NTNU
Kunnskap for en bedre verden

Institutt for teknisk kybernetikk

IELET2104 - Automatiseringsprosjekt

Prosjektrapport

Regulering av væskehøyde i vanntank

Gruppe 6:
Eikeland, Adrian
Garberg, Anders
Liland, Kristian Hognestad
Rein, Trond Jakob Grø
Tran, Luan Nguyen

4. mai 2025

Innhold

Figurer

Tabeller

Forord

Denne rapporten dokumenterer et gruppeprosjekt gjennomført i emnet IELET2104 – Automatiseringsprosjekt. Rapporten er skrevet for et akademisk publikum med bakgrunn innen ingeniørfag, særlig elektro eller maskin. Det forutsettes ikke inngående kunnskap om automatisering, men det antas at leseren har grunnleggende kjennskap til PID-regulatorer og forstår hva en PLS er. I tillegg forventes det en viss forståelse for matematikk i Laplace-domenet og sammenhengen mellom dette og tidsdomenet.

Rapportens struktur er valgt for å gjenspeile oppbygningen av den endelige løsningen. Før selve hovedkapitlene presenteres en teoridel som gir nødvendig faglig bakgrunn og forklarer sentrale begreper og prinsipper brukt i prosjektet. Deretter følger hoveddelen, hvor hvert av de sentrale elementene i løsningen behandles i egne kapitler. Dette inkluderer kommunikasjonsoppsettet, brukergrensesnittet, regulatoren og alarmsystemet. Etter hoveddelen følger en diskusjonsdel der valg av metoder vurderes, potensielle forbedringer drøftes, og ulike kilder til usikkerhet identifiseres. Til slutt vurderes det i hvilken grad prosjektet har oppfylt de gitte kravene og målsettingene, før det konkluderes med om det ferdige produktet lever opp til spesifikasjonene.

1 Introduksjon AG

I moderne industri benyttes automatisering i stor utstrekning for å sikre effektive og stabile prosesser. Som en del av dette er nivåregulering av væske i tanker en sentral oppgave i prosessindustrien, blant annet innen vannbehandling og kjemisk produksjon. På bakgrunn av dette behovet er et prosjekt blitt bestilt av en kunde, hvor det kreves utvikling, programmering og driftsetting av en reguleringsløsning for en tank. Prosjektet ble løst i henhold til spesifikasjoner fastsatt av kunden, med mål om å etablere et funksjonelt og pålitelig automatiseringssystem.

1.1 Problemstilling AG

Prosjektet går ut på å utvikle og implementere en helhetlig automasjonsløsning for nivåregulering i en tankprosess. Løsningen skal kombinere avansert PLS-programmering, robust kommunikasjon mellom master- og slaveenheter, samt integrerte brukergrensesnitt; både lokalt på operatørpanelet og via PC-basert SCADA. Målet er å oppnå stabil, nøyaktig og sikker regulering, samtidig som systemets tilstand overvåkes og kontrolleres fullstendig.

Systemet skal inneholde et fleksibelt reguleringsoppsett med støtte for både ventilstyring og frekvensomformer, og det skal implementeres et fullverdig alarmsystem, historikk/trend-visning og funksjoner for feilhåndtering. Brukergrensesnittene skal skille mellom operatørnivå og avansert systemtilgang, samt utformes etter prinsippene for høy ytelse (High Performance HMI). Funksjonsblokker for PID og lead-lag skal utvikles med fokus på modulær og gjenbrukbar struktur.

Ettersom industri ofte består av større systemer, er det også ønskelig at den endelige løsningen har god skalerbarhet. I søken etter en ekstra utfordring settes det derfor som mål at den helhetlige løsningen tillater skalering innenfor gitte rammer. Dette tydeliggjøres i vedlegg A om målsettinger. Resultatet av dette vil gjøre produktet mer passende for flere forskjellige anvendelser.

1.2 Målsetning AG

Prosjektets målsetninger tar utgangspunkt i systemkrav og -spesifikasjoner fra kunde og har som hensikt å sikre at den ferdige løsningen både oppfylle de tekniske kravene og er praktisk anvendbar. I starten av prosjektperioden ble det derfor utarbeidet en konkret og detaljert målliste som et styringsverktøy for utviklingsarbeidet. Målene fungerte som et sikkerhetsnett gjennom hele prosjektforløpet og la grunnlaget for tekniske prioriteringer, teststrategier og evaluering av sluttproduktet. Den fullstendige oversikten over prosjektmålene finnes i vedlegg A.

1.3 Avgrensinger KL

For å unngå misforståelser rundt hva prosjektet skal oppnå er det nødvendig å avklare hvilke begrensninger som legges til grunn. Denne seksjonen vil ta for seg hva prosjektet ikke skal oppnå.

Et poeng som er viktig å få frem er at selv om kommunikasjonen skal være robust, er det ikke satt noe mål for håndtering av tap av kommunikasjon. Dette skyldes at kommunikasjonsprotokollene i bruk håndterer dette selv ved normal drift, og at tap som skyldes fjerning av kommunikasjonsmediumet anses som usannsynlig.

Det må også tydeliggjøres at selv om det er satt som selvstendig mål i prosjektet at løsningen skal være skalerbar, betyr ikke dette at det skal implementeres noe skalert system. Løsningen skal bare tilby muligheten for sømløs utvidelse av systemet.

Senere i rapporten vil det komme frem at det er implementert både P, PD, PI og PID-regulatorer. Dette er gjort for å holde muligheten åpen for at en teknisk operator skal kunne benytte seg av forskjellige regulatorer. Det er likevel nødvendig å poengtere at det bare er én regulator som vil være optimal for driften, og bare den valgte regulatoren som kan forventes å oppfylle alle målsettingene.

2 Bakgrunnsteori

For å kunne utvikle et funksjonelt og pålitelig automasjonssystem er det nødvendig med en god forståelse av de teoretiske prinsippene som ligger til grunn. Dette kapittelet gir en oversikt over relevant teori benyttet i prosjektet.

2.1 Teoretisk grunnlag for regulering AG

PID-regulatoren er en mye brukt metode for regulering av dynamiske systemer. Den beregner pådraget $u(t)$ basert på avviket $e(t) = r(t) - y(t)$ som differansen mellom referanse og målt prosessverdi. I kontinuerlig tid uttrykkes PID-regulatoren som:

$$u(t) = u_{nom} + K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{de(t)}{dt} \tag{1}$$

hvor K_p, K_i og K_d er forsterkninger for henholdsvis det proporsjonale, integrerende og deriverende leddet. u_{nom} er eventuelt nominelt pådrag for å fjerne stasjonært avvik ved fravær av integralledd. [1, s.103-104]

2.1.1 PI-D regulator AG

En vanlig utfordring ved bruk av derivatledd i PID-regulatorer er såkalt "derivative kick", som oppstår ved brå endringer i referanseverdien. For å unngå dette kan man anta referansen som konstant og derav utelate referansen fra derivatleddet, slik at det kun virker på prosessverdien. Dette reduserer unødvendige rykk og bidrar til økt robusthet, samt redusert slitasje på pådragsorganet. En slik struktur kalles en PI-D-regulator, og har følgende overføringsfunksjon: [1, s.107]

$$U(s) = K\left(\left(1 + \frac{1}{T_I s}\right) E(s) - T_d s Y(s)\right) \tag{2}$$

2.1.2 Derivatfilter AG

D-leddet forsterker høyfrekvent støy. Et førsteordens lavpassfilter benyttes for å gjøre dette leddet realiserbart og mindre støyfølsomt: [1, s.108]

$$U_D(s) = \frac{T_d s}{T_f s + 1} \tag{3}$$

$T_f = \frac{T_d}{N}$ der T_d er derivattiden og N er høypassforholdet.

2.1.3 Parallell og ideell form

AG

To vanlige representasjoner av regulatoren i Laplace-domenet er:

- **Parallell form:**

$$U(s) = \left(K_p + \frac{K_i}{s} + K_d \frac{Ns}{s+N}\right) E(s) \tag{4}$$

- **Ideell form:**

$$U(s) = K\left(1 + \frac{1}{T_i s} + \frac{T_d s}{T_f s + 1}\right) E(s) \tag{5}$$

Der K er forsterkning, $T_i = K_p/K_i$, $T_d = K_d/K_p$ og T_f (og N) er en filterparameter som brukes i D-leddet. [1, s. 106-107] Ideell form passer generelt sett bedre til implementasjon da den gir oversiktlig kode med tydelig kobling til systemets fysiske tidsforløp. Parallell form er mer intuitivt da man lettere ser bidraget fra hver forsterkning.

2.1.4 Tracking og rykkfri overgang

AE

For å sikre en rykkfri overgang mellom ulike regulatorer, pådragsorgan og moduser, benyttes *tracking*. Ved bytte av regulator eller pådragsorgan initialiseres regulatorens interne tilstander basert på et forhold mellom parameterne i den gamle og den nye regulatoren. Dette gjelder særlig integratorleddet, som er rekursivt og ellers ville brukt tid på å tilpasse seg.

Ved å holde integratorens indre tilstand synkronisert med det faktiske pådraget, kan regulatoren overta styringen umiddelbart, uten å føre systemet inn i en *kvekkig* tilstand[1] som introdusere uønskede transienter i systemet. [1, s.156]

2.1.5 Diskretisering

AG

PID-regulatoren i dette prosjektet skal implementeres digitalt og er derfor diskretisert med eksplisitte differenslikninger for hver del. Dette muliggjør praktisk bruk på en PLS-plattform med begrenset prosesseringskapasitet.

- **P-ledd:** Proporsjonalleddet implementeres direkte som:

$$U_P[k] = K \cdot e[k] \tag{6}$$

- **I-ledd:** Integrasjonen skjer via akkumulert sum med anti-windup:

$$U_I[k] = U_I[k-1] + K \cdot \frac{T_s}{T_i} \cdot e[k] + \frac{T_s}{T_t}(U_{\text{sat}}[k-1] - U[k-1]) \tag{7}$$

  hvor T_s er samplingstid, T_i er integraltid, og T_t er trackingtid. Det siste leddet gir anti-windup-beskyttelse ved metning hvor U_{sat} er begrenset pådrag brukt på det fysiske pådraget, og U er ønsket pådrag fra regulatoren.

- **D-ledd:** Derivatleddet blir realisert med et rekrusivt filtreringsledd basert på $\beta = \frac{T_d}{T_d + T_s \cdot N}$, der N er høypassforholdet, og benytter kun endringen i målt prosessverdi:

$$U_D[k] = \beta \cdot U_D[k-1] - K \cdot \frac{T_d}{T_s} \cdot (1-\beta) \cdot (y[k] - y[k-1]) \tag{8}$$

  med filterfaktor $\beta \in [0, 1)$

[1] Begrepet *kvekkig* viser til en tilstand som ikke er *stedig*, altså ikke-periodisk eller ujevn i sin utvikling. Se definisjoner i [2, begrep 16 og 15]

2.1.6 Lead-lag-foroverkobling med Tustin-diskretisering AE

Foroverkoblingen fra en målt forstyrrelse realiseres som en diskret lead-lag-blokk med bilineær (Tustin) approksimasjon. Den kontinuerlige strukturen:

$$F(s) = \frac{T_1 s + 1}{T_2 s + 1} \tag{9}$$

diskretiseres med substitusjonen $s \approx \frac{2}{T_s} \cdot \frac{z-1}{z+1}$. Tustins metode har fordelaktige egenskaper som stabilitetsbevaring og en unik frekvensavbildning. [3]

Denne diskretiseringen gir en differenslikning på formen:

$$U_{\text{ff}}[k] = a_1 U_{\text{ff}}[k-1] + b_0 d[k] + b_1 d[k-1] \tag{10}$$

hvor:

$$a_1 = \frac{2T_2 - T_s}{2T_2 + T_s}, \quad b_0 = \frac{2T_1 + T_s}{2T_2 + T_s} \cdot P_{ff}, \quad b_1 = \frac{T_s - 2T_1}{2T_2 + T_s} \cdot P_{ff}$$

og P_{ff} er foroverkoblingsforsterkning. Dette gir en effektiv og stabil foroverkobling som reagerer presist og kompenserer raskt for forstyrrelser.

2.1.7 Foroverkopling og ratebegrensning AE

Lead-lag-kompensasjon brukes ofte for å oppnå ønsket dynamisk respons i reguleringssystemer ved å gi både forberedelse (lead) og demping (lag). Brå settpunktsendringer kan føre til uønskede transienter og redusert regulatorytelse. Derfor bør settpunktsendringer være glatte.

En ratebegrensning kan implementeres basert på et 5.-ordens polynom, som sikrer en jevn og kontinuerlig overgang mellom settpunktene. Det matematiske kravet til polynomet er:

$$r(t) = \begin{cases} a, & t = t_0 \\ a + (b-a)s(t), & t_0 < t < t_f \\ b, & t = t_f \end{cases} \tag{11}$$

Der a og b er henholdsvis gammel og nytt referansepunkt, og $r(t)$ er alltid en verdi mellom disse. Tidsintervallet $[t_0, t_f]$ angir varigheten av referanseendringen. Løsningen som oppfyller disse kravene er gitt ved:

$$s(t) = 10\tau^3 - 15\tau^4 + 6\tau^5, \quad \tau = \frac{t - t_0}{t_f - t_0} \tag{12}$$

Metoden genererer en glatt kurve med kontinuerlig hastighet og akselerasjon, og gjør at systemet robust håndterer dynamiske endringer i settpunktene. Dette reduserer belastningen på regulatoren og forbedrer systemets stabilitet og ytelse.

2.1.8 Valg av tastetid i digitale reguleringssystemer AE

Valg av tastetid (samplingstid) er avgjørende for hvor godt et digitalt reguleringssystem kan registrere og behandle målesignaler fra et kontinuerlig system. Tastetiden bestemmer hvor ofte signaler samples og reguleringsberegninger utføres.

Ifølge *Nyquist-Shannons samplingsteorem* må samplingsfrekvensen $\omega_s = \frac{2\pi}{T}$ være minst dobbelt så høy som den høyeste frekvensen $\omega_{\max}$ i signalet for å kunne gjenskape det uten aliasing:

$$\omega_s \geq 2\omega_{\max}$$

Her er $\omega_N = \frac{\omega_s}{2}$ den såkalte Nyquist-frekvensen. I praksis velges ofte en høyere samplingsfrekvens for å gi sikkerhetsmargin mot modellavvik og støy, ofte minst 5–10 ganger raskere enn den dominerende tidskonstanten i systemet. [4]

Tastetiden må også være tilstrekkelig lang til at hele reguleringssløyfen (inkludert ADC, filtrering, beregninger og DAC) kan fullføres innen én syklus. Dette er viktig for å sikre deterministisk oppførsel og unngå forsinkelser mellom måling og respons.

Et riktig valgt samplingstidspunkt gir dermed et mer robust, nøyaktig og forutsigbart reguleringssystem, samtidig som det tar hensyn til både systemets dynamikk og tilgjengelige prosesseringsressurser.

2.2 Kommunikasjonsprotokoller AE

- **PROFIBUS DP** er en deterministisk feltbuss der masteren har full kontroll over når og hvilke slaver som får sende. Dette sikrer forutsigbar overføring av sanntidskritiske signaler, men innebærer også en fast begrensning i antall kanaler og datamengde per syklus. I dette prosjektet er denne begrensningen løst med en tilpasset overføringsprotokoll for mindre tidssensitive data, samtidig som målinger og styringssignaler oppdateres direkte i hver syklus.
- **Ethernet/IP** brukes til overføring av styringskommandoer, parameterendringer og statusmeldinger mellom PC/HMI og master. Kommunikasjonen går over standard IP-nettverk og tillater fleksibel og rask dataflyt. I dette prosjektet benyttes OPC UA som protokoll på toppen, noe som gir strukturert og pålitelig kommunikasjon.

2.3 Modellering AE

Whitebox og blackbox-modeller

Ettersom systemets simulerte miljø ble modellert gjennom systemidentifikasjon, er det hensiktsmessig å introdusere begrepene *whitebox*- og *blackbox*-modeller.

Whitebox-modeller er basert på kjente fysiske lover og en dyp forståelse av systemets struktur. De beskriver systemets oppførsel gjennom analytiske ligninger, noe som gir høy transparens og mulighet for å tolke modellens indre mekanismer. Denne tilnærmingen er nyttig når man har detaljert kunnskap om systemets interne prosesser og ønsker en modell som er lett å tolke og analysere.

I kontrast baserer *blackbox*-modeller seg utelukkende på observerte data, uten eksplisitt kunnskap om systemets indre struktur. Slike modeller, som ofte benytter metoder som NLARX [5], kan fange opp kompleks og ikke-lineær dynamikk, men gir begrenset innsikt i underliggende årsakssammenhenger. Denne tilnærmingen er spesielt nyttig når systemets indre mekanismer er ukjente eller for komplekse til å modellere direkte.

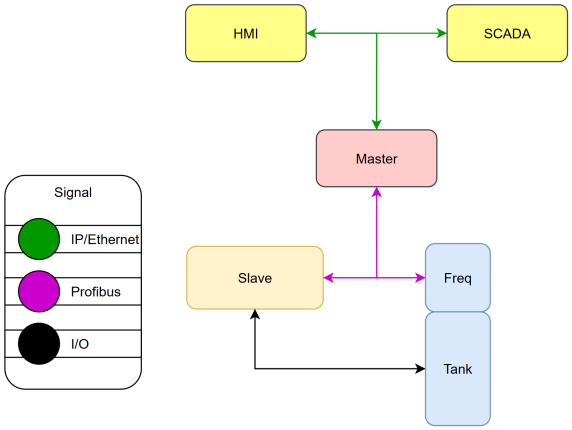
3 Kommunikasjonsoppsett AE

Kommunikasjonsarkitekturen er utformet for å sikre synkron og pålitelig samhandling mellom alle deler av systemet. Master-PLS fungerer som et sentralt knutepunkt (en HUB) og sørger for koordinert tilstand og kommunikasjonsflyt gjennom hele anlegget. Den styrer dataflyt mellom feltbuss (PROFIBUS), IP-nettverk og de ulike enhetene i systemet. Dette sikrer at alle komponenter opererer ut fra en felles tilstandsforståelse, selv om styringssløyfene er distribuerte og logikken er lokalisert i slave-PLSen.

For å oppnå dette er det implementert deterministisk utveksling av tilstandsdata og kvitteringer, og kommunikasjonen er tilpasset variasjoner i scantid. Arkitekturen gir systemet robusthet, skalerbarhet og god oversikt – hvor alle enheter, fra HMI til slaven, arbeider med samme datagrunnlag.

Systemet bruker tre hovedkanaler for kommunikasjon. Slaven og frekvensomformeren er begge PROFIBUS-slaver og kommuniserer kun med masteren. PC og HMI kommuniserer med masteren via Ethernet/IP, men ikke direkte med hverandre. All lokal I/O, som styring av ventiler og måling av nivå og utstrømning, håndteres direkte av slaven uten behov for involvering fra masteren.

Figur 1 viser hvordan datakommunikasjonen er strukturert i systemet.

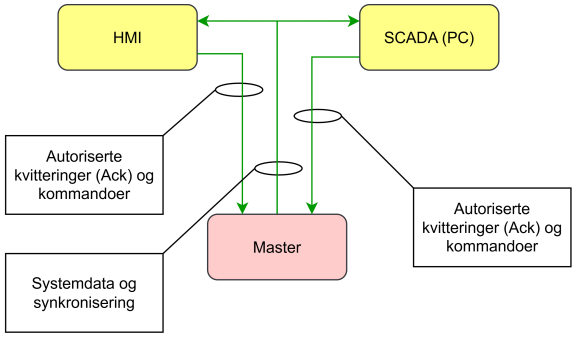


Figur 1: Systemoversikt over hovedkanalane for datakommunikasjon.

3.1 Ethernet-baserte forespørsler og tilstandssynkronisering AE

Over Ethernet-kanalen håndteres autoriserte forespørsler fra brukere via PC og HMI. Dette inkluderer blant annet styringskommandoer, parameterforespørsler og kvitteringer. All slik kommunikasjon går via masteren, som sørger for at data valideres, synkroniseres og videresendes til riktige mottakere i rett rekkefølge.

Denne arkitekturen bidrar til at systemet kan opprettholde en konsistent og koordinert tilstand, selv om flere brukere samhandler samtidig. Figur 2 viser flyten av Ethernet-basert kommunikasjon inn til masteren og hvordan denne synkroniserer dataflyten videre.



Figur 2: Oversikt over Ethernet-basert kommunikasjon mellom HMI/PC og master

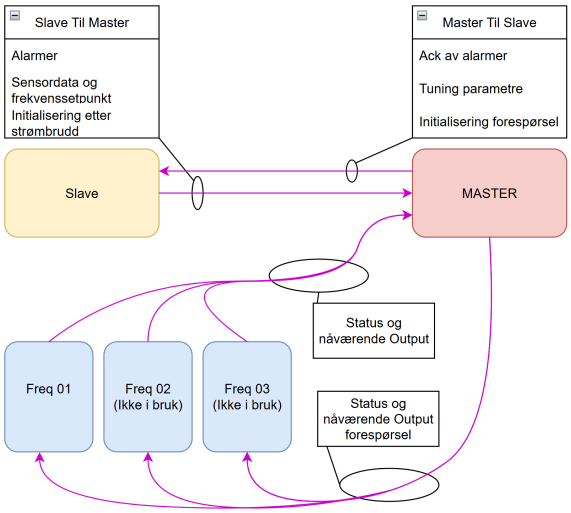
3.2 PROFIBUS-topologi og distribusjon av styringsansvar AE

Frekvensomformeren styres via to dedikerte register i masteren koblet opp mot PROFIBUS-modulen. På grunn av at PROFIBUS er en master-slave-protokoll, er det kun masteren som kan sende kommandoer direkte til omformeren. Regulatorlogikken ligger derimot i slaven, som beregner ønsket pådrag.

Så lenge det ikke er aktivert nødstopp eller andre kritiske avvik, videresender masteren dette pådraget til frekvensomformeren uten å påvirke beregningene. På denne måten abstraheres styringsansvaret bort fra masteren, og mest mulig av logikk og programkode lokaliseres i slaven. Dette

bidrar til skalerbarhet og modularitet i systemet, og frigjør ressurser i masteren til å fokusere på synkronisering og ruting av datatrafikk.

Figur 3 viser hvordan dataflyten over PROFIBUS mellom slave, master og frekvensomformer er organisert, og illustrerer rollene til de ulike komponentene i kommunikasjonen. Figuren viser også hvordan systemet er designet med tanke på skalerbarhet – dersom anlegget skulle få flere tanker med egne frekvensomformere, kan disse enkelt kobles til og styres gjennom samme arkitektur, uten behov for større endringer i logikk eller struktur.



Figur 3: Illustrasjon av PROFIBUS-topologi mellom master, slave og frekvensomformer

3.3 Kommunikasjon fra slaven AE

Den største utfordringen rundt prosjektets målsettinger for skalerbarhet er at kommunikasjonen over feltbuss-PROFIBUS mellom slaver og master har begrenset kapasitet. Dette gjør at den implementerte strukturen for meldinger fra slaver er mer kompleks enn den kunne vært uten dette målet.

Strukturen for meldinger fra slave til master består av 20 PROFIBUS-kanaler av typen *16-bit Word (Signed)*. Disse kanalene er satt opp i et array i slaven, hvor hver kanal har fått tildelt dedikerte oppgaver som vist i Tabell 1 og 2.

3.4 Protokoll for robust overføring av ikke-tidssensitiv data AE

Hver tank, samt fellesalarmer, har fått tildelt en protokollkanal. Disse kanalene er utviklet for å overføre mye data som ikke er tidssensitive på millisekund-nivå, men som likevel må sikres mot tap.

3.4.1 Motivasjon for egen protokoll

Begrenset kapasitet i PROFIBUS-kommunikasjonen krevde en løsning som kunne overføre et stort antall parametere uten å bruke én egen kanal for hver verdi. Et system som baserte seg på direkte sending av hver enkelt parameter ville raskt ha overbelastet de tilgjengelige 20 kanalene per slave, og gjort det umulig å oppnå prosjektmålet for skalerbarhet.

For å møte denne utfordringen ble det utviklet en egen protokoll basert på sending av adresse og data som separate meldinger. Innføringen av bitmanipulering, adresseringslogikk og dekoding medførte noe økt kompleksitet i implementasjonen, men ble vurdert som nødvendig for å frigjøre de faste PROFIBUS-kanalene til sanntidskritisk kommunikasjon. Dette ville gjøre det mulig å dele

Tilhører	Kanal	Brukes til
Alarmer	0	Protokoll
Tank 1	1	Protokoll
	2	Nivå
	3	Pådrag frekvens
	4	Pådrag ventil
	5	Ubrukt
Tank 2	6	Protokoll
	7	Nivå
	8	Pådrag frekvens
	9	Pådrag ventil
	10	Ubrukt
Tank 3	11	Protokoll
	12	Nivå
	13	Pådrag frekvens
	14	Pådrag ventil
	15	Ubrukt
Felles	16	Ubrukt
	17	Ubrukt
	18	Nødstopp
	19	Strømbrudd sekvens

Tabell 1: Kanaler fra slave til master

Tilhører	Kanal	Brukes til
Alarmer	0	Protokoll
Tank 1	1	Protokoll
	2	Ubrukt
	3	Ubrukt
	4	Ubrukt
	5	Ubrukt
Tank 2	6	Protokoll
	7	Ubrukt
	8	Ubrukt
	9	Ubrukt
	10	Ubrukt
Tank 3	11	Protokoll
	12	Ubrukt
	13	Ubrukt
	14	Ubrukt
	15	Ubrukt
Felles	16	Ubrukt
	17	Ubrukt
	18	Nødstopp
	19	Strømbrudd sekvens

Tabell 2: Kanaler fra master til slave

store mengder parametere uten endringer i fysisk nettverksstruktur og sikre et robust og skalerbart system som tåler fremtidige utvidelser.

På grunn av systemets rolle i overføring av alarmsignaler og kritiske parametere ble det gjennomført omfattende testing og kvalitetssikring for å verifisere robustheten. Testene viste at alle meldinger blir levert korrekt under både normale og belastende driftsforhold.

3.4.2 Prinsipper for adressering og datatype

Datamelding og adressemelding skilles ved å kontrollere om mest signifikante bit (MSB) er satt høyt. Dette sikrer at alle adresser er unike, og at data og adresse ikke kan forveksles, selv om numeriske verdier skulle være like.

Datatypene differensieres basert på adressenummer:

- Adresser fra 0 til 99 er reservert for boolske verdier.
- Adresser fra 100 til $2^{14} - 1$ benyttes for signerte 15-bits heltallsverdier, som dekker området fra omtrent $-16\,000$ til $+16\,000$.

Selv om det teoretisk kunne vært mulig å sende opptil $2^{14} - 1$ unike adresser, er protokollen praktisk begrenset til 40 boolske meldinger og 40 word-meldinger. Begrensningen ble innført for å unngå overfylling av buffer under normal drift, samt sikre at overføringstiden etter oppstart eller spenningsbortfall holder seg innen akseptable grenser. Dette ville også holde CPU-belastningen for håndtering av protokollen på et fornuftig nivå.

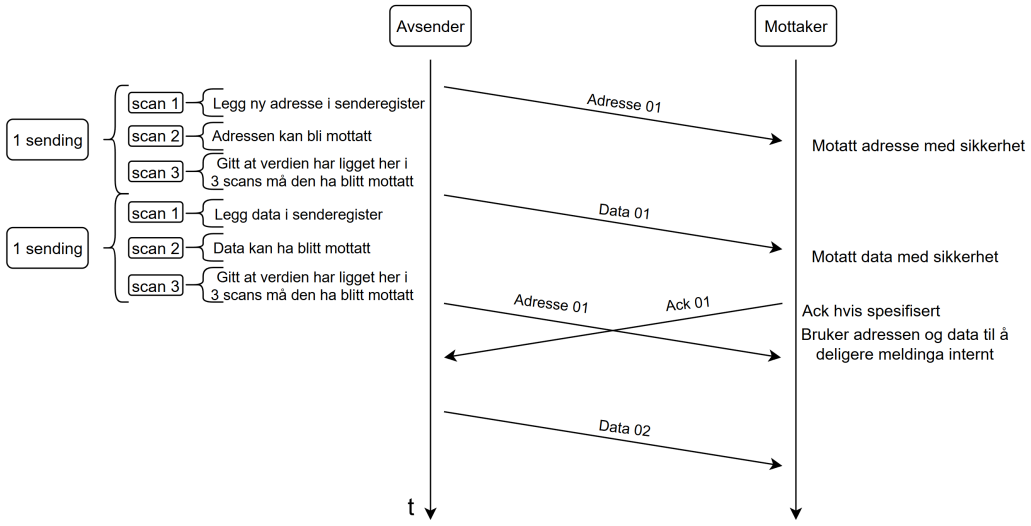
3.4.3 Flyt og synkronisering av overføringer

Protokollen er utformet for å utnytte PLS-ens scantider ved å plassere meldinger i dedikerte minneceller som kontinuerlig overføres via feltbussen. En digital verdi må være tilgjengelig i minst to påfølgende scansykluser for å garantere at en mottaker kan lese den. På grunn av at scansyklusene

til PLS-ene ikke er synkroniserte, blir det stilt krav om tre scansykluser basert på den tregeste PLSen for å være sikker på at verdien blir fanget opp.

Denne levetiden sikrer at selv den tregeste PLS-en i nettverket rekker å lese verdien innen den overskrives. Forutsetningen er at feltbussen, PROFIBUS, fungerer korrekt. Dersom dette ikke er tilfelle, er det implementert et kvitteringsystem for kritiske datapunkter for å sikre pålitelig overføring.

Dataflyten er illustrert i Figur 4, og viser at det krever seks påfølgende scansykluser per melding, ettersom datapakken består av to separate overføringer: først en adresse, deretter selve dataverdien. Dette er nødvendig for å sikre at verdien når frem og tolkes korrekt av mottakeren som skal delegere dataen videre internt.



Figur 4: Flyten i protokollen

Figur 4 kan tolkes slik:

1. På avsendersiden sendes adressefeltet først, deretter datafeltet.
2. Mottaker leser adressefeltet så snart det blir oppdatert.
3. Når datafeltet er mottatt og validert, utføres ruting til korrekt destinasjon.
4. På avsendersiden vil meldingen forblir gyldig i tre scansykluser for sikker sending.

3.4.4 Automatisering av meldingshåndtering

Protokollen er utformet med et klart mål om å abstrahere bort behovet for eksplisitt databehandling i programkoden. I stedet knyttes data til forhåndsdefinerte minneceller, som overvåkes og håndteres automatisk i bakgrunnen. Denne abstraksjonen gjør at funksjonaliteten enkelt kan vedlikeholdes og utvides, uten at utvikleren trenger detaljkunnskap om protokollens interne virkemåte – så lenge riktige minneadresser benyttes.

Et dedikert program overvåker alle endringer i minnecellenes innhold. Når en endring oppdages, legges verdien i en FIFO-basert buffer og sendes fortløpende over feltbussen. Hele prosessen er skjult for brukeren og implementeres gjennom to funksjonsblokker: én for sending og én for mottak og ruting.

3.4.5 Bruksområde og begrensninger

Protokollen er ikke egnet for hyppig endrende data, da dette kan overfylle bufferen på grunn av FIFO-prinsippet. Den blir derfor brukt for parameterinnstillinger, statuser og alarmer. Testing viser at all kritisk informasjon blir korrekt overført.

Med fire forskjellige regulatortyper og to pådragsorganer sendes omkring 40 parametere, noe som hadde vært umulig uten protokollen gitt begrensningen på 20 PROFIBUS-kanaler og fem kanaler per tank. Løsningen muliggjør derfor effektiv overføring av stort datavolum, og frigjør kanaler for sanntidskritiske signaler som nivåovervåking, pådragsstyring og andre målinger.

3.5 Sanntidskrav og kanalprioritering AE

I et styringssystem der både parametere og direkte prosessignaler skal kommuniseres, er det nødvendig å prioritere de mest tidssensitive dataene. PROFIBUS-kanalene som er reservert for tanknivå, ventilpådrag og nødstopp, oppdateres og leses kontinuerlig i hver scansyklus. Disse blir aldri blokkert av annen trafikk eller protokollbasert overføring.

Protokollen for ikke-tidssensitiv data er eksplisitt utformet for å kjøre parallelt, uten å forstyrre sanntidskanalene. Dette sikrer at regulator og alarmsystem alltid har tilgang på ferske målinger og kontrollsignaler, mens parametere og meldinger som tåler noe forsinkelse (for eksempel kvitteringer, settpunktsendringer og logiske statuser) håndteres i bakgrunnen.

3.6 Gjenoppretting etter oppstart og spenningstap AE

Dersom en av enhetene mister strøm, er det et krav fra kunde at driften skal kunne fortsette uten manuell inngripen. Masteren i systemet har hverken batteribackup, eller minne som bevarer data over spenningsbortfall. Det betyr at ved omstart vil masteren miste all kunnskap om hvilke verdier som ble sendt eller mottatt. For å unngå avhengighet av eksterne systemer (som InTouch SCADA) for lagring, ble løsningen å basere gjenopprettingen på slaven. Dette fordi slaven har batterimatede minneceller og har mulighet til å lagre både konfigurasjonsparametere og driftsverdier over spenningsbortfall.

Systemet er konstruert slik at både master og slave kan initiere en synkroniseringssekvens etter oppstart eller spenningsbortfall. Dersom en av partene oppdager at motpartens statuskode er endret eller nullstilt, kan sekvensen startes for å gjenopprette en synkron systemtilstand. Det stilles ingen krav til hvilken enhet som starter prosessen først, men begge må eksplisitt godkjenne synkroniseringen før normal drift kan gjenopptas.

Synkroniseringen skjer over den etablerte protokollen for ikke-tidssensitiv data, som beskrevet tidligere. Dette innebærer at alle relevante parametere må sendes og mottas via de samme mekanismene som vanlig datakommunikasjon, og prosessen krever derfor en viss tid basert på scansyklustider og tilgjengelig bufferkapasitet. Denne tilnærmingen sikrer robusthet mot asynkrone oppstarter og ulik opplastningstid mellom enhetene, samtidig som konsistens mellom master og slave blir garantert før normal drift tillates.

1. Oppstart og statusmelding Etter oppstart (enten etter strømbrudd eller kodelasting) sender masteren en spesifikk statuskode på kanal 19, som er reservert for synkronisering. Denne koden signaliserer at masteren er klar til å motta data.

2. Slavens overføring av lagrede verdier Når slaven mottar klar-signal fra masteren lagrer den sin normale protokollbuffer midlertidig. Deretter starter slaven overføring av de verdiene som tidligere var mottatt fra masteren.

3. Mottak og lagring hos master Masteren mottar de overførte verdiene fra slaven og skriver dem til de respektive kanalene som benyttes i protokollen for normal kommunikasjon.

4. Validering av synkronisering Når alle verdier er overført sender slaven en ny statusmelding på kanal 19 som signaliserer at sendingen er ferdig. Masteren leser så verdiene, skriver dem tilbake over protokollen og sender dem til slaven. Til slutt validerer slaven at de mottatte verdiene stemmer overens med det som ble sendt.

5. Avslutning av gjenopprettingssekvens Etter at begge parter har verifisert at synkroniseringen er korrekt, utveksles en siste bekreftelse via kanal 19, og systemet går over i normal drift.

Begrensninger under initialisering

Under gjenopprettingssekvensen er normal drift midlertidig blokkert. Normale parameterverdier kan ikke sendes og alarmoverføring fra slaven er deaktivert. Dette gjelder også stoppkommandoer fra master, mens nødstopp alltid er tilgjengelig for sikkerhetskritiske situasjoner. Dette er vurdert som et akseptabelt kompromiss mellom sikkerhet og behovet for å få systemet raskt tilbake i normal drift. Gjenopprettingssekvensen tar noe tid, avhengig av antall parametere som må utveksles, slik som diskutert i seksjon 3.4.3.

3.7 Oppsummering av kommunikasjonsstrategi

Kommunikasjonsoppsettet kombinerer rask overføring for sanntidskritiske signaler, fleksibel distribusjon av parameterdata, og robust gjenoppretting etter feiltilstander. Ved å sentralisere synkronisering i masteren, men delegere logikk og minne til slaven, har systemet oppnådd både modularitet og skalerbarhet. Begrensningar i PROFIBUS-kapasitet er løst gjennom en tilpasset protokoll, og testresultat viser stabil og korrekt overføring også ved høy belastning.

PC og HMI fungerer som operatørgrensesnitt, og gir brukeren mulighet til å overvåke og styre systemet via enhetlige og validerte forespørsler. All kommunikasjon går via masteren, som sikrer at data fra flere kilder behandles konsistent og uten konflikter.

4 Brukergrensesnitt KL

Ved design av brukergrensesnitt er det alltid viktig at det oppleves intuitivt. Dette gjelder spesielt på industrielle anlegg hvor det ved mangel på slike hensyn kan oppleves svært overveldende for en operatør. For å designe brukergrensesnitt som unngår overflod av informasjon er det derfor tatt utgangspunkt i prinsipper om High Performance i HMI-design (HPHMI) [6]. HPHMI er en form for standardisering av hvordan HMI bør designes for å appellere til menneskers intuisjon. Blant elementene HPHMI forsøker å standardisere, er fargebruk, ikonbruk, analoge illustrasjoner og og generelt grafisk design sentrale.

Fargebruk

HPHMI fremhever at det er viktig at fargebruk ikke alene tydeliggjør viktigheten av en tilstand, og at fargebruk i løsningen må være gjennomgående [6, s. 9-10]. Det settes også føringer for hvilke farger som bør brukes for forskjellige anvendelser. Tabell 3 viser hvordan dette er implementert i HMI og SCADA på produktet.

Som vist i tabellen brukes fargene rød, grønn og gul til forskjellige elementer. Det kan argumenteres for at dette strider imot HPHMI. Grunnen til at det likevel er implementert slik er fordi alarmer også tydeliggjøres ved blinking. Derfor ble det vurdert som forsvarlig å bruke grønn, gul og rød også for indikatorlampene. Gjennkjenneligheten til disse fargene, blant annet fra trafikklys, var grunnen til at dette var ønskelig.

Analoge illustrasjoner

Enten det er prosessverdier, alarmer, pådragsorganer eller andre elementer som skal presenteres, er den beste måten å bruke analoge illustrasjoner. [6, s. 7] Analoge illustrasjoner som representerer verdier eller elementer har mye større appell til intuisjon enn tallverdier og knapper. I prosjektet er dette implementert ved at verdier som kan endres er representert med glidebrytere, og komponenter

Farge	Visning	Bruk
Hvit		Inkdikerer PÅ status for ikoner, og nivå på glidebryter-verdi
Lysere grå		Markerer trykkbare knapper
Lys grå		AV status på ikoner
Grå		Bakgrunnsfarge. Bruker noen variasjoner for fremheving
Svart		Statisk tekst
Mørkeblå		Prosessverdier som ikke kan interageres med direkte.
Rød		Alarmer, kritiske verdier og AV-indikator for indikatorlamper
Gul		Ikke-kritiske alarmer og indikator for mellom-status for indikatorlamper
Grønn		PÅ-indikator for indikatorlamper

Tabell 3: Fargebruk implementert i brukergrensesnitt som følger HPHMI

som kan slås av og på er representert med ikoner som kan klikkes på. Selve tanken er også illustrert med et større ikon som kan vise både trendlinje og en representasjon av hvor full tanken er. Ikoner som er klikkbare indikeres også med en pekende hånd. De har gjennomgående fått farger som indikerer om de er på eller av.

4.1 SCADA LT

SCADA på InTouch er konfigurert med tre ulike operatørnivåer, hver med definerte lese- og skriverettigheter i henhold til kravene spesifisert i tabell fra vedlegg B. For å skille mellom operatørene, er det implementert en innloggingsside.

Den eneste synlige forskjellen mellom operatørene vises i "Control"-vinduet, mens de øvrige vinduene ("Login", "Trend", "Historical" og "Alarms and events") er felles og identiske for alle operatørnivåer.

Alle vinduene med unntak av innloggingssiden inkluderer en persistent sidemeny som brukes for å navigere mellom de ulike vinduene. Sidemenyen inneholder også en nødstopp-funksjon som er tilgjengelige for alle operatører. Tilbakestilling av nødstopp er derimot tilgangsbegrenset og kan kun utføres av operatør 3. I tillegg er det integrert en alarmindikator i form av en trekant i sidebaren som visualiserer gjeldende alarmsituasjon i systemet. Øvrige indikatorlamper som visualiserer status i forskjellige deler av systemet er runde og benytter konsekvente farger for å indikere forskjellige tilstander.

Alle interaktive objekter er merket med et pekesymbol. Objekter som inneholder tilleggsinformasjon via verktøytips er visuelt merket med et informasjonssymbol for å tydeliggjøre tilgjengelig hjelpedata. Nedtrekksmenyer er tydelig merket med et pil-ned symbol for å indikere at flere valg er tilgjengelige.

Justerbare verdier som for eksempel settpunkter kan endres ved hjelp av glidebrytere, som gir brukeren en intuitiv måte å regulere prossessparametere på. I tillegg er både justerbare verdier og knapper som utfører handlinger visuelt representert som lysere grå rektangler som gir en gjenkjennbar brukeropplevelse.

Skjermdumper fra grensesnittet finnes i vedlegg C.

4.2 Operatørpanel LT

På operatørpanelet er det ingen behov for en innloggingsskjerm ettersom alle operatørene har samme tilgang. På grunn av den begrensede størrelsen på panelet, er det lagt til et eget "Home"-vindu for å vise verdier, mens "Control"-vinduet brukes til styring. "Trend" og "Historical" er slått

sammen til ett vindu kalt “Trend”. “Alarms and events” er forenklet til kun “Alarm”.

Slik som i SCADA inkluderer panelet også en sidemeny for navigasjon mellom de tilgjengelige vinduene. Sidemenyen inneholder en nødstopp-funksjon som kan aktiveres, men ikke tilbakestilles via panelet. Her er det også integrert en alarmindikator i form av en trekant som signaliserer nåværende alarmsituasjon i systemet. Øvrige indikatorlamper er runde og bruker de samme konsekvente fargene som SCADA. På grunn av at panelet ikke har så høy oppdateringfrekvens, er blinkefrekvensen for kritisk alarmindikator 2.5Hz istedenfor 5hz.

Panelet inneholder interaktive objekter som alle er merket med pekesymbol for å indikere brukerinteraksjon. Panelet benytter ikke verktøytips og ingen objekter har tilknyttet informasjonssymbol. Glidebrytere og knapper i operatørpanelet er utformet på samme måte som på SCADA slik at funksjonaliteten er konsekvent og intuitivt på tvers av de to brukergrensesnittene.

Skjermdumper fra grensesnittet finnes i vedlegg D.

4.3 Historikk og trend KL

Trendvisning av verdier kan være svært nyttig for å analysere hvordan et system oppfører seg, og det er derfor viktig at dette implementeres på en tydelig måte. HPHMI er tydelig på at fravær av nyttige trender forårsaker tap i operatørers forståelse over situasjoner. [6, s. 16] I kundens spesifikasjon er det også etterspurt historisk visning av trender. Dette åpner for mer dyptgående analyse av systemets drift over tid.

Normal trend

Det er viktig at trender oppleves ryddige, og at de er enkle å avlese. Produktet er derfor utviklet med hensyn på hva som faktisk er nyttige verdier å presentere som trend. Det er blitt vurdert at de viktigste verdiene å representere som trend er referanseverdi og tankens nivå. Dette begrunnes ved at sammenligning mellom referanseverdi og faktisk tanknivå er svært sentralt i analyse av produktets ytelse. Derfor vises disse også i samme vindu.

Videre er det også besluttet å ha et eget trendvindu for visning av de individuelle aktuatorenes nivå. Her er det en trendlinje for frekvensomformeren og en trendlinje for magnetventilen. Disse kan være interessante å lese av, spesielt for operatør 3, som kan endre på regulatorens parametre. Operatøren kan da se hvilke effekter endringer i parametre har for pådraget.

Til slutt er det også implementert en trendlinje for utløp ut av tanken. Dette er gjort i henhold til kundens spesifikasjon.

Historisk trend

Den implementerte løsningen for historisk trendvisning er i stor grad basert på løsningene som finnes i programvaren for de to brukergrensesnittene. Disse har derfor ikke helt lik funksjonalitet på SCADA som på operatørpanelet. Ettersom programvaren for brukergrensesnittet på PC har den beste løsningen før historisk trendvisning, anbefales operatører å bruke PC basert SCADA for analyse av trender tilbake i tid. Både bilder fra historikk og trend er tilgjengelige i vedlegg C.

5 Regulering AG

Før implementasjon av regulering på tankriggen ble en regulatorprototype utviklet for å teste de ulike reguleringsaspektene i et simulert miljø. Målet med dette var å finne en regulator som ville oppfylle kundens spesifikasjoner uten å faktisk realisere denne i den faktiske tankprosessen.

Det ble gitt overordnede krav fra kunden til hvordan den totale regulatoren skulle oppføre seg. Der var det ønskelig med mulighet for både manuell og automatisk (PID) styring av utgangsverdi,

med rykkfri overgang mellom disse. Dette skulle realiseres ved hjelp av en tracking-inngang på PID-regulatoren. For å motvirke eventuelle forstyrrelser i prosessen ble det også etterspurt en foroverkopling med et innstillbart lead-lag-element som kunne kompensere effektivt. I selve PID-regulatorblokken skulle det være uavhengig innstilt derivatfilter, anti-windup i integraldelen og muligheter for nominelt pådrag ved mangel på integralvirkning. I tillegg skulle settpunktet som ble gitt til regulatoren være ratebegrenset for å unngå brå dynamikk. [7]

Det ble tidlig i reguleringsarbeidet gjort betraktninger om hvilken funksjonalitet som var viktig å bemerke under utviklingen. Disse tok utgangspunkt i både kundens spesifikke krav og utviklernes egne tekniske vurderinger knyttet til den senere implementeringen i en reell vanntankprosess. Blant de ulike betraktningene ble derfor regulatorens programflyt sentral. Denne ble i prototypen utarbeidet på en slik måte at den ble skalerbar og implementert parallelt med hovedfunksjonen i PLS-koden, slik den faktiske regulatoren senere ble implementert. Tankegangen ble også viktig i oppsettet av variablene brukt til reguleringen, da disse ble utarbeidet med skalerbarhet i fokus. På denne måten kunne prototypregulatoren brukes direkte i hovedprosjektet så snart den var utviklet og fungerte som ønsket.

5.1 Regulatoroppsett AG

Regulatoren er implementert i flere distinkte funksjoner og er skrevet i strukturert tekst (ST). Koden er implementert på en slik måte at den kan kalles på fra andre operasjoner i kjøretiden, så den kan kombineres med annen funksjonalitet i slave-PLS. Ettersom PLS-en opererer sekvensielt innenfor hver skannesyklus kan den heller ikke behandle flere ekte samtidige datastrømmer parallelt. Regulatorkoden er derfor strukturert slik at den fungerer korrekt når funksjonsblokkene blir evaluert sekvensielt, og sikrer at alle nødvendige data er oppdatert i riktig rekkefølge for å oppnå forutsigbar styring av systemet.

Oppsett av variable

For å sikre skalerbarhet og muliggjøre en løsning med flere tanker vil hver tank få tildelt ett sett med parametre for hvert pådragsorgan i tillegg til måleverdier og settpunkt, gitt i strukturen kalt `slave_basic`. Dette innebærer alle parametre for PID-, PI-, PD- og P-regulering, filterkonstanter, metningsverdier, manuelt pådrag, ratebegrensning og tidskonstanter for begge pådragsorganene. Ved å knytte en slik struktur til hver slave vil man lett kople riktige parametre til riktig tank ved implementering i et skalert nettverk. [8]

Videre vil det aktive pådragsorganet, enten det er frekvensomformer eller ventil, få tildelt en struktur av typen `actuator`. Tabellen inneholder alle pådragsbidragene, målinger og saturert pådrag fra aktuatoren. Internt i reguleringen benyttes denne til tilstandsbevaring da reguleringen tar inn tidligere tilstander for å regne ut nye verdier. Man får derfor en diskret regulatorimplementasjon med internt minne som legger til rette for blant annet derivatfilter og anti-windup. Å ha alle disse verdiene i strukturer gir også bedre lesbarhet i den totale koden. [8] Variabelstrukturene kan finnes vedlagt i vedlegg E.

Modusvalg

For å implementere valg mellom ulike regulatormodi, -tilstander og pådragsorganer er tre parametre implementert; `mode`, `reg_mode` og `Frequency_regulating`.

Basert på verdien av `reg_mode` i tabell 4 vil regulatoren skifte mellom de fire ulike regulatorene og de ulike underblokkene i koden vil oppføre seg deretter. Dette muliggjør raskt skifte av modi underveis i kjøretiden. Manuell/auto switching styres derimot av en annen variabel `mode` som velger mellom følgende tilstander vist i tabell 4. Denne er implementert som en separat parameter da verdien styrer systemets driftsstatus i HMI. Til sist vil det være mulig å velge mellom de to pådragsorganene ved hjelp av den boolske parameteren `Frequency_regulating`. Denne vil ved høy verdi bety styring via frekvensomformer, og ved lav verdi tilsi ventilstyring. [8]

reg_mode		mode		Frequency_regulating	
1	PID	0	Pådrag AV	0	Styring fra magnetventil
2	PI	1	Manuell styring	1	Styring fra frekvensomformer
3	PD	2	Automatisk styring		
4	P				

Tabell 4: Modi for `reg_mode`, `mode`, og `Frequency_regulating`

Implementasjon på denne måten muliggjør enkelt bytte mellom regulatortilstander og -modi fra HMI og operatørpanel. Løsningen sikrer muligheten for å endre regulatorens respons underveis i kjøretiden, samt at den i et skalert nettverk gir god struktur da de ulike parametrene og tilstandene henger sammen med sin respektive tank.

5.2 Regulatorprogram AG

Regulatoren er utviklet for å være adaptiv for å muliggjøre tilpasning til et bredt spekter av ulike systemer. Den er konstruert med støtte for to forhåndsdefinerte pådragsorgan, hvor hvert organ kan benytte fire ulike regulatortyper. Totalt støttes åtte separate regulatorer, hver med sine egne individuelt konfigurerbare parametere.

Systemet er utformet for å kunne bytte sømløst mellom regulatorene og pådragsorganene uten å forstyrre driften. Dette muliggjør håndtering av systemer med varierende dynamikk eller operative moduser, for eksempel ved veksling mellom grove og fine aktuatorer, eller ved behov for ulike kontrollstrategier avhengig av driftskontekst.

Denne generaliserte utformingen gjør regulatoren godt egnet for både testing, videreutvikling og overføring til andre prosjekter med minimal tilpasning.

5.2.1 Oppdatering av reguleringsparametre

For å håndtere begrensningen i master-PLSen, som ikke støtter flyttall, benyttes funksjonen `Update_Reg_Params` til å skalere og konvertere reguleringsparametrene før de benyttes videre i slave-PLSen. Dette skjer ved hjelp av en intern funksjon som omgjør heltall til flyttall og deretter skalerer dem med faste faktorer, slik at brukeren kan angi verdier med høyere presisjon enn masteren i utgangspunktet tillater.

De omregnede verdiene pakkes i en struktur av typen `Paralell_PID`, som videre konverteres til ideell form i funksjonen `Par_To_Ideal`. Siden regulatoren er implementert i ideell form (K, T_i, T_d), mens operatøren oppgir parametere i parallell form (K_p, K_i, K_d), er denne omregningen nødvendig. (se avsnitt 2.1.3) Parallell form ble valgt for operatørgrensesnittet fordi den gir bedre intuisjon og forståelse av parameterenes effekt, mens ideell form ble valgt for selve regulatorimplementasjonen fordi den gir en mer modulær struktur som er enklere å diskretisere. Funksjonen `Par_To_Ideal` mottar en `Paralell_PID`-struktur og returnerer en tilsvarende `Ideal_PID`-struktur med omregnede variable.

5.2.2 Reguleringstimer

Reguleringen er avhengig av at dens delfunksjoner kjører synkront og til angitt samplingsfrekvens, beskrevet i avsnitt 2.1.8. Derfor er det implementert en funksjon `FreqTimer` som brukes i flere av reguleringsblokkene. Denne funksjonen tar inn en ønsket samplingsfrekvens, og mater ut en pulsrekke til den gitte frekvensen. I tillegg gir funksjonen ut tilhørende reell samplingstid da dette trengs til utregningen av flere parametre. Funksjonen fungerer i praksis som en utvidet "`TON_10_Timer`" og sparer mye kodelinjer gjennom denne felles implementasjonen. [8]

5.2.3 PID-regulatorblokk

PID-reguleringen finner sted i funksjonsblokken `PIDfn` og baserer seg på en klassisk PID-struktur med tillegg for anti-windup, derivatfilter og foroverkopling. Den opererer i diskret tid og beregner pådragsverdiene for hver syklus basert på nåværende settpunkt, målt prosessverdi, samplingstid og regulatorparametre. Bidragene fra proporsonal-, integral- og derivatddelen, samt foroverkoplingen, blir beregnet hver for seg og lagres i den nevnte `actuator`-strukturen.

Funksjonen sjekker ved hjelp av `reg_mode` hvilke av de ulike bidragene som er aktive. De inaktive delene blir satt lik 0. Det er også lagt inn en sjekk på at integraltiden er ulik null hvor dette er nødvendig for å unngå eventuelle beregningsfeil. På denne måten vil man sikre robust skifte av modi og forenkle summeringen i hovedfunksjonen.

Regulatorens virkemåte følger den teoretiske oppbygningen beskrevet i avsnitt 2.1.5 og er diskretisert med eksplisitte differenslikninger gitt i samme avsnitt. Feilen $e[k]$ mellom settpunkt og målt verdi utgjør grunnlaget for reguleringen.

Proporsjonalledd $U_P[k] = K \cdot e[k]$

Det proporsjonale leddet utgjør en direkte respons på målt avvik og er aktivt ved alle modi. Pådraget beregnes som en skalert versjon av feilen og følger implementasjonen som vist i likning 6 fra teoridelen. Leddet gir umiddelbar respons og påvirker regulatorens stivhet og følsomhet.

Integralledd med anti-windup $U_I[k] = U_I[k-1] + K \cdot \frac{T_s}{T_i} \cdot e[k] + \frac{T_s}{T_t}(U_{\text{sat}}[k-1] - U[k-1])$

Integralleddet beregnes akkumulativt over tid, basert på både nåværende avvik og en tracking-mekanisme. Tracking sørger for at integratoren holder seg synkronisert med faktisk pådrag, noe som reduserer risikoen for windup ved metningssituasjoner. Anti-windup implementeres i tråd med beskrivelsen i teoridelen med likning 7, der det siste leddet justerer integratorens verdi basert på differansen mellom ønsket og faktisk pådrag. Dette gir rykkfri overgang ved aktivering og deaktivering av regulatoren, som omtalt i avsnitt 2.1.5.

Derivatledd med derivatfilter $U_D[k] = \beta \cdot U_D[k-1] - K \cdot \frac{T_d}{T_s} \cdot (1-\beta) \cdot (y[k] - y[k-1])$

Derivatleddet er basert på endringen i målt prosessverdi fremfor endringen i feilen, og gir derfor en såkalt PI-D regulator. (se avsnitt 2.1.1) Dette reduserer risikoen for derivatspark ved raske endringer i referanseverdien, og gir en mer robust respons i transientfaser. For å håndtere høyfrekvent målestøy er det implementert et førsteordens lavpassfilter. Implementasjonen følger også diskretiseringen fra avsnitt 2.1.5 og vises i likning 8. Filteret tilpasses ved hjelp av en filterfaktor $\beta = \frac{T_d}{T_d + T_s \cdot N}$ som kan justeres av operatør.

Foroverkopling fra forstyrrelse $U_{\text{ff}}[k] = a_1 U_{\text{ff}}[k-1] + b_0 d[k] + b_1 d[k-1]$

Foroverkopling fra målt forstyrrelse er implementert som en diskretisert lead-lag-funksjon. Denne benytter en Tustin-approksimasjon av overføringsfunksjonen $F(s) = \frac{T_1 s+1}{T_2 s+1}$ og implementeres som en rekursiv differenslikning, beskrevet i avsnitt 2.1.6. Pådraget som beregnes gjør det mulig å motvirke effekten av forstyrrelsen raskere enn en klassisk regulator ville gjort alene, og man får dermed bedre dynamisk ytelse.

5.2.4 Ratebegrensning

Både setpunkt og pådrag blir ratebegrenset når de får endringer. Dette gjøres for å sikre glatte overganger og rykkfri reguleringsytelse. Til dette formålet er det implementert to ulike funksjoner; `QInterPol` og `first_order_lowpass`, som håndterer ratebegrensningen på to ulike måter.

For referanseglattingen benyttes den 5.ordens polynombaserte skalafunksjonen i `QInterPol`. Gitt den ønskede raten blir tilsvarende overgangstid beregnet, og en jevn kurve genereres mellom nåværende og nytt settpunkt. Funksjonen sikrer kontinuerlig verdi, hastighet og akselerasjon, og tillater dynamiske endringer underveis uten rykk. Skalafunksjonen ligger under teoridelen i avsnitt 2.1.7.

For ratebegrenset pådrag brukes et simpelt første ordens lavpassfilter implementert i funksjonen `first_order_lowpass`. Ved endring i ønsket pådrag gjennomgår funksjonen en initialisering der filteret igangsettes med gjeldende sensorverdi. Deretter utføres kontinuerlig filtrering basert på valgt vekt, med avrunding til nærmeste heltall.

Denne løsningen gir en enkel og robust implementasjon i regulatorkoden. `QInterPol` håndterer settpunktendringer jevnt og kontinuerlig, noe som gir kontrollert referansebevegelse uten rykk. `first_order_lowpass` simulerer fysisk treghet i aktuatorene ved å glatte ut pådraget, slik at systemet reagerer mykt og uten unødig rykk i pådragsverdi. Sammen sikrer de en god robusthet mot plutselige endringer i pådrag og settpunkt. [8]

5.2.5 Hovedfunksjon for regulering

Hovedfunksjonen for høyderegulering kalles `PosCtrReg` og fungerer som en samlende funksjon som tar inn alle parametre og bidrag fra de andre funksjonsblokkene.

Funksjonen vil kontinuerlig oppdatere reguleringsparametre fra brukeren gjennom `Update_Reg_Params`. Det samme blir gjort med endringer i setpunktet gjennom ratebegrensningen i `QInterPol`, slik at regulatoren får en fin referanseglatting uavhengig av modus. Videre initialiserer funksjonen en reguleringstimer av typen `FreqTimer` for å styre kjøretiden. Med pulstoget fra timeren kjøres `PIDfn`, funksjonen som beregner alle fire ulike pådragsbidrag. I tillegg brukes pulstoget til å hente nye måleverdier for prosessverdi og forstyrrelse til bruk i utregningen. Funksjonen regner dermed alltid ut et pådrag fra regulatoren, uavhengig om dette skal brukes eller ikke. At PID-regulatoren alltid kjører gjør at tracking-inngangen i integralleddet hele tiden er oppdatert. Dette er nødvendig for at integralet skal følge faktisk bidrag til enhver tid og dermed sikre rykkfri overgang mellom reguleringsmodi.

Hovedlogikken i reguleringsfunksjonen avhenger av parameteren `mode`, og velger mellom en av tre modi; *Pådrag AV*, *Manuell modus* og *Automatisk modus*. Under førstnevnte vil hovedregulatoren sette ønsket pådrag lik 0, slik at pådraget sakte skrus av gjennom ratebegrensingen. Manuell modus lar brukeren selv velge pådraget, og omregner dette til riktig utgangsverdi til bruk i DAC-en. Den nye verdien går også gjennom ratebegrensningen og sikrer dermed en gradvis overgang mellom modiene. Automatisk regulering summerer de ulike pådragsbidragene fra funksjonen `PIDfn` før disse også omregnes og ratebegrenses. [8]

5.3 Simulert prosess for testing AG

For å kunne utvikle og teste regulatoren på en trygg og effektiv måte, ble det laget et simulert miljø som etterlignet den fysiske vanntanken. Modellen ble basert på en digital tvilling, etablert gjennom systemidentifikasjon. Dette muliggjorde utvikling, tuning og verifisering av regulatoren uavhengig av den fysiske riggen.

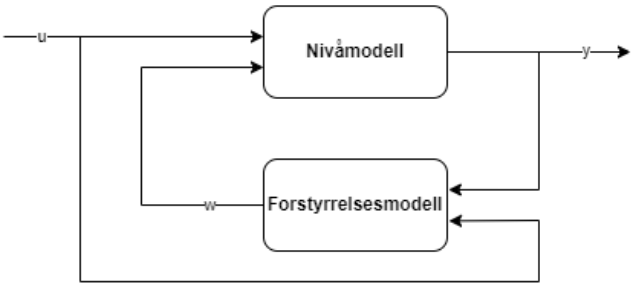
5.3.1 Systemidentifikasjon

Det ble først forsøkt å utvikle en whitebox-modell basert på kjente fysiske egenskaper ved tanksystemet (se avsnitt 2.3). Modellen ble formulert som en andreordens overføringsfunksjon med

tidsforsinkelse, utledet gjennom linearisering av tankdynamikken. Tidskonstanter og forsterkning ble estimert gjennom målinger fra systemet.

Denne modellen ga et godt teoretisk grunnlag, men viste seg lite egnet for simulering av sanntidsatferd. Avviket mellom modell og virkelighet ble for stort, hovedsakelig grunnet systemets ikke-lineære karakter.

En datadrevet *blackbox*-tilnærming ble derfor valgt, med bruk av *System Identification Toolbox* i MATLAB [9]. Råverdier ble hentet direkte fra PLS, og systemet ble modellert med to gjensidig avhengige NLARX-strukturer: én for nivå og én for utstrømning. Disse ble så koplet sammen i simulink vist i figur 5. Dette muliggjorde at ikke-lineær dynamikk og samspill mellom komponenter kunne fanges opp.



Figur 5: Illustrasjon av gjensidig avhengige modeller

5.3.2 Modellvalg

Flere modelltyper ble vurdert. Enkle lineære modeller som TF- og tilstandsrommodeller viste begrenset evne til å fange systemets dynamikk, mens enkelte ikke-lineære nettverk overtilpasset treningsdata og generaliserte dårlig.

Valget falt derfor på en NLARX-modell med lineær utgang, for både nivå- og forstyrrelsesmodell. Denne ga best balanse mellom presisjon, stabilitet og generaliseringsevne. Samtidig tillot den manuell justering av antakelser om tidsforsinkelse og hvor mange tidligere tidssteg som skulle påvirke prosessverdien, noe som styrket transparens og justerbarhet.

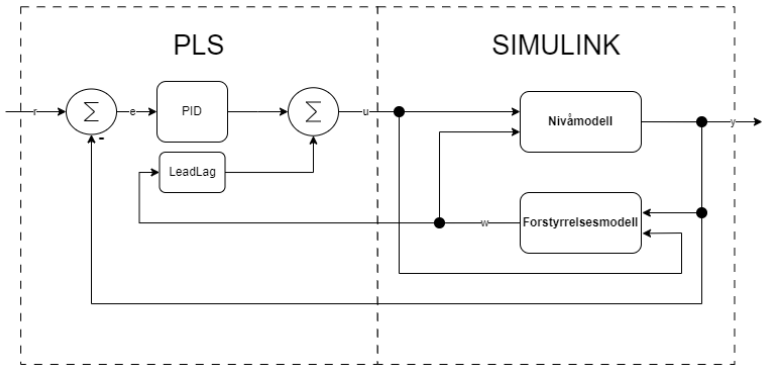
Modellen forklarte omtrent 75% av variansen i valideringsdataene, og responderte realistisk på systemets dynamikk. Dette muliggjorde testing med realistiske pådragsverdier i simulert miljø, og ga et solid grunnlag for finjustering av regulatoren før fysisk implementering. Ventilens respons var derimot vanskelig å modellere presist grunnet uforutsigbar ytelse. I simuleringen ble det derfor benyttet et forenklet lavpassfiltert pådragssignal inn til frekvensomformermodellen. Dette ga i realiteten den modellerte ventilen en tidskonstant på 20 sekunder mer enn frekvensomformeren.

En mer detaljert gjennomgang av modellutprøving, datagrunnlag og evalueringsmetodikk er gitt i vedlegg F.

5.3.3 Simulasjonsoppsett

Simuleringen ble implementert som en lukket reguleringssløyfe med PLS og Simulink koblet via OPC. Måleverdier og pådrag ble kontinuerlig utvekslet. For å redusere belastningen ble det utviklet et MATLAB-script som kun sendte reguleringsparametre til PLS når verdier faktisk endret seg. Dette forbedret ytelsen betydelig og muliggjorde simulering i sanntid.

Den simulerte tanken tok imot pådrag, beregnet forstyrrelse og væskehøyde gjennom de koblede NLARX-modellene, og sendte prosessverdiene tilbake til PLS. Dette dannet et realistisk testmiljø der regulatorens respons kunne evalueres i sanntid og med samme struktur som i det fysiske systemet, lik det gitt av figur 6.



Figur 6: Forenklet oversikt over det simulerte miljøet

5.4 Simulering og valg av reguleringsstrategi AG

Den simulerte tanken ble brukt til å teste funksjonaliteten i den utviklede koden og undersøke designvalgene gjort underveis. Simuleringen ble brukt både for å validere regulatorens oppførsel under ulike driftsbetingelser og for å finne en hensiktsmessig regulator for hver aktuator. Under simulasjonen var alle regulatorfunksjoner aktivert, inkludert anti-windup og foroverkopling, slik at simuleringen ble en fullverdig funksjonstest.

Testene som ble gjennomført fulgte et utarbeidet testskjema med konkrete deltester og kriterier for ventet respons. Skjemaet dekket alle hovedfunksjonene i regulatoren, inkludert manuell modus, rykkfri overgang, anti-windup, foroverkopling og ratebegrensning. Dette sikret at regulatorens komponenter ble validert separat før implementasjon på riggen.

Simuleringen ga i stor grad realistiske resultater for frekvensomformeren, og regulatoren viste rask og stabil respons. Dette bidro til økt trygghet rundt at regulatorstrukturen ville fungere etter hensikten. Tilsvarende forsøk med ventilstyring var derimot mindre vellykkede, hovedsaklig på grunn av manglende presisjon i ventilmodellen.

På bakgrunn av testene ble det konkludert med at PID-regulering ga best resultat for begge pådragsorganene. For frekvensomformeren ga derivatleddet mulighet til å øke forsterkningen i det proporsonale leddet uten å miste stabilitet. Integralleddet var også nødvendig for å eliminere stasjonært avvik, noe som utelukket ren P- eller PD-regulering. Tilsvarende ble PID også valgt for ventilen, selv om resultatene der var preget av en forenklet ventilmodell som reduserte realismen i forsøket.

Samlet sett fungerte simuleringen som et effektivt og trygt verktøy for validering og valg av reguleringsstrategi. Simuleringen viste stor verdi som prinsippsjekk da den bekreftet regulatorens funksjonalitet og muliggjorde grundig feilsøking før fysisk implementasjon.

5.5 Regulatorimplementasjon

Overgangen fra simulering til fysisk rigg foregikk uten store strukturelle endringer, ettersom regulatorfunksjonaliteten ble utviklet med dette i fokus. Den samme PLS-koden ble brukt i simuleringsoppsettet og i det ferdige produktet, med kun mindre justeringer. Alle sentrale reguleringsfunksjoner ble testet og verifisert i praksis, slik at også det fysiske systemet oppfylte de samme kravene som det digitale.

Det måtte velges en passende samplingstid for hele regulatoren, slik at alle funksjoner kunne operere synkront. Basert på teorien i avsnitt 2.1.8 ble en tastetid på $100ms$ ($10Hz$) valgt for å gi god balanse mellom responstid og stabilitet. Dette er raskt nok til å følge prosessens dynamikk, samtidig som regulatorberegning og signalbehandling får tilstrekkelig tid til å gjennomføres uten overbelastning.

Valgt tastetid tillot også lavpassfiltrering av målesignalet, noe som reduserte støy og forbedret regulatoroppførsel. Måling, beregning og pådragsutgang utføres sekvensielt innenfor hver syklus, som gir deterministisk og forutsigbar systemoppførsel. For å sikre at regulatorparametrenes påvirkning forblir deterministisk, logges faktisk tastetid fortløpende med $10ms$ oppløsning. Dette gjør det mulig å bruke faktisk tastetid under beregning av pådrag.

Under test viste regulatoren seg å ha sammenlignbar oppførsel med simuleringen ved bruk av frekvensomformer som pådragsorgan. For ventilstyring ble resultatet faktisk bedre enn forventet, noe som sannsynligvis skyldes at den fysiske ventilen responderte langt bedre enn modellen som ble brukt i simuleringen. Reguleringsytelsen var likevel begrenset i starten da tuningparametrene fra tankmodellen ikke lot seg overføre. I tillegg ble det observert nødvendigheten av en enkel skalering av pådragsverdiene for å kompensere for dødsoner i aktuatorene. Dette omtales i avsnitt 5.5.1.

Samtlige åtte regulatorer ble implementert; PID-, PI- PD- og P-regulering for både ventil og frekvensomformer. Alle disse ble aktivt testet og gjort tilgjengelige for valg i sluttproduktet.

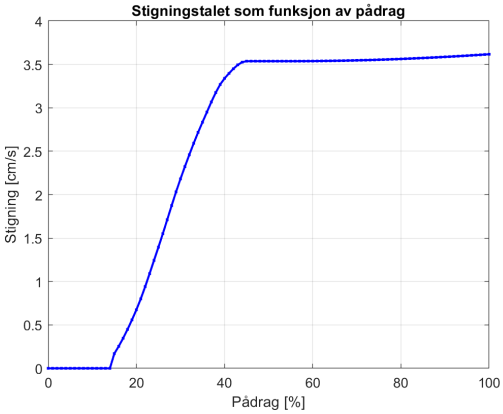
Samlet sett fungerte regulatoren stabilt og pålitelig, også i den fysiske riggen, og oppfylte de funksjonelle kravene definert for prosjektet. Implementasjonen bekreftet også at struktur, dokumentasjon og funksjonsinndeling fungerte godt i praksis, og la til rette for videre drift og justering.

5.5.1 Justering av pådragsfunksjon KL

Under tuning av regulator ble det oppdaget at regulatoren opplevde noen utfordringer knyttet til hvilket pådrag som faktisk ble resultat av pådragsverdiene.

For å kartlegge dette ble det satt opp en test for å sjekke hvor mye pådrag de forskjellige pådragsverdiene ga. Testen ble gjennomført ved å teste hvor lang tid det tok å fylle tanken fra 10 cm til 50 cm, for begge organene. Det ble tatt tid for påfylling for hver femte prosentverdi av pådraget. Resultatene kan brukes til å lage en grafisk representasjon av stigningstallet til tanknivået som funksjon av prosentverdi for pådraget, gitt av figur 7 og 8:

Figur 7: Stigningstallet fra forskjellige pådragsverdier for frekvensomformer



Figur 8: Stigningstallet fra forskjellige pådragsverdier for ventil

Fra grafene ser man at begge aktuatorene har dødsoner, hvor endring i sendt pådrag ikke har noen effekt på det virkelige pådraget på tanken. Frekvensomformeren har dødsone på det nedre sjiktet av pådragsområdet, hvor 50% i pådragsverdi må til før tanknivået faktisk begynner å stige. For ventilen er det dødsoner både på nedre og øvre sjikte av området. Effekten fra pådraget er synlig ved 15%, og ventilen når maksimalt pådrag ved 45%. Dette er problematisk for regulatoren da det vil føre til at I-leddet i regulatoren bygger seg opp feil. Dette skjer til tross for tracking, fordi trackingen først begynner ved 0% og 100% pådrag. Videre vil det medføre stasjonært avvik for P og PD-regulatorene, ettersom virkelig pådrag vil ta slutt før regulatoren setter ønsket pådrag til 0%.

For å forhindre denne problematikken var det ønskelig å endre noe på hvordan pådraget sendes ut til pådragsorganene. Fra grafene i figur 7 og 8 kan man se at arbeidsområdet til begge pådragsorganene

er tilnærmet lineære. På grunn av dette var det bare nødvendig å kartlegge pådragsverdiene etter aktuatorens arbeidsområde. Denne kartleggingen ble utviklet som en funksjon etter regulatorens utregning av pådrag. Funksjonen tar inn et pådrag u og kjente satureringsverdier fra regulatoren u_{min} og u_{max} for å finne en prosentverdi for pådraget representert som desimaltall:

$$u_{\text{frac}} = \frac{u - u_{\text{min}}}{u_{\text{max}} - u_{\text{min}}} \tag{13}$$

Funksjonen tar deretter inn aktuatorens virkelige arbeidsområde med p_{min} og p_{max}, hvor p beskriver en prosentverdi som desimaltall:

$$p_{\text{range}} = p_{\text{max}} - p_{\text{min}} \tag{14}$$

Til slutt beregner funksjonen hvilket pådrag som skal sendes til aktuatoren, basert på DAC-omformerens bitoppløsing, N:

$$u_{\text{DAC}} = (2^N - 1)p_{\text{min}} + (2^N - 1)u_{\text{frac}} \cdot p_{\text{range}} \tag{15}$$

Funksjonen er utviklet slik for å kunne brukes til enhver aktuator, så sant man kjenner oppløsning til aktuatorens digitale omformer, dens virkelige minimums- og maksimumsverdier og hvilken saturering av pådrag som er implementert i regulatoren. På dette viset kan funksjonen brukes til begge pådragsorganene på systemet.

5.5.2 Tuning AG

Som utgangspunkt ble det benyttet SIMC-metoden [1, s.127] for tuning, med eksperimentell justering i etterkant. Dette ble utført separat for de to pådragsorganene med fokus på PID-regulatoren ettersom den ga best ytelse. Målet for tuningen var å oppnå rask respons uten oversving, slik kravet fra kunde spesifiserte.

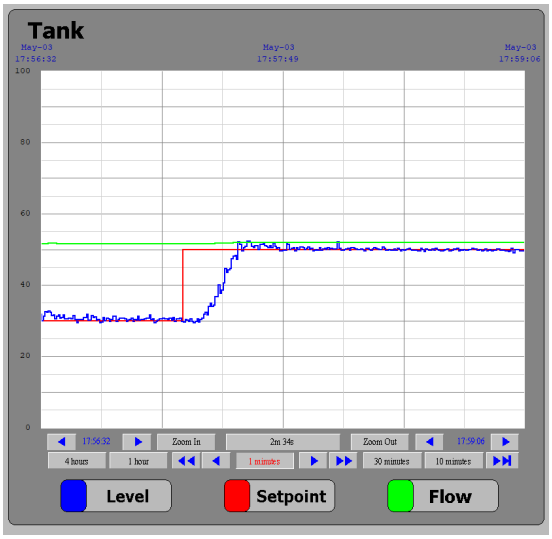
Prosessen ble støttet av sanntids trendvisning på HMI, som ga umiddelbar tilbakemelding på hvordan justeringer påvirket responsen. Tuning av frekvensomformeren forløp relativt effektivt og resultatene stemte overens med forventningene fra simulering. Ventilstyringen viste seg derimot å være mer krevende, og krevde flere iterasjoner før tilfredsstillende respons ble oppnådd. Dette skyldtes i hovedsak ventilens mer utfordrende karakteristikk.

5.5.3 Regulatorytelse AG

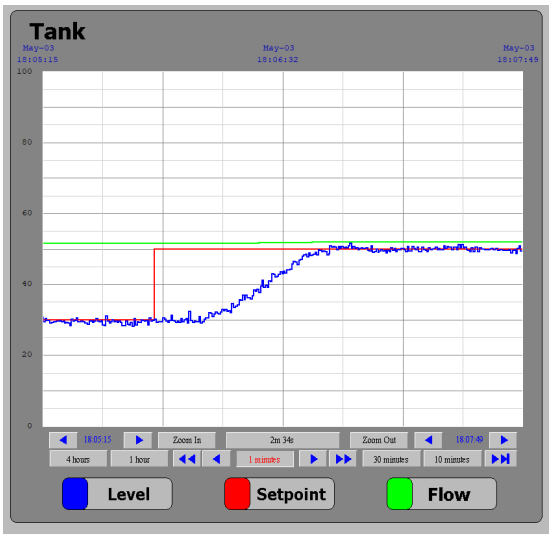
Etter tuning ble ytelsen til den implementerte PID-regulatoren grundig testet og viste svært gode resultater på begge pådragsorganer. For frekvensomformer oppnådde regulatoren rask respons uten oversving og tilnærmet null stasjonærfeil. Responsen samsvarte i stor grad med den simulerte modellen, med kun mindre forskjeller i innsvigningstid. For ventilregulering var det tydelig større treghet i pådraget, men dette ble effektivt håndtert ved å justere ratebegrensningen på både settpunkt og pådrag. I motsetning til simuleringen, som slet med svigninger på grunn av en forenklet ventilmodell, oppnådde den fysiske riggen en stabil respons uten oversving. Dette vises tydelig i figurene 9 og 10, sammenlignet med de simulerte forløpene i figur 11 og 12.

PI-, PD- og P-regulatorene ble også testet og tunet, og oppnådde akseptabel reguleringsytelse, men med begrensninger sammenlignet med PID-regulatoren. Spesielt mangelen på integral- og derivatledd påvirket evnen til å håndtere henholdsvis stasjonærfeil og raskt endrende signaler. Likevel gir disse alternativene fleksibilitet i driftssituasjoner med enklere krav.

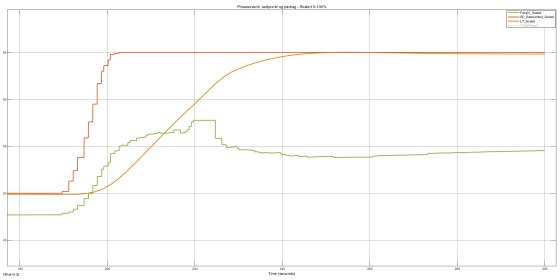
Systemet var utsatt for noe støy og dødsoner i aktuatorene, men disse ble effektivt håndtert gjennom lavpassfiltrering og justering av pådragsverdier (se avsnitt 5.5.1). Totalt sett fremstod regulatoren som robust og pålitelig, også i møte med forstyrrelser, og oppfylte alle funksjonelle krav til reguleringsytelse i prosjektet.



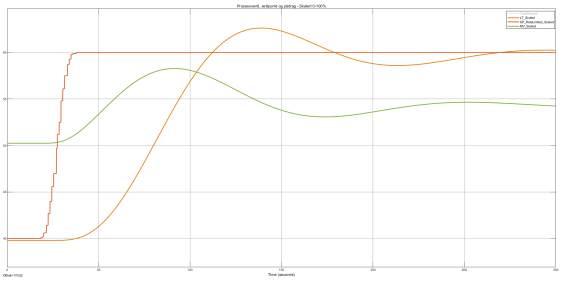
Figur 9: PID med frekvensomformer, reell tank



Figur 10: PID med magnetventil, reell tank



Figur 11: PID med frekvensomformer, simulert tank



Figur 12: PID med magnetventil, simulert tank

Rate-begrenset referanse
Prossessverdi
Pådrag

6 Alarmsystem og feilhåndtering TR

Denne seksjonen tar for seg hvordan systemet er designet for å håndtere forskjellige typer feil, hvordan dette er implementert og hvilke alarmer som er definert. Det skal også ses noe på hvordan alarmsystemet er implementert teknisk, og hvordan kvittering av alarmer fungerer.

Vanlige feil
Det er definert fire forskjellige alarmsituasjoner som regnes som ikke-kritiske og som ikke har annen håndtering enn indikasjon. Dette gjelder når tanknivået er under 20% eller over 80%, og om regulerinsavviket er mer enn 10% i lengre enn 60 sekunder. Om noen av disse alarmene aktiveres blinker indikatorlampene med en frekvens på 1 Hz.

Kritiske feil
De to kritiske alarmene som er definert for systemet er for tanknivå under 10% eller over 90% av tankens høyde. Ved disse alarmene blinker indikatorlampene med 5Hz. I tillegg til indikasjon, håndteres disse alarmene ved at systemet går i "Recovery mode". Denne modusen innebærer at tanken reguleres til 50% og holdes der frem til alarmen kvitteres.

6.1 Kvittering av alarm AE

Systemet er bygget med redundans ved at en kvittering (ack) fra en bruker håndteres sentralt av masterenheten. Når en bruker sender en kvittering, vil masteren informere alle relevante enheter i systemet om at kvitteringen er mottatt. Masteren opprettholder denne statusen til den enheten (slaven) som opprinnelig utløste alarmen, eksplisitt bekrefter at den har mottatt kvitteringen. Først da regnes kvitteringen som fullført, og masteren kan videreformidle dette til øvrige enheter.

Dette er gjort for å sikre at kvitteringer ikke går tapt eller ignoreres i tilfeller med midlertidig kommunikasjonsfeil, og for å sørge for at hele systemet er konsistent. Kun godkjente enheter, som HMI eller SCADA, har tillatelse til å sende kvitteringer, noe som bidrar til kontrollert og trygg alarmbehandling.

Denne mekanismen sikrer at alle deler av systemet er samkjørte med hensyn til alarmstatus og kvittering. Prosessen minimerer risikoen for feil i alarmlogikk og sørger for at ingen enheter går tilbake til normaltilstand før kvitteringen er korrekt mottatt og behandlet.

Det var også nødvendig å sørge for at implementasjonen for kvittering håndterte kundens spesifikasjoner for atferd fra alarmlampene. Dette er implementert som i tabell 5.

Prioritering	**Kritisk**	**Aktiv**	**Kvittert**	**Lampe**
1	Ja	-	Nei	5 Hz
2	Nei	-	Nei	1 Hz
3	-	Ja	Ja	Fast
4	-	Nødmodus	-	Fast
5	-	-	-	Av

Tabell 5: Prioriteringsliste for lampe

Tabellen viser en prioriteringsliste for hvordan alarmlampene lyser, ut fra alarmsituasjonen. Tabellen leses som at om det finnes en alarminstans med status høyere oppe, er det denne som styrer hvordan lampene lyser.

6.2 Statuskoder for deling av systemtilstand TR

For å dele systemets tilstand på en kompakt og dataeffektiv måte benyttes statuskoder. Disse statuskodene bærer med seg informasjon om alarmtilstander i systemet og er utformet for effektiv overføring og tolkning mellom enheter. Hver alarmtilstand i systemet kan beskrives med tre boolske verdier, satt sammen til en bitstrøm av 3 bits som representerer tilstanden slik:

- Bit 2 (MSB): Kritisk (1) / Ikke kritisk (0)
- Bit 1: Aktiv (1) / Inaktiv (0)
- Bit 0 (LSB): Kvittert (1) / Ikke kvittert (0)

Denne bitstrømmen kan enkelt konverteres til et desimaltall mellom 0 og 7. Dette gjør det mulig å bruke én byte per alarm, og gjør at tilstanden til hver alarm raskt kan identifiseres basert på tallverdien.

Binær	**Desimal**	**Kritisk**	**Aktiv**	**Ikke ACK**	**Statuskode (hex)**
000	0	Nei	Nei	Nei	0x00
001	1	Nei	Nei (Var aktiv)	Ja	0x02
010	2	Nei	Ja	Nei	0x01
011	3	Nei	Ja	Ja	0x03
100	4	Ja	Nei	Nei	0x00
101	5	Ja	Nei (Var aktiv)	Ja	0x02
110	6	Ja	Ja	Nei	0x01
111	7	Ja	Ja	Ja	0x03

Tabell 6: Tolkning av statuskoder (3 bit)

Hver alarm tildeles en statuskode i henhold til tabell 6. Kode `0x03` indikerer en aktiv alarm med en pågående feilsituasjon. Kode `0x02` representerer en aktiv alarm som ennå ikke er kvittert (ACK), men hvor feilsituasjonen er opphørt. Kode `0x01` tilsier at feilen fortsatt er til stede, men at alarmen er kvittert, mens kode `0x00` angir at det verken foreligger en aktiv feil eller en ubehandlet alarm.

For å skille mellom normale og kritiske alarmer kombineres statuskoden med en typekode. Kritiske alarmer tildeles typekode `0x20`, mens normale alarmer bruker `0x10`. Disse to kodene kombineres i slavenheten ved hjelp av en bitvis OR-operasjon før de sendes til masteren.

Eksempelvis vil en ikke-kvittert, aktiv kritisk alarm ha statuskode `0x03` og typekode `0x20`. Ved å utføre OR-operasjonen:

$$\texttt{0x20}\ \text{OR}\ \texttt{0x03} = \texttt{0x23}$$

kan masteren tolke denne sammensatte koden ved hjelp av AND-operasjoner:

$$\text{Status} = \texttt{0x23}\ \text{AND}\ \texttt{0x0F} = \texttt{0x03}$$

$$\text{Type} = \texttt{0x23}\ \text{AND}\ \texttt{0xF0} = \texttt{0x20}$$

Dette gjør det mulig for masteren å skille både alarmtype og alarmstatus fra én enkelt tallverdi.

Desimalkoden brukes hovedsaklig i slaven til å bestemme blinkingen av alarmlampe, men sendes også til master for å styre blinking av alarm i HMI og SCADA.

7 Diskusjon

AG

Denne seksjonen skal ta for seg forskjellige metoder som er benyttet i prosjektet, bruk og usikkerhet i resultatene og til slutt teknisk måloppnåelse av prosjektets målsettinger.

7.1 Metoder

Prosjektet har benyttet en helhetlig og systematisk tilnærming til utviklingen av et komplett automasjonsanlegg. Metodene spenner fra tradisjonell PLS-programmering til mer avanserte teknikker som systemidentifikasjon ved bruk av datadrevne NLARX-modeller. Dette ga innsikt i ikke-linearieter og forsinkelser som ellers ville vært vanskelig å fange i en teoretisk modell. Valget av å implementere en egen kommunikasjonsprotokoll viste seg også hensiktsmessig med tanke på skalerbarhet og fleksibilitet.

Arbeidet med design og utvikling av brukergrensesnitt ble gjennomført i tråd med High Performance HMI-prinsipper, noe som bidro til et ryddig, effektivt og oversiktlig sluttprodukt. Hele løsningen ble kontinuerlig testet i en lukket sløyfe, både med fysisk rigg og digital tvilling, noe som muliggjorde høy grad av iterasjon og forbedring underveis.

7.2 Resultater

Prosjektets delsystemer ble i stor grad realisert som planlagt. Kommunikasjonsoppsettet mellom master og slave fungerte robust med lav forsinkelse og god redundans, og var avgjørende for systemets helhetlige pålitelighet. Protokollen basert på adresser og data i separate meldinger muliggjorde stor grad av fleksibilitet og la grunnlaget for den ønskede skalerbarheten.

Reguleringssystemet oppnådde høy presisjon med null oversving og rask innsvigning, og regulatorene for både ventil og frekvensomformer ble nøye tunet og verifisert. I tillegg til den planlagte enkeltregulatoren per aktuator, ble det utviklet seks fullverdige regulatorvarianter, noe som ga økt fleksibilitet og bedre verktøy for operatøren.

Brukergrensesnittene på både PC og operatørpanel ble funksjonelle, intuitive og godt integrert med resten av systemet. Trendvisning og historikk muliggjorde god overvåkning og evaluering av prosessforløpet. Alarmsystemet inkluderte kvitteringslogikk, prioritetsnivåer og feilsikret kommunikasjon mellom master og HMI, som fungerte godt.

7.3 Usikkerheter

Noen utfordringer ble avdekket underveis. Ventilpådraget viste seg vanskelig å modellere presist, grunnet lang og variabel tidsforsinkelse. Dette ble løst med en forenkling i simuleringen, som kan ha påvirket presisjonen i noen tilfeller. Videre ble ikke brukergrensesnittet testet av en uerfaren operatør som planlagt, noe som kunne gitt ytterligere innsikt i brukervennlighet.

Mengden treningsdata for simuleringen var også noe begrenset, og sensordataene inneholdt støy, noe som trolig påvirket prestasjonen til de mer avanserte modellen i systemidentifikasjonen. Til tross for dette ble sluttvalgt modell vurdert som tilstrekkelig for simulering og regulatorutvikling.

Grunnet begrensninger i operatørpanelets oppdateringfrekvens har det vært nødvendig å redusere blinkfrekvensen til kritiske alarmer til 2.5Hz istedenfor 5Hz.

7.4 Måloppnåelse

Evaluert opp mot prosjektets målsetninger fra vedlegg A, vurderes måloppnåelsen som svært høy.

Effektmål:

- **Anvendbar regulator:** De implementerte regulatorblokkene er fleksible og gjenbrukbare. Tuning og struktur kan enkelt tilpasses andre prosesser.
- **Presis og stabil regulering:** Null oversving og rask respons ble oppnådd for begge pådragsorganer, og ble bekreftet i fysisk test.
- **Økt kompetanse:** Gruppen har demonstrert avansert bruk av både PLS-programmering, industriell kommunikasjon og utvikling av profesjonelt brukergrensesnitt. Dette viser klar kompetanseheving.
- **Robust kommunikasjon:** Kommunikasjonsflyten med både feedback og kvittering ble utviklet etter spesifikasjon og fungerte stabilt under testing.
- **Brukervennlige grensesnitt:** Begge grensesnittene ble intuitive, stabile og i tråd med HPHMI-prinsippene.
- **Skalerbarhet:** Systemarkitekturen og kanalfordelingen muliggjør fremtidig utvidelse til fire slaver med tre tanker hver, slik definert i prosjektmålene.

Resultatmål:

- Alle resultatmål relatert til PLS-programmering, SCADA/HMI-funksjonalitet, regulatoroppførsel, logging, feilhåndtering og verifisering ble møtt.
- Det ble utviklet testprosedyrer og systemet ble grundig validert gjennom testing.
- Simulering og fysisk test viste høy samsvar med kravene om respons, stabilitet og funksjon.

8 Konklusjon AG

Prosjektet har i all hovedsak levert et helhetlig og teknisk robust system for nivåregulering med høy presisjon og fleksibilitet. Målene som ble satt i forprosjektet, både på effekt- og resultatmål, er oppnådd i stor grad. Gjennom systematisk arbeid, avanserte simuleringsmetoder og vekt på både teknisk kvalitet og brukervennlighet, har det blitt realisert en løsning som både fungerer i praksis og er egnet for fremtidig utvidelse. Eventuelle svakheter vurderes som mindre og har ikke hindret prosjektets overordnede måloppnåelse. Samlet sett fremstår prosjektet som en vellykket leveranse av høy kvalitet.

Referanser

[1] C. F. Sætre, «Dynamiske systemer: Modellering, Simulering og Regulering,» *Forelesningsnotater for IELET2101*, 2025.

[2] K. Børdalen, *Forslag til norsk terminologi for reguleringsteknikk, servoteknikk m.m.* 1957. **url**: https://www.ntnu.no/documents/10355/1262750362/Rokseteknikk.pdf.

[3] T. Anstensrud, «Tema 5: Alternative avbildninger,» *Forelesning for IELET2102*, 2022.

[4] T. Anstensrud, «Tema 4: Folding,» *Forelesning for IELET2102*, 2022.

[5] MathWorks, *Nonlinear ARX Model Structures*, Accessed April 7, 2025, 2024. **url**: https://www.mathworks.com/help/ident/ug/nonlinear-arx-model-structures.html (**urlseen** 7 **april** 2025).

[6] B. Hollifield **og** H. Perez, *Maximize Operator Effectiveness: High Performance HMI Principles and Best Practices*. PAS Global, 2017.

[7] A. Eikeland, A. Garberg, K. H. Liland, T. J. G. Rein **og** L. N. Tran, «Forprosjekt,» *IELET2104 - Automatiseringsprosjekt gruppe 6*, 2025.

[8] A. Garberg **og** A. Eikeland, «Arbeidsnotat: PID-funksjonsblokk,» *IELET2104 - Automatiseringsprosjekt gruppe 6*, 2025.

[9] *System Identification Toolbox*, MathWorks, 2025. **url**: https://www.mathworks.com/products/sysid.html (**urlseen** 7 **april** 2025).

Vedlegg

A Prosjektmål

Prosjektmålene deles inn i to hovedkategorier; effektmål og resultatmål. Hensikten med dette er å skille mellom tekniske, målbare mål og de overordnede målene som sier noe om hva prosjektet skal oppnå i sin helhet. Disse ble utarbeidet i forprosjektet [7] i forkant av utviklingen og deres oppnåelse vil drøftes i denne rapporten.

Effektmål

Effektmål beskriver de overordnede og langsiktige fordelene prosjektet skal oppnå. Målene beskriver hvilken nytte prosjektet skal levere i et bredere perspektiv enn det rent tekniske.

- **Anvendbar regulator:** Regulatoren skal, på et overordnet nivå, muliggjøre gjenbruk av utformings- og tuningmetodene på tvers av flere ulike prosesser. Dette gir en mer kostnadseffektiv og fleksibel drift, samtidig som den langsiktige verdien øker når teknologien kan tilpasses ulike typer anlegg eller applikasjoner.
- **Presis og stabil regulering:** Systemet skal regulere tanknivået med *null oversving* og *rask responstid*. Ved å oppnå null oversving og rask responstid i regulatoren, sikres en mer stabil og forutsigbar drift over tid. Dette tærer mindre på komponentene og bidrar til økt sikkerhet i reguleringssløyfa.
- **Økt kompetanse innen reguleringsteknikk:** Prosjektet skal løfte den praktiske kunnskapen og gi økt erfaring med implementering og utvikling av reguleringssystemer i praksis for de involverte partene.
- **Robust kommunikasjon og tilbakemelding:** En sømløs dataflyt mellom master- og slaveenheter gir høyere driftssikkerhet og færre feil. Dette styrker videre tilliten til systemet og minimerer nedetid.

  Det innføres en tre-trinns tilbakemeldingsmekanisme der:

  1. En kommando sendes fra masteren.
  2. Systemet venter på kvittering fra slaven.
  3. Slaven kvitterer at operasjonen er utført.

  Dersom dette innebærer en kommando fra brukergrensesnittet på toppen, blir det fem trinn, der HMI/SCADA sender en forespørsel til masteren, som utfører trinnene over, og deretter sender bekreftelse til alle nødvendige brukergrensesnitt.

  Dette sikrer at operatøren ikke får falsk informasjon om at en handling er gjennomført.
- **Brukervennlige grensesnitt:** Ved utvikling av intuitive HMI-løsninger som tydelig viser systemets tilstand, får systemoperatører bedre arbeidsflyt, kortere opplæringstid. På sikt vil sikkerhet og kostnadseffektiviteten bedres.
- **Skalerbarhet:** Løsningen skal enkelt kunne utvides med nye slaveenheter og flere tanker på nye eller eksisterende slaver. Dette skal tas hensyn til i alle deler av sluttproduktet. Brukergrensesnittene, KepServerEX og PLS-ene skal alle være konfigurert på en måte som tillater utvidelse av systemet.

Resultatmål

Resultatmålene er de målbare, konkrete reslutatene som prosjektet skal oppnå. Ved å verifisere disse målene vil man kunne fastslå om prosjektet har lyktes eller ikke.

- **Implementert PLS-kommunikasjon:**
  - Oppsett av master- og slaveenhet(er) med robust kommunikasjonsprotokoll over PROFIBUS-DP i GX Works2.
  - Systemet skal håndtere både Ethernet-kommunikasjon for SCADA, HMI og PROFIBUS-kommunikasjon mellom PLS og slaver.
  - Slaveenheter skal håndtere AD/DA-omformere og frekvensomformere over PROFIBUS.
  - Kommunikasjonen skal være tapsfri, og ny data som ikke krever hyppige oppdateringer, skal kunne sendes minst hvert 200 ms.
- **Fungerende SCADA- og HMI-løsninger:**
  - Brukergrensesnitt utvikles i Beijer IX Developer (for operatørpanelet) og Wonderware InTouch med Kepware KEPServerEX (for PC-basert SCADA).
  - Systemet skal følge High Performance HMI-prinsipper for tydelig og effektiv visualisering.
  - Systemet skal ha fleksible tilkoplingspunkter for å muliggjøre utvidelser av systemet.
- **Rykkfri Overgang:** Overgangen mellom ulike regulatortyper og driftmoduser skal være fullstendig rykkfri for å unngå unngå uønskede dynamiske effekter.
- **Reguleringsresultat:**
  - Reguleringen skal oppnå et innsvingningsforløp med *høy nøyaktighet* og *null oversving*.
  - Regulatoren skal være så rask som mulig gitt at stabilitet og presisjon opprettholdes.
- **Fullverdig alarmeringssystem:**
  - Systemet skal ha en integrert alarmering som registrerer og loggfører både vanlig og kritiske feil.
  - Bekreftelse av kommandoer fra slaveenhetene skal loggføres.
  - Tids- og tilstandsalarmer skal overvåkes kontinuerlig.
- **Overvåkning av kommunikasjon:**
  - Profibus-kommunikasjonen mellom master og slave skal overvåkes kontinuerlig.
  - Ved kommunikasjonsbrudd skal slaven automatisk gå i offlinemodus.
  - Kommunikasjonen mellom HMI-operatøren og SCADA-PC skal også overvåkes, og feil skal føre til en sikker tilstand der både master og slave får offline.
- **Skalerbarhet:**
  - Det skal være rom for å utvide løsningen opptil 4 slaver med 3 tanker på hver slave totalt.
  - Hver slave skal konfigureres med 20 PROFIBUS-kanaler[2]. Kanal 0 vil brukes til alarm og 5 kanaler vil tilegnes hver tank for tankdata.
- **Dokumentert trend:**
  - Systemet skal loggføre tanknivåer og andre relevante verdier.
  - Det skal være både sanntids- og historiske trendvinduer i brukergrensesnittene, slik at operatøren kan analysere systemets oppførsel over tid.
- **Dokumenterte testprosedyrer:**
  - Utarbeiding av testskjemaer for regulatoren, hovedsystemet, alarmhåndtering og anleggskommunikasjon.
  - Testskjemaene skal ha tydelige verifiseringspunkter for respons, stabilitet og alarmhåndtering.
  - Hver regulator skal testes i et kontrollert miljø for å verifisere ytelse og stabilitet.

[2]Begrepet 'kanal' refererer til en indeks i arrayet slaven bruker ved PROFIBUS-kommunikasjon. Dette tydeliggjøres i seksjon 3 om kommunikasjon

B Tilgangsnivå i HMI

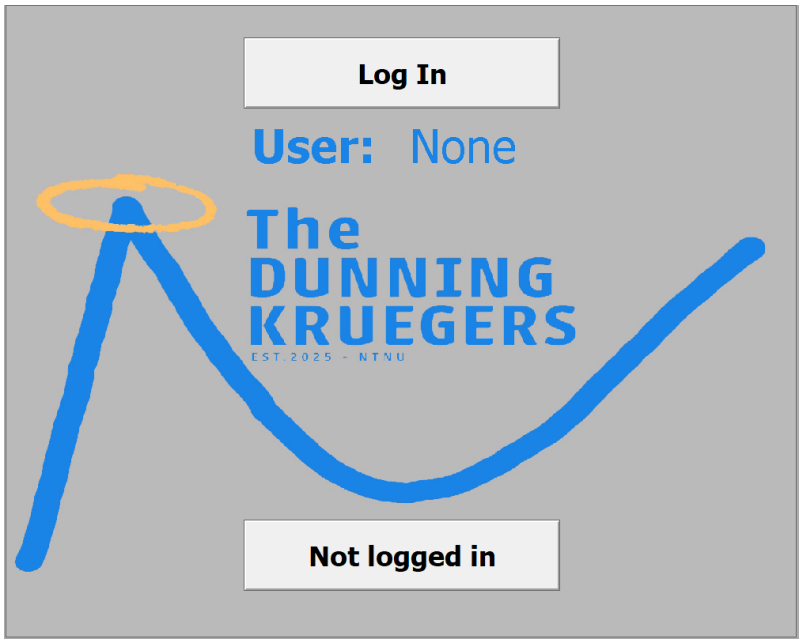
Variabel	**Leses**			**Skrives**		
Operatør	1	2	3	1	2	3
Referanse til nivået	X	X	X		X	X
Nivået i tank	X	X	X			
Pådrag fra regulatoren	X	X	X			
Strømning fra strømningsmåler	X	X	X			
Auto/manuell tilstand	X	X	X		X	X
Manuelt pådrag i manuell modus		X	X		X	X
Valg av regulatortype			X			X
Regulatorparametrene			X			X
Nominelt pådrag i P og PD			X			X
Foroverkoblingsparametrene			X			X
Anti-windup parametere			X			X
Alarmvindu for alarmer	X	X	X			
Kvittering av alarmer					X	X
Real-time trendvindu	X	X	X			
Historisk trendvindu	X	X	X			
Start/stopp pumpe	X	X	X		X	X
Nødstopp	X	X	X	X	X	X
Tibakestill nødstopp	X	X	X			X
Status	X	X	X			

Tilgang til variabler på PC-basert SCADA

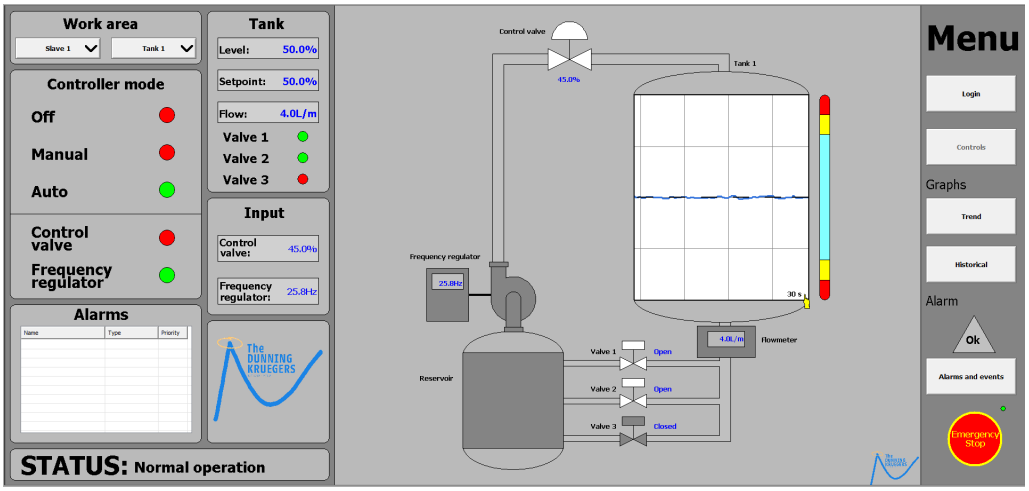
Variabel	**Skrives**	**Leses**
Referanse til nivået	X	X
Nivået i tank		X
Pådrag fra regulatoren		X
Strømning fra strømningsmåler		X
Auto/manuell tilstand	X	X
Manuelt pådrag i manuell modus	X	X
Alarmvindu for alarmer		X
Kvittering av alarmer	X	
Real-time trendvindu		X
Historisk trendvindu		X
Start/stopp pumpe	X	X
Nødstopp	X	
Tilbakestill nødstopp		X

Tilgang til variabler på operatørpanel

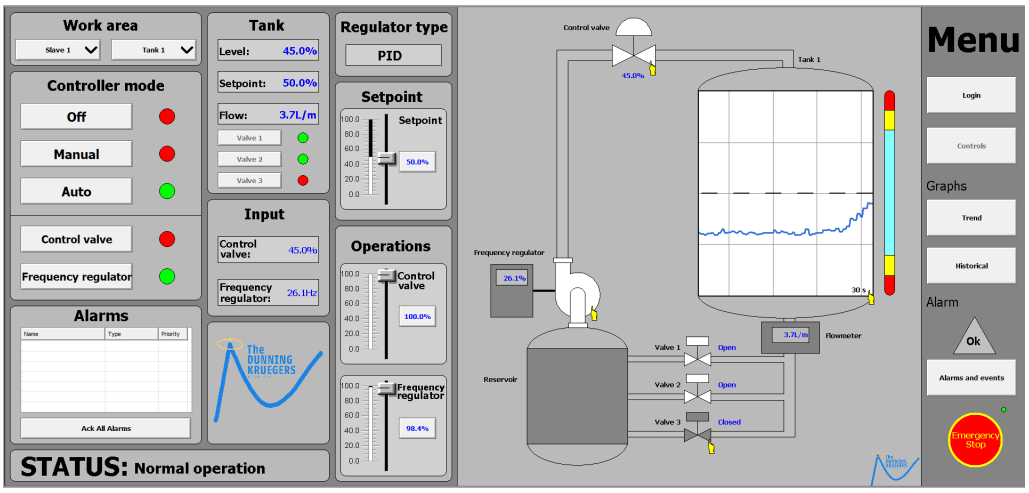
C Grafisk brukergrensesnitt SCADA



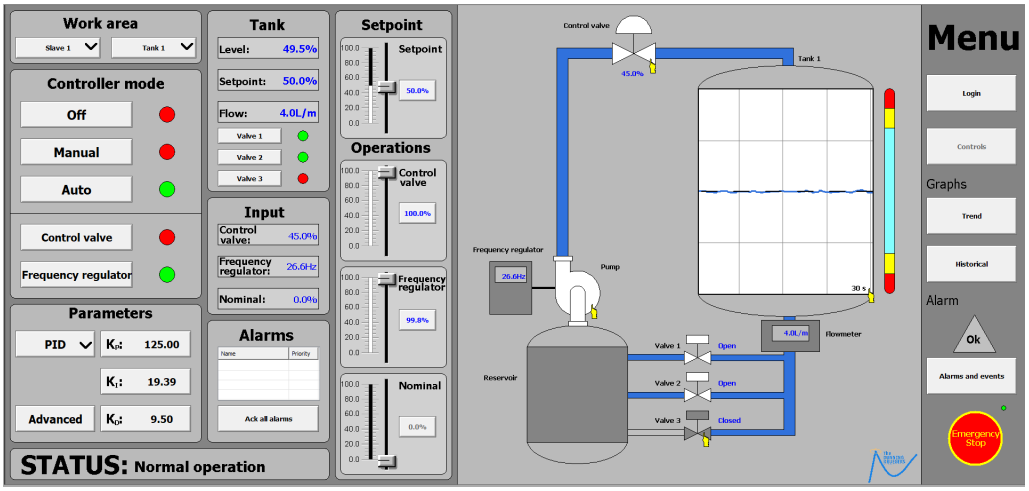
Innloggingsside på InTouch



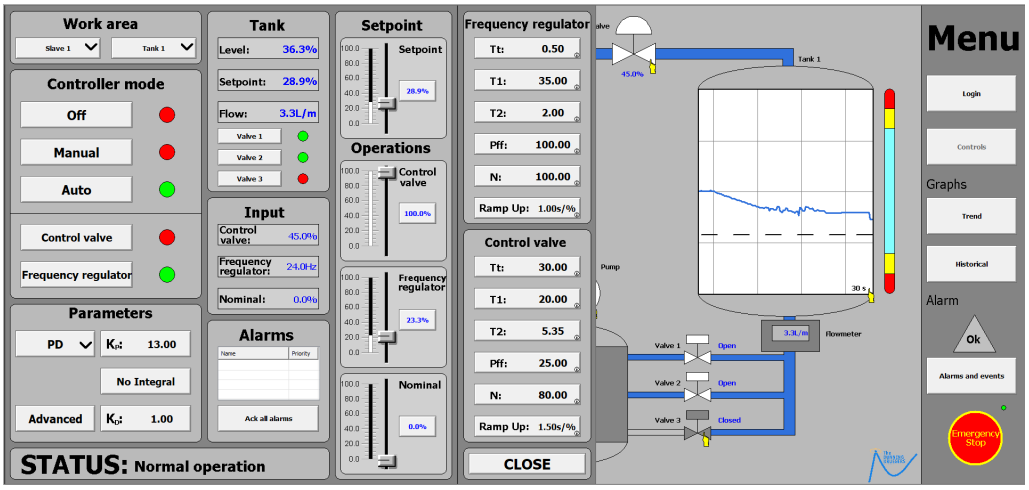
"Controls"-vindu for operatør 1



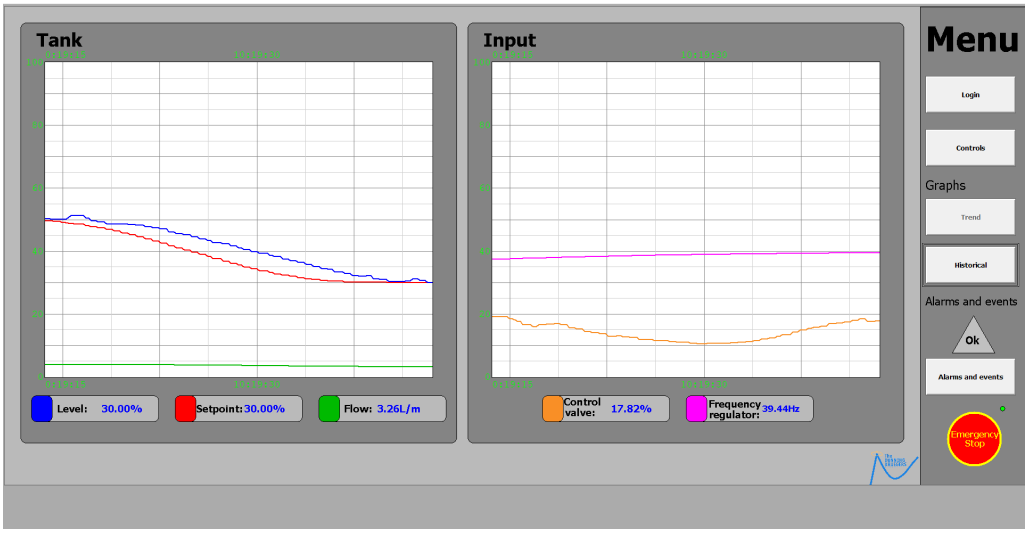
"Controls"-vindu for operatør 2



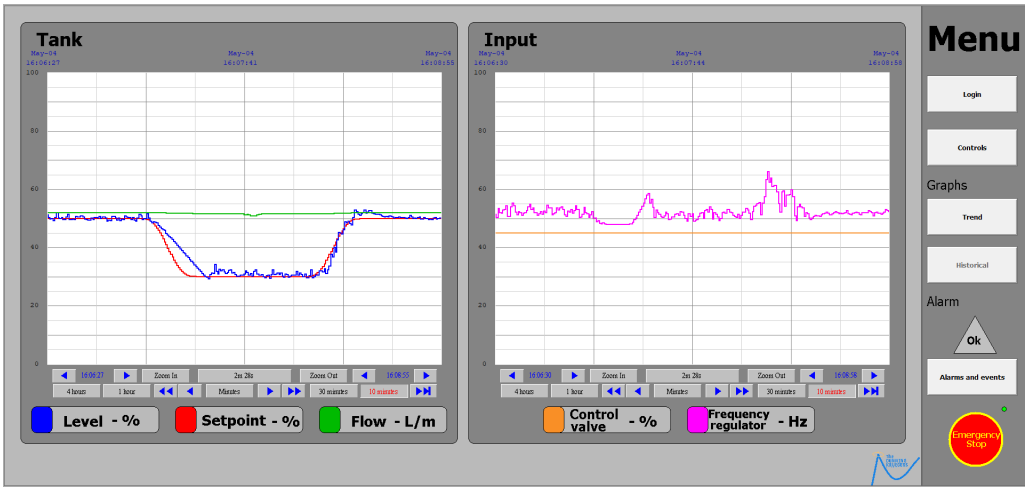
"Controls"-vindu for operatør 3



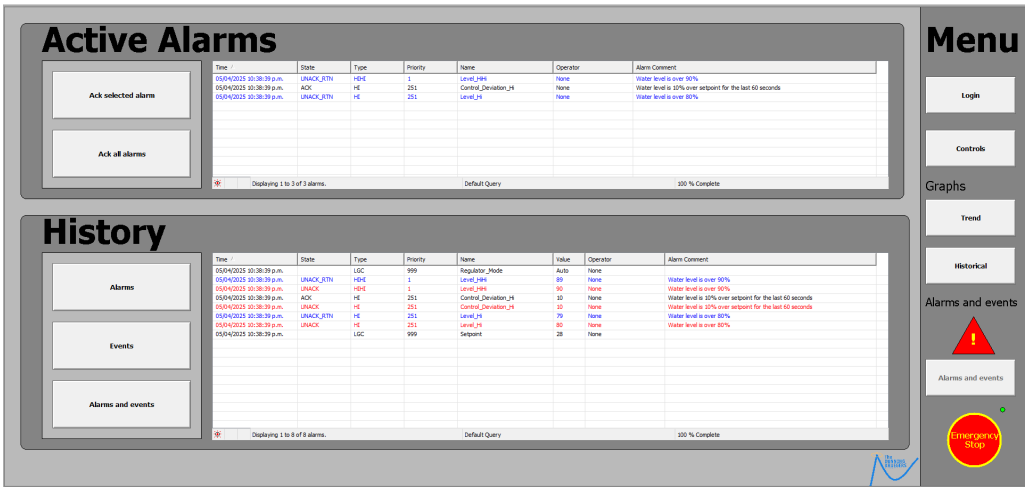
"Controls"-vindu for operatør 3 med avanserte parametere



"Trend"-vindu

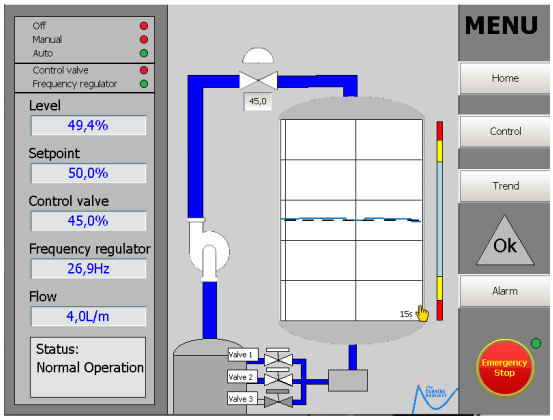


"Historical"-vindu

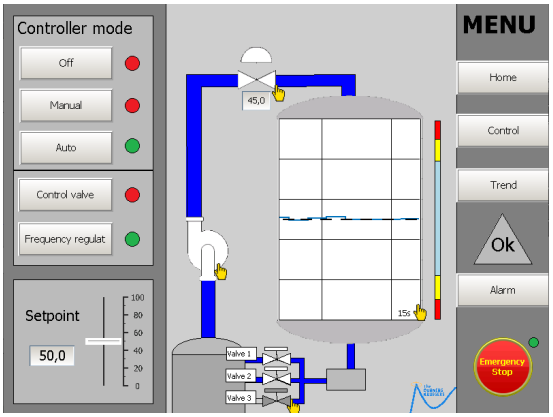


"Alarms and event"-vindu

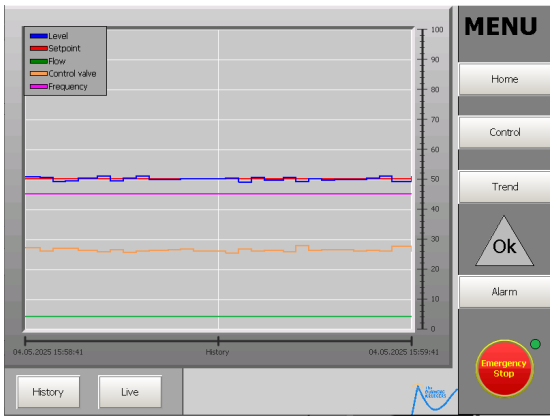
D Grafisk brukergrensesnitt operatørpanel



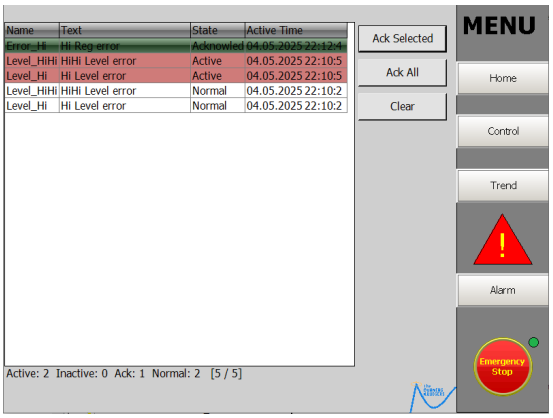
"Home"-vindu på operatørpanelet



"Control"-vindu på operatørpanelet



"Trend"-vindu på operatørpanelet



"Alarm"-vindu på operatørpanelet

E Reguleringsvariable

slave_basic	
Felles parametre	
setpoint	Reg_Freq
reg_mode	mode
Frequency_regulating	
Frekvensomformer	**Magnetventil**
FREQ_PID_KP	MV_PID_KP
FREQ_PID_KI	MV_PID_KI
FREQ_PID_KD	MV_PID_KD
FREQ_PI_KP	MV_PI_KP
FREQ_PI_KI	MV_PI_KI
FREQ_PD_KP	MV_PD_KP
FREQ_PD_KD	MV_PD_KD
FREQ_P_KP	MV_P_KP
FREQ_Ttx	MV_Ttx
FREQ_Nx	MV_Nx
FREQ_Pff	MV_Pff
FREQ_sat_max	MV_sat_max
FREQ_sat_min	MV_sat_min
FREQ_Tustin1	MV_Tustin1
FREQ_Tustin2	MV_Tustin2
FREQ_RampUp	MV_RampUp
FREQ_man_Actuation	MV_man_Actuation

Reguleringsparametre i slave_basic

actuator
Ux
Usat
Uff
UPx
UIx
UDx
MeasuredValue
MeasuredValue_prev
noise
noise_prev

Interne pådragsverdier i actuator

F Evaluering og valg av simuleringsmodell

AG + AE

Dette vedlegget oppsummerer arbeidet med å utvikle og evaluere ulike simuleringsmodeller for vanntanken, med hovedfokus på modelltyper, tilpasningsresultater og begrunnelse for modellvalg.

Modelltyper og tilnærming

Systemet ble forsøkt modellert både analytisk (*Whitebox*) og datadrevet (*Blackbox*). De analytiske modellene basert på kjente fysiske komponenter klarte ikke å gjengi sanntidsdynamikk grunnet forenklinger, som manglende tidsforsinkelse og linearitetsantakelser.

Det ble derfor valgt en datadrevet tilnærming med *System Identification Toolbox* i MATLAB. [9] Sensor- og pådragsdata ble brukt til å trene modeller, hovedsakelig basert på NLARX-strukturer. [5] Disse modellerer systemet som to gjensidig avhengige delmodeller: én for tanknivå og én for utstrømning.

Testede modeller

Følgende modelltyper ble testet:

- **Lineære modeller:** ARX, TF (overføringsfunksjon), og SS (tilstandsrom)
- **Ikke-lineære modeller:** NLARX med henholdsvis lineær output, wavelet-nettverk og sigmoidnettverk

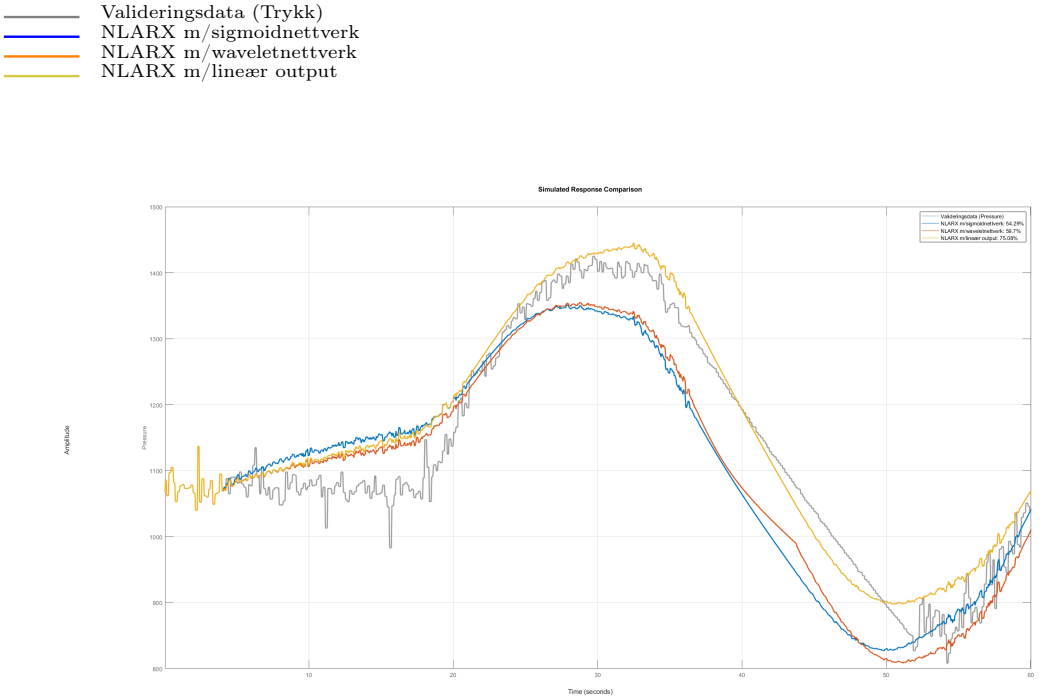
Alle modellene ble trent på identiske step-respons-data og validert mot et separat datasett. Evalueringsmetoden var forklart varians (R^2) samt visuell inspeksjon i tidsdomenet.

Modell	**Treningsdata (%)**	**Valideringsdata (%)**
TF (lineær)	81,7	68,5
ARX (lineær)	78,2	65,3
SS (lineær)	80,0	66,1
NLARX (lineær)	80,6	**75,0**
NLARX (wavelet)	72,0	56,7
NLARX (sigmoid)	**83,0**	54,2

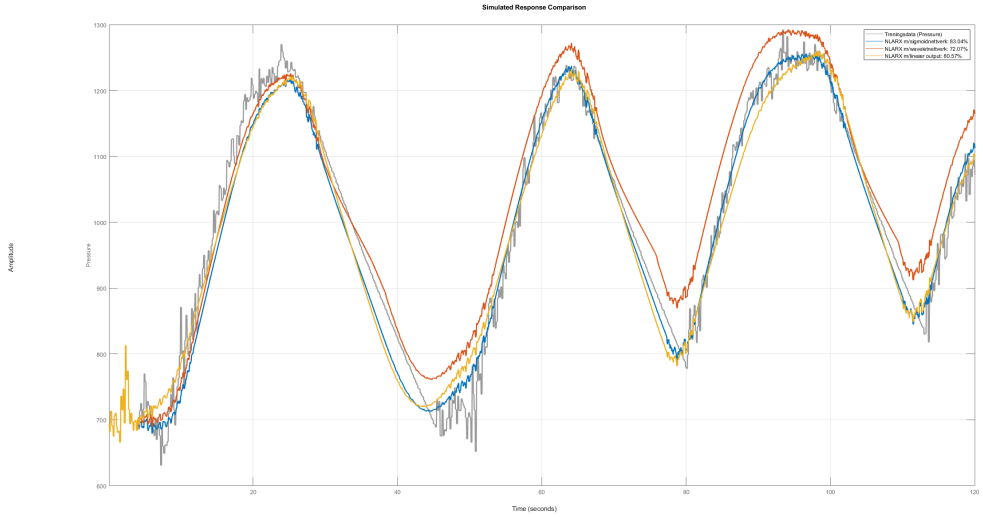
Forklart varians (R^2) for ulike modeller

Resultater og vurdering

Figurene under viser hvordan de ulike NLARX-modellene presterte sammenlignet med måledata under validering og trening. Den lineære NLARX-modellen (gult) gir en nær respons til målesignalet og demonstrerer best generalisering.



Validering av NLARX-modeller mot trykksignal.



Treningsresultater for NLARX-varianter.

De mer komplekse modellene, særlig sigmoidnettverket, overtilpasset treningsdata og hadde svak generalisering. Den **lineære NLARX-modellen** oppnådde høy forklart varians både i trening (80,6 %) og validering (75,0 %), og ble derfor valgt som simuleringsmodell for både nivå og forstyrrelse.

De lineære modellene presterte rimelig godt på treningsdata, men klarte ikke å generalisere til valideringsdata. Ikke-lineære modeller med wavelet- og sigmoidnettverk viste høy treningspresisjon, men led av betydelig overtilpasning.

Den **lineære NLARX-modellen** oppnådde høy forklart varians både i trening (80,6 %) og validering (75,0 %). Den kombinerte robusthet og presisjon uten å overtilpasse, og ble derfor valgt som simuleringsmodell for både nivå og forstyrrelse.

Valg av modell og implementering

Valgt modellstruktur ble NLARX med lineær output-funksjon med egne modeller for nivå og utstrømning. Modellen er egnet for sanntidssimulering, gir realistiske responser, og er direkte kompatibel med PLS-miljøet via OPC. Resultatet er en troverdig digital tvilling som muliggjør testing og tuning av regulatoren uten tilgang til fysisk rigg.