Universitetet
i Stavanger

DET TEKNISK-NATURVITENSKAPELIGE FAKULTET
BACHELOROPPGAVE

Studieprogram/spesialisering: Automatisering og elektroteknikkdesign	Vårsemesteret, 2021 Åpen
Forfattere: Alide Irene Gundersen Niluja Mahesan	
Fagansvarlig: Tormod Drengstig Veileder(e): Tormod Drengstig og Roald Klingsheim	
Tittel på bacheloroppgaven: Posisjonsregulering og hastighetsregulering av en vinsj Engelsk tittel: Position control and speed control of a winch	
Studiepoeng: 40 (2*20)	
Emneord: vinsj open loop vektorkontroll closed loop vektorkontroll posisjonsregulering hastighetsregulering	Sidetall: 94 + Vedlegg/annet : 4 Vedlegg Dato/år: Stavanger, 15.06.21

Bacheloroppgave

Alide Irene Gundersen og Niluja Mahesan

Våren 2021

Sammendrag

Denne bacheloroppgave tar for seg to forskjellige måter å regulere et vinsjsystem med en demomodell gitt av Omron. Demomodellen bestod av en frekvensomformer, motor og enkoder. På motoren ble det implementert en trommel slik at motor og trommel fremstår som en vinsj. Vinsjsystemet ble regulert via en PLS. Kommunikasjonen mellom PLSen og frekvensomformeren ble gjort via EtherCat.

Det ble testet hvilken motor som passet best for systemet. Det var et valg mellom tre motorer, hvor alle var asynkrone trefasemotorer.

En matematisk modell av en vinsj ble utledet, som ble implementert i PLS programvaren Sysmac Studio. Modellen var grunnlaget for alle funksjonene som ble utviklet i systemet.

En *teach*-funksjon ble utviklet basert på den matematiske modellen, som går ut på at funksjonen lærer seg hvor høyt over bakken en last har kommet. *Teach* funksjonen ble brukt til å sette en maks høyde, som automatisk stopper innspoling.

Posisjonsregulering ble utviklet for å regulere vinsjen basert på høyden over bakken til en last. Til posisjonsregulering ble en P-regulator, fartsbegrensning og høydebegrensning implementert.

Hastighetsregulering ble også implementert hvor innspolingshastigheten til systemet forblir konstant. Til hastighetsregulering ble en PI-regulator og fartsbegrensning implementert, men ikke høydebegrensning.

Innhold

Figurer

Tabeller

Videoer

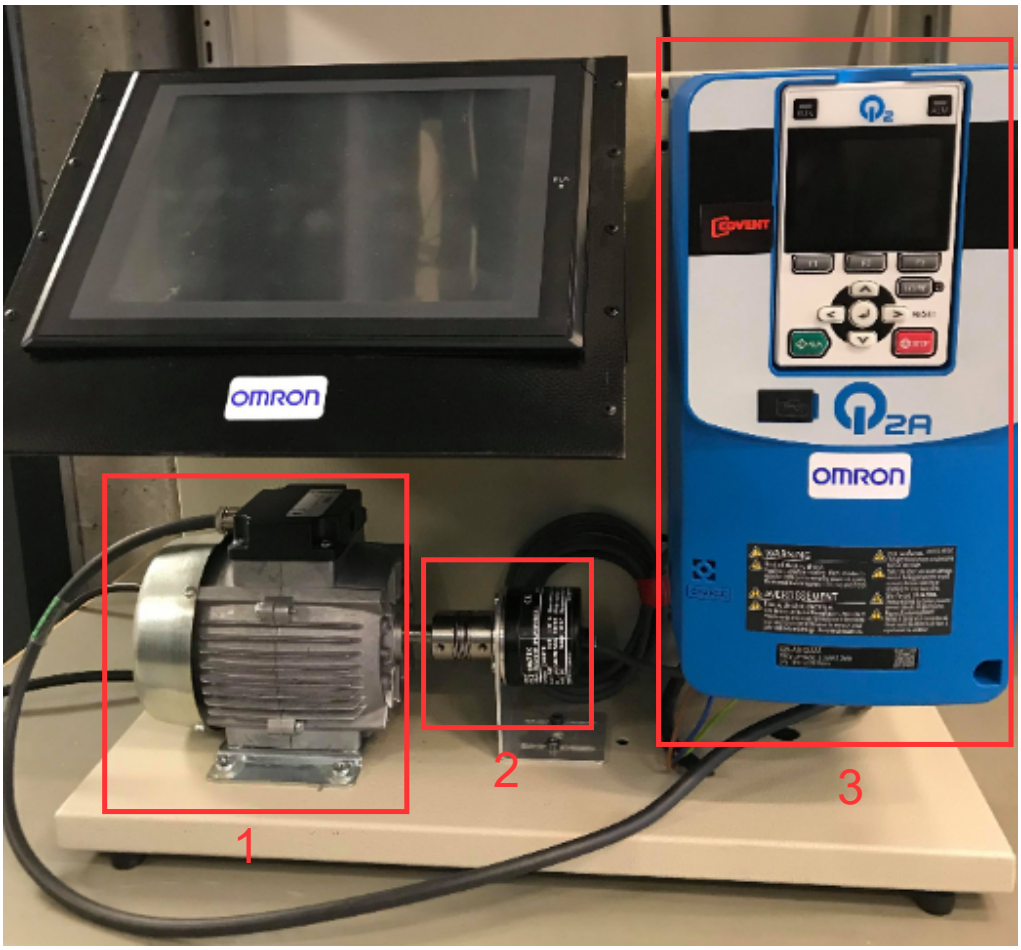
Kapittel	Navn på testen	Videolinken
4.2	Testing av *Teach* funksjonen	`https://www.youtube.com/watch?v=hGuJlLimJDY`
5.3.1	Posisjonsregulering $K_p = 0.01$	`https://www.youtube.com/watch?v=yT5BmVIFYO4`
5.3.1	Posisjonsregulering $K_p = 0.04$	`https://www.youtube.com/watch?v=BCxTIUNEtSs`
6.3.3	Hastighetsregulering $K_i = 0.1$	`https://www.youtube.com/watch?v=powrU1KBiSA`
6.3.3	Hastighetsregulering $K_i = 1$	`https://www.youtube.com/watch?v=dlohKr7IZww`
6.3.4	Hastighetsregulering $K_p = 0.05$ og $K_i = 0.1$	`https://www.youtube.com/watch?v=KbwIeTrQLhk`

Tabell 1: Videoer til prosjektet

Kapittel 1

Innledning

Omron har en demomodell bestående av en motor, frekvensomformer og enkoder, som vist i figur 1.1. På akslingen til motoren ble en trommel montert, for å lage et vinsjsystem. Denne vinsjen ble posisjonsregulert og hastighetsregulert. Videre skal Omron bruke systemet til å holde kurs for industrien.



Figur 1.1: Demomodellen fra Omron med motoren (1), enkoderen (2), og frekvensomformeren (3).

Det var et ønske fra Omron å implementere i PLS-koden det som på engelsk kalles en *teach*-funksjon. Ved å aktivere denne funksjonen vil styresystemet finne ut (lære) hvor høyt over bakken en last har kommet og markerer maks høyden. For å lære styresystemet dette, må vaieren på forhånd være helt utspolt før funksjonen aktiveres og vaieren trekkes helt opp. Denne maks høyden brukes som en begrensing for systemet og stopper innspoling automatisk.

1.1 Oppgavebeskrivelse

Under er oppgaveteksten gjengitt:

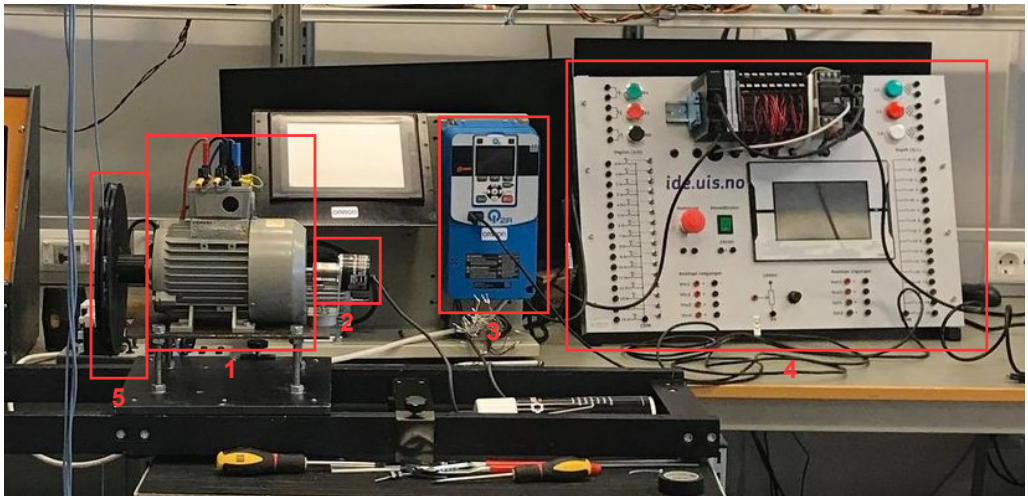
Oppgaven blir derfor å gjøre et litteratursøk på matematiske modeller av vinsjer, og deretter programmere frekvensomformeren på en slik måte at posisjonsregulering og hastighetsregulering demonstreres. En av de reguleringstekniske utfordringene er at momentet og innspolingshastigheten endres etterhvert som vaier spoles opp på vinsjen.

Oppgaveteksten var åpen for tolkning og det var flere alternativer for å løse oppgaven. I oppgaven er det gjort en del konkretisering slik at oppgaven ikke ble for omfattende:

- Lage en matematisk modell av en vinsj og implementere denne i Sysmac Studio.
- Operere systemet i *closed loop* vektorkontroll, og forklare hvorfor dette er optimalt for systemet.
- Utvikle en HMI på en berøringsskjerm hvor trommeldata legges inn.
- Innspolt vaierlengde vises på skjermen.
- Når trommel er full så skal innspoling stoppe.
- Posisjonsregulere vinsjen.
- Hastighetsregulere vinsjen.

1.2 Overordnet om systemet

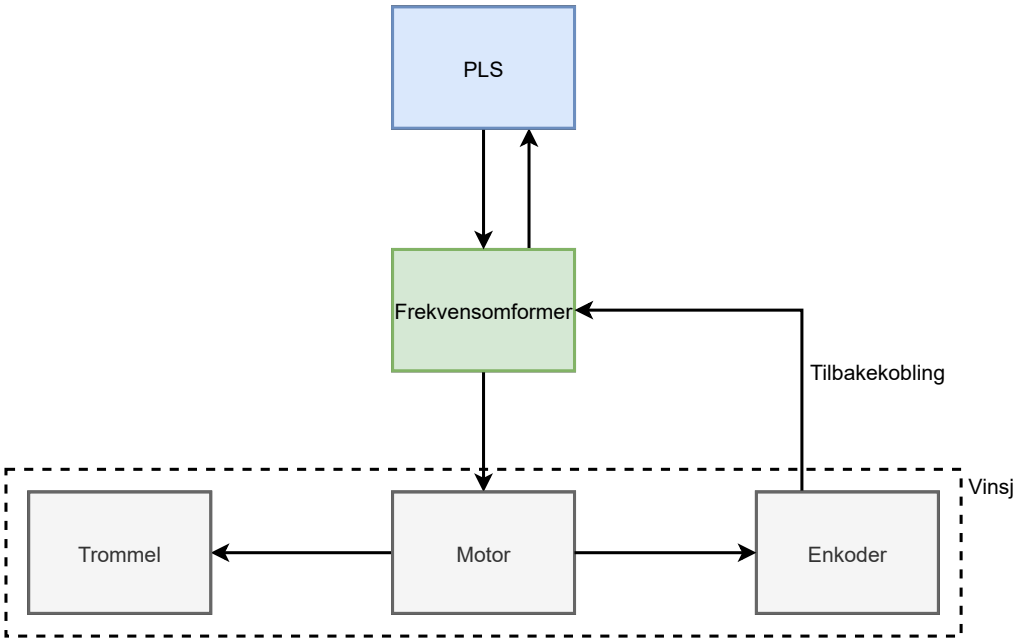
Figur 1.2 viser bilde av hele systemet.



Figur 1.2: Demomodellen fra Omron med motoren (1), enkoderen (2), frekvensomformeren (3), PLSen med skjerm (4), og trommelen (5).

Figur 1.3 viser en illustrasjon av hele systemet, basert på figur 1.2.

1.2 Overordnet om systemet



Figur 1.3: Blokkskjema av det overordnede systemet.

PLSen som ble brukt i prosjektet er av NX102 typen, og ble brukt til å kommunisere med frekvensomformeren. EtherCAT ble benyttet for å forsikre kommunikasjon mellom PLS og frekvensomformeren. Programvaren Sysmac Studio ble brukt til å lage programmene, som ble skrevet i både ladderkode og ST-kode (strukturert tekst).

Frekvensomformeren Q2A kontrollerer hastigheten og dreiemomentet til induksjonsmotoren ved å variere inngangsfrekvensen og spenningen til motoren. *Closed loop* vektorkontroll er en måte å styre motoren via frekvensomformeren med tilbakekobling til en enkoder, som vi benytter i denne oppgaven. Frekvensomformeren hadde sin egen programvare Q2edit, som ble brukt til å overvåke og hente dataer fra frekvensomformeren. I tillegg ble den også brukt til å sette inn motorparametere og *autotune* motoren.

Motoren kontrolleres med et elektrisk signal sendt fra frekvensomformeren, som bestemmer hastigheten og posisjonen til vinsjen. Rotasjonen i motoren overføres videre til enkoderen.

Rotasjonsenkoderen virker som en sensor, som registrerer posisjonen til vinsjen. Enkoderen sender pulser til frekvensomformeren som sender datane videre til PLSen,

der pulsene ble benyttet til å lage programmer som beregnet høyden til lasten over bakken.

1.3 Rapportens struktur

Rapportens struktur er som følger:

- Kapittel 1 - Introduksjon til oppgaven.
- Kapittel 2 - Her presenteres utstyret som ble brukt i oppgaven. De to forskjellige vektorkontroll metodene testet i oppgaven er også forklart, *open loop* og *closed loop* vektorkontroll. Hvor den ene er vektorkontroll uten enkoder og den andre er vektorkontroll med enkoder henholdsvis.
- Kapittel 3 - Her presenteres valget mellom tre utdelte motorer, og begrunnelse av valget.
- Kapittel 4 - Presentasjon av teach funksjonen, og testing av funksjonen.
- Kapittel 5 - Presenterer posisjonsreguleringen av vinsjen, koden bak reguleringen og tester av reguleringen.
- Kapittel 6 - Kapittelet presenterer hastighetsreguleringen av vinsjen, koden bak reguleringen og tester av reguleringen.
- Kapittel 7 - Kapittelet inkluderer diskusjon om vanskeligheter møtt ved løsing av oppgaven. Den fremstiller også måter systemet kunne bli videreutviklet på.
- Kapittel 8 - Konklusjon av oppgaven.

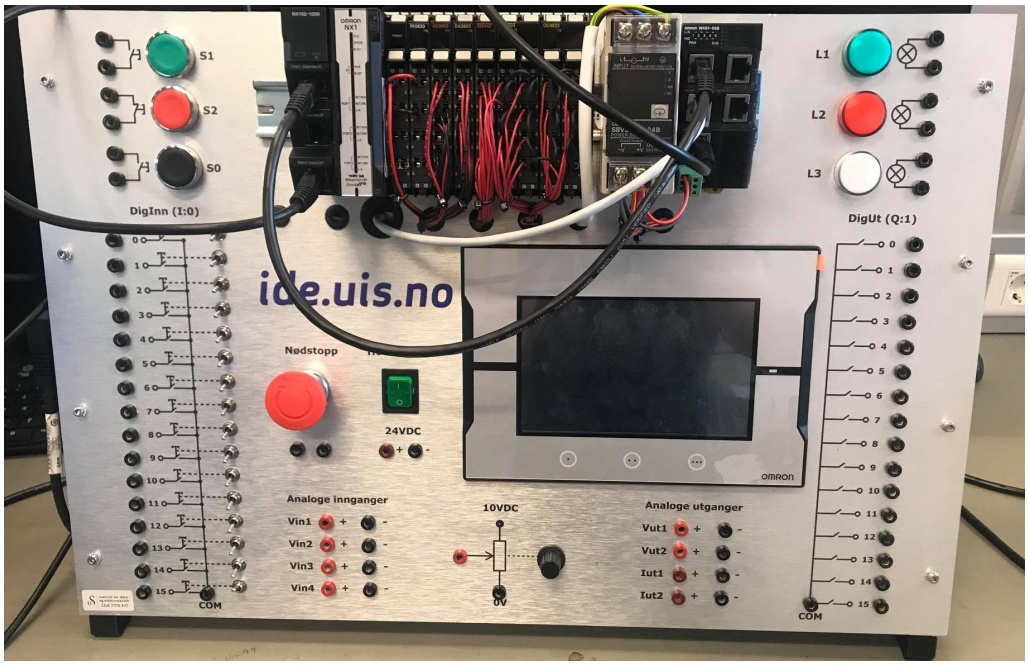
Kapittel 2

Beskrivelse av utstyr og programvare

Kapittelet tar for seg beskrivelse av utstyr som ble brukt i oppgaven. I tilleg tar kapittelet for seg teorien om de ulike metodene som ble benyttet i systemet.

2.1 PLS

PLSen Omron NX102-1000 ble benyttet i oppgaven og ble brukt til å regulere vinsj-systemet. Figur 2.1 viser PLSen som ble brukt i oppgaven. Det er antatt at kunnskap om PLS er forhåndsforstått, og blir derfor ikke forklart videre.



Figur 2.1: Bilde av PLSen som ble benyttet.

For å programmere PLSen ble programvaren *Sysmac studio* brukt. Programvaren ble benyttet til å programmere alle funksjonene i oppgaven. For å oppnå kommunikasjon mellom frekvensomformeren og PLSen ble ferdigprogrammerte funksjonsblokker fra Omron benyttet.

2.1.1 Ferdigprogrammerte funksjonsblokker

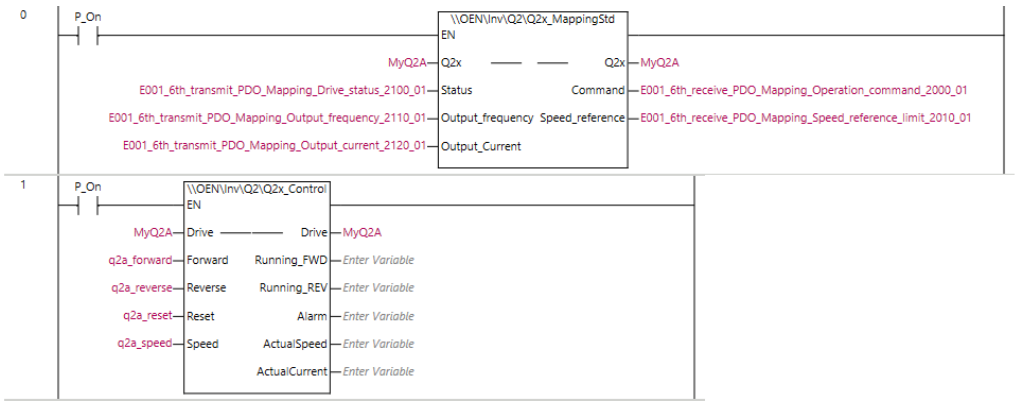
To ferdiglagde funksjonsblokker ble utdelt, Q2x_MappingStd og Q2x_Control. Funksjonsblokkene ble brukt sammen for å styre frekvensomformeren via EtherCAT. For at det er mulig å styre frekvensomformeren via EtherCat må parameterene b1-03 og b1-02 settes lik 3 i frekvensomformeren. Disse parameterene er forklart i delkapittel 2.3. En ESI-fil ble også utdelt fra Omron, som måtte bli lagt i ESI-mappa til Sysmac Studio før oppstart. Dette gjør at PLSen kjenner igjen frekvensomformeren. Deretter legger man til frekvensomformeren på EtherCAT, og redigerer PDO Mapping som vist i figur 2.2.

2.1 PLS

Position	Port	Description	R/W	Data Type	Variable	Variable Comment	Variable Type
	EtherCAT Network Configuration						
Node1	Q2A series						
	6th receive PDO Mapping_Operation command_2000_01		W	UINT	E001_6th_receive_PDO_Mapping_Operation_command_2000_01		Global Variables
	6th receive PDO Mapping_Speed reference/limit_2010_01		W	UINT	E001_6th_receive_PDO_Mapping_Speed_reference_limit_2010_01		Global Variables
	6th transmit PDO Mapping_Drive status_2100_01		R	UINT	E001_6th_transmit_PDO_Mapping_Drive_status_2100_01		Global Variables
	6th transmit PDO Mapping_Output frequency_2110_01		R	UINT	E001_6th_transmit_PDO_Mapping_Output_frequency_2110_01		Global Variables
	38th transmit PDO Mapping_Output current_2120_01		R	UINT	E001_6th_transmit_PDO_Mapping_Output_current_2120_01		Global Variables
	38th transmit PDO Mapping_U1-08_2651_08		R	UINT	effekt		Global Variables
	39th transmit PDO Mapping_Motor speed_2200_01		R	UINT	E001_39th_transmit_PDO_Mapping_Motor_speed_2200_01		Global Variables
	39th transmit PDO Mapping_U6-18_2656_12		R	UINT	enkoder_puls		Global Variables

Figur 2.2: I/O Map

Detaljert forklaring om fremgangsmåte for å oppnå kommunikasjon mellom PLSen og frekvensomformeren er vist i vedlegg A. De ferdiglagde funksjonsblokkene ble lagt inn i programmet og fylt ut som vist i figur 2.3.



Figur 2.3: Ferdiglagde funksjonsblokkene for å styre omformeren.

Fra figur 2.3 ser man at det er kun fylt inn for `q2a_forward`, `q2a_reverse`, `q2a_reset` og `q2a_speed`. Det er fordi det er disse funksjonene som ble brukt for å styre vinsjen. `q2a_forward`-funksjonen går fremover/heiser opp, mens `q2a_reverse`-funksjonen går bakover/heiser ned. `q2a_reset` resetter frekvensomformeren, og `q2a_speed` styrer referansefrekvensen `Freq_ref`. Det ble valgt å sette `q2a_speed` lik `frek_hast` i koden vist i figur 2.4.

Figur 2.4: Setter `q2a_speed` lik `frek_hast`.

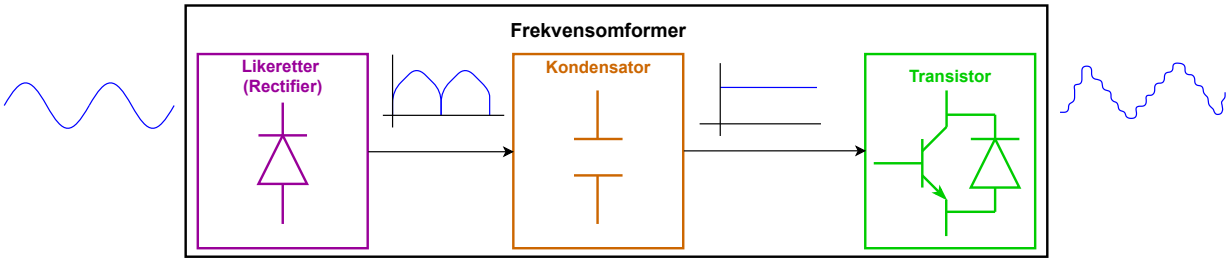
2.2 Frekvensomformeren Q2A

Frekvensensomformeren Q2A-A2012-AAA ble tildelt fra Omron. Frekvensomformeren er i stand til å kontrollere forskjellige elektriske motortyper og har kapasitet til å styre motorer opptil 590 Hz. Den har Ethernet basert kommunikasjon og for å oppnå kommunikasjon med PLSen ble EtherCat benyttet. Figur 2.5 viser bilde av frekvensomformeren.

Figur 2.5: Bilde av frekvensomformeren brukt i oppgaven.

En frekvensomformer er en form for motorkontroll som styrer en elektrisk motor ved å variere frekvensen og spenningen som leveres til motoren. Motorhastigheten er direkte relatert til frekvensen, høyere frekvens tilsvarer høyere motorhastighet. Hvis en applikasjon krever at en motor ikke drives med maks hastighet, kan frekvensomformeren brukes til å begrense motorhastigheten. I tillegg kan man også bruke frekvensomformeren til å rampe opp en motor for jevn oppstart, eller for å forhindre at tung belastning ved oppstart skader motoren. Dette oppnås ved å justere frekvensen som leveres til

motoren [5]. Figur 2.6 viser prosessen som skjer i frekvensomformeren for å styre en elektrisk motor.

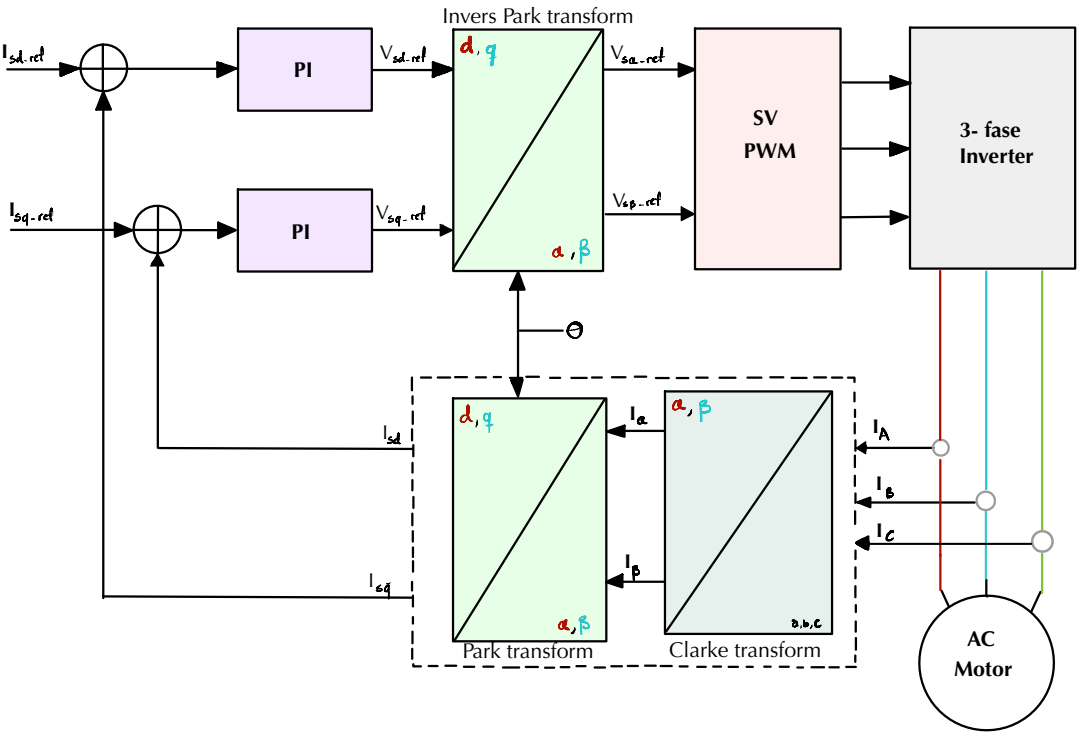


Figur 2.6: Prosessen som skjer i frekvensomformeren for å styre en elektrisk motor, forklaring gitt i teksten.

Systemet består av en elektrisk forsyning, frekvensomformer og en vekselstrøm-motor. Figur 2.6 viser en enkel frekvensomformer som består av en likeretter, kondensator og vekselretter (transistor). Vekselstrøm (AC) blir konvertert til likestrøm (DC) via likeretteren. Deretter blir signalet sendt til kondensatoren, som filtrerer spenningen. Til slutt konverterer frekvensomformeren likestrøm tilbake til vekselstrøm ved bruk av transistoren som oppfører seg som en bryter. Transistoren bruker moduleringsteknikker som pulsbreddemodulering (PWM) til å vareiere utgangsfrekvensen for å kontrollere hastigheten på motoren [2].

2.2.1 Vektorkontroll (feltorientert styring, FOC)

Vektorkontroll er en måte å kontrollere en elektrisk motor ved hjelp av strøm- og spenningsmålinger, som vist i figur 2.7. Vektorkontrollprosessen går ut på å transformere et trefasesystem som er avhengig av tid og hastighet til et to-koordinat system (d og q). Transformeringen gjennomføres ved bruk av en matematisk modell som er basert på Clarke transformasjon og Park transformasjon [4].



Figur 2.7: Et enkelt blokkskjema av vektorkontroll for vekselstrømsmotorer.

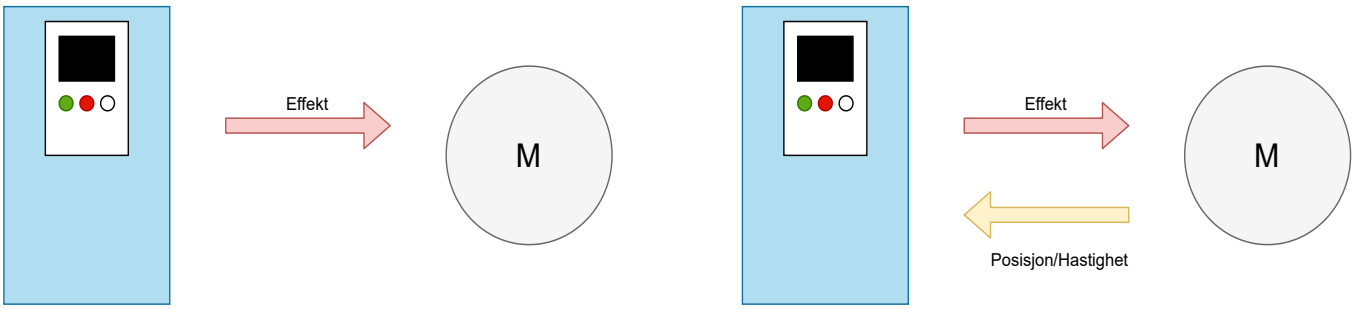
Frekvensomformeren prøver å kontrollere to forskjellige typer strøm i en vekselstrømsmotor, som vist i figur 2.7:

1. **Reaktiv strøm** som brukes til å skape magnetfelt (I_B, q).
2. **Aktiv strøm** som brukes til å produsere dreiemoment for å vri rotoren (I_A, d).

Den reaktive strømmen er tilnærmet konstant og styres av en fluks regulator. Den aktive strømmen brukes enten til å kontrollere dreiemoment, hastighet og/eller effekt. Dreiemoment er gitt direkte av mengden aktiv strøm, som igjen er gitt av spenning. Altså høyere spenning tilsvarer mer strøm, som øker hastigheten og/eller effekt [10]. Videre forklaring av vektorkontrollprosessen var ikke en del av oppgaven.

2.2.2 *Open loop* vektorkontroll

Det finnes hovedsakelig to typer vektorkontroll, *open loop* og *closed loop*. Hovedforskjellen mellom disse er at *closed loop* vektorkontroll har tilbakekobling via en enkoder, mens *open loop* er ikke tilbakekoblet, som vist i figur 2.8.

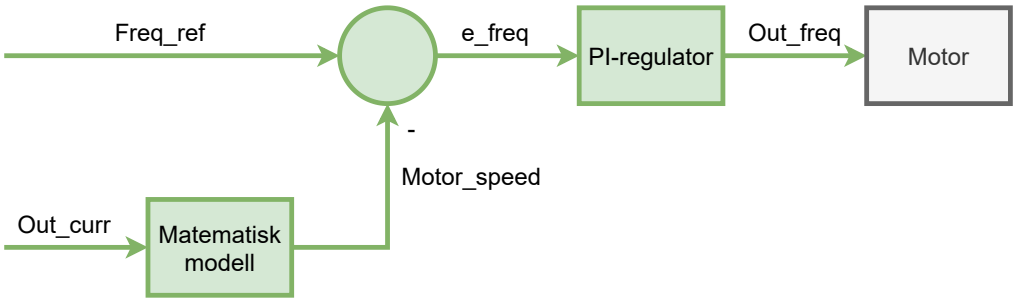


(a) *Open loop* (b) *Closed loop*

Figur 2.8: *Open loop* og *closed loop* vektorkontroll fra frekvensomformer til motor.

Som nevnt tidligere har ikke *open loop* vektorkontroll en tilbakemeldingsenhet. Det er fordi *open loop* vektorkontroll er basert på en matematisk modell av motorparameterene. Frekvensomformeren overvåker strømmen og spenningen levert til motoren. Deretter sammenligner frekvensomformeren disse målingene med den matematiske modellen, og gjør feilkorreksjoner på den leverte strømmen. Endringene gjort på strømmen endrer også momentet og motorhastigheten [4].

Open loop vektorkontroll bruker en matematisk modell for å finne den optimale utgangsspenningen for at motoren oppnår ønsket frekvens, som vist i figur 2.9. Ved kontinuerlig strømavlesning og raske beregninger utført i frekvensomformeren, beregnes utgangsfrekvensen. Figur 2.9 viser et forenklet blokkskjema av *open loop* vektorkontroll i denne frekvensomformeren, som er basert på informasjon gitt av Omron.



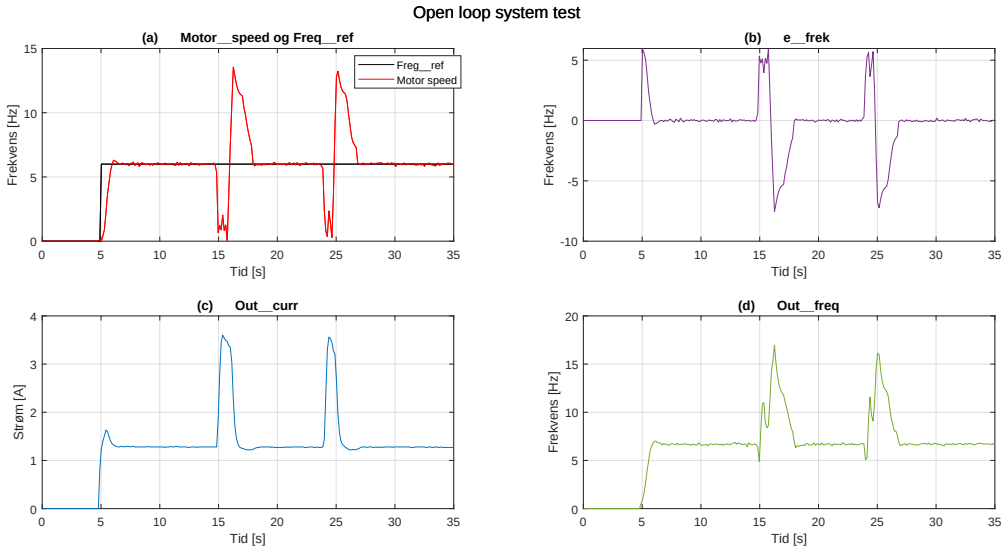
Figur 2.9: Forenklet blokkskjema av *open loop* vektorkontroll i denne frekvensomformeren, den grønne delen er frekvensomformeren og den gråe er motoren.

Parameteren `Freq_ref` er referansen i reguleringsløyfa, og tilsvarer ønsket motorhastighet oppgitt i frekvens [Hz]. Begrepsmessig ser det umiddelbart litt merkelig ut å sammenligne en *referansefrekvens* (`Freq_ref`) med en *motorhastighet* (`Motor_speed`), men poenget er at begge har benevning [Hz]. For å beregne `Motor_speed` anvender frekvensomformeren en matematisk modell av motoren. Denne modellen bruker motorparameterene som frekvensomformeren får via en initialiseringsprosess kalt *autotuning*. Under denne prosessen skrives motorparameterene inn i frekvensomformeren som videre gjør strøm- og spenningsmålinger basert på dette. Mer om autotunignsprosessen kommer videre i delkapittel 2.3.1.

Som figur 2.9 viser, er strømmålingen `Out_curr` den eneste inngangsvariabelen til den matematiske modellen. Siden `Motor_speed` på denne måten er et modellbasert estimat på hvor fort motoren roterer, kalles styremåten for *open loop vektorkontroll*. Basert på reguleringsavviket `e_freq` beregner en innebygget PI-regulator pådraget `Out_freq` som også har benevning [Hz]. Det er ikke mulig å endre på regulatorparameterene til denne PI-regulatoren. Signalet `Out_freq` blir omgjort i frekvensomformeren til strøm og spenning, som blir levert til motoren.

Open loop system test

For å vise et eksempel på hvordan *open loop* vektorkontroll fungerer, ble det utført en test for å demonstrere dette. Motoren ble kjørt med en konstant hastighet hvor `Freq_ref` var satt til 6 Hz. Ved 15 s og 25 s ble akslingen belastet i ca. 1 s, ved å holde hånden over akslingen. Figur 2.10 viser resultatet av forsøket.



Figur 2.10: Resultatet fra testen demonstrerer hvordan open loop vektorkontroll virker ved et kort sprang i forstyrrelsen.

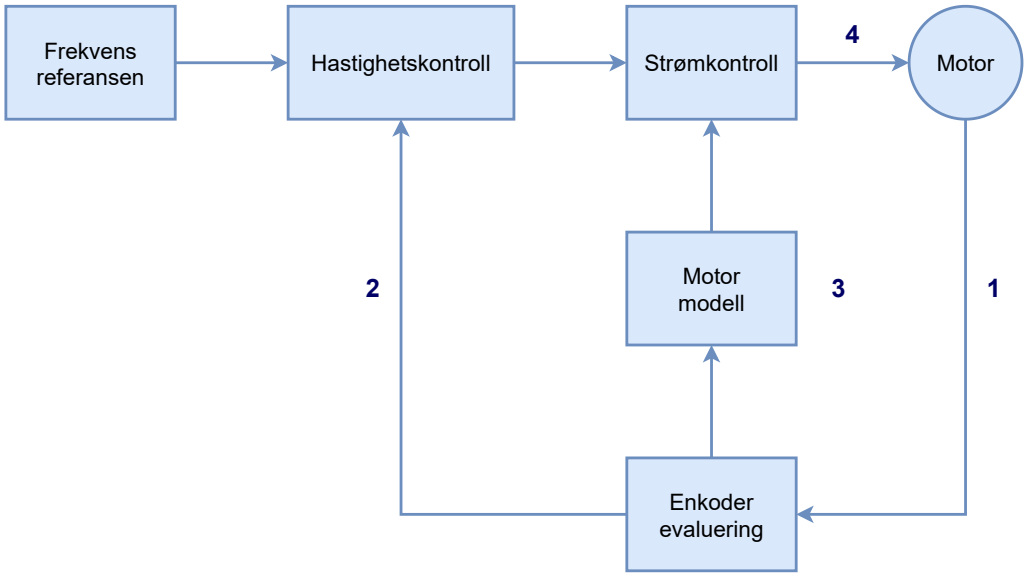
I dette eksperimentet klemte vi hånden rundt akslingen i periodene $15 < t < 16$ s og $24 < t < 25$ s. I disse periodene ser vi at strømtrekket `Out_curr` øker fra 1.3 A til ca. 3.5 A, og estimatet av rotasjonshastighet `Motor_speed` synker fra 6 Hz ned til ca 1 Hz. Reguleringsavviket `e_freq` gir opphav til at `Out_freq` øker raskt i løpet av tidsperiodene, og dette opplevde vi som at momentet på motoren ble større, og som også var grunnen til at vi ikke klarte å holde på akslingen i mer enn 1 sekund. Etter at vi slapp taket i akslingen opplevde vi at motoren roterte mye raskere i en kort periode, og dette bekreftes av estimatet av `Motor_speed` som viser et oversving i rotasjonshastighet. Årsaken er at PI-regulatoren har integrert reguleringsavviket `e_freq` og at pådraget `Out_freq` er nær 15 Hz når vi slipper.

2.2.3 *Closed loop* vektorkontroll

Closed loop vektorkontroll benytter en enkoder for å sende pulser til mikroprosessoren som er koblet til frekvensomformeren. Pulsene blir andvendt for å gi indikasjon om posisjonen. Hvis mikroprosessoren forventer en posisjon på x radianer, men enkoderen har posisjonen x-2 radianer endrer mikroprosessoren PWM-signalet for å feilkorrigere. Kort sagt brukes enkoderen til å gi presis posisjons tilbakemelding ved å telle pulser [9].

I *closed loop* vektorkontroll er det mulig å få en vekselstrømsmotor til å utvikle kontinuerlig fullt dreiemoment ved null rotasjonshastighet. Dette gjør *closed loop* vektorkontroll egnet for kran og heiseanvendelser der motoren må produsere fullt dreiemoment før bremsen slippes [9].

Hvordan *Closed loop* vektorkontroll fungerer er vist i figur 2.11:

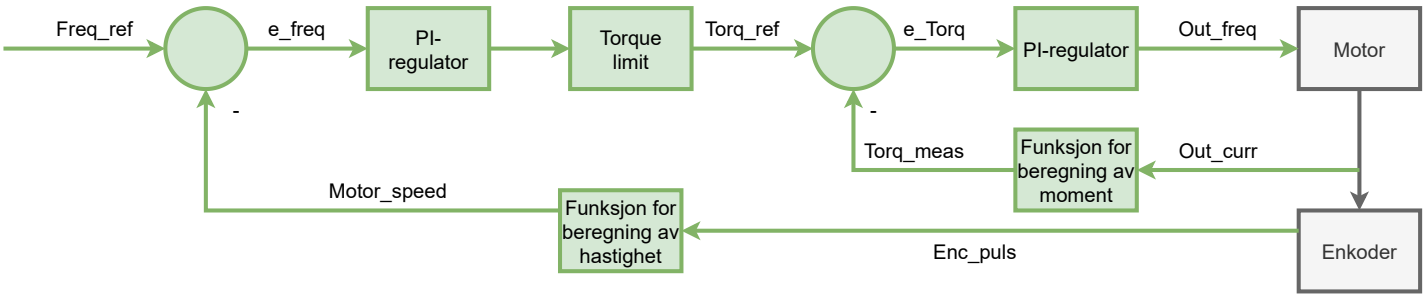


Figur 2.11: Generellt blokkskjema for *closed loop* vektorkontroll i frekvensomformeren [11].

1. Motorens rotasjon skaper pulser i enkoderen og pulsene blir overvåket av frekvensomformeren via tilbakekoblingen.
2. Pulsene fra enkoderen blir andvendt i hastighetskontroll. Hastighetskontrollen sammenligner frekvensomformerens frekvensreferansen og den målte hastigheten. En PI-regulator justerer pådraget for å gjøre opp for feilen.
3. Pulsene fra enkoderen blir også brukt til motormodellen. Informasjon om motoren blir brukt til å lage en virtuell modell av motoren. Ved å bruke modellen og den målte utgangsspenningen og utgangsstrømmen kalkulerer frekvensomformeren hva motorhastigheten bør justeres til.
4. Strømkontroll bruker informasjonen fra motormodell og hastighetskontroll til å kalkulere hvordan strømmen bør justeres for å oppnå ønsket hastighet og moment ut ifra frekvensreferansen.

2.2 Frekvensomformeren Q2A

Basert på den generelle teorien om *closed loop* vektorkontroll i figur 2.11 og et blokkdiagram fra brukermanualen til frekvensomformeren i vedlegg B, ble et blokkskjema for systemet utviklet vist i figur 2.12.



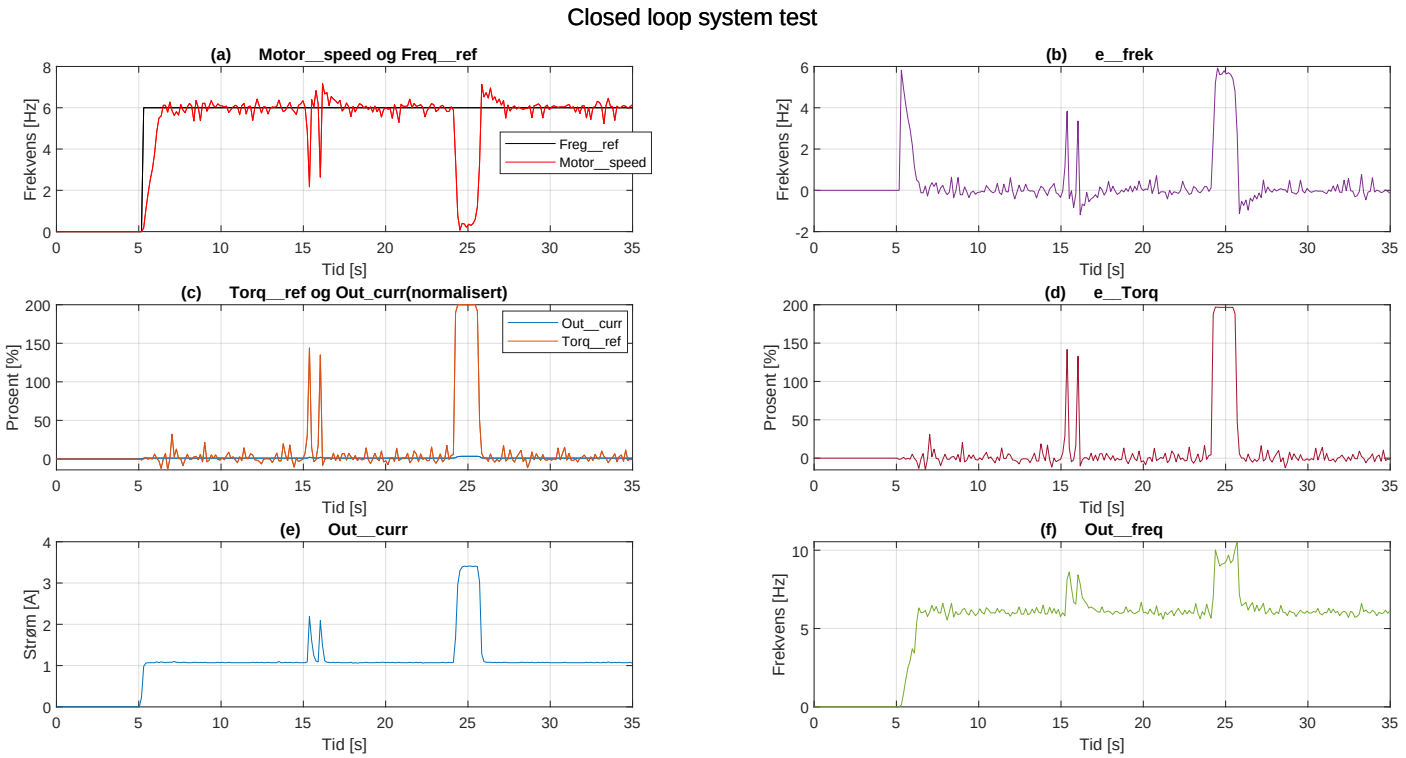
Figur 2.12: Blokkskjema av *closed loop* vektorkontroll i frekvensomformeren, den grønne delen er omformeren og den grå er maskinvare.

Figur 2.12 viser blokkskjema av *closed loop* vektorkontroll for systemet. Pulsene fra enkoderen `Enc_puls` blir anvendt til å beregne *motorhastigheten* `Motor_speed` gjennom en matematisk modell. *Frekvensreferansen* `Freq_ref` og *motorhastigheten* `Motor_speed` blir sammenlignet, sammenligningen mulig fordi begge parameterene er oppgitt i [Hz] som nevnt før. Reguleringsavviket `e_freq` går gjennom en innebygget PI-regulator og videre til en funksjon som begrenser momentet kalt *Tourque limit*. Tourqe limit er prosent basert på hvor mye strøm motoren tåler og skaper *momentreferansen* `Torq_ref`. Vi antok at `Torq_meas` er momentet beregnet fra strømmen, fordi `Torq_ref` er basert på strømmen. Basert på reguleringsavviket `e_Torq` beregner en innebygget PI-regulator pådraget `Out_freq` som har benevning [Hz]. I likhet med open loop vektorkontroll er det ikke mulig å endre på reguleringsparameterene til den innebygde PI-regulatoren.

Closed loop system test

For å se hvordan *closed loop* vektorkontroll reagerer på belastning ble et forsøk utført. Motoren ble kjørt med en konstant hastighet på 6 Hz, og rundt 15 s og 25 s ble akslingen belastet i ca. 1 sekund ved å holde hånden rundt akslingen. Figur 2.13 viser resultatet fra forsøket.

2.2 Frekvensomformeren Q2A



Figur 2.13: Resultatet fra testen demonstrerer hvordan *closed loop* vektorkontroll virker ved et kort sprang i forstyrrelsen.

I dette eksperimentet klemte vi hånden rundt akslingen i periodene $15 < t < 16$ s og $24 < t < 25$ s. I perioden $15 < t < 16$ s glapp vi akslingen, som er grunnen til at den ikke har like stort endring som grafen i perioden $24 < t < 25$ s. Men vi antar at grafene hadde sett like ut hvis akslingen hadde blitt holdt konstant i 1 s, og de blir forklart som om de er det.

I disse periodene ser vi at strømtrekket `Out_curr` øker fra 1 A til ca 3.4 A, og estimatet av rotasjonshastighet `Motor_speed` synker fra 6 Hz ned til ca 0 Hz. Reguleringsavviket `e_freq` gir opphav til at `Out_freq` øker raskt i løpet av tidsperiodene, og dette opplevde vi som at momentet på motoren ble større, og som også var grunnen til at vi ikke klarte å holde på akslingen i mer enn 1 sekund. Etter at vi slapp taket i akslingen opplevde vi at motoren roterte litt raskere i en kort periode, og dette bekreftes av estimatet av `Motor_speed` som viser et lite oversving i rotasjonshastighet. Årsaken er at PI-regulatoren har integrert reguleringsavviket `e_freq` og at pådraget `Out_freq` er nær 10 Hz når vi slipper.

Torq_ref er en matematisk modell basert på Out_curr og fra figur 2.13 ser man at Torq_ref følger etter Out_curr. e_torq er forskjellen mellom Torq_meas og Torq_ref. Torq_meas var ikke tilgjengelig og derfor måtte den normaliserte strømmen bli brukt i stedet, som er Out_curr multiplisert med en faktor. e_torq grafen ligner på Torq_ref grafen fordi vi ikke har den matematiske modellen som regner ut Torq_meas, og derfor er det bare ren gjetting på hvordan Torq_meas ser ut.

2.2.4 *Closed loop* og *open loop* rotasjonshastighet ved lave hastigheter

For å finne ut om det var en forskjell på rotasjonshastighet med de to forskjellige vektorkontroll metodene ble et forsøk utført. Dette ble gjort ved å avlese verdier fra frekvensomformeren, og å måle omdreininger på akslingen over ett minutt. Resultatet av testen er vist i tabell 2.1.

	Closed loop			*Open loop*		
Output frequency [Hz]	1	2	3	1	2	3
O/min målt med tachometer	30	60	90	82	102	130
Motor speed målt med enkoder [Hz]	0.98 - 1.06	1.98 - 2.07	2.96 - 3.09	-	-	-
Motor speed målt uten enkoder [Hz]	-	-	-	0.98 - 1.06	1.97 - 2.19	2.98 - 3.03
O/min beregnet ut i fra motor speed	29.4 - 31.8	59.4 - 62.1	88.8 - 92.7	29.4 - 31.8	59.1 - 65.7	89.4 - 90.9

Tabell 2.1: Forskjellen på rotasjonshastighet på *closed loop* og *open loop* målt.

Fra tabellen ser man at det er stor forskjell på hastigheten ved *open loop* og *closed loop*.

Omdreininger per minutt utregnet ut i fra motorspeed (nederste rad) og omdreininger målt (andre rad) er nesten det samme ved *closed loop*. Fra tabellen kan man se at ved 1 Hz har motoren en omdreiningshastighet på 30 o/min, som er den samme omdreiningshastigheten beregnet ut i fra motor speed rundt 29.4-31.8 o/min. Dette gjelder også for 2 Hz og 3 Hz.

Fra tabell 2.1 for *open loop* kan man se at omdreiningshastigheten målt (andre rad) og omdreiningshastigheten beregnet ut i fra `motor speed` (nederste rad) ikke er like. For 1 Hz var omdreiningene målt lik 82 o/min, mens beregnet var på 29.4-31.8 o/min. Denne feilen mellom målt og beregnet omdreiningshastighet gjelder også for 2 Hz og 3 Hz, men forskjellen er ikke konsistent.

Closed loop vektorkontroll er mer nøyaktig enn *open loop* vektorkontroll grunnet tilbakekoblingen. Dette gjelder spesielt ved lave hastigheter og ble vist i tabell 2.1.

2.2.5 Fordeler og ulemper med *open loop* og *closed loop*

En ulempe med *open loop* vektorkontroll er at den har vanskeligheter med å overvåke strøm, og utføre dreiemomentjusteringer ved lave motorhastigheter. Av den grunn anbefales ikke *open loop* vektorkontroll for applikasjoner som krever at en last holdes stille. *Open loop* vektorkontroll har muligheten til å produsere 100 prosent av nominelt dreiemoment ved motorhastigheter ned til ca. 8 Hz [4].

Den viktigste fordelen med *closed loop* vektorkontroll er at den gjør det mulig å produsere opptil 200 prosent av motorens nominelle dreiemoment ved 0 rotasjonshastighet. Dette er spesielt nyttig i applikasjoner som krever at en last står stille, for eksempel kraner og heiser. En annen fordel er at *closed loop* mer nøyaktig enn `open loop` ved lave hastigheter, og er derfor bedre å bruke hvor en vinsj arbeider med lave hastigheter [4]. Siden vi jobber med en vinsj var denne egenskapen essensiell.

Da vinsjsystemet ble montert ble ikke motorhastigheter høyere enn 3 Hz benyttet. Fra testen i delkapittel 2.2.4 ble det demonstrert at *open loop* ikke roterte med ønsket hastighet ved lave hastigheter. Dette er også en av de største grunnene til at *closed loop* ble benyttet for systemet.

2.3 Q2edit

Q2edit er et program som lar brukeren programmere frekvensomformeren Q2A. Programmet ble brukt til å bestemme verdier i frekvensomformeren og logge data.

Nummer	Parameter	Nominell verdi
A1-02	Control Method	3: CLVector
A1-03	Init Parameters	0: No Initialization
b1-01	Freq. Ref. Sel 1	0: Keypad / 3: Option PCB
b1-02	Run. Comm. Sel 1	0: Keypad / 3: Option PCB
b1-03	Stopping Method Selection	0: Ramp->Stop
C1-01	Accel Time 1	10,0 sec
C1-02	Decel Time 1	10,0 sec
F1-01	Enc1 Pulse Count (PPR)	2000 PPR

Tabell 2.2: Nyttige innstillinger i frekvensomformeren.

Tabell 2.2 inneholder noen viktige innstillinger i frekvensomformeren som ble brukt. Gjennom parameterene/nummerene har man muligheten til å konfigurere frekvensomformeren etter behov, enten direkte i frekvensomformeren eller via programvaren Q2edit.

- A1-02: bestemmer kontroll metoden. Systemet ble operert i *closed loop* vektorkontroll som har verdien 3. *Open loop* vektorkontroll har verdien 2.
- A1-03: blir brukt til å resette frekvensomformeren med kode: 2220.
- b1-01: brukeren velger å styre frekvensen til frekvensomformeren direkte (*keypad*) eller fra ekstern enhet som for eksempel modbus eller PCB (EtherCAT kort).
- b1-02: brukeren velger å styre run-command til frekvensomformeren direkte (*keypad*) eller fra en ekstern enhet som for eksempel digital signal, modbus eller PCB (EtherCAT kort).
- b1-03: bestemmer hvordan frekvensomformeren blir stoppet.
- C1-01: bestemmer akselerasjonstiden.
- C1-02: bestemmer retardasjonstiden.

- F1-01: setter enkoderens *pulses per revolution* [ppr] ifølge databladet dens.

I tillegg til de nevnte parameteren ble også motor parameterene brukt, altså *autotuning* parameterene som blir forklart videre. Tabell 2.3 viser nødvendige parametere for oppgaven, hvilken nummerkode de har, hva de heter i frekvensomformeren og hva de ble kalt i blokkskjemaer.

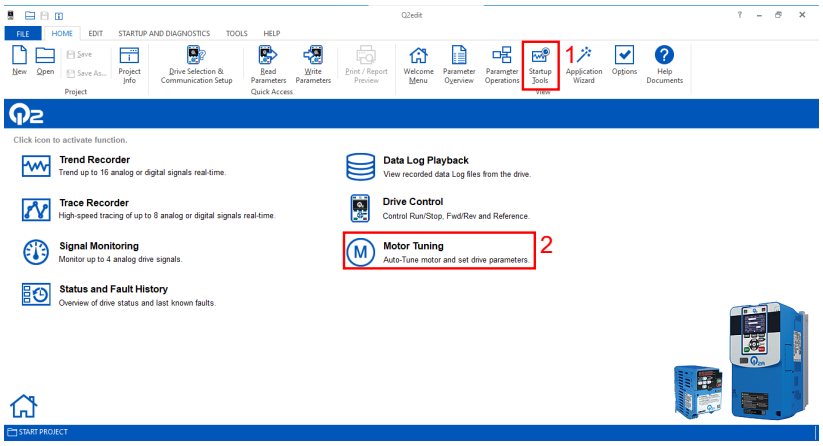
Nummer	Parameter i frekvensomformeren	Blokkskjema
U1-01	Frequency Reference	Freq_ref
U1-02	Output Frequency	Out_freq
U1-05	Motor Speed	Motor_speed
U1-03	Output Current	Out_curr
U1-09	Torque Reference	Torq_ref
U6-18	Encoder Pulse Counter	Enc_puls

Tabell 2.3: Overvåkingsparametere fra frekvensomformeren.

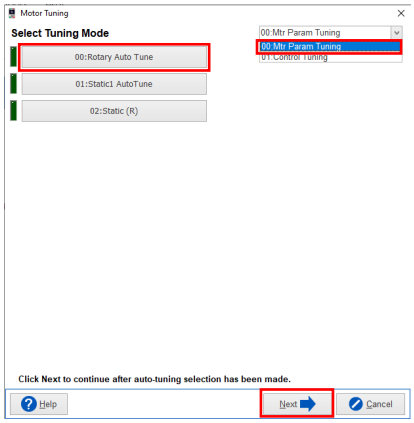
2.3.1 *Autotuning*

Autotuning av motor går ut på at frekvensomformeren kjører gjennom en rekke frekvenser innenfor motorverdier skrevet inn. Når frekvensomformeren er ferdig med *autotuning* lagrer den verdier som strøm, spenning, magnetisering osv. For at frekvensomformeren og motoren kommuniserer bedre med hverandre blir *autotuning* utført. En *autotuner* motoren direkte i frekvensomformeren via *keypad* eller gjennom programvaren Q2edit, som vist i figur 2.14.

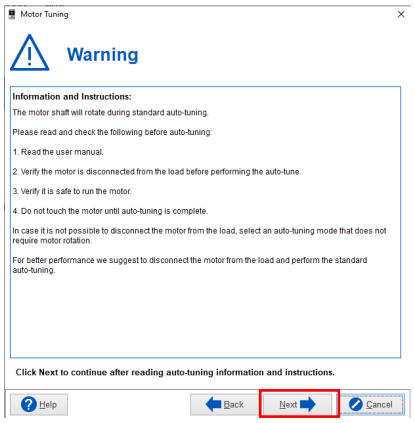
2.3 Q2edit



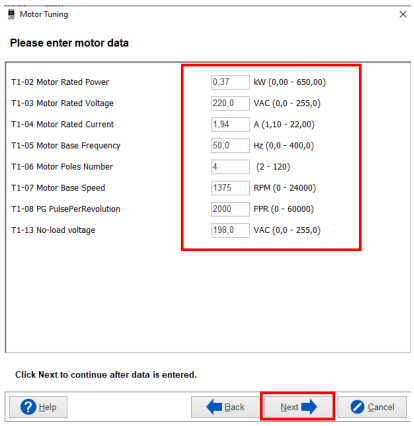
(a) Startup tool -> motor tuning



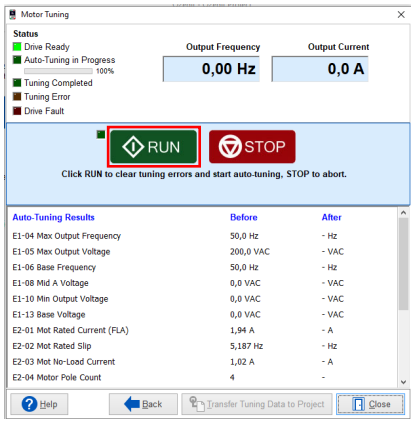
(b) mtr param tuning og rotary autotune mode



(c) Anerkjenner advarselen og trykker next



(d) Sett inn motordataene

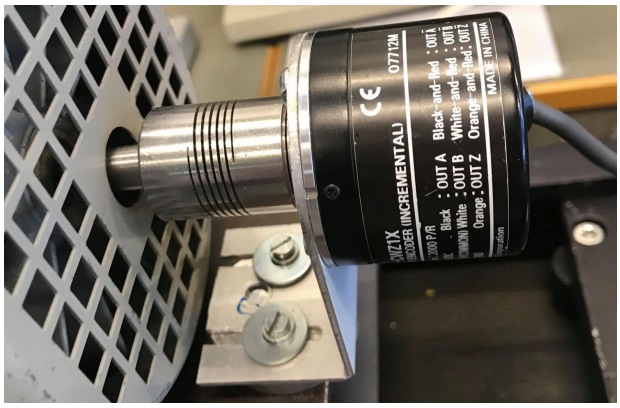


(e) Kjører autotuning

Figur 2.14: Demonstrasjon av *autotuning* via Q2edit.

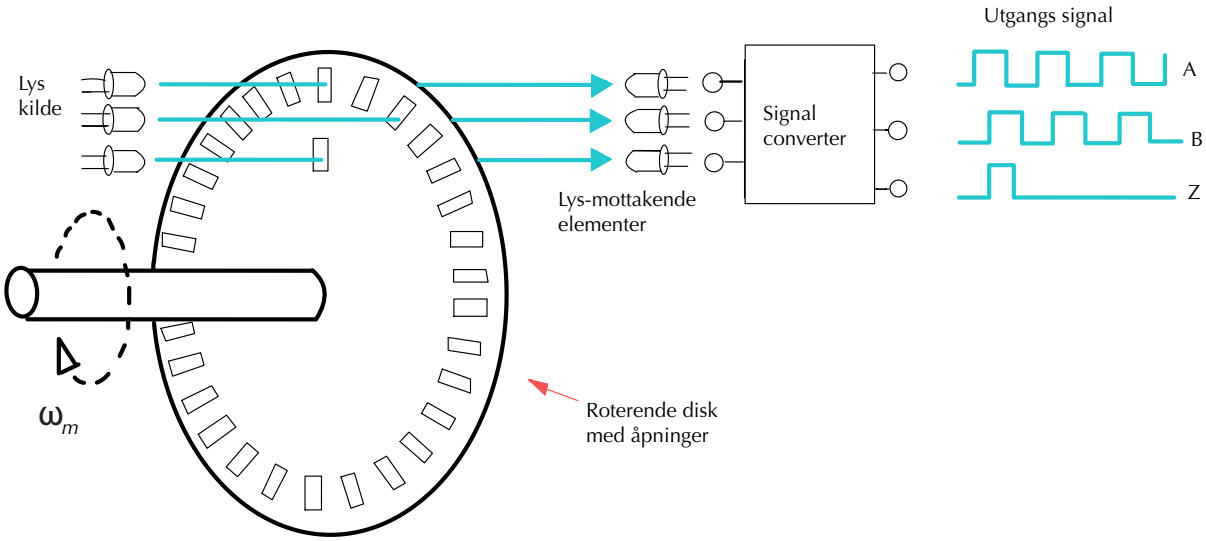
2.4 Enkoder

Enkoderen E6C2-CWZ1X er en inkrementell roterende enkoder og ble tildelt fra Omron sammen med frekvensomformeren. Figur 2.15 viser bilde av enkoderen. En enkoder er en sensor som gir tilbakemelding om hastighet, retning, posisjon eller pulser i et kontrollsystem.



Figur 2.15: Bilde av enkoderen montert på motoren.

En roterende enkoder er en sensor som festes mekanisk til en roterende aksel, og kobles elektrisk til et logikksystem for å sende informasjon til logikksystemet angående rotasjonen av akselen. Den samler inn data og gir ut informasjon basert på rotasjonen til et objekt, som vist i figur 2.16.



Figur 2.16: Skisse av prosessen inne i en enkoderen.

En inkrementell roterende enkoder er en klasse av enkoderenheter som gir en utgang i digital form. De måler vinkelposisjonen til akselen i forhold til et vilkårlig utgangspunkt, men gir ikke en indikasjon på den absolutte posisjonen til akselen. Prinsippet for operasjonen er ved rotasjon genereres pulser og den totale vinkelrotasjonen utledes fra pulstellingen.

En inkrementell enkoder har tre utgangssignal som standard:

- Signal A bestående av n pulser per omdreining.
- Signal B er identisk til signal A men er 90 grader forskjøvet.
- Signal Z er en initialiseringspuls, som signaliserer en omdreining.

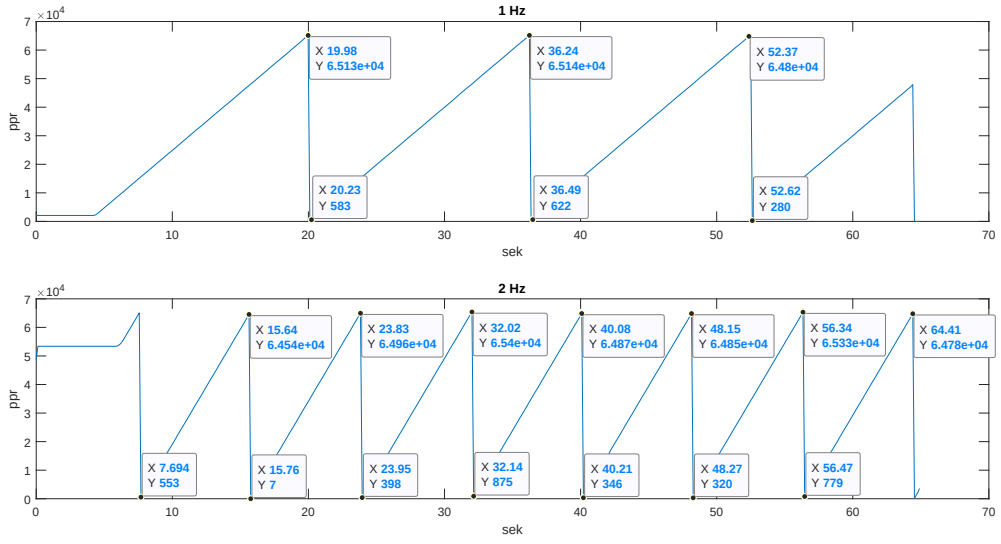
Med A, B og Z kombinert er det mulig å:

1. Bestemme posisjonen til akselen ved å bruke Z-puls og deretter telle pulsene A og B.
2. Bestemme retningen av rotasjonen ved å sammenligne A med B.

2.4.1 Parameteren `Encoder pulse counter`

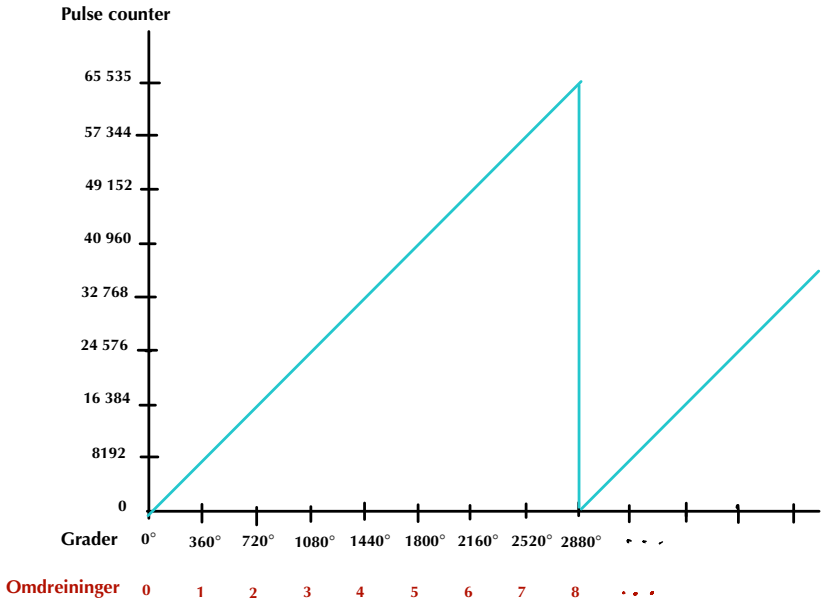
`Encoder pulse counter` er en parameter som ble overvåket fra frekvensomformeren. `Encoder pulse counter` oppfører seg som en teller, hvor ordet den produserer er på 16 bit. Det betyr at telleren har en verdi på $2^{16} = 65536$. Altså at telleren når 65 535 før den resettes til null.

Figur 2.17 viser parameterverdiene fra telleren via Q2edit med to forskjellige frekvenser. Her ser man at toppunktene ikke når 65535 som er fordi Q2edit har en samplingsrate som gjør at toppunktene varierer. Det samme skjer med bunnpunktene som ikke når verdien 0.



Figur 2.17: `Encoder pulse counter` data fra Q2edit.

Etter et forsøk ble det oppdaget at etter åtte rotasjoner på akslingen ble telleren resatt. Dette betyr at en runde tilsvarer 8192 pulser på telleren. Oppdelingen blir da som vist i figur 2.18.



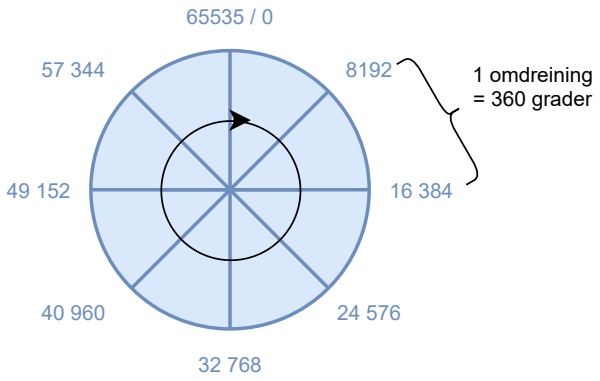
Figur 2.18: Oppdelingen på `encoder pulse counter` tegnet.

Det er også viktig å få med seg at enkoderen husker på posisjonen den har etter strømbrudd. Derfor må man definere et nullpunkt for å beregne riktig antall omdreininger. Hvis den starter målingen på teller verdi lik 16 384, må det bli dette nye nullpunktet fordi enkoderen teller videre fra denne verdien.

Enkoderen i bruk har 2000 puls per omdreining [ppr], som betyr at den har en puls verdi på 8000 ved en omdreining. Dette tilsvarer ikke 8192 pulser som ble registrert i parameteren `encoder pulse counter`. Under testing ble det funnet at en omdreining tilsvarte 8000 pulser, men dette var ikke konsistent. Siden vi ikke har tilgang til A, B ,Z- signalene til enkoderen må vi bruke den ferdiglagde telleren fra frekvensomformeren. Derfor måtte funksjoner programert basere seg på parameteren `encoder pulse counter` og ikke signalene enkoderen faktisk gir ut. Feilmarginen mellom telleren `encoder pulse counter` og virkelig enkoder puls var ikke stor nok til å ha en betydelig påvirkning på prosjektet.

2.4.2 Bergne posisjonen ut ifra `encoder pulse counter`

Parameteren `Encoder pulse counter` gir et ord på 16 bit, som tilsvarer 65 536 pulser. Som sagt tidligere resettes telleren etter åtte omdreininger på akslingen, oppdelingen blir vist i figur 2.19.



Figur 2.19: Illustrerer oppdelingen av `encoder pulse counter` med en omdreining på akselen markert.

Figur 2.19 viser en sirkel som er delt inn i åtte deler. Hver av de delene representerer en omdreining fra motoren og en omdreining tilsvarer $360°$. Ut ifra dette er det mulig å finne puls per grader.

$$Puls\ per\ grader = \frac{8192}{360°} = 22.76 \tag{2.1}$$

Ligning (2.1) ble brukt til å finne posisjonen til vinsjen gitt i grader og er kalkulert fra:

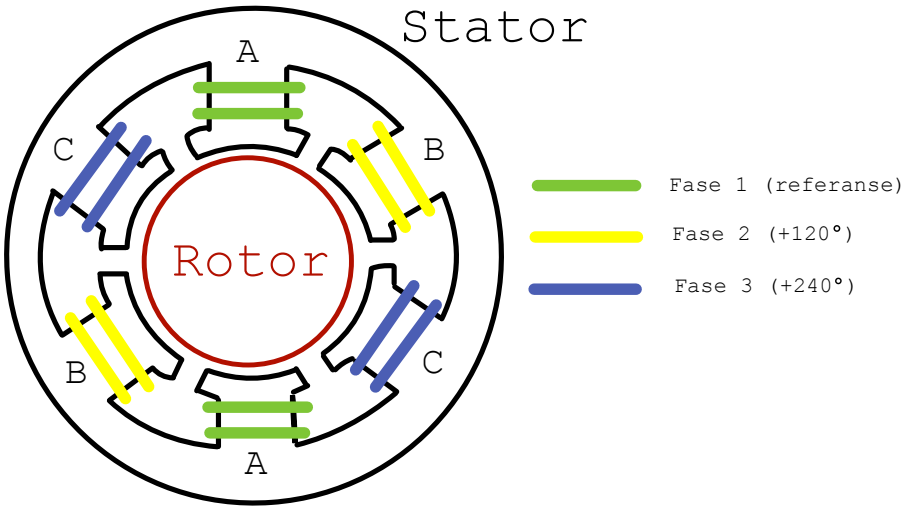
$$Vinsj\ posisjon\ i\ grader = \frac{pulser}{22.76} \tag{2.2}$$

2.5 Asynkron trefasemotor

De forskjellige motorene som ble brukt i dette bachelorprosjektet var alle vekselstrømsmotorer. For å gi en kort introduksjon til denne typen motor, presenterer dette kapitelet egenskaper og prinsipiell virkemåte satt opp mot styring fra en frekvensomformer.

2.5 Asynkron trefasemotor

Asynkron trefase motor er en type vekselstrømsmotor, som består av statorer og rotorer, altså stasjonære og roterende motorkomponenter. Statoren inneholder viklinger av ledninger som er koblet til en vekselstrømskilde, og rotoren er konfigurert slik at den samhandler med statorens spoler via elektromagnetisme, slik som figur 2.20 viser. Statoren produserer et roterende magnetfelt, som de faste polene på rotoren følger etter. Rotoren har vanligvis ingen elektriske kontakter, rotorviklingen er erstattet med staver av kobber og aluminium som er kortsluttet i begge ender [3].



Figur 2.20: Rotor og stator i en asynkron motor.

Rotoren følger statorens roterende magnetfelt, men de oppnår ikke samme hastighet, dette kalles sakking som blir forklart mer senere. Forskjellen på rotorens hastighet og hastigheten på startorens roterende magnetfelt øker ved mer belastning på motoren. Da faller dreiemomentet mot null, slik at det aldri oppnår synkront turtall.

Rotasjonsretning på motoren endres ved å bytte om ledningene på to faser, men grunnet enkoderen koblet på motoren som også har tre faser, gir dette feil strøm- og spennings verdier. Derfor blir dreieretningen bare endret ved hjelp av frekvensomformeren, og ikke ved fysisk bytting av ledningene.

2.5.1 Hastigheten til en asynkron motor

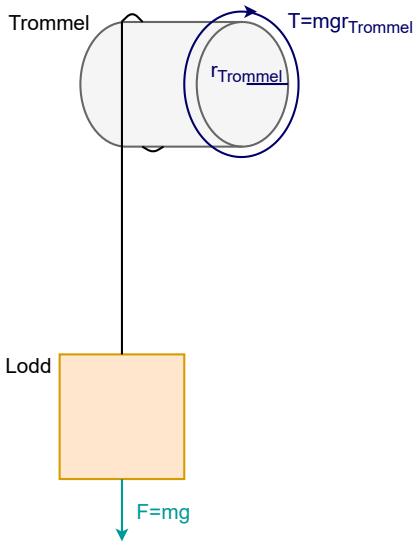
Det synkrone turtallet (ns) motoren har mulighet til å kjøre på, er avhengig av antall poler (p), og frekvensen den kjører på (f), som vist i ligning (2.3). Det synkrone turtallet definerer hvor fort det magnetiske feltet til statoren roterer.

$$ns = \frac{120 \cdot f}{p} \tag{2.3}$$

Fra ligning (2.3) ser man at flere poler reduserer motorhastigheten. Asynkronmotorer har ikke muligheten til å nå den synkrone hastigheten på grunn av sakkeeffekten, som er grunnen til navnet [3].

2.6 Vinsj

Vinsj er en mekanisk enhet som brukes til å trekke inn eller slippe ut tau eller vaier. I systemet vårt er en trommel festet til en elektrisk motor for å løfte en last opp og ned. Drivkreftene bak et vinsj løft er moment og effekt. Figur 2.21 viser et forenklet bilde over kreftene som fungerer på en vinsj under et løft.



Figur 2.21: Krefter som fungerer på en vinsj under et løft.

Betegnelse	Beskrivelse	Benevning
F	Kreftene som virker på et objekt	N
m	Massen til et objekt	kg
g	Gravitasjonen til jorden	m/s^2
T	Moment	Nm
$r_{trommel}$	Radiusen til trommelen	m
d_{vaier}	diameter på vaiaren	mm

Tabell 2.4: Beskrivelse over tegn brukt i vinsjdynamikken.

Tabell 2.4 er en liste over alle variablene som blir brukt til å forklare kreftene som fungerer på en vinsj under et løft. Man tar utgangspunktet i at friksjonen er neglisjerbar.

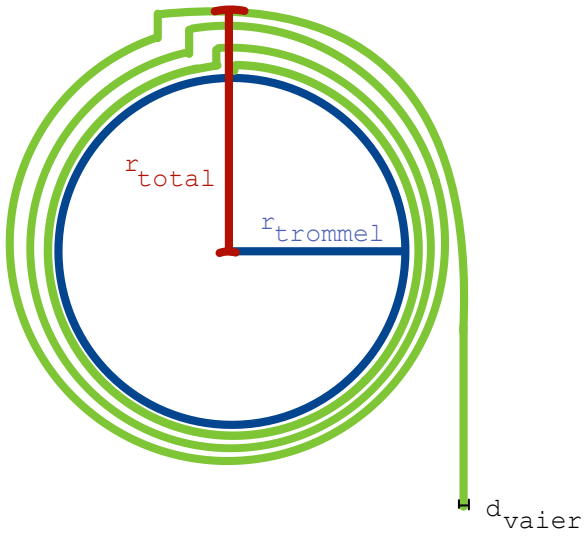
2.6.1 Lengden på innspolt vaier

For å finne lengden på innspolt vaier må ligningen for omkrets til en sirkel bli brukt, vist i ligning (2.4).

$$O = 2 \cdot \pi \cdot r_{total} \tag{2.4}$$

For å finne høyden fra lasten og ned til bakken blir ligning (2.2) og ligning (2.4) tatt i bruk.

Som illustrert i figur 2.22 endres den totale radiusen avhengig av hvor mye vaier som er kveilet opp på trommelen. Den totale radiusen er altså avhengig av diameteren på vaieren.



Figur 2.22: Demonstrerer den økende radius (r_{total}) ved hver omdreining.

$r_{trommel}$ er radiusen til selve trommelen, og n er antall omdreininger trommelen har rotert. Ligningen for å finne den totale radiusen blir som vist i ligning (2.5).

$$r_{total} = r_{trommel} + d_{vaier} \cdot n \tag{2.5}$$

Trommelen som ble brukt i oppgaven er like bred som vaieren, derfor ble ikke bredden til trommelen tatt hensyn til.

2.6.2 Momentet til vinsjen

Under et vinsj løft er det hovedsakelig to moment som er viktige: motormoment og lastmoment. Motormoment må være større enn lastmoment for å forsikre at lasten blir løftet opp. Lastmomentet er som regel avhengig av friksjon, treghet til de bevegelige delene og selve lasten [1]. Som sagt tidligere ble friksjonen ansett som neglisjerbar, og det ble derfor valgte å se bort ifra dette.

For å finne lastmoment benyttes ligning (2.6).

$$T_{last} = m \cdot g \cdot r_{total} \tag{2.6}$$

Ut i fra ligning (2.6) ser man at momentet til lasten er avhengig av den totale radiusen. Det betyr at ved innspoling øker lastmomentet, og ved utspoling minkes lastmomentet.

For å beregne motormoment benyttes ligning (2.7).

$$T_{motor} = 9550 \cdot \frac{Effekt\ [kW]}{hastighet\ [o/min]} \tag{2.7}$$

2.6.3 Vinkelhastigheten og innspolingshastigheten

Motorens hastighet (N) er definert med enhet o/min, men motorhastigheten fra frekvensomformeren (M) er gitt i Hz. Derfor må motorhastigheten bli konvertert fra Hz til o/min. Fra forsøket i delkapittel 2.2.4 er det vist at for dette systemet er 1 Hz lik 30 o/min, og konverteringen skjer som følger:

$$N[o/min] = M[Hz] \cdot 30 \tag{2.8}$$

Derreter blir vinkelhastigheten til vinsjen lik:

$$w[rad/s] = \frac{2\pi}{60} \cdot N[o/min] \tag{2.9}$$

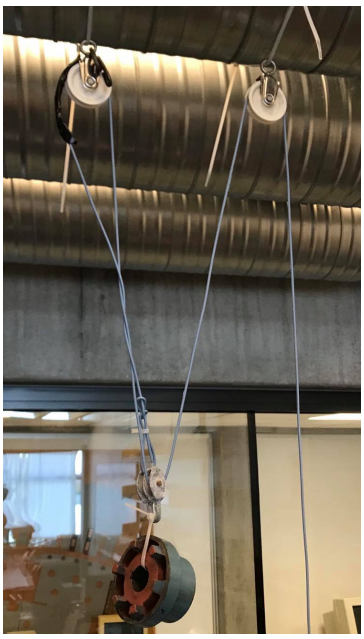
Innspolingshastigheten er basert på vinkelhastigheten, og uttrykket blir som i ligning (2.10) [6]:

$$v[m/s] = r_{total} \cdot w[rad/s] \tag{2.10}$$

Fra ligningen ser man at innspolingshastigheten er avhengig av den totale radiusen. Det betyr at større total radius resulterer i raskere innspolingshastighet.

2.6.4 Trinser

I systemet er det brukt tre trinser, to i taket som er stasjonære, og den tredje er ved lasten som beveger seg med lasten. Figur 2.23 viser bilde av trinsene. Trinsene ble benyttet for å få mer høyde å eksperimentere med.



Figur 2.23: Bilde av trinsene med last heiset helt opp.

Lastens høyde over bakken blir derfor 1/3 av vaieren som er innspolt i trommelen, fordi vaieren blir fordelt på de tre trinsene.

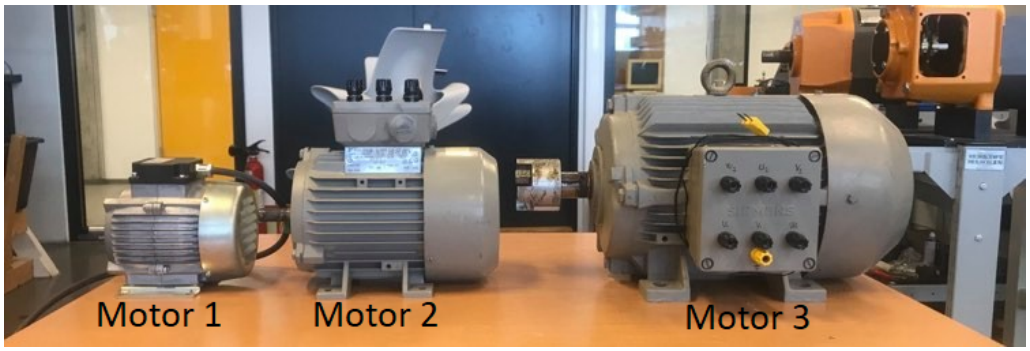
Trinsene gjør også at lastendring ikke har så stor påvirkning på systemet, fordi vekten blir fordelt jevnt utover de tre trinsene. Trinser blir opprinnelig brukt til å redusere mengden kraft som skal til for å løfte en last.

Kapittel 3

Valg av motorer

3.1 Anvendte motorer

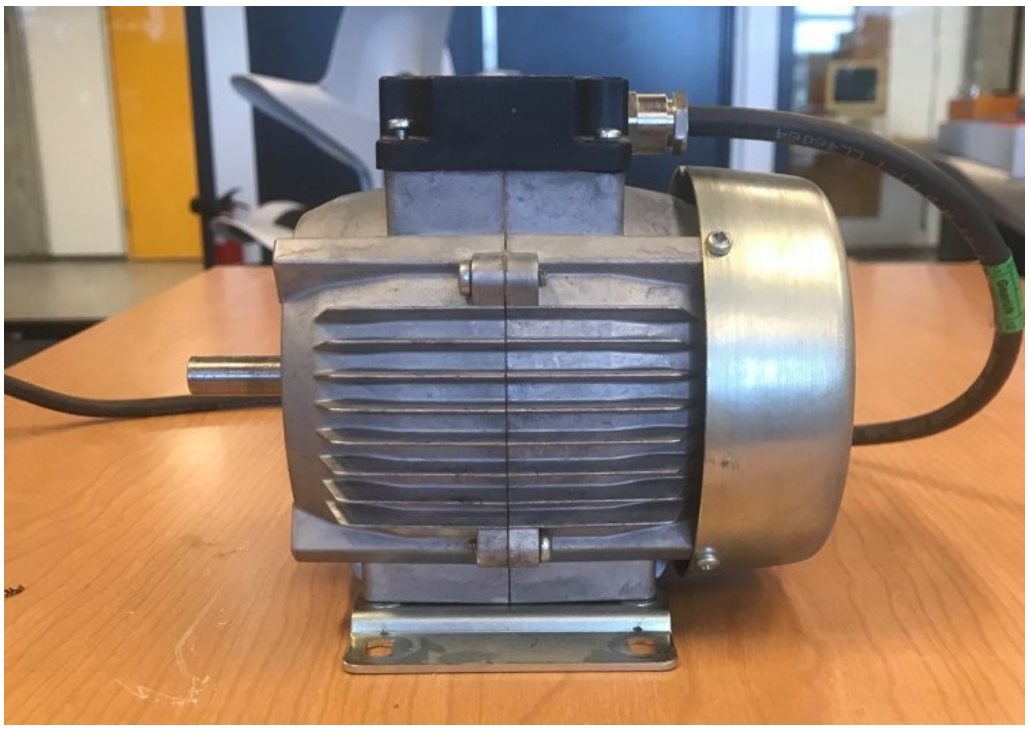
Tre forskjellige motorer som passet til frekvensomformeren og enkoderen ble brukt. De ble testet for å bestemme hvilken av dem egnet seg best for prosjektet. Figur 3.1 viser alle de tre motorene ved siden av hverandre for sammenligning.



Figur 3.1: Alle tre motorene brukt i prosjektet.

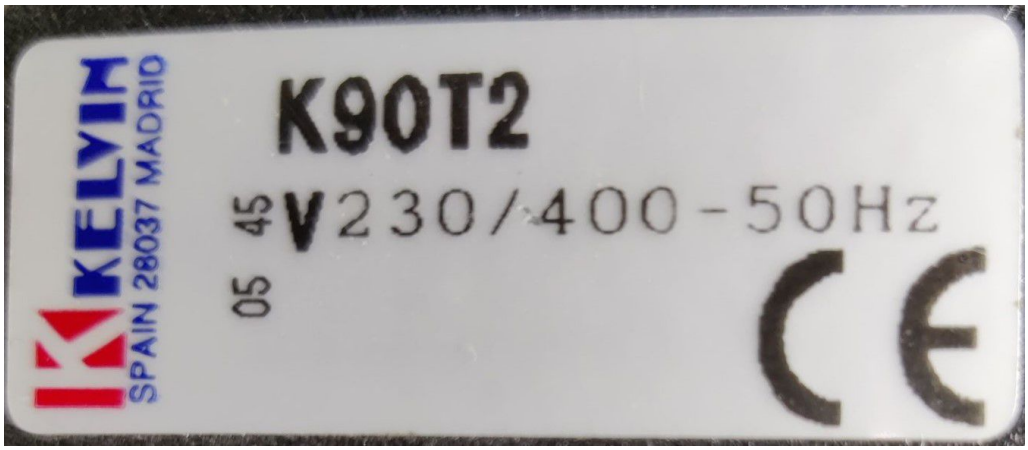
3.1.1 Motor 1 - Kelvin K90T2

Den første motoren, Kelvin K90T2 fikk vi tildelt fra Omron, som var en asynkron trefasemotor. Figur 3.2 viser bilde av den første motoren.



Figur 3.2: Bilde av den første motoren.

Det var lite informasjon om motoren på motormerking sett i figur 3.3. Den eneste nyttige informasjonen fra motormerkingen var spenningen og frekvensen motoren arbeidet på. Vi prøvde å finne mer informasjon om motoren på nettet, men det var lite som hjalp. Motoren holdt ikke lenge, mest sannsynligvis fordi for mye strøm ble gitt til motoren. Det er vanskelig å være helt sikker på at dette var grunnen, siden det var ingen merking på hvor mye strøm motoren tålte. Motoren måtte byttes ut på starten av prosjektet.



Figur 3.3: Motormerkingen til første motoren.

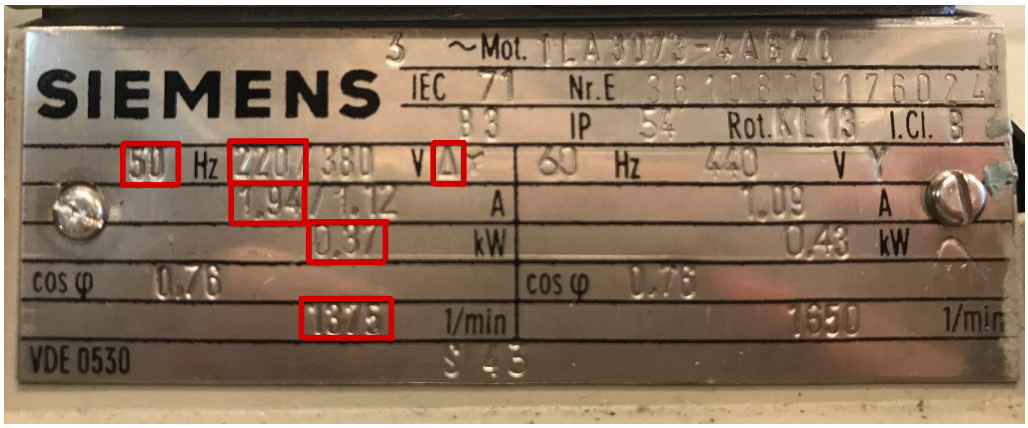
3.1.2 Motor 2 - Siemens VDE 0530 - 0.37 kW

Den andre motoren er en asynkron trefasemotor, Siemens VDE 0530 modell, som ble lånt fra Universitetet i Stavanger. Figur 3.4 viser et bilde av motoren.

3.1 Anvendte motorer

Figur 3.4: Bilde av den andre motoren.

Motoren er en induksjonsmotor, som betyr at den gir høy startstrøm. Den er delta-koblet, som på 50 Hz har en nominell verdi på 220 V, 1.94 A og 0.37 kW, som vist i figur 3.5.



Figur 3.5: Motormerkingen til motor 2 med markeringer på nominelle verdier.

Ligning (2.3) for det synkrone turtallet ble brukt til å regne ut turtallet ved nominelle verdier. Dette er nødvendig i forhold til *autotuning* og valg av motor, som blir utdypet senere. Motoren har en nominell frekvens på 50 Hz og fire poler, så det synkrone omdreiningstallet ble regnet ut som vist i ligning (3.1).

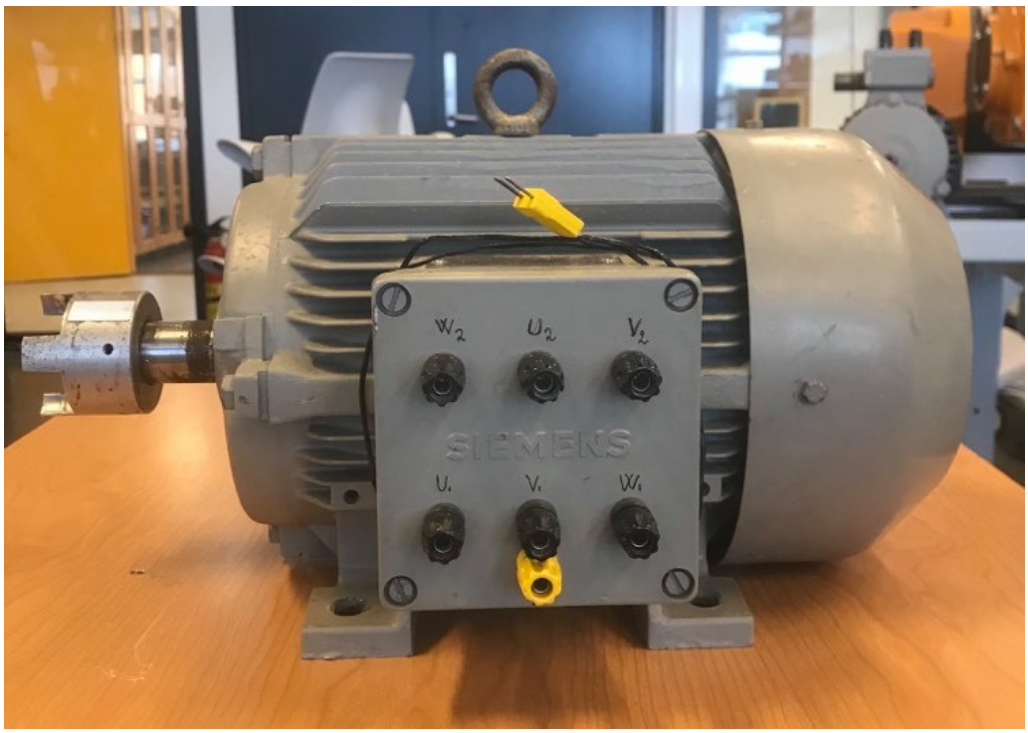
$$ns = \frac{f \cdot 120}{p} = \frac{50 \cdot 120}{4} = 1500\ o/min \tag{3.1}$$

På motorskiltet i figur 3.5 er det nominelle turtallet (n) gitt på 1375 o/min. Ved nominell last er det dette turtallet motoren har. Det nominelle turtallet er alltid lavere enn det synkrone, som er grunnen til at det kalles asynkronmotor. Forskjellen mellom statoren og rotorens rotasjon er sakking (s). Sakking varierer basert på lasten, som betyr at informasjonen på motormerkingen bare gjelder i dette driftspunktet. Sakkingen regnes ut som vist i ligning (3.2).

$$s = \frac{ns - n}{ns} \cdot 100\% = \frac{1500 - 1375}{1500} \cdot 100\% = 8.33\% \tag{3.2}$$

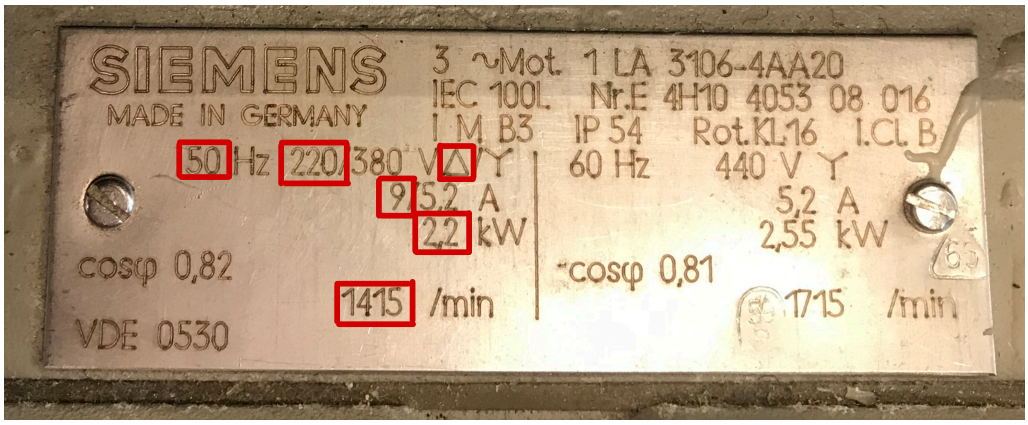
3.1.3 Motor 3 - Siemens VDE 0530 2.2 kW

Den tredje motoren er den samme Siemens VDE 0530 modellen som den andre motoren, men har evne til å levere 2.2 kW. Motoren er i likhet med de andre av induksjonstypen. Figur 3.6 viser et bilde av motoren.



Figur 3.6: Bilde av den tredje motoren.

Den er deltakoblet, som på 50 Hz har en nominell verdi på 220 V, og 9 A. Den har et nominelt omdreiningstall på 1415 o/min, etter motormerkingen vist i figur 3.7.



Figur 3.7: Motormerkingen til motor 3 med markeringer på nominelle verdier.

Denne motoren har også et synkront omdreiningstall på 1500 o/min. Ved å ta hensyn til nominell omdreiningstall på 1415, har motoren sakking som vist i ligning (3.3).

$$s = \frac{ns - n}{ns} \cdot 100\% = \frac{1500 - 1415}{1500} \cdot 100\% = 5.67\% \tag{3.3}$$

3.2 Autotuning

3.2.1 Andre motor - 0.37 kW

Autotuning av motor ble gjort for å forsikre at frekvensomformeren styrte motoren med de riktige motorverdiene. Verdiene fra motormerkingen i figur 3.5 ble skrevet inn i frekvensomformeren som vist i tabell 3.1.

Parametere	Verdier skrevet inn	Verdier etter autotuning
Power [kW]	0.37	0.50
Voltage [V]	220	220
Current [A]	1.94	1.94
Frekvens [Hz]	50	50
Pole count	4	4
RPM [o/min]	1375	1450
Encoder [ppr]	2000	2000
No load voltage [V]	198	198

Tabell 3.1: Tabellen viser verdiene før og etter autotuning, for motor 2.

"No load voltage" som ifølge brukermanualen side 846 [8] er 90% av motor max voltage, som ble regnet ut til å bli 198 V. Encoder ppr blir satt til 2000 ppr, ifølge databladet til enkoderen [7].

Etter *autotuningen* blir antall omdreining per minutt satt til 1450 o/min. Det er ingen andre steder å endre rotasjonshastigheten på, men det kan hende at frekvensomformeren endrer omdreiningshastigheten fordi den prøver å redusere sakkingen til motoren. Ved en omdreiningshastighet på 1450 o/min og synkron hastighet lik 1500 o/min slik som vist i ligning (3.1) blir sakkingen som vist i ligning (3.4):

$$s = \frac{ns - n}{ns} \cdot 100\% = \frac{1500 - 1450}{1500} \cdot 100\% = 3.33\% \tag{3.4}$$

Dermed har motoren nå en sakking på 3.33% istedet for 8.33%. Ved kontakt med Omron oppklarte de at det burde ikke endre mye ved å kjøre på de nye verdiene i stedet for de nominelle verdiene. Fordi det ikke er mulig å endre omdreiningshastigheten annet enn i *autotuning*, må verdiene forbli slik.

3.2.2 Tredje motor - 2.2 kW

Autotuning ble også gjort på den tredje motoren. Ut ifra motormerkingen vist i figur 3.7, ble de nominelle verdiene skrevet inn. Verdiene fått før og etter *autotuning* er vist i tabell 3.2.

Parametere	Verdier skrevet inn	Verdier etter autotuning
Power [kW]	2.2	2.2
Voltage [V]	220	220
Current [A]	9	9
Frekvens [Hz]	50	50
Pole count	4	4
RPM [o/min]	1415	1415
Encoder [ppr]	2000	2000
No load voltage [v]	198	198

Tabell 3.2: Tabellen viser verdiene før og etter *autotuning*, for motor 3.

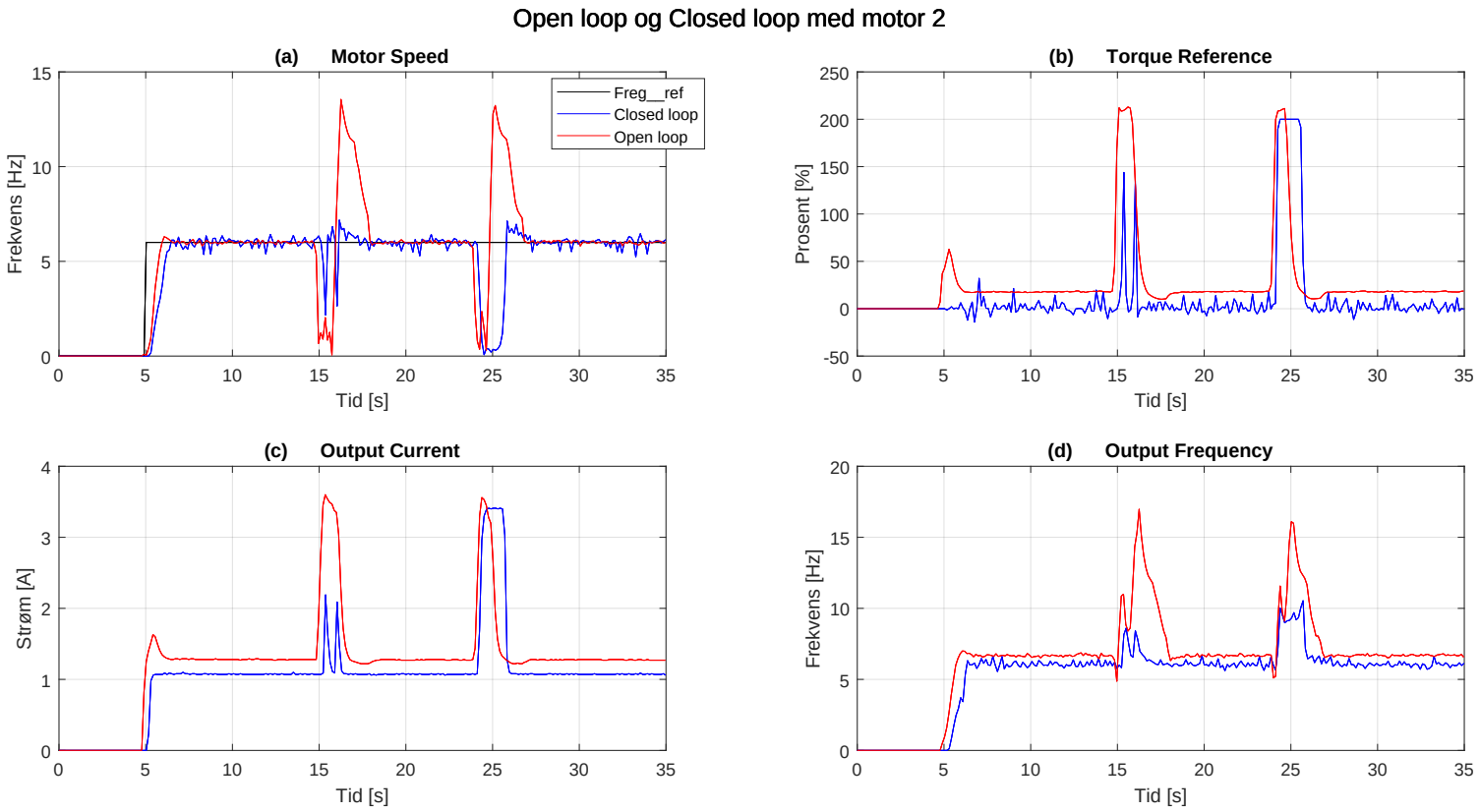
Her er det tydelig at det ikke er noen endring etter *autotuning*. Omdreininger per minutt endret seg ikke, i motsetning til det den gjorde med motor 2. Dette kan være fordi sakkingen ikke er så stor, og omformeren godtar innlagte verdier. I vedlegg C er det en figur som viser verdiene fått ved å kjøre *autotuning* via Q2A-edit programmet.

3.3 Testing av motor 2 og motor 3

3.3.1 *Open loop* og *closed loop* kompenseringsegenskaper

Testen ble utført for å se på forskjellen på responsen mellom *open* og *closed loop*. Motoren ble startet etter 5 s, og akselen ble belastet, ved å holde hånden over akselen i tidsperiodene $15 < t < 16$ s og $24 < t < 25$ s. Motor 2 ble kjørt med 6 Hz som var den høyeste frekvensen som var mulig å belaste akslingen. Motor 3 ble kjørt med 1 Hz, selv dette var vanskelig å belaste akslingen.

3.3 Testing av motor 2 og motor 3

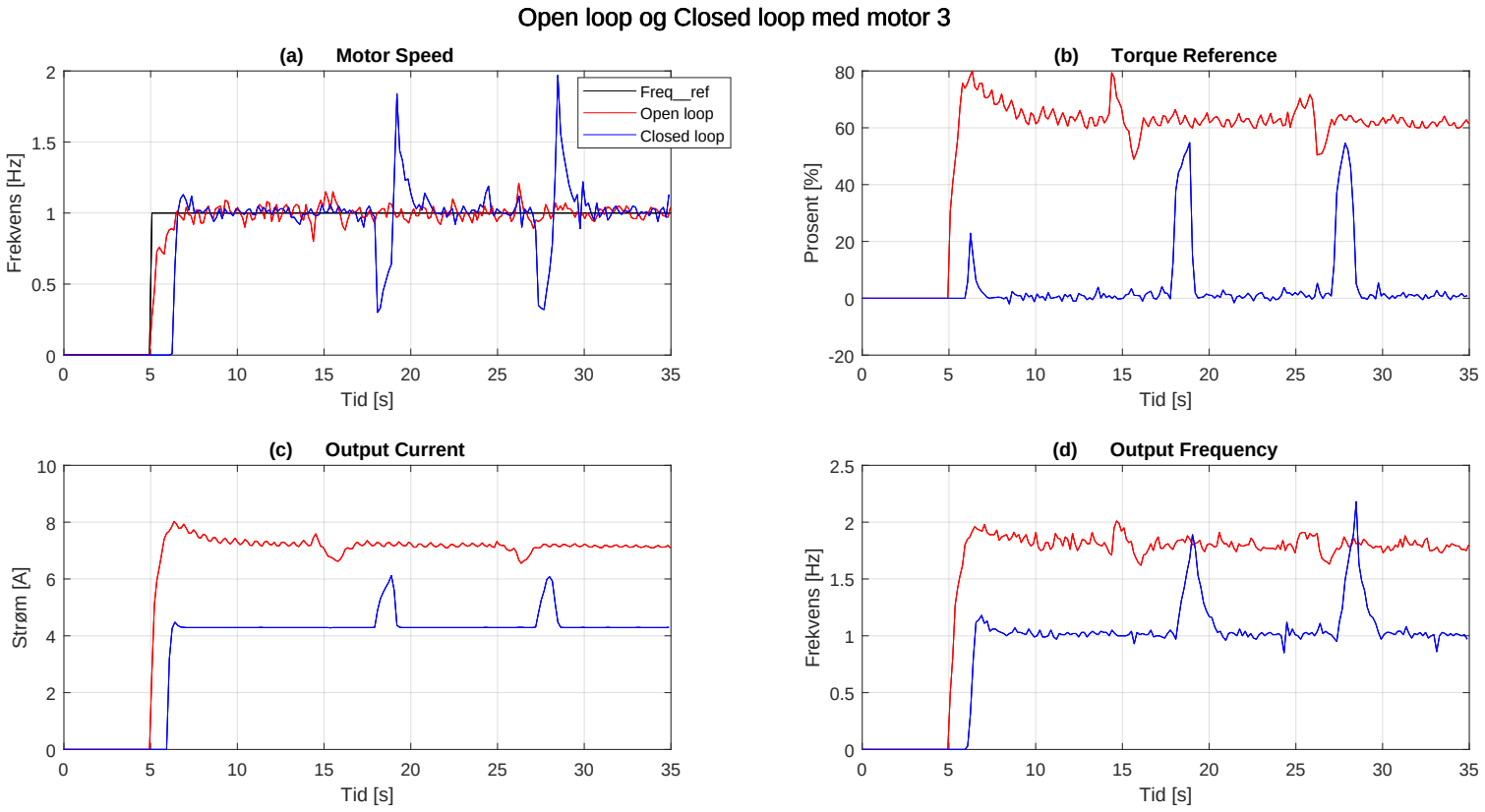


Figur 3.8: Open loop og closed loop kompenseringsegenskaper med motor 2, 6 Hz.

Figur 3.8 viser at forskjellen mellom *open* og *closed loop* ikke er så stor med motor 2. Det er tydelig at de responderer relativt likt, men man la merke til forskjellen på hvordan de to styremåtene reagerte på når akselen ble belastet. Det var vanskeligere å holde igjen ved *open loop* enn ved *closed loop*. Dette er mulig å se ved at samme motorhastighet (`Motor speed`) fall til 1 Hz i tidsperioden $24 < t < 25$ s, produserte *open loop* høyere `Output frequency` i forhold til *closed loop*. *Open loop* har i tidsperioden $24 < t < 25$s en `Output frequency` på 16 Hz, sammenlignet med *closed loop* med en `Output frequency` på 10 Hz. På grunn av den høye `Output frequency` til *open loop* i tidsperioden $24 < t < 25$ s gir det et oversving på `Motor speed` til 16 Hz

når belastningen stoppet. Oversvingen til *closed loop* i tidsperioden $24 < t < 25$ s er bare til 7 Hz.

I figur 3.9 er resultatet ved samme forsøk med motor 3. Her ble akslingen belastet ved tidsperiodene $17 < t < 18$ s og $26 < t < 27$ s.



Figur 3.9: Open loop og closed loop kompenseringsegenskaper med motor 3, 1 Hz.

Fra figur 3.9 ser man med engang at det er en forskjell på styremåtene. Som sagt før er `Output frequency` det frekvensomformeren prøver å gi ut for å oppnå riktig motorhastighet. I *open loop* gir den en `output frequency` lik 2 Hz for å oppnå en `motor speed` på 1 Hz, men man så at den roterte raskere enn det. *Open loop* gjør en beregning på hastigheten ut i fra strømtrekket, og ut ifra grafen ser vi at motoren tok mer strøm ved *open loop* enn ved *closed loop*. Dette korresponderer med at faktiske hastigheten på akselen ikke var lik 1 Hz.

`Tourque reference` (delfigur b) er også mye høyere i *open loop*, og dette er fordi torque ref er basert på strømtrekket. Dette er fordi *closed loop* er mye mer nøyaktig ved lave hastigheter, grunnet tilbakekoblingen om posisjon, og derfor hastighet, fra enkoderen.

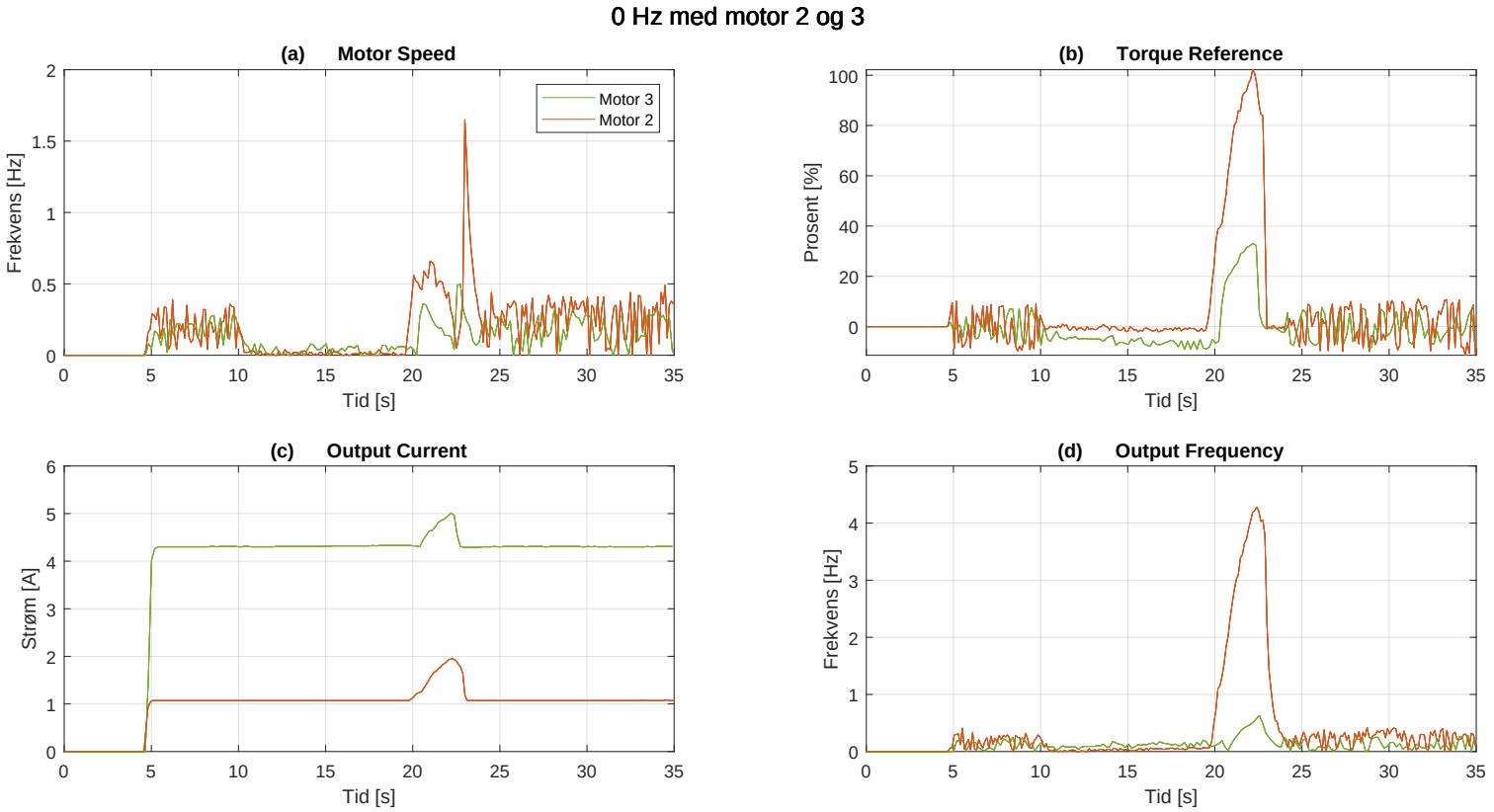
Fra figur 3.9 er det også mulig å se at det var lite respons ved tilføring av motstand i *open loop*. Som sagt var det større motstand i *open loop* enn i *closed loop*, så det var veldig vanskelig å holde for hånd på denne sterke motoren. Motoren har evne til å gi ut mye mer kraft enn den forrige motoren, og resultatet ser annerledes ut enn i figur 3.8, grunnet vanskeligheter med å holde igjen.

Ut ifra de to figurene konkluderte vi at *closed loop* fungerer fint uavhengig av motor, og spesielt ved lave frekvenser. Dette er fordi i *closed loop* har den tilbakemelding fra enkoderen, men det har den ikke i *open loop*. I *open loop* må frekvensomformeren gjøre beregninger ut i fra hva den sender ut til motoren og de gitte verdiene ut ifra *autotuning*.

3.3.2 Closed loop ved 0 Hz kompenseringsegenskaper

Tidligere er det nevnt at det er mulig å kjøre motoren på 0 Hz i *closed loop*. Testen ble utført for å verifisere dette og se hvordan den reagerer når akslingen blir belastet. Motoren ble startet etter 5 s, i tidsperioden $10 < t < 20$ s belastes akselen ved å holde hånden over akslingen. Deretter i tidsperioden $20 < t < 24$ s blir akselen vridd rundt, og til slutt blir akselen sluppet.

3.3 Testing av motor 2 og motor 3



Figur 3.10: Resultat av closed loop test ved 0 Hz, og to forskjellige motorer.

Grunnen til at toppene på motor 2 er høyere enn motor 3, er fordi motor 2 var svakere og ble vridd lengre. Jo lengre akselen ble vridd, jo mer motstand ble følt på begge motorene.

I tidsperioden $5 < t < 10$ s var ikke akselen belastet, og i delfigur a (`Motor Speed`), b (`Torque reference`) og d (`Output frequency`) ser man mye støy. Dette er fordi motoren forventer motstand. Arbeid på 0 Hz er ment for å holde en last i ro, for eksempel ved kranløft.

I tidsperioden $10 < t < 20$ s ble en hånd lagt over akselen, dette ga systemet en liten belastning. Her er det mindre støy på de tre grafene `Motor Speed`, `Torque reference` og `Output frequency`. Siden systemet jobber mot en belastning gir det mindre støy.

I tidsperioden $20 < t < 24$ s ble akslingen vridd rundt. Systemet arbeider for å opprettholde 0 Hz og kompenserer for rotasjonen. Det betyr at hvis en last blir holdt i ro med 0 Hz under vinsj løft, kompenserer systemet for lastendringen. Ved lastendring reguleres systemet tilbake til 0 Hz og lasten står i ro når den oppnår 0 Hz uavhengig av startposisjon.

3.4 Motoren anvendt til vinsjsystemet

Motor 1 ble ødelagt med en gang i prosjektet og var derfor ikke brukbar. Denne motoren var en del av demomodellen fra Omron, og var den motoren som var ment for prosjektet.

Motor 2 Siemens VDE 0530 - 0.37 kW ble mest brukt fordi den var best egnet til frekvensomformeren og lett å jobbe med. Selv om *autotuningen* ga andre verdier enn de nominelle, burde dette ikke endre så mye på prosjektet fordi vinsjsystemet ble aldri operert på de nominelle verdiene. Motor 2 er følsom ved lave frekvenser som gir god evne til å teste systemet og se tydelige responser.

Motor 3 Siemens VDE 0530 - 2.2 kW var på den øvre grensa av hva frekvensomformeren har evne til å levere. Motoren var veldig kraftig og det var vanskelig å se respons ved belastning. Som nevnt tidligere vil bruken av trinser redusere lasten systemet opplever med 1/3, og derfor vil det bli enda vanskeligere å se respons til systemet ved belastning.

Motor 2 er derfor motoren brukt videre i prosjektet til vinsjsystemet.

Kapittel 4

Teach funksjonen

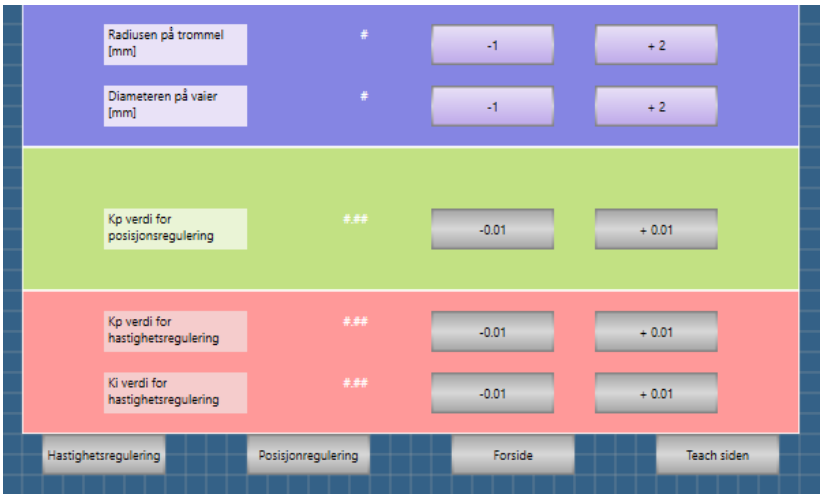
For å lære systemet hvor høyt over bakken en last har blitt heist opp, ble en *teach*-funksjon implementert. Dette gjør at systemet vet når innspoling eller utspoling stoppes når maksimum eller minimum høyde er nådd. Funksjonen ble også brukt for å vise foreløpig høyde over bakken på HMI-skjermen til PLSen. Det er viktig å notere at teach funksjonen er definert basert på `Encoder Pulse Counter` verdien, og tar ikke hensyn til feilen mellom virkelig enkoder puls og tellerverdien som forklart i delkapittel 2.4.1.

4.1 Koden for teach funksjonen

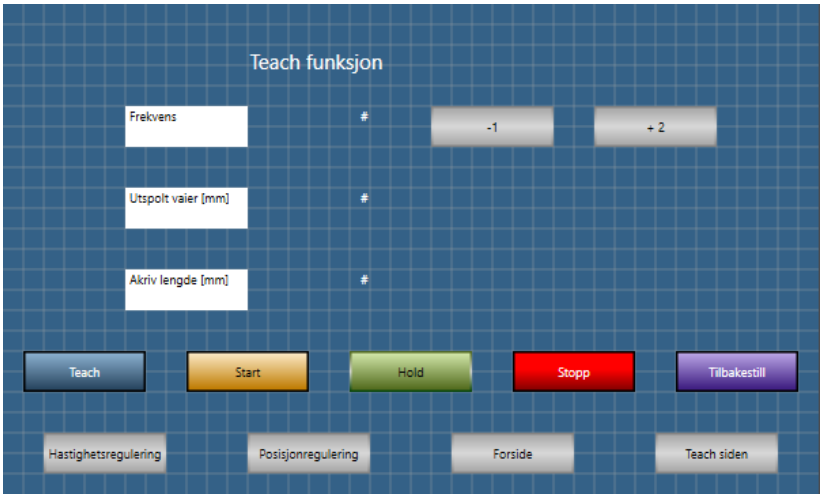
Funksjonen fungerer på følgende måte:

- Vaieren starter helt utspolt, altså at lasten står på bakken.
- Brukeren skriver inn radiusen til trommelen, og diameteren på vaieren brukt på forsiden vist i figur 4.1.
- Brukeren trykker på teach-knappen på skjermen i figur 4.2.
- Brukeren får vinsjen til å dra inn vaieren med å trykke startknappen.
- Når lasten har nådd toppen trykker brukeren på hold-knappen for at lasten står i ro.

- PLSen bruker `Encoder Pulse Counter` informasjonen, trommel og vaier dimensjonene til å regne ut antall runder trommelen roterte og høyden til lasten.
- Brukeren trykker på teach-knappen igjen for å stoppe funksjonen.
- PLSen vet nå maks høyde over bakken.



Figur 4.1: Skjermen til forsiden, hvor radius til trommelen og diameter til vaieren blir skrevet inn, i tillegg til reguleringsparametere.



Figur 4.2: Skjermen til teach funksjonen.

4.1 Koden for teach funksjonen

Verdiene fra funksjonen blir lagret helt frem til brukeren trykker på teach-knappen. Under er koden for funksjonen forklart steg for steg.

Figur 4.3 linje 2 setter `maks_hoyde` lik 0. Hver gang brukeren trykker på teach knappen resettes `maks_hoyde` slik at brukeren har muligheten til å lære systemet en ny maks høyde. Linje 3 definerer `ny_tellerverdi_m` lik pulsene gitt fra `Encoder Pulse Counter` (`enkoder_tellerverdi`). `Encoder Pulse Counter` starter ikke telling av pulser på 0, men starter tellingen på en vilkårlig verdi. Derfor må man definere et startpunkt (`start_tellerverdi`) på den verdien `Encoder Pulse Counter` har når man starter teach funksjonen, som blir gjort i linje 5 og 6.

```
1  IF teach=TRUE THEN
2      maks_hoyde:=0;
3      ny_tellerverdi_m := enkoder_tellerverdi;
4
5      IF start_tellerverdi = 0 THEN
6          start_tellerverdi := enkoder_tellerverdi;
7      END_IF;
```

Figur 4.3: Starten på teach funksjonen.

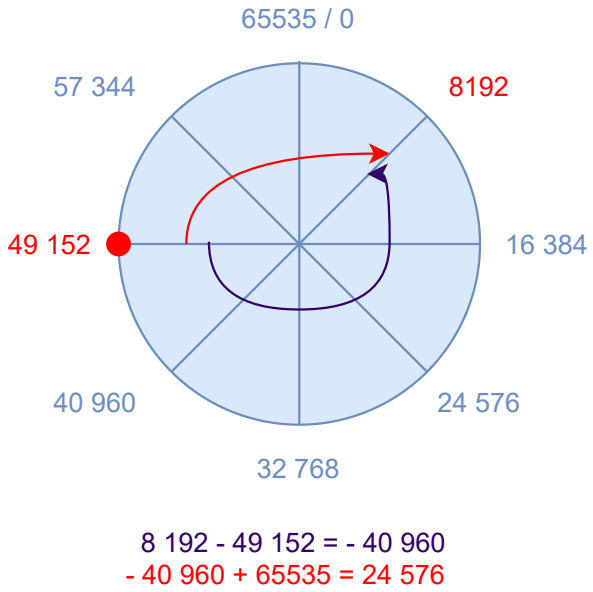
Figur 4.4 viser koden for å definere differansen mellom nåværende tellerverdi (`ny_tellerverdi_m`) og forrige tellerverdi (`gammel_tellerverdi_m`). Differansen (`diff_tellerverdi`) mellom ny og gammel tellerverdi indikerer hvor langt enkoderen har rotert.

```
9   IF ny_tellerverdi_m-gammel_tellerverdi_m < -32768 THEN
10      diff_tellerverdi := ny_tellerverdi_m - gammel_tellerverdi_m +65536;
11  ELSIF ny_tellerverdi_m=start_tellerverdi THEN
12      IF omdreininger=0 THEN
13          diff_tellerverdi:=0;
14          sum_tellerverdi:=0;
15      ELSE
16          diff_tellerverdi:=ny_tellerverdi_m - gammel_tellerverdi_m;
17      END_IF;
18  ELSIF ny_tellerverdi_m - gammel_tellerverdi_m>32768 THEN
19      diff_tellerverdi := ny_tellerverdi_m - gammel_tellerverdi_m-65536;
20  ELSE
21      diff_tellerverdi := ny_tellerverdi_m - gammel_tellerverdi_m;
22  END_IF;
```

Figur 4.4: Definerer steget mellom ny puls og gammel puls.

Linje 9 og 10 i figur 4.4 definerer at hvis differansen mellom `ny_tellerverdi_m` og `gammel_tellerverdi_m` er mindre en -32768, adderes forskjellen med 65536. For

eksempel hvis `gammel_tellerverdi_m` var 49152 og `ny_tellerverdi_m` er 8192, blir differansen -40960. Dette stemmer ikke fordi motoren roterer fremover. Funksjonen "vet" når motoren beveger seg fremover og bakover basert på `q2a_forward` og `q2a_reverse` parameterene som forklart i delkapittel 2.1.1. For å få riktig differanse adderer man med 65535, dermed blir forskjellen lik 24576. Figur 4.5 demonstrerer dette konseptet.



Figur 4.5: Illusterer linje 9 og 10 i figur 4.4. Røde prikken indikerer gammel tellerverdi, lilla pil og tekst viser feil differanse (baklengs), mens rød pil og tekst illustrerer riktig differanse (fremover).

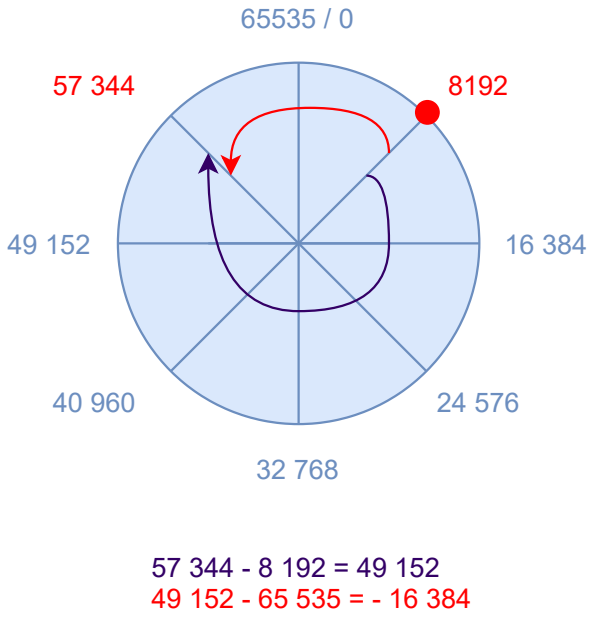
Linje 11 til 14 definerer at dersom `start_tellerverdi` er lik `ny_tellerverdi_m` og det ikke har gått noen runder er differansen lik 0. Siden det er startpunktet så er også summen av antall rotasjoner lik 0.

Linje 16 definerer at når det har gått en runde, som betyr at `ny_tellerverdi_m` har nådd startpunktet, så er differansen lik `ny_tellerverdi_m` minus `gammel_tellerverdi_m`. Dette indikerer at summen av antall rotasjoner ikke er lik 0 lenger, og funksjonen teller antall runder oppover når `ny_tellerverdi_m` har nådd startpunktet.

Linje 18 og 19 definerer at dersom differansen mellom `ny_tellerverdi_m` og `gammel_tellerverdi_m` er mer enn 32768, blir differansen trekket fra med 65536, fordi enkoderen roterer nå

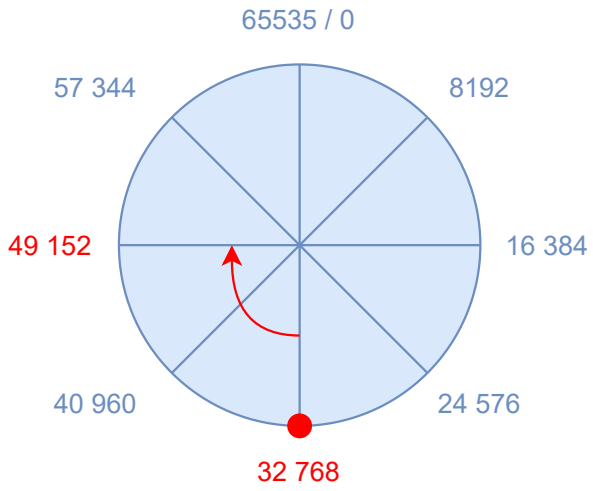
bakover. Funksjonen "vet" når motoren beveger seg fremover og bakover basert på `q2a_forward` og `q2a_reverse` parameterene som forklart i delkapittel 2.1.1.

For eksempel hvis `gammel_tellerverdi_m` var 8192 og `ny_tellerverdi_m` er 57344, blir differansen lik 49152. Dette stemmer dersom enkoderen hadde gått framover, men det gjør den ikke her. For å få riktig steg må man ta differansen minus 65535. Dermed blir forskjellen lik -16384. Dette konseptet er illustrert i figur 4.6.



Figur 4.6: Illusterer linje 18 og 19 i figur 4.4. Røde prikken indikerer gammel tellerverdi, lilla pil og tekst viser feil steg (forover), mens rød pil og tekst illustrerer riktig differanse (bakover).

Linje 21 angir at dersom ingen av kravene i if-løkken ovenfor er oppfylt, er differansen lik `ny_tellerverdi_m` minus `gammel_tellerverdi_m`. Et eksempel er vist i figur 4.7.



Figur 4.7: Illusterer linje 21 i figur 4.4. Røde prikken indikerer gammel tellerverdi, rød pil og tekst illustrerer riktig differanse.

Figur 4.8 linje 24 viser at `sum_tellerverdi` summerer tellerverdiene som var i den tiden teach ble aktivert. I linje 25 blir omdreininger beregnet ved å dele `sum_tellerverdi` på 8192 (en runde), som forklart i delkapittel 2.4.2. Som nevnt tidligere øker radiusen ved hver omdreining, og ligning (2.5) ble brukt til å finne den økende radiusen (`total_radius`) i linje 26. Linje 27 regner ut vinkelen til enkoderen ved bruk av ligning (2.2), slik at man vet vinkelposisjonen til trommelen.

```
24  sum_tellerverdi:=sum_tellerverdi+diff_tellerverdi;
25  omdreininger := LINT_TO_INT((sum_tellerverdi) / 8192);
26  total_radius := radius_trommel + diameter_tau * omdreininger;
27  vinkel:= LINT_TO_LREAL(sum_tellerverdi - 8192*omdreininger) /22.76;
```

Figur 4.8: Beregner summen av tellerverdiene, antall omdreininger, den totale radiusen og vinkel.

I første gjennomkjøring av koden er total radiusen lik trommel radius som blir spesifisert i linje 29-31 i figur 4.9. Som sagt øker radiusen etter hver omdreining, derfor er det nødvendig å ta vare på gammel radius. Linje 33 definerer at hvis `total_radius` er større enn `gammel_total_radius` blir vaieren spolt inn. Linje 34 definerer vaier-

lengen (`total_vaier_lengde`) ved å bruke ligning (2.4) for omkrets til en sirkel og konseptet i figur 2.22. Linje 35-37 går ut på samme prinsipp som linje 33-34, bare at denne er for utspoling.

I linje 39 blir høyden over bakken (`hoyde`) beregnet og må bli delt på tre på grunn av de tre trinsene i systemet forklart i delkapittel 2.6.4. Dersom det ikke er noen trinser så trenger man ikke å dele på tre, da blir forholdet mellom innspolt vaier og høyden over bakken 1:1. Dette er det som ofte blir kalt for den matematiske modellen av systemet. Høyden over bakken blir beregnet på samme måte for alle funksjonene i systemet.

```
29  IF gammel_total_radius = 0 THEN
30      gammel_total_radius := total_radius;
31  END_IF;
32
33  IF total_radius > gammel_total_radius THEN
34      total_vaier_lengde := total_vaier_lengde + gammel_total_radius * 2 * 3.14;
35  ELSIF total_radius < gammel_total_radius THEN
36      total_vaier_lengde := total_vaier_lengde - total_radius * 2 * 3.14;
37  END_IF;
38
39  hoyde := (total_vaier_lengde + (vinkel/360)*total_radius * 2 * 3.14 )/3;
```

Figur 4.9: Beregner vaierlengden og høyden over bakken ut ifra vinkel og radius.

Mot slutten av koden blir de gamle verdiene lagret for å bruke dem igjen, vist i figur 4.10. Linje 44 definerer at når teach blir deaktivert blir høyden lasten har over bakken lagret som en maksimumshøyde. Maksimumshøyden ble brukt som en begrensning i posisjonsregulering.

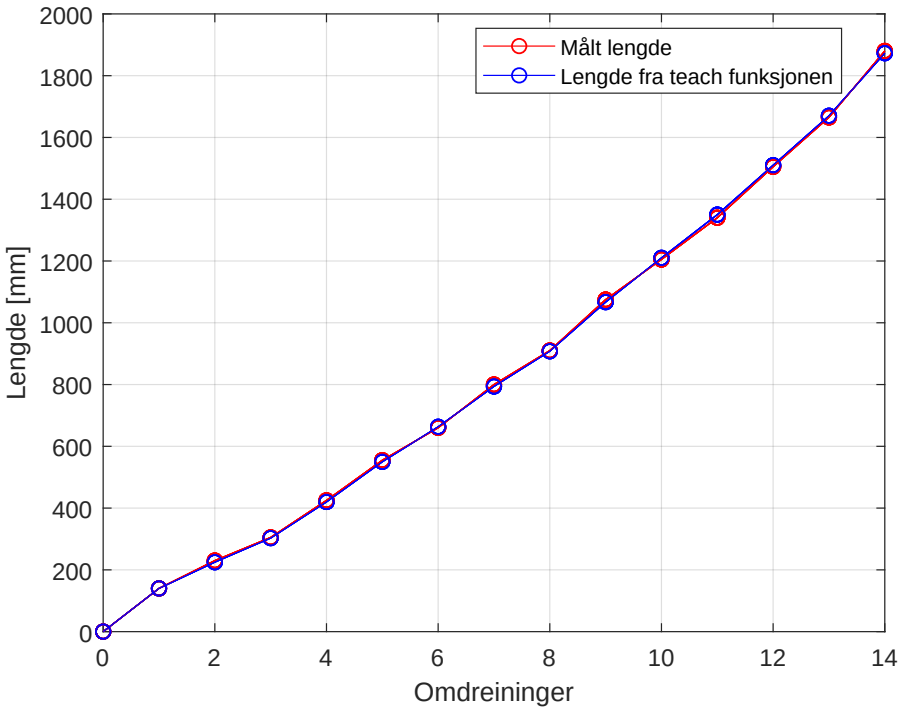
Linje 45-47 definerer at dersom teach funksjonen ikke blir benyttet er maksimumshøyden en egendefinert verdi på 1500 mm. Vårt system hadde 1500 mm å regulere med, derfor ble denne verdien satt.

```
41      gammel_tellerverdi_m := ny_tellerverdi_m;
42      gammel_total_radius := total_radius;
43  ELSE
44      maks_hoyde:=hoyde;
45      IF maks_hoyde<=0 THEN
46          maks_hoyde:=1500;
47      END_IF;
48  END_IF;
```

Figur 4.10: Lagrer gamle verdier og setter maksimumshøyde.

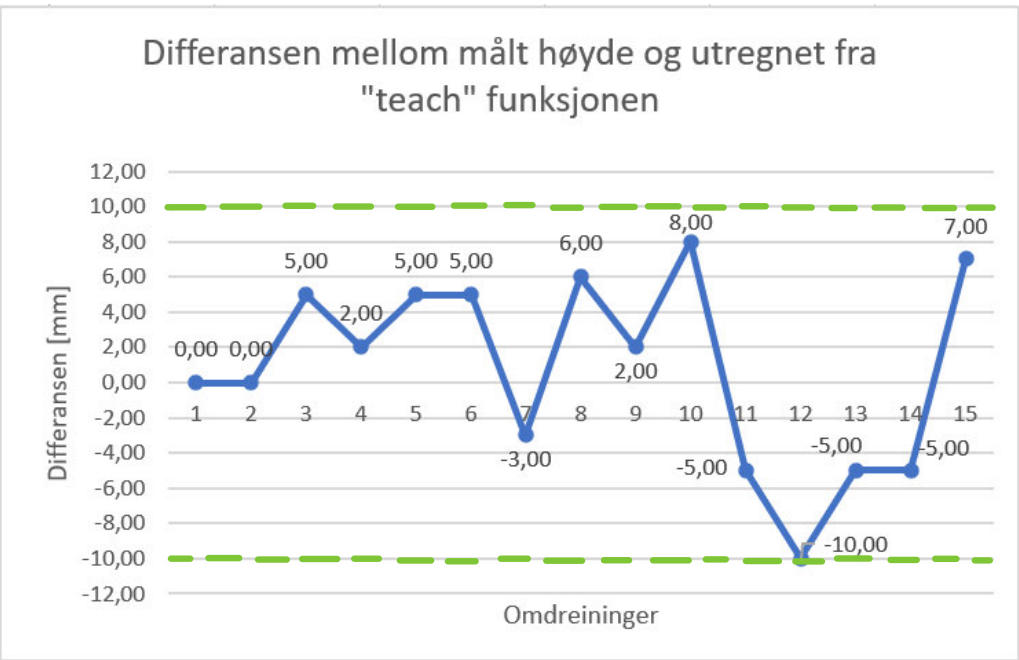
4.2 Verifisering av teach funksjonen

For å teste *teach* funksjonen beskrevet i delkapittel 4.1 ble flere forsøk utført. Forsøkene gikk ut på at en last på 1.5 kg ble heist opp og stoppet hvor høyden ble målt med tommestokk og avlest fra *teach* funksjonen. Grafene i figur 4.11 viser målte verdier og det *teach* funksjonen beregnet.



Figur 4.11: Grafene viser målt lengde og det *teach* funksjonen beregnet.

Fra figur 4.11 er det tydelig at funksjonen fungerer som forventet. Det var liten forskjell på virkelig og utregnet høyde. Dette eksperimentet ble utført flere ganger og figuren over er representativ for alle forsøkene. Feilen i alle forsøkene var nesten aldri større en 10 mm. Et til forsøk av *teach* funksjonen med 3 kg last er vist i vedlegg D. Figur 4.12 viser forskjellen mellom målt og beregnet høyde.



Figur 4.12: Grafene viser differansen mellom høyden målt og utregnet fra *teach* funksjonen.

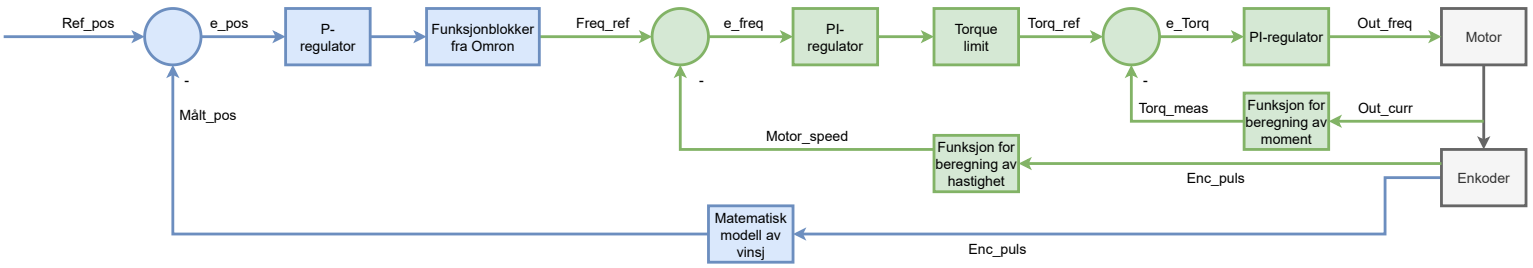
Figur 4.12 viser forskjellen mellom målt høyde og beregnet via *teach* funksjonen. De grønne stiplede linjene er tegnet på +-10 mm for å demonstrere at *teach* funksjonen ikke har større feil enn det.

En video av at lengden blir lært av *teach* funksjonen, og at oppspoling stopper ved maks lengde er her: `https://www.youtube.com/watch?v=hGuJlLimJDY`

Kapittel 5

Posisjonsregulering av last

For å forsikre at ønsket høyde over bakken blir oppnådd og vedlikeholdt ved lastendring ble posisjonsregulering implementert. Posisjonsregulering er basert på den matematiske modellen av systemet, som forklart i kapittel 4. Figur 5.1 viser blokkskjema av posisjonsreguleringen.



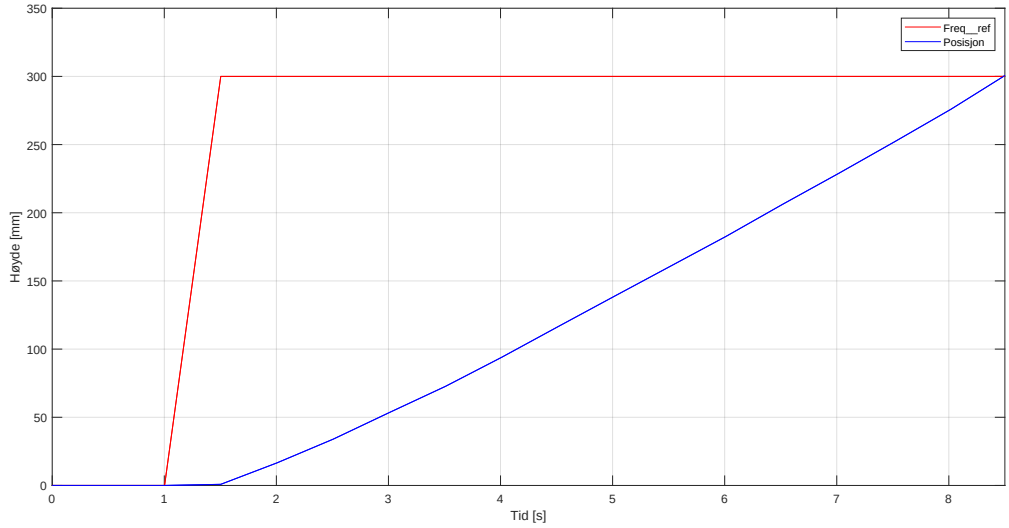
Figur 5.1: Blokkskjema for posisjonsregulering, hvor blå del er inne i PLS-en, grønn er internt i frekvensomformeren, og grå representerer de fysiske komponentene.

Enkoder puslene `Enc_puls` blir anvendt i den matematiske modellen av vinsjen for å beregne høyden over bakken (`Målt_pos`). *Referanseposisjonen* `Ref_pos` er referansen i reguleringsløyfa, og tilsvarer ønsket posisjon oppgitt i [mm]. Reguleringsavviket e_{pos} er differansen mellom referansen og målt posisjon. Differansen går inn til en P-regulator som via en funksjonsblokk fra Omron sender signalet `Freq_ref` til frekvensomformeren. Videre blir signalet beregnet i reguleringsløyfa for closed loop vektorkontroll (forklart i kapittel 2.2.3) og beregner pådraget `Out_freq` i [Hz].

5.1 P-regulator

En P-regulator overvåker hele tiden avviket e, fra ønsket posisjon og posisjonen den egentlig er på, og justerer for å redusere avviket. P-leddet påfører et pådrag for å jevne ut posisjonsavviket med justeringsfaktor K_p.

Systemet har allerede innebygde PI-regulatorer som vist i closed loop blokkskjemaet i figur 2.12. Ved et sprang på referansen `Freq_ref` vil posisjonen øke lineært som indikerer en integrerende prosess, som vist i figur 5.2.



Figur 5.2: Ved et sprang på `Freq_ref` vil posisjonen øke lineært.

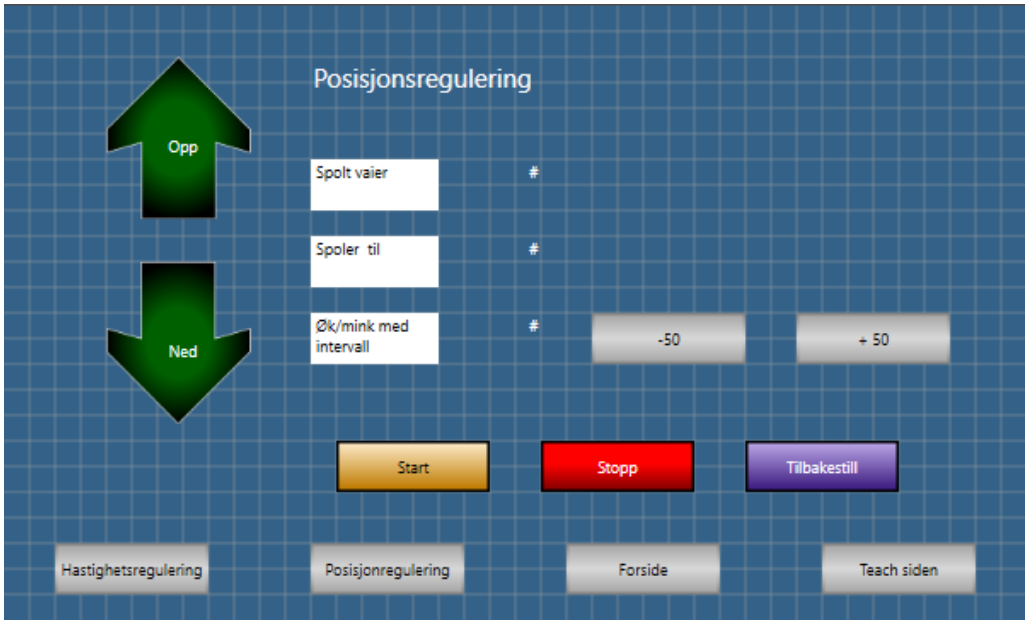
Derfor ble det valgt å bare ha en P-regulator i posisjonsregulering vist i ligning (5.1).

$$u(t) = K_p \cdot e \tag{5.1}$$

5.2 Kode for posisjonsregulator

Koden for posisjonsregulatoren er forklart i denne delen av kapittelet. For å styre posisjonsregulatoren ble to HMI-skjermer utviklet. Brukeren må skrive inn ønsket K_p verdi for systemet på forsiden vist i figur 4.1. Den gule start-knappen på skjermen for posisjonsreguleringen vist i figur 5.3 aktiverer regulatoren. Den røde stopp-knappen fungerer som en nødstopp som stopper motoren. Den lilla tilbakestilling-knappen setter målt høyde lik null, og blir brukt til å sette bakken som nullpunkt.

På skjermen vist i figur 5.3 skriver brukeren inn ønsket høydeendring. Når brukeren trykker på den grønne pilopp-knappen eller pilned-knappen utfører posisjonsregulatoren endringene nødvendig for å nå ønsket høyde. Når ønsket høyde og nåværende høyde er like, holder funksjonen lasten i ro ved 0 Hz. Hvis lasten når maks høyde, altså den verdien `maks_hoyde` gitt fra teach funksjonen, blir lasten holdt i ro ved 0 Hz.



Figur 5.3: Skjermen til posisjonsregulering funksjonen.

Ønsket høydeendring er definert av variabelen `pos_endring`. Linje 1 og 2 i figur 5.4 sjekker om `pos_endring` er mindre eller lik 0, og setter i så fall `pos_endring` til 100. Dette er for å unngå negative tall fordi høydeendring skal alltid bli oppgitt som et positivt tall.

```
1 IF pos_endring<=0 THEN
2     pos_endring:=100;
3 END_IF;
```

Figur 5.4: Kode for å sette posisjonsendring.

Figur 5.5 viser koden for pilopp-knappen og pilned-knappen på skjermen vist i figur 5.3. I linje 5-8 blir `pos_endring` brukt til å øke ønsket høyde (`onsket_hoyde`) hvis pilopp-knappen blir trykket på. I linje 10-13 blir `pos_endring` brukt til å redusere ønsket høyde (`onsket_hoyde`) hvis pilned-knappen blir trykket på. Linje 6 og 11 resetter pilopp og pilned knappene, slik at brukeren ikke trenger å gjøre det selv.

```
5  IF auto_opp THEN
6      auto_opp:=FALSE;
7      onsket_hoyde:=onsket_hoyde+ pos_endring;
8  END_IF;
9
10 IF auto_ned THEN
11     auto_ned:=FALSE;
12     onsket_hoyde:=onsket_hoyde - pos_endring;
13 END_IF;
```

Figur 5.5: Kode for posisjonsregulering, opp og ned knapp vist i figur 5.3.

Figur 5.6 viser koden for P-regulatoren. Linje 1 sjekker om reguleringen er aktivert ved trykk av startknappen på skjermen vist i figur 5.3. `onsket_hoyde` er ønsket høyde og `aktiv_hoyde` er den målte høyden. Ved å ta differansen mellom de to finner man avviket som vist i linje 2. P-leddet er avhengig av avviket og Kp verdien, som vist i ligning (5.1). Man justerer P-leddet ved å justere på Kp verdien, som forklart i delkapittel 5.1. Som nevnt tidligere blir Kp verdien justert på forsideskjermen, vist i figur 4.1.

```
1 IF reguler THEN
2     e:=onsket_hoyde-aktiv_hoyde;
3     P:=Kp*e;
```

Figur 5.6: Kode for P-leddet for posisjonsregulatoren.

I linje 27 i figur 5.7 blir hastigheten (`frek_hast`) satt lik P-leddet (`P`). Hastigheten er

nå avhengig av P-leddet som er basert på avviktet e. Dette gjør at regulatoren påfører en hastighet så lenge avviket ikke er null.

Linje 28 sjekker om avviket (`frek_hast`) er negativt. Hvis avviket er negativt blir også hastigheten `frek_hast` negativ. Frekvensomformeren tar ikke i mot negative tall så fortegnet på `frek_hast` må bli snudd i linje 29. Hvis avviket er negativt er lasten over ønsket posisjon, og lasten må bli senket ned ved å sette `q2a_reverse` aktiv som blir gjort i linje 30. `q2a_forward` blir deaktivert i linje 31 fordi både `q2a_reverse` og `q2a_forward` kan ikke være aktiv samtidig.

Betingelsen i linje 32 blir oppfylt hvis avviket (`frek_hast`) er positivt. Hvis avviket er positivt er lasten under ønsket posisjon, og lasten må bli heist opp ved å sette `q2a_forward` aktiv som blir gjort i linje 34. `q2a_reverse` blir deaktivert i linje 33 fordi både `q2a_forward` og `q2a_reverse` kan ikke være aktiv samtidig.

```
27    frek_hast:=LREAL_TO_REAL(P);
28    IF frek_hast<0 THEN
29       frek_hast:=-1*frek_hast;
30       q2a_reverse:=TRUE;
31       q2a_forward:=FALSE;
32    ELSE
33       q2a_reverse:=FALSE;
34       q2a_forward:=TRUE;
35    END_IF;
```

Figur 5.7: Kode for retningsbestemmelse.

I figur 5.8 blir hastigheten `frek_hast` begrenset til 1 Hz i linje 37-38. Hastighetsbegrensingen gjør at man unngår brå posisjonsendringer når avviket er stort. Dette ble testet i kapittel 5.3.3.

```
37    IF frek_hast >1 THEN
38       frek_hast:=1;
39    END_IF;
```

Figur 5.8: Kode for hastighetsbegrensning.

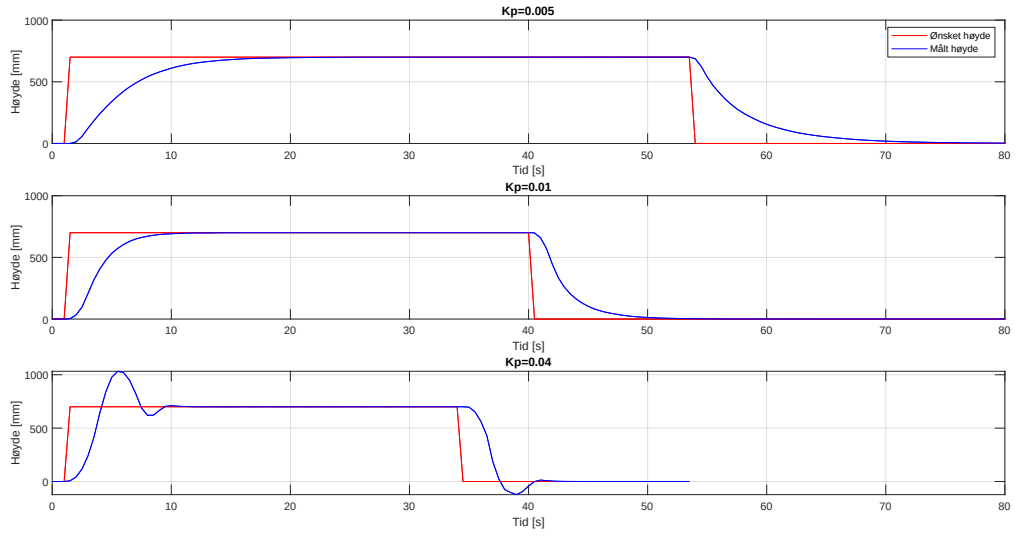
5.3 Posisjonsregulering eksperiment

I dette delkapittelet ble regulatoren utviklet testet. Ut ifra testen utført i delkapittel 4.2 ble det demonstrert at funksjonen som regner ut høyden har en feilmargin innen 10 mm. Derfor ble det valgt å ikke måle den fysiske høyden med hjelp av tommestokk, men data fra PLSen ble brukt.

Testene ble utført med "prøv og feil" metoden, som gikk ut på å teste flere parameterverdier og se hva som fungerte best for systemet. Reguleringsverdier kan også bli beregnet matematisk via Skogestads metode, men det ble ikke gjort på grunn av mangel på tid.

5.3.1 Valg av K_p verdi

For å finne en god K_p verdi til P-regulatoren ble flere forsøk utført. Testen ble gjort tre ganger med forskjellige K_p verdier. Alle gangene startet lasten på bakken og ble dratt opp til 700 mm. Lasten ble heist ned igjen til startposisjonen litt etter at lasten hadde stabilisert seg. Grafene i figur 5.9 viser resultatet av forsøket.



Figur 5.9: Grafen viser resultatet av testen med forskjellig K_p verdier.

Fra figur 5.9 ser man at høyere K_p verdier fører til at lasten blir heist opp raskere. Ved en K_p lik 0.005 bruker lasten 20 sekunder på å bli heist opp, og ved en K_p lik 0.01 bruker lasten 10 sekunder på turen opp. Grafene viser at ved dobbelt så stor K_p, bruker lasten halvparten så lang tid.

$K_p = 0.005$ er for treg for dette systemet, vist i øverste delfigur i figur 5.9. Det tok lang tid før lasten nådde 700mm, og tok lang tid på vei ned igjen. Både opp og ned brukte lasten 20s. Den hadde en tidskonstant $\tau = 5.25$, som betyr at systemet er tregt.

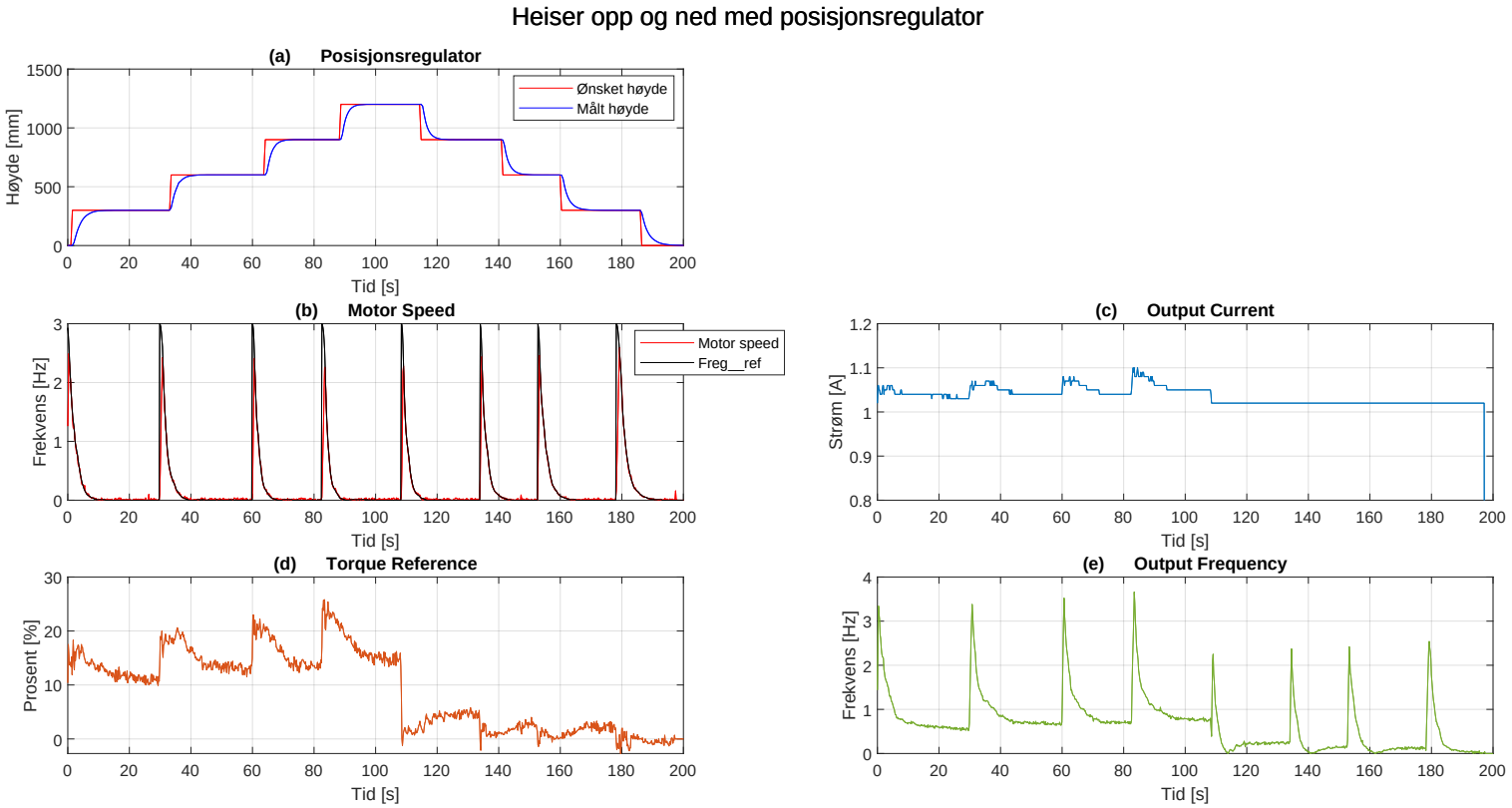
$K_p = 0.01$ vist i midterste delfigur i figur 5.9 ble lasten heist helt opp i løpet av 10 s uten oversving. Den hadde en tidskonstant $\tau = 3.25$, som er en indikasjon på hvor raskt systemet er. Grunnet ingen oversving og rask respons ble denne verdien for K_p brukt videre. Video av $K_p = 0.01$ finnes her: `https://www.youtube.com/watch?v=yT5BmVIFYO4`

$K_p = 0.04$ var for rask for dette systemet, vist i nederste delfigur i figur 5.9. Der oversteg høyden den skulle med 300 mm, før den svinget seg tilbake til 700 mm. På veien ned oversteg lasten ønsket posisjon før den svingte seg tilbake igjen. Oversving, spesielt på en vinsj, er ikke ønskelig fordi det har muligheten til å skade systemet. Den hadde en tidskonstant $\tau = 2.5$, som man kan se på grafen er dette for raskt. En video av oversving med K_p lik 0.04 finnes her: `https://www.youtube.com/watch?v=BCxTIUNEtSs`

5.3.2 Posisjonsregulator uten hastighetsbegrensning

For å demonstrere hvordan posisjonsregulatoren utviklet påvirker systemet uten hastighetsbegrensning ble et forsøk utført. Forsøket ble utført ved å heise en last på 1.5 kg opp 300 mm om gangen. Da lasten nådde 1200 mm ble den heist ned igjen med samme intervaller. Figur 5.10 viser resultatene av forsøket.

5.3 Posisjonsregulering eksperiment



Figur 5.10: Resultatene oppnådd fra testing av posisjonsregulatoren.

Fra figur 5.10 ser man at hver gang det er en endring i ønsket høyde gir det sprang for resten av systemet. Dette er fordi med en gang det er et avvik får P-regulatoren et stort sprang.

Fra delfigur b ser man at `Motor Speed` når 3 Hz og minkes når lasten nærmer seg ønsket høyde. Den har en rask oppstart fordi avviket er stort på starten av spranget. Da testen ble utført var ikke hastighetsbegrensningen implementert og er grunnen til de spisse toppene i `Motor Speed`. For å unngå de spisse toppene (bråe hastighetsendringer) og oppnå en jevnere posisjonsendring ble hastighetsbegrensning implementert og testet i delkapittel 5.3.3.

Delfigur c viser at `Output Current` også får store endringer når det er endring i avviket. Strømtrekket økes når lastens høyde over bakken økes. Dette er fordi når det

er mye innspolt vaier, blir total radius stor og det kreves mer strøm for å heise lasten opp. Rundt 115 s er lasten på vei ned og da hjelper tyngdekraften med arbeidet, derfor blir strømtrekket konstant 1.02 A og er mindre enn når den heiser opp.

`Tourque reference` (delfigur d) er et uttrykk for momentet basert på strømtrekket. Det er mulig å se ut ifra grafen at en økning i høyden fører til en økning i strømtrekket og moment. Dette er fordi større total radius fører til mer moment, som forklart i delkapitel 2.6.2. Rundt 115 s er det et stort dropp, dette er på grunn av endring i retning.

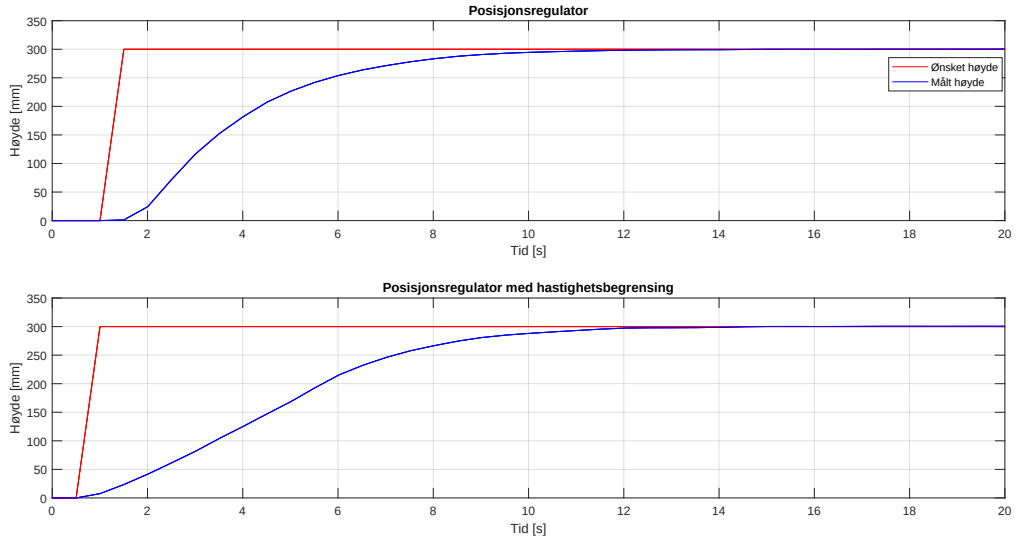
Fra figur 5.10 delfigur e ser man at `Output frequency` går litt over 3 Hz, grunnen til at denne må være høyere enn `Motor Speed` er fordi den må kompensere for lasten, som er på 1.5 kg. Som sagt tidligere er `Output Frequency` den frekvensen frekvensomformeren må sende ut for å oppnå ønsket `Motor Speed`. Man ser også at ved endring i retning, rundt 115 s, er `Output Frequency` bare rundt 2.5 Hz. Her hjelper tyngdekraften med å heise lasten ned, og det er ikke nødvendig å sende ut like høy frekvens.

5.3.3 Posisjonsregulator med hastighetsbegrensning

Fra delfigur b i figur 5.10 ser man at motorhastigheten blir høy med en gang høyden endres. Når den nærmer seg ønsket posisjon minker hastigheten, før den holder 0 Hz ved ønsket posisjon. For å unngå brå hastighetsendringer og jevnere posisjonsendring ble hastighetsbegrensning implementert, som vist i figur 5.8.

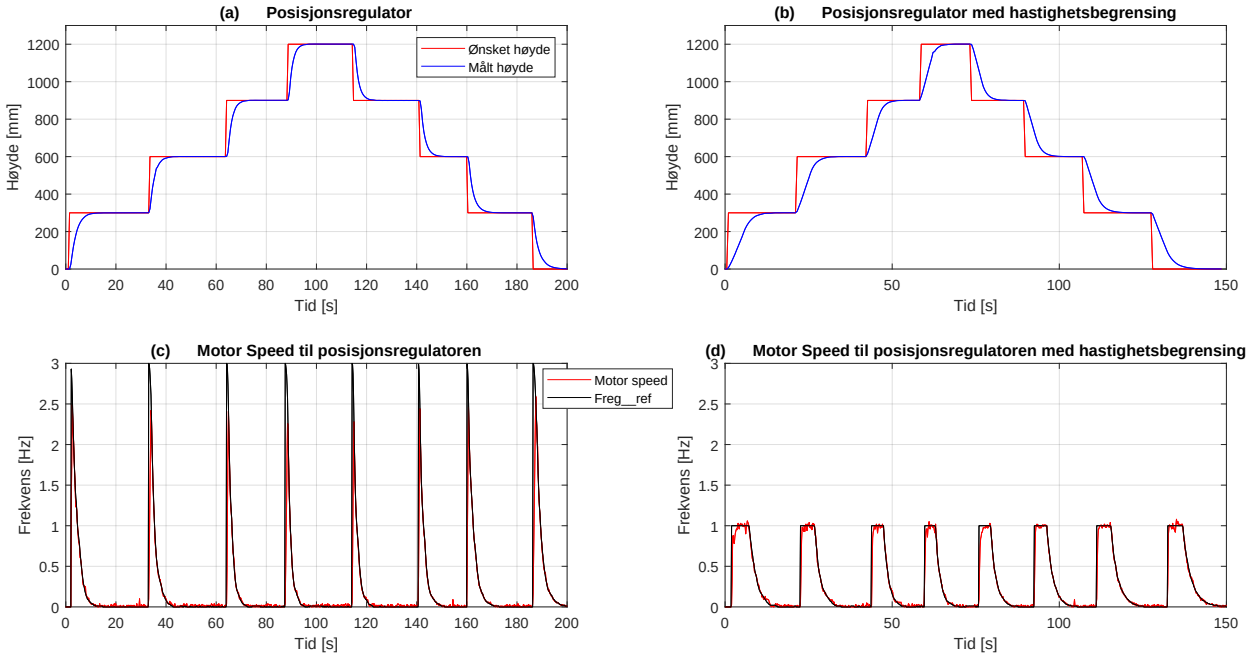
For å teste hastighetsbegrensningen ble $K_p = 0.01$ benyttet, slik som i delkapittel 5.3.2. Sprang på 300 mm ble brukt og maks motorhastighet systemet tillater er på 1 Hz. Figur 5.11 viser forskjellen på posisjonsendringen til posisjonsregulatoren med og uten hastighetsbegrensning.

5.3 Posisjonsregulering eksperiment



Figur 5.11: Grafene viser sprang på 300 mm uten hastighetsbegrensning og med hastighetsbegrensning.

Fra øverste delfigur i figur 5.11 ser vi at lasten når ønsket posisjon uten hastighetsbegrensning på ca. 11 s. Med hastighetsbegrensning når lasten ønsket høyde på ca. 12 s. Tidskonstanten uten hastigehtsbegrensing er på $\tau = 3.25$ og tidskonstanten med hastighetsbegrensing er $\tau = 5$. Ut ifra dette er det tydelig at med hastighetsbegrensning er systemet tregere enn uten. Selv om det gir et tregere system er det ønskelig å benytte hastighetsbegresning for å oppnå en jevnere hastighet ved høydeendring. Figur 5.12 viser motorhastighetene da forsøkene ble utført.



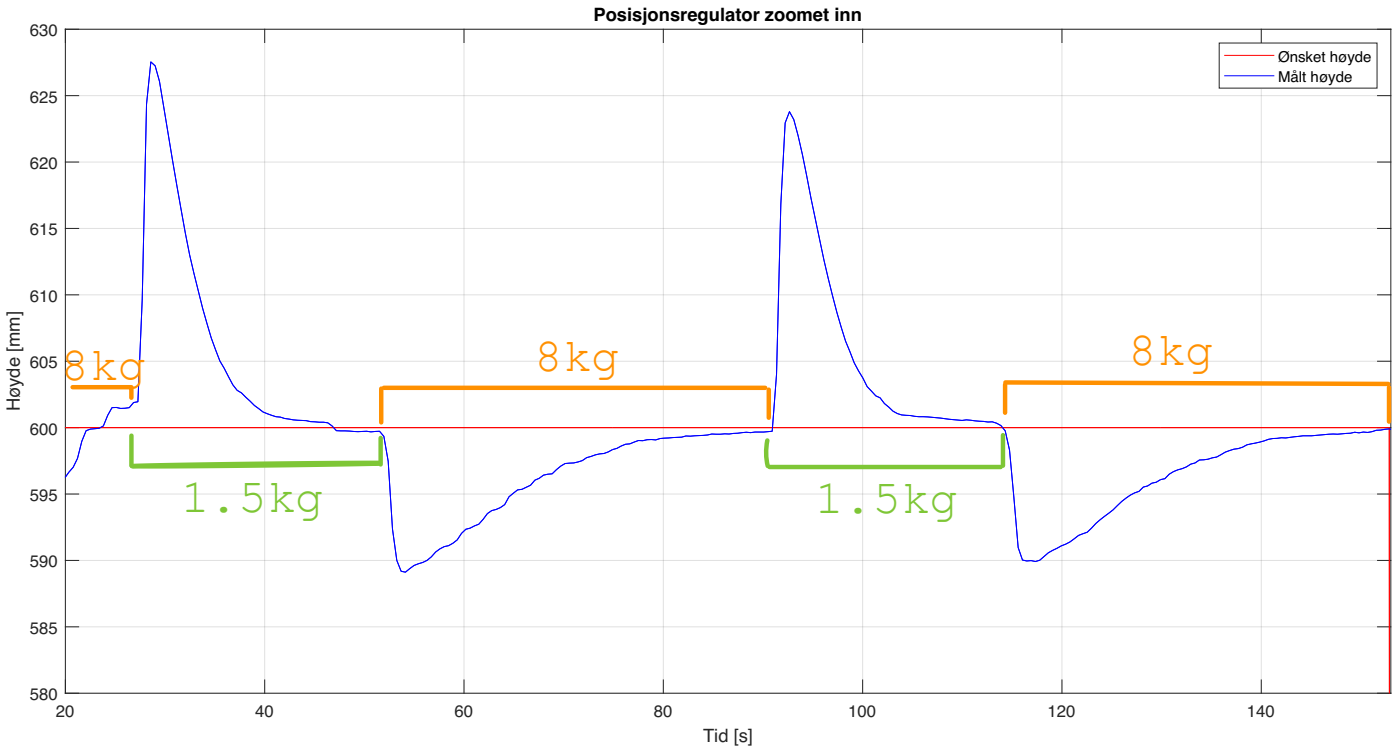
Figur 5.12: Resultatene oppnådd fra testing av P-regulator med hastighetsbegrensning i posisjonsregulatoren.

Figur 5.12 viser de forskjellige motorhastighetene uten og med hastighetsbegrensning. Med hastighetsbegrensning (delfigur d) ser man at `Motor Speed` stoppes nå på 1 Hz, som er det maks frekvensen er satt på. Det er ikke lengre de spisse toppene på 3 Hz, slik som i delfigur c. Med hastighetsbegrensning holder systemet makshastigheten litt lengre. Fra grafen ser man også at når lasten nærmer seg ønsket høyde, blir motorhastigheten lavere, som er resultatet av at avviket minker.

5.3.4 Posisjonsregulering kompenseringsegenskaper

For å teste kompenseringsegenskapene til posisjonsregulatoren ble et forsøk utført. Forsøket gikk ut på at en last ble heist opp til 600 mm hvor lastendringer ble utført. Figur 5.14 viser resultatet med merker på hvor lastendringene ble gjort. I dette forsøket var ikke hastighetsbegrensning implementert.

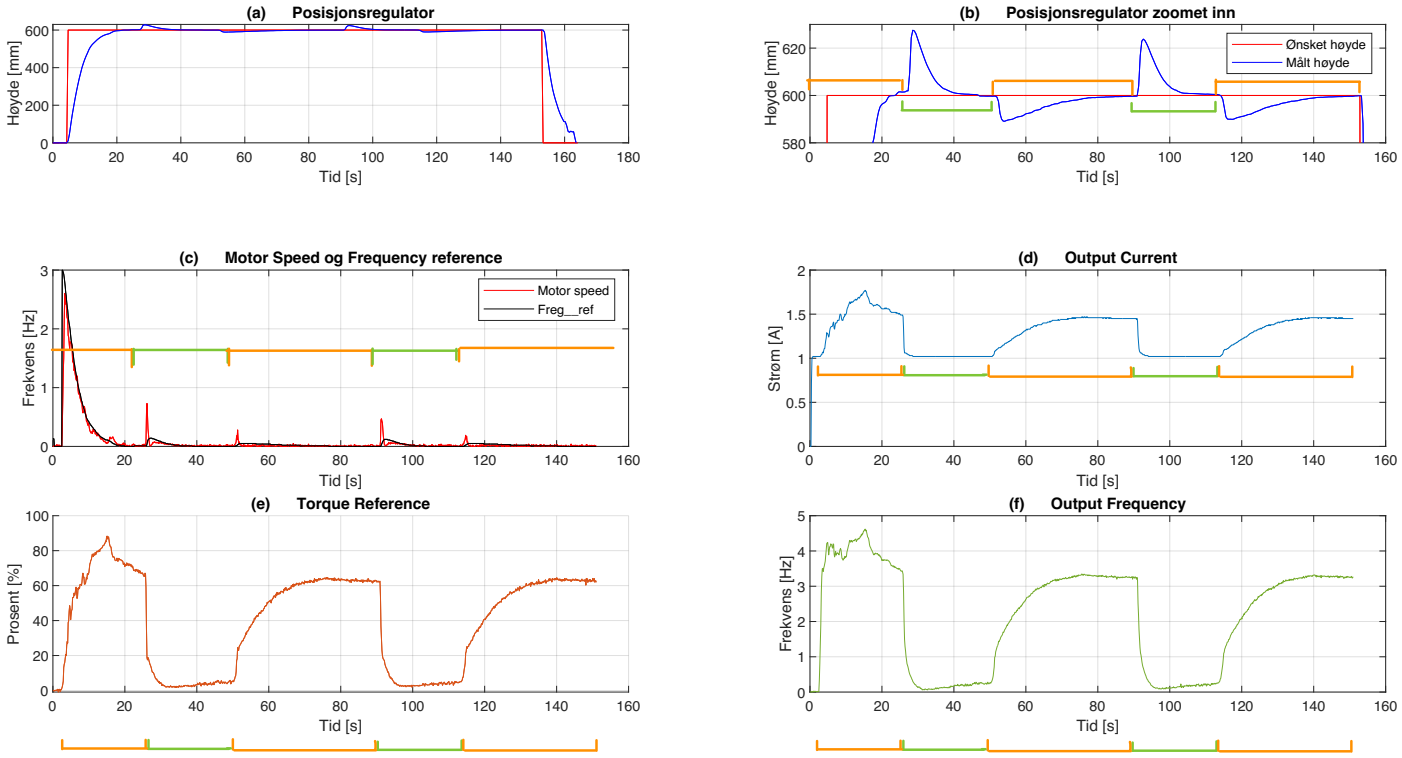
5.3 Posisjonsregulering eksperiment



Figur 5.13: Posisjonsregulator ($K_p = 0.01$ kompenseringstesten med forklaring på hvor tung lasten er.

Fra figur 5.13 ser man tydelig at ved lastendring regulerte systemet seg tilbake til ønsket posisjon. Når lasten ble minket til 1.5 kg var det en 25 mm endring i posisjonen, fordi frekvensomformeren jobbet med en hastighet for å holde oppe 8 kg. Systemet stabiliserte seg etter 25 sekunder. Da lasten ble økt til 8 kg sank lasten ned med ca. 10 mm, før den kom seg opp igjen i løpet av 50 sekunder. Synkingen er fordi den jobbet med en frekvens for å holde 1.5 kg. Figur 5.14 viser parameterene målt fra frekvensomformeren under forsøket.

5.3 Posisjonsregulering eksperiment



Figur 5.14: Posisjonsregulator ($K_p = 0.01$ kompenseringstesten med forklaring på hvor tung lasten er, Grønn=1.5 kg og Oransje=8 kg.

Fra figur 5.14 delfigur c ser man at hver gang det skjer en lastendring vil motorhastigheten `Motor speed` øke for å regulere lasten tilbake til ønsket posisjon.

Delfigur d viser at `Output Current` minker ved minking av last. Dette er fordi lasten har kommet over ønsket posisjon og lasten må bli heist ned. Som forklart tidligere hjelper gravitasjonen med å senke lasten, dette fører til lavere strømtrekk. Ved lastøkning kommer lasten under ønsket posisjon, og systemet må kompensere for dette. Når lasten blir heist opp, krever dette høyere strømtrekk fordi motoren jobber mot gravitasjonen.

Figur 5.14 delfigur e viser `Torque Reference`. Som forklart før er `Torque Reference` basert på strømtrekket, og har derfor samme form som `Output Current`.

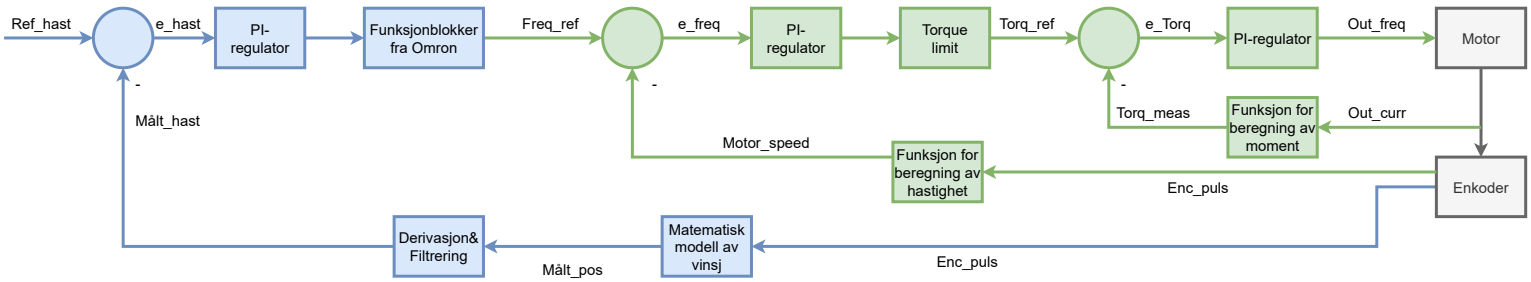
Delfigur f i figur 5.14 viser `Output Frequency`. Ved lastminking synker `Output`

`Frequency` hurtig for å ikke heise den lette lasten høyere opp. I tidsperiodene $25 < t < 35$ s og $95 < t < 115$ s ser man en liten øking som er for å regulere lasten tilbake til ønsket posisjon. Ved lastøkning stiger `Output Frequency` hurtig for å ikke slippe den tunge lasten lavere ned.

Kapittel 6

Regulering av innspolingshastigheten

For å oppnå konstant innspolingshastighet ble hastighetsregulering implementert. Som forklart i delkapittel 2.6.3 gir ikke en konstant vinkelhastighet en konstant innspolingshastighet, derfor må vinkelhastigheten bli regulert for å oppnå konstant innspolingshastighet. Ved å derivere posisjonen til lasten blir innspolingshastigheten beregnet. Lastens høyde over bakken (posisjon) blir beregnet basert på den matematiske modellen av systemet, som forklart i delkapittel 4.1. Figur 6.1 viser blokkdiagram av hastighetsregulering.



Figur 6.1: Blokkskjema for hastighetsregulering basert på innspolt vaier, hvor blå del er inne i PLSen, grønn er internt i frekvensomformeren, og grå representerer de fysiske komponentene.

Enkoder pulsene `Enc_puls` blir anvendt i den matematiske modellen av vinsjen for å

beregne høyden over bakken (`Målt_pos`). Videre blir `Målt_pos` anvendt til en egendefinert derivasjon og filtrerings funksjon, som blir til den målte hastigheten `Målt_hast`. *Referansehastigheten* `Ref_hast` er referansen i reguleringsløyfa, og tilsvarer ønsket innspolingshastighet oppgitt i [mm/s]. Reguleringsavviket e_{hast} er differansen mellom referansen og målt hastighet. Differansen går inn til en PI-regulator som via en funksjonsblokk fra Omron sender signalet `Freq_ref` til frekvensomformeren. Videre blir signalet beregnet i reguleringsløyfa for *closed loop* vektorkontroll og beregner pådraget `Out_freq` i [Hz].

6.1 PI-regulator

For hastighetsregulering må en PI-regulator bli implementert. Her er det ikke tilstrekkelig med en ren P-regulator fordi hastighetsregulatoren er derivasjonsbasert. En PI-regulator overvåker hele tiden avviket e fra ønsket hastighet og hastigheten lasten egentlig har, og justerer for å redusere avviket.

PI-regulatoren består av to deler: P-ledd og I-ledd. P-leddet får et avvik mellom målt hastighet og ønsket hastighet, altså e-en. P-leddet påfører et pådrag for å jevne ut hastighetsavviket med justeringsfaktor K_p. I-leddet følger med på det forrige hastighetsavviket og integrerer over tid, som den bruker til å fjerne restavviket. Justeringsfaktoren for I-leddet er K_i. Under er ligningen for PI-regulatoren brukt til hastighetsregulering.

$$u(t) = K_p \cdot e + K_i \int_0^t e dt \tag{6.1}$$

Funksjonen for en ren I-regulator blir som vist i ligning (6.2).

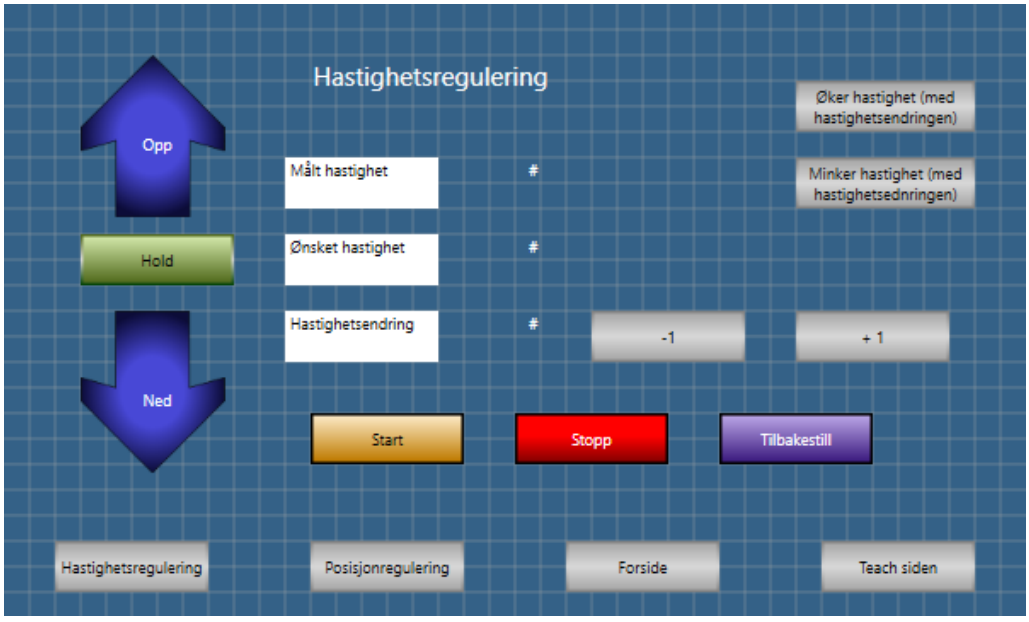
$$u(t) = K_i \int_0^t e dt \tag{6.2}$$

6.2 Hastighetsregulering kode

Denne delen omhandler koden for å regulere innspolingshastigheten. Mye av koden for hastighetsregulering ligner på koden for posisjonsregulering, fordi begge er basert på samme prinsipp.

6.2 Hastighetsregulering kode

Figur 6.2 viser skjermen for hastighetsregulering. Når pilopp- eller pilned knappen blir trykket på beveger lasten seg opp eller ned i konstant hastighet, som er lik ønsket hastighet. Ved trykk av den grønne hold-knappen holdes lasten i ro der den er. Start, stopp og tilbakestill- knappene fungerer på samme måte som i posisjonsregulering. Brukeren benytter +1 og -1 knappene for å sette ønsket hastighetsendring i mm/s. Ved å trykke på en av knappene øverst i høyre hjørne øker eller minker hastigheten basert på verdien i hastighetsendring.



Figur 6.2: Skjerm til hastighetsregulering.

I figur 6.3 linje 2 blir posisjonen (`posisjon`) satt lik høyden lasten har over bakken (`aktiv_hoyde`). `aktiv_hoyde` kommer fra den matematiske modellen av systemet forklart i delkapittel 4.1. I linje 3 blir hastigheten (`hast`) beregnet via numerisk derivasjon av posisjonen. Linje 4 setter variabelen `gammel_posisjon` lik den forrige posisjonen for å benytte variabelen videre.

I linje 6 blir hastigheten filtert for å fjerne støy. Filteret er et IIR-filter og er avhengig av variabelen `parameter` som vi bestemmer. For å regulere hastigheten både opp og ned blir absoluttverdien av hastigheten (`hast`) beregnet. Dette er fordi frekvensomformeren bare aksepterer positive tall, men registrerer kjøreretningen (`q2a_reverse` og `q2a_forward`). Linje 7 setter variabelen `gammel_filter_hast` lik den forrige filtrerte hastigheten for å benytte variabelen videre.

```
2    posisjon:=aktiv_hoyde;
3    hast:=(posisjon - gammel_posisjon)/ (LINT_TO_LREAL( TimeToNanoSec(Ts))/1000000000);
4    gammel_posisjon:=posisjon;
5
6    filtrert_hast:=parameter* ABS(hast)+(1-parameter)*gammel_filtrert_hast;
7    gammel_filtrert_hast:=filtrert_hast;
```

Figur 6.3: Deriverer posisjonen og filtrerer derivert verdi.

Figur 6.4 viser koden for P-leddet i hastighetsreguleringen. I linje 10 blir avviket (`e`) definert som differansen mellom det ønskede hastigheten (`onsket_hast`) og den filterte hastigheten (`filtrert_hast`). For å beregne P-leddet til hastighetsreguleringen blir ligning (5.1) benyttet i linje 11.

```
10   e:= onsket_hast - filtrert_hast;
11   P:=Kp_hast*e;
```

Figur 6.4: P-leddet for hastighetsregulering.

Figur 6.5 viser koden for I-leddet. I linje 13 til 15 blir tidsskrittet definert. I linje 17 blir I-leddet (`I_ledd`) definert basert på Euler forover metoden.

```
13   tid_nå:=GetTime();
14   Ts:=SUB_DT_DT(tid_nå,tid_før);
15   tid_før:=tid_nå;
16
17   I_ledd:=gammel_I_ledd +Ti_hast*(e* LINT_TO_LREAL( TimeToNanoSec(Ts))/1000000000);
```

Figur 6.5: Tidskritt og I-leddet for hastighetsregulering.

I linje 19 til 23 i figur 6.6 begrenses integralet. Det ble brukt en grenseverdi for å hindre integralet i å bli uendelig stort. Når ønsket hastighet er nådd, altså at avviket er null, blir integralet større enn null. Dette gir avvik, og systemet vil korrigere en feil som ikke eksisterer. Grenseverdien gjør at I-leddet integrerer mot null veldig fort. Det ble utført tester for å bestemme denne grenseverdien.

I linje 25 blir den forrige verdien for I-leddet (`gammel_I_ledd`) satt. Denne verdien blir brukt i linje 17 i figur 6.5 for å integrere basert på Euler forover metoden.

I linje 26 i figur 6.6 blir hastigheten (`frek_hast`) satt lik summen av P- og I-leddet som vist i ligning (6.1). Hastigheten er nå avhengig av P-leddet og I-leddet som er

basert på avviktet e. Dette gjør at regulatoren påfører en hastighet så lenge avviket ikke er null.

```
19      IF l_ledd > 1 THEN
20          l_ledd:= 1;
21      ELSIF l_ledd<-1 THEN
22          l_ledd:=-1;
23      END_IF;
24
25  gammel_l_ledd:=l_ledd;
26  frek_hast:=LREAL_TO_REAL(P+l_ledd);
```

Figur 6.6: Begrenser integralet og beholder gamle verdier.

Figur 6.7 viser koden for kjøreretningen til vinsjen. Linje 30-34 gjør at når brukeren trykker på pilopp-knappen vist i figur 6.2 blir lasten heist oppover. Linje 36-40 gjør at når brukeren trykker på pilned-knappen vist i figur 6.2 blir lasten heist nedover. Linje 31 og 37 resetter opp og ned knappene selv om motoren fortsatt utfører kommandoen.

```
30  IF kjor_opp THEN
31      kjor_opp:=FALSE;
32      q2a_reverse:=FALSE;
33      q2a_forward:=TRUE;
34  END_IF;
35
36  IF kjor_ned then
37      kjor_ned:=FALSE;
38      q2a_reverse:=TRUE;
39      q2a_forward:=FALSE;
40  END_IF;
```

Figur 6.7: Definerer kjørerretning for hastighetsregulering.

Figur 6.8 linje 42-44 blir variabelen `hast_endring` definert som hastighetsendringen brukeren ønsker. If-setningen gjør at det ikke er mulig med negative tall, og setter 5 som normalverdi. Dette er fordi hastighetsendringen bare skal bli oppgitt i positive tall.

Figur 6.8 viser koden for å sette ønsket innspolingshastighet, holde lasten i ro og hastighetsbegrensning. I linje 46 og 48 blir variabelen `hast_endring` benyttet til å endre hastigheten. Ved å trykke på "øker hastighet"-knappen økes ønsket hastighet med det variabelen `hast_endring` er satt til. Det samme skjer ved å trykke på "minker hastighet"-knappen, men at ønsket hastighet minker med variabelen `hast_endring`.

```
42  IF hast_endring<=0 THEN
43      hast_endring:=5;
44  END_IF;
45
46  IF oke THEN
47      oke:=FALSE;
48      onsket_hast:=onsket_hast + hast_endring;
49  ELSIF minke THEN
50      minke:=FALSE;
51      onsket_hast:=onsket_hast - hast_endring;
52  END_IF;
```

Figur 6.8: Koden for å øke eller minke farten, holde lasten i ro og hastighetsbegrensning.

Linje 54-56 i figur 6.9 inneholder hold funksjonen for å holde lasten i ro ved 0 Hz ved trykk av den grønne hold knappen i figur 6.2. I linje 58-60 er det satt en hastighetsbegrensing til 1.5 Hz.

```
54  IF hold THEN
55      frek_hast:=0;
56  END_IF;
57
58  IF frek_hast >1.5 THEN
59      frek_hast:=1.5;
60  END_IF;
```

Figur 6.9: Koden for holdknappen og hastighetsbegrensning.

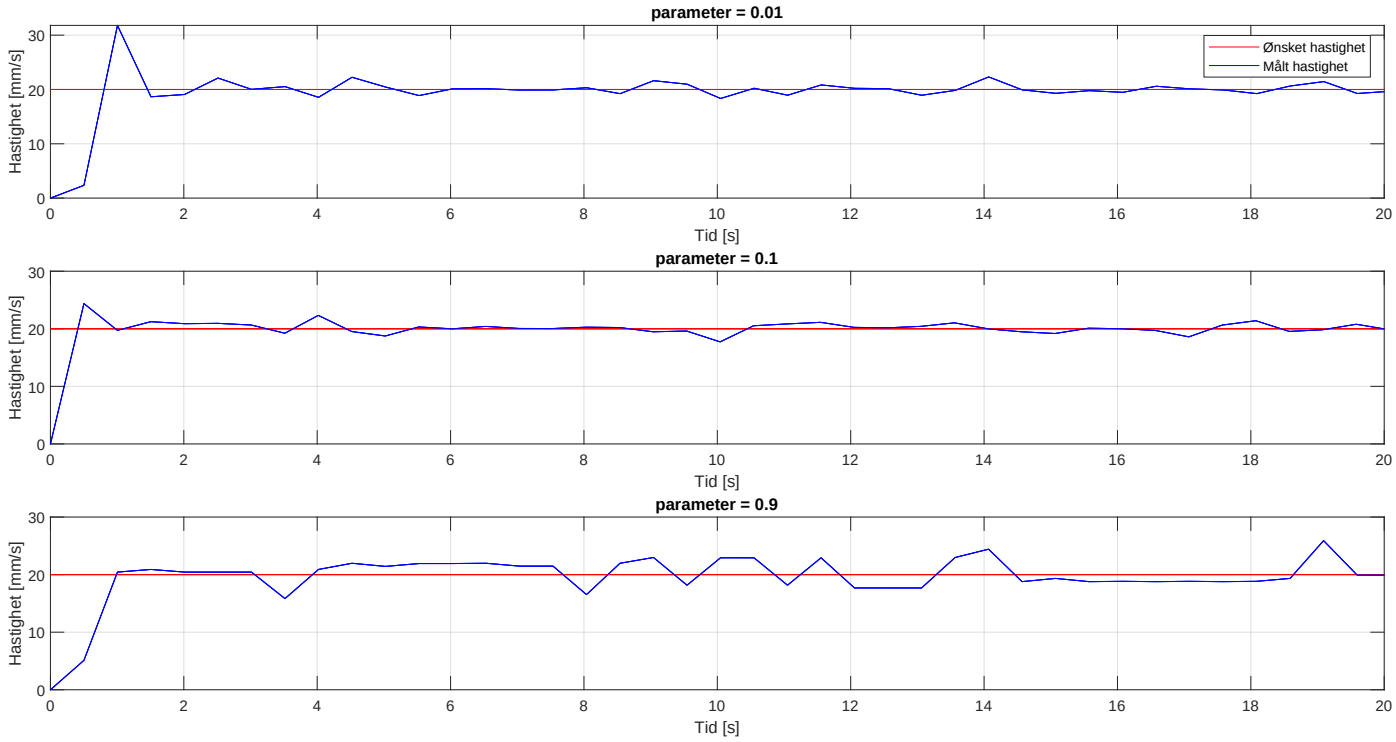
6.3 Hastighetsregulering eksperiment

For å bestemme parameterverdiene til hastighetsregulatoren ble flere tester utført på systemet. Testene ble utført med "prøv og feil" metoden, som gikk ut på å teste flere parameterverdier og se hva som fungerte best for systemet. Reguleringsverdier kan også bli beregnet matematisk via Skogestads metode, men det ble ikke gjort på grunn av mangel på tid.

I koden blir absoluttverdien av hastigheten beregnet, derfor har ikke hastigheten på vei opp og ned forskjellige fortegn. Markører på når den er på vei opp og når den er på vei ned er derfor blitt påpekt i figurer som inkluderer retningsendring.

6.3.1 Valg av filter parameterverdi

For å velge filterets parameterverdi ble tre forskjellige parametere testet. I testene ble en I-regulator brukt med $K_i = 0.1$. Alle testene ble utført ved at lasten ble heiset opp med en innspolingshastighet på 20 mm/s. Parameterverdiene 0.01, 0.1 og 0.9 ble testet, figur 6.10 viser resultatet etter testene.



Figur 6.10: Resultatet av forskjellige filter parameterverdier med $K_i = 0.1$.

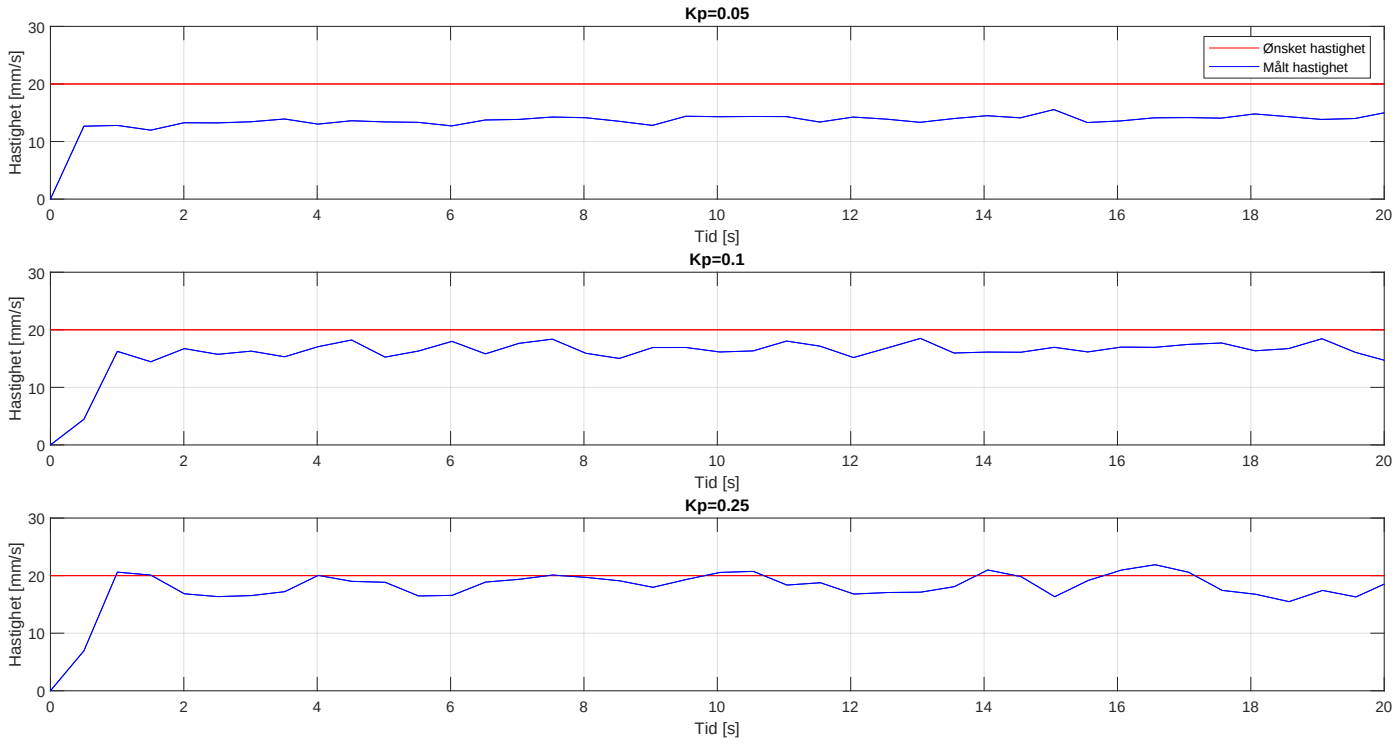
Filtreringsverdiene på 0.01 (øverste delfigur) og 0.1 (midterste delfigur) ser relativt like ut. Når filterverdien er lik 0.01 dominerer de filtrerte verdiene og den deriverte verdien har nesten ingen påvirkning. Med filterverdi lik 0.1 er det den filtrerte verdien som dominerer, men den deriverte verdien har påvirkning. Derfor ble parameter=0.1 benyttet videre for dette systemet.

Nederst i figur 6.10 ser man tydelig at en filterverdi på 0.9 er for høy. Systemet hadde store hastighetsendringer som var tydelig på hvordan lasten bevegde seg. Her

dominerer den deriverte verdien, og den filtrerte verdien har nesten ingen betydning, som er grunnen til at signalet blir så støyete.

6.3.2 Valg av K_p verdi

For å finne en god K_p verdi til P-regulatoren ble flere forsøk utført. Testene gikk ut på å heise en last opp med en konstant innspolingshastighet på 20 mm/s. I-leddet var ikke aktivt i denne testen. Figur 6.11 viser resultatet av forsøkene.



Figur 6.11: Resultatet av forskjellige K_p verdier for hastighetsregulering.

Fra grafene i figur 6.11 ser man den målte hastigheten og ønsket hastighet plottet mot hverandre. Gjennomsnittshastigheten ble beregnet ved å måle tiden lasten brukte på å komme seg til en viss høyde.

Øverste delfigur i figur 6.11 viser en K_p verdi lik 0.05. Med liten K_p verdi blir avviket

større, dette er mulig å se ved at den faktiske innspolingshastigheten er alltid under den ønskede. Gjennomsnittshastigheten beregnet er lik 13.75 mm/s, som er ca. det man ser fra grafen. Selv om den har stort avvik er innspolingshastigheten stabil.

Midterste delfigur viser en K_p verdi lik 0.1. Her ser man at innspolingshastigheten holder seg under den ønskede hastigheten. Grunnet en større K_p verdi ligger den virkelige innspolingshastigheten nærmere den ønskede, men innspolingshastigheten er mindre stabil. Gjennomsnittshastigheten beregnet er lik 16.57 mm/s, som er ca. det man ser fra grafen.

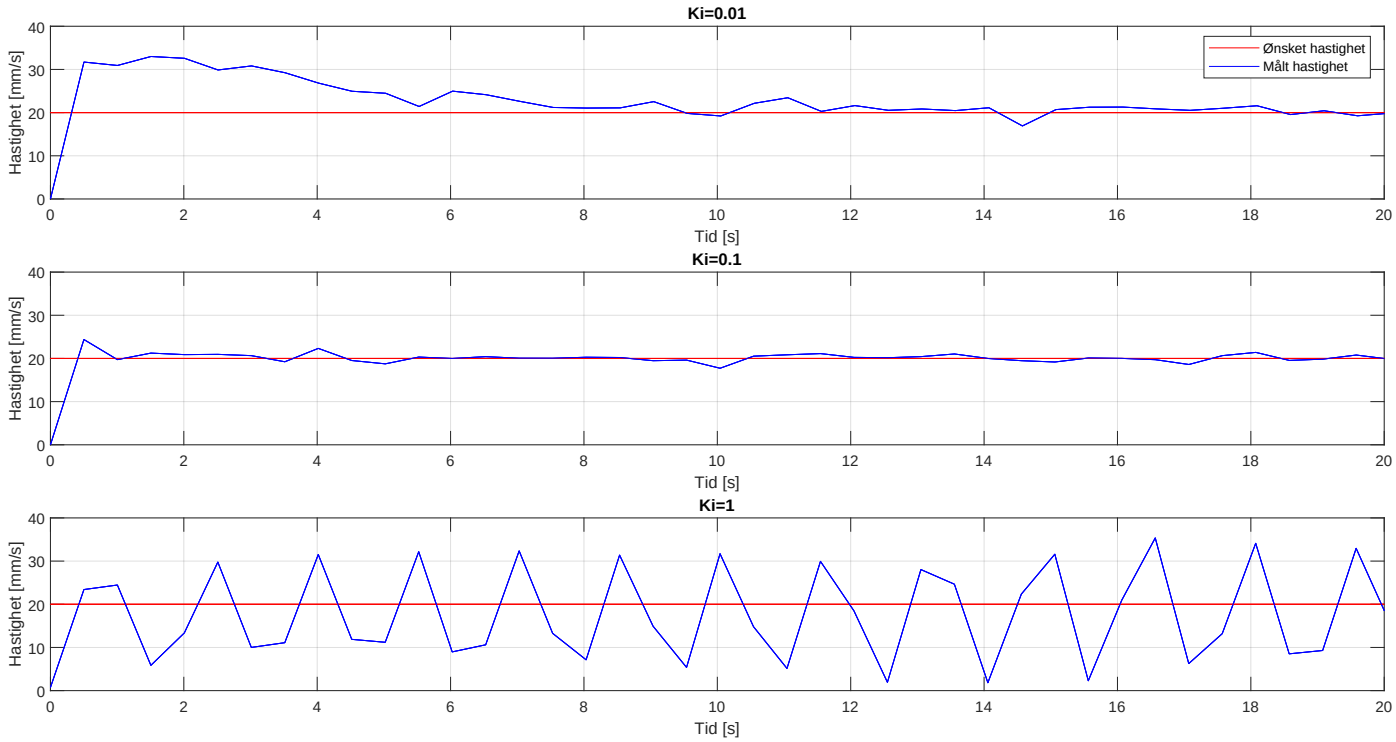
Nederste delfigur viser K_p verdi lik 0.25. Signalet svinger rundt den ønskede innspolingshastigheten med en del støy. Gjennomsnittshastigheten beregnet er lik 18.58 mm/s, som er ca. det man ser fra grafen. Denne gjennomsnittshastigheten ligner på den ønskede innspolingshastigheten, men den er veldig støyete.

Testen demonstrerer at lave K_p verdier gir lavere innspolingshastighet og derfor større avvik. En ren P-regulator er ikke tilstrekkelig for dette systemet, siden regulatoren ikke klarer å oppnå den ønskelige innspolingshastigheten uten støy. Som forklart i delkapittel 5.1 er systemet en integrerende prosess, ved å derivere posisjonen fjernes integrasjonen. Funksjonen for hastighetsreguleringen er derivasjonsbasert, derfor vil en integrator forbedre systemet.

6.3.3 Valg av K_i verdi

For å finne en god K_i verdi til I-regulatoren ble flere forsøk utført. Testene utføres ved at en last ble heiset opp med en konstant innspolingshastighet på 20 mm/s. P-leddet var ikke aktivt i denne testen. Figur 6.12 viser hastigheten målt mot den ønskede hastigheten.

6.3 Hastighetsregulering eksperiment



Figur 6.12: Resultatet av forskjellige K_i verdier for hastighetsregulering.

Fra den øverste grafen i figur 6.12 ser man at $K_i = 0.01$ har et stort oversving på starten opp til ca. 30 mm/s. Etter 8 s reguleres innspolingshastigheten nærmere den ønskede hastigheten. Det er ønskelig å ha minst mulig oversving, og derfor ble ikke denne K_p verdien benyttet.

Midterste graf med K_i lik 0.1 har oversving til 25 mm/s ved start, men reguleres tilbake til ønsket hastighet innen 1 s. Videre var innspolingshastigheten rundt den ønskede hastigheten med lite støy. Videoen viser denne testen: `https://www.youtube.com/watch?v=powrU1KBiSA`

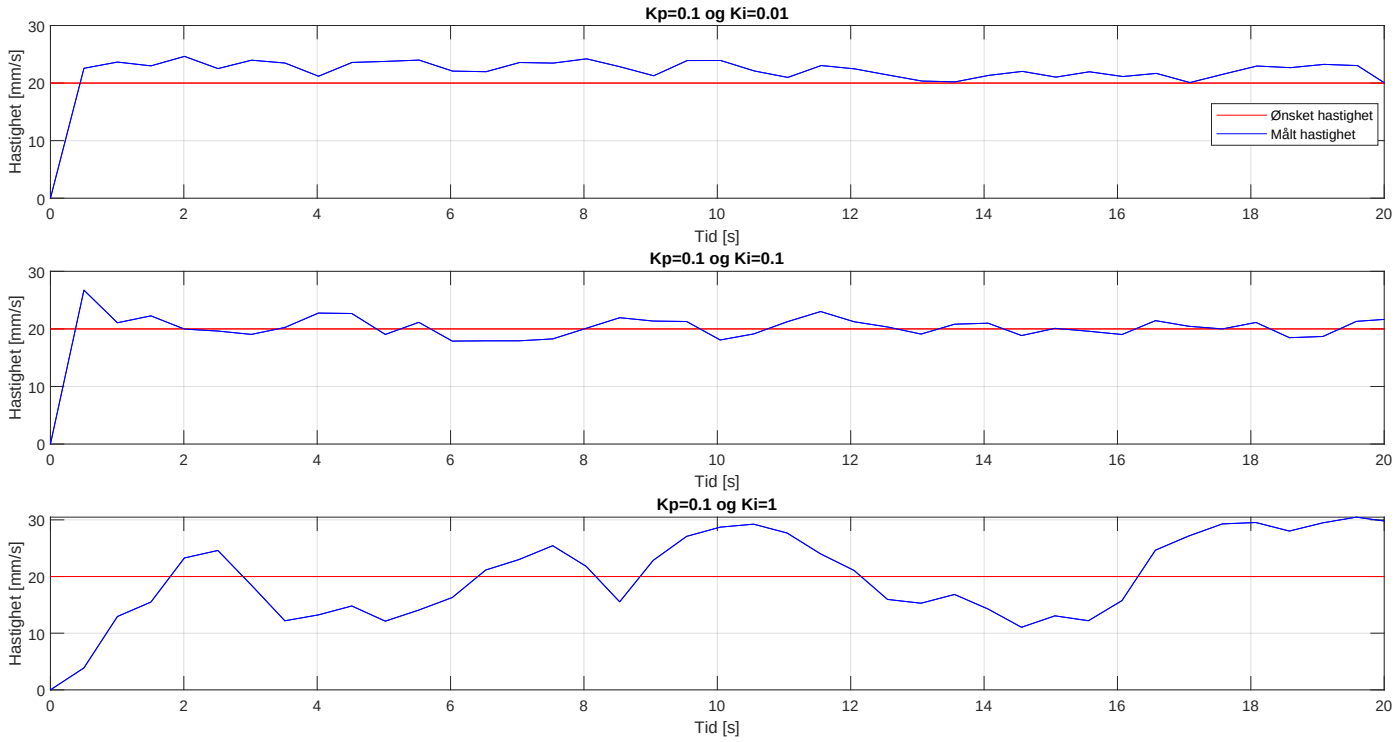
Fra nederste graf ser man at ved K_i lik 1 var hastigheten ekstremt varierende. Dette førte til at lasten ble løftet opp med rykk, som er mulig å se i denne videoen: `https://www.youtube.com/watch?v=dlohKr7IZww`. $K_i = 1$ er for stor til å bruke som reguleringsverdi basert på figur 6.12 og videoen.

Ut ifra grafene over er en K_i verdi på 0.1 den beste verdien for en ren I-regulator.

Ren I-regulator fungerer bedre enn en ren P-regulator. Med en kombinasjon av begge leddene blir avviket mest mulig lik null.

6.3.4 Testing av PI-regulator

For å bestemme en god kombinasjon av K_p verdi og K_i verdi for PI-regulatoren ble flere forsøk utført. Lasten ble heist opp med en konstant innspolingshastighet på 20 mm/s. De første grafene går ut på å ha konstant K_p verdi men endre K_i verdien. K_p verdiene er tatt fra P-ledd testen i delkapittel 6.3.2 og K_i verdiene er tatt fra I-ledd testen i delkapittel 6.3.3. Figur 6.13 viser de forskjellige K_i verdiene med $K_p = 0.1$.

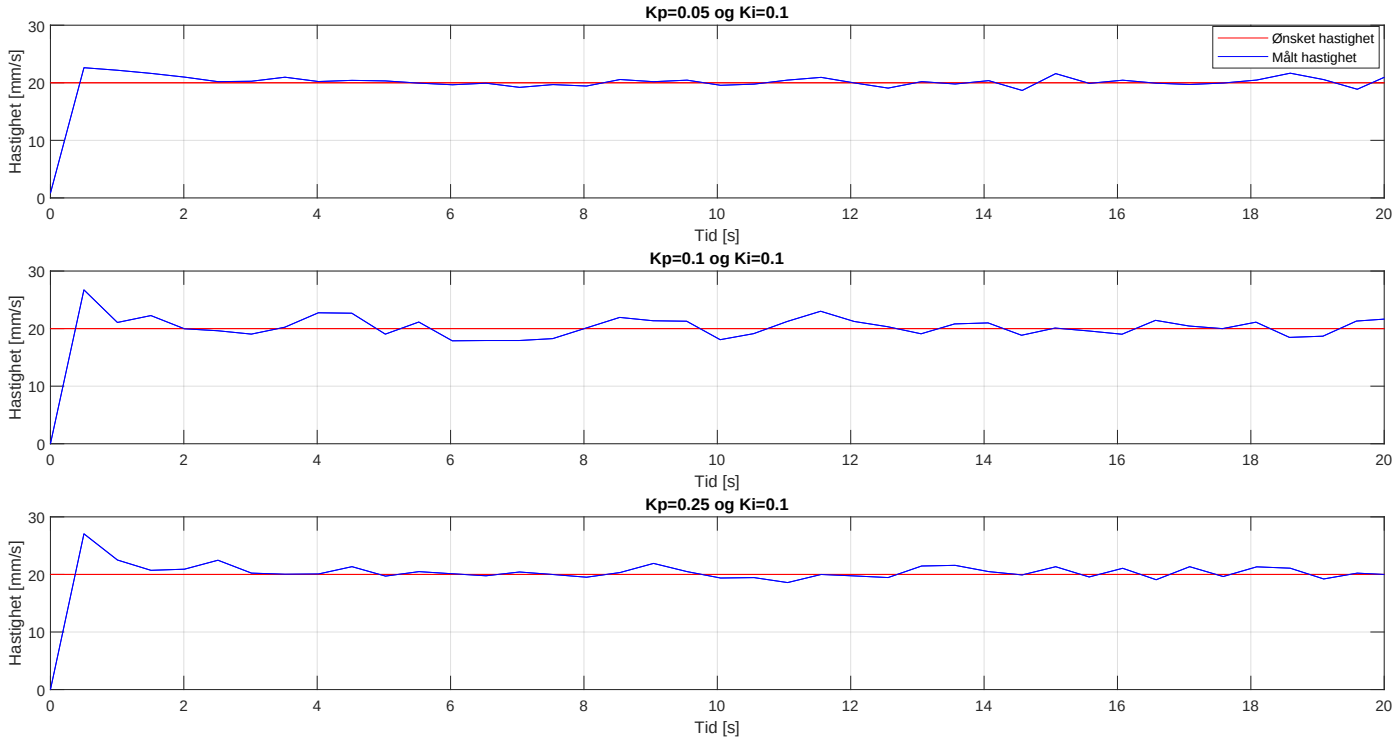


Figur 6.13: Konstant $K_p = 0.1$ med ulike K_i tallverdier.

De to øverste delfigurene i figur 6.13 holder seg innenfor den ønskede innspolings-hastigheten. Begge grafene er støyete grunnet høy K_p verdi. Med videre testing av forskjellige K_p verdier viser en bedre kombinasjon av K_p og K_i verdier, denne testen

er vist i figur 6.14.

Fra den nederste grafen i figur 6.13 ser man at lasten beveger seg med rykk grunnet for høy K_i verdi. Dette er samme resultat og K_i verdi som nederste delfigur i figur 6.12.



Figur 6.14: Konstant K_i med ulike K_p tallverdier.

Øverste delfigur i figur 6.14 viser en kombinasjon av $K_p = 0.05$ og $K_i = 0.1$. Ved start er det et oversving til 23 mm/s, men dette reguleres til ønsket hastighet i løpet av ca. 2 s. Etter 3 s er innspolingshastigheten stabil og det er lite avvik mellom ønsket og målt hastighet. Video av testen er vist i: `https://www.youtube.com/watch?v=KbwIeTrQLhk`

For referanse er $K_p = 0.1$ og $K_i = 0.1$ vist igjen i midten av figur 6.14, og det er tydelig at andre K_p verdier gir bedre resultater.

En kombinasjon av $K_p = 0.25$ og $K_i = 0.1$. er vist i nederste delfigur i figur 6.14.

Den har et oversving til 26 mm/s ved starten, men dette reguleres til ønsket hastighet etter ca. 0.5 s. Det er lite avvik mellom ønsket og målt hastighet, men innspolingshastigheten er støyete.

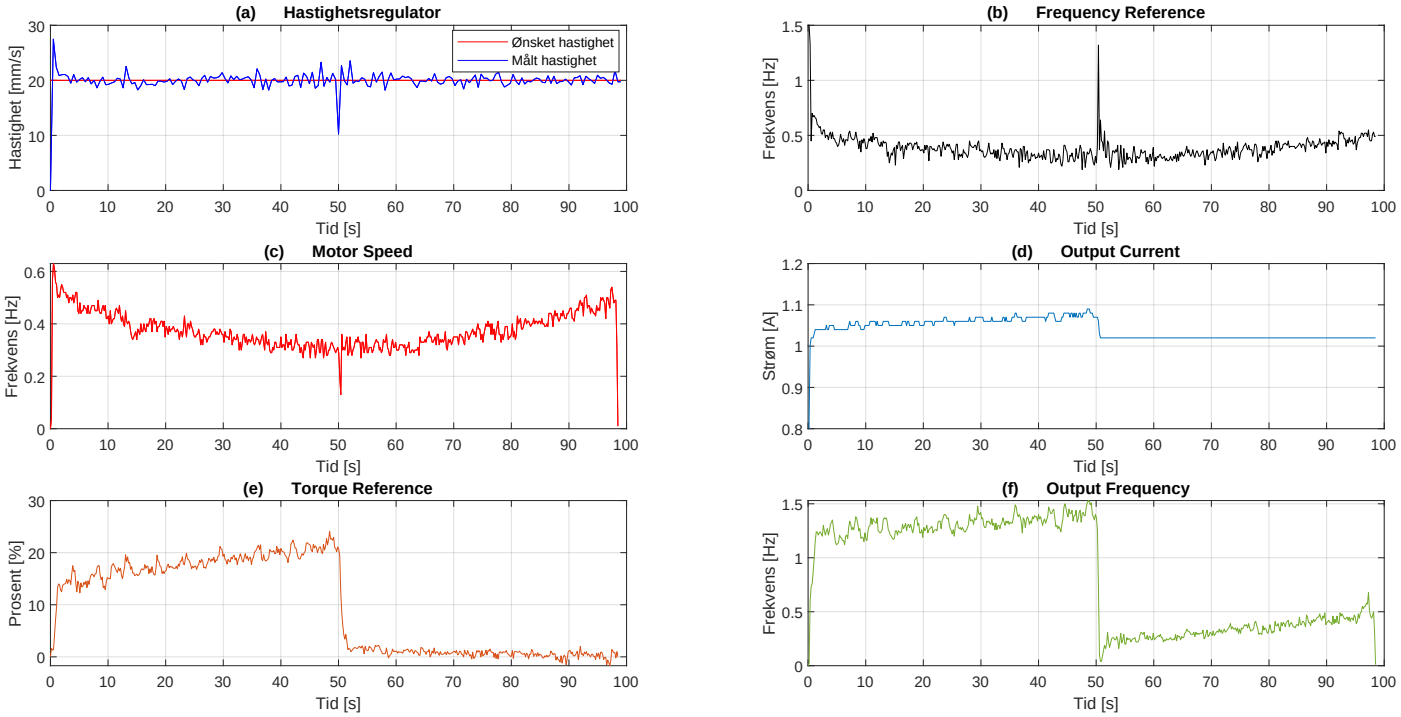
Fra testen i delkapittel 6.3.2 var en K_p verdi lik 0.05 minst støyete. Fra testen i delkapittel 6.3.3 hadde en K_i verdi lik 0.1 minst avvik. Det er derfor ikke overraskende at en kombinasjon av disse verdiene gir gode reguleringsegenskaper. For dette systemet er derfor $K_p = 0.05$ og $K_i = 0.1$ verdiene for PI-regulatoren brukt videre.

6.3.5 Resultatet av hastighetsregulering med PI-regulator

Som nevnt før vil konstant vinkelhastighet (motorhastighet) på en vinsj spole lasten opp raskere grunnet den økende totale radiusen. Hensikten med hastighetsregulering var at innspolingshastigheten forblir konstant, og da må motorhastigheten minke etterhvert som mer vaier spoles inn.

For å verifisere at motorhastigheten minker ved innspoling og øker ved utspoling ble en test utført. Testen gikk ut på at en last ble heist opp til en bestemt høyde og så ned igjen med samme innspolingshastighet på 20 mm/s. Figur 6.15 viser resultatet av forsøket med en PI-regulator.

6.3 Hastighetsregulering eksperiment



Figur 6.15: Resultatet av testen med PI-regulator der $K_p = 0.05$ og $K_i = 0.1$, retningsendring skjer ved ca 50s.

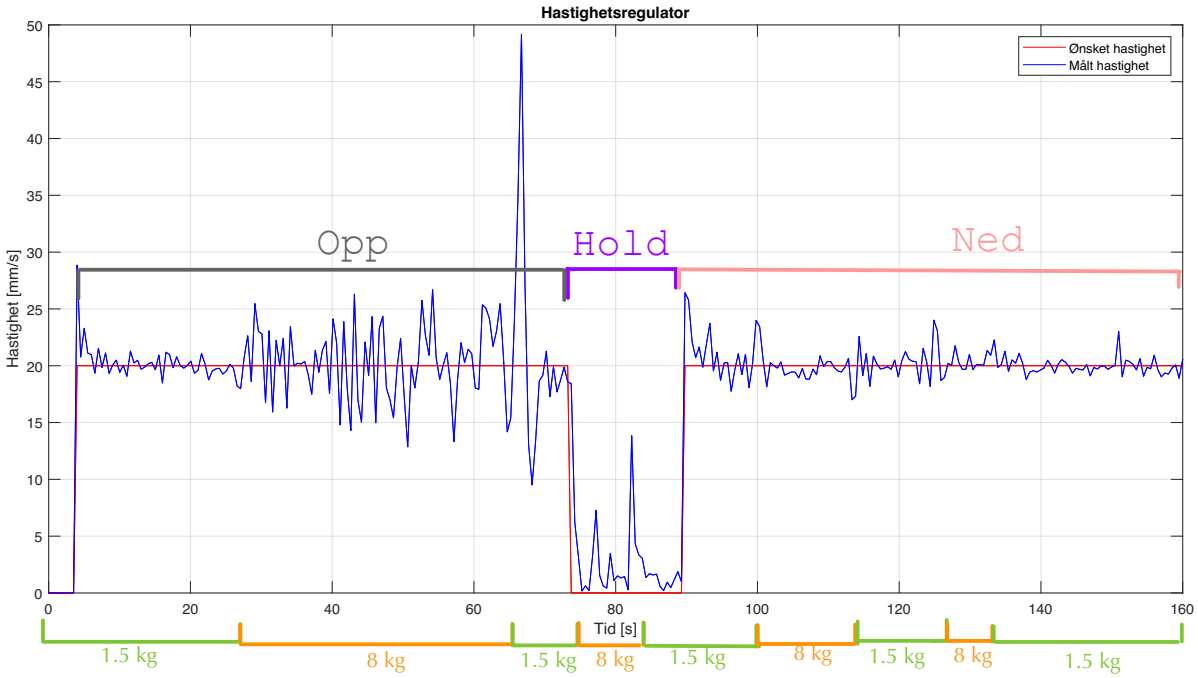
I likhet med posisjonsreguleringstestene går `Output Current` (delfigur d) og `Tourque Reference` (delfigur e) sakte oppover ved innspoling. Dette er grunnet det økende momentet ved økende radius på trommelen som forklart i delkapittel 2.6.2. På veien ned er strømtrekket rundt 1 A, systemet trenger ikke bruke mer strøm fordi gravitasjonskraften hjelper lasten på veien ned. Selv om `Torque Reference` følger etter `Output Current`, synker `Torque Reference` gradvis etter retningsendringen. Dette er også grunnet mindre moment når radiusen minker.

`Output Frequency`, vist i figur 6.15 delfigur f, må stige for å oppnå den ønskede hastigheten ved høyere moment. Når lasten er på vei ned må den også sakte stige, selv med minkende moment. Grunnen er at motorhastigheten (`Motor speed`) må øke for å oppnå ønsket innspolingshastighet på vei ned med minkende total radius.

Fra delfigur c i figur 6.15 ser man at motorhastigheten (`Motor speed`) går ned ved innspoling. Ved retningsendring rundt 50 s går den sakte opp igjen. Dette er en bekreftelse av at hastighetsregulatoren fungerer som forventet.

6.3.6 Kompenseringsegenskaper til hastighetsregulatoren

For å teste kompenseringsegenskapene til hastighetsregulatoren ble et forsøk utført. Forsøket gikk ut på heise en last opp og ned mens lastendringer ble utført. Lasten ble også holdt i ro hvor lastendringer ble gjort. Figur 6.16 viser resultatet fra forsøket, med forklaringer på hvor last ble lagt til og fra.

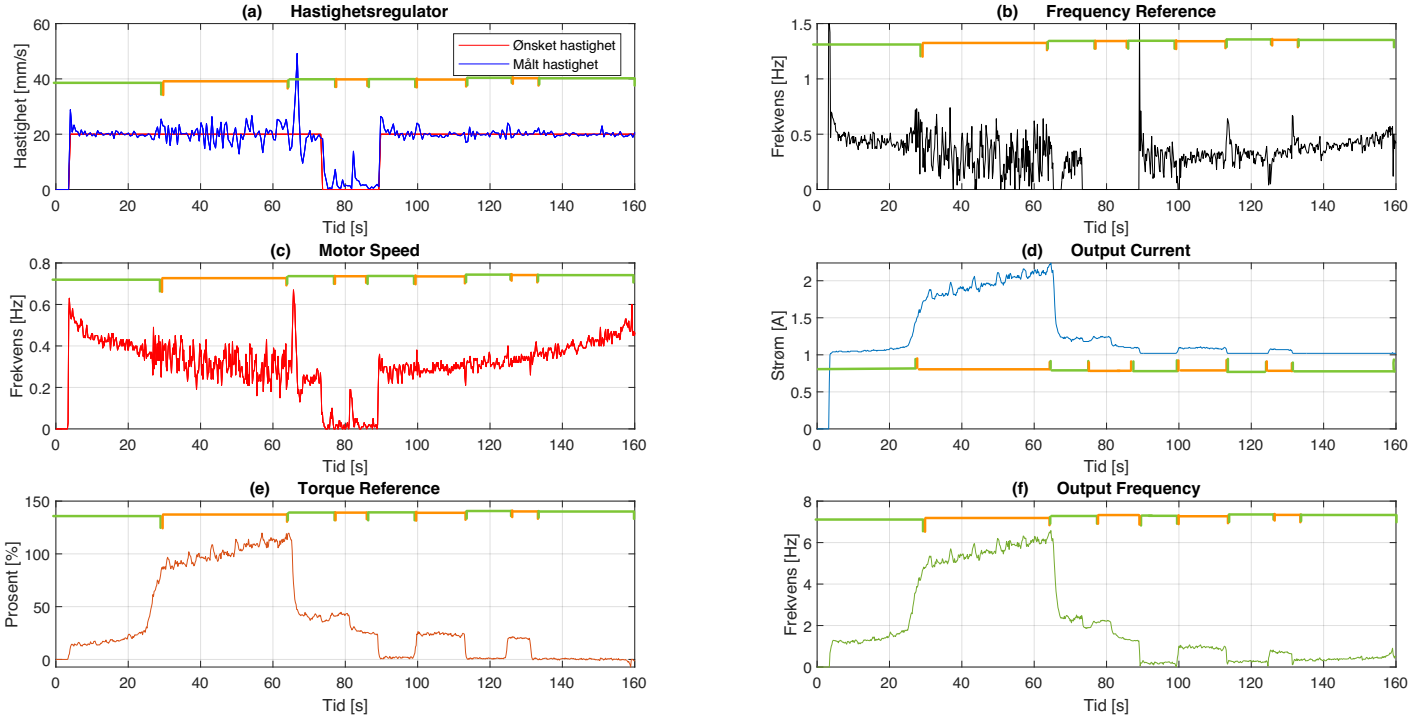


Figur 6.16: Hastighetsregulerings kompenseringsegenskaper med forklaring på hvor tung lasten er, og hvilken retning lasten beveger seg

Figur 6.16 viser hvor tung lasten var til hvilket tidspunkt, og når holdefunksjonen ble benyttet. Når lasten er på vei opp og får en lastøkning til 8 kg, ser man at den målte hastigheten blir støyete. Dette er fordi systemet prøver å regulere for den ekstra lasten, men innspolingshastigheten holder seg rundt 20 mm/s.

6.3 Hastighetsregulering eksperiment

Figur 6.17 viser parameterene hentet fra frekvensomformeren under forsøket.



Figur 6.17: Resultatet fra hastighetsregulerings kompenseringstesten, $K_p = 0.05$ og $K_i = 0.1$, Grønn=1.5 kg og Oransje=8 kg

Ved 25 sekunder ble lasten økt til 8 kg, og det er tydelig på alle parameterene (alle delfigurene) at systemet prøver å kompensere for den ekstra lasten.

`Motor speed`, vist i delfigur c i figur 6.17, følger kurven den skal. Som forklart i delkapittel 6.3.5 må motorhastigheten synke på vei opp og minke på vei ned for å opprettholde konstant innspolingshastighet. Samme reguleringsavvik som man ser i delfigur a i figur 6.17 skjer også i `Motor Speed`.

I delfigur d ser man at `Output current` får en økning når ekstra last blir tilført på vei opp, rundt 25 s. Når lasten blir heist opp er det en betydelig økning i `Output current`, dette er fordi det krever mer strøm for å opprettholde samme hastighet. I det hold-knappen blir benyttet minker strømmen til litt over 1 A. På vei ned blir `Output current` kun påvirket ved lastendring rundt 80 s, 100 s og 125 s. Dette er

fordi gravitasjonskraften hjelper lasten på veien ned og systemet krever ikke like mye strøm.

`Torque Reference` (delfigur e) har også en tydelig økning hver gang det blir tilført ekstra last, fordi den er basert på utgangsstrømmen.

Delfigur f i figur 6.17 viser `Output Frequency`. I likhet med `Output current` og `Torque reference`, øker også `Output Frequency` når ekstra last blir tilført, og synker når lasten blir redusert. Dette er fordi pådraget må kompensere for lasten.

Kompenseringstesten beviser at hastighetsregulatoren utviklet har relativt god evne til å kompensere for lastendring. Den opprettholder ønsket innspolingshastighet selv med lastøkning og lastreduksjon.

Kapittel 7

Diskusjon

Oppgaven som ble gitt av Omron kom med krav om å programmere i Sysmac Studio. Men det var ingen krav om koden skulle være i ladder eller i ST (strukturert tekst). Vi valgte å skrive koden i både ladder og ST. Sysmac Studio er en ny programvare som vi ikke hadde kjennskap til. Fra faget ELE310 Styringsteknikk fikk vi kunnskap om PLS og PLS programmering. I det faget brukte man programvaren CX-Programmer. Grunnen til at vi ikke brukte CX-Programmer var at den er utdatert og mange bedrifter har skiftet CX-Programmer med Sysmac Studio.

Utstyret brukt i oppgaven var også ett problem. Den første motoren ble ødelagt etter bare to uker med bruk. Ny motor var nødvendig, men den måtte på verkstedet for å montere enkoderen. På grunn av koronasituasjonen tok verkstedet ekstra lang tid til å bli ferdig. Det var meningen at motor 2 skulle bare være midlertidig løsning, frem til vi fikk tildelt motor 3. Motor 3 måtte også på verkstedet, som tok sin tid. Hvis vi hadde visst at motor 2 egnet seg bedre for oppgaven, hadde vi ikke ventet på motor 3, men uheldigvis fant vi ikke ut av dette før tester på vinsjen ble utført på motor 3. Det ble oppdaget at motor 3 var alt for sterk til å gi store responser ved lastendring. Derfor ble motor 2 brukt under resten av prosjektet.

Da motoren endelig var på plass, og forsøk på å styre motoren via PLS skulle bli utført, oppstod det enda ett problem. EtherCAT-kortet koblet til frekvensomformeren fungerte ikke, og derfor var det ingen kommunikasjon mellom PLS og frekvensomformer. Ett nytt kort ble bestilt, men det ankom ikke før etter påskeferien. Dette medbrakte at systemtester på vinsjen ikke ble utført før seint inn i prosjektet. Hoveddelen av prosjektet, regulering av vinsj, måtte nå bli gjort på en måned.

Når tester på vinsjen ble utført oppstod det et problem, det var ikke tilstrekkelig høyde til å gjøre tester med. Derfor måtte trinser bli benyttet for å gi plass til å regulere. Dette påvirket den matematiske modellen av vinsjen som hadde blitt laget. Det viste seg at det var et forhold mellom høyden over bakken og innspolt vaier på 1:3, på grunn av de tre trinsene. Dette tok vi ikke hensyn til på starten siden trinser var ikke i planen. Derfor trenger også alle andre som skal bruke koden vår på nytt ta hensyn til dette forholdet, og redigere koden hvis de ikke bruker trinser.

PLSen og frekvensomformeren hadde forskjellige loggetider. Dette gjorde at grafene hvor PLS-data og frekvensomformer-data var i samme figur ikke var like. De ble justert for at tiden skulle være lik, men det var ikke alltid mulig å oppnå gode resultater. Derfor er noen av grafene ikke helt riktig tidsmessig.

Mot slutten av prosjektet oppdaget vi feilen mellom parameteren `Encoder Pulse Counter` og virkelige enkoder pulser. Telleren `Encoder Pulse Counter` resettes etter 65536 pulser, som er en fast verdi vi ikke har konroll over. Enkoderen har 2000 ppr, som betyr at åtte omdreininger gir 64000 pulser. `Encoder Pulse Counter` resettpunktet er derfor litt over åtte omdreininger, men har ikke blitt kompensert for i funksjonene. Det hadde ikke så stor betydning for prosjektet men vi er usikre på når denne feilen er merkbar. Testene viser ingen merkbare feil i teach funksjonen og regulatorene utviklet, derfor var det vanskelig å oppdage denne feilen før utstyret ble levert tilbake. Siden `Encoder Pulse Counter` var den eneste enkoder parameteren tilgjengelig for oss, er det uklart hvordan denne feilen kunne blitt ungått i prosjektet.

7.1 Videreutvikling

Hvis mer tid hadde vært tilgjengelig for oppgaven er det flere funksjoner som kunne blitt implementert eller forbedret.

For å bedre finne K_p og K_i for PI-regulering for posisjonsregulering og hastighetsregulering kunne blitt gjort av å bruke Skogstads metode for dynamiske systemer. Dette var ikke noe vi fikk tid til, men er mulig å gjøre.

For hastighetsregulering er det ønskelig at vinsjen stopper og holder ved en viss maks høyde. Dette ble ikke gjort på grunn av mangel på tid. Videre er ikke filteret til hastighetsreguleringen optimalt. Ved videreutvikling av denne blir ikke hastigheten så støyete, og blir jevnere.

7.1 Videreutvikling

Kompensering for forskjellen mellom `Encoder Pulse Counter` resettverdien og virkelige pulser fra enkoderen er ønskelig. Dersom Z-signalet var tilgjengelig er det mulig å se når en ny omdreining ble fulført. Hvis Z-signalet ikke blir tilgjengelig for videreutvikling må parameteren `Encoder Pulse Counter` fortsette å bli brukt, men matten bak figur 4.4 blir mer kompleks og koden mer omfattende. En lettere form for kompensering er ved å bruke en enkoder med 2048 ppr, som gjør at virkelige pulser og `Encoder Pulse Counter` samsvarer bedre.

Det var ønskelig at disse funksjonene også ble implementert i systemet:

Posisjonsregulering og hastighetsregulering i samme funksjon: En funksjon som både posisjonsregulerer og hastighetsregulerer utvikler et bedre system hvor den kontrollerer både posisjonen den er i, og at den har konstant innspolingshastighet.

Bytt last: En enkel funksjon for å endre vekten på lasten. Når lasten har blitt skrevet inn tar PLSen vare på dataene, og den bruker dette til å beregne stress på vaieren for at den ikke ryker under bruk.

Bølge følging funksjon: For å kompansere mot bølger der en vinsj står på en båt, må en funksjon for dette bli laget. Den mottar sinussignaler fra en ekstern kilde, som simulerer bølgebevegelser ved havet. Programmet bruker dataene til å trygt løfte eller senke lasten mens bølgene slår mot båten.

Bedre logging av data for PLS: Loggingen av datane fra PLSen ble gjort ved en USB, men den logger hele tiden. Derfor måtte vi finne ut av hvor riktig data var ut ifra et Excel ark, basert på hvilken tid forsøket ble gjort. Det er ønskelig å ha en funksjon hvor loggingen starter manuelt.

Kapittel 8

Konklusjon

I denne oppgaven har vi regulert en vinsj med en demomodell gitt av Omron. Det ble utledet en matematisk modell av en vinsj og implementerte dette i PLS programvaren Sysmac Studio. Den matematiske modellen er mest tydelig i form av teach funksjonen. Teach funksjonen måler høyden over bakken lasten har oppnådd. Dette blir blant annet brukt til å sette en maks lengde hvor vinsjen stopper innspolingen.

En del av tiden gikk på å finne motor som var optimal for frekvensomformeren og systemet. Vi kom fram til at motor 2 var et lurt valg, fordi den klarte å utføre diverse oppgaver, samtidig viste den også gode resultater. Den var følsom for lave hastigheter og reagerte på lastendringer, som var ønskelig for å se reguleringsavviket.

Posisjonsregulering ble oppnådd med god nøyaktighet. Gode reguleringsparametere ble funnet via tester, hvor det ble konkludert at $K_p = 0.1$ fungerte best for dette systemet. Her fant vi ut at P-leddet alene var godt nok, fordi systemet er en integrerende prosess. Dermed ble det bestemt å posisjonsregulere med kun P-leddet.

Hastighetsregulering ble implementert, med forbedringspotensiale. Gode reguleringsparametere ble funnet via tester, hvor det ble konkludert at $K_p = 0.1$ og $K_i = 0.1$ med filterparameter lik 0.1 fungerte best for dette systemet. Vi fant at med kun I-leddet var systemet godt, men med en PI-regulator oppnådde systemt ønsket hastighet bedre.

Det ble utviklet en skjerm hvor man setter inn radiusen til trommelen og diameteren til vaieren. I tillegg ble det også utviklet skjerm for hver av funksjonene, hvor brukeren

hadde muligheten til å sette inn verdier og samtidig overvåke målte verdier.

Bibliografi

[1] ABB. Teknisk veiledning nr. 4 veiledning for frekvensomformere for hastighetsstyring. https://library.e.abb.com/public/06961a5060b74233c125795b002b9029/NO_Technical_guide_No.4_REVC.pdf?fbcli, 25.November 2011. Hentet:23.03.21.

[2] ABB. Teknisk veiledning nr. 7 dimensjonering av frekvensomformer og motor. https://library.e.abb.com/public/ab94567b522bad0ac125795b002cb987/NO_Technical_guide_No.7_REVC.pdf, 25.November 2011. Hentet:23.03.21.

[3] Christian Cavallo. All about induction motors - what they are and how they work. https://www.thomasnet.com/articles/machinery-tools-supplies/all-about-induction-motors-what-they-are-and-how-they-work/, 22.februar 2021. Hentet: 22. februar 2021.

[4] Danielle Collins. Faq: When are closed-loop and open-loop vector control useful? https://www.motioncontroltips.com/faq-when-are-closed-loop-and-open-loop-vector-control-useful/, 30.Juli 2016. Hentet:03.02.2021.

[5] Kevin Cope. What is a vfd? https://realpars.com/vfd/#, 29.Oktober 2018. Hentet: 01.02.21.

[6] Phillipe Lucidar. Convert revolutions per minute [rpm] to meters per second [m/s] and vice-versa. https://lucidar.me/en/unit-converter/revolutions-per-minute-to-meters-per-second/, Mai 2021. Hentet:14.05.2021.

[7] Omron. *Slim Incremental 50-mm-dia. Rotary Encoder E6C2-C*, c edition, Oktober 2015.

[8] Omron. *Q2A Technical manual*, c edition, Desember 2018.

[9] Santhana Raj. Vector control in variable frequency drives. https://sites.google.com/site/santhanarajarunachalam/my-research/vfd/vector-control-vfd, Juli 2019. Hentet:03.02.2021.

[10] Yngve Solbakken. Vector control for dummies. https://www.switchcraft.org/learning/2016/12/16/vector-control-for-dummies, 14.Mars 2017. Hentet:01.03.21.

[11] Jake Wattenphul. How does closed loop control work in a vfd? https://www.kebamerica.com/blog/how-does-closed-loop-control-work-in-a-vfd/, 15.februar 2019. Hentet: 10. februar.2021.

Vedlegg A

A.1 Oppskrift for tilkobling av frekvensomformer til PLS

For å lettere vise hvordan oppkobling mot frekvensomformeren Q2A mot PLSen er det en punktvis oppskrift laget.

- Ha riktige Esi filer i form av XML dokument plassert riktig sted:
  OS(C:)–> Programmfiler –> OMRON –> Sysmac studio –> IODevice Profiles –> EsiFiles –> UserEsiFiles
- Få kontakt mellom PCen og PLSen
- Legg til omformeren i EtherCAT
- Høyreklikk på master ikonet og velg "compare and merge"
- Nå leses nettverket inn og alle nodene er synlige, med riktig frekvensomformer vist
- Trykk "apply actual network configuration " for å overføre til prosjektet
- Endre adressen til et ledig nummer med "write slave node address"
- Frekvensomformeren må restartes
- Riktig I/O mapping må nå bli gjort, slik som beskrevet i delkapittel 2.1.1

A.2 Ved feil eller feilmeldinger

Et par feil vi fikk under kjøringen og det som løste problemet.

A.2.1 Sysmac studio gjenkjenner ikke riktig omformer som slave

Her var løsningen å putte riktige ESI-filer i form av XML-document på riktig plass: OS(C:)–> Programmfiler –> OMRON –> Sysmac studio –> IODevice Profiles –> EsiFiles –> UserEsiFiles

Videre måtte alle IO-filene i Sysmac studio være riktige, og det måtte ikke være sendt flere bits gjennom samme bitstrøm enn det er mulig. Hvis det ikke funker er det lurt å teste i en annen kode med bare de nødvendige I/O-tilkoblingene. Dette blir endret på ved å gå inn på:
Communication and setup –> EtherCAT –> Q2A - series –> PDO Map settings

Hvis dette ikke funker er det lurt å sjekke om PLSen klager på Ethernet kabelen. Dette gjøres ved å trykke på Troubleshooting knappen.

Selvfølgelig burde frekvensomformeren også prøvd å bli skrudd av og på. Hvis dette heller ikke funker kan det være noe galt med EtherCAT kortet koblet til frekvensomformeren. Da må dette bli byttet ut.

A.2.2 Verdier fra PLS blir ikke sendt gjennom selv med tilkobling

Dette kan være fordi PLSen prøver å sende gjennom for mange bits gjennom samme I/O map. Dette gjøres ved å endre plasseringen på de nødvendige verdiene i: Communication and setup –> EtherCAT –> Q2A - series –> PDO Map settings

Vedlegg B

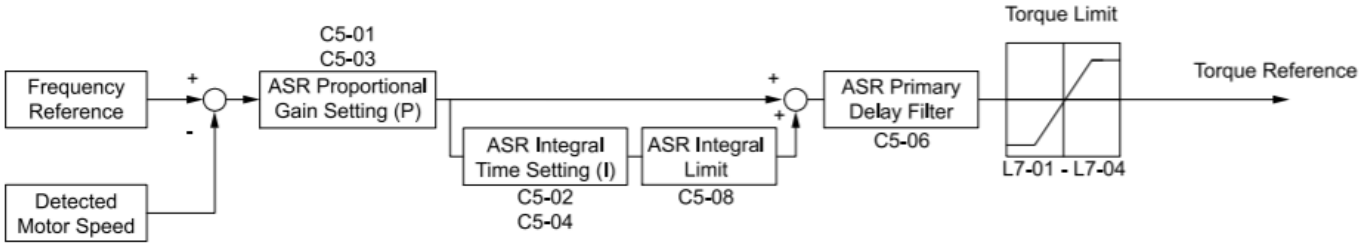
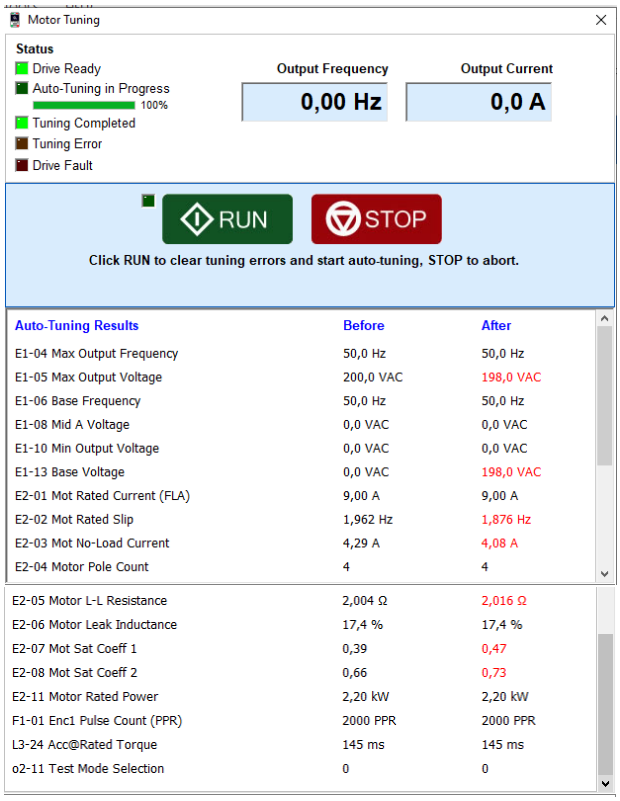


Figure 12.48 Speed Control Block Diagram for CLV, AOLV, CLV/PM, AOLV/PM, and EZOLV

Blokk diagram for speed control i closed loop tatt ut fra brukermanualen, brukt til å basere videre blokk diagram

Vedlegg C

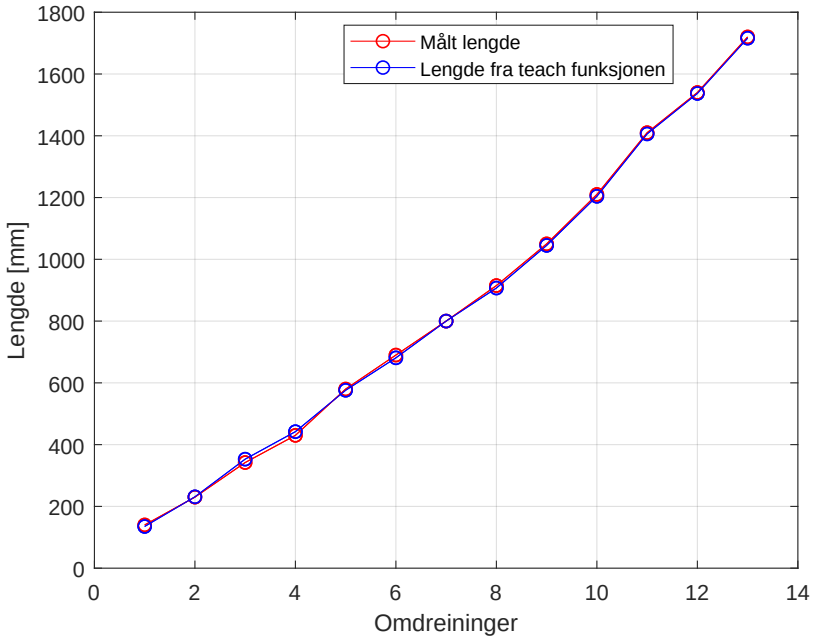


Auto-Tuning Results	Before	After
E1-04 Max Output Frequency	50,0 Hz	50,0 Hz
E1-05 Max Output Voltage	200,0 VAC	198,0 VAC
E1-06 Base Frequency	50,0 Hz	50,0 Hz
E1-08 Mid A Voltage	0,0 VAC	0,0 VAC
E1-10 Min Output Voltage	0,0 VAC	0,0 VAC
E1-13 Base Voltage	0,0 VAC	198,0 VAC
E2-01 Mot Rated Current (FLA)	9,00 A	9,00 A
E2-02 Mot Rated Slip	1,962 Hz	1,876 Hz
E2-03 Mot No-Load Current	4,29 A	4,08 A
E2-04 Motor Pole Count	4	4
E2-05 Motor L-L Resistance	2,004 Ω	2,016 Ω
E2-06 Motor Leak Inductance	17,4 %	17,4 %
E2-07 Mot Sat Coeff 1	0,39	0,47
E2-08 Mot Sat Coeff 2	0,66	0,73
E2-11 Motor Rated Power	2,20 kW	2,20 kW
F1-01 Enc1 Pulse Count (PPR)	2000 PPR	2000 PPR
L3-24 Acc@Rated Torque	145 ms	145 ms
o2-11 Test Mode Selection	0	0

Verdiene viser før og etter autotunning via Q2edit.

Vedlegg D

Resultat fra teach funksjonen



Grafene viser målt lengde og det teach funksjonen viser, et til forsøk gjort med annen last.