vil det være mulig å estimere/ beregne kraften den utstikkende plastplaten presser med på hodet til personen som står under. I programmet leses servomotorens moment i [%] fra servodriveren, som er basert på effekten servomotoren bruker (der 100 [%] tilsvarer 400 W for servomotor 1). Utfordringen her vil bli at det må tas hensyn til energitapet som er mellom effekten tilført servomotoren og kraften den utstikkende platplaten utøver på personens hode. Vi tror denne forbedringen vil bidra til å forbedre studenters forståelse av omgjøring av elektrisk effekt til mekaniske krefter.

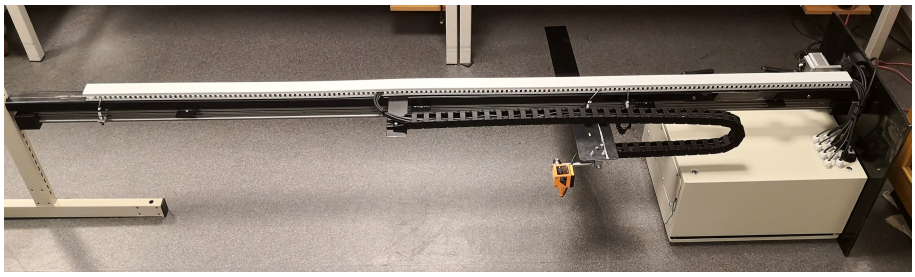
I kapittel 6 hvor servomotorene følger en andre ordens respons som referansesignal kan programmet utbedres ved å legge til en referansefunksjon for kritisk dempet system (referansefunksjonene som er implementert i programmet tillater kun overdempet-, underdempet og udempet- system).

7.2 Videreutvikling

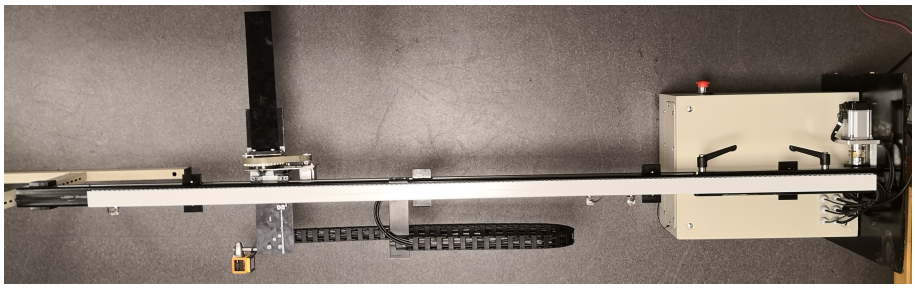
Heismodellen er bygget med tanke på at ting kan videreutvikles/ endres. Kabelføringene kan åpnes og lukkes for å legge til nye kabler slik at det enkelt kan ettermonteres/ flyttes på komponenter. I tillegg er kabellengden til komponentene montert på heismodellen beholdt slik at det er mulig å gjøre endringer i fremtiden.

Servomotorene er allsidige og kan programmeres til å utføre flere forskjellige oppgaver, så mulighetene for å lage ulike typer programmer er store. Videre gis det forslag til andre anvendelser av heismodellen.

- Lage simulering av heismodellen i Matlab ved hjelp av Simulink.
- Det kan lages ett program som regulerer en vinsj ved hjelp av aktiv hiv-kompensasjon. Aktiv hiv-kompensasjon brukes for å kompensere for bevegelsene til fartøy på havet når det utføres løfteoperasjoner (bølgene simuleres i vertikal retning av servomotor 1, og båtens roterende bevegelser simuleres av servomotor 2). Da må det monteres en servomotor som fungerer som vinsj på den roterende plastplaten på heisvognen.
- Ett annet program som kan realiseres er balansering av objekter. Dette kan gjøres ved å legge heismodellen vertikalt som vist i figur 7.1, og la servomotor 1 gå mellom gitte posisjoner samtidig som servomotor 2 balanserer ett objekt som plasseres på den roterende plastplaten som er montert på heisvognen. Servomotor 2 vil da kunne balansere objektet ved å motvirke kreftene som bevegelsene fra servomotor 1 utøver på objektet.



Figur 7.1: Figuren viser heismodellen lagt ned i horisontal retning. Til venstre i figuren (det som er øvereste punkt når heismodellen står i vertikal retning) er det montert en fot som gjør at heismodellen står stødig i horisontal retning.



Figur 7.2: I figuren vises det samme som i figur 7.1 fra en annen vinkel.

Bibliografi

[1] Omron Industrial Automation, *NX102-1000, PLS*. Hentet 12. februar 2021, fra `https://industrial.omron.no/no/products/nx1-`.

[2] Omron Industrial Automation, *NA5-7W001S-V1, HMI*. Hentet 12. februar 2021, fra `https://industrial.omron.no/no/products/NA5-7W001S-V1`.

[3] Omron Industrial Automation, *S8VS-A/B, Strømforsyning*. Hentet 12. februar 2021, fra `https://industrial.omron.no/no/products/s8vs-a-b`.

[4] Omron Industrial Automation, *W4S1-05B, Ethernet-svitsj*. Hentet 12. februar 2021, fra `http://www.ia.omron.com/products/family/2005/specification.html`.

[5] Omron Industrial Automation, *NX-ECC203, EtherCat modul*. Hentet 12. februar 2021, fra `https://industrial.omron.no/no/products/NX-ECC203`.

[6] Omron Industrial Automation, *NX-AD3604, AD kort*. Hentet 12. februar 2021, fra `https://industrial.omron.no/no/products/NX-AD3603`.

[7] Omron Industrial Automation, *R88D-KN01H-ECT, Servodriver 100W*. Hentet 12. februar 2021, fra `https://industrial.omron.no/no/products/R88D-KN01H-ECT`.

[8] Omron Industrial Automation, *R88D-KN04H-ECT, Servodriver 400W*. Hentet 12. februar 2021, fra `https://industrial.omron.no/no/products/R88D-KN04H-ECT`.

[9] Omron Industrial Automation, *R88M-K10030H-S2, Servomotor 100W*. Hentet 12. februar 2021, fra `https://industrial.omron.no/no/products/R88M-K10030H-S2`.

[10] Omron Industrial Automation, *R88M-K40030H-S2, Servomotor 400W*. Hentet 12. februar 2021, fra `https://industrial.omron.no/no/products/R88M-K40030H-S2`.

[11] IFM, *O1D105, Avstandsensor*. Hentet 12. februar 2021, fra `https://www.ifm.com/us/en/product/O1D105`.

[12] Omron Industrial Automation, *E2A-s, Nærhets-sensor*. Hentet 12. februar 2021, fra `https://industrial.omron.no/no/products/e2a-s`.

[13] Dag Håkon Hanssen, *Programmerbare Logiske Styringer*. Fagbokforlaget, 4 ed., 2018.

[14] Finn Haugen, *Dynamiske Systemer*. TechTeach, 3 ed., 2016.

Vedlegg A

Timeliste

Navn/uke	Ørjan	Vetle	Emil
1	45	45	45
2	43	43	43
3	44	44	12
4	39	39	12
5	40	40	40
6	42	42	42
7	41	41	41
8	40	40	12
9	30	30	12
10	30	30	30
11	43	43	43
12	40	40	40
13	42	42	42
14	45	45	45
15	43	43	43
16	46	46	12
17	50	50	50
18	16	16	16
19	53	53	53
Sum	772	772	633
Totalt	2177 timer		

Vedlegg B

Programfiler

Vedlegg C

Teknisk dokumentasjon

Vedlegg D

Generelt om Sysmac Studio og Matlab

Vedlegg E

Programfiler til laboppgaver