HØGSKOLEN I ÅLESUND

Aalesund University College

Bacheloroppgåve

Emnekode og emnenamn

IE303612Bachloroppgave

Tittel på oppgåva

Ulstein IAS Simulator

Kandidatnummer

1808 og 1823

Totalt antall sider inkludert framsida: 51

Innlevert Ålesund, 29.05.2015

HØGSKOLEN I ÅLESUND

Aalesund University College

Obligatorisk egenerklæring/gruppeerklæring

Den enkelte student er selv ansvarlig for å sette seg inn i hva som er lovlige hjelpemidler, retningslinjer for bruk av disse og regler om kildebruk. Erklæringen skal bevisstgjøre studentene på deres ansvar og hvilke konsekvenser fusk kan medføre. **Manglende erklæring fritar ikke studentene fra sitt ansvar**.

Du/dere fyller ut erklæringen ved å klikke i ruten til høyre for den enkelte del 1-6:		
1.	**Jeg/vi erklærer herved at min/vår besvarelse er mitt/vårt eget arbeid, og at jeg/vi ikke har brukt andre kilder eller har mottatt annen hjelp enn det som er nevnt i besvarelsen.**	☒
2.	**Jeg/vi erklærer videre at denne besvarelsen:** • ikke har vært brukt til annen eksamen ved annen avdeling/universitet/høgskole innenlands eller utenlands. • ikke refererer til andres arbeid uten at det er oppgitt. • ikke refererer til eget tidligere arbeid uten at det er oppgitt. • har alle referansene oppgitt i litteraturlisten. • ikke er en kopi, duplikat eller avskrift av andres arbeid eller besvarelse.	☒
3.	**Jeg/vi er kjent med at brudd på ovennevnte er å betrakte som fusk og kan medføre annullering av eksamen og utestengelse fra universiteter og høgskoler i Norge, jf. Universitets- og høgskoleloven §§4-7 og 4-8 og Forskrift om eksamen §§30 og 31.**	☒
4.	**Jeg/vi er kjent med at alle innleverte oppgaver kan bli plagiatkontrollert i Ephorus, se Retningslinjer for elektronisk innlevering og publisering av studiepoenggivende studentoppgaver**	☒
5.	**Jeg/vi er kjent med at høgskolen vil behandle alle saker hvor det forligger mistanke om fusk etter høgskolens studieforskrift §30**	☒
6.	**Jeg/vi har satt oss inn i regler og retningslinjer i bruk av kilder og referanser på biblioteket sine nettsider**	☒

Publiseringsavtale

Studiepoeng: 2 kandidatar, totalt 40 studiepoeng.

Rettleiar: Rune Volden, Hans Støle, Anders Sætersmoen

Fullmakt til elektronisk publisering av oppgaven

Forfatter(ne) har opphavsrett til oppgaven. Det betyr blant annet enerett til å gjøre verket tilgjengelig for allmennheten (Åndsverkloven §2).
Alle oppgaver som fyller kriteriene vil bli registrert og publisert i Brage HiÅ med forfatter(ne)s godkjennelse.
Oppgaver som er unntatt offentlighet eller båndlagt vil ikke bli publisert.

Jeg/vi gir herved Høgskolen i Ålesund en vederlagsfri rett til å gjøre oppgaven tilgjengelig for elektronisk publisering:	☒ **ja**	☐ **nei**
Er oppgaven båndlagt (konfidensiell)? (Båndleggingsavtale må fylles ut)	☐ **ja**	☒ **nei**
- Hvis ja: **Kan oppgaven publiseres når båndleggingsperioden er over?**	☐ **ja**	☐ **nei**
Er oppgaven unntatt offentlighet? (inneholder taushetsbelagt informasjon. Jfr. Offl. §13/Fvl. §13)	☐ **ja**	☒ **nei**

Dato: **26.05.2015**

Forord

I denne bacheloroppgåva tek ein føre seg simulering av delsystem opp mot Ulstein IAS(Integrated automation system). Ulstein IAS simulatoren er eit verktøy for verifisering og feilsøking på Ulstein IAS.
Oppgåva er forma saman med ingeniør og utviklings avdelinga hjå UPC(Ulstein Power & Control).
I forkant av prosjektet er det utarbeida ein forprosjektrapport der beskriving av bacheloroppgåva og retningslinjer for oppgåva er lagt fram. Vidare er det jobba med oppgåva i tråd med framdriftsplan og dei retningslinjer som er sett.
Til slutt vil det rettast ei takk til Rune Volden hjå UPC for god rettleiing og spennande diskusjonar gjennom prosjektet. Vidare ei takk til Stian Lid Kleppe, Jogeir Emil Gjerdsdal, Kai Verner Røsvik og Arne Johan Helle hjå UPC for god fagleg oppfølging og raske konkrete svar under vegs.
Ved HIALS(Høgskolen i Ålesund) ei takk til rettleiarane Anders Sætersmoen og Hans Støle for gode råd og konstruktive tilbakemeldingar gjennom heile prosjektet.

Samandrag

Ulstein IAS(Integrert automasjonssystem) simulator er eit produkt utvikla som eit verktøy for fabrikktesting(FAT) av Ulstein IAS. Simulatoren vil vere med på å kvalitetssikre det ferdige IAS produktet til Ulstein. På dagens FAT er det delar av systemet som ikkje kan testast på grunn av manglande verktøy. Ved å ta i bruk Ulstein IAS simulator vil Ulstein også kunne teste disse systema, og dermed avdekke eventuelle feil på eit tidlegare tidspunkt. Ulstein IAS simulator er ei hovudoppgåve hjå Høgskolen i Ålesund og er utvikla i samarbeid med Ulstein Power & Control.
Simulatoren er programmert i Java og tek føre seg fire delsystem: MRU, tankpeiling, pumpe og testing av analoge og digitale IO. Brukargrensesnittet til simulatoren oppretta i ei GUI (Graphical user interface) ramme som er inndelt i faner, der kvart delsystem har si fane. Dette gjer systemet oversiktleg og enkelt å ta i bruk. MRU-simulatoren skal simulere skipets rørsle ved generering av sinuskurver. Tankpeiling-simulatoren gir ut volumet av veska i tanken og kompenserar for rørsla til skipet. Med pumpe-simulatoren kan ein simulere tre forskjellige type pumper. Pumpe med ei hastigheit, pumpe med to hastigheiter og frekvensstyrt pumpe. Simulatoren for testing av analoge og digitale I/O er sett opp for funksjonstesting av I/O.
Ulstein IAS simulator er programmert for å enkelt kunne utvidast eller legge til simulatorar for fleire delsystem. Kvart delsystem er programmert som sjølvstendige system, og deretter lagt til som faner i sjølve simulatoren. Simulatoren opererer i sanntid og er programmert som eit multitrådsystem. Simulatoren er knytt opp mot ein Wago kontroller med tilhøyrande analoge og digitale IOar. MRU-simulator som gir ut informasjon via seriellkommunikasjon på NMEA 0183- og EM3000-protokoll. Mellom Java-applikasjon og kontroller kommuniserast det ved bruk av Modbus-TCP.

INNHALD

TERMINOLOGI

Begrep

Redundans	Sett i system er det slik at om ein komponent eller ein del av systemet feilar, vil eit anna ta over[1].
LBP	LBP(Length Between Perpendiculars) er lengda på skipet langs vasslinja. Frå akterskip til baug.
Kontroller	Programerbar logisk styring(PLS)

Symbol

ρ	Rho: Massetettleik kg/m3
Pa	Pascal, måleeining for trykk
h	Høgd i meter
g	Gravitasjon ms2
°	Grader

Forkorting

IAS	Integrated automation system
HMI	Human machine interace
SCADA	Supervisory control and data acquisition
DNV	Det Norske Veritas
MAC	Media access control
I/O	Input/output
UPC	Ulstein Power & Control
FAT	Factory Acceptense Test
GUI	Graphical User Interface
MRU	Motion Refferance Unit
DOF	Degrees of freedom
DP	Dynamic positioning
OOP	Objektorientert programmering
PC	Personal computer
NMEA	National Marine Electronics Association
API	Application Programming Interface

INNLEIING

Denne oppgåva er gitt av UPC. Formålet med oppgåva er å lage ein simulator som skal brukast i samband med FAT av Ulstein IAS. FAT er ein del av klassegodkjenninga av IAS, og blir utført på lab hjå UPC i Ulsteinvik. Sidan denne testen blir utført på eit tidleg stadie og ikkje om bord i skip så er det fleire segment som ikkje kan testast. Det gjer at feil kan dukke opp så seint som ved oppstart om bord i skip, noko som kan vere tidkrevjande å rette opp i og føre til unødvendige kostnadar. Ved å simulere desse manglande segmenta allereie ved FAT så kan desse feila utelukkast på eit tidlegare stadie. Simulatoren vil dermed føre til betre kvalitetssikring av FAT.
UPC utviklar og produserer elektroniske system for skip. I dag leverer dei brusystem, automatiserte fjernstyrings og overvakingssystem, powersystem og integrerte kommunikasjonssystem til skip over heile verda. I Noreg har UPC avdeling i Ulsteinvik og Fremmerholen i Ålesund. [2]
Oppgåva er utvikla i samarbeid med sentrale personar som jobbar med Ulstein IAS hjå UPC. Kravspesifikasjonane til oppgåva er sett på bakgrunn av feil dei har erfart frå tidlegare skip. Endeleg skildring av oppgåva er laga i samarbeid med rettleiar frå UPC.
Problemstilling:

- Realisere og integrere ei metode for å sikre tiltenkt funksjonalitet til Ulstein IAS, ved å teste fullstendige system før levering frå fabrikk.

Til denne problemstillinga blei fylgjande resultatmål sett:

- Simulere digitale og analoge signal for generell testing.
- Simulere to pumper med fylgjande signal: running, remote, start, stop og standby. Med simulering av feil og korrekt tilbakemelding på signal til IAS.
- MRU simulering (skipets rørsle i sjøen).
- Funksjonstesting av tankpeiling ved simulering av trykkgivarar. Simulatoren slår opp i tanktabellar og les signal frå MRU simulator for korrekt gjenskaping av ein ideell situasjon for tankpeiling.

Gjennom denne rapporten går det fram korleis oppgåva er løyst. Kapittel 2 Teoretisk grunnlag, går gjennom grunnleggande teori som har vore nødvendig for å kome fram til målet. Nødvendig utstyr og viktig metodikk går fram i kapittel 3, Material og metode. Endeleg løysing blir forklart i kapittel 4, Resultat, her blir simulatoren gjennomgått punktvis opp mot resultatmåla. Gjennom prosjektet har fleire faktorar vore diskutert både når det gjeld funksjonalitet og design, dette har blitt samla i kapittel 5, Drøfting. Til slutt i kapittel 6, kjem ein konklusjon som gir eit synspunkt på den ferdige oppgåva.

TEORETISK GRUNNLAG

Ulstein IAS

Ei automasjonsløysing for maritime offshore farty. Systemet er i hovudsak eit fjernstyringssystem som integrerar mykje av den tekniske drifta til fartyet. Frå det grafiske brukargrensesnittet til Ulstein IAS kan ein overvake og kontrollere system som generator, hovudmotorar, propell- og thrustersystem. I tillegg til dette er systemet kopla opp mot ei rekke hjelpesystem for alarmovervaking og fjernstyring. Systemet er ei redundant løysning som sikrar driftsikkerheit for brukar. Ein Ulstein IAS kan variere i storleik. [3]
Vidare i dette kapittel vert det sett på komponentane til Ulstein IAS samt relevante system kopla opp mot IAS.

Server, Database og Klient

Ulstein IAS har minimum 2 serverar som opererar uavhengig av kvarandre. På kvar av serverane ligg fullstendig SCADA(Supervisory control and data acquisition). Ved bruk av 2 serverar opprettheld systemet redundans.
I databasen til systemet ligg konfigurasjonar, tabellar og trendhistorikk.
Klientar er brukerstasjonar(HMI) som er uavhengig av kjernen til IAS. Desse er ikkje ein del av det redudante systemet, men likevel har dei ei viktig rolle med tanke på tilgjengelegheit for brukar der denne er plassert.[4]

Hovudkabinett

Det er minimum to hovudkabinett i eit Ulstein IAS. I kvart hovudkabinett er det ein kontroller. Kontroller i kabinett A er lik kontroller i kabinett B. Desse kontrollarane er kopla saman via det lokale IAS Ethernet nettverket. I tillegg til dette er kontrollerane kopla saman via lukka internbus. kontrollerane opererar i par der den eine står som master og den andre står som slave. Masterkontroller er den som har kommando. Slave er backup og har ingen påverknad på systemet. Ved eventuell feil på master vil ein få eit master/slave bytte. Då vil slave bytte rolle og ta på seg masterrolla. Dermed opprettheld en drift på systemet. [4]

U1: Ethernet Switch A
U2: Ethernet Switch B

U22: Brannmur for Ekstern tilkopling
U51: Modbus til Ethernet omformer
U41: Seriell til Ethernet omformer

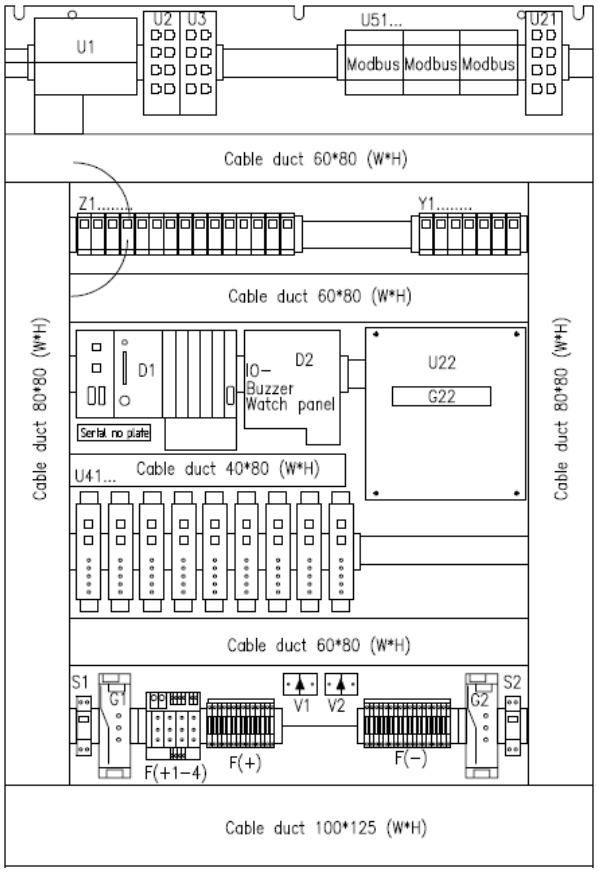
F1-4: Elektronisk sikring

G1: Straumforsyning 1 230Vac til 24vDC
G2: Straumforsyning 2 230Vac til 24vDC

D1: kontroller
D2: Buscoupler IO buzzer Watch Panels

S1: 230Vac sikring
S2: 230Vac sikring

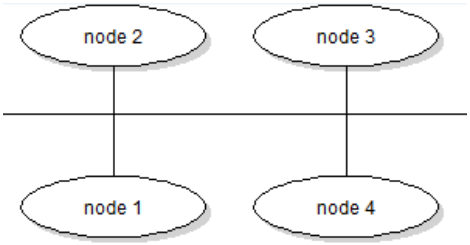
V1: Sperrediode
V2: Sperrediode
[4]



Figur 1: Ulstein IAS Main Cabinet[4]

I/O Kabinett

På eit offshore farty er dei ulike hjelpesystema plassert på ulike posisjonar som kan vere langt frå hovudkabinetta. For å gjere kabelkostnadar meir kostnadseffektive og montering enklare, har Ulstein IAS I/O kabinett. Desse fungerar som noder i systemet og samlar inn data frå hjelpesystem lokalt for vidare distribusjon via det lokale IAS Ethernet nettverket. I/O kabinetta fungerar som ei forlenga arm til hovudkabinettet.[4]



Figur 2: Illustrering av nodenettverk

Eit standar I/O kabinett består av følgjande komponentar:

U1: Ethernet Switch A
U2: Ethernet Switch B

U41: Seriell til Ethernet omformer
U141: Seriell til Ethernet omformer

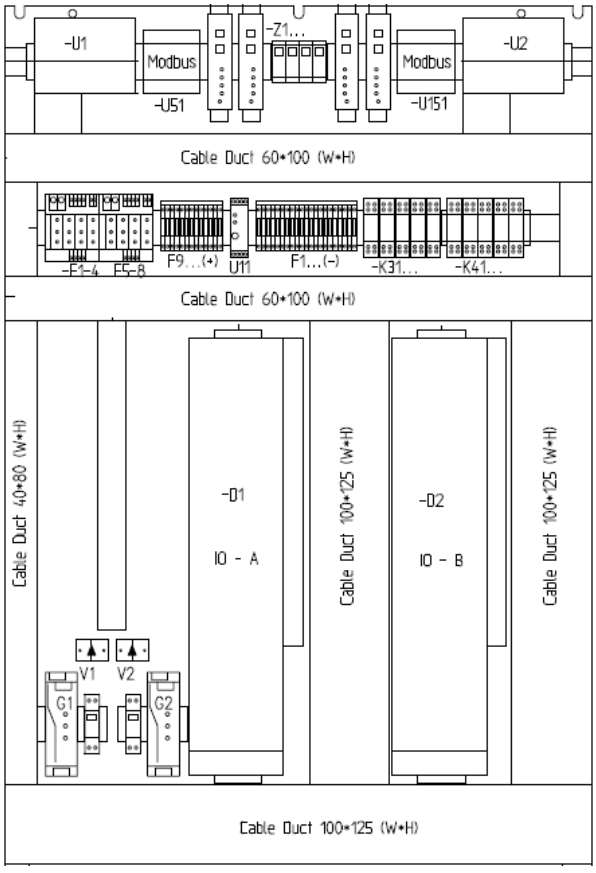
U51: Modbus til Ethernet omformer
U151: Modbus til Ethernet omformer

F1-8: elektronisk sikring
K31-K41: Relè

G1: Straumforsyning 1 230Vac til 24vDC
G2: Straumforsyning 2 230Vac til 24vDC

V1: Sperrediode
V2: Sperrediode

IO-A: Buscoupler A
IO-B: Buscoupler B
[4]



Figur 3: Ulstein IAS I/O Cabinet[4]

Lineær Interpolering og Ekstrapolering

Figuren under viser temperatur i forhold til tidspunkt. Ynskjer ein å vite kva temperaturen var klokka 12:00 kan tabellen gi oss desse data ved direkte oppslag. Men vil ein vite temperaturen klokka 11:00 gir ikkje tabellen kjennskap til dette. Metoden lineær interpolering kan gi oss i tilnærming av temperaturen klokka 11:00 ved å teikne ei rett linje mellom dei kjente nabomålepunkta.

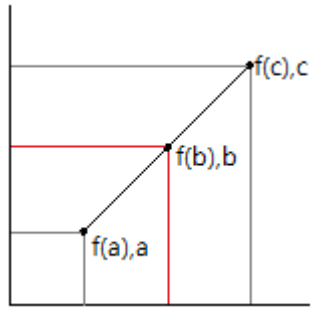
TEMP	TID
10	10:00
11	12:00
14	14:00
14	16:00
13	18:00
8	20:00
7	22:00
2	00:00

Tabell 1: Eksempel temperaturmålingar

Ved oppslag i tabellar kan ein nesten alltid bruke interpolasjon. Dette på grunn av at tabellar vanlegvis er laga med så liten differanse mellom opplista verdiar for den frie variabelen, at lineær interpolasjon mellom to naboverdiar fungerer.

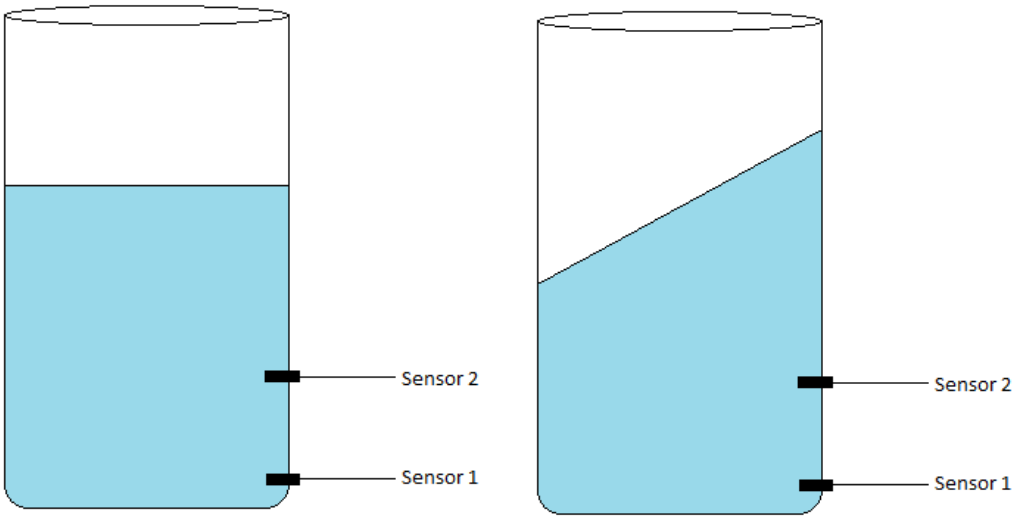
Prinsippet for lineær interpolasjon: Definisjonsmengda *a*,*b* og *c* antek en tilhøyrar funksjonen *f*. *f(a)* og *f(c)* kjenner ein funksjonsverdiane til, men *f(b)* er ukjend. Ved bruk av toppunktsformelen for rett linje, vil den rette linja gjennom punkta (*f(a)* ,*a)* og *(f(c),c)* gi likninga[5]:

$$f(b) - f(a) = \frac{f(c) - f(a)}{c - a}(b - a)$$



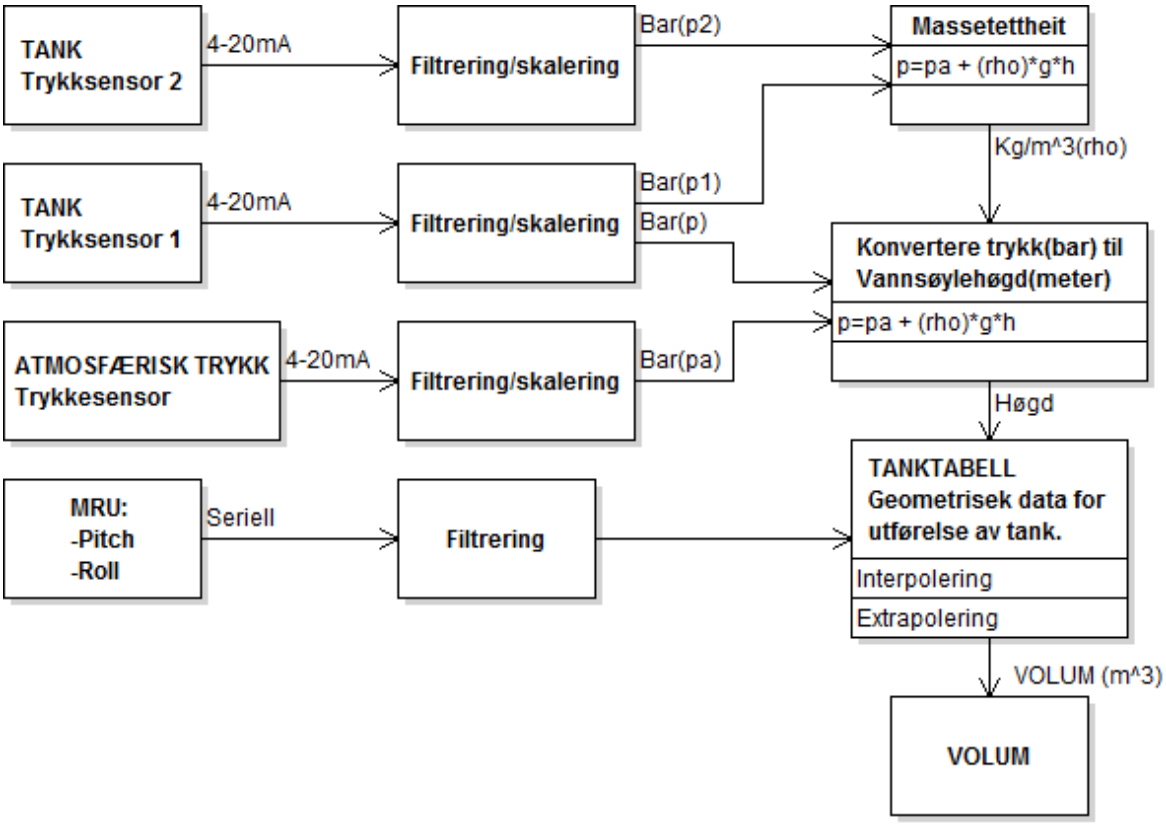
Figur 4:Lineær interpolering

Tankpeilingssystem



Figur 5: Viser tank med innhald. Tanken til venstre viser null grader helling, medan tanken til høgre viser X grader helling. Volum til væska i dei to tankane er likt.

Figur 6 viser flyten i tankpeilingssystemet. Ulstein IAS brukar analoge trykksensorar for måling av trykk i tanken. Sensorane gir ut analog straumsignal(4-20mA). Det analoge signalet vert skalert i forhold til måleområdet for sensor og omgjort til trykk(bar). På same måte finn ein atmosfærisk trykk. Sensor for atmosfærisk trykk er plassert utanfor tanken.



Figur 6: Flytskjema for tankpeilingssystem

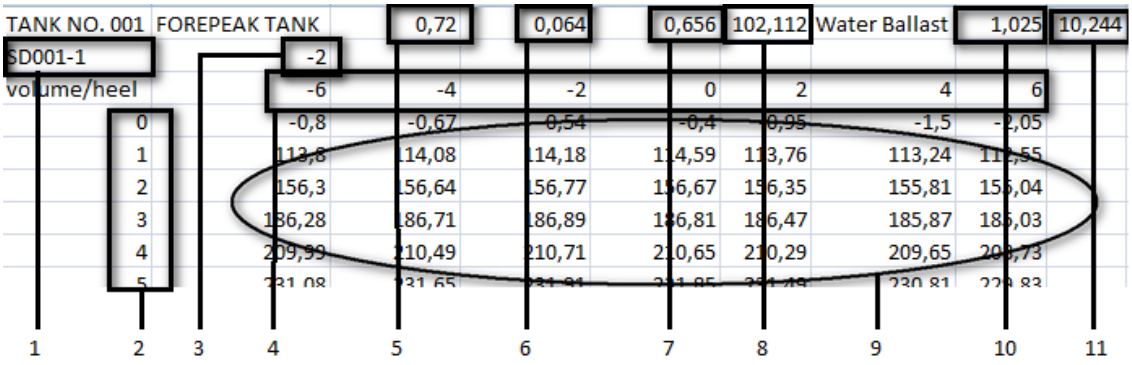
Massetettleik kan settast manuelt eller kalkulerast ved:

1. Måle differansen på trykk mellom sensor 1 og sensor 2.
2. Kalkulere massetettleik (rho) ved pascals lov.

Når ein veit atmosfærisk trykk(Pa), sensor trykk(P), massetettleik(ρ) og gravitasjon(g), finn ein høgda på væskesøyla ved å bruke Pascals lov[6]:

$$P = Pa + \rho * g * h$$

Når ein kjenner til høgda på væskesøyla og pitch/roll til fartyet, avgjer ein volumet til væska ved oppslag i tanktabell. Om verdiane ikkje er eksakt som i tabell vert det nytta interpolering mellom naboverdiane. Det vert interpolert for høgd over sensor, pitch og roll for å finne volum til væska. Er verdiane utanfor tabellområdet, vert det nytta ekstrapolering. Det vert ikkje gjort ekstrapolering utover maksimumsgrenser for volum på tanken.

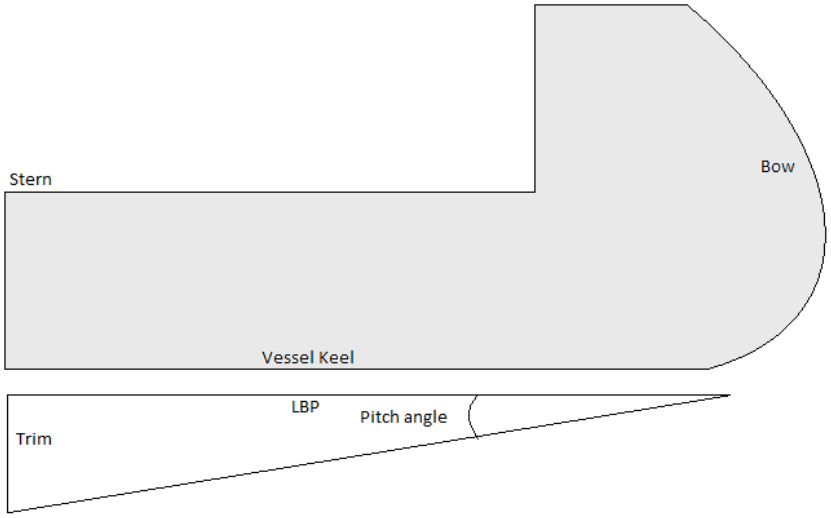


Figur 7: Utklipp frå tanktabell.

1	Sensor namn
2	Volum(m3)
3	Trim (m)
4	Roll(grader)
5	Posisjon til sensor frå botnlinja(m)
6	Differansen mellom lågaste punkt og sensorposisjon(m)
7	Lågaste punkt for tanken frå botnlinja(m)
8	Totalt volum i tank(m3)
9	Høgde i frå sensor(cm)
10	Massetettheit
11	Høgd til tanken(m)

Tabell 2: Forklaring til figur for tanktabell.

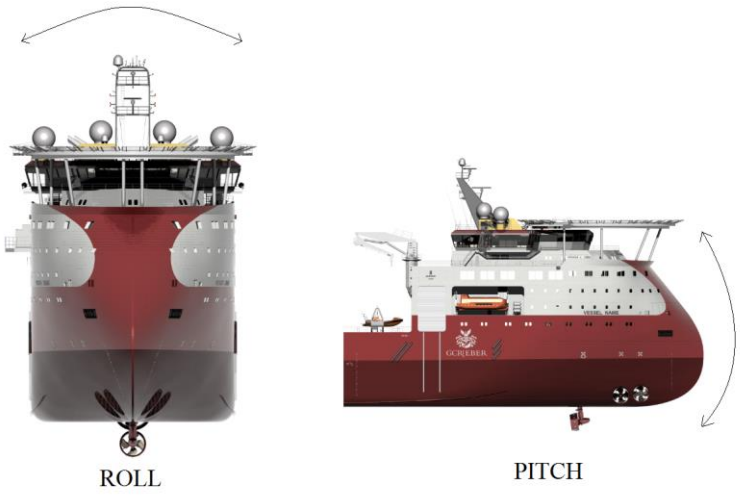
For kvar tank i tanktabellen er der eit sett av verdiar til trim for kvar halve meter. Området er mellom -2 til 2 meter med auking på 0,5 meter. I kvar av desse setta er der verdiar er roll mellom -6 til 6 grader med auking på 2 grader.
For kalkulering av trim vert LBP(Length Between Perpendiculars) brukt. LBP er lengda på fartyet frå roret til vasslinja sitt fremste punkt i baugen. Sett fartyet i profil er trim høgdeforskjellen mellom baug og akterskip[7].



Figur 8: Illustrasjon av fartøyets trim.

Rørslemonitorering av skip

MRU(Motion reference unit) er Kongsberg Maritime sit system for måling av rørsle, i denne samanheng rørsla til eit skip i vatn. Den består av akselerometer og gyroskop som måler skipets rørsle. Måleapparata er vare for støy og gir derfor ut filtrerte signal. Ved å kople til eksterne datakilder som fart og posisjon så kan ein også eliminere denne dataen frå MRUen. Då vil data frå MRU alltid gi ut nøyaktig pitch/roll uavhengig av krefter påverka av skipet sjølv.
Om bord i skip kan ein måle rørsla til skipet i 6 DOF(Degrees of freedom). Pitch og roll blir målt i grader, medan heave blir målt i cm. Kongsberg måler pitch og roll med ei nøyaktigheit på 0.01° og heave med ein nøyaktigheit på 5cm.



Figur 9: Roll og Pitch illustrasjon, grafikk av skip utlevert av Ulstein.

Data frå MRU kan brukast ved fleire anledningar. For eksempel ved kranopperasjonar i røff sjø der ein kan kompensere for bølgene, stabilisere båten i bølgene med stabilisatortankar eller ved DP(Dynamisk posisjonering) operasjonar osv. I IAS simulatoren blir MRU data bruk til tankpeiling for å finne veskemengda i tanken til ei kvar tid uavhengig av rørsle.[8][9]

Kommunikasjon

Seriell EIA-422

EIA-422, tidligare kjent som RS-422,er ein 4-leder balansert data kommunikasjonsprotokoll. Protokollen gir full dupleks, har avgrensingar til ein sender men kan ha mange mottakarar. [10]

Modbus

Modbus er ein master/slave data overførings protokoll utvikla for prosesskontrollsystem. Protokollen er open for offentligheita. Normalt er det brukt seriell kommunikasjon, RS232/422/485 eller nettverkskommunikasjon der Modbusdata vert send i TCP/IP pakkar.

Eit typisk Modbustelegram består av rammen:

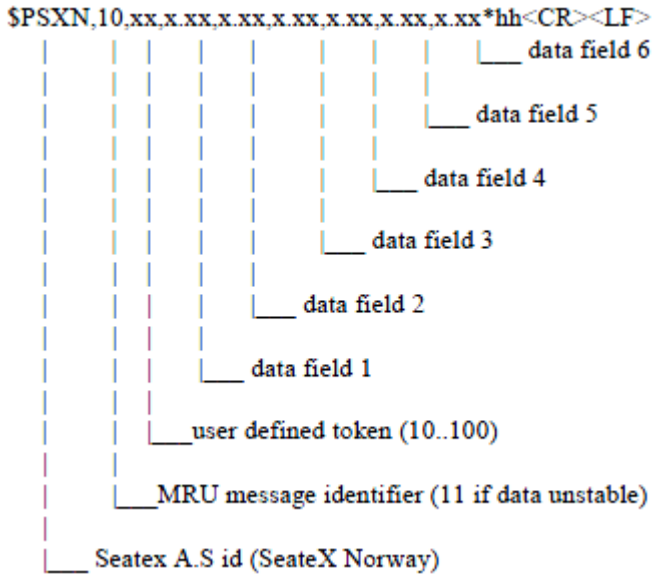
Adressefelt	**Funksjonsfelt**	**Datafelt**	**Errorsjekk felt**
1 byte	1 byte	Variable	2 bytes

Tabell 4: Modbus melding

Det første feltet i ein modbus melding er adressefeltet, som består av 1 byte. I Modbusførespurnad telegrammet sei adressefeltet noko om identiteten til der Modbuførespurnaden rettes mot. Svartelegrammet begynner med adressa til den som responderer til Modbusførespurnaden.
Funksjonsfeltet fortel kva funksjon som skal utførast på kontroller. Om kontroller kan utføre førespurt handling vil den sende eit ekko av den originale meldinga. Elles vil kontroller sende ekko der det mest signifikante bit sett til 1 i funksjonsfeltet, som signaliserer ein unntakstilstand(exception).
Datafeltet varierer lengda ut frå kva funksjon meldinga er sett til. I denne meldinga finn ein data som kontroller kan trenge for å fullfører førespurt funksjon. Responsen inneheld den førespurte data.
Siste feltet inneheld errorsjekk for at eininga ikkje skal utføre handling når det er feil på meldinga. [11]

NMEA 0183

NMEA 0183 er ein kommunikasjonsstandard for overføring av informasjon mellom navigasjon utstyr. Overføring av signal frå Kongsberg Maritime MRU kan gjerast med denne standaren.



Figur 10: NMEA string oppbygging.[9]

Dette er ein string som gir ut opp til seks variablar ilag med eit teikn som er definert av brukar. Om ein skal gi ut mindre enn seks variablar så sett ein desse felta tomme. Float som skal sendast skal skrivast med vitskapleg notasjon (f.eks. 1.987e4 eller -3.589e2), medan integer blir skrivne som tal (f.eks. 12358 eller -654). Vinklar skal vere i radianar. $PSXN viser kva informasjon stringen inneheld, SXN er ikkje ein normal standar innanfor NMEA, men brukt av Kongsberg Seatex i system som dei leverar. Stjerna (*) viser at dette er enden på variablane, teikna etter er sjekksummen som viser at data som er mottatt er dei same som blei sendt. Sjekksummen er i hexadesimalar.[9]

EM3000

Simrad EM3000 er også ein kommunikasjonsprotokoll for maritim bruk. Dette formatet består av ei melding med sett lengde og ingen sjekksum. Meldinga blir bråte ned i bytes. Dataelementa brukar single-bytes unsigned, 2-bytes unsigned og 2-bytes to komplement integer. To komplement er ein måte å representere negative tal i bitform. For to bytes elementa blir least significant byte sendt først. [9]

Element	Scaling	Format	Bytes	Value
Status byte		Unsigned	1	
Header		Unsigned	1	90 Hex
Roll	0.01 degrees	Integer	2	-17999 to 17999
Pitch	0.01 degrees	Integer	2	-17999 to 17999
Heave	1 cm	Integer	2	-999 to 999
Heading	0.01 degrees	Unsigned	2	0 to 35999

Tabell 5: EM3000 oppbygging.[9]

Programmering

Objektorientert programmering

OOP(Objektorientert programmering) er ein definert framgangsmåte for programmering og har røter heilt tilbake på 60-talet då to nordmenn utvikla det første programmeringsspråket(Simula 67) som støtta OOP.
OOP implementerer modellar frå verkelegheita på ein modulbasert måte. Ein samlar bestemte oppgåver i kvar sine objekt som igjen til slutt utgjer det fullstendige programmet. Kvart objekt har si oppgåve og kan samarbeide med kvarandre og veksle informasjon. På denne måten unngår ein også kodeduplisering. Skal ein endre på noko i koden så treng ein gjere det kun i eit objekt.
Funksjonaliteten til eit objekt blir samla i klassa til objektet. Når klassa blir kalla i run-time så blir objektet av klassa oppretta. Dette objektet kan bruke og manipulere data frå denne klassa. Ein kan også opprette fleire objekt ut frå ei klasse. Kva objekt som skal opprettast blir bestemt i den endelege applikasjonen som skal kjøre programmet, her blir også nødvendig input til objektet sett opp.[12]

Fri kjeldekode

Fri kjeldekode, eller open source på engelsk, er kjeldekode som ligg fritt tilgjengeleg på nettet for den som måtte ynskje å nytte seg av den. Brukar som lastar ned fri kjeldekode står fritt til å bestemme kva den skal brukast til, enten på eigen PC, kopiere den eller distribuere den vidare. Ein står også fritt til å endre og forbetre kjeldekode slik ein ynskjer. Brukar ein fri kjeldekode i utviklinga av ein teneste kan ein gjere det utan noko godkjenning eller betaling av lisens. Det gjeld både om ein tek pengar for tenesta eller ikkje.
Sjølv om ein i teorien kan gjere kva ein vil med fri kjeldekode så må ein fylgje nokre spelereglar. «Copyleft» er ein regel som seier at når du distribuerer fri kjeldekode vidare kan du ikkje sette restriksjonar som hindrar den frie kjeldekoda å vere fri. Fri kjeldekode skal alltid flyte fritt og kunne brukast av alle. Altså motsatt av reglane for «copyright». Det same gjeld om ein skal lage ein pakke som fri kjeldekode er ein del av, for eksempel ein applikasjon.[13]

Grafisk brukargrensesnitt (GUI)

Når ein lagar ein applikasjon treng ein noko som snakkar til brukaren og får brukaren til å forstå korleis applikasjonen fungerer. I denne samanheng brukar ein GUI. Dette er ei grafisk framstilling av funksjonaliteten til programmet. Brukaren kan trykke på knappar, velje frå menyar, velje med musa, skrive i tekstfelt og liknande. På denne måten får ein ei enkel forståing utan å vite kor mykje som ligg bak koda.
Korleis GUIen skal sjå ut bestemmer den som programmerar sjølv. Java har ein eigne rammeverk for å lage slike GUIar. Ein har tre hovudbyggesteinar å jobbe ut frå; komponentar, layout og eventhandtering. Komponentar er alt som er synleg i GUIen og responderer til brukarens handlingar, knappar, tekstfelt, menyar, rullegardiner, panel osv. Desse komponentane kan ein plassere der ein vil inne i eit rammeverk etter korleis ein vel å designe applikasjonen. Ein har metode for inndeling av rammeverket, dette er også ein type komponent, med desse delar ein bildet i fleire bolkar som ein samlar komponentar i. Med layout bestemmer ein korleis komponentane skal sjå ut. Her endrar ein størrelse, farge, tekst osv. på komponentane i applikasjonen. Layout tek også føre seg utsjånaden og storleiken på komponentane som deler opp bilete. For eksempel blir ofte bilete delt inn konteinarar plassert etter kompassretningane nord, sør, aust, vest og senter.
Eventhandtering lyttar på alle komponentane i applikasjonen og utfører ein opperasjon når

ein brukar for eksempel trykker på ein knapp. Opperasjonen som skal utførast blir handtert i koda når eventet skjer. I dette prosjektet er Java Swing brukt til å lage GUI. Swing er eit bibliotek i Java som fungerer som ei verktøykasse og inneheld det ein treng for å kunne lage ein GUI.[14]

Tråd

Alle program i Java blir kjørt som ein tråd (Thread), det går fint heilt til ein har hendingar som skal inntreffe samtidig eller skal ha operasjonar som skal gå uavhengig. I dette prosjektet har ein ei metode som skal generere to uavhengige sinussignal samstundes ut frå ei og same klasse. For å få til dette må ein opprette to objekt og kjøre dei som trådar.
Ein kan sette opp ei klasse for å kjøre som tråd på to måtar. Den eine måten er å utvide ei klasse til å verte ei subklasse av tråd (extends Thread). Klassa må ha ei run() metode der ein sett opp kva som skal utførast når tråden blir kjørt. Denne run() metoden vil overstyre run() metoden til tråd. Klassa kan bli oppretta som eit objekt og startast som ein tråd. Den andre måten å sette opp ei klasse som tråd på er å implementere klassa i Runnable grensesnittet. Klassa må også då ha ei run() metode, denne blir implementert i Runnable. Når ein opprettar objekt som trådar, er det viktig å opprette den tråden med høgst prioritet først. Den første tråden vil få første prioritet til å kjøre om ei konflikt skulle dukke opp. Det er derfor viktig at metode som fleire trådar har tilgang til er synkroniserte, det vil sei at kun ein tråd har tilgang til metoden i gangen. Metoden vil dermed vere trådsikker. Når trådane er oppretta kan ein starte og stoppe dei slik ein vil. [15]

Sanntid

Omgrepet sanntid (real-time) er ofte brukt i samanheng med data og programmering. Ein seier at systemet opererer i sanntid. Programmet garanterer respons innanfor ei gitt tidsramme, ein deadline. Denne tidsramma er gitt av den som lagar systemet. Uavhengig av om tidsramma er i millisekund eller dagar så er programmet definert som sanntid. Eksempel på sanntid er tastaturet på datamaskina, når ein trykker ein tast på tastaturet så vil en ha respons på skjermen med ein gang.
I tillegg til prosessorkraft og effektive algoritmar så er sanntidsprogrammering ein vesentlig faktor i eit system som skal fungere i sanntid. Java er eit programmeringsspråk som støttar sanntidsprogrammering og er designa for produktivitet. Her har ein eigne metodar og spesifikasjonar som tilfredsstiller krav for sanntid.[16]

Fabrikk aksept test (FAT)

Fabrikk aksept test refererer til test som vert utført på leverandørens fabrikk, før levering av systemet til sluttbrukar. Testen er ei verifisering på at systemet møter tenkt funksjonalitet. Ein god praksis er å ta ein fullstendig gjennomgang av FAT-prosedyrane i forkant av FAT(pre-FAT). Dette for å avdekke eventuelle feil og rette desse før den verkelege FAT. [1]
Ved utføring av FAT på ein Ulstein IAS leveranse er Ulstein Power & Control, klasseselskap og kunde tilstades.

Klasseselskap, krav til FAT.

Del 4, kapittel 9 "Control and Monitoring Systems" i DNV Rules for Classification of Ships 2015-01, finn ein regelverk for test av system som Ulstein IAS.

I kapittel D102 står følgjande:
Tests in the presence of a DNV surveyor according to 200, 300 and 400 shall be performed at the manufacturers works.

Vidare finn vi krav til integrasjonstesting under D301-d:
Function tests of normal system operation and normal EUC(Equipment Under Control) performance, in accordance with the rules. Function tests are also to include a degree of performance testing outside of the normal operating parameters.[17]

Klasseselskapet DNV Loyds har majoriteten av klassegodkjenningar til Ulstein IAS. Det er utarbeida standard FAT dokument som er grunnlaget for testen.

MATERIALER OG METODE

Dokumentasjon

Forprosjektrapport

Forprosjektrapporten er ei førebuing på hovudprosjektet. I rapporten skal det kome fram ei klar utgreiing av prosjektet. Dette inneber[18]:

- Målsetjing og spesifisering av prosjektet
- Ei oversikt over hovudaktivitetane
- Kva informasjon som må samlast inn
- Internkontroll og evaluering av prosjektet undervegs
- Prosjektorganisering
- Vedlagt framdriftsplan for hovudaktivitetane

GANTT-diagram

Som ein del av prosjektoppfølginga er det teke i bruk Gantt-diagram. Dette diagrammet har dei hovudaktivitetane, brote ned i delaktivitetar langs eine aksen, medan den andre aksen viser tidsløpet. Etterkvart som aktivitetane er starta eller slutta fører ein inn dette i diagrammet. Dette er eit levande diagram som må følgjast opp heile vegen av prosjektet. Ei fordel med ein Gantt-diagram er at det er oversikteleg og lett å forstå.[19]

Avviksrapportering

I forprosjektrapporten er det utgreia om avvikshandtering. Det er utarbeida eigen mal for utfylling av avvik. Vidare er avvika delt inn i tre risikogrupper(lav, middels og høg). Det vert utført ulike tiltak for risikovurderinga til avviket.

Framdriftsrapport

I forbindelse med dette prosjektet har HIALS eigen rapport for oppfølging av framdrift. Denne rapporten er periodebasert. Ein samanliknar planlagde aktivitetar og utførte aktivitetar for inneverande periode. Samstundes legg ein fram aktivitetar for neste periode. Ei periode er i utgangspunktet fjorten dagar, men vert tilpassa etter behov og rådgiving frå veiledarar.

Funksjonsskildring og funksjonstest

Kvar hovudaktivitet i Gantt-diagrammet er ein fullstendig modul i det totale prosjektet. Til eksempel er pumpesimulering ein hovudaktivitet. I starten av kvar hovudaktivitet som går på software utvikling er funksjonsskildring første punkt. Dette er ei utgreiing på kva funksjonar og visualisering som er tenkt til denne modulen. Det er oppretta ein funksjonsskildringsmal som vert brukt som standard.
Etter utvikling og utføring av modulen avsluttar en ved å funksjonsteste i samsvar med funksjonsskildringa. Dette gir kvalitetssikring i forhold til tenkt resultat av modulen. Er der avvik, vert dette handsama ved å bruke avvikshandtering.

Brukarmanual

Brukarmanualen skal skildre korleis Ulstein IAS testingeniør skal kunne bruke simulatoren. Manualen tek føre seg delsystem for delsystem med situasjonsbilete og flytdiagram for beskriving av tilstandar.

Programmeringsguide

Software konstruksjonsmanual er ei skildring på integrering av nye delsystem i Ulstein IAS simulator.

- Implementering av grafisk brukargrensesnitt
- Tek føre seg oppretting av kommunikasjon til kontroller
- Korleis dokumentere I/O
- Korleis opprette og kjøre objekt i hovudfunksjonsklasse.

Sikkerheitskopiering

For sikker lagring av data er det valt å bruke Dropbox som fildeling og lagring for prosjektets dokumenter og filer. Det er oppretta mappe for prosjektet med nummererte undermapper. Det vert teke sikkerheitskopiar av programvare og koder som vert lagra under spesifikk mappe.

Prosjektstyring

Prosjektets organisasjon består av to grupper, prosjektgruppa og styringsgruppa. Prosjektgruppa planlegg og utfører arbeidet, medan styringsgruppa er med på viktige avgjersle i prosjektet og kontrollerar at framdrifta i prosjektet blir oppretthaldt etter planen. Faste møte med styringsgruppa kvar andre/tredje veke gjennom heile prosjektperioden. Til kvart møte er det lagt fram framdriftsrapportar. Disse rapportane inneheld planlagde aktivitetar i forhold til utførte aktivitetar i perioden fram til møtet, vidare plan for neste periode og eventuelle problem. Prosjektet er oppdelt i delmål og underaktivitetar og sett opp i gnatt-diagram. Gnatt-diagram har også blitt fylt ut og lagt fram på styringsmøte for å vise framdrift.
Prosjektet er delt opp 7 hovudmål der kvart av medlemane i prosjektgruppa har sine ansvarsområder. Til dei hovudmåla som omhandlar den praktiske delen av prosjektet er det laga funksjonsskildringar. Disse funksjonsskildringane blir ei base for testrapporten som blir laga når målet er nådd. Dette er med på å kvalitetssikre arbeidet. Eventuelle avvik har vorte handtert i eigne avviksskjema undervegs.

Utviklingsmetodikk

Utviklingsmetodikken i dette prosjektet tek utgangspunkt i fossefallmetoden. Fossefallmetoden tek føre seg eit og eit problem i gangen og utfører dei sekvensielt. Dette gir klare mål å jobbe mot og strukturerer arbeidsprosessen. Som nemnt er dette prosjektet delt inn i 7 hovuddelar med tilhøyrande underaktivitetar. Gjennom prosjektperioden har disse 7 delmåla blitt utført ein etter ein i planlagd rekkefylgje. Delmåla er oppdelt i underaktivitetar som alle er utført etter bestemt framgangsmåte. Kvar aktivitet blei starta med å lage ei funksjonsskildring, deretter oppsett/design, utføring, test og avslutta med ein testrapport. Med dette oppsettet blir det lettare å fylgje planen og ein unngår dermed større endringar midtveis i prosjektet.[19]

Java

Swingworker

Swingworker er ein abstrakt klasse i utgangspunktet for å utføre lange GUI-oppgåver i ein bakgrunnstråd. Der er tre trådar som er involvert i livsløpet til eit Swingworker objekt.

- Current thread: Metoden execute() startar denne tråden. Den legg worker thread i schedular og returnere umiddelbart.
- Worker Thread: metoden doInBackground er kalla på denne tråden. Dette er der all bakgrunnsaktivitet skal utførast. Oppgåver som varsling til propertyChangeListeners om endringar og bruke firePropertychange metoder.
- Event Dispatch Thread: Alle Swing realiterte aktivitetar skjer på denne tråden.

FirePropertychange, ein metode i SwingWorker klassa. Denne rapporterer ein oppdatering til alle registrerte lyttarar. Ingen melding vert sendt viss gammal og ny verdi er lik.
[20]

Swing

Ein pakke som gir eit sett av lettvekts java komponentar for generering av GUI for Java programmering.
Når ein skriv multi-threaded system ved bruk av Swing er det to avgrensinga ein må tenke på.

1. Tidskrevande oppgåver bør ikkje verte kjørt på Event Dispatch Thread. Då vil applikasjonen reagere tregt.
2. Kun Swing komponentar bør få tilgang til Event Dispached Thread.

Jamod

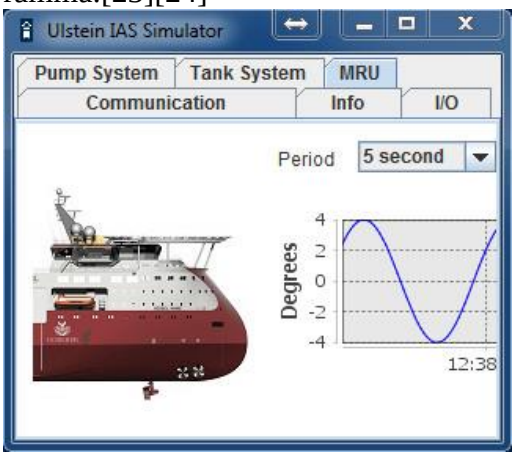
Eit Java bibliotek for implementering av Modbus master/slave. Biblioteket støttar seriellkommunikasjon samt TCP eller UDP. Biblioteket er utvikla som eit gratis, open source bibliotek tenkt til Java utviklarar som treng tilgang til metodar for deling av data via Modbus. Biblioteket er utvikla med mål om brukarvennlegheit på varierte Java plattformer og Java-einingar. [21]
Modbus er eit sikkert val for kommunikasjonsprotokoll i industrielle styresystem.[11]

JSSC

JSSC som står for Java Simple Serial Connector, er eit bibliotek for lesing og skriving til seriell port vi Java.[22]

JFrame/JPanel

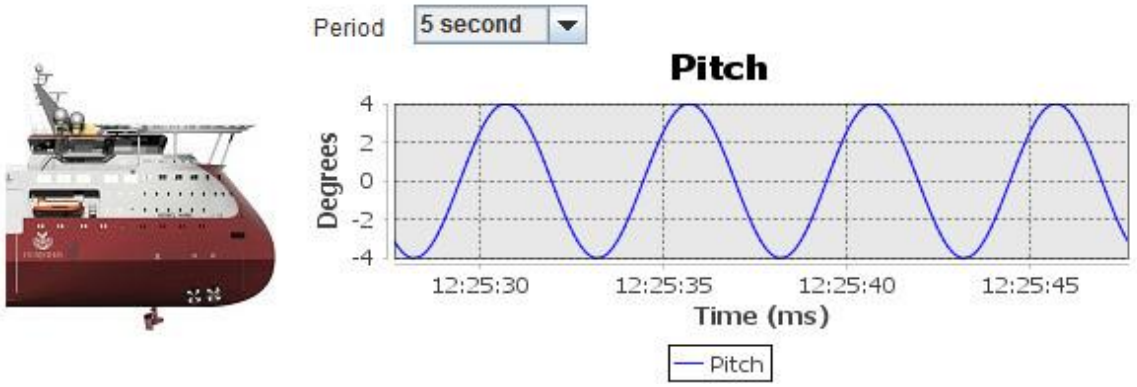
JFrame og JPanel er begge ein del av Swing pakken i Java og blir begge brukt til å opprette ein GUI. Når ein skal lage ein applikasjon med GUI er det vanleg å utvide GUI-klassa til subklassa JFrame. Med JFrame lagar ein eit vindauge til applikasjonen etter ynskt storleik. Vindauget består av ei ramme, tittel og knappane for lukke, minimere og justere storleiken på bilete. Når ein skal legge til innhald inne i ramma er det vanleg å legge til eit panel. Måten å gjere dette på er å opprette ei ny klasse som er utvida til subklassa JPanel. I dette panelet opprettar ein tabs, knappar, menyar, grafar osv. Her bestemmer ein utsjånaden og virkemåten til applikasjonen.
Figur 11 viser eit eksempel frå GUIen til Ulstein IAS Simulatoren med ramme laga i JFrame og innhald laga i JPanel. Ei ramme kan innehalde fleire panel, i figur 11 ser ein fleire faner øvst i bilde. Desse er laga i kvart sitt panel og deretter lagt inn i ramma.[23][24]



Figur 11: JFrame/JPanel eksempel, grafikk av skip utlevert av Ulstein.

JFreeChart

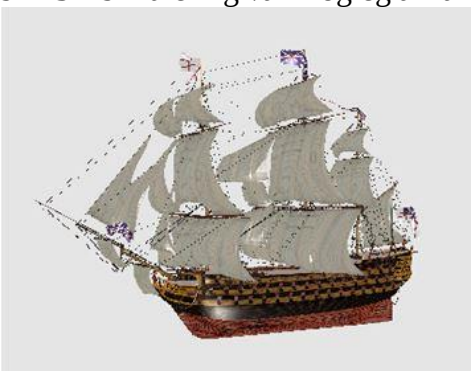
JFreeChart er eit bibliotek i Java for utvikling av grafar i GUI. Dette er fri kjeldekode som er gratis tilgjengeleg for alle på JFree si heimeside. Med dette biblioteket kan ein lage xy-grafar, pai-diagram, gnatt-diagram, histogram osv. Ein kan også animere dei ved å oppdatere grafen etter kontinuerlig innkomande verdiar. I Ulstein IAS Simulatoren blir dette gjort, her oppdaterer grafen seg etter verdiar frå ein sinusgenerator dynamisk. Som vist på biletet under ser ein nyaste sinusverdi til høgre med gamle verdiar frå dei 20 siste skunda springande mot venstre.[25]



Figur 12: Implementering dynamisk graf av JFreeChart i GUI, grafikk av skip utlevert av Ulstein.

Java 3D

Java 3D er ein API tilgjengeleg for nedlasting til Java. Med denne API kan ein framstille modellar i eit eige 3D univers ved hjelp av vektorar. I dette prosjektet blei Java 3D version 1.5.1 brukt til å framstille eit skip som bevegde seg i sjøen i forbindelse med MRU-simulatoren. Ein gratis nedlastbar 3D modell av ei seglskute blei brukt til å finne ut om ei slik 3D simulering var mogleg å framstille. Modellen er laga i 3ds format.[26][27]



Figur 13: 3D modell av skip for MRU-simulator.[27]

NMEA - Sjekksum

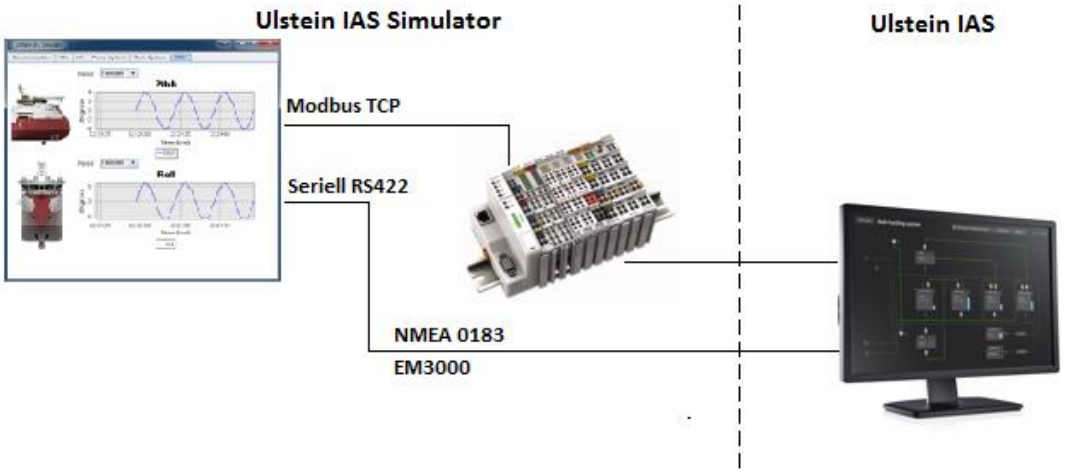
Sjekksum er ei metode for å kontrollere overført data. Ein sjekksum blir rekna ut når NMEA-stringen blir laga, og lagt til stringen før den blir sendt. Når denne dataen blir tatt imot blir ein ny sjekksum rekna ut og sjekka opp mot den sjekksummen som blei sendt. Stemmer desse overeins så indikerer det at rett data er motteke. Sjekksummen blir skrive som hexadecimal.
Måten sjekksummen blir rekna ut på er at ein først skil ut stringen mellom «$» og «*» til ein ny string.
Orginal NMEA-string:
*$PSXN,10,019,0.0000,1.3125e-02,0.0000e+0,,,,**
Utskild string for utrekning av sjekksum:
PSXN,10,019,0.0000,1.3125e-02,0.0000e+0,,,,
Når den nye stringen er utskild blir det tatt ein XOR opperasjon mellom den binære verdien til kvart teikn i stringen. Den binære verdien av teiknet blir gitt ved hjelp av åtte bit. Resultatet av denne opperasjonen blir så lagt ved i enden av stringen skrive i hexadecimal.
*$PSXN,10,019,0.0000,1.3125e-02,0.0000e+0,,,,*2e*
[28]

EM3000 – To komplement

To komplement er ei mattematisk metode for binære tal. Metoden gjer det same som ved en komplement, men legg til ein. I dataverda er denne metoden brukt til å framstille negative tal med ein binær verdi. Måten ein gjer det på er å ta utgangspunkt i den binære verdien til det positive talet. For å finne to komplementet til det positive talet bytter ein om på einarane og nullane for så og legge til ein.
Eksempel: 120, -120, -8 vist med binærkode ved hjelp av to komplement
120 = + (01111000) = 01111000
-120 = - (01111000) = 10000111 + 1 = 10001000
-8 = - (00001000) = 11110111 + 1 = 11111000
[29]

RESULTATER

Ulstein IAS Simulator



Figur 14: Illustrasjon av Ulstein IAS Simulator tilkopla Ulstein IAS, grafikk av skip og skjerm utlevert av Ulstein[30].

Ulstein IAS Simulator er eit verktøy til fabrikktesting av Ulstein IAS. Systemet består av ei datamaskin som kjører ein Java applikasjon og ein kontroller. Simulatoren består av fire delsystem: MRU-simulator, Tankpeiling-simulator, Pumpe-simulator og IO-simulator. Applikasjonen er fanebasert og består av seks faner: ei for kvar av delsystema, ei kommunikasjonsfane og ei informasjonsfane. Hardware består av ein kontrollar med tilkopla analoge og digitale IOar. Disse IOane har eigne tilkoplingsadresser som ein finn oversikt over i eiga IO-liste. Internt mellom Java-applikasjon og kontroller kommuniserast det ved bruk av Modbus TCP. MRU-simulatoren kommuniserer direkte til Ulstein IAS over seriell RS422. Det blir sendt informasjon med to typer protokoller: NMEA 0183 og EM3000.

Hardware

Oppsett

Hardware består av ein kontroller, analoge og digitale IOar og rekkeklemmer. Desse står samla på ei dinskinne. Kontrollaren er ein Wago 750-881. Krav til kontroller er at den må støtte Modbuskommunikasjon. Tal på analoge og digitale IOar i systemet er ikkje fast. Systemet er sett opp med ein base av IOar. Kor mange IOar ein har i systemet kan skalerast etter behov, det er det også tatt høgde for i IO-lista. Modbusadresser er linka opp mot IO-adresser i Codesys og lagra i kontroller.

IO-liste

WAGO ad	Modbus ad	System	Description	Var	Scale	Input/Outp
MW0	12288	General Signals	Digital Output	Boolena	0-1	DO
MW1	12289	General Signals	Analog Output	Integer	0-32767	AO
MW2	12290	Pump-simulator	Pump 1 signal remote	Boolean	0-1	DO
MW3	12291	Pump-simulator	Pump 1 signal running	Boolean	0-1	DO
MW4	12292	Pump-simulator	Pump 2 signal remote	Boolean	0-1	DO
MW5	12293	Pump-simulator	Pump 2 signal running	Boolean	0-1	DO
MW6	12294	Pump-simulator	Pump 1 speed feedback signal	Integer	0-32767	AO
MW7	12295	Pump-simulator	Pump 2 speed feedback signal	Integer	0-32767	AO

Figur 15: Utdrag frå IO-liste.

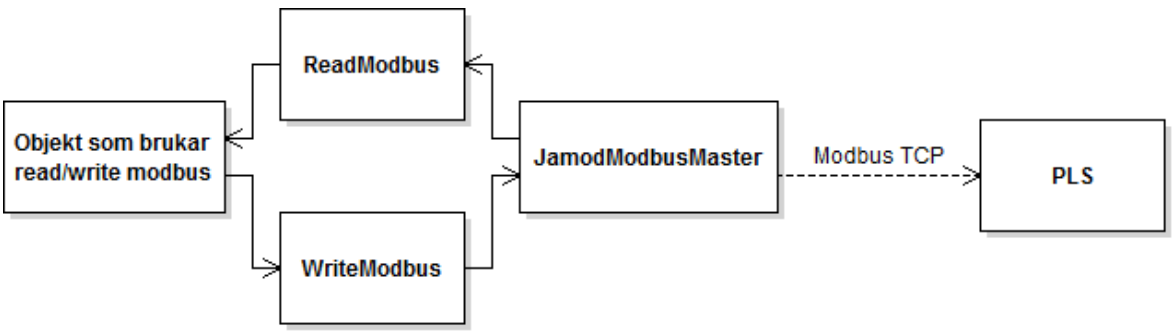
IO-lista er laga i Excel og inneheld informasjon for kvart tilkoplingspunkt på IOane. Lista består av sju kolonner. Første kolonne viser adressa til koplingspunktet i kontrolleren. Andre kolonne viser kva Modbusadresse som er sett opp mot koplingspunktet. Kolonne nummer tre viser kva system koplingspunktet høyrer til. Fjerde kolonne skildrar oppgåva til koplingspunktet. Kolonne nummer fem viser kva type variabel ein opererer med. Kolonne nummer seks viser skalering av signalet. Siste kolonne viser kva type IO ein opererer med. Lista er laga for å kunne utvidast. Dei hundre første felta er sett opp for utgangar, medan dei hundre neste er sett opp for inngangar.

Grafisk brukargrensesnitt

For brukarinteraksjon med IAS simulatoren blei det valt å skape ein GUI med enkel utføring og moglegheit for utviding. Eit krav som blei sett i funksjonsskildringa til GUI var at den ikkje skulle innehalde teknisk funksjonalitet. Dette skal ligge under eigne klasser, for å ha lause koplingar og veldefinerte klasser.
Ved bruk av Java Swing JFrame blei det programmert klassa GUIBase. Klassa arva frå JFrame og bygger i hovudsak ei rame i gitt størrelse med moglegheit for utviding via metoden addTab. Metoden addTab tek inn objekt av type JPanel og legg til dette objektet som ein ny fane i hovudrama.

Kommunikasjon

Det er sett opp ein felles kommunikasjonskanal for objekt som brukar kontroller. Modbus protokoll er valt til kommunikasjonsprotokoll. Som vist på figur 16 er det JamodModbusMaster klassa som er siste ledd i Java applikasjonen. Deretter går Modbus melding ut på nettverket og videre til Modbus slave, som er kontroller. Vidare i dette delkapittel vert det tatt ein gjennomgang, steg for steg korleis ein les og skriv til kontroller ved hjelp av Modbus og Java SwingWorker.



Figur 16: Kommunikasjonsflyt Modbus

JamodModbusMaster

Denne klassa importerer Jamod biblioteket. Vidare er det programmert eigne metoder for set av IP-adressa, oppretting og fråkopling av forbindelse, skriving og lesing av Modbusregister. Klassa er ei vidareutvikling av ei klasse brukt i hovudprosjektet i faget IE203512 Industrielle Styresystem.
WriteRegister metoda i klassa tek inn parameter for Modbusregister som skal skrivast til, samt variabel for verdi. Metoden pakkar desse inn i ein singleRegisterRequest og sender til Modbus slave. Metoden er synkronisert for å oppretthalde eit trådsikkert system.
ReadRegister metoden har eit parameter. Parameteret er adressa til Modbus registeret som skal lesast. Metode returnerer verdien til det førespurte registeret. Metoden er synkronisert for trådsikker behandling.

Read Modbus

ReadModbus klassa arva SwingWorker. Klassa har ein hovudfunksjon, å lese Modbus register til Modbus slave. Når ein opprettar eit objekt av type ReadModbus sett ein følgjande parameter:

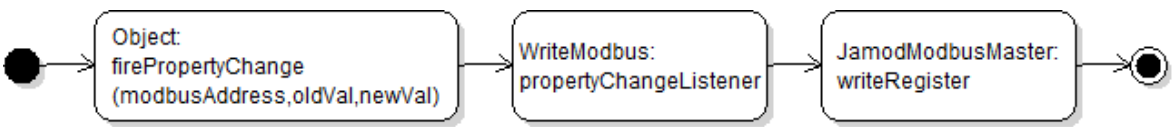
- JamodModbusMaster: objekt av type JamodModbusMaster. Kommunikasjon for modbus-TCP master/slave.
- Kva registerområde som skal lesast. Eksempel: frå register 12388 til 12398.
- Frekvens. Kor ofte Modbusregistera skal lesast.

Når eit objekt av type ReadModbus er oppretta, startar en worker thread. Denne tråden går i løkke i frekvens som er sett for objektet. For kvar loop vert registera lest ved å bruke JamodModbusMaster-objektet med metoda readRegister. For kvart register som vert lest vert det kjørt ein firePropertyChange. Her vert det sjekka om registeret har endring i ny verdi forhold til gammel verdi. Er dette uendra vert ikkje PropertyChangeEvent sendt. Når det er endringar ulikheit mellom gamal og ny verdi vert PropertyChangeEvent fanga opp av alle registrerte lyttarar.

Figur 17: Tilstandsdiagram for ReadModbus.

Write Modbus

WriteModbus tek inn JamodModbusMaster objekt. WriteModbus klassa har kun ein metode. Denne metoden opprettar ein lyttarar til SwingWorker objekt. Swingworker objekt og modbusadresse er parametera for å opprette ein lyttar. Denne lyttaren fangar opp PropertyChangeEvent på det objektet den er knytt mot. PropertyName er sett likt som Modbusadressa for å skilje dei ulike PropertyChangeEvents. Når den registrerte lyttaren fangar opp ein PropertyChangeEvent vil denne sende ein Modbus reguest til Modbus slave. JamodModbusMaster objektet og metoden writeRegister vert brukt for å sende Modbus request.



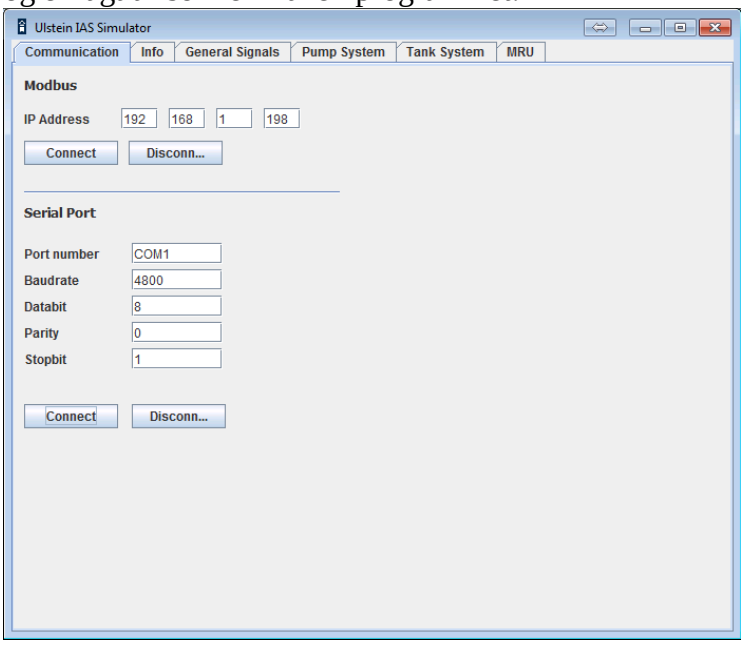
Figur 17: Tilstandsdiagram for WriteModbus.

Seriell kommunikasjon

Klassa SerialCommunication importerer biblioteket JSSC. Klassa har metodar for oppretting av seriell forbindelse, lukke seriell forbindelse, sjekk om seriell forbindelse er oppretta, lese seriell port og skrive til seriell port.

GUI kommunikasjon

For oppretting av forbindelse for Modbus og seriell kommunikasjon er det utvikla ein GUI med namnet GUICommunication. GUICommunication er ein del av Ulstein IAS Simulator og er lagt til som ein fane i programmet.



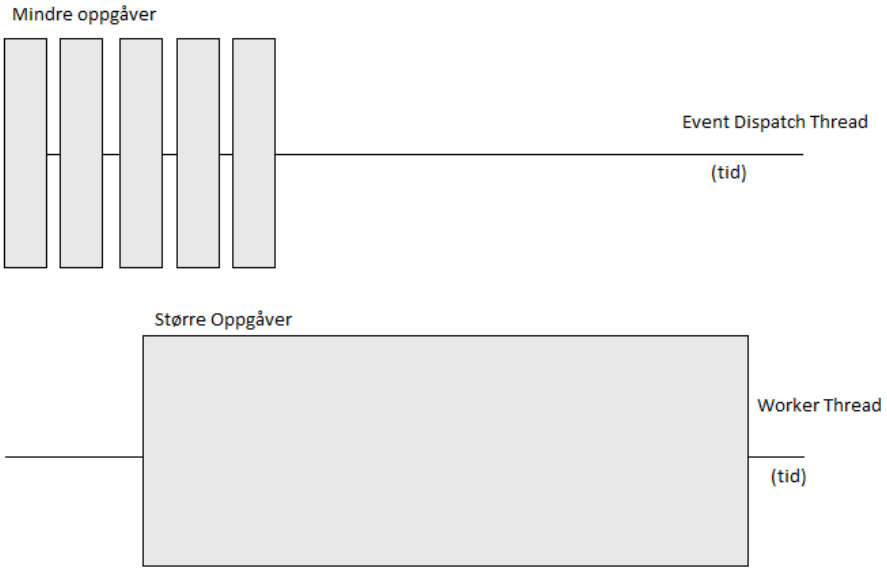
Figur 18: Communication fana i Ulstein IAS Simulator

Modbuskommunikasjon via TCP/IP brukar 502 som fast portnummer. For oppretting av modbusforbindelse via TCP/IP er det kun naudsynt å sette IP adresse til slave å deretter trykke connect knapp.

For oppretting av forbindelse til seriellkommunikasjon må følgjande parameter settast:

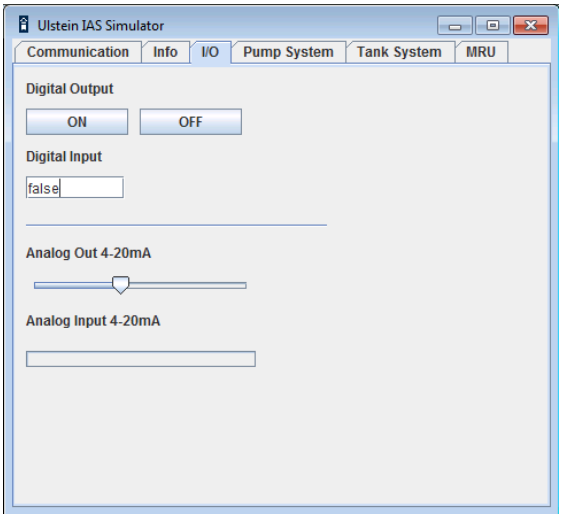
1. Portnummer
2. Baudrate
3. Databit
4. Parity
5. Stopbit

Tilkopling av seriell- og modbus-forbindelse er ein større prosess som kan føre til at applikasjonen heng. For å unngå dette er det brukt innerklasser i GUICommunication som arvar SwingWorker. Desse kjører eigne doInBackground metoder som nyttar Worker Thread for tilkoplingane. Metodane har feilhandtering og vil vise eventuelle feilmeldingar i eit nytt vindauge.



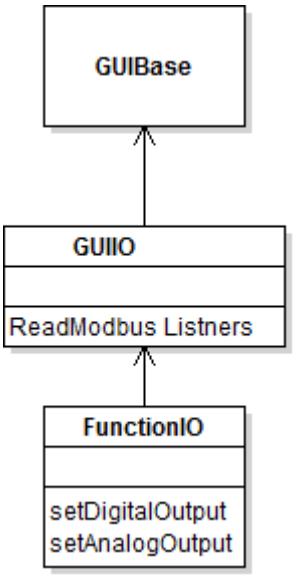
Figur 19: Illustrasjon på trådar som brukar lite resursar samanlikna med trådar som brukar mykje resursar.

I/O-simulator



Figur 20: I/O-simulator.

Funksjonen til I/O simulatoren er å kunne kople på analoge eller digitale I/O. Simulatoren har fire I/O: ein digital utgang, ein digital inngang, ein analog utgang og ein analog inngang.
Den digitale og analoge inngangen har registrerte lyttarar for propertyChangeEvent tilknytta ReadModbus klassa. Felta i GUI vert oppdatert når I/O på kontroller vert endra. Oppdatering av I/O status på ReadModbus er sett til 10Hz. Lyttarane er oppretta i GUIGeneralSignal. For den digitale inngangen står der true i tekstfeltet om inngangen er høg, eller false om inngangen er låg. Den analoge inngangen viser ved ein progressbar den analoge verdien.
I FunctionGeneralSignal klassa er det metodar for setting av digitale og analoge utgongar. I metodane vert det kjørt ein firePropertyChange. Denne vert fanga opp av WriteModbus klassa og sendt vidare til JamodModbusMaster klassa og ut til kontroller. Den digitale utgongen vert sett ved å bruke knappane ON for å sette utgong høg og OFF for å sette utgongen låg. Den analoge utgongen sett ein ved å bevege på skyveknappen.



Figur 21: Klassediagram for General Signal

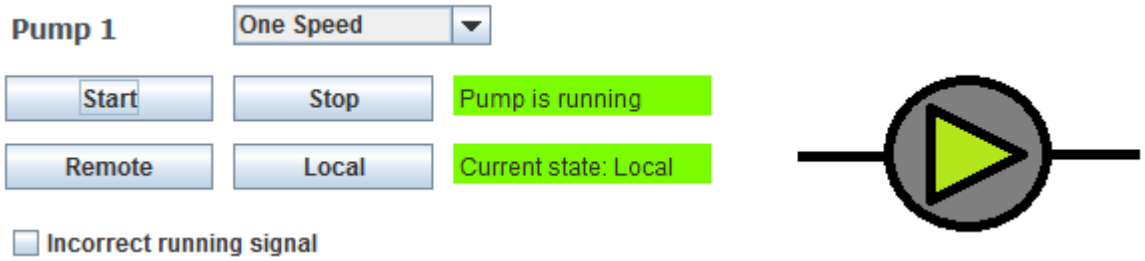
Pumpesimulator

Pumpesimulatoren simulerer fjernstyrte pumper. Hovudtanken med pumpesimulatoren er at ein skal kunne starte og stoppe pumper, gi signal om ei pumpe går eller ikkje, gi feil signal om ei pumpe går og velje om ein skal kjøre pumpa lokalt eller fjernstyrt. Kjører ein pumpa med fjernstyringa på, kan ein starte og stoppe pumpa i pumpesimulatoren via IASen. Feil signal vil sei at pumpesimulatoren sender melding til IAS at pumpa har stoppa sjølv om den i verkelegheita går. Det er valt å fokusere på tre forskjellige type pumper:

- Pumper med ei hastigheit
- Pumper med to hastigheiter
- Frekvensstyrte pumper

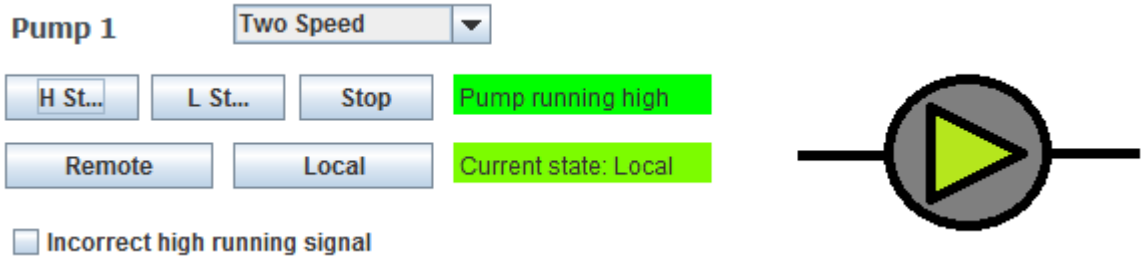
I pumpesimulatoren vel ein mellom desse i rullegardinmenyen tilhøyrande pumpa som skal styrast. Første versjon av pumpesimulatoren hadde kun moglegheit for å simulere pumper med ei hastigheit. I eit møte med ein av prosjektingeniørane for IAS kom det fram at pumpesimulatoren burde også ta føre seg to hastigheitspumper og frekvensstyrte pumper(Vedlegg 4, møtereferat 16.02). Dette resulterte i mindre endringar og blei difor implementert.

Pumpe med ei hastigheit kjem opp automatisk kvar gong pumpesimulatoren blir starta. Denne er identisk med første utkastet av pumpesimulatoren. Her har ein enkel simulering av start og stopp ilag med dei andre funksjonane til pumpa.



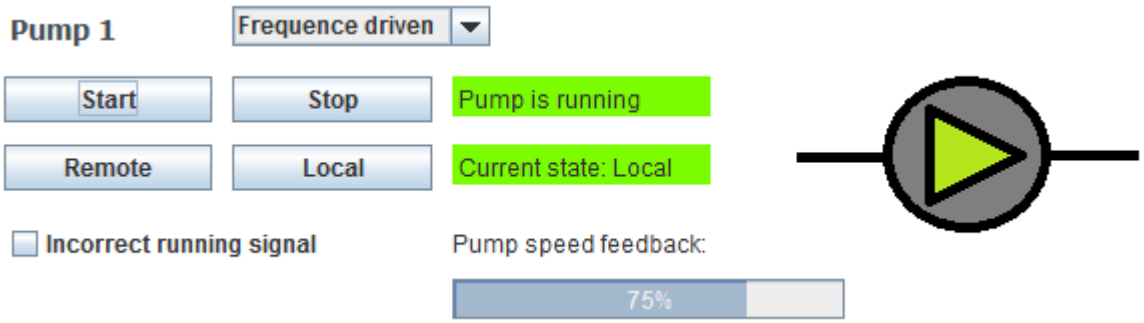
Figur 22: Illustrasjon av pumpesimulator for pumpe med ei hastigheit.

Forskjellen mellom pumpe med ei hastigheit og to hastigheiter er måten å starte den på. Når ein vel å simulere pumpe med to hastigheiter får ein to val ved start av pumpa, ein knapp for høg start og ein for låg start.



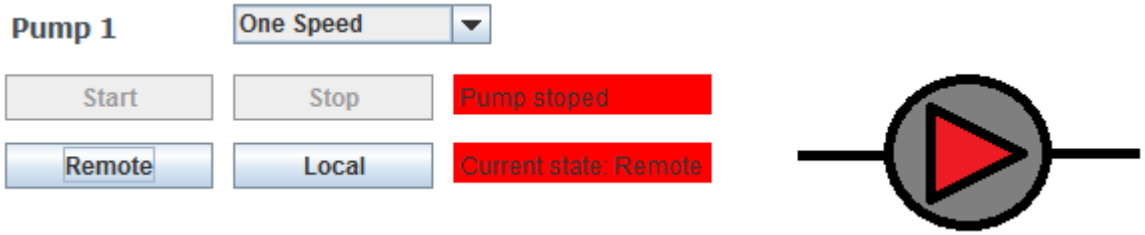
Figur 23: Illustrasjon av pumpesimulator for pumpe med to hastigheiter, her starta i høg hastigheit.

Vel ein å simulere frekvensstyrt pumpe får ein opp same grafikken som for pumpe med ei hastigheit. Denne pumpa blir starta og stoppa på same måte, men har i tillegg ein indikator som viser hastigheita til pumpa frå 0 til 100 prosent.



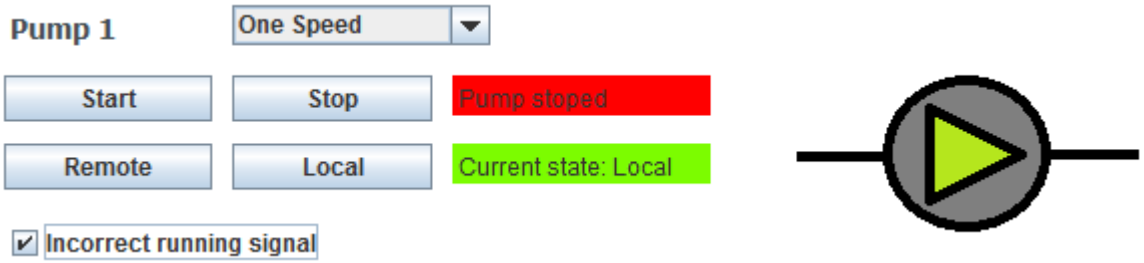
Figur 24: Illustrasjon av pumpesimulator for frekvensstyrt pumpe, her med 75% pådrag.

Skal ein styre pumpene via IASen så sett ein pumpa i fjernstyrttilstand. Då vil start- og stoppknappane bli utilgjengelege i GUIen og ein indikasjon på at pumpa står i fjernstyrt tilstand kjem fram.



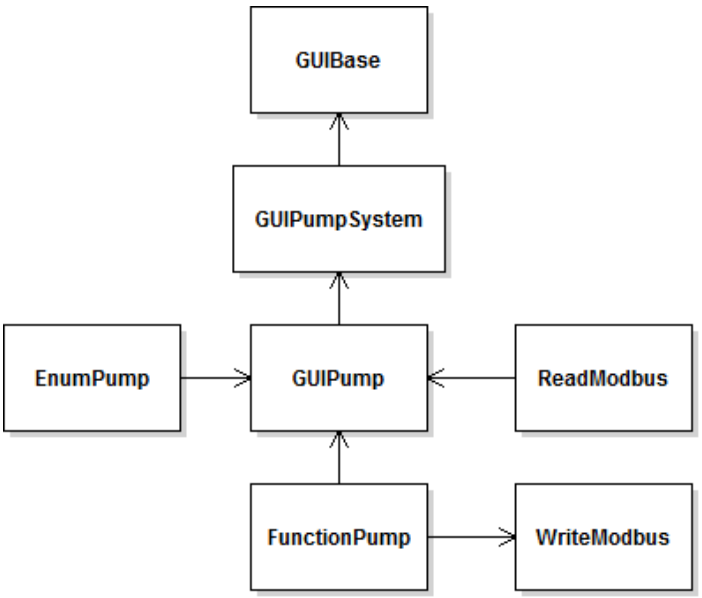
Figur 25: Illustrasjon av pumpesimulator med pumpe i fjernstyrt tilstand.

Knappen for feil signal er kun synleg og tilgjengeleg når pumpa har starta. Trykker ein på knappen sender pumpesimulatoren signal til IAS at pumpa ikkje går. Bilete av pumpa i pumpesimulatoren indikerer at pumpa framleis går, og IAS har helder ikkje mottat stop signal.



Figur 26: Illustrasjon av pumpesimulator med korrupt signal til IAS.

Pumpa blir testa på dagens FAT utan pumpesimulator. Måten dette blir gjort på er å kople opp eit brytarpanel med tre knappar for å simulere start-, stopsignal og signal som indikerer om pumpa går. Saman med rettleiar frå Ulstein blei det bestemt at denne burde bli implementert i ein IAS simulatoren for å samle all simulering på ei plattform.



Figur 27: Klassediagram Pumpe-simulator.

I Java er Pumpe-simulatoren delt opp i fire klasser: FunctionPump, GUIPump, EnumPump og GUIPumpSystem. FunctionPump utvider Swingworkerog tek seg av den funksjonelle delen av pumpesimulatoren. Her finn ein metode for å lese og sette variablar. Metode som skal gi signal vidare til IAS sender ut ein firePorperyChange når ny verdi blir sett. Ser ein på eit tilstandsdiagram for opperasjonen som skjer når ei pumpe blir starta i pumpesimulator ser det slik ut.

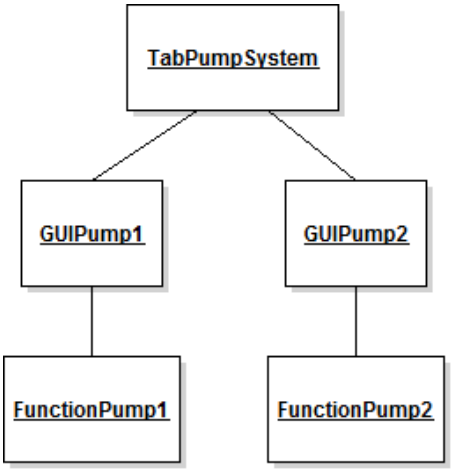
Figur 28: Tilstandsdiagram for start av pumpe i pumpesimulator.

Her blir ei pumpe starta ved å trykke på startknappen. Pumpesimulator indikerer at pumpa går og metoden setRunning blir kalla med inndataverdi 1. I denne metoden ligg det ein firePropertyChange som sjekkar ny verdi opp mot gammal verdi. Gammal verdi er 0 og ny som kjem inn er 1. Når ny og gammal verdi er ulik blir ei propertyChange-hending sendt ut med tilhøyrande IO-adresse som kodenamn. Dette blir fanga opp av WriteModbuss-klassa som ligg og lyttar etter propertyChange-hendingar. WriteModbus ser på kodenamnet til hendinga og sender den vidare over Modbus til IO med tilhøyrande adresse. IOen blir sett høg og IAS får melding om at pumpa går.
GUIPump handterer alt det grafiske i pumpesimulatoren og er ei utviding av klassa JPanel. Utsjånad på knappar, bilete, rullegardin og layout sett ein opp her ilag med kva som skal skje når dei ulike elemnta i GUIen blir aktivert. EnumPump-klassa inneheld namna til dei forskjellige pumpene. Disse blir brukt i rullegardinmenyen i GUIen. For å få GUIen til å respondere når noko blir kalla i IASen har GUIPump-klassa eigne lyttarar som fylg med på propertyChange-hendingar.

Tilstandsrapport på ein situasjon der pumpe skal startast frå IAS så ser slik ut.

Figur 29: Tilstandsdiagram for start av pumpe frå IAS.

Her blir pumpa starta i IAS, dette vil sette tilhøyrande IO i pumpesimulator høg. Denne opperasjonen kan kun skje om pumpa er sett i fjernstyrt tilstand. ReadModbus-klassa registrer dette og ved hjelp av firePropertyChange blir det sendt ut ei propertyChange-hending med tilhøyrande kodenamn. I GUIPump ligg metode som lyttar etter hendingar med dette kodenamnet. Når hendinga blir fanga opp viser GUIen at pumpa har starta. IAS får då tilbakemelding frå pumpesimulator at pumpa går på same måte som forklart i tilstandsdiagrammet for start av pumpe.
GUIPumpSystem set opp layouten til pumpene i pumpesimulator. Korleis ei pumpe skal sjå ut blir gjort i GUIPump-klassa, men for å vise denne i pumpesimulatoren må den leggast til i GUIPumpSystem. Dette blir gjort for å kunne legge til fleire pumper i GUIen på ein enkel måte utan kodeduplisering. Når ein opprettar GUIPumpSystem bestemmer ein kor mange pumper som skal vere mogleg å legge til og korleis dei skal plasserast i GUIen. I denne versjonen av pumpesimulator er det laga til for testing av to pumper. Under ser ein oppretting av to pumper illustrert i eit objektdiagram.

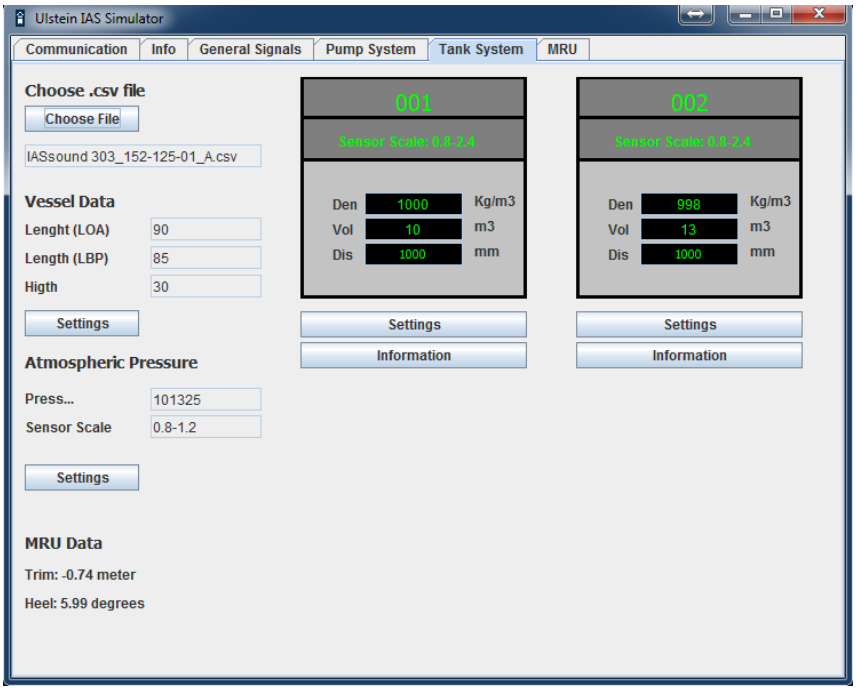


Figur 30: Objektdiagram for Pumpe-simulator med to pumper.

Tankpeiling-simulator

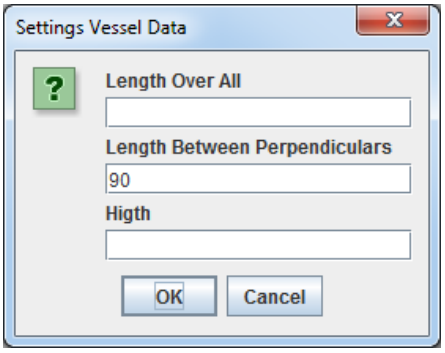
Funksjonaliteten til tankpeiling-simulatoren er simulering av væskenivå i tank. Simulatoren kommuniserer med kontroller ved bruk av SwingWorker, propertyChangeEvent og WriteModbus klassa. Som vist i teorikapittel er måling av tanknivå utført ved bruk av analoge trykksensorar. Tankpeiling-simulatoren simulerer trykksensor for atmosfærisk trykk og trykksensorar for tankpeiling.
Tankpeiling-simulatoren tek inn same MRU verdiane som Ulstein IAS tek imot serielt frå MRU. Tankpeiling-simulatoren brukar desse til kalkulering av tanksensor verdi. MRU verdiane pitch og roll vert fanga opp som propertyChangeEvents sendt frå sinusgenerator. Total volum på væska i tanken vert avgjort ved oppslag i tanktabell. Forholda mellom væskenivå i tank og bevegelse på fartyet er faktorane som påverkar tanksensorane.

GUI Tankpeiling-simulator



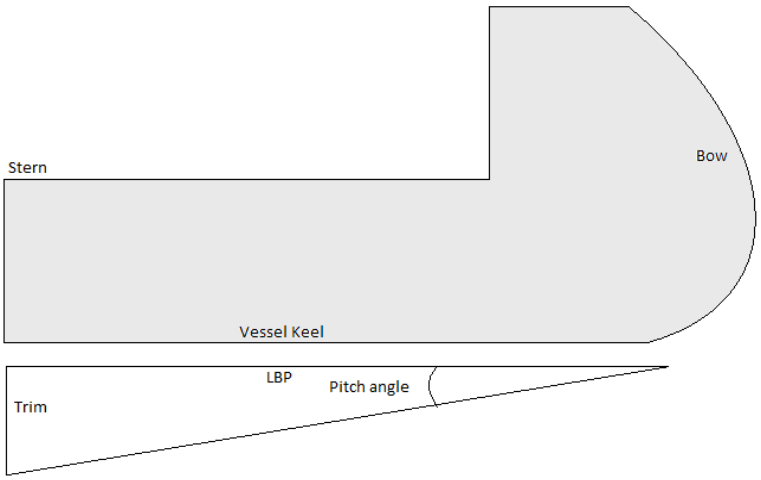
Figur 31: GUI tank system

I GUI til tankpeiling-simulatoren vel ein tanktabell CSV-fil(Character Seperated Value) ved å trykke på Choose File knappen. Då dukkar det opp ein filvelgar der ein navigerer seg i filsystemet til datamaskinen.



Figur 32: GUI vessel data settings

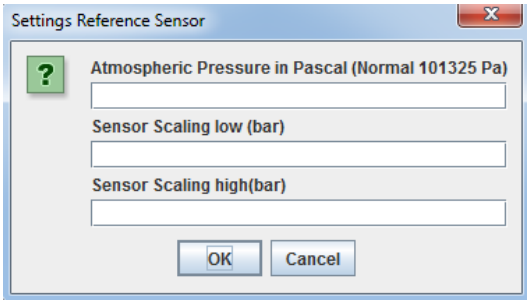
Feltet Vessel Data sett parametera for høgd og lengder på fartyet. Ved å trykke på settings knappen kjem det opp eit nytt vindu for å sette parameter. LOA (Length Over All) og LBP(Length Between Perpendiculars) er måleeiningar for lengd på fartyet. LBP vert brukt til kalkulering av trim for fartyet. Sett fartyet i profil frå sida er trim høgdeforskjellen mellom baug og akterskip.



Figur 33: Illustrasjon av fartøyets trim.

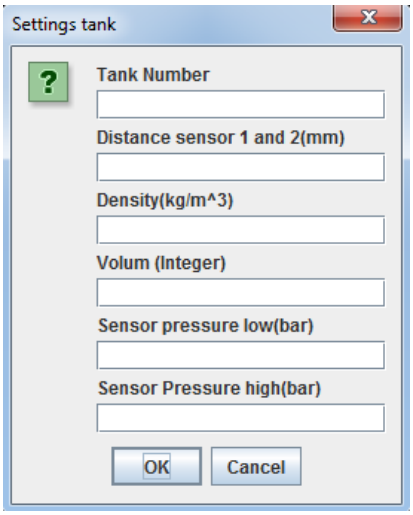
For kalkulering av trim er det brukt trigonometrisk funksjon for rett trekant:[6]

$$Trim = \sin(pitch\ angle) * LBP$$



Figur 34: GUI reference sensor settings

I Atmospheric Pressure legg ein inn parameter for ønska atmosfærisk trykk og sensor måleområde. Ein sensor kan eksempelvis vere oppgitt til 4-20mA i måleområdet 0,8-1,2 bar. Det gir 4mA signal ved 0,8 bar og 20mA ved 1,2 bar. Måleområdet for sensor er sett på som lineært. Dermed vert det brukt lineær interpolering for skalering av signal.



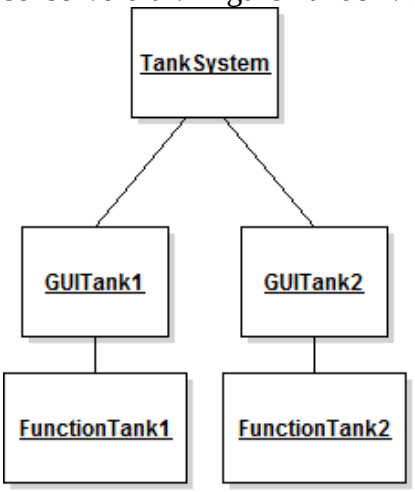
Figur 35: GUI Tank settings

Parameter for tanken er sett ved å trykke på knappen settings under den grafiske framstillinga av tanken. Nytt vindauge dukkar opp. Parametera som skal settast er:

1. Val av tank som skal simulerast. Tankane er nummerert som for eksempel "001", "041", "151".
2. Distanse mellom sensor 1 og sensor 2 i tanken. Dette vert brukt til simulering av trykkforskjell mellom dei to sensorane, for å teste automatisk tettleikmåling til IAS.
3. Massetettheita til væska på tanken.
4. Volum. Væskeinnhald i tanken.
5. Start måleområde til sensor.
6. Slutt måleområdet til sensor.

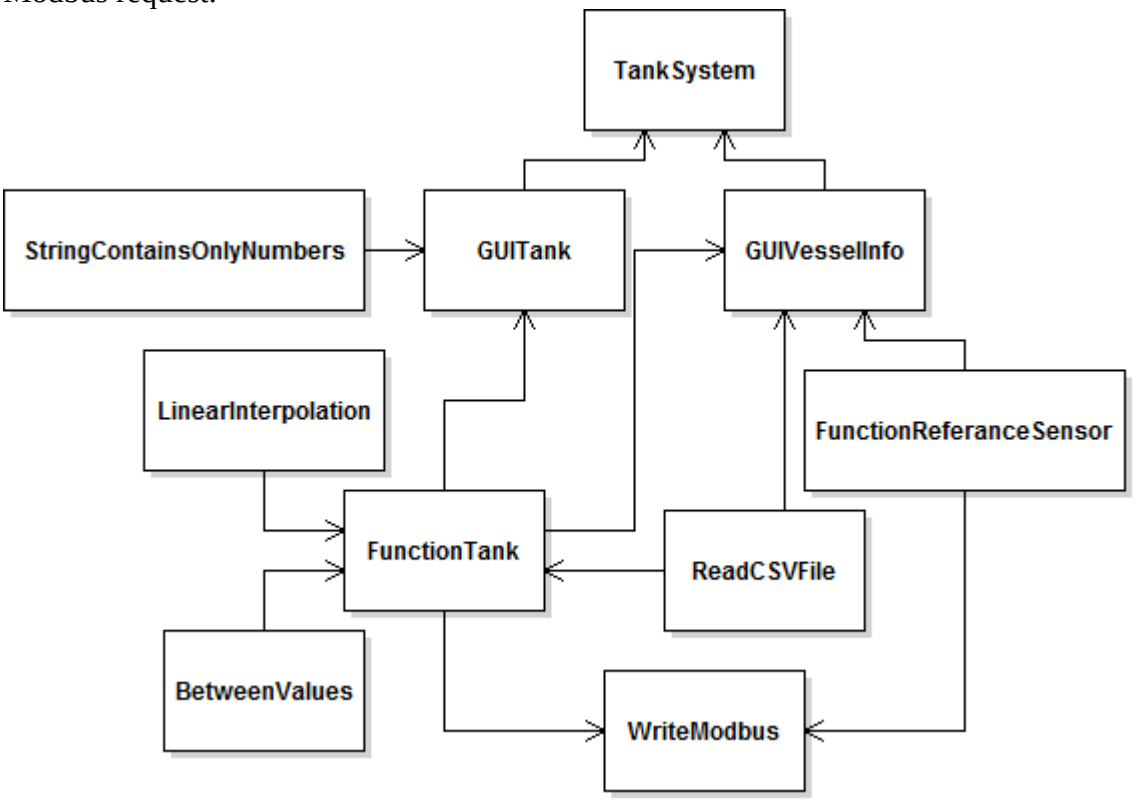
Virkemåte til tankpeiling-simulator

Tankpeiling-simulatoren er objektorientert programmert der klassa GUITank skapar det grafiske brukargrensesnittet til tanken, klassa FunctionTank tek seg av kalkulering av sensorverdiar. Figuren under viser eit objektdiagram med to tankar i tanksystem.



Figur 36: Objektdiagram tanksystem

Tråden i Klasse FunctionTank går i løkke med 1hz intervall. I denne tråden blir det utført henting av sanntids verdiar, kalkuleringar og skalering av signal. Til slutt blir sensorverdi sendt som ein propertyChangEvent til WriteModbus klassa og vidare til kontroller via ein Modbus request.



Figur 37: Klassediagram tankpeiling-simulator

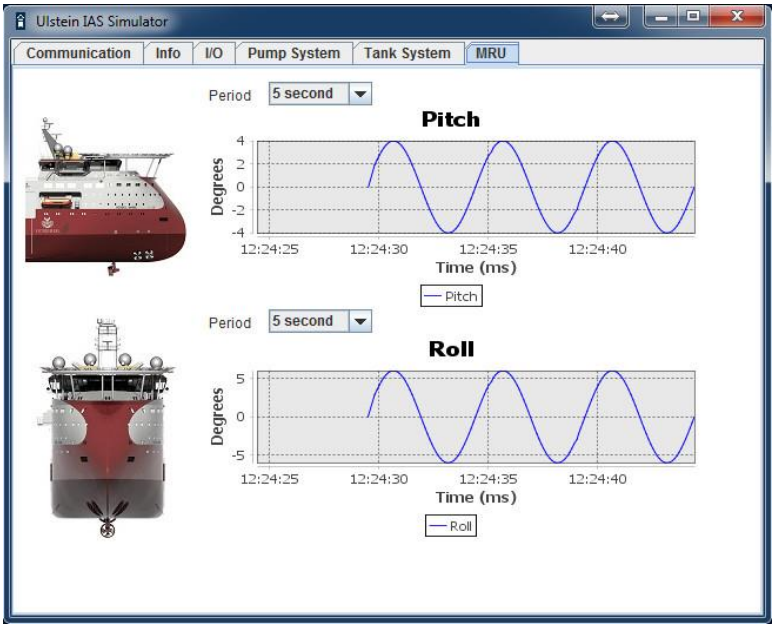
Parameter for oppsett av tankpeiling-simulator:

1. Legg til tanktabell fil ved å bruke filvelger.
2. Legg til parameter for *Vessel Data, Atmospheric Pressure* og tank settings.
3. Klassa StringContainsOnlyNumbers som sjekkar karakterane som vert innsett i tekstfelta. Det kjem opp feilmelding og rettleiing dersom parametera som vert sett ikkje er godkjent.

Stegvis gjennomgang av trådane til klassa FunctionTank:

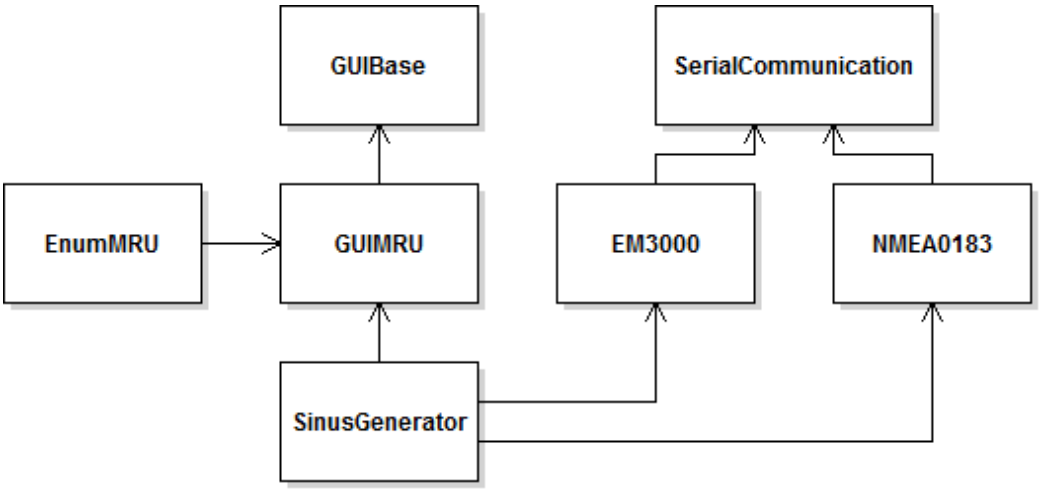
1. Verdiar for pitch og roll vert fanga opp frå SinusGenerator ved bruk av propertyChangeEvent lyttar, og lagra som variablar i FunctionTank klassa.
2. Trim kalkulerast ut frå pitch-vinkel og LBP ved bruk av trigonometrisk funksjoner for rettvinkla trekant.
3. For kvar gjennomgang av løkka vert det lagra sanntidsverdiar for trim og roll. Variablane er lagra lokalt i løkka med variabelnamna trimNow og rollNow.
4. Klassa BetWeenValues sjekkar verdien for trimNow og rollNow. For trimNow sjekkar den om verdien er eksakt som i tanktabell(-2. -1.5, -1, -0.5, 0, 0.5, 1, 1.5, 2) eller ikkje. Er verdien eksempelvis 1.7 vil metoden getBetweenValues returnere {1.5,2.0}. Dette vil sei at trimNow er mellom 1.5 og 2.0 meter. For rollNow vert det sjekka om verdien er (-6, -4, -2, 0, 2, 4, 6). Er verdien eksempelvis 1.3 for rollNow vil metoda getBetweenValues returnere {0 , 2}, som vil sei at rollNow er melom 0 og 2 grader.
5. Før kalkulering startar, sjekkar ein om alle parameter er sett samt tanktabell er lasta inn. Dette for å unngå feilmelding som nullpoint exception.
6. ReadCSVFile klassa har ein metode getHight. Denne les tanktabell-fila som er valt. Metoda getHight returnerer høgda på vassøyla over sensor. For å kunne kalkulere væskeinnhaldet i tanken treng ein høgda på vassøyla for alle trim{t1,t2} og roll{r1,r2} verdiar. Er trimNow verdi mellom 1 og 1,5 meter, vert det gjort oppslag i tabell for begge verdiar. Er verdi for rollNow mellom 0 og 2 grader vert det gjort oppslag for begge verdiar der også. Dette vil gi totalt fire høgder. For å finne ein tilnærming av høgda til væskesøyla vert det bruk lineær interpolering i tre steg på alle 4 høgder.
7. Kalkulering av trykk og skalering av sensorsignalet vert gjort for både sensor 1 og sensor 2. Ved skalering av sensorsignal vert det nytta lineær interpolering.
8. Nye sensorverdiar vert sendt som ein propertyChangeEvent og fanga opp av WriteModbus klassa.

MRU-simulator



Figur 38: MRU-simulator, grafikk av skip utlevert av Ulstein.

Med MRU-simulator kan ein simulere skipets rørsle i bølger. MRU-simulatoren sender informasjon direkte inn på Ulstein IAS via seriellkommunikasjon og har moglegheit for å sende via to forskjellige protokollar, NMEA og EM3000. MRU-simulatoren kommuniserer med Ulstein IAS gjennom ein USB til seriell-konverter. Tankpeiling-simulatoren brukar også signala frå MRU-simulator til sine simuleringar. Ein kan simulere to rørsler, pitch og roll. Desse rørslene blir generert av ein sinusgenerator.
I grafikken til MRU-simulator får ein opp to felt. Det eine feltet viser båten frå sida og ein graf som viser sinussignal som simulerer pitch. Felt nummer to viser fronten av båten og ein graf som simulerer roll. I kvart av disse felta har ein rullegardinmenyen som bestemmer frekvensen til bølgene. Her har ein fire val, frå 5-20 sekund med inkrement på fem sekund. Grafane som viser sinusbølgene er dynamiske og viser sinusverdiane frå dei 20 siste sekunda.
3D simulering av eit skip i MRU-simulator var også eit alternativ. Det er sett på moglegheiter for dette ved hjelp av Java 3D API. Eit krav for å ta i bruk dette var at UDS (Ulstein Design & Solution) kunne stille med ein 3D modell av eit Ulstein skip. Grunna rettigheiter til deira 3D modellar let ikkje dette seg gjere, ideen blei dermed forkasta.

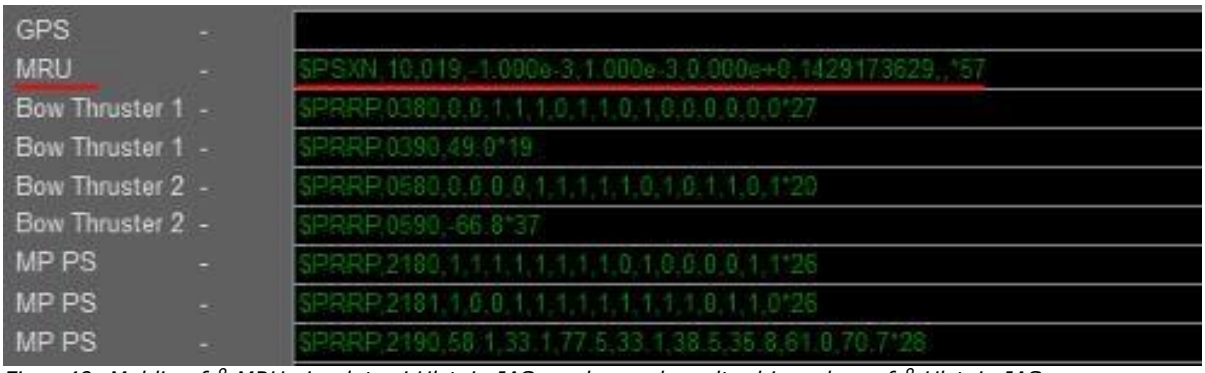


Figur 39: Klassediagram for MRU-simulator.

MRU-simulatoren er bygd opp av fem klasser:
SinusGenerator, GUIMRU, NMEA0183, EM3000 og EnumMRU.
SinusGenerator utvidar Swingworker og har som hovudoppgåve å generere sinusbølger. Her sett ein og les verdiar tilhøyrande sinusbølga som blir generert. Amplituden til sinusbølga er fast og blir sett i konstruktøren av klassa saman med namnet på generatoren. Perioden til sinusbølga kan ein endre i GUIen med å kalle ei sett-metode i SinusGenerator. For å passe inn i NMEA 0183 protokollen blir den genererte sinusverdien runda av med 4 desimalar etter komma. Verdien blir også gjort om til string med vitskapleg notasjon (f.eks. 1.234e3 eller -3.456e-2). SinusGenerator gir ut sinusverdi i grader, men sidan NMEA 0183 skal ha vinklar i radianar er det laga ei metode som reknar om frå grader til radianar. Ved nye sinusverdiar blir eit property change event sendt.
GUIMRU utvidar JPanel klassa. Klassa tek seg av det grafiske i MRU-simulator. I denne klassa ligg det lyttarar som lyttar på property change event frå SinusGenerator. Ein for rollverdi og ein for pitchverdi. Disse verdiane blir framstilt i dynamiske grafar som er laga ved hjelp av JFreeChart biblioteket. Grafen er sett opp med amplituden til sinuskurva i y-retning, og tidsakse i x-retning. Tidsaksen viser klokkeslettet verdien blei vist på og blir skriven kvart femte sekund. Ved å velje frekvensen til sinuskurva i rullegardinmenyen blir sett-metoden for perioden kalla med ny verdi. EnumMRU-klassa held på namna til valmulegheitene i rullegardinmenyen.
NMEA0183 er ei subklasse av Tråd og har i oppgåva å sende informasjon etter NMEA 0183 protokollen. Klassa tek inn verdiar for roll og pitch frå to forskjellige sinusgeneratorar og sett dei inn i ein samla string. Stringen blir kjørt gjennom ein sjekksum-generator. Den utrekna sjekksummen blir lagt til den endelege stringen før den blir sendt via seriell linje. Ny oppdatert string blir sendt ein gang i sekundet.
EM3000 er også ei subklasse av Tråd. Denne skal sende informasjon etter EM3000 protokollen. På same måte som NMEA0183 tek den inn roll og pitch frå to forskjellige sinusgeneratorar. EM3000 protokollen opererer på bitnivå. Alle verdiar som blir lagt til blir skrivne som bits i ei bytearray. Kvar bytearray blir lagt til i ein endeleg arraybuffer som representerer EM3000. Verdiar som består av to bytes blir lagt til i bytearrayen med den minst signifikante byten først. Her er ingen form for sjekksum, arraybufferen med bytearrays blir sendt ut på seriell linje direkte.

Testing

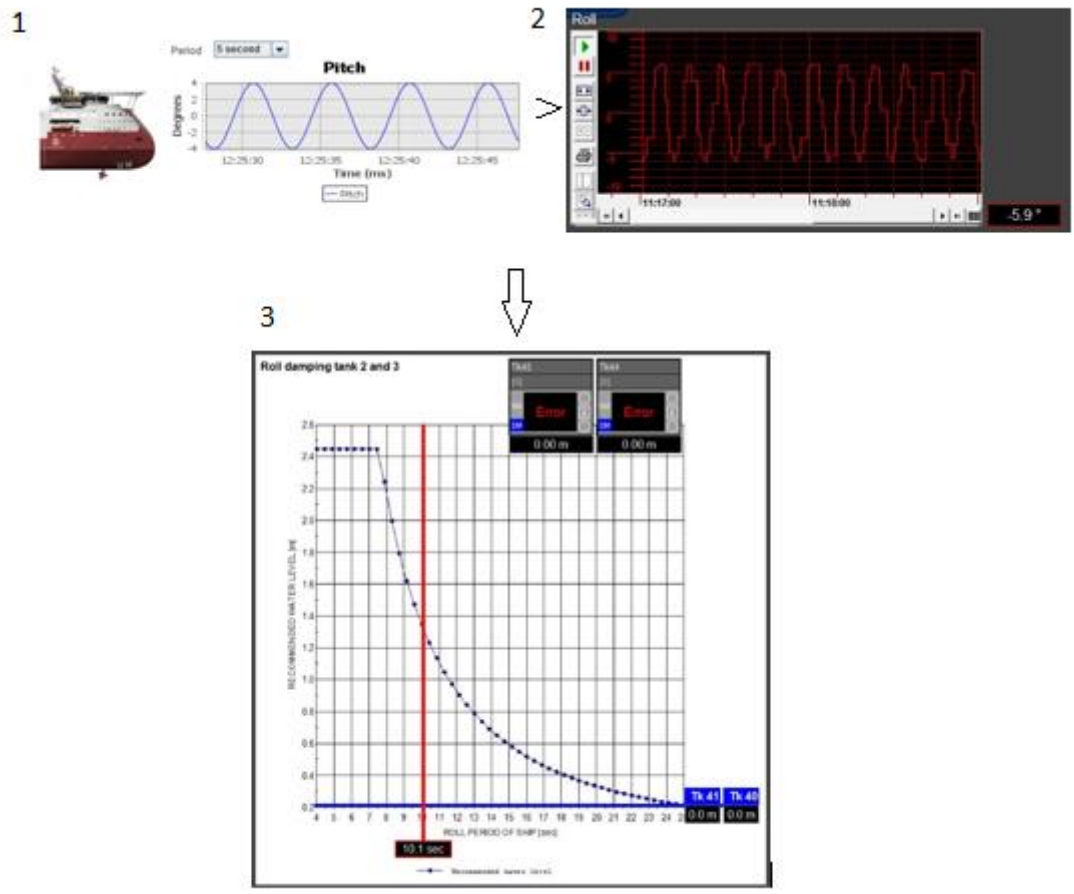
Det er utført både interntestar og testar opp mot Ulstein IAS. For kvart delsystem er det laga testrapportar som viser at tiltenkt funksjonalitet er oppnådd. Sjå testrapportar i vedlegg. Testing opp mot Ulstein IAS er utført på Ulstein sin labb i Ulsteinvik. Grunna mangel på tanktabell og feil på lesing av signal inn på kontroller på testdag, er kun MRU-simulator testa opp mot Ulstein IAS. Sjå testrapport for MRU-simulator i vedlegg for testprosedyre. Testing av MRU-simulator blei utført under pre-FAT på eit system som skal leverast til skip. Ulstein IAS les signal frå MRU-simulator ein gang i sekundet.



Figur 40: Melding frå MRU-simulator i Ulstein IAS merka med raudt, skjermdump frå Ulstein IAS.

Figur 41: Omrekning til grader i Ulstein IAS merka med raudt. Pitch øvst, roll nr 2, skjermdump frå Ulstein IAS.

Fleire system er avhengig av signal frå MRU, mellom anna stabilisatortanksystemet. Dette systemet prøvar å skape ei sinusbølge ut frå rollverdien inn frå MRU. Rollverdien skal her vere i grader. Under testing vart det avdekkt feil i Ulstein IAS ved omrekning frå radianar til grader.



Figur 42: Stabilisatortanksystem med signal frå MRU-simulator, grafikk av skip utlevert av Ulstein saman med skjermdump frå Ulstein IAS.

Over ser ein korleis stabilisatortanksystemet opererer med signal inn frå MRU-simulator.

1. MRU-simulatoren sender rollverdiar på ±6 grader med ein frekvens på 10 sekund.
2. Ulstein IAS les rollverdi en gang i sekundet. Stabilisatortanksystemet estimerar ei sinuskurve ut frå leste verdiar.
3. Ulstein IAS finn frekvensen til den estimerte sinuskurva og finn korleis vatnet i stabilisatortankane skal fordelast.

DRØFTING

Ulstein IAS Simulator fungerer som planlagt med tiltenkte funksjonalitetar. Gjennom utvikling av systemet er det fokusert på at systemet skal kunne utvidast. Det har resultert i ein fanebasert simulator der kvar fane representerer eit delsystem. Dette gjer simulatoren oversiktleg. Delsystema er programmert kvar for seg og deretter lagt til som ei fane i simulatoren. På denne måten kan ein utvikle nye delsystem og legge dei til som nye faner i ettertid. Delsystema er også laga for å kunne utvidast, men her må ein inn i koden for å legge til nye objekt. Dette kunne vore gjort annleis med ei anna vinkling på prosjektet. Hadde ein utvikla systemet for mengdetesting kunne ein sett opp systemet med fleire valmoglegheiter. Ein startmeny med val for kva og kor mange einingar som skal testast kunne vore eit alternativ. Med å forenkle GUIane til dei forskjellige delsystema kan ein gjere kvar eining mindre og dermed få plass til fleire for testing. Skal ein lage til for massetesting må ein også sjå på hardware. Oppkopling må skje raskt. Ein må ta utgangspunkt i at simulatoren skal brukast uansett kva leverandør av kontroller som blir brukt. Tilkopling for testing av IO må derfor bestå av ei universell kontakt som ein pluggar direkte i IO på Ulstein IAS.

Design av GUI er utvikla med tanke på at simulatoren skal vere brukarvennleg å enkel å forstå. Ein ser til ei kvar tid kva som kva som skjer i simulatoren. På ein FAT kjem personar frå godkjenningsorgan som ikkje jobbar med Ulstein IAS til dagleg. Med ein forståeleg GUI vil denne personen sjå kva som skjer på utsida av Ulstein IAS, og lettare forstå funksjonalitet av det som blir testa. GUIen blir på denne måten ein stor del av prosjektet og tek dermed ein del tid å utvikle. Med tanke på mengdetesting er GUI mindre viktig, her står funksjonalitet i fokus.

Applikasjonen er utvikla med OOP. Skal ein endre på noko i programmet treng ein kun gjere det ein plass. Dette er også positivt om fleire delsystem skal leggast til i simulatoren. Deler ein opp klassene innanfor fornuftige rammer med tanke på mengde i kvar klasse, er dette ei oversiktleg framgongsmetode.

Java applikasjon og kontroller kommuniserer via Modbus ved hjelp av Jamod-biblioteket. Jamod-biblioteket er ei fri kjeldekode som ikkje er oppdatert sidan 2010. Seinare i utviklinga har ein kome over eit bibliotek som heiter J2mod, som er eit alternativ til Jamod-biblioteket. J2mod er seinast oppdatert i 2014 og derfor ei betre kjelde. Sidan funksjonaliteten til kommunikasjon i dette prosjektet fungerer som det skal med Jamod blei det bestemt å ikkje gå over til J2mod. Utvikling av sjølve simulatoren blir sett på som viktigare[31].

MRU-simulatoren kan kommunisere med to forskjellige protokollar. Den er i utgangspunktet sett opp for å kommunisere med NMEA 0183 standaren. For å kommunisere med EM3000 standaren må ein inn i koden for applikasjonen og endre frå NMEA 0183 til EM3000. Ei betre løysing er å ha ei valmoglegheit i kommunikasjonsfana der ein vel kva protokoll ein skal bruke til kommunikasjon.

Tildeling av adresser for kvar IO skjer i Java i forhold til IO-liste. Kvar IO brukar SwingWorker saman med klasser som les og skriv Modbus for å kommunisere med kontroller. Eit alternativ som har vore sett på er å ha ein server som tek seg av all kommunikasjon via Modbus til kontroller. Ved å laga eit eige datagram for å kommunisere med socket til server, unngår ein å bruke Swingworker. Ein kunne då brukt Modbuskommuniksajon for å sende datagram til server. Å lage eit eige datagram vil krevje tid å sette seg inn i og utvikle. I SwingWorker har ein ferdige metode for å skrive å lytte ved hendingar. Det blei derfor sett på som unødvendig å bruke tid på å lage noko som ein hadde frå før som fungerte.

Interntesting av Ulstein IAS sikrar at planlagde funskjonalitetar er implementerte og fungerer. Under testing av MRU-simulator opp mot Ulstein IAS dukka fleire mindre feil opp som ein ikkje kunne sett på interntesting. Dette viser at for å teste Ulstein IAS Simulatoren optimalt må ein vere kopla opp mot Ulstein IAS. Simulatoren er då sett inn i sitt bruksområde, ein ser dermed om funksjonaliteten fungerer som tiltenkt.
Prosjektet er planlagt og utført etter fossefalmetoden. Det ferdige resultatet har heile tida vore godt definert gjennom prosjektet. Dette har gjort at ein har unngått større uforutsette endringar undervegs i prosjektet. Hovudmål og delmål er sett opp i gnatt-diagram der medlemar i prosjektgruppa har hatt kvar sine ansvarsområder. Kvart delmål har vorte ferdigstilt før det neste er byrja på. Resultatet av dette er at ein gjennom heile prosjektet har hatt god kontroll på kva som er gjort og kva som står att. For kvart hovudmål er det laga funksjonsskildring før utføring er sett i gang. Dette er med på å konkretisere målet. Etter gjennomført hovudmål er det laga testrapportar for å kontrollere at alle planlagde funksjonalitetar er laga og fungerer som dei skal. Dette er med på å ferdigstille hovudmålet. Ein bakdel med ein så kontrollert måte å styre, er at prosjektet blir lite fleksibelt. Feil og uspesifiserte endringar kan føre til forskyving i tidsplan. Møter med styringsgruppa er haldt kvar andre/tredje veke, her blir framdriftsrapport lagt fram for kvart møte. På denne måten har styringsgruppa kontroll på korleis prosjektgruppa ligg an og kan tidsnok kome med rettleiing.

KONKLUSJON

Målet med oppgåva var å lage ein test-simulator til Ulstein IAS for bruk under FAT. Med denne simulatoren får ein testa fleire segment som ikkje er mogleg å teste på eit så tidleg stadie utan. Simulatoren simulerer reelle situasjonar om bord i skip og kommuniserer med Ulstein IAS via seriellkommunikasjon og ein kontroller.
Ulstein IAS Simulator består av fire delsystem: MRU-simulator, Tankpeiling-simulator, Pumpe-simulator og IO-simulator. Kvart av delsystema er designa for å auke kvalitet og forståing av Ulstein IAS opp mot FAT. God planlegging og tidleg innhenting av informasjon har gjort at målet for prosjektet har vore godt definert gjennom heile prosessen. Prosjektgruppa har gjennom heile prosjektet hatt tett samarbeid med UPC. Spørsmål som har dukka opp har gått direkte til aktuelle personar i UPC, med rask respons og tilbakemeldingar.
Kvart delsystem fungerer som planlagt. Dei er delt opp i faner som gir god oversikt. Systemet er laga for å kunne utvidast om ynskjeleg. Vedlagt ligg programmeringsguide som tek for seg korleis ein kan legge til nye program i ny fane. Kvart delsystem er interntesta, men kun MRU-simulator er testa opp mot Ulstein IAS. Det finst ingen prosedyre på testing av tankpeiling ved dagens FAT. Nødvendige tanktabellar blir derfor ikkje produsert før ved oppstart ombord i skip. Disse tanktabellane må framskundast til FAT for å kunne ta i bruk Tankpeiling-simulator. Det har ikkje budd seg fleire moglegheiter for å teste dei andre simulatorane opp mot Ulstein IAS.
Metode og material som er tatt i bruk er basert på fagfelt prosjektgruppa har lært om gjennom tidlegare semester ved HIALS. I hovudsak gjeld dette Java-programmering, kommunikasjon og kontrollera. På denne måten har ein hatt fokus på funksjonalitet og kvalitet av simulatoren. Samstundes har ein bygd vidare og fått meir tyngde kring disse fagfelta. Prosjektgruppa har gjennom prosjektet sett seg inn i programmeringsdesign, eventhandtering, lesing av filer, GUI, seriellkommunikasjon og Modbus-kommunikasjon i Java. Ein har sett seg inn i oppbygging av kommunikasjonsprotokollane NMEA 0183 og EM3000 samt kommunikasjon mellom Java og kontroller.
Gjennomføring av prosjektet har gått som planlagt. Med ei god definert målsetting tidleg i prosjektet har ein kunne jobba målretta. Ein har ikkje møtt på større uforutsette problem. Nokre uspesifiserte segment har ført til mindre endringar, men ingen betydelege forseinkingar i tidsplan. Material og metode brukt i prosjektet har fungert som planlagt og vore gode val for å nå målet.
Ein har gjennom prosjektet lært korleis ein skal handtere eit større prosjekt. Med forprosjektrapporten går ein teoretisk gjennom heile prosjektet. Ein unngår dermed større problem undervegs i gjennomføringa. Gjennomtenkt og kontrollert tidsplan hjelp ein til å halde fokus og kontroll på prosjektet. Prosjektet er delt inn i hovudmål og delmål. Kvart hovudmål har ei milepæl å jobbe mot. Milepæl er nådd når testrapport er skriven. Med tidsplan og jamne møte med styringsgruppa har ein heile tida hatt jamt press på å levere etter planen.

REFERANSER

[1] J. Love, *Process Automation Handbook: A Guide to Theory and Practice*. Springer Science & Business Media, 2007.

[2] Ulstein, "Ulstein Group," 2014. [Online]. Available: www.ulstein.com. [Accessed: 20-Jan-2015].

[3] Ulstein, "Ulstein ias ®," 2014.

[4] Ulstein Power & Control, "Order Production Manual IAS," 2014.

[5] F. Frisvold and J. G. Moe, *Statistikk for ingeniører*. 2004.

[6] J. Haugan, *Tabeller og formler i fysikk*. 2011.

[7] Ulstein Power & Control, "Ulstein Tank sounding_OppdatertBLS_eng_101102," 2010.

[8] D. Bishop, "Qualifying Heave Performance at Sea," 2009. [Online]. Available: http://www.hydro-international.com/issues/articles/id1089-Qualifying_Heave_Performance_at_Sea.html. [Accessed: 20-Apr-2015].

[9] C. De Gelder, "Installation manual Kongsberg MRU," no. September, 2011.

[10] J. W. N. Com and J. Dong, *Network Dictionary*. Javvin Technologies Inc., 2007.

[11] S. Mackay, E. Wright, D. Reynders, and J. Park, *Practical Industrial Data Networks*. 2004.

[12] N. Dale, D. Joyce, and C. Weems, *Object-Oriented Data Structures Using Java*. Jones & Bartlett Publishers, 2011.

[13] R. Dixon, *Open Source Software Law*. Artech House, 2004.

[14] M. J. Laszlo, *Object-oriented Programming Featuring Graphical Applications in Java*. Addison-Wesley, 2002.

[15] Oracle, "Thread (Java Platform SE 7)," 2014. [Online]. Available: https://docs.oracle.com/javase/7/docs/api/java/lang/Thread.html. [Accessed: 20-May-2015].

[16] E. J. Bruno and G. Bollella, *Real-Time Java Programming: With Java RTS*. Pearson Education, 2009.

[17] "DNV Service Docs." [Online]. Available: https://exchange.dnv.com/servicedocuments/dnv. [Accessed: 20-May-2015].

[18] M. I. Mørk, "Forprosjektrapport (presentasjon)," 2015.

[19] A. Rolstadås, N. Olsson, A. Johansen, and J. A. Langlo, *Praktisk Prosjektledelse*. 2014.

[20] "SwingWorker (Java Platform SE 7)." [Online]. Available: https://docs.oracle.com/javase/7/docs/api/javax/swing/SwingWorker.html. [Accessed: 28-Apr-2015].

[21] "Jamod." [Online]. Available: http://jamod.sourceforge.net/. [Accessed: 28-Apr-2015].

[22] "JSSC - java serial port communication library." [Online]. Available: https://code.google.com/p/java-simple-serial-connector/. [Accessed: 20-May-2015].

[23] Oracle, "JFrame (Java Platform SE 7)," 2014. [Online]. Available: http://docs.oracle.com/javase/7/docs/api/javax/swing/JFrame.html. [Accessed: 27-Apr-2015].

[24] Oracle, "JPanel (Java Platform SE 7)," 2014. [Online]. Available: http://docs.oracle.com/javase/7/docs/api/javax/swing/JPanel.html. [Accessed: 27-Apr-2015].

[25] JFree, "JFreeChart," 2014. [Online]. Available: http://www.jfree.org/jfreechart/index.html. [Accessed: 27-Apr-2015].

[26] "Java SE Desktop Technologies - Java 3D API." [Online]. Available: http://www.oracle.com/technetwork/articles/javase/index-jsp-138252.html. [Accessed: 27-Apr-2015].

[27] "3D Ships & Vessels | Ship hms victory frigate nelson N270214 - 3D model (*.3ds) for exterior 3d visualization." [Online]. Available: http://archive3d.net/?a=download&id=038e4598. [Accessed: 27-Apr-2015].

[28] A. Norris, A. D. Wall, A. G. Bole, and W. O. Dineley, *Radar and ARPA Manual: Radar and Target Tracking for Professional Mariners, Yachtsmen and Users of Marine Radar*. Butterworth-Heinemann, 2005.

[29] L. Null, P. S. U. L. Null, and J. Lobur, *The Essentials of Computer Organization and Architecture*. Jones & Bartlett Publishers, 2014.

[30] "Modular I/O-System (Series 750, 753) | WAGO," 2015. [Online]. Available: http://global.wago.com/en/products/product-catalog/components-automation/modular-io-system-series-750-753/overview/index.jsp. [Accessed: 20-May-2015].

[31] "j2mod," 2015. [Online]. Available: http://sourceforge.net/projects/j2mod/files/?source=navbar. [Accessed: 18-May-2015].

VEDLEGG

Vedlegg 1	Forprosjektrapport
Vedlegg 2	Framdriftsrapport
Vedlegg 3	Framdriftsplan (Gantt-diagram)
Vedlegg 4	Møtereferat
Vedlegg 5	Avvik
Vedlegg 6	Funkjsonsskildringar
Vedlegg 7	Testrapportar
Vedlegg 8	Brukarmanual
Vedlegg 9	Programmeringsguide
Vedlegg 10	Klassediagram
Vedlegg 11	I/O-liste
Vedlegg 12	Poster

Vedlegg 1

Forprosjektrapport

FORPROSJEKT - RAPPORT

FOR BACHELOROPPGAVE

HØGSKOLEN I ÅLESUND

Aalesund University College

TITTEL:
Ulstein IAS Simulator

KANDIDATNUMMER(E):			
DATO: **28.01.2015**	EMNEKODE: **IE303612**	EMNE: **Bacheloroppgåve**	DOKUMENT TILGANG: - Open
STUDIUM: **AUTOMATISERINGSTEKNIKK**		ANT SIDER/VEDLEGG: /	BIBL. NR: - Ikkje i bruk -

OPPDRAGSGIVAR(E)/VEILEDER(E):
Ulstein Power & Control(UPC) / Rune Volden, Hans Støle og Anders Sætersmoen

OPPGÅVE/SAMANDRAG:

Ulstein IAS (Integrated Automation System) er eit fjernstyring og overvakingssystem for offshorefarty. Systemet er eit hjelpemiddel for teknisk personell ombord i fartyet, der dei kan oppnå oversikt over eit mangfald av statusar og alarmar frå ulike delsystem samt fjernstyring av ulike system.

For å kvalitetssikre Ulstein IAS leveransar vert det utført FAT(Factory Acceptance Test) før levering til kunde. Ein slik test går ut på å verifisere delar av anlegget på eit laboratorium, ved simulering av påkopla utstyr.

Dette prosjektet omhandlar å realisere eit system for simulering av utstyr og komponentar til bruk på FAT. I denne omgang vert det tatt føre seg simulering av pumper, tanksensorar og MRU(Motion Referance Unit). Simulatoren skal vere fleksibel og utvidbar for vidare utvikling.

Denne oppgaven er en eksamensbesvarelse utført av student(er) ved Høgskolen i Ålesund.

Postadresse
Høgskolen i Ålesund
N-6025 Ålesund
Norway

Besøksadresse
Larsgårdsvegen 2
Internett
www.hials.no

Telefon
70 16 12 00
Epostadresse
postmottak@hials.no

Telefax
70 16 13 00

Bankkonto
7694 05 00636
Foretaksregisteret
NO 971 572 140

INNHALD

1 INNLEIING

Oppgåva går ut på å lage ein simulator for fabrikktesting av Ulstein IAS som er Ulstein sitt automatiserte fjernstyrings og overvakingssystem. Simulatoren skal simulere reelle operasjonar om bord på skip, noko som gjer at Ulstein får testa fleire delar av systemet ved fabrikk test enn dei gjer i dag. Formålet med denne simulatoren er å kvalitetssikre produktet før levering til skip.

Problemstilling: Realisere og integrere ei metode for å sikre tiltenkt funksjonalitet til Ulstein IAS, ved å teste fullstendige system før levering frå fabrikk.

Eit av medlemene i prosjektgruppa er tilsett hjå UPC og har no studiepermisjon. Begge medlemene i prosjektgruppa har også erfaring frå UPC gjennom faget studiepoenggivande praksis som dei gjennomførte hjå UPC i Ulsteinvik hausten 2014. UPC hadde ei oppgåve som prosjektgruppa fann interessant og har i samarbeid med veileder hjå UPC bestemt hovudtrekka til oppgåva.

UPC utviklar og produserer elektroniske system for skip. I dag leverer dei brusystem, automatiserte fjernstyrings og overvakingssytem, power pakkar og integrerte kommunikasjonssystem til skip over heile verda. I Noreg har UPC avdeling i Ulsteinvik og Fremmerholen i Ålesund. [1]

2 FORKORTINGAR

- IAS - Integrated Automation System
- FAT - Factory Acceptance Test
- MRU - Motion Reference Unit
- UPC - Ulstein Power & Control
- DNV GL - Det Norske Veritas Germanischer Lloyd
- GUI - Graphical User Interface
- IMO - International Maritime Organization
- NMEA - National Marine Electronics Association
- I/O - Input/Output
- R&D - Research & Development

3 PROSJEKTORGANISASJON

3.1 Prosjektgruppe

Studentnummer(e)
120144 110442

3.1.1 Oppgåver for prosjektgruppa - organisering

Ansvar for at prosjektet går i riktig retning, til dels støtte med fagkunnskap og drøfte spørsmål som har betyding for prosjektets utvikling. Godkjenning av prosjektets tidsplanar og oppfølging av dette samt rapprotering til styringsgruppa kvar 14 dag etter oppstart (1.feb).

3.1.2 Oppgåver for prosjektleiar

Prosjektleiar har dagleg ansvar for gjennomføring av prosjektet. Prosjektleiar arbeidar etter dei mål og retningslinjer som er satt i prosjektet.
Som arbeidsoppgåver har prosjektleiar ansvaret å sjå til at prosjektplanen vert fyld, fastsette prosjektmøter, oppfølging av framdriftsplan. [2]

3.1.3 Oppgåver for sekretær

Sekretær er eit kontrollorgan til prosjektleiar, der kvalitetssikring av dokument og avgjersler vert eit samarbeid mellom sekretær og prosjektleiar.
Til arbeidsoppgåver har sekretær ansvaret for innkalling til møter, protokoll/møtereferat til prosjektmøte, og avlaste prosjektleiar. [2]

3.1.4 Oppgåver for øvrige medlemmar

Øvrige medlemmar har ansvar for å sjå til at prosjektet følg dei mål og retningslinjer som er satt i prosjektet.

3.2 Styringsgruppe (veileder og kontaktperson oppdragsgivar)

Navn	Adresse	Tlf-arb	Mob	e-post
Rune Volden	PO Box278, 6065 Ulsteinvik	7000 8000	9288 7753	Rune.volden@ulstein.com
Hans Støle	PO box 1517, 6025 Ålesund	7016 1267		hst@hials.no
Anders Sætersmoen	PO box 1517, 6025 Ålesund	7016 1636		anse@hials.no

4 AVTALER

4.1 Arbeidsstad og ressursar

- Tilgong til kontorplass/arbeidsplass hjå UPC i Ulsteinvik og Fremmerholen.
- Tilgong til rettleiing og teknisk assistanse innan for produktområdet Ulstein IAS.
- Nøkkelkort/tilgong til UPC i Ulsteinvik.

5 PROSJEKTBESKRIVELSE

5.1 Problemstilling - målsetting - hensikt

Problemstilling: Realisere og integrere ei metode for å sikre tiltenkt funksjonalitet til Ulstein IAS, ved å teste fullstendige system før levering frå fabrikk.

Målsetting: I denne oppgåva skal gruppa lage ein simulator for UPC sitt IAS system. Målet med simulatoren er å forbetre og kvalitetssikre fabrikktestane som UPC gjer for kvart anlegg. Måten dette blir gjort på i dag gjer at fleire kritiske delar av systemet ikkje blir testa før anlegget blir montert om bord i båt.

Effektmål:

- Kvalitetssikre fabrikktest
- Forenkle testprosedyre
- Brukarvennleg, for klasseselskap som må sette seg inn i systemet for kvar gang dei skal godkjenne system. Dette vil då effektivisere FAT.

Resultatmål: Sjå kap 5.2.

Prosessmål: Sjå vedlagt gantt diagram.

5.2 Krav til løysning eller prosjektresultat – spesifikasjon

Ulstein IAS Simulator, som vert resultatet for dette prosjektet skal brukast til å gjennomføre FAT (factory acceptance test) for Ulstein IAS. Simulatoren skal simulere påkopla hardware for å gjenskape ein best mogleg situasjon i forhold til verkelig situasjon.

Spesifikasjon:

- Simulere digital og analoge signaler for generell test.
- Simulere 2 pumper med følgjande signal: running, remote, stop, start, standby. Simulering av både feil og korrekt tilbakemelding på signal.
- MRU simulering.
- Funksjonstesting av tankpeiling ved simulering av trykkgivarar. Simulatoren tar oppslag i tanktabellar og MRU referanse for korrekt gjenskaping av ein ideell situasjon for tankpeiling.

***Funksjonelle krav**:*

- Brukarvenleg konfigurasjon
- Enkel i bruk.
- Modulbasert/objektorientert oppbygging.
- Moglegheit for simulere signal:

  Digitale: på-av.

  Analoge: 4-20mA, +-10vDC

  BUS: Modbus

  Seriell: RS422(NMEA)

Økonomi:

Prosjektet har ei økonomisk ramme på kr 5000,-.

Leveranse:

Prosjektet er rekna som levert når følgjande krav er oppfylt:

- Produktet har nådd spesifikasjonskrav og funksjonelle krav.
- Produktmanual er utarbeida.
- Prosjektrapport er utarbeida.

5.3 Planlagt framgangsmåte for utviklingsarbeidet – metodar

Prosjektstyringsmetode:
Prosjektgruppa vil ha eit tett samarbeid, der rollene som prosjektleiar og sekretær vil bli delt. Rollebytte vil skje for kvar veke for dei to gruppemedlemmane. Dette for at begge medlem skal få erfaring i dei forskjellige rollene.
Styringsgruppa vil bli oppdatert ved faste møte annankvar veke. Disse møta skal så godt som mogleg haldast tysdagar kl. 10.15. Prosjektgruppa har ansvar for å kalle inn til møte. Til kvart møte skal det fyllast ut ein framdriftsrapport som skildrar kva som er gjort den førre perioden og kva som skal gjerast i den neste. Den skal også innehalde problem vi har støtt på og eventuelle avvik.

Utviklingsmetode:
For utvikling av simulatoren som skal lagast vil gruppa bruke metode for programmering og kommunikasjon som dei allereie har kjennskap til. Styringsgruppa var einige i at dette ville gagne prosjektet, då prosjektgruppa kan ha fokus på utvikling av simulatorens virkemåte.
Prosjektgruppa har våre i kontakt med fleire som arbeider kring UIAS i dag for å kome fram til kva som skal simulerast. Disse kontaktane kan gruppa også kontakte ved eventuelle spørsmål som kan dukke opp.
Når det gjeld simulering av tankpeiling må prosjektgruppa tileigne seg ein del informasjon kring Ulstein sine tanktabellar og deira virkemåtar.

5.4 Informasjonsinnsamling – utført og planlagt

I forprosjektperioden er det framskaffa informasjon om eksisterande rutinar rundt FAT samt utviklingsverktøy for testing av nye system. Informasjonen er teken frå ulike fagdisiplinar innan for produktet Ulstein IAS gjennom møter og samtaler. Dei ulike disiplinane var R&D leiar, Produktingeniørar, serviceingeniør og utviklingsingeniørar. Ut frå dette er det framkome ein spesifikasjon og beskriving på kva ein Ulstein IAS simulator skal innehalde. Spesifikasjonen finnes i kap 5.2.
Vidare er det sett opp at følgjande informasjon skal framskaffast:

- Klassekrav(DNV GL, evt IMO) til FAT for eit slikt anlegg.
- Eksisterande FAT dokumentasjon for Ulstein IAS.
- Innføring i tanktabellar og virkemåte, med tanke på simulering av tankpeiling.
- Dokumentasjon til protokollar for dei ulike seriell- og bus-kommunikasjon.
- Oversikt for planlagde FAT for Ulstein IAS, vår 2015.

5.5 Vurdering – analyse av risiko

Prosjektgruppa meiner at rammene som sett kring utføring og funksjonalitet i prosjektet skal vere realiserbare innanfor tidsfristen.
Hovudmål og delmål som er sett av prosjektgruppa er viktige for å oppnå ynskt resultat. Sidan prosjektgruppa skal bruke programeringsplatform og kommunikasjonsmetode dei allereie har kjennskap utgjer ein sikkerheitsfaktor. Å lære seg nye metode her ville tatt mykje tid. Prosjektgruppa får no konsentrere seg om å utvikle ein simulator som kan tilfredstille UPC sine krav.
Truslar mot suksess kan vere simulering av tankpeiling, her er nokre usikkerheiter som ikkje er på plass enda. Her vil prosjektgruppa kontakte UPC for informasjon og rettleiing.

5.6 Hovudaktivitetar i vidare arbeid

Oversikt over aktivitetar er vedlagt rapporten. Gantt diagram framstilt med hovudaktivitetar nedbråte i delaktivitetar.

5.7 Framdriftsplan – styring av prosjektet

5.7.1 Hovudplan

I hovudplanen er det sett opp 7 hovuddelar for gjennomføringa av prosjektet. Dei ulike Hovuddelane er definerte modular i prosjektet. Frå og med hovudmål A2 til A6 er det sett opp funksjonsskildring som første delmål. Funksjonsskildring er ein del av kvalitetssikringa av hovudmålet. I sluttfasen av hovudmålet vert det utført ein funskjonstest. Denne skal samsvare med tenk funksjonalitet som vil resultere i at hovudmålet er nådd.

A1: Forprosjekt: I denne fasen er skal det utarbeidast ein plan for innhaldet og gjennomføringa av prosjektet. Det skal etableras rutinar og føringslinjer for oppfølging og avviksbehandling. Desse skal presenterast i forprosjektrapport.

A2: Konfigurasjon kontroller: Den første del av praktisk arbeid startar med å kople opp maskinvarer å sette opp grensesnittet mellom datamaskin og kontroller. Her er det valt å kommunisere mellom einheitane ved hjelp av modbus kommunikasjon. Vidare skal all IO konfigurerast i kontroller.

A3: GUI base: I botnen av brukargrensesnittet skal det ligg ein fleksibel GUI base som fundament til delsystema. Denne skal vere veldefinert og enkel. I tillegg skal det realiserast eit delsystem for simulering av generelle signal som analog og digital.

A4: Pumpe-simulator: Eit delsystem av simulatoren som simulerer 2 pumper. Desse pumpene skal simulere både riktig og feil virkemåte i signalgiving. Brukargrensesnittet skal grafisk framstillast som eit delsystem i GUI basa.

A5: Tankpeiling-simulator: Simulere trykkgivarar ved simulering av 4-20mA signal mot IAS. Tankane vert grafisk framstilt som eit eige delsystem i GUI base der man kan sette tanknivå som igjen generera eit 4-20ma signal ut frå tabell.

A6: MRU simulator: Simulere bevegelse av eit fartøy ved generering av pitch, roll og yaw. Denne dataen skal samlast i ein standardisert seriell protokoll type NMEA og sendes til IAS via ein ethernet-seriel konverter.

A7: Dokumentasjon: Dokumentering av prosjektet vert ein flytande aktivitet gjennom heile prosjektet med noko høgre intensitetsnivå mot slutten.

Kvar enda hovudaktivitet vert sett på som ein milepæl i prosjektet. Ved hjelp av gantt-diagrammet som vise oversikta over aktivitetar vil man kunne sjå når dei ulike milepælane skal vere nådd.

5.7.2 Styringshjelpemidlar

For å halde oversikt over framdrifta i prosjektet så vil gantt-diagram bli brukt. Dette diagrammet vil vere inndelt i hovudmål som igjen har underpunkter som utgjer delmål. Her vil prosjektgruppa fylle ut prosentvis kor langt dei har kome i delmålet/hovudmålet. Gantt-diagrammet viser også tidrom når dei forskjellige måla er planlagt start og slutt. Kven som har ansvar for oppgåva kjem også fram her.
Denne framdriftsplanen kan bli endra undervegs i prosjektet. Avvik vil bli tatt hand om med eigen avviksrapport.

5.7.3 Intern kontroll – evaluering

Oppfølging av framdrift vert gjort ved statusmøter i prosjektgruppa kvar veke. Her vert det drøfta framdriftsplan med vekt på faktisk arbeid i forhold til planlagt. Eit kjenneteikn på eit mål som er nådd, er at funksjonsskildring samsvarar med tenkt funskjonalitet.

5.8 Avgjersler – Avgjersleprosess

Avgjersle kring fastsetting av oppgåva og kva som skal simulerast er tatt i samarbeid med styringsgruppa. Viktige punkt som dukkar opp undervegs i prosjektet vil også leggast fram for styringsgruppa for endeleg avgjersle. Mindre avgjersle kan bli tatt hand om av prosjektgruppa, eventuelt med rådføring frå vegleiar.

6 DOKUMENTASJON

6.1 Rapporter og tekniske dokumenter

- Undervegs i prosjektet vert det skrifteleg rapportert for kvart hovudmål.
- IAS simulatoren vert dokumentert i form av ei brukarveiledning.
- Hovudrapport.

7 PLANLAGTE MØTER OG RAPPORTER

7.1 Møter

7.1.1 Møter med styringsgruppa

Møter med styringsgruppa er sett til annankvar tysdag gjennom prosjektet med fast tidspunkt kl. 10.15. På grunn av påska så blir møtet som eigentleg skulle vere 31.03.15 flytta fram til 24.03.15.
Prosjektgruppa sett opp agenda og kallar inn til møte.

	Planlagde møtedatoar	Tidspunkt
1	03.02.2015	kl. 10.15
2	17.02.2015	kl. 10.15
3	03.03.2015	kl. 10.15
4	17.03.2015	kl. 10.15
5	24.03.2015	kl. 10.15
6	14.04.2015	kl. 10.15
7	28.04.2015	kl. 10.15
8	12.05.2015	kl. 10.15

7.1.2 Prosjektmøter

Det vert heldt prosjektmøte kvar veke med følgjande agenda:

1. Oppfølging av framdriftsplan
2. Avvikshandtering
3. Kva er gjort sidan sist?
4. Eventuelt

7.2 Periodiske rapporter

7.2.1 Framdriftsrapporter (inkl. milepæl)

Framdriftsrapport blir laga ut frå Høgskulen i Ålesund sin mal. Denne rapporten blir fylt ut at prosjektgruppa og lagt fram ved kvart møte med styringsgruppa.

8 PLANLAGT AVVIKSBEHANDLING

Registrering av avvik vert gjort ved utfylling av avviksskjema. Skjemaet er delt inn i tre risikogrupper:

1. Lav risiko: Handterast innan 10 virkedagar. Varsling til prosjektgruppa.
2. Middels risiko: Handterast innan 5 virkedagar. Varsling til prosjektgruppa.
3. Høg risiko: Handterast innan 3 virkedagar. Varsling til styringsgruppa.

Avviksbehandling er fast agenda på prosjektmøta.

9 UTSTYRSBEHOV/FORUTSETNINGER FOR GJENNOMFØRING

Utstyr:

- Datamaskin/server
- Monitor
- Kontroller med støtte for modbus (f.eks Wago 750-881)
- I/O modul - 1 åttekanals digital innput
- I/O modul – 1 åttekanals digital output
- I/O modul – 3 firekanals analog output
- Moxa 5150I seriellconverter

Programvare:

- Nport Administration Suite for Moxa
- Codesys for Wago
- Ubuntu Operativsystem for server
- Eclipse for programering

10 REFERANSER

[1] Ulstein, "Ulstein Group," *2014*. [Online]. Available: www.ulstein.com. [Accessed: 20-Jan-2015].

[2] A. Rolstadås, N. Olsson, A. Johansen, and J. A. Langlo, *Praktisk Prosjektledelse*. 2014.

VEDLEGG

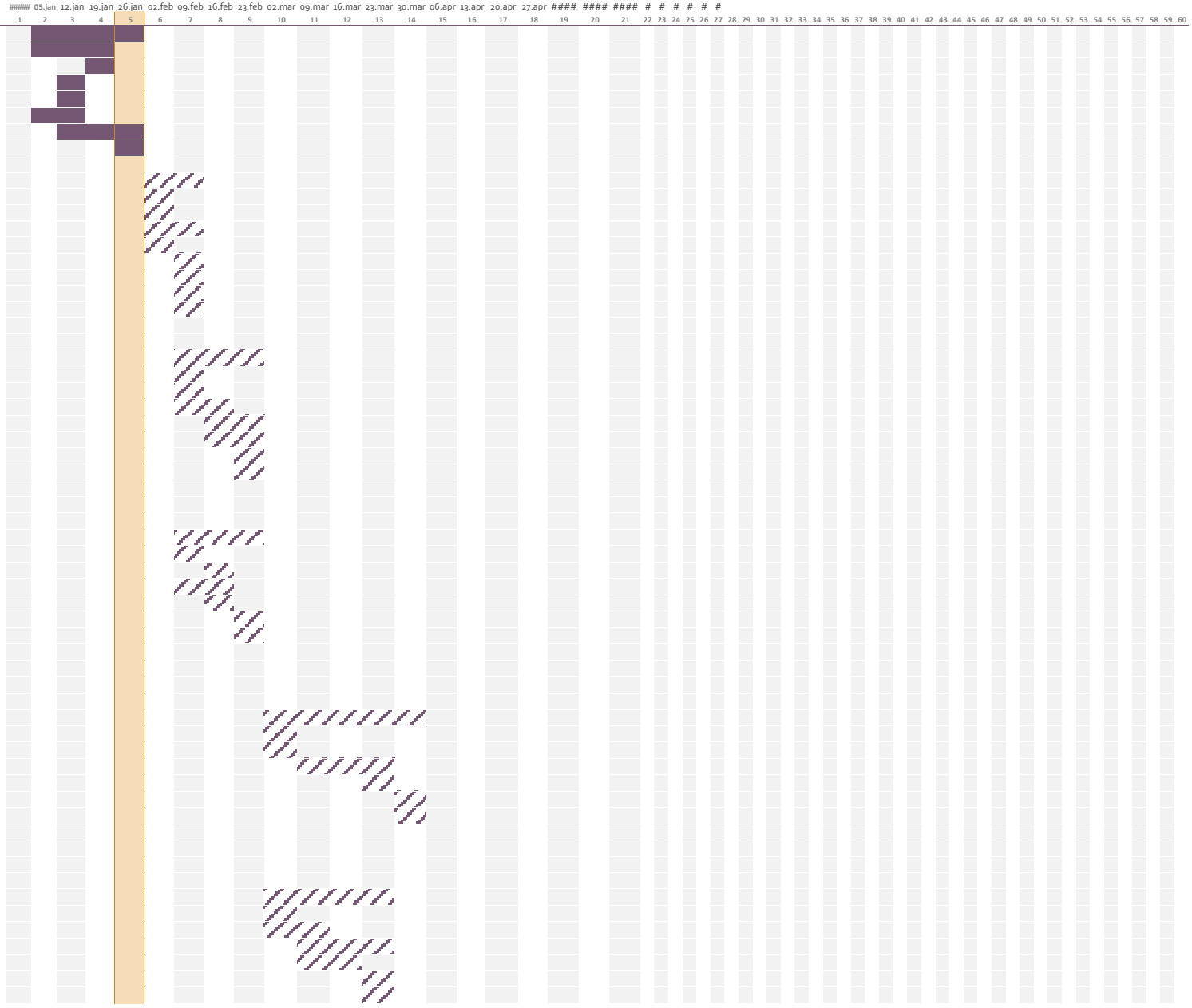
Vedlegg 1	Gantt diagram - framdriftsplan
Vedlegg 2	Mal avviksrapport

IAS SIMULATOR

Period Highlight: 5

Plan | Actual | % Complete | Actual (beyond plan) | % Complete (beyond plan)

	ACTIVITY	PLAN START	PLAN DURATION	ACTUAL START	ACTUAL DURATION	PERCENT COMPLETE
A1	**Forprosjekt**	2	4	2	4	100 %
A11	Informasjonsinnsamling	2	3	2	3	100 %
A12	Framsakaffe informasjon om krav til innhald	4	1	4	1	100 %
A13	Fordele oppgåver	3	1	3	1	100 %
A14	Møte med veiledere og bedrift	3	1	3	1	100 %
A15	Møte med bedrift - brainstorming	2	2	2	2	100 %
A16	Forprosjektrapport	3	3	3	3	100 %
A17	Korrektur	5	1	5	1	100 %
A2	**Konfigurasjon kontroller(PLS)**	6	2			0 %
A21	funksjonsbeskrivelse	6	1			0 %
A22	Oppsett hardware	6	1			0 %
A23	Konfigurere oppsett	6	2			0 %
A24	Modbus kommunikasjon	6	1			0 %
A25	IO konfigurering	7	1			0 %
A26	Modbus kommunikasjon med Java	7	1			0 %
A27	Test av funksjonalitet	7	1			0 %
A28	Rapport	7	1			0 %
A29	tom					
A3	**GUI base**	7	3			0 %
A31	funksjonsbeskrivelse	7	1			0 %
A32	Grafikk	7	1			0 %
A33	Hovudbilete	7	2			0 %
A34	Standard fane	8	2			0 %
A35	Fane for digitale/analoge signal for generel test	8	2			0 %
A36	Test av funksjonalitet	9	1			0 %
A37	Rapport	9	1			0 %
A38	tom					
A39	tom					
A4	**Pumpe-simulator**	7	3			0 %
A41	funksjonsbeskrivelse	7	1			0 %
A42	Grafikk	8	1			0 %
A43	Funksjonsklasse	7	2			0 %
A44	Grafisk klasse	8	1			0 %
A45	Test av funksjonalitet	9	1			0 %
A46	Rapport	9	1			0 %
A47	tom					
A48	tom					
A49	tom					
A5	**Tankpeiling-simulator**	10	5			0 %
A51	funksjonsbeskrivelse	10	1			0 %
A52	Grafikk	10	1			0 %
A53	Funksjonsklasse	11	3			0 %
A54	Grafisk klasse	13	1			0 %
A55	Test av funksjonalitet	14	1			0 %
A56	Rapport	14	1			0 %
A57	tom					
A58	tom					
A59	tom					
A6	**MRU-simulator**	10	4			0 %
A 61	funksjonsbeskrivelse	10	1			0 %
A 62	Grafikk	10	2			0 %
A 63	Funksjonsklasse	11	3			0 %
A 64	Grafisk klasse	11	2			0 %
A 65	Test av funksjonalitet	13	1			0 %
A 66	Rapport	13	1			0 %



IAS SIMULATOR

Period Highlight: 5

Plan | Actual | % Complete | Actual (beyond plan) | % Complete (beyond plan)

	ACTIVITY	PLAN START	PLAN DURATION	ACTUAL START	ACTUAL DURATION	PERCENT COMPLETE
A 67	tom					
A 68	tom					
A 69	tom					
A7	**Dokumentasjon**	14	7			**0 %**
A71	Brukerveiledning	14	2			**0 %**
A72	Programmerings guide	14	2			**0 %**
A73	Sluttføring hovudrapport	16	5			**0 %**
A74	tom					
A75	tom					
A76	tom					
A77	tom					
A78	tom					
A79	tom					

PERIODS

#####	05.jan	12.jan	19.jan	26.jan	02.feb	09.feb	16.feb	23.feb	02.mar	09.mar	16.mar	23.mar	30.mar	06.apr	13.apr	20.apr	27.apr	####	####	####	#	#	#	#	#	#
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27

28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60

Avviksrapport

Saksnummer:	Emne:	Rapportert Dato:	Avviksnumer:

Avvik:

Kort beskrivelse av avviket....

Prioritet:

Lav □
Middels □
Høg □

Korrigerende Tiltak:

Kort beskrivelse av tiltak.....

Korrigerende tiltak utført. Dato/sign:

Vedlegg 2

Framdriftsrapport

ID301702 Hovedprosjekt	Prosjekt Ulstein IAS sumulator	Antall møter denne periode 1). 1	Firma - Oppdragsgiver Høgskolen i Ålesund /	Side 1 av 2
Rapport fra prosess Framdriftsrapport	Periode/uke(r) 6-7	Antall timer denne per. (fra logg) -	Prosjektgruppe (navn) Kristian Østgaard, Rolf Ottar Rovde	Dato 12.02.15

Hovedhensikt / fokus for arbeidet i denne perioden

Fokuset i veke 4-5 har vår å få på plass og konfigurert hardware. Vi har laga program i JAVA for handtering av Modbuskommunikasjon samt laga testfunksjon for setting av inngangar og utgangar av I/O. Oppsett av hardware er gjort i forhold til funksjonsbeskrivelsen som vart laga i starten av perioda.

Planlagte aktiviteter i denne perioden

1. Funksjonsbeskrivelse PLS IO.
2. Oppsett hardware PLS og server.
3. Konfigurere PLS og server.
4. Kommunikasjon mellom dei ulike komponentane.
5. Programmering av PLS.
6. Skalering av signal.
7. Link mellom modbus og IO.
8. Skaffe oversikt over FAT og planlegge framtidig test.
9. Funksjonalitetstest av PLS IO
10. Rapport

Virkelig gjennomførte aktiviteter i denne perioden

Avvik på punkt 8.
Ellers er alle planlagde aktivitetar for denne perioda fullført.

Beskrivelse av/begrunnelse for eventuelle avvik mellom planlagte og virkelige aktiviteter

Avvik punkt 8: Det er skaffa ein oversikt over framtidig FAT. Det er ikkje planlagt nokon framtidig test endå.

Beskrivelse av /begrunnelse for endringer som nå ønskes i selve prosjektets innhold eller i den videre framgangsmåten - eller framdriftsplanen

-

Hovederfaring fra denne perioden

- Vi har på ny sett oss inn i konfigurering av kontroller og I/O.
- Sett oss inn i Modbus -ommunikasjon.
- Vi har sett nytt av å ha ein funksjonsbeskrivelse på plass før vi byrja med utvikling.

Hovedhensikt/fokus neste periode

I den neste perioda vil fokuset dreie over på å starte utvikling av GUI for simulatoren. Vi vil starte med å lage e in base for utforminga av GUIen. Pumpe-simulator vil også bli starta på, i første omgang vil vi ta for oss grafisk design og starting og stopping av pumper.

1) Noter her kort tilbakemelding om antall møter – fordelt på typer (interne, styringsgruppe, møte med veileder) - i denne rapportperioden

ID301702 Hovedprosjekt	Prosjekt Ulstein IAS sumulator	Antall møter denne periode 1). 1	Firma - Oppdragsgiver Høgskolen i Ålesund /	Side 2 av 2
Rapport fra prosess Framdriftsrapport	Periode/uke(r) 6-7	Antall timer denne per. (fra logg) -	Prosjektgruppe (navn) Kristian Østgaard, Rolf Ottar Rovde	Dato 12.02.15

Planlagte aktiviteter neste periode

A3 GUI base Rolf Ottar
A31 Funksjonsbeskrivelse
A32 Grafikk
A33 Hovudbilete
A34 Standard fane

A4 Pumpe-simulator Kristian
A41 Funksjonsbeskrivelse
A42 Grafikk
A43 Funksjonsklasse
A431 Start
A432 Stop

Annet

Ønske om /behov for veiledning, tema i undervisningen – drøfting ellers

Godkjenning/signatur gruppeleder	Signatur øvrige gruppedeltakere

1) Noter her kort tilbakemelding om antall møter – fordelt på typer (interne, styringsgruppe, møte med veileder) - i denne rapportperioden

ID301702 Hovedprosjekt	Prosjekt Ulstein IAS Simulator	Antall møter denne periode 1). 1	Firma - Oppdragsgiver Høgskolen i Ålesund / Ulstein Power & Control	Side 1 av 2
Rapport fra prosess **Framdriftsrapport**	Periode/uke(r) 2-3	Antall timer denne per. (fra logg) -	Prosjektgruppe (navn) Kristian Østgaard, Rolf Ottar Rovde	Dato 13.01.15

Hovedhensikt / fokus for arbeidet i denne perioden

Denne perioden har vert semesterstart. Fokus har vert å komme i gang med forprosjektrapporten, samt etablere kontakt med UPC(Ulstein Power & Control) og HIALS(Høgskolen i Ålesund).

Planlagte aktiviteter i denne perioden

1. Komme igang med forprosjekt rapporten
2. Møte med UPC: Diskutere oppgåve
3. Møte med veiledar få HIALS og UPC. Info om prosjektet

Virkelig gjennomførte aktiviteter i denne perioden

Alle aktivitetar som blei planlagt denne perioden vart gjennomført.

Beskrivelse av/begrunnelse for eventuelle avvik mellom planlagte og virkelige aktiviteter

-

Beskrivelse av /begrunnelse for endringer som nå ønskes i selve prosjektets innhold eller i den videre framgangsmåten - eller framdriftsplanen

-

Hovederfaring fra denne perioden

Det er viktig med tidlig start for å komme raskt i gang med prosjektet. Etablering av gode arbeidsvanar og rutinar er bra for kontinuiteten og flyten i arbeidet som vert lagt ned.

Hovedhensikt/fokus neste periode

Neste periode skal forprosjektraporten vere ferdigstilt(31.jan). Dette medførar at problemstilling og hovudmål for prosjektet er sett, samt retningslinjene for gjennomføring. Vegen vidare i prosjektet skal vere klar og tydelig med veldefinerte mål/delmål og realistiske målsetjingar.

Planlagte aktiviteter neste periode

1. Ferdigstilling av forprosjektrapport.
2. Etablering av total framdriftsplan for prosjektet, nedbråte i delaktivitetar (Gantt diagram).

Annet

-

1) Noter her kort tilbakemelding om antall møter – fordelt på typer (interne, styringsgruppe, møte med veileder) - i de.nne rapportperioden

ID301702 Hovedprosjekt	Prosjekt Ulstein IAS Simulator	Antall møter denne periode 1). 1	Firma - Oppdragsgiver Høgskolen i Ålesund / Ulstein Power & Control	Side 2 av 2
Rapport fra prosess Framdriftsrapport	Periode/uke(r) 2-3	Antall timer denne per. (fra logg) -	Prosjektgruppe (navn) Kristian Østgaard, Rolf Ottar Rovde	Dato 13.01.15

Ønske om /behov for veiledning, tema i undervisningen – drøfting ellers -	
Godkjenning/signatur gruppeleder Kristian Østgaard	Signatur øvrige gruppedeltakere Rolf Ottar Rovde

1) Noter her kort tilbakemelding om antall møter – fordelt på typer (interne, styringsgruppe, møte med veileder) - i de.nne rapportperioden

ID301702 Hovedprosjekt	Prosjekt Ulstein IAS Simulator	Antall møter denne periode 1). 1	Firma - Oppdragsgiver Høgskolen i Ålesund / Ulstein Power & Control	Side 1 av 2
Rapport fra prosess Framdriftsrapport	Periode/uke(r) 4-5	Antall timer denne per. (fra logg) -	Prosjektgruppe (navn) Kristian Østgaard, Rolf Ottar Rovde	Dato 28.01.15

Hovedhensikt / fokus for arbeidet i denne perioden

Neste periode skal forprosjektraporten vere ferdigstilt(31.jan). Dette medfører at problemstilling og hovudmål for prosjektet er sett, samt retningslinjene for gjennomføring. Vegen vidare i prosjektet skal vere klar og tydelig med veldefinerte mål/delmål og realistiske målsetjingar.

Planlagte aktiviteter i denne perioden

1. Ferdigstilling av forprosjektrapport.
2. Etablering av total framdriftsplan for prosjektet, nedbråte i delaktivitetar (Gantt diagram).

Virkelig gjennomførte aktiviteter i denne perioden

Alle aktivitetar som blei planlagt denne perioden vart gjennomført.

Beskrivelse av/begrunnelse for eventuelle avvik mellom planlagte og virkelige aktiviteter

-

Beskrivelse av /begrunnelse for endringer som nå ønskes i selve prosjektets innhold eller i den videre framgangsmåten - eller framdriftsplanen

-

Hovederfaring fra denne perioden

- Det er erfart at oppdeling av prosjektet resulterer i bedre oversikt over krav til resursar og tidsforbruk.
- Gjennom møter og samtaler med ulike fagpersonar innan Ulstein IAS, har resultert i ein brei dekning av problemstillinga. På basis av dette er spesifikasjonen til prosjektet sett.
- Planlegging av prosjektet har tvunge oss til å tenkte heilheitlig, og gitt oss eit bra grunnlag på det totale prosjektet.

Hovedhensikt/fokus neste periode

Neste periode skal det fokuserast på materiell (hardware) og konfigurering av dette. Kommunikasjon mellom hovudkomponentane via bus.

Planlagte aktiviteter neste periode

1. Funksjonsbeskrivelse PLS IO.
2. Oppsett hardware PLS og server.
3. Konfigurere PLS og server.
4. Kommunikasjon mellom dei ulike komponentane.
5. Programmering av PLS.
6. Skalering av signal.
7. Link mellom modbus og IO.
8. Skaffe oversikt over FAT og planlegge framtidig test.
9. Funksjonalitetstest av PLS IO
10. Rapport

1) Noter her kort tilbakemelding om antall møter – fordelt på typer (interne, styringsgruppe, møte med veileder) - i de.nne rapportperioden

ID301702 Hovedprosjekt	Prosjekt Ulstein IAS Simulator	Antall møter denne periode 1). 1	Firma - Oppdragsgiver Høgskolen i Ålesund / Ulstein Power & Control	Side 2 av 2
Rapport fra prosess Framdriftsrapport	Periode/uke(r) 4-5	Antall timer denne per. (fra logg) -	Prosjektgruppe (navn) Kristian Østgaard, Rolf Ottar Rovde	Dato 28.01.15

Annet -	
Ønske om /behov for veiledning, tema i undervisningen – drøfting ellers -	
Godkjenning/signatur gruppeleder Kristian Østgaard	Signatur øvrige gruppedeltakere Rolf Ottar Rovde

1) Noter her kort tilbakemelding om antall møter – fordelt på typer (interne, styringsgruppe, møte med veileder) - i de.nne rapportperioden

ID301702 Hovedprosjekt	Prosjekt Ulstein IAS sumulator	Antall møter denne periode 1). 1	Firma - Oppdragsgiver Høgskolen i Ålesund /	Side 1 av 2
Rapport fra prosess Framdriftsrapport	Periode/uke(r) 6-7	Antall timer denne per. (fra logg) -	Prosjektgruppe (navn) Kristian Østgaard, Rolf Ottar Rovde	Dato 12.02.15

Hovedhensikt / fokus for arbeidet i denne perioden

Fokuset i veke 4-5 har vår å få på plass og konfigurert hardware. Vi har laga program i JAVA for handtering av Modbuskommunikasjon samt laga testfunksjon for setting av inngangar og utgangar av I/O. Oppsett av hardware er gjort i forhold til funksjonsbeskrivelsen som vart laga i starten av perioda.

Planlagte aktiviteter i denne perioden

1. Funksjonsbeskrivelse PLS IO.
2. Oppsett hardware PLS og server.
3. Konfigurere PLS og server.
4. Kommunikasjon mellom dei ulike komponentane.
5. Programmering av PLS.
6. Skalering av signal.
7. Link mellom modbus og IO.
8. Skaffe oversikt over FAT og planlegge framtidig test.
9. Funksjonalitetstest av PLS IO
10. Rapport

Virkelig gjennomførte aktiviteter i denne perioden

Avvik på punkt 8.
Ellers er alle planlagde aktivitetar for denne perioda fullført.

Beskrivelse av/begrunnelse for eventuelle avvik mellom planlagte og virkelige aktiviteter

Avvik punkt 8: Det er skaffa ein oversikt over framtidig FAT. Det er ikkje planlagt nokon framtidig test endå.

Beskrivelse av /begrunnelse for endringer som nå ønskes i selve prosjektets innhold eller i den videre framgangsmåten - eller framdriftsplanen

-

Hovederfaring fra denne perioden

- Vi har på ny sett oss inn i konfigurering av kontroller og I/O.
- Sett oss inn i Modbus -ommunikasjon.
- Vi har sett nytt av å ha ein funksjonsbeskrivelse på plass før vi byrja med utvikling.

Hovedhensikt/fokus neste periode

I den neste perioda vil fokuset dreie over på å starte utvikling av GUI for simulatoren. Vi vil starte med å lage e in base for utforminga av GUIen. Pumpe-simulator vil også bli starta på, i første omgang vil vi ta for oss grafisk design og starting og stopping av pumper.

1) Noter her kort tilbakemelding om antall møter – fordelt på typer (interne, styringsgruppe, møte med veileder) - i denne rapportperioden

ID301702 Hovedprosjekt	Prosjekt Ulstein IAS sumulator	Antall møter denne periode 1). 1	Firma - Oppdragsgiver Høgskolen i Ålesund /	Side 2 av 2
Rapport fra prosess Framdriftsrapport	Periode/uke(r) 6-7	Antall timer denne per. (fra logg) -	Prosjektgruppe (navn) Kristian Østgaard, Rolf Ottar Rovde	Dato 12.02.15

Planlagte aktiviteter neste periode

A3 GUI base Rolf Ottar
A31 Funksjonsbeskrivelse
A32 Grafikk
A33 Hovudbilete
A34 Standard fane

A4 Pumpe-simulator Kristian
A41 Funksjonsbeskrivelse
A42 Grafikk
A43 Funksjonsklasse
A431 Start
A432 Stop

Annet

Ønske om /behov for veiledning, tema i undervisningen – drøfting ellers

Godkjenning/signatur gruppeleder	Signatur øvrige gruppedeltakere

1) Noter her kort tilbakemelding om antall møter – fordelt på typer (interne, styringsgruppe, møte med veileder) - i denne rapportperioden

ID301702 Hovedprosjekt	Prosjekt IAS Simulator	Antall møter denne periode 1).	Firma - Oppdragsgiver Høgskolen i Ålesund /	Side 1 av 2
Rapport fra prosess Framdriftsrapport	Periode/uke(r) 8-9	Antall timer denne per. (fra logg)	Prosjektgruppe (navn)	Dato 27.02.15

Hovedhensikt / fokus for arbeidet i denne perioden

Denne perioden har konsentrert seg om etablering av grafisk brukargrensesnitt og simulering av pumpesystem. Som beskreve i forprosjektrapport er det brukt eit objektorientert språk (Java) der det er stort fokus på lav kobling og høgt samhald mellom klassene. Dette har resultert i eit gjennomtenkt system der blandt anna kodeduplisering ikkje skal forekome.

Planlagte aktiviteter i denne perioden

Rolf Ottar:
A3 GUI base
A31 Funksjonsbeskrivelse
A32 Grafikk
A33 Hovudbilete
A34 Standard fane

Kristian:
A4 Pumpe-simulator
A41 Funksjonsbeskrivelse
A42 Grafikk
A43 Funksjonsklasse
A431 Start
A432 Stop

Virkelig gjennomførte aktiviteter i denne perioden

Alle planlagte aktivitetar er gjennomført. Det blei og gjennomført fleire aktivitetar en planlagt. Følgjande punkt vart og gjennomført:

A433 Running
A434 Remote
A44 Grafisk klasse
A45 Test av funksjonalitet
A46 Rapport

A35 Fane for digitale/analoge signal for generell test
A36 Test av funksjonalitet
A37 Rapport

Beskrivelse av/begrunnelse for eventuelle avvik mellom planlagte og virkelige aktiviteter

Effektiv og gode arbeidsdagar resulterte i at vi kunne starte på aktivitetar som ikkje var planlagt.

Beskrivelse av /begrunnelse for endringer som nå ønskes i selve prosjektets innhold eller i den videre framgangsmåten - eller framdriftsplanen

-

1) Noter her kort tilbakemelding om antall møter – fordelt på typer (interne, styringsgruppe, møte med veileder) - i denne rapportperioden

ID301702 Hovedprosjekt	Prosjekt IAS Simulator	Antall møter denne periode 1).	Firma - Oppdragsgiver Høgskolen i Ålesund /	Side 2 av 2
Rapport fra prosess Framdriftsrapport	Periode/uke(r) 8-9	Antall timer denne per. (fra logg)	Prosjektgruppe (navn)	Dato 27.02.15

Hovederfaring fra denne perioden

Sjølv eit enkelt system som start og stopp av pumper krevjar god planlegging og gjennomtenkt funksjonalitet for å skape ein stabil programvare. Grunnlegjande metodar for programmering går att i alle koder, uavhengig av kompleksitet til systemet som skal programmeres.

Hovedhensikt/fokus neste periode

Fokusering på tanksimulering og MRU simulering. Desse går hand i hand.

Planlagte aktiviteter neste periode:

Rolf Ottar:
A5 Tankpeiling-simulator
A51 Funksjonsbeskrivelse
A52 Grafikk
A53 Funksjonsklasse
A531 Oppslag i tanktabell
A532 Skalering av signal

Kristian:
A6 MRU-simulator
A 61 funksjonsbeskrivelse
A 62 Grafikk
A621 Sjekk ut om 3D grafikk av båt er mogleg
A622 Framstilling av data grafisk(api)
A 63 Funksjonsklasse
A631 NMEA(protokoll)
A632 EM3000(protokoll)
A633 Seriell kommunikasjon

Annet

-

Ønske om /behov for veiledning, tema i undervisningen – drøfting ellers

-

Godkjenning/signatur gruppeleder	Signatur øvrige gruppedeltakere
Rolf Ottar Rovde	Kristian Østgaard

1) Noter her kort tilbakemelding om antall møter – fordelt på typer (interne, styringsgruppe, møte med veileder) - i denne rapportperioden

ID301702 Hovedprosjekt	Prosjekt IAS Simulator	Antall møter denne periode 1).	Firma - Oppdragsgiver Høgskolen i Ålesund /	Side 1 av 3
Rapport fra prosess Framdriftsrapport	Periode/uke(r) 10-12	Antall timer denne per. (fra logg)	Prosjektgruppe (navn)	Dato 20.03.15

Hovedhensikt / fokus for arbeidet i denne perioden

Denne perioden har det vert fokus på simulering av MRU og tankpeilingsystem.

Planlagte aktiviteter i denne perioden

Rolf Ottar:
A5 Tankpeiling-simulator
A51 Funksjonsbeskrivelse
A52 Grafikk
A53 Funksjonsklasse
A531 Oppslag i tanktabell
A532 Skalering av signal

Kristian:
A6 MRU-simulator
A 61 funksjonsbeskrivelse
A 62 Grafikk
A621 Sjekk ut om 3D grafikk av båt er mogleg
A622 Framstilling av data grafisk(api)
A 63 Funksjonsklasse
A631 NMEA(protokoll)
A632 EM3000(protokoll)
A633 Seriell kommunikasjon

Virkelig gjennomførte aktiviteter i denne perioden

A51 Funksjonsbeskrivelse
A52 Grafikk
A53 Funksjonsklasse
A531 Oppslag i tanktabell
A532 Skalering av signal

A 61 funksjonsbeskrivelse
A 62 Grafikk
A621 Sjekk ut om 3D grafikk av båt er mogleg
A622 Framstilling av data grafisk(api)
A 63 Funksjonsklassa
A631 NMEA(protokoll)
A64 Grafisk klasse

Beskrivelse av/begrunnelse for eventuelle avvik mellom planlagte og virkelige aktiviteter

Pga sjukdom blei ikkje alle planlagte aktivitetar for tankpeiling-simulator utført. Dette vil ikkje skape forsinkelsar for framdrifta totalt sett.

Det blei brukt noko meir tid en planlagt på utforsking av 3D-simulering av MRU simulator. Dette medførte til noko forskyving av aktivitetar vidare. Dette vil ikkje skapa noko forsinkelsar totalt

1) Noter her kort tilbakemelding om antall møter – fordelt på typer (interne, styringsgruppe, møte med veileder) - i denne rapportperioden

sett.

Beskrivelse av /begrunnelse for endringer som nå ønskes i selve prosjektets innhold eller i den videre framgangsmåten - eller framdriftsplanen

-

Hovederfaring fra denne perioden

Denne perioden har en oppnådd erfaringar innan tankpeilingssystem. Vi har fått opplæring frå både skipsdesigneselskapet Ulstein Design, samt Ulstein Power & Control på dette temaet. Vidare har en oppnådd erfaringar innan 3D-simulering ved bruk av .3ds filformat og Java. Også erfaringar og forståelse av NMEA kommunikasjons protokoll.

Hovedhensikt/fokus neste periode

Neste periode skal man ha fokus på ferdigstilling av MRU og tankpeiling -simulator. Dette vil gjere simulatoren komplett. Vidare vert fokuset på dokumentasjon og rapport.

Planlagte aktiviteter neste periode

Rolf Ottar Rovde:
A53 Funksjonsklasse
A531 Oppslag i tanktabell
A532 Skalering av signal
A533 Fysikk: Tettheit (autotettheit IAS)
A54 Grafisk klasse
A55 Test av funksjonalitet
A56 Rapport

Kristian Østgaard:
A632 EM3000(protokoll)
A633 Seriell kommunikasjon
A 65 Test av funksjonalitet
A 66 Rapport

Annet

-

1) Noter her kort tilbakemelding om antall møter – fordelt på typer (interne, styringsgruppe, møte med veileder) - i denne rapportperioden

ID301702 Hovedprosjekt	Prosjekt IAS Simulator	Antall møter denne periode 1).	Firma - Oppdragsgiver Høgskolen i Ålesund /	Side 3 av 3
Rapport fra prosess Framdriftsrapport	Periode/uke(r) 10-12	Antall timer denne per. (fra logg)	Prosjektgruppe (navn)	Dato 20.03.15

Ønske om /behov for veiledning, tema i undervisningen – drøfting ellers -	
Godkjenning/signatur gruppeleder Rolf Ottar Rovde	Signatur øvrige gruppedeltakere Kristian Østgaard

1) Noter her kort tilbakemelding om antall møter – fordelt på typer (interne, styringsgruppe, møte med veileder) - i denne rapportperioden

ID301702 Hovedprosjekt	Prosjekt	Antall møter denne periode 1).	Firma - Oppdragsgiver Høgskolen i Ålesund /	Side 1 av 2
Rapport fra prosess Framdriftsrapport	Periode/uke(r) 13-15	Antall timer denne per. (fra logg)	Prosjektgruppe (navn)	Dato 10.04.15

Hovedhensikt / fokus for arbeidet i denne perioden

Også denne perioden har vi hatt fokus på tankpeiling- og MRU-simulator.

Planlagte aktiviteter i denne perioden

Rolf Ottar Rovde:
A53 Funksjonsklasse
A531 Oppslag i tanktabell
A532 Skalering av signal
A533 Fysikk: Tettheit (autotettheit IAS)
A54 Grafisk klasse
A55 Test av funksjonalitet
A56 Rapport

Kristian Østgaard:
A632 EM3000(protokoll)
A633 Seriell kommunikasjon
A 65 Test av funksjonalitet
A 66 Rapport

Virkelig gjennomførte aktiviteter i denne perioden

Rolf Ottar Rovde:
A53 Funksjonsklasse
A531 Oppslag i tanktabell
A532 Skalering av signal
A533 Fysikk: Tettheit (autotettheit IAS)
A54 Grafisk klasse

Kristian Østgaard:
A632 EM3000(protokoll)
A633 Seriell kommunikasjon

Beskrivelse av/begrunnelse for eventuelle avvik mellom planlagte og virkelige aktiviteter

A5 – Tankpeiling
På siste møte vart det lagt fram at vi skulle bruke interpolering og ekstrapolering for å finne verdiar mellom oppgitte verdiar i tanktabell. Dette førte til meir arbeid enn antat pga feilhandtering og testing. Det har gjort at A55 og A56 ikkje er fullført.

A6 – MRU-simulator
På grunn av manglande utstyr er ikkje A65 fullstendig utført. Som det kjem fram i gantt-diagram har vi sett dette punktet til 80% fullført. MRU-simulatoren er fullført og verkar som den skal grafisk. Det som står att er testing av seriell kommunikasjon. Grunna dette er helder ikkje A66 fullført.

Beskrivelse av /begrunnelse for endringer som nå ønskes i selve prosjektets innhold eller i den videre framgangsmåten - eller framdriftsplanen

1) Noter her kort tilbakemelding om antall møter – fordelt på typer (interne, styringsgruppe, møte med veileder) - i denne rapportperioden

ID301702 Hovedprosjekt	Prosjekt	Antall møter denne periode 1).	Firma - Oppdragsgiver Høgskolen i Ålesund /	Side 2 av 2
Rapport fra prosess Framdriftsrapport	Periode/uke(r) 13-15	Antall timer denne per. (fra logg)	Prosjektgruppe (navn)	Dato 10.04.15

Hovederfaring fra denne perioden

Gjennom denne perioden har en fått enda meir forståing av funksjonaliteten til tankpeiling og korleis signala frå tank blir handtert i IAS. Lært å bruke interpolering og ekstrapolering i praksis. Sett på fysikk i praksis i utrekning av tettheit og mengde veske i tankane. Erfaring og forståing av Simrad EM3000 kommunikasjonsprotokoll og seriell kommunikasjon.

Hovedhensikt/fokus neste periode

Neste periode blir fokuset testing og endeleg ferdigstille MRU- og tank-simulator. Dokumentasjon blir også ein del av hovudfokuset.

Planlagte aktiviteter neste periode

Rolf Ottar:
A56 Rapport

Kristian:
A 65 Test av funksjonalitet
A 66 Rapport

Begge:
A71 Brukarvegleiing
A74 Gjennomgang av java klasser - Javadoc

Annet

Ønske om /behov for veiledning, tema i undervisningen – drøfting ellers

Godkjenning/signatur gruppeleder	Signatur øvrige gruppedeltakere
Kristian Østgaard	Rolf Ottar Rovde

1) Noter her kort tilbakemelding om antall møter – fordelt på typer (interne, styringsgruppe, møte med veileder) - i denne rapportperioden

ID301702 Hovedprosjekt	Prosjekt	Antall møter denne periode 1).	Firma - Oppdragsgiver Høgskolen i Ålesund /	Side 1 av 2
Rapport fra prosess Framdriftsrapport	Periode/uke(r) 16-19	Antall timer denne per. (fra logg)	Prosjektgruppe (navn)	Dato 10.04.15

Hovedhensikt / fokus for arbeidet i denne perioden

Også denne perioden har vi hatt fokus på rapportskriving

Planlagte aktiviteter i denne perioden

Rolf Ottar:
A56 Rapport

Kristian:
A 65 Test av funksjonalitet
A 66 Rapport

Begge:
A71 Brukarvegleiing
A74 Gjennomgang av java klasser - Javadoc

Virkelig gjennomførte aktiviteter i denne perioden

Rolf Ottar Rovde:
A 72: Rapport påbegynt

Kristian Østgaard:
A 65: Test av funksjonalitet
A 72: Rapport påbegynt

Begge:
A74: Gjennomgang av java klasser - Javadoc

Beskrivelse av/begrunnelse for eventuelle avvik mellom planlagte og virkelige aktiviteter

A5 – Tankpeiling
Dette punktet er ikkje fått testa opp mot IAS. Dette vil vi prøve å få til veke 19.

Beskrivelse av /begrunnelse for endringer som nå ønskes i selve prosjektets innhold eller i den videre framgangsmåten - eller framdriftsplanen

-

Hovederfaring fra denne perioden

Vi har fått testet MRU på ein verkeleg FAT hjå Ulstein Power & Control. Dette resulterte i små "bug fix" i simulatoren, men samstundes fann ein feil på Ulstein IAS og fekk retta opp i desse under vegs. Fekk god oppfylging av Ulstein under FAT test.

Hovedhensikt/fokus neste periode

Neste periode blir fokus på rapportskriving og dokumentasjon som manglar.

Planlagte aktiviteter neste periode

1) Noter her kort tilbakemelding om antall møter – fordelt på typer (interne, styringsgruppe, møte med veileder) - i denne rapportperioden

ID301702 Hovedprosjekt	Prosjekt	Antall møter denne periode 1).	Firma - Oppdragsgiver Høgskolen i Ålesund /	Side 2 av 2
Rapport fra prosess Framdriftsrapport	Periode/uke(r) 16-19	Antall timer denne per. (fra logg)	Prosjektgruppe (navn)	Dato 10.04.15

Annet	
Ønske om /behov for veiledning, tema i undervisningen – drøfting ellers	
Godkjenning/signatur gruppeleder Kristian Østgaard	Signatur øvrige gruppedeltakere Rolf Ottar Rovde

1) Noter her kort tilbakemelding om antall møter – fordelt på typer (interne, styringsgruppe, møte med veileder) - i denne rapportperioden

Vedlegg 3

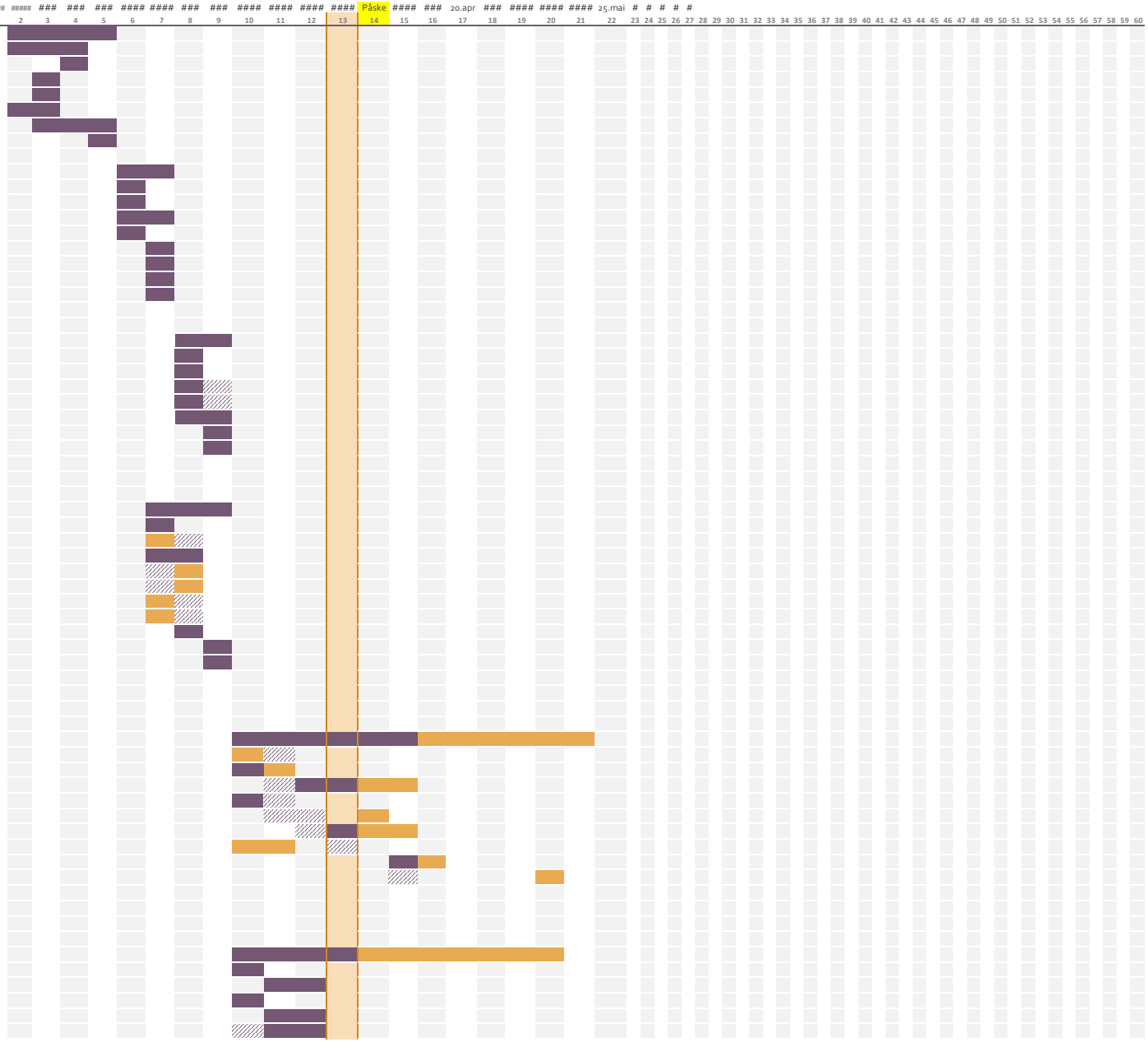
Framdriftsplan

IAS SIMULATOR

Period Highlight: 13

Plan | Actual | % Complete | Actual (beyond plan) | % Complete (beyond plan)

RESPONSIBLE	ACTIVITY NR.	ACTIVITY	PLAN START	PLAN DURATION	ACTUAL START	ACTUAL DURATION	PERCENT COMPLETE
ROR/KØS	**A1**	**Forprosjekt - KØS/ROR**	2	4	2	4	100 %
ROR/KØS	A11	Informasjonsinnsamling	2	3	2	3	100 %
ROR/KØS	A12	Framsakaffe informasjon om krav til innhald	4	1	4	1	100 %
ROR/KØS	A13	Fordele oppgåver	3	1	3	1	100 %
ROR/KØS	A14	Møte med veiledere og bedrift	3	1	3	1	100 %
ROR/KØS	A15	Møte med bedrift - brainstorming	2	2	2	2	100 %
ROR/KØS	A16	Forprosjektrapport	3	3	3	3	100 %
ROR/KØS	A17	Korrektur	5	1	5	1	100 %
KØS	**A2**	**Konfigurasjon kontroller(PLS) - KØS/ROR**	6	2	6	2	100 %
KØS	A21	funksjonsbeskrivelse	6	1	6	1	100 %
ROR	A22	Oppsett hardware	6	1	6	1	100 %
KØS	A23	Konfigurere oppsett	6	2	6	2	100 %
ROR	A24	Modbus kommunikasjon	6	1	6	1	100 %
KØS	A25	IO konfigurering	7	1	7	1	100 %
ROR	A26	Modbus kommunikasjon med Java	7	1	7	1	100 %
ROR	A27	Test av funksjonalitet	7	1	7	1	100 %
ROR	A28	Rapport	7	1	7	1	100 %
	A29	tom					
ROR	A3	**GUI base - ROR**	8	2	8	2	100 %
ROR	A31	Funksjonsbeskrivelse	8	1	8	1	100 %
ROR	A32	Grafikk	8	1	8	1	100 %
ROR	A33	Hovudbilete	8	2	8	1	100 %
ROR	A34	Standard fane	8	2	8	1	100 %
ROR	A35	Fane for digitale/analoge signal for generel test	8	2	8	2	100 %
ROR	A36	Test av funksjonalitet	9	1	9	1	100 %
ROR	A37	Rapport	9	1	9	1	100 %
	A38	tom					
	A39	tom					
KØS	A4	**Pumpe-simulator - KØS**	7	3	7	3	100 %
KØS	A41	Funksjonsbeskrivelse	7	1	7	1	100 %
KØS	A42	Grafikk	8	1	7	1	100 %
KØS	A43	Funksjonsklasse	7	2	7	2	100 %
KØS	A431	Start	7	1	8	1	100 %
KØS	A432	Stop	7	1	8	1	100 %
KØS	A433	Running	8	1	7	1	100 %
KØS	A434	Remote	8	1	7	1	100 %
KØS	A44	Grafisk klasse	8	1	8	1	100 %
KØS	A45	Test av funksjonalitet	9	1	9	1	100 %
KØS	A46	Rapport	9	1	9	1	100 %
	A47	tom					
	A48	tom					
	A49	tom					
	A5	**Tankpeiling-simulator**	10	6	10	12	100 %
	A51	Funksjonsbeskrivelse	11	1	10	1	100 %
	A52	Grafikk	10	1	10	2	100 %
	A53	Funksjonsklasse	11	3	12	4	100 %
	A531	Oppslag i tanktabell	10	2	10	1	100 %
	A532	Skalering av signal	11	2	14	1	100 %
	A533	Fysikk: Tettheit (autotettheit IAS)	12	2	13	3	100 %
	A54	Grafisk klasse	13	1	10	2	100 %
	A55	Test av funksjonalitet	15	1	15	2	100 %
	A56	Rapport	15	1	20	1	100 %
	A57	tom					
	A58	tom					
	A59	tom					
	A6	**MRU-simulator**	10	4	10	11	100 %
	A 61	funksjonsbeskrivelse	10	1	10	1	100 %
	A 62	Grafikk	11	2	11	2	100 %
	A621	Sjekk ut om 3D grafikk av båt er mogleg	10	1	10	1	100 %
	A622	Framstilling av data grafisk(api)	11	2	11	2	100 %
	A 63	Funksjonsklasse	10	3	11	2	100 %

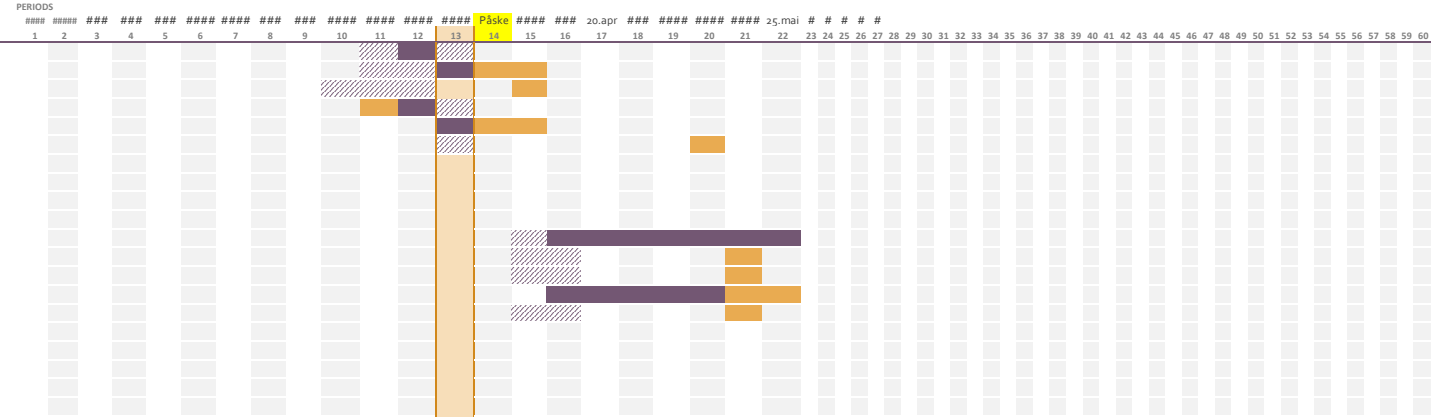


IAS SIMULATOR

Period Highlight: 13

Plan | Actual | % Complete | Actual (beyond plan) | % Complete (beyond plan)

RESPONSIBLE	ACTIVITY NR.	ACTIVITY	PLAN START	PLAN DURATION	ACTUAL START	ACTUAL DURATION	PERCENT COMPLETE
	A631	NMEA(protokoll)	11	3	12	1	100 %
	A632	EM3000(protokoll)	11	3	13	3	100 %
	A633	Seriell kommunikasjon	10	3	15	1	100 %
	A64	Grafisk klasse	12	2	11	2	100 %
	A65	Test av funksjonalitet	13	1	13	3	100 %
	A66	Rapport	13	1	20	1	100 %
	A67	tom					
	A68	tom					
	A69	tom					
	A7	**Dokumentasjon**	15	8	16	7	100 %
	A71	Brukerveiledning	15	2	21	1	100 %
	A72	Programmerings guide	15	2	21	1	100 %
	A73	Hovudrapport	16	5	16	7	100 %
	A74	Gjennomgang av java klasser - Javadoc	15	2	21	1	100 %
	A75	tom					
	A76	tom					
	A77	tom					
	A78	tom					
	A79	tom					



Vedlegg 4

Møtereferat

Referat møte 13.01.15

Stad: Høgskulen i Ålesund, rom F313

Tid: 10.15

Oppmøtte: KØS, ROR, ROV, HST, ASE

Agenda:

1. Presentasjon av prosjektgruppa og styringsgruppa

   Rask presentasjon av prosjektgruppa, Kristian og Rolf Ottar.
   Presentasjon av styringsgruppa, Rune Volden, Hans Støle og Anders Sætersmoen.

2. Presentasjon av prosjektet

   Framlegg av bachelor oppgåve. Rolf Ottar la fram prosjektgruppa sine idear og tankar rundt oppgåva. Problemstilling av prosjektet vart også framlagt. Endeleg skildring av prosjektet vil kome i forprosjektraporten.
   Rune kom med tilleggsinformasjon og forklarte bakgrunnen til oppgåva.

3. Eventuelt

   Det vart diskutert kva programmeringsplattform som skal brukast, fleire vart nemnt, men det blei fastslått att prosjektgruppa skulle ta i bruk plattformer som dei har lært å bruke tidlegare i utdanninga (Java, Codesys).
   Kontaktpersonar har Ulstein fleire av som kan vere aktuelle, prosjektgruppa tek derfor kontakt med dei sjølve. Hardware vil også Ulstein kunne stille med det vi treng.
   Prosjektgruppa vil til neste møte jobbe med forprosjektraport og nedbryting av prosjektet.
   Det vart bestemt at vi skal frå og med neste møte ha faste møter med styringsgruppa annankvar veke. Desse møta vil bli helt tysdagar kl. 10.15. Vegleiar frå Ulstein møter om han har tid.

Neste møte: Tysdag 03.02.15
Møte avslutta: 10.45

Høgskulen i Ålesund 13.01.15
Kristian Østgaard, Prosjektleiar
Rolf Ottar Rovde, Sekretær

Referat møte 14.01.15

Sted: Ulstein Power & Control, Fremmerholen.

Tid: 1030-1130

Oppmøtte: Kristian Østgaard, Rolf Ottar Rovde, Kai Verner Røsvik, Helge Sverre Røsvik, Arne Johan Helle.

Møtte Ikkje: -

Agenda:

1. Uformell diskusjon rundt kva som er viktig å simulere på ein FAT for Ulstein IAS.

Referat:

- Forslag frå Kai Verner var å simulere pumper og tilhøyrande autofunksjon på eit sett med 2 pumper.
- Forslag frå Helge Sverre og Arne Johan var å simulere tanksystem der man tek føre seg simulering av trykksensorar saman med MRU for å teste konfigurasjon(tanktabellar) i IAS.
- Forslag frå Helge Sverre om å simulere PMS, power grid.
- Tok ein gjennomgang av ein IAS som var rigga på labben, saman med Kai Verner.

Referat møte 15.01.15

Sted Ulsteinvik, UPC

Tid: 08.30

Oppmøtte: Kristian Østgaard, Rolf Ottar Rovde og Jogeir Emil Gjersdal

Agenda:

1. Uformell diskusjon kring programmering og kommunikasjon.

Referat:

- La fram oppgåva for Jogeir
- Jogeir var einig i tankegangen kring programmeringsplattform og hardware.
- Når det gjeld kva kommunikasjonsprotokoll som skal takast i bruk nemnde Jogeir at vi kunne lage vår eigen protokoll ved hjelp av UDP datagram. Dette fordi Jogeir har brukt modbusbibliotek i fleire prosjekt og funne desse uferdige.
  Etter litt diskusjon kom vi fram til at i vårt tilfelle, med dei delane vi skal simulere vil vi ikkje kome bort i desse problema med Modbus. Med tanke på vidare utvikling av simulator der kanskje andre skal inn å bygge vidare på det som blir laga i dette prosjektet, så vil også bruk av modbuss gjere programmet lettare å sette seg inn i og forstå for andre.
  Å lage ein protokoll frå start ville også blitt ein tidkrevjande del i prosessen.
- Jogeir viste oss også eit par verktøy som han har laga til utvikling av IAS.

Møte avslutta: 10.00

Ulstein Power & Controll, Ulsteinvik 15.01.15
Kristian Østgaard, Prosjektleiar
Rolf Ottar Rovde, Sekretær

MØTE STYRINGSGRUPPA IAS SIMULATOR

Tid: 02.02.15 kl. 1015.

Stad: HIALS, rom 425

Frammøtte: Kristian Østgaard, Rolf Ottar Rovde, Hans Støle.

Fråverande: Rune Volden, Anders Sætersmoen.

Agenda:

1. Gjennomgang av framdriftsplan.
2. Gjennomgang av forprosjektrapport.
3. Gjennomgang av gantt-diagram.
4. Eventuelt

Referat:

1. Framdriftsplanen (Gantt-diagram) må brytast ned i fleire delaktivitetar. Desse må samsvare med planlagde aktivitetar som er vist i framdriftsrapport.
   I framdriftsplanen må det kome fram kven som har ansvaret for kva aktivitet.
2. Legge til ein generell beskriving av tanksystemet i forprosjektrapport.
   Lag ein oversikt over framtidig utvikling av systemet og legg dette til i forprosjektrapport.
3. Gantt-diagram vert fortløpande oppdatert og sendt ut til styringsgruppa i forkant av kvart møte.

Idémyldring IAS pumpe simulering

Tid: 16.02.15
Stad: Ulstein Power & Control, Ulsteinvik, vrimleareal ingeniør avdeling.
Frammøtte: Rolf Ottar Rovde, Stian Lid Kleppe, Arne Johan Helle.
Agenda: Ingen fast agenda.

Referat:
Under planlegging av prosjektet (forprosjektrapport) var det beskreve ein pumpe-simulator som skulle simulere ei pumpe med start/stop funksjon, opplyste Rolf Ottar. Arne Johan kom med innspel at der var fleire typar pumper som var aktuelle for simulering. Stian Lid kleppe tok eit oppslag i dokumentasjon og ut frå dette var der tre gjengangarar av pumper på Ulstein IAS. Dette var:

1. Start/stop
2. Start lav/start høg/stop
3. Start/stop/frekvensstyring

MØTE STYRINGSGRUPPA IAS SIMULATOR

Tid: 17.02.15 kl. 1015.

Stad: HIALS, rom B434

Frammøtte: Kristian Østgaard, Rolf Ottar Rovde, Hans Støle.

Fråverande: Rune Volden, Anders Sætersmoen.

Agenda:

1. Gjennomgang av framdriftsrapport
2. Gjennomgang av gantt-diagram
3. Val av språk i rapport og dokumentasjon
4. Eventuelt

Referat:

1. Prosjektgruppa har kome fram til måla som var sett for denne perioden. Som ynskt frå førre møte er måla no delt opp i underaktivitetar og kven som har ansvaret for oppgåvene.
2. Få fram også i gantt-diagram kven som har ansvar for dei forskjellige oppgåvene.
3. Språk står prosjektgruppa fritt til å velje. Endeleg avgjersle vil bli tatt saman med UPC.
4. Design av GUI kan Helge Tor Kristiansen vere til hjelp med, han vil bli kontakta om vi har ledig tid mott slutten av prosjektet. Først og fremst blir fokuset no på funksjonalitet.

MØTE STYRINGSGRUPPA IAS SIMULATOR

Tid: 03.03.15 kl. 1015.

Stad: HIALS, rom B434

Frammøtte: Kristian Østgaard, Rolf Ottar Rovde, Hans Støle, Anders Sætersmoen.

Fråverande: Rune Volden.

Agenda:

1. Gjennomgang av framdriftsrapport
2. Framvising av aktivitetar denne perioden
3. Eventuelt

Referat:

1. Prosjektgruppa har nådd måla for denne perioden. I framdriftsrapporten ligg også konkrete mål får den neste perioden i form av aktivitetar og underaktivitetar.
2. Prosjektgruppa har sett opp ei din-skinne med straumforskyning, PLS og tilhøyrande I/Oar. Det er laga ein GUI i Java med fane for pumpe og generelle signal. Modbuskommunikasjon frå Java til PLS der handtering I/O adresser skjer i Codesys. Det vart vist at kommunikasjon virka ved bruk av fana for generelle signal, her blir digitale og analoge I/Oar sett og lest. Virkemåte til fana for pumpe blei også gjennomgått.
3. Grunna påske blir neste møte haldt 24.03.15, innkalling kjem når dato nærmar seg.

MØTE STYRINGSGRUPPA IAS SIMULATOR

Tid: 25.03.15 kl. 1015.
Stad: HIALS, Fagskulen
Frammøtte: Kristian Østgaard, Rolf Ottar Rovde, Hans Støle, Anders Sætersmoen og Rune Volden.

Agenda:

1. Gjennomgang av fradriftsrapport.
2. Framvisning av aktivitetar denne perioden.
3. Eventuelt.

Referat:

1. Framdriftsraport vart lagt fram. Prosjektgruppa har iløpet av perioda kome i mål med dei fleste aktivitetane som var planlagt. Dei få aktivitetane som ikkje vart ferdig vil ikkje skape forsenking totalt sett og vil igjen bli sett opp som mål for neste periode.
   Hans la fram at prosjektgruppa opptrer ryddig og systematisk.
   Rolf informerte om bruk av interpolering og ekstrapolering som skal brukast til å finne verdiar mellom oppgitte verdiar i tanktabellar når det oppstår. Dette kom fram under samtale på mail med Kai Verner hjå UPC 23.03 og vil dermed føre til litt meir arbeid på tank-simulator.
2. Prosjektgruppa viste fram simulatoren slik den ser ut per i dag. Tankt- og MRU-simulator vart trekt fram sidan desse har våre hovudaktivitetar siste perioda.
3. Ønske frå Rune var å gjere bruken av simulatoren så enkel som mogleg når det gjeld oppkopling i forhold til mengda som blir testa. Eit forslag var for eksempel til fana Generelle signal, der vi no testar ein I/O om gangen. Her kunne ein sett opp for testing av fleire I/Oar i staden og ha ei leidning med ferdigkopla koplingsstykker i begge endar. På denne måten vil oppkobling bli effektivt og ein får også testa meir.
   Rune la også vekt på at det funksjonelle er viktigare enn det grafiske og at MRU-simulatoren var meir enn god nok slik som den er no i 2D.

MØTE STYRINGSGRUPPA IAS SIMULATOR

Tid: 14.04.15 kl. 1015.
Stad: HIALS, F420
Frammøtte: Kristian Østgaard, Rolf Ottar Rovde, Hans Støle og Anders Sætersmoen
Agenda:

1. Gjennomgang av fradriftsrapport.
2. Framvisning av aktivitetar denne perioden.
3. Eventuelt.

Referat:

1. Systematisk gjennomgang av aktivitetane frå denne perioden. Har kome i mål med mesteparten av måla. Det som står att er testing og ferdigstilling av tank- og MRU-simulator. Grunnen til at desse ikkje er ferdige er at interpolering mellom verdiar i tanktabell og utvikling av EM3000 protokoll har tatt lengre tid en planlagt.
   Testing av simulator er planlagt denne eller neste veke, kjem ann på når anlegget er oppe å går på lab i Ulsteinvik. Det skal vere ein FAT den 23.04.
2. Gjennomgang av korleis interpoleringa/ekstrapolering vart løyst og korleis det er implementert i simulatoren.
3. Tips til rapport:
   Når prosjektgruppa skal teste simulatoren opp mot IAS, så ta bilete/video til rapport.
   Skriv forklarande, tenk på at ein skriv til personar som ikkje kjenner til alt som står i rapporten.
   Få med at prosjektgruppa har våre med på å forme oppgåva.
   Viktig å presentere ein bra rapport, men heilheita av oppgåva tel mykje.
   Prosjektgruppa spurte om problemstilling er for generell, styringsgruppa blei einige om at med å ta med resultatmål og effektmål som beskrive i forprosjektrapporten så er den bra.

   Neste møte blir 07.05.15.

MØTE STYRINGSGRUPPA IAS SIMULATOR

Tid: 07.05.15 kl. 0915.
Stad: HIALS, F313
Frammøtte: Kristian Østgaard, Rolf Ottar Rovde, Hans Støle og Anders Sætersmoen
Agenda:

1. Gjennomgang av fradriftsrapport.
2. Gjennomgang hovudraport.
3. Gjennomgang av gantt-diagram
4. Eventuelt.

Referat:

1. Har hatt fokus på hovudrapport for å kunne legge fram eit utkast til dette møtet. Testrapport for MRU- og tankpeilingssimulator saman med brukarvegleiing utsett. Har våre på Ulstein og testa simulator opp mot IAS. MRU-simulator fungerte veldig bra. Dei andre fekk vi ikkje testa grunna mangel på tanktabell og feil ved lesing av signal inn på kontroller. Feilen er no retta opp.
2. OK oppbygging av rapport. Ein del å jobbe med. Figurtekst og referansar kjem i endeleg rapport. Unngå forteljarstil. Utlede forkortingar første gang dei blir brukt. Tankpeiling bør utledast meir. Lag eige flytdiagram for betre forståing av tankpeilingsystemet. Utrekningar tilhøyrande tankpeiling bør flyttast til teori. Fleire figurar generelt for betre og enklare forklaring. Skrivefeil må rettast opp i. Tekniske uttrykk på engelsk bør ikkje omsettast om der ikkje er konkrete norske ord.
3. Gnatt-diagram fylgjer planen.
4. Info og tips til presentasjon. 20 min presentasjon, 5 min med spørsmål etterpå. Forklar systemet i heilheit og ikkje for avansert.

Dette var siste møtet i styringsgruppa.

Vedlegg 5

Avvik

Avviksrapport

Saksnummer:	Emne:	Rapportert Dato:	Avviksnumer:
1	Oversikt over FAT datoar	12.02.2015	1

Avvik:
Oversikt over FAT datoar er ikkje framskaffa til rettleingsmøte som planlagt.

Prioritet:

Lav	x
Middels	□
Høg	□

Korrigerende Tiltak:

Få tak i FAT datoar til neste møte.

Korrigerende tiltak utført. Dato: 18.02.2015

Avviksrapport

Saksnummer:	Emne:	Rapportert Dato:	Avviksnumer:
2	Simulering av fleire pumper	16.02.2015	2

Avvik:

Etter samtale med Stian Lid Kleppe og Arne Johan Helle kom det fram at ein pumpesimulator bør ha me alle tre type pumper ein finn om bord i skip. Pumpe med ei hastigheit, to hastigheiter og frekvensstyrte.

Prioritet:

Lav	□
Middels	x
Høg	□

Korrigerende Tiltak:

Implementere alle tre type pumpe i simulator.

Utforske betre under planlegging før ein startar utvikle.

Korrigerende tiltak utført. Dato/sign: 20.02.2015

Avviksrapport

Saksnummer:	Emne:	Rapportert Dato:	Avviksnumer:
3	Planlagde mål for Tankpeiling-simulator ikkje fullført.	20.03.2015	3

Avvik:
Grunna sjukdom er ikkje planlagde mål for Tankpeiling-simulator denne perioden fullført.

Prioritet:

Lav	x
Middels	□
Høg	□

Korrigerende Tiltak:

Ta igjen det tapte til neste periode. Det vil ikkje skape forseinkingar totalt.

Korrigerende tiltak utført. Dato: 25.03.2015

Avviksrapport

Saksnummer:	Emne:	Rapportert Dato:	Avviksnumer:
4	Utforsking av 3D simulering	20.03.2015	4

Avvik:
Det blei brukt lengre tid på utforsking av 3D simulering for MRU-simulator en planlagt.

Prioritet:

Lav	x
Middels	□
Høg	□

Korrigerende Tiltak:

Jobbe vidare utan 3D simulering. Det vil ikkje skape forseinkingar totalt.

Korrigerende tiltak utført. Dato: 20.03.2015

Avviksrapport

Saksnummer:	Emne:	Rapportert Dato:	Avviksnumer:
5	Interpolering og ekstrapolering i Tankpeiling-simulator	25.03.2015	5

Avvik:
Ulstein IAS brukar interpolering og ekstrapolering for å runde av verdiar. Tankpeiling-simulator må gjere det same. Vil skape ekstra arbeid.

Prioritet:

Lav	□
Middels	x
Høg	□

Korrigerende Tiltak:

Implementere interpolering og ekstrapolering i Tankpeiling-simulator.

Burde vore utforska under planlegging.

Korrigerende tiltak utført. Dato: 20.04.2015

Avviksrapport

Saksnummer:	Emne:	Rapportert Dato:	Avviksnumer:
6	MRU-simulator ikkje testa	10.04.2015	6

Avvik:
MRU-simulator ikkje interntesta eller testa opp mot Ulstein IAS grunna manglande utstyr.

Prioritet:

Lav	x
Middels	□
Høg	□

Korrigerende Tiltak:

Vil bli ordna til neste møte.

Korrigerende tiltak utført. Dato: 16.04.2015

Avviksrapport

Saksnummer:	Emne:	Rapportert Dato:	Avviksnumer:
7	Tankpeiling-, I/O- og Pumpe-simulator ikkje testa opp mot Ulstein IAS.	18.05.2015	7

Avvik:
Det er ikkje utført testing av Tankpeiling-, I/O- og Pumpe-simulator opp mot Ulstein IAS.

Prioritet:

Lav	□
Middels	x
Høg	□

Korrigerende Tiltak:

For fullstendig kvalitetssikring av Ulstein IAS Simulator må heile simulatoren testast opp mot Ulstein IAS.

Korrigerende tiltak utført. Dato:

Vedlegg 6

Funksjonsskildringar

FUNKSJONSBESKRIVELSE – KONFIGURASJON AV KONTROLLER(PLS)

I denne delen av prosjektet skal PLS settast opp med kommunikasjon mot PC. PLS med tilhøyrande I/Oar skal vere sett opp for modbuskommunikasjon. Konfigurering av I/O skal gjerast i PLS ved hjelp av Codesys, her blir verdiane også skalert til ynskt verdiar. PLSen tek imot modbus meldingar frå PC over TCP, desse meldingane blir sett til globale variablar i PLS som er sett til modbusregister. Desse variablane vil bli skalert og kopla til variablar som tilhøyrar adressene til dei forskjellige I/Oane. Det skal settast opp 11 analoge utgangar, 10stk for trykksensorar ved tankpeiling og ein for generell testing av 4-20mA analoge signal. I tillegg skal det settast opp 9 digitale inngangar og 9 digitale utgangar, 8stk I/O for simulering av pumper og ein I/O for generell testing av digitale signal.

I JAVA skal det lagast ein applikasjon som handterer modbuskommunikasjon mellom PC og PLS. I dette programmet skal det lagast funksjonar for å opprette og lukke kontakt med PLS, i tillegg skal det kunne skrive og lese frå I/Oar. Skriving og lesing vil skje som bytes, som nemnt tidlegare vil dette bli skalert i PLS. Når det gjeld skriving til register hjå PLS vil ein variabel bli sett om gongen.

Når det gjeld testing av funksjonalitet så vil denne funksjonsbeskrivelsen vere utgangspunkt.
Når endeleg testrapport er skriven ser ein deloppgåva som fullført.

FUNKSJONSBESKRIVELSE GUIBASE - JAVA

I BOTNEN AV BRUKARGRENSESNITTET SKAL DET LIGG EIN FLEKSIBEL GUI BASE SOM FUNDAMENT TIL DELSYSTEMA. DENNE SKAL VERE VELDEFINERT OG ENKEL. I TILLEGG SKAL DET REALISERAST EIT DELSYSTEM FOR SIMULERING AV GENERELLE SIGNAL SOM ANALOG OG DIGITAL.

GRAFIKK

Grafikk skal gjenspegle Ulstein varemerket gjennom farge og logo. Lagar eit ukast på dette for vidare godkjenning av Ulstein Power & Control.

LAYOUT

I hovudbilete skal det ligge informasjon om IAS Simulator som beskriving, versjon, produsent, støtte etc. Kvart "system" skal ligge under eiga fane på hovudbilete. Ei slik fane skal vere uavhengig av andre faner å gå som eigen tråd. Ved oppretting av nytt system skal det legge seg til ny fane etter dei andre.

Fane og hovudbilete vert laga ved å bruke Jframe. Panels og frames.

FUNKSJONALITET

GUI skal ikkje innehalde teknisk funksjonalitet. Dette skal ligge under egne klasse. Dette for å ha lause koplingar og veldefinerte klasser.

FUNKSJONSBESKRIVELSE – GENERELL TEST

SIMULATOR FOR GENERELL TEST SKAL HA BASIS SIGNALA DO,AO,DI OG AI. DETTE TIL BRUK OM MAN VIL HA EIN SJEKK OPP MOT UTSTYR SOM IKKJE ER DEFINERT.

GRAFIKK

Enkel grafikf der man sette ein digital utgang på eller av ved hjelp av knapper. Eit tekstfelt som les digital inn verdi. Slider som sett analog utverdi. Progressbar som viser analog innverdi.

FUNKSJON

Lite funksjonalitet anna en skalering av analoge signal.

FUNKSJONSBESKRIVELSE – PUMPE-SIMULATOR

SKAL SIMULERE 2 PUMPER. PUMPENE SKAL SIMULERE BÅDE KORREKT OG FEIL TILBAKEMELDING AV SIGNAL TIL UIAS. SIMULATOREN SKAL IMPLEMENTERAST I GUI BASEN, HER VIL EIN KUNNE BESTEMME HANDLINGA DEI SIMULERTE PUMPENE SKAL FORETA. SJØLVE OPPERASJONEN AV SIGNALA VIL SKJE SKJULT I BAKGRUNNEN I EIGNE KLASSER.

GRAFIKK

I GUIen skal pumpene visast med symbol som simulerer kva tilstand pumpa er i. I tillegg skal kvar av pumpene ha knappar som tek seg av start-, stop-, remote-, running- og feilsignal.

FUNKSJON

Start og stop funksjonane til begge pumpene skal starte eller stoppe pumpene. Desse kan kunn settast av knappane i GUI når systemet er i lokal tilstand.

Remote/Local skal gi tilbakemelding til UIAS at pumpene er i fjernstyrings modus eller ikkje. Blir sett i GUI

Running skal gi tilbakemelding til UIAS når pumpene går.

Feilsignal skal simulere eit feilsenario der runningsignal ikkje aktiverast når pumpa skal starte

TILLEGG

Etter samtale med Stian Lid Kleppe og Arne Johan Hellen kom det fram at pumpesimulator bør ta seg av tre type pumper. Pumper med ei hastigheit, to hastigheiter og frekvensstyrt.

FUNKSJONSBESKRIVELSE – TANKPEILING SIMULATOR

ULSTEIN IAS FRAMSTILLER FARTØYETS TANKSYSTEM GRAFISK. GRAFIKKEN VISER BLANT ANNA RØR, PUMPER, FILTER OG VENTILAR MELLOM DEI ULIKE TANKANE. VÆSKENIVÅ ER VIST MED VERDIAR., MÅTEN ULSTEIN IAS MÅLAR TANKNIVÅET ER VED TRYKKSENSORAR OG OPPSLAG I TANKTABELLAR. MÅLER TRYKSØYLA TIL VÆSKA OG FINN HØGDA VED HJELP AV FORMELEN $\rho = P * h * g$ *. MASSETETTEHEITA VERT AUTOMATISK UTREKNA VED Å MÅLE TRYKKFORSKJELLEN MELLOM TO GITTE PUNKT. TYPISK NIVÅDIFFERANSE PÅ* 1000*MM I HØGDA PÅ DESSE SENSORANE. MASSETETTHEITA KAN OGSÅ LEGGJAST INN MANUELT. TABELLEN TEK OG HØGDE FOR TRIM OG HEEL.*

ULSTEIN IAS SIMULATOR SKAL SIMULERE VÆSKENIVÅET I TANKANE VED Å TA OPPSLAG I TABELL, SKALERING AV ANALOGSIGNAL OG SENDE DESSE DATAENE TIL ULSTEIN IAS. SIMULERINGA SKAL OG TA HØGDE FOR TRIM OG HEEL

GRAFIKK

Tanken skal grafisk framstillast likt som IAS der verdiar står inni ein tank figur.

FUNKSJON

1. Simulere trykkgivar for atmosfærisk trykk(referanse for alle målingar).
2. Simulering av tank sensor 1: Simulering av 4-20mA som gir høgda til væskesøyla.
3. Simulering av tank sensor 2: Simulering av 4-20mA som gir høgda til væskesøyla.
4. Differansen mellom sensor 1 og 2 skal gi tettheita til stoffet på tanken.
5. Oppslag i tanktabell utfrå innhald i volum i tanken og trim, heel frå MRU.
6. opplasting av tanktabell via fil-velger.

FUNKSJONSBESKRIVELSE – MRU-SIMULATOR

MRU-SIMULATOREN SKAL SIMULERE RØRSLE TIL EIT SKIP PÅ SJØEN. RØRSLENE SOM SKAL SIMULERAST ER HEEL OG TRIM. DESSE RØRSLENE VIL BLI GENERERT AV SINGUSKURVER. DENNE DATAEN SKAL SAMLAST I EIN STANDARDISERT SERIELL PROTOKOLL TYPE NMEA OG EIN TYPE EM3000 FØR DEN SENDAST TIL IAS VIA EIN ETHERNET-SERIELL KONVERTER. SOM DEI ANDRE FANENE VIL FUNKSJONALITET OG GUI BLI DELT I TO KLASSER.

GRAFIKK

Grafisk skal GUIen vise skipets rørsle i sjøen ved hjelp av grafar. Ein graf vil vise skipet frå sida opp mot trim-rørsle, ein annan vil vise skipet framan ifrå opp mot heel-rørsle. Framstillinga i grafen vil være trim/heel på y-akse og tid på x-akse, sinuskurva vil bli teikna midt på grafen til ei kvar tid og tidligare verdiar vil bevege seg mot venstre i grafen. I tilleg vil GUIen innehalde to knappar, ein for autogenerering av trim-rørsle og ein for autogenerering av heel-rørsle.
Når det gjeld grafisk framstilling av rørsle til skipet vil det bli forska på moglegheitene for å kunne vise dette med ein 3D modell av eit skip.

FUNKSJON

Funksjonsklassa skal handtere mattematikken bak autogenerering av rørslene som skal simulerast. Den skal samle data som skal sendast over seriell linje i ei eiga metode. Denne metoden vil bli brukt av ei eiga klasse som sender data som NMEA eller EM3000 ut til IAS. MRU-simulatoren skal også sende verdiane for trim og heel til tank-simulator til ei kvar tid når ein av dei eller begge er generert.

Vedlegg 7

Testrapportar

TESTRAPPORT

Rapporten er skrevet av: Rolf Ottar Rovde	**Dato**: 12.01.15
Aktivitetsnr: A2-A28	**Namn**: Konfigurasjon kontroller (PLS)

Testpunkt:

1. Kommunikasjon mellom Java applikasjon og PLS.
2. Kommunisere med digitale utgongar.
3. Kommunisere med digitale inngongar.
4. Kommunisere med analoge utgongar.
5. Skalering av analoge utgongar

Resultat:

1. Det vart brukt Jamod som er ein modbus implementasjon til java. Java applikasjonen(server) vert sett som master. Dette vil sej at den teke seg av alle førespurnadar på verdiar og skriving av verdiar mot slaven som er PLS. Ei rekke globale modbus variablar vart instansiert i PLS for kommunikasjon opp mot java(server). Både lesing å skriving av modbus register fungerte fint. Ut frå manual til Wago PLS var det dokumentert at PLS kan takle opp mot 10 trådar(registerkall) samtidig. Dette er ikkje testa.
2. Test av utgongar var gjort ved å sende ein single registerkall på gitt modbusadresse i PLS. Knytingar i ein lite PLS program vart laga slik at modbus variabel vart knytta opp mot ein utgong.
3. Som punkt 2.
4. Som punkt 2.
5. Den analoge utgongen er ein 16 bit+-. Eit kal på 4-20mA vart då ein integer mellom 0 og 32767. Brukte ein slider i ein enkel java GUI for testing av varierande analog signal.

TESTRAPPORT

Rapporten er skrevet av: Rolf Ottar Rovde	**Dato**: 16.02.15
Aktivitetsnr: A3 - A37	

Testpunkt:

1. Opprette ramme med ulike dimensjoner.
2. Opprette ramme med 1stk JPanel fane.
3. Opprette ramme med 10 JPanel faner.

Resultat:

1. Dette er testet og fungerer.
2. Dette fungerer. Objektet som skal tas inni GUIBase må arve fra JPanel.
3. Ved å legge til fleire faner legg dei seg i samme rekkefølgje som dei er sett i programmet. Fungerer fint. Vert der for mange faner på ei linje utvidare den seg til fleire linjer.

TESTRAPPORT

Rapporten er skrevet av: Rolf Ottar Rovde	**Dato**: 27.02.15
Aktivitetsnr: A35	

Testpunkt:

1. Sett av digitale utgang
2. Lese av digital inngang
3. Skrive analog utgang
4. Lese analog utgang

Resultat:

1. Når ein trykte ON blei utgangen sett høg. Når ein trykte *OFF* blei utgangen sett lav.
2. Når ein sette inngong høg, viste tekstfels true. Når en sette inngong lav viste tekstfelt false.
3. Kopla eit multimeter på den analoge utgongen for måling av mA. Sette skyveknapp i GUI for IO simulator heilt til høgre. Målte 3.99mA. Sette skyveknapp heil til venstre. Målte 19.98mA.
4. For test av analog inngang vart analoge utgong kopla på analog inngang. Ved endring av skyveknapp for analog utgong verifiserte en den analoge inngangen fungerte. Progressbar for analog inngang viste likt som slider for analog utgang.

TESTRAPPORT

Rapporten er skrevet av: Kristian Østgaard	**Dato**: 25.02.15
Aktivitetsnr: A4-A46	**Namn**: Pumpesimulator

Testpunkt:

1. Kommunikasjon med PLS
2. Remote/Local status
3. Start/Stop frå IAS
4. Running status
5. Feilsignal på running
6. Start/Stop I Local status
7. Fleire pumper

Resultat:

1. Når ein opprettar pumpene ein skal simulere i Mainapplication i Java så sett ein modbusadressene vi skal skrive til og lese frå. Det er laga I/O liste med tilhøyrande modbussvariablar for å ha kontroll på adressene.
   I testen vart koplingane mellom modbusadressene og I/Oane i Codesys skjekka opp mot I/O lista. Det vart då kontrollert at dei rette I/Oane vart sett og at vi las frå rett I/Oar. Denne kontrollen viser at kommunikasjonen fungerer som tenkt.
2. Når GUIen startar står pumpene automatisk i remote, det er indikert i tekstfelt på raud bakgrunn. Trykker ein på Local-knappen går pumpene i Local-status, tekstfeltet skiftar då tekst og får grøn bakgrunn.
3. Start/stop signal frå IAS blei testa ved at vi laska inngangane som GUIen lyttar på. Vi kunne då starte og stoppe begge pumpene vi hadde sett opp. Dette viste også på indikasjonane i GUI, illustrasjonen av pumpa skifta farge og running-signal blei gitt når pumpa starta.
4. Running-signal blir sendt når pumpa starta, denne opperasjonen sett ein utgang i PLSen. Når pumpa stoppar blir denne utgangen låg igjen. I GUIen blir Running indikert med skildrande tekstfelt som også skiftar bakgrunnsfarge, grønt når Running, raud ved stop.
5. I GUIen har vi ein knapp for feilsignal på runningsignalet. Når ein trykker på denne knappen sett ein Running-signalet lågt medan pumpeillustrasjonen indikerer at pumpa framleis går.
6. Når pumpene er i Local-status får ein tilgang til ein startknapp og ein stoppknapp, desse knappana kan starte og stoppe pumpene lokalt. Ved bruk av desse knappane vil Running-signal framleis bli oppdatert. Går ein over i Remote-status så har ein ikkje tilgang til desse start/stopp knappane lenger.
7. Ein kan enkelt legge til fleire pumper så lenge ein har sett opp I/Oar og sett opp koplingar til modbusadresser i Codesys. I vår test har vi sett opp for to pumper. Desse fungerer begge som dei skal.

TESTRAPPORT

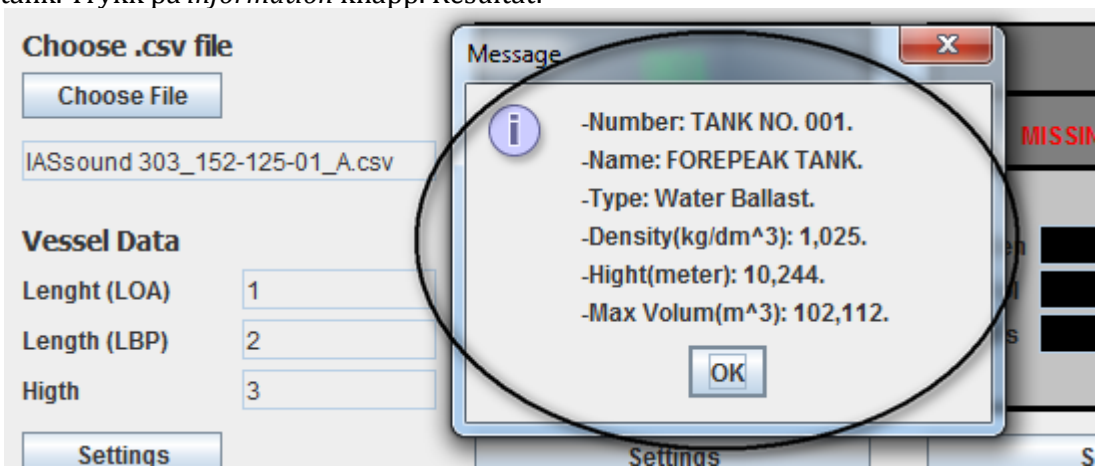
Rapporten er skrevet av: Rolf Ottar Rovde	**Dato**: 15.05.2015
Aktivitetsnr:	

Testpunkt:

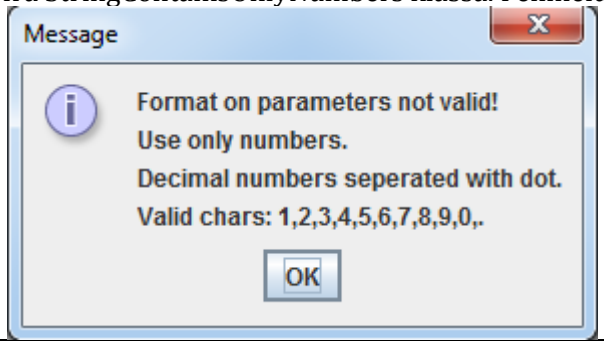
1. GUI test: knappar og tekstfelt.
2. Filvelger.
3. Tank informasjon .
4. Feilhandtering.
5. Oppslag i tanktabell.
6. Kontroll av interpolering klassa

Resultat:

1. Gjennomgang av knapper: Choose File, Settings(Vessel Data), Settings(Atmospheric Pressure), Settings(tank), Information. Sjekk av knappar for vindauge til parameter settings. Lukkeknapp i øvre del til høgre i vindauge fungerte ikkje. Ikkje lagt inn sjekk for dette i kode. Kode endra. Lukeknapp fungerer.
   Test av tekstfelt: Vessel Data: LOA, LBP, hight. Atmospheric Pressure: pressure, sensor scale. Settings Tank: Tank Number, Distance, Density, Volum, Sensor low, sensor high. Tekstfelt ok!
2. Val av fil ved bruk av filvelger ok.
3. Val av tanktabell for prosjekt UVE303 ved bruk av filvelger. Val av tank 001 i settings for tank. Trykk på *information* knapp. Resultat:

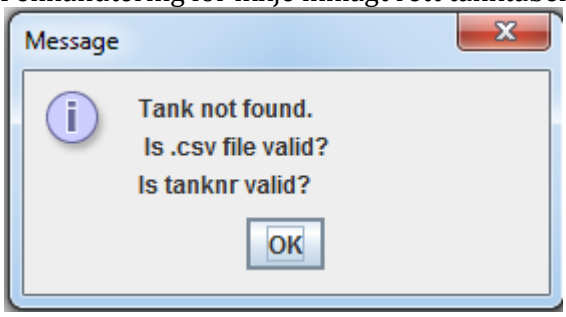


4. Feilhandtering for ikkje godkjent karakterar som parameter fungerer ok. Dette er styrt frå StringContainsOnlyNumbers klassa. Feilmelding dukkar opp og rettleiar bruker:



Feilhandtering for innlegging av tanknummer som ikkje eksisterar i tanktabell fungerer ok:

Feilhandtering for ikkje innlagt rett tanktabell. Eksempel ein JPG fil.



5. Oppslag i tanktabell vert testa ved å sette trim og roll til fast verdi og sjekke om programmet tek rett oppslag i tanktabell ut frå valt tank, volum på tank, trim og roll.
Tanktabell: prosjekt UVE 303

Parameter	Forventa verdi	Resultat
Tank 001, volum 10, trim 1, roll -2.	311,11	311.11
Tank 002, volum 30, trim -1, roll 2.	222,7	222,7
Tank 002, volum 2, trim 0, roll 0.	34,14	34,14

6. Visertil eksempel 1.3 side 8 i boka *Statistikk for Ingeniørar* av Frede Frisvold og Jan Gunnar Moe:
7. $f(3.23) \approx 4.523 + \left(\frac{4.537-4.523}{3.30-3.20}\right)(3.23-3.20) = 4.527$

Ved å kalkulere same eksempel via klassa LinearInterpolation får ein resultatet:

```
25
26          LinearInterpolation li = new LinearInterpolation();
27          System.out.println("Result LI: "+li.interpolata(3.23, 3.20, 3.30, 4.523, 4.537));
28
```

Console

```
<terminated> TestApplication1 [Java Application] C:\Program Files\Java\jre1.8.0_40\bin\javaw.exe (18. mai 2015, 11:19:54)
Result LI: 4.5272
```

TESTRAPPORT

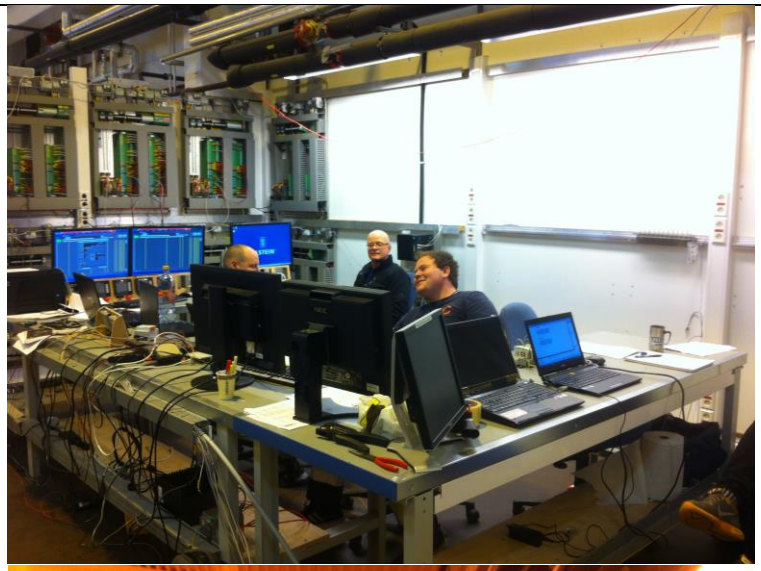
Rapporten er skrevet av: Kristian Østgaard	**Dato**: 14.05.15
Aktivitetsnr: A6	MRU-simulator

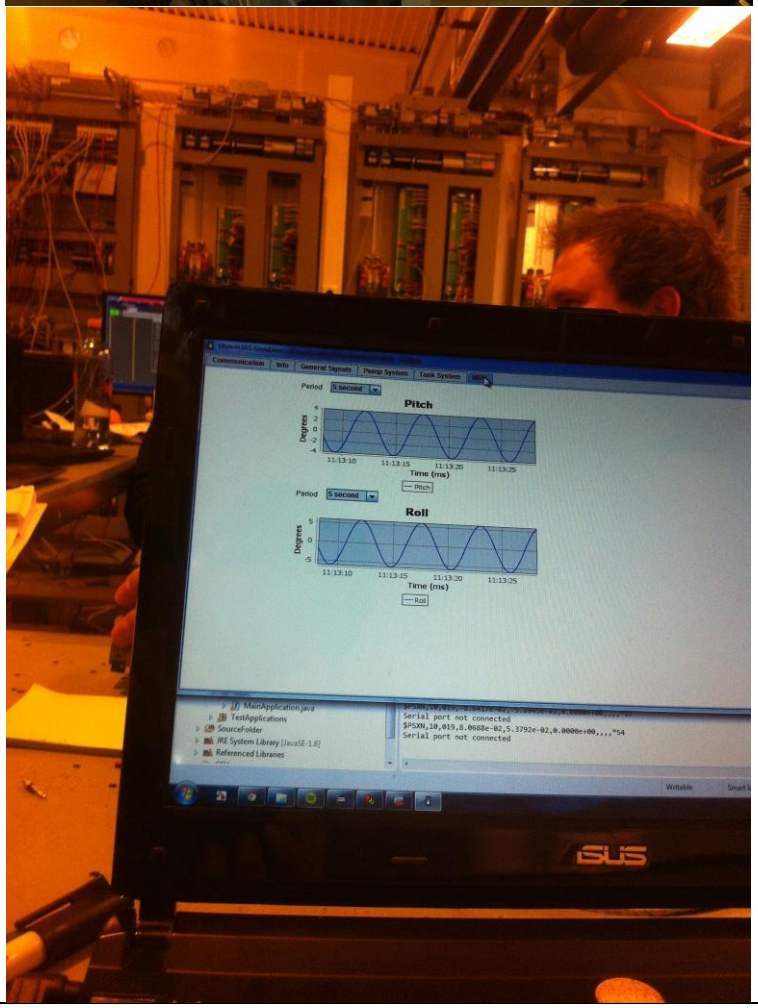
Testpunkt:

1. Sinusgenerator
2. Skifte frekvens på sinuskurve
3. Dynamisk graf
4. Seriellkommunikasjon mor Ulstein IAS
5. NMEA 0183 protokoll
6. EM3000 protokoll

Resultat:

1. Sinusgenerator generar pitch og roll kvar for seg kontinuerleg etter spesifikasjonar som er lagt inn for sinuskurvene. Property change event for sending og lesing av nye sinusverdiar fungerer som planlagt.
2. Frekvensen til sinuskurva endrar seg når ein vel ein annan verdi i rullegardinmeny i GUI.
3. Graf oppdaterer seg kva gang nye sinusverdiar kjem inn og viser verdiane frå dei 20 siste sekunda. Grafane for pitch og roll har korrekte verdiar i forhold til spesifikasjonar.
4. Tilkopling av MRU-simulator med seriell RS422 med USB til seriellkonverter. Fekk ikkje inn noko på Ulstein IAS. Seriell til ethernet-boks på IAS var sett opp feil.
5. Fekk ikkje signal inn frå MRU-simulator på NMEA 0183. Grunnen var feil i oppsett av string i MRU-simulator. Vitskapleg notasjon skulle skrivast med to desimalar, f.eks. e03 i staden for e3. Roll og pitch skulle skrivast med fire desimalar etter komma, vi skreiv med 3. Endring av dette i koden gjorde at IAS kunne lese string frå MRU-simulator. Funksjon for klokke i NMEA 0183 string er laga, men Ulstein IAS brukar ikkje denne, denne blir derfor ikkje sendt med stringen frå MRU-simulator. NMEA 0183 string frå MRU-simulator fungerer dermed som planlagt, også opp mot Ulstein IAS. Testa også system som var avhengig av signal frå MRU. Stabilisatortanksystem viste at omrekning frå radianar (som NMEA 0183 sender roll og pitch i) til grader var feil. Ved å rette opp i dette fungerte stabilisatortanksystem som det skal. Interntest av NMEA 0183 skriv ut endeleg string med sjekksum i konsoll i eclipse. Utskiven string: `$PSXN,10,019,1.9623e-02,-0.9959e-01,0.0000e+0,,,,*4d`
   Sjekksum fungerer også som den skal.
6. EM3000 er ikkje testa opp mot Ulstein IAS. Her er kunn interntest gjennomgått. På same måte som NMEA 0183 blir EM3000 bytebufferen skriven ut i konsoll. Her ser ein bytebufferen med innhal. Utskriven bytebuffer: `[-112, -112, 112, 0, -58, -3, 0, 0, 0, 0]`



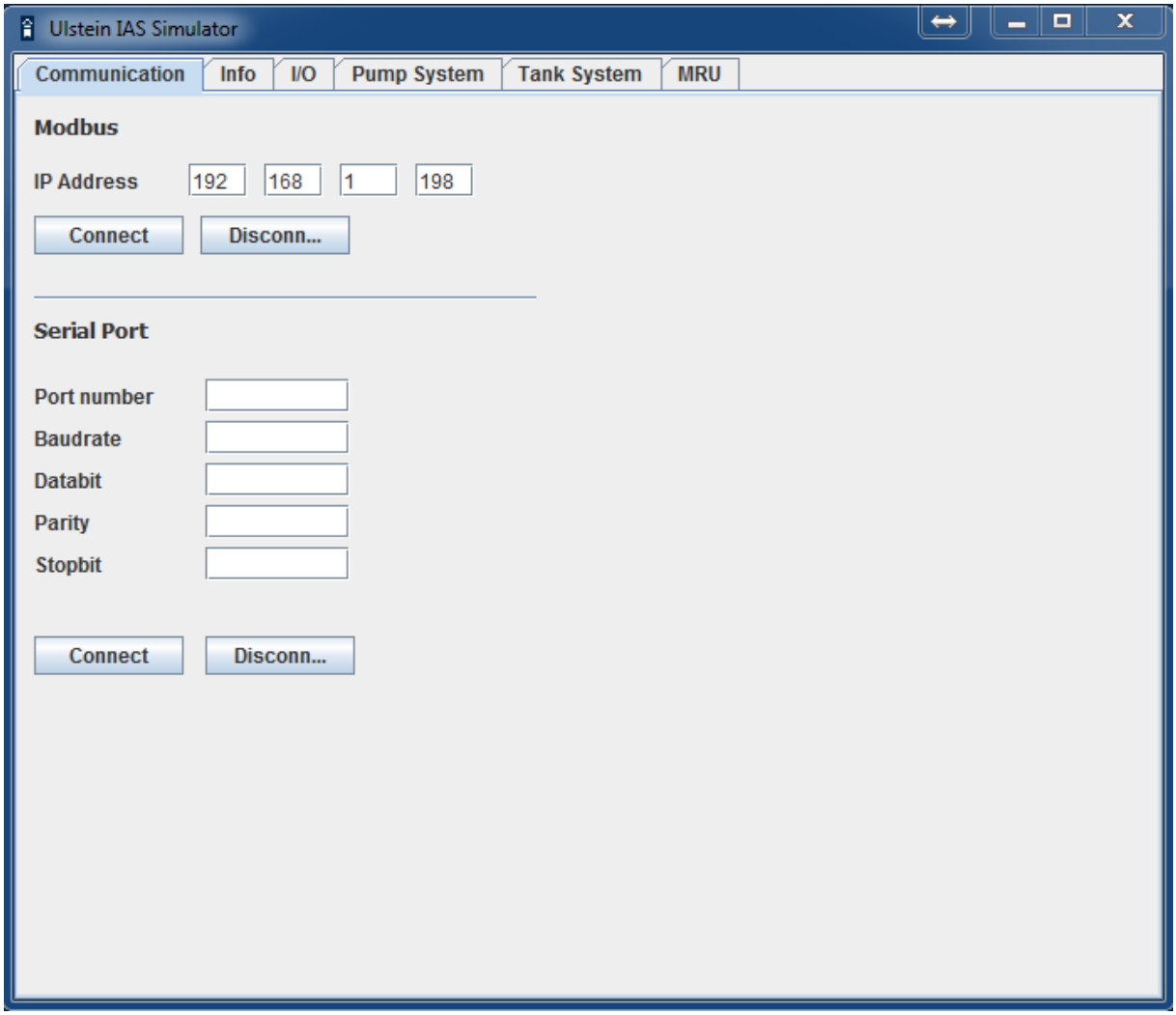

Pitch
Roll
Time (ms)
Degrees
Period
5 second
Communication
Info
General Signals
Pump System
Tank System
MainApplication.java
TestApplications
SourceFolder
JRE System Library [JavaSE-1.8]
Referenced Libraries
Serial port not connected
$PSXN,10,019,0.0688e-02,5.3792e-02,0.0000e+00,,,*54
Serial port not connected
ASUS

Vedlegg 8

Brukarmanual

ULSTEIN IAS SIMULATOR QUICK START GUIDE

KOMMUNIKASJON



Figur 1: Fane for kommunikasjon.

Fane for kommunikasjon inneheld tilkopling for Modbus TCP og Seriell kommunikasjon.

Tilkopling til kontroller(Modbus slave):

1. Verifiser den fysiske nettverksforbindelse mellom datamaskin som kjøre IAS Simulator og kontroller til simulatoren.
2. Skriv inn IP-addresse til kontroller i IPadresse-feltet som vist over.
3. Trykk *Connect* knappen for oppretting av forbindelse.

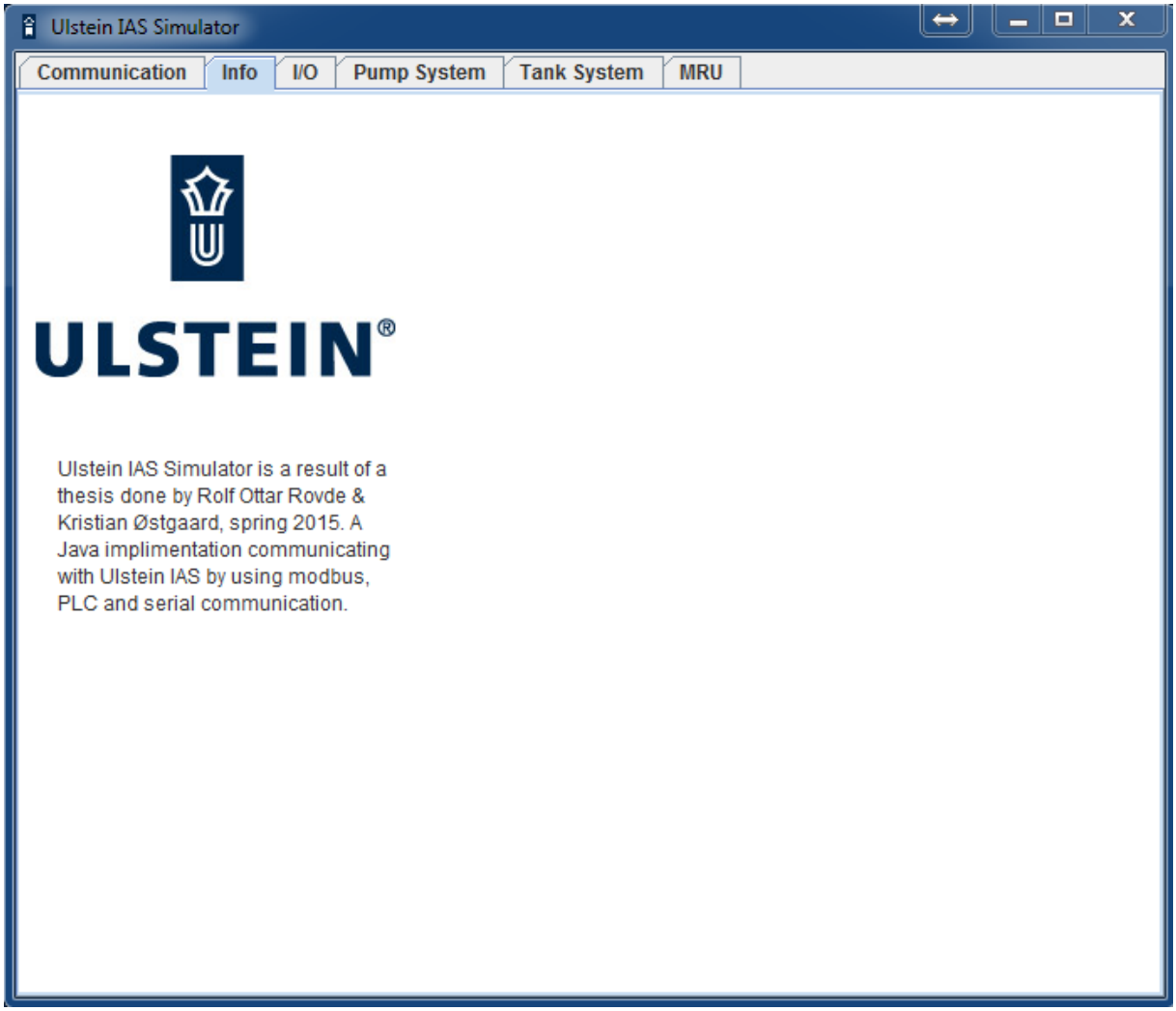
Tilkopling av seriell kommunikasjon:

1. Skriv inn port-namn til seriell porten som skal tilkoplast(eks: "COM1").
2. Skriv inn:

Baudrate	9600
Databit	8
Parity	0
Stopbit	1

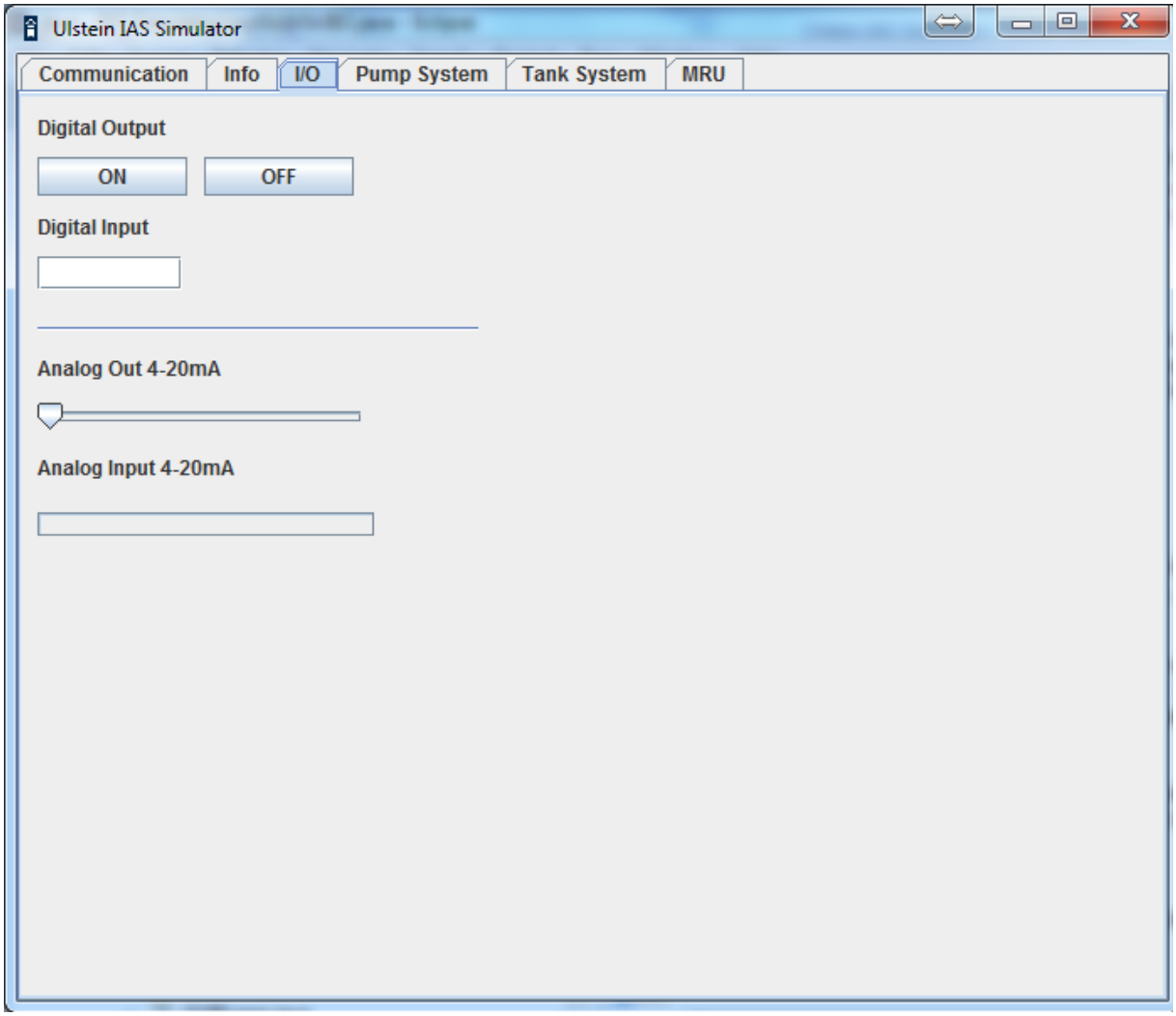
3. Trykk på knappen *connect.*

INFORMASJON



Figur 2: Fane for informasjon om IAS Simulator(about)

I/O TEST



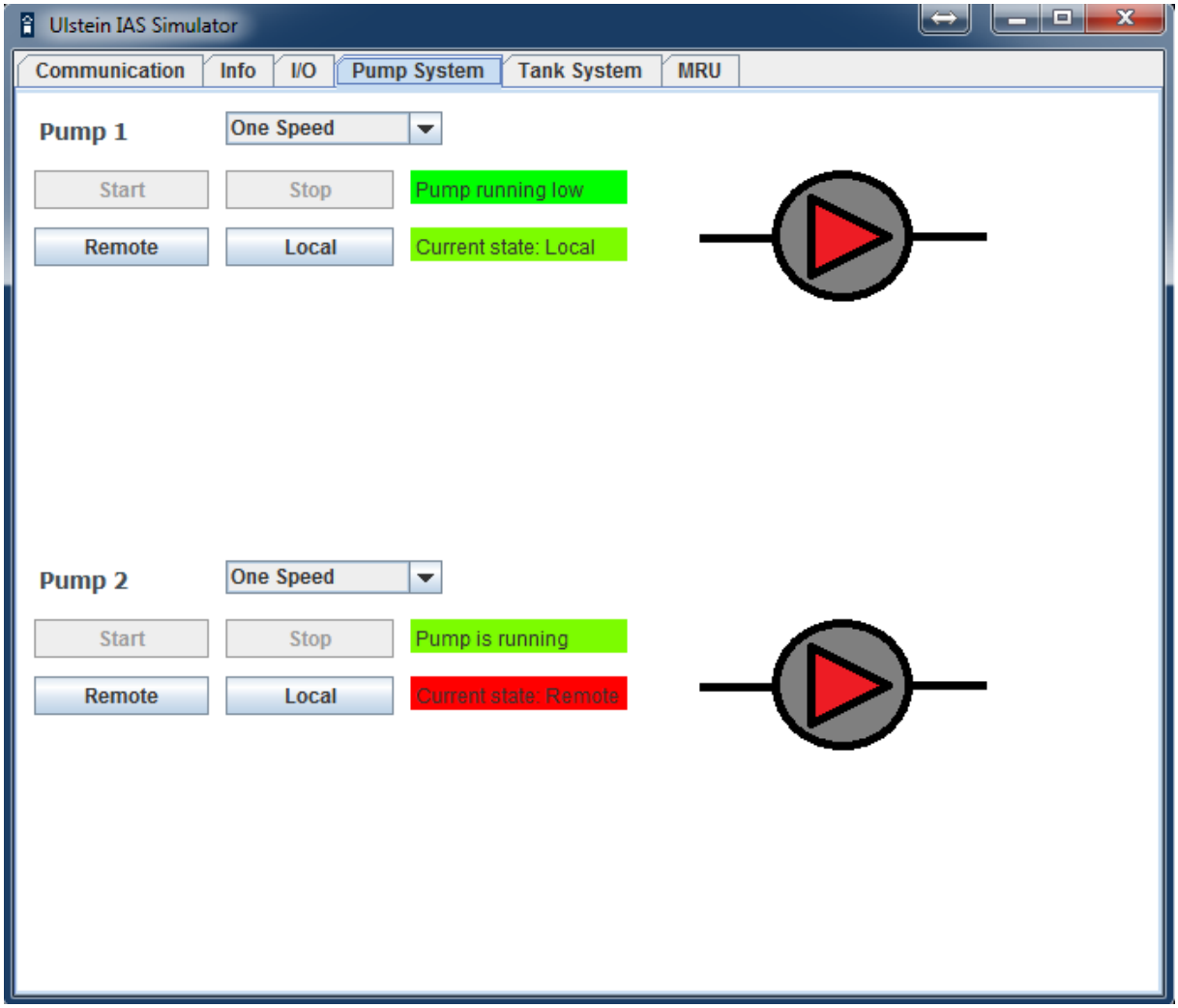
Figur 3: Fane for test av generell I/O

Fane for I/O(input/output) test av generell I/O eller system. Funksjonen til I/O simulatoren er å kunne kople på analoge eller digitale I/O uavhengig av funksjon til det som er påkopla. Simulatoren har 4 I/O. 1stk digital utgang, 1 stk digital inngang, 1 stk analog utgang og 1 stk analog inngang.

- *Digital Output* har to knappar. Eit trykk på *ON* knappen vert utgongen på kontroller sett høg. Trykk på *OFF* nappen sett utgong lav.
- *Digital Input* viser stauts til utgong. Tekstfeltet under viser *true* vist inngangen er sett høg. Viser *false* vist inngangen er sett lav.
- *Analog Out* er ein slider som er kopla opp mot ein analog utgong på kontroller. Den analoge utgongen gir 4-20mA signal ut. 4mA når slider er heilt til venstre. 20mA når slider er heilt til høgre. Signaler er lineært.
- *Analog Input* viser ein progressbar på signalet. Progressbaren er tilkopla en analog 4-20mA inngang på kontroller.

For koplingsskjema vises det til I/O liste for IAS simulator.

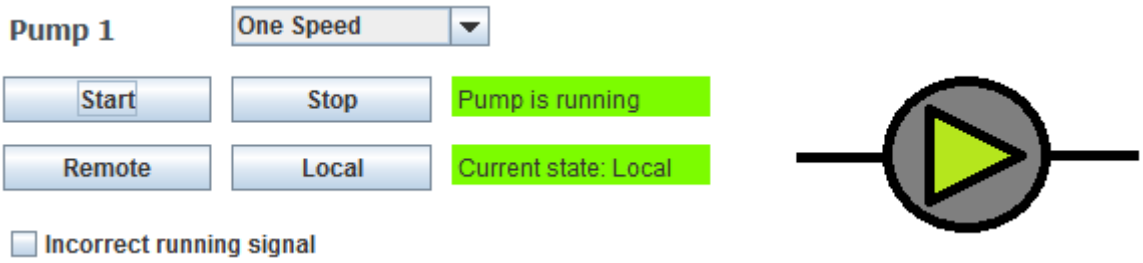
PUMPESYSTEM



Figur 4: Fane for test av pumper

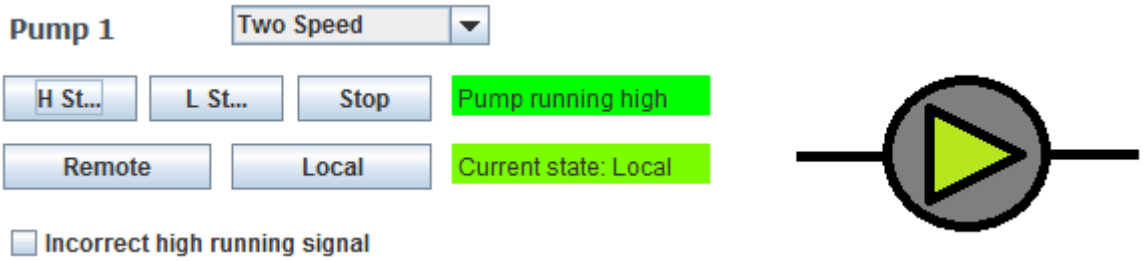
Pumpesimulatoren simulerer fjernstyrte pumper. Hovudtanken til pumpesimulatoren er at ein skal kunne starte og stoppe pumper, gi signal for om ei pumpe går eller ikkje, gi feil signal for om ei pumpe går og velje om ein skal kjøre pumpa lokalt eller fjernstyrt. Kjører ein pumpa med fjernstyringa på så kan ein starte og stoppe pumpa i pumpesimulatoren via IASen. Feil signal vil sei at pumpesimulatoren sender melding til IAS at pumpa har stoppa sjølv om den i verkelegheita går

- Pumpe med ei hastigheit kjem opp automatisk kvar gong pumpesimulatoren blir starta. Denne er identisk med første utkastet av pumpesimulatoren. Her har ein enkel simulering av start og stopp ilag med dei andre funksjonane til pumpa.



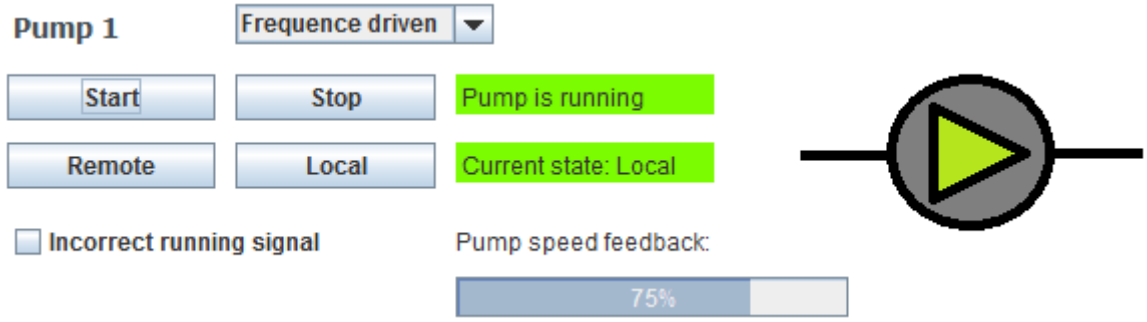
Figur 5: Illustrasjon av pumpesimulator for pumpe med ei hastigheit.

- Forskjellen mellom pumpe med ei hastigheit og to hastigheiter er måten å starte den på. Når ein vel å simulere pumpe med to hastigheiter får ein to val ved start av pumpa, ein knapp for høg start og ein for låg start.



Figur 6: Illustrasjon av pumpesimulator for pumpe med to hastigheiter, her starta i høg hastigheit.

- Vel ein å simulere frekvensstyrt pumpe får ein opp same grafikken som for pumpe med ei hastigheit. Denne pumpa blir starta og stoppa på same måte, men har i tillegg ein indikator som viser hastigheita til pumpa frå 0 til 100 prosent.

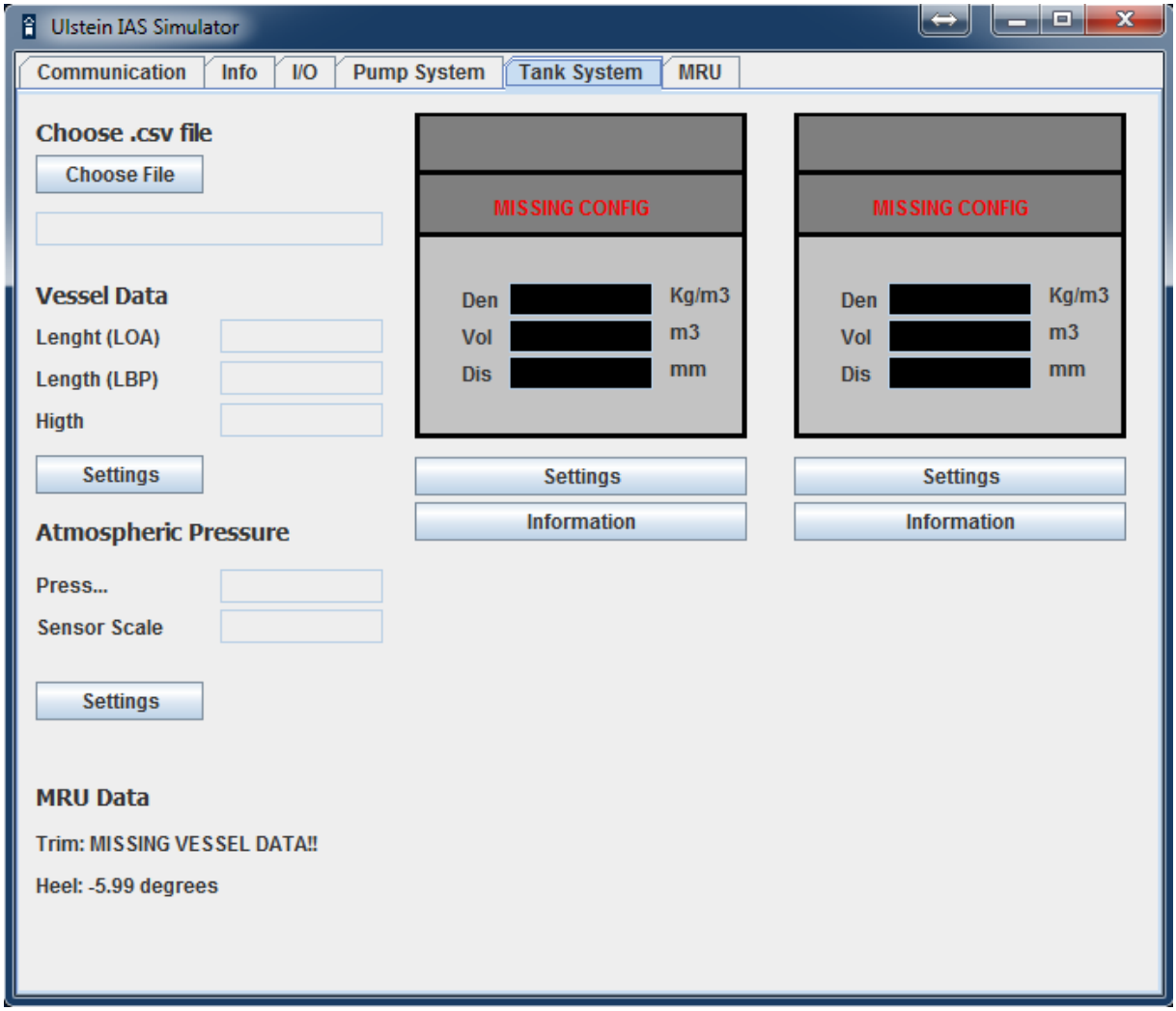


Figur 7: Illustrasjon av pumpesimulator for frekvensstyrt pumpe, her med 75% pådrag.

- Skal ein styre pumpene via IASen så sett ein pumpa i fjernstyrt-tilstand. Då vil start- og stopp-knappane bli utilgjengelege i GUIen og ein indikasjon på at pumpa står i fjernstyrt tilstand kjem fram.

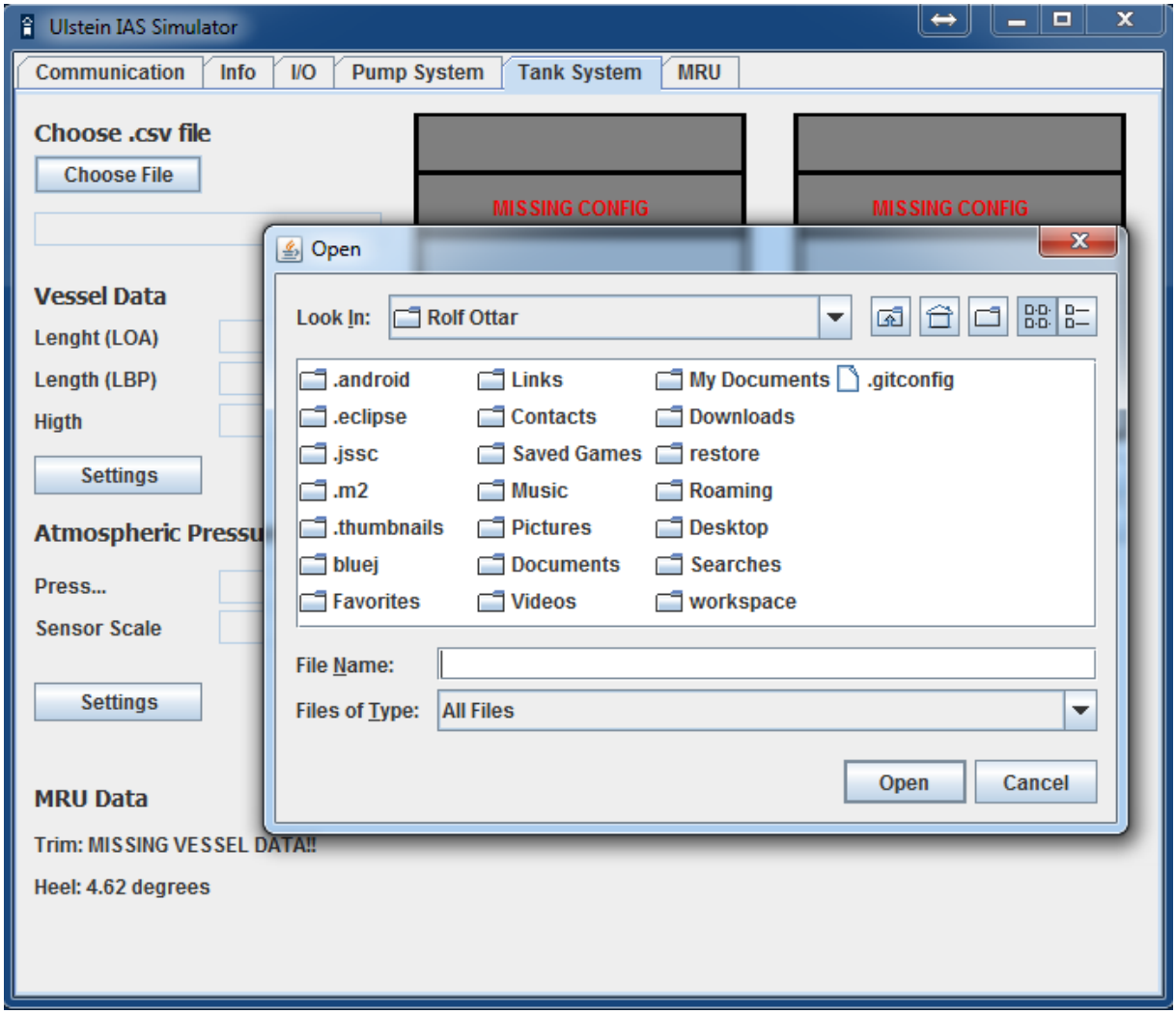
For koplingsskjema vises det til I/O liste for IAS simulator.

TANKPEILING SIMULATOR



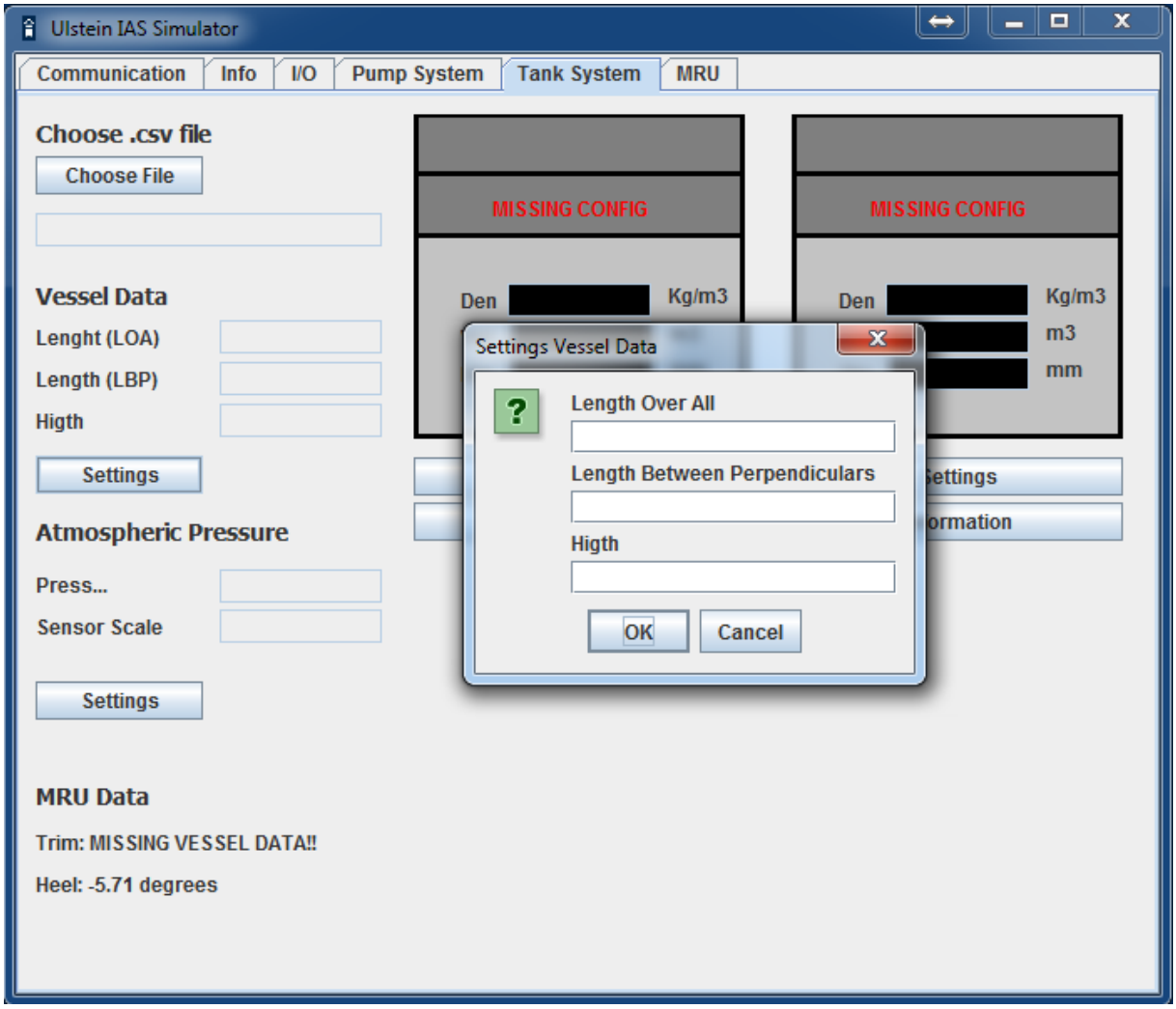
Figur 8: Tankpeiling-simulator

- Funksjonaliteten til tankpeiling-simulatoren er simulering av væskenivå i tank. Tankpeiling-simulatoren tek inn same MRU verdiane som Ulstein IAS motek serielt frå MRU. Tankpeiling-simulatoren brukar desse til kalkulering av tanksensor verdi.
- Total volum på væska i tanken vert avgjort ved oppslag i tanktabell. Forholda mellom væskenivå i tank og bevegelse på fartyet er faktorane som påverkar tanksensorane.



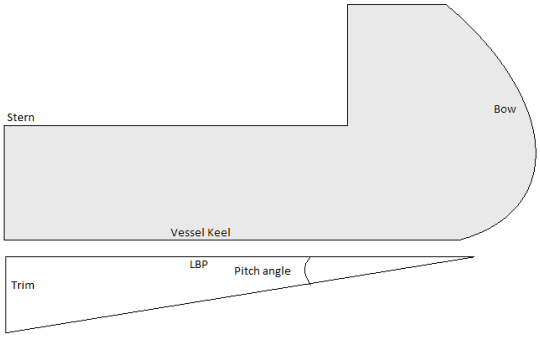
Figur 9: Tankpeiling-simulator fil-velger

- For simulering av tankpeiling system må tanktabell på IAS og IAS sumilator samsvare. IAS simulator bruker tanktabell av type csv-fil. Denne leggast til i simulatoren ved bruk av fil-velger.

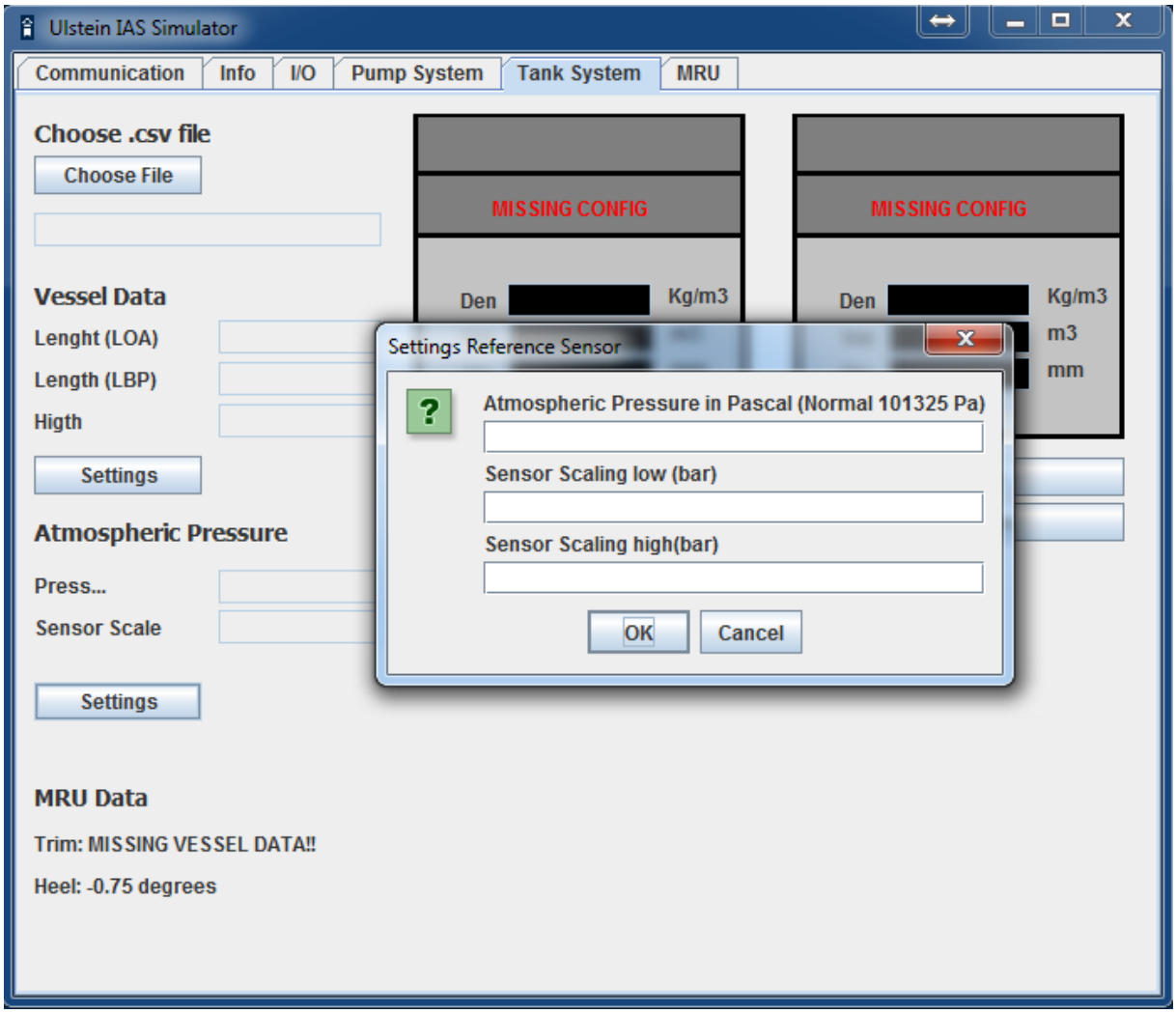


Figur 10: Tankpeiling-simulator sett av Vessel Data parameter

- Feltet *Vessel Data* ver parametera for høgd og lengder på fartyet sett. Ved å trykke på *settings* knappen kjem det opp eit nytt vindu for å sette parameter. LOA (Length Over All) LBP(Length Between Perpendiculars) er måleeiningar for lengd på fartyet. LBP vert brukt til kalkulering av trim for fartyet. Sett fartyet i profil frå sida er trim høgdeforskjellen mellom baug og akterskip.

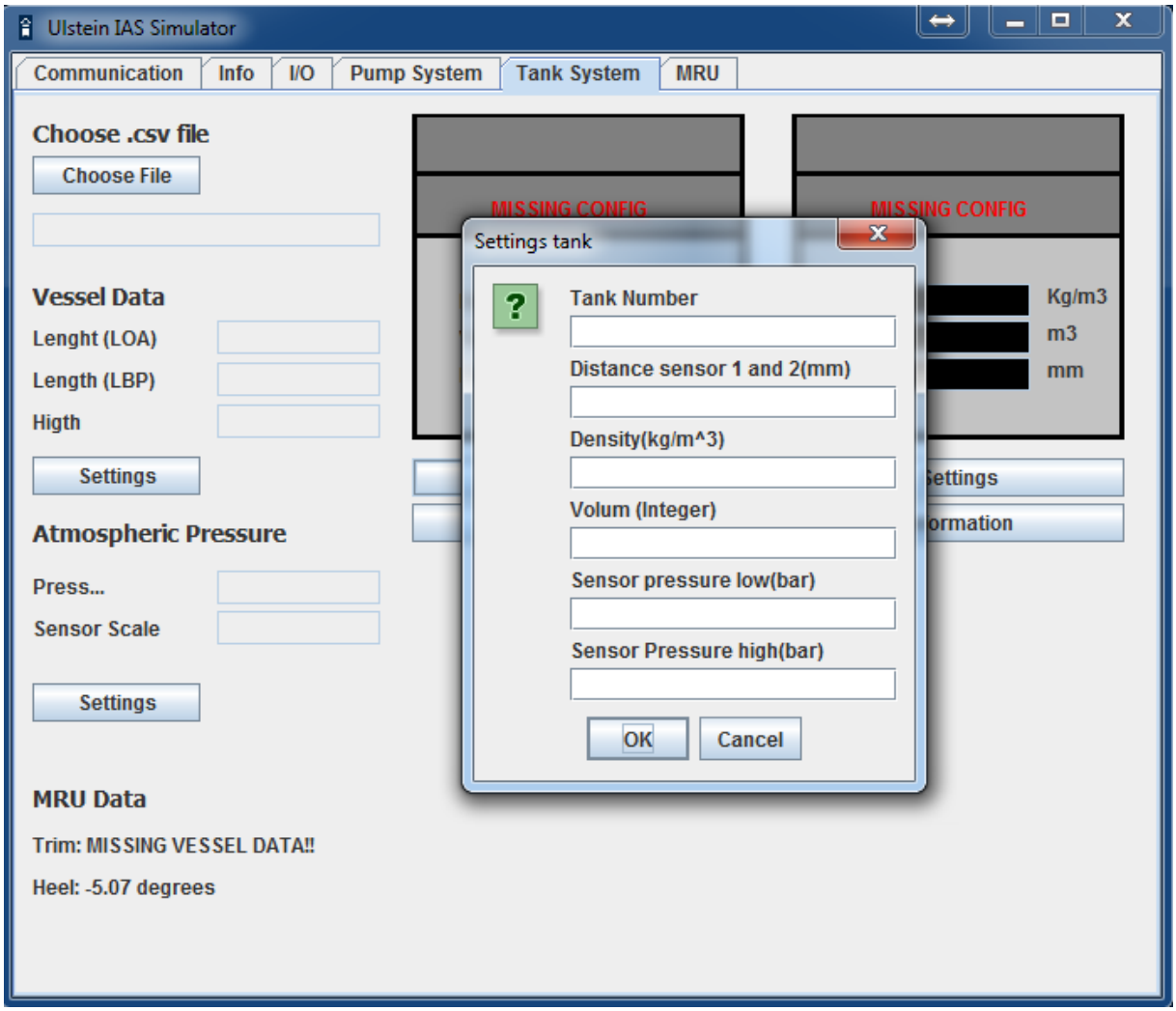


Figur 11: Illustrasjon av fartøyets trim.



Figur 12: Tankpeiling-simulator, sett av referanse sensor

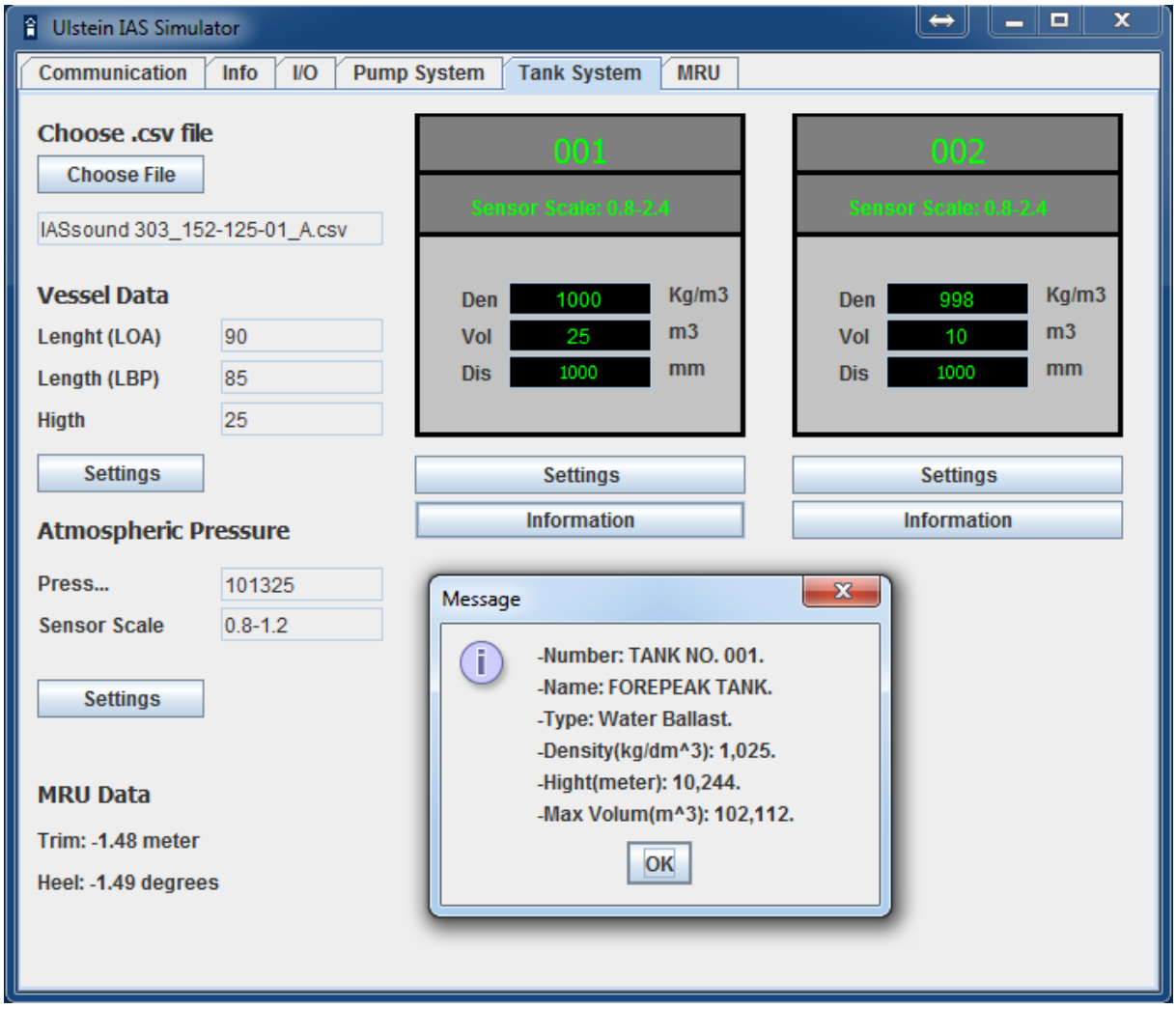
- Atmospheric Pressure legg ein inn parameter for ønska atmosfærisk trykk og sensor måleområde. Ein sensor kan eksempelvis vere oppgitt til 4-20mA i måleområdet 0,8-1,2 bar. Det gir 4mA signal ved 0,8 bar og 20mA ved 1,2 bar. Måleområdet for sensor er sett på som lineært. Dermed vert det brukt lineær interpolering for skalering av signal.



Figur 13: Tankpeiling-simulator, sett av tank data

Parameter for tanken er sett ved å trykke på knappen *settings* under den grafiske framstillinga av tanken. Nytt vindauge dukkar opp. Parametera som skal settast er:

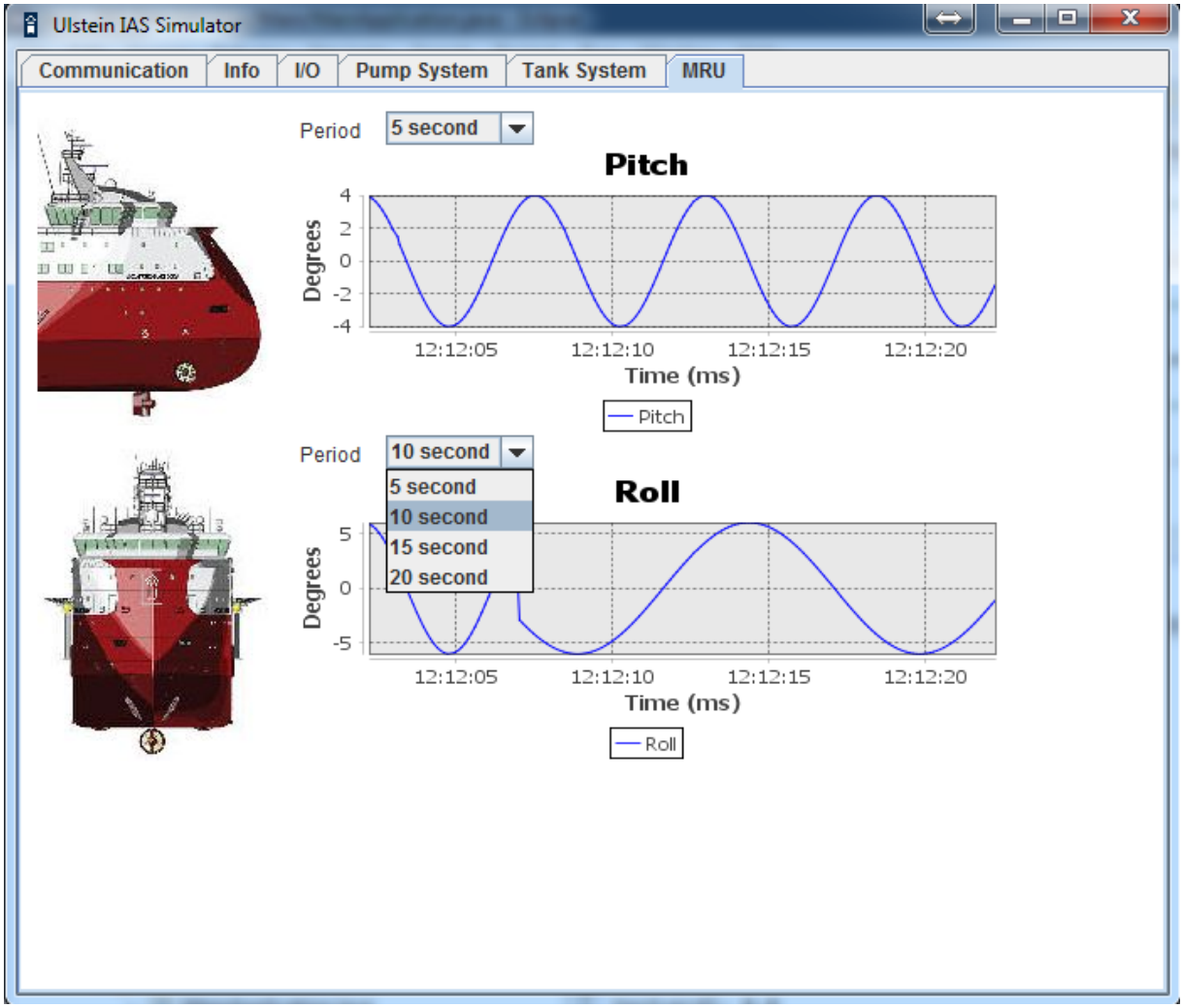
- Val av tank som skal simulerast. Tankane er nummerert som for eksempel "001", "041", "151".
- Distanse mellom sensor 1 og sensor 2 i tanken. Dette vert brukt til simulering av trykkforskjell mellom dei to sensorane. Dette for å teste av automatisk tettleikmåling til IAS.
- Massetettheita til væska på tanken.
- Volum. Væskeinnhald i tanken.
- Start måleområde til sensor.
- Slutt måleområdet til sensor.



Figur 14: Tankpeiling-simulator, informasjon om valt tank

For infomasjon om valt tank, trykk på *Information* knappen. Eit extra vindauge dukkar opp med informasjonom den valte tanken.

For koplingsskjema vises det til I/O liste for IAS simulator.



Figur 15: Tankpeiling-system, MRU

Med MRU-simulator kan ein simulere skipets rørsle i bølger. MRU-simulatoren sender informasjon direkte inn på IAS via NMEA seriellkommunikasjon. Seriell tilkopling skjer i fana for kommunikasjon. Ved å endre på tiden i rullegardina, endra en perioden for sinus funksjonen.

Vedlegg 9

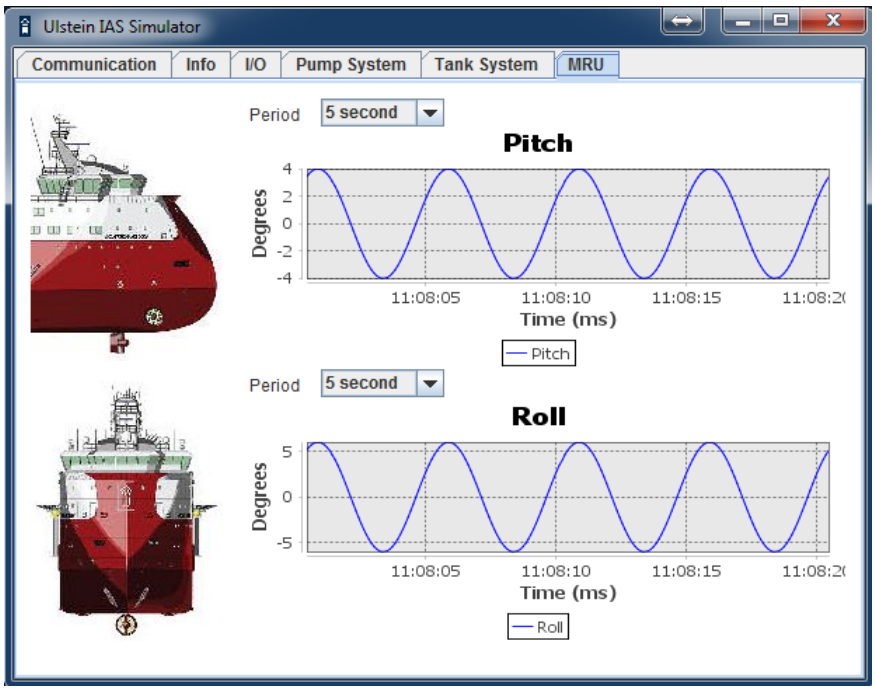
Programmeringsguide

ULSTEIN IAS SIMULATOR PROGRAMMERING GUIDE

GUI

GUIBase er hovudramma til Ulstein IAS Simulator. Klassa GUIBase er ei utviding av Java Swing JFrame. Klassa opprettar ei ramme utan innhald. GUIBase har metoda *addTab* der en kan legge til GUI av type Jpanel. Metoda *addTab* vil legge fortløpande til faner i GUIBase.

```
public void addTab(JPanel jpanel, String title) {
	jtp.addTab(title, jpanel);

}
```



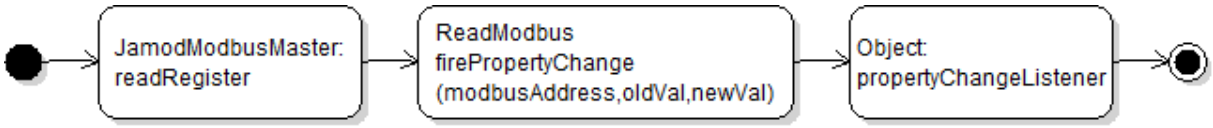
Figur 1: GUI Ulstein IAS Simulator

READ AND WRITE MODBUS

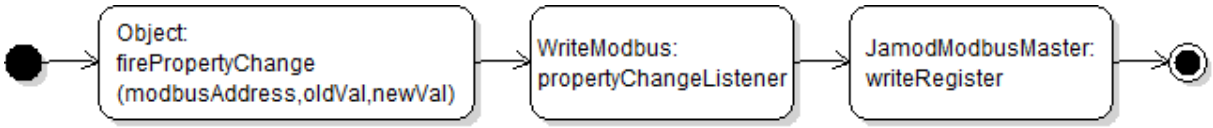
ReadModbus klassa arva SwingWorker. Klassa har en hovudfunksjon, å lese Modbus register til Modbus slave. Når ein opprettar eit objekt av type ReadModbus sett ein følgjande parameter:

- ✓ JamodModbusMaster: objekt av type JamodModbusMaster. Kommunikasjon for modbus-TCP master/slave.
- ✓ Kva registerområde som skal lesast. Eksempel: frå register 12388 til 12398.
- ✓ Frekvens. Kor ofte Modbusregistera skal lesast.

Når eit objekt av type ReadModbus er oppretta, startar en *worker thread.* Denne tråden går i løkke i frekvens som er sett for objektet. For kvar loop vert registera lest ved å bruke JamodModbusMaster-objektet med metoda *readRegister*. For kvart register som vert lest vert det kjørt ein *firePropertyChange.* Her vert det sjekka om registeret har endring i ny verdi forhold til gammel verdi. Er dette uendra vert ikkje *PropertyChangeEvent sendt.* Når det er endringar ulikheit mellom gammel og ny verdi vert *PropertyChangeEvent* fanga opp av alle registrerte lyttarar.



WriteModbus tek inn JamodModbusMaster objekt. WriteModbus klassa har kun en metode. Denne metoda opprettar ein lyttarar til SwingWorker objekt. Swingworker objekt og modbusadresse er parametera for å opprette ein lyttar. Denne lyttaren fangar opp *PropertyChangeEvent* på det objektet den er knytt mot. *PropertyName* er satt likt som Modbusadressa for å skilje dei ulike *PropertyChangeEvents*. Når den registrerte lyttaren fangar opp ein *PropertyChangeEvent* vil denne sende ein Modbus reguest til Modbus slave. JamodModbusMaster objektet og metoda *writeRegister* vert brukt for å sende Modbus request.



SERIELL KOMMUNIKASJON

Klassa SerialCommunication importerar biblioteket JSSC. Klassa har metodar for oppretting av seriell forbindelse, lukke seriell forbindelse, sjekk om seriell forbindelse er oppretta, lese seriell port og skrive til seriell port.

MAINAPPLICATION

I MainApplication klassa er metoda *main.* Denne metoda vert kjørt når javaprogrammet starter. I denne metoden vert alle Javaobjekt for Ulstein IAS Simulator oppretta og samhandla. Under viser eit eksempel på korleis opprette eit objekt som arvar JPanel å vidare legge det til i GUIBase.

```
GUIInformation info = new GUIInformation();

guiBase.addTab(info, "Info");
```

Skal klassa kommunisere med kontroller må den arve SwingWorker for skriving til kontroller, og/eller opprette lyttar for klassa ReadModbus for lesing av kontroller.

Eksempel for oppretting av skriving til kontroller:

```
public class MyClass extends SwingWorker {
	private Object oldVal;

	public synchronized void sendDataToController(int val) {
		this.firePropertyChange("modbusAddress", oldVal, val);
		oldVal = val;
	}

	@Override
	protected Object doInBackground() throws Exception {
		// Do something ones thread, or create loop and get a running thread.
		return null;
	}

}
```

Objektet til WriteModbus klassa har metoda *addSingleListner.* For å sende Modbus request til kontroller må det legges til ein lyttar på Myclass.

```
JamodModbusMaster modbusMaster = new JamodModbusMaster();
WriteModbus wm = new WriteModbus(modbusMaster);


wm.addSingleListner(myClass, "modbusAddress");
```

For lesing av inngangar til kontroller må ReadModbus objektet vere sett til å lese den gitte adressa. Eksempel på lesing frå adresse 12388 til adresse 12397, 10 gongar i sekundet.

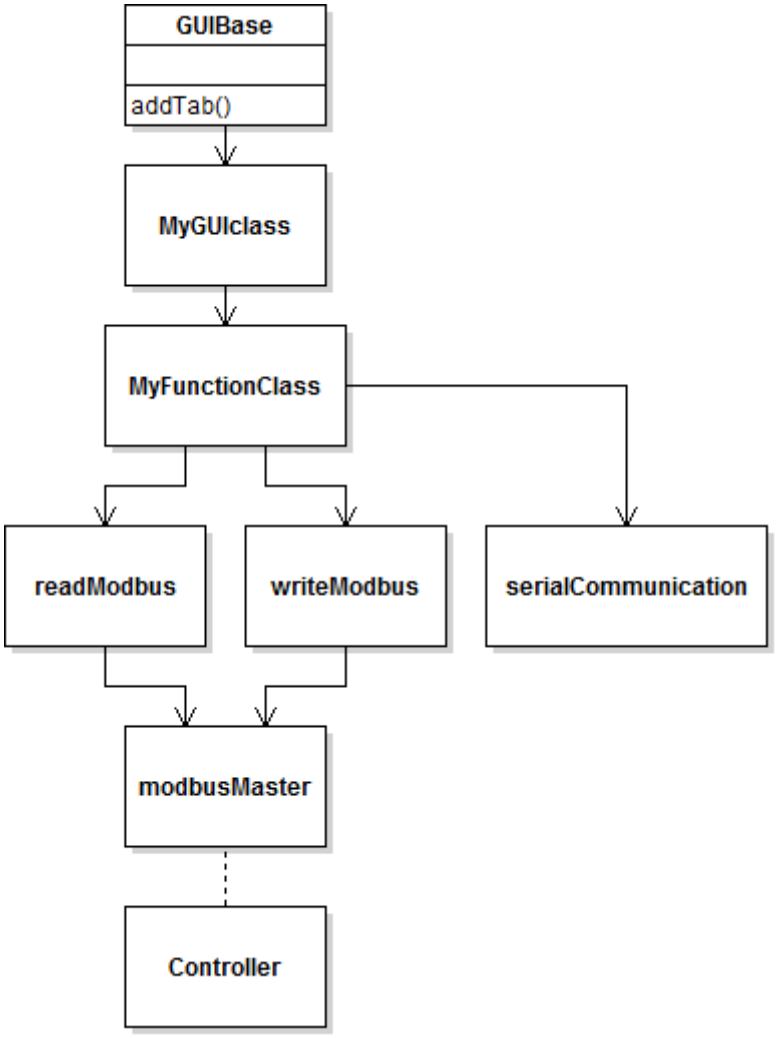
```
ReadModbus readModbus = new ReadModbus(modbusMaster, 12388, 12397, 10);
```

For at MyClass skal fange opp ModbusRequest må den legge til ein lyttar på ReadModbus objektet. Eksempel:

```
public class MyClass extends SwingWorker {
	private Object oldVal;
	private ReadModbus readModbus;

	public MyClass(ReadModbus readModbus) {
		this.readModbus = readModbus;

		((SwingWorker) readModbus).addPropertyChangeListener(new
PropertyChangeListener() {
		@Override
		public void propertyChange(PropertyChangeEvent evt) {
			if (evt.getPropertyName().equals("modbusAddress")) {
				// Do something

					}
				}
			});
	}

	public synchronized void sendDataToController(int val) {
		this.firePropertyChange("modbusAddress", oldVal, val);
		oldVal = val;
	}

	@Override
	protected Object doInBackground() throws Exception {
		// Do something ones thread, or create loop and get a running thread.
		return null;
	}

}
```

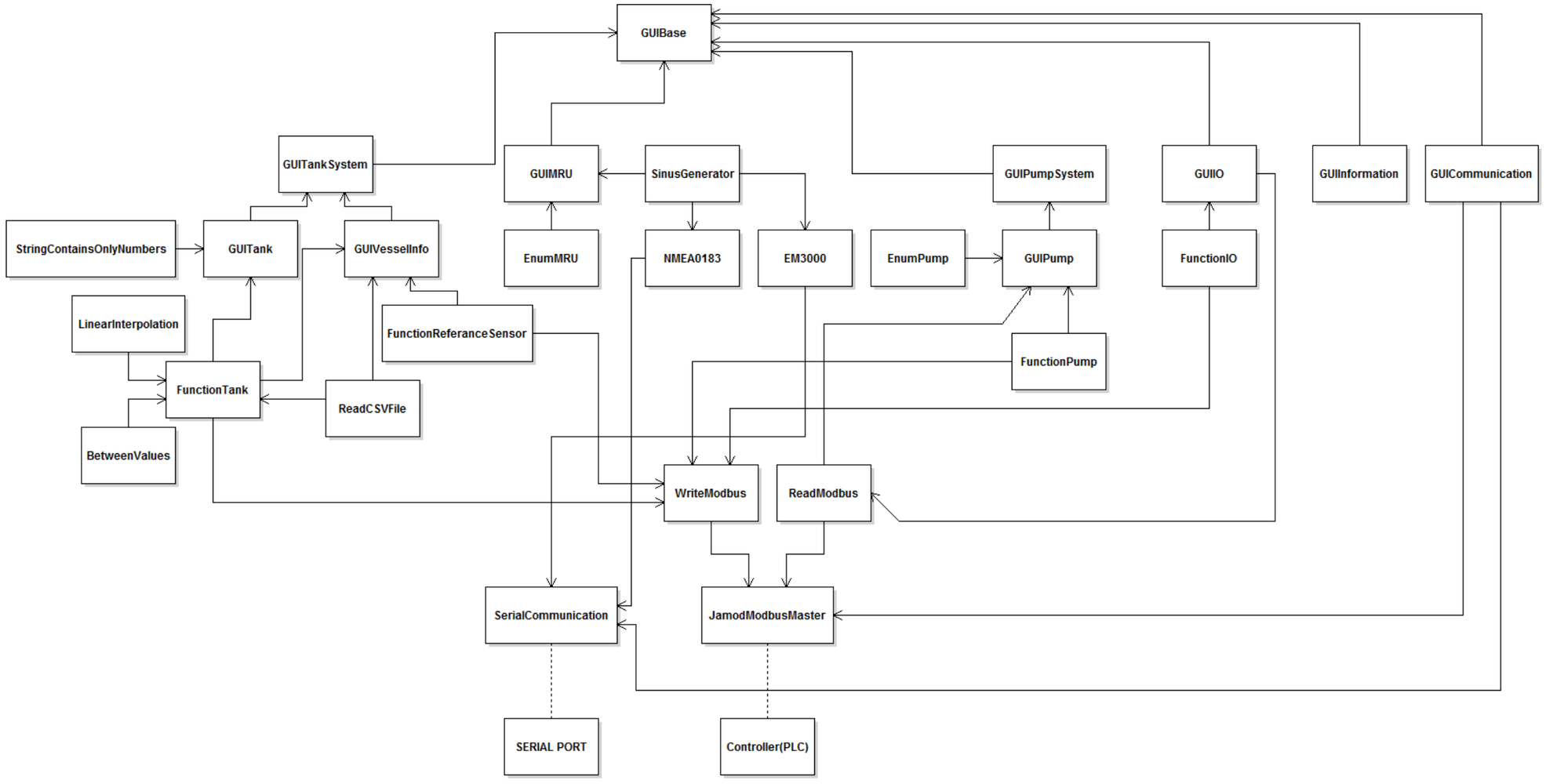
For skriving og lesing av seriell kommunikasjon må klassa ta in objektet *serialCommunication.*Dette objektet er av type SerialCommunication klassa og har metodar for skriving og lesing av serieport.

KLASSEDIAGRAM



Vedlegg 10

Klassediagram


GUIBase
GUITankSystem
GUIMRU
SinusGenerator
GUIPumpSystem
GUIIO
GUIInformation
GUICommunication
StringContainsOnlyNumbers
GUITank
GUIVesselInfo
EnumMRU
NMEA0183
EM3000
EnumPump
GUIPump
FunctionIO
LinearInterpolation
FunctionReferanceSensor
FunctionPump
FunctionTank
ReadCSVFile
BetweenValues
WriteModbus
ReadModbus
SerialCommunication
JamodModbusMaster
SERIAL PORT
Controller(PLC)

Vedlegg 11

I/O-liste

WAGO addr	Modbus addr	System	Description	Var	Scale	Input/Output
MW0	12288	General Signals	Digital Output	Boolena	0-1	DO
MW1	12289	General Signals	Analog Output	Integer	0-32767	AO
MW2	12290	Pump-simulator	Pump 1 signal remote	Boolean	0-1	DO
MW3	12291	Pump-simulator	Pump 1 signal running	Boolean	0-1	DO
MW4	12292	Pump-simulator	Pump 2 signal remote	Boolean	0-1	DO
MW5	12293	Pump-simulator	Pump 2 signal running	Boolean	0-1	DO
MW6	12294	Pump-simulator	Pump 1 speed feedback signal	Integer	0-32767	AO
MW7	12295	Pump-simulator	Pump 2 speed feedback signal	Integer	0-32767	AO
MW8	12296	Pump-simulator	Pump 1 signal running high	Boolean	0-1	DO
MW9	12297	Pump-simulator	Pump 2 signal running high	Boolean	0-1	DO
MW10	12298	Tankpeiling-simulator	Tank 1 - sensor 1	Integer	0-32767	AO
MW11	12299	Tankpeiling-simulator	Tank 1 - sensor 2	Integer	0-32767	AO
MW12	12300	Tankpeiling-simulator	Tank 2 - sensor 1	Integer	0-32767	AO
MW13	12301	Tankpeiling-simulator	Tank 2 - sensor 2	Integer	0-32767	AO
MW14	12302	Tankpeiling-simulator	Tank 3 - sensor 1	Integer	0-32767	AO
MW15	12303	Tankpeiling-simulator	Tank 3 - sensor 2	Integer	0-32767	AO
MW16	12304	Tankpeiling-simulator	Tank 4 - sensor 1	Integer	0-32767	AO
MW17	12305	Tankpeiling-simulator	Tank 4 - sensor 2	Integer	0-32767	AO
MW18	12306	Tankpeiling-simulator	Refsensor	Integer	0-32767	AO
MW19	12307					
MW20	12308					
MW21	12309					
MW22	12310					
MW23	12311					
MW24	12312					
MW25	12313					
MW26	12314					
MW27	12315					
MW28	12316					
MW29	12317					
MW30	12318					
MW31	12319					
MW32	12320					
MW33	12321					
MW34	12322					
MW35	12323					
MW36	12324					
MW37	12325					
MW38	12326					
MW39	12327					
MW40	12328					
MW41	12329					
MW42	12330					
MW43	12331					
MW44	12332					
MW45	12333					
MW46	12334					
MW47	12335					
MW48	12336					

MW49	12337					
MW50	12338					
MW51	12339					
MW52	12340					
MW53	12341					
MW54	12342					
MW55	12343					
MW56	12344					
MW57	12345					
MW58	12346					
MW59	12347					
MW60	12348					
MW61	12349					
MW62	12350					
MW63	12351					
MW64	12352					
MW65	12353					
MW66	12354					
MW67	12355					
MW68	12356					
MW69	12357					
MW70	12358					
MW71	12359					
MW72	12360					
MW73	12361					
MW74	12362					
MW75	12363					
MW76	12364					
MW77	12365					
MW78	12366					
MW79	12367					
MW80	12368					
MW81	12369					
MW82	12370					
MW83	12371					
MW84	12372					
MW85	12373					
MW86	12374					
MW87	12375					
MW88	12376					
MW89	12377					
MW90	12378					
MW91	12379					
MW92	12380					
MW93	12381					
MW94	12382					
MW95	12383					
MW96	12384					
MW97	12385					
MW98	12386					

MW99	12387					
MW100	12388	General Signals	Digital Input	Boolean	0-1	DI
MW101	12389	General Signals	Analog Input	Integer	0-32767	AI
MW102	12390	Pump-simulator	Pump 1 signal start IAS	Boolean	0-1	DI
MW103	12391	Pump-simulator	Pump 1 signal stop IAS	Boolean	0-1	DI
MW104	12392	Pump-simulator	Pump 2 signal start IAS	Boolean	0-1	DI
MW105	12393	Pump-simulator	Pump 2 signal stop IAS	Boolean	0-1	DI
MW106	12394	Pump-simulator	Pump 1 signal High start IAS	Boolean	0-1	DI
MW107	12395	Pump-simulator	Pump 2 signal High start IAS	Boolean	0-1	DI
MW108	12396	Pump-simulator	Pump 1 speed signal	Integer	0-32767	AI
MW109	12397	Pump-simulator	Pump 2 speed signal	Integer	0-32767	AI
MW110	12398					
MW111	12399					
MW112	12400					
MW113	12401					
MW114	12402					
MW115	12403					
MW116	12404					
MW117	12405					
MW118	12406					
MW119	12407					
MW120	12408					
MW121	12409					
MW122	12410					
MW123	12411					
MW124	12412					
MW125	12413					
MW126	12414					
MW127	12415					
MW128	12416					
MW129	12417					
MW130	12418					
MW131	12419					
MW132	12420					
MW133	12421					
MW134	12422					
MW135	12423					
MW136	12424					
MW137	12425					
MW138	12426					
MW139	12427					
MW140	12428					
MW141	12429					
MW142	12430					
MW143	12431					
MW144	12432					
MW145	12433					
MW146	12434					
MW147	12435					
MW148	12436					

MW149	12437					
MW150	12438					
MW151	12439					
MW152	12440					
MW153	12441					
MW154	12442					
MW155	12443					
MW156	12444					
MW157	12445					
MW158	12446					
MW159	12447					
MW160	12448					
MW161	12449					
MW162	12450					
MW163	12451					
MW164	12452					
MW165	12453					
MW166	12454					
MW167	12455					
MW168	12456					
MW169	12457					
MW170	12458					
MW171	12459					
MW172	12460					
MW173	12461					
MW174	12462					
MW175	12463					
MW176	12464					
MW177	12465					
MW178	12466					
MW179	12467					
MW180	12468					
MW181	12469					
MW182	12470					
MW183	12471					
MW184	12472					
MW185	12473					
MW186	12474					
MW187	12475					
MW188	12476					
MW189	12477					
MW190	12478					
MW191	12479					
MW192	12480					
MW193	12481					
MW194	12482					
MW195	12483					
MW196	12484					
MW197	12485					
MW198	12486					

MW199	12487					
MW200	12488					

Vedlegg 12

Poster

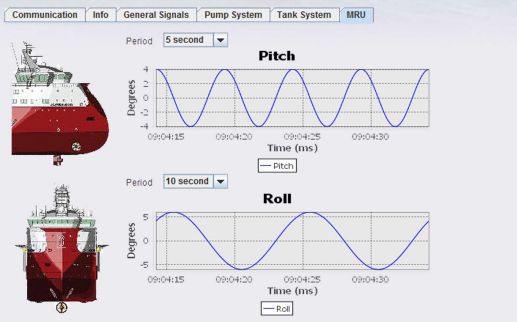
ULSTEIN IAS SIMULATOR

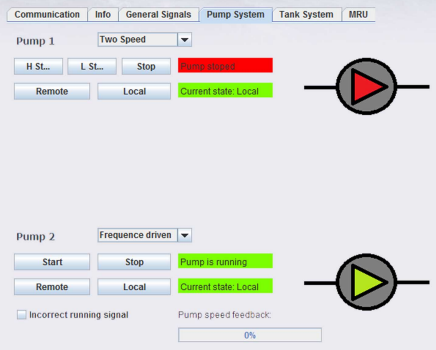
Rolf Ottar Rovde - Kristian Østgaard

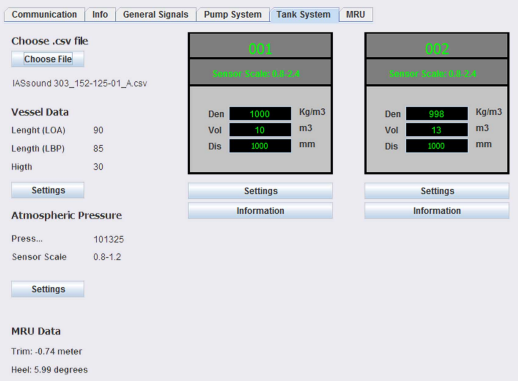
Ulstein IAS simulator er eit produkt utvikla som eit hjelpemiddel for fabrikktesting(FAT) av Ulstein IAS. Simulatoren vil vere med på å kvalitetssikre det ferdige IAS produktet til Ulstein. På dagens FAT er det fleire kritiske punkt som ikkje kan testast pga. mangel på signal frå andre system. Ved å ta i bruk Ulstein IAS simulator vil Ulstein også kunne teste disse punkta, og dermed avdekke feil på eit tidlegare tidspunkt. Det vil føre til eit betre produkt og innsparing av både tid og kostnad ved i gangkjøring om bord i skip. Ulstein IAS simulator er ei hovudoppgåve hjå Høgskolen i Ålesund og er utvikla i samarbeid med sentrale personar innan utvikling av dagen Ulstein IAS.

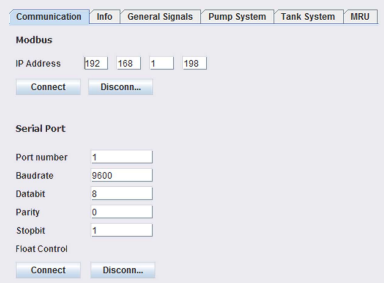
Simulatoren er programmert i Java og tek føre seg fire delsystem: MRU, tankpeiling, pumpe og testing av analoge og digitale IO. Brukargrensesnittet til simulatoren oppretta i ei ramme som er inndelt i faner, der kvart delsystem har si fane. Dette gjer systemet oversiktleg og enkelt å ta i bruk. MRU-simulatoren skal simulere skipets rørsle gjennom bølger. Den tek for seg roll og pitch som blir vist i ein dynamisk graf i fana. Tankpeiling-simulatoren skal gi ut volumet av veska i ein tank kompensert for skipets rørsle. Med pumpe-simulatoren kan ein simulere tre forskjellige type pumper. Pumpe med ei hastigheit, pumpe med to hastigheiter og frekvensstyrt pumpe. Simulatoren for testing av analoge og digitale IO er sett opp for funksjonstesting.

Ulstein IAS simulator er programmert for å enkelt kunne utvidast eller legge til simulatorar for fleire delsystem. Måten dette er gjort på er at kvart delsystem er programmert som sjølvstendige system kvar for seg, og deretter lagt til som faner i sjølve simulatoren. Simulatoren opererer i sanntid og er programmert som eit multitrådsystem. Simulatoren er knytt opp mot ein Wago kontroller med tilhøyrande analoge og digitale IOar. Kommunikasjon skjer via Modbus og seriell kommunikasjon. Seriell kommunikasjon blir brukt av MRU-simulator som gir ut informasjon via både NMEA 0183- og EM3000-protokoll. Mellom Java-applikasjon og kontroller kommuniserast det over Modbus.









ULSTEIN®

HØGSKOLEN I ÅLESUND
Aalesund University College