NTNU

Kunnskap for en bedre verden

Bacheloroppgave

IE303612 Automatiseringsteknikk

USV - Unmanned Surface Vessel

Kandidatnummer: 807, 819, 829

Totalt antall sider inkludert forsiden: 368

Innlevert Ålesund, 26.05.2016

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.**	☒
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 NTNUs studieforskrift.**	☒
6.	**Jeg/vi har satt oss inn i regler og retningslinjer i bruk av kilder og referanser på biblioteket sine nettsider**	☒

Publiseringsavtale

Studiepoeng: 20

Veileder: Ottar L. Osen, Rune Volden

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 med forfatter(ne)s godkjennelse.
Oppgaver som er unntatt offentlighet eller båndlagt vil ikke bli publisert.

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

Dato: 26.05.2016

Bachelor oppgave
Rapport

NTNU
Kunnskap for en bedre verden

Tittel:
USV - Unmanned Surface Vessel

Kandidatnummer(e):			
807; 819; 829			
Dato:	Fagkode:	Fagnavn:	Dokument tilgang:
26.05.2016	IE303612	Bacheloroppgave	
Studium:		Ant sider/Vedlegg:	Bibl. nr:
Automatiseringsteknikk		103/11	

Veileder(e):
Ottar Osen; Rune Volden

Sammendrag:

Denne rapporten omhandler utvikling av et ubemannet overflatefartøy, som er gitt som en bacheloroppgave av NTNU i Ålesund. Formålet med oppgaven er utvikling av et konsept for et ubemannet overflatefartøy som danner basis for videre utvikling i senere prosjekter. I tillegg skal det utvikles en prototype for testing av mulige løsninger for fremdriftssystem, kontrollsystem, instrumentering og dynamisk posisjonering. Viktige moment ved utvikling og design av ubemannede overflatefartøy er belyst i rapporten, og en prototype basert på en Pioner 8-fots jolle er utviklet, med fokus på funksjonalitet og lav kostnad. Prototypen har fire fastmonterte utenpåliggende thrustere. Dynamisk posisjonering bestående av tre PID-regulatorer er utviklet, og thrusterallokeringsproblemet er løst med kvadratisk programmering. Kontrollsystemet er basert på en Odroid XU4, og programvare er utviklet i Java. Til instrumentering er det benyttet billige komponenter, og Arduinoer for I/O. Resultatene fra testingen viser at det vil være mulig å oppnå dynamisk posisjonering innenfor de gitte kravene med løsningen som presenteres.

Denne oppgaven er en eksamensbesvarelse utført av studenter ved NTNU i Ålesund.

Postadresse
NTNU Ålesund
PB 1517
6025 Ålesund
Norway

Besøksadresse
Larsgårdsvegen 2
6009 Ålesund
Internett
www.ntnu.no

Telefon
73 59 50 00
Epostadresse
postmottak@alesund.ntnu.no

Faktura
PB 50, Økern
0508 Oslo
Foretaksregisteret
NO 974 767 880

Forord

Denne bacheloroppgaven er skrevet at tre studenter i Automatiseringsteknikk ved NTNU i Ålesund, og markerer avslutningen på tre års ingeniørstudier. Formålet med prosjektet er å utrede mulige løsninger for et ubemannet overflatefartøy, samt bygging av en prototype for uttesting av instrumenteringsløsninger, programvare, fremdriftssystem og dynamisk posisjonering.

I en tid der offshore-næringen blir presset hardt på kostnad er det tid for å tenke nytt og utvikle nye løsninger. I tilegg til kostnadsbesparelsen med å ha mindre sjømenn ute på havet kan automatisering også bidra til økt sikkerhet, siden det gjennom historien er bundet med stor fare i å ferdes på havet. Automatisering av fartøy og utvikling av ubemannede skip kan derfor også føre til store samfunnsmessige konsekvenser, særlig for folk som jobber på sjøen og deres nærmeste. I tillegg er det stort fokus på utnyttelse av havets ressurser, og stadig mer automatisering innen havbruk og fiskeri.

Motivasjonen bak oppgavevalget var muligheten for å kunne benytte store deler av kunnskapene vi har ervervet oss gjennom tre års studier i en praktisk oppgave. Oppgaven bærer også preg av tverrfaglighet, og gruppen så på dette som en utfordring og mulighet til å utvide våre faglige horisonter. Vi håper at senere prosjekter kan utnytte erfaringer fra dette prosjektet i den videre utviklingen av et ubemannet overflatefartøy ved NTNU i Ålesund.

Vi vil gjerne takke alle bidragsytere som har gitt oss støtte gjennom prosjektet, og spesielt vil vi takke:

- Våre veiledere Ottar Osen og Rune Volden for god støtte, motiverende samtaler og god veiledning gjennom hele prosjektet.
- Labingeniør Anders Sætersmoen for bistand ved innkjøp og utlån av utstyr.
- Morten Sandseth og Milad Moradi ved Fablab for bistand med 3D-printing.
- Jostein Berge for tips og råd angående materialvalg og ekstern hjelp for plastsveising.
- Ferd Båt A/S for god pris på jolle.
- Steinsvik Group A/S for gratis plastsveising og strålende service.
- Sunnmøre Museum for lån av flytebrygge for testing av prototype.
- Værgudene for strålende vær under testingen.

Sammendrag

Denne rapporten omhandler utvikling av et ubemannet overflatefartøy, som er gitt som en bacheloroppgave av NTNU i Ålesund. Formålet med oppgaven er utvikling av et konsept for et ubemannet overflatefartøy som danner basis for videre utvikling i senere prosjekter. I tillegg skal det utvikles en prototype for testing av mulige løsninger for fremdriftssystem, kontrollsystem, instrumentering og dynamisk posisjonering. Viktige moment ved utvikling og design av ubemannede overflatefartøy er belyst i rapporten, og en prototype basert på en Pioner 8-fots jolle er utviklet, med fokus på funksjonalitet og lav kostnad. Prototypen har fire fastmonterte utenpåliggende thrustere. Dynamisk posisjonering bestående av tre PID-regulatorer er utviklet, og thrusterallokeringsproblemet er løst med kvadratisk programmering. Kontrollsystemet er basert på en Odroid XU4, og programvare er utviklet i Java. Til instrumentering er det benyttet billige komponenter, og Arduinoer for I/O. Resultatene fra testingen viser at det vil være mulig å oppnå dynamisk posisjonering innenfor de gitte kravene med løsningen som presenteres.

Innhold

Terminologi

Begreper

Treghetssystem ..... Et ikke-akselererende koordinatsystem, hvor Newtons lover gjelder

Kinematikk ..... Beskriver et legemes bevegelse uten hensyn til årsaken til bevegelsen

Dynamikk ..... Beskriver et legemes bevegelse ut fra krefter og momenter som virker på legemet

Flattrykkhet ..... Beskriver hvor flattrykt en planet er

Prime Meridian ..... Sirkelen rundt nord- og sørpol som ligger på lengdegrad 0.0°

Geodetiske koordinater . Breddegrad, lengdegrad og høyde

Peiling ..... Retningen til baugen i grader i forhold til den magnetisk nordpol

Knop ..... Nautisk enhet for hastighet (1 knop $\approx$ 0,5144 m/s)

Rang ..... Dimensjonen til vektorrommet utspent av vektorene i en matrise

Identitetsmatrise ..... Matrise med 1 langs hoveddiagonalen, og 0 i alle andre posisjoner

Klient-tjener-prinsippet . Kommunikasjon hvor den ene komponenten etterspør tjenester fra den andre

Three-way-handshake ... Trestegs metode for å etablere TCP-kommunikasjon mellom to maskiner

I/O ..... Input/output, kommunikasjon mellom en datamaskin og utenomverdenen

Seriell kommunikasjon .. Kommunikasjon der data sendes som en strøm av bits over 2 ledere

IPxx ..... Ingress Protection, kapslingsgrad, hvor xx er tall som bestemmer graden

String ..... Klasse som representerer en streng av tegn

char ..... Primitiv datatype for tegn

int ..... Primitiv 32-bits datatype for heltall

float ..... Primitiv 32-bits datatype for desimaltall

double ..... Primitiv 64-bits datatype for desimaltall, dobbel presisjon

bit ..... Enhet for digital informasjon, som kan innta en av to definerte tilstander

byte ..... Gruppe på 8 bit

<CR> ..... Carriage return, returnerer pekeren til starten av linjen

<LF> ..... Line feed, starter en ny linje

Notasjoner

$s\cdot$ ..... Sinusfunksjon, $\sin(\cdot)$

$c\cdot$ ..... Cosinusfunksjon, $\cos(\cdot)$

$\mathbf{v}$ ..... En vektor, v

$\mathbf{T}$ ..... En matrise, T

$\mathbf{u}^T$ ..... Transponerte av vektoren u

$\mathbf{I}_{n\times n}$ ..... Identitetsmatrise med n rader og n kolonner

$\dot{\mathbf{v}}$ ..... Tidsderiverte av vektoren v

$y_{ref}(t)$ ..... Referanse-/Ønsket funksjonsverdi

$y(t)$ ..... Faktisk funksjonsverdi

$y_m(t)$ ..... Målt funksjonsverdi

$e(t)$ ..... Avvik mellom $y_{ref}(t)$ og $y_m(t)$

$u(t)$ ..... Pådrag fra regulator

$x(t)$ ..... Pådrag fra aktuator

$\forall$ ..... For alle

$f(x)$ ..... Funksjon av x

$f(x,y)$ ..... Funksjon av x og y

$\mathbb{R}$ ..... Alle reelle tall

$\in$ ..... Er element i

g ..... Gravitasjonsakselerasjonen, $9,81\frac{m}{s^2}$

$\frac{\partial f}{\partial x}$ ..... Partiellderiverte av f med hensyn på x

Forkortelser

DP ..... Dynamisk Posisjonering

CFD ..... Computational Fluid Dynamics, numerisk fluiddynamikk

GPS ..... Globalt Posisjoneringssystem

DGPS ..... Differensiell GPS

IMU ..... Inertial Measurement Unit

EGNOS ..... European Geostationary Navigation Overlay System

NMEA ..... National Marine Electronics Association

WGS84 ..... World Geodetic System 1984

GPRMC ..... Recommended minimum specific GPS/Transit data

GPGGA ..... Global Positioning System Fix Data

GNC ..... Guidance, Navigation and Control

ROV ..... Remotely Operated Vehicle, fjernstyrt undervannsfarkost

ESC ..... Elektronisk Motorkontroller

DOF ..... Frihetsgrad(er) / Degree(s) of Freedom

SBC ..... Single Board Computer, datamaskin på ett kretskort

OS ..... Operativsystem

TCP ..... Transmission Control Protocol

IP ..... Internet Protocol

USB ..... Universal Serial Bus

DC ..... Direct Current, likestrøm

A ..... Ampere

V ..... Volt

W ..... Watt

Hz ..... Hertz, svingninger per sekund

IDE ..... Utviklingsmiljø / Integrated Development Environment

JVM ..... Java Virtual Machine

PWM ..... Pulse Width Modulation, pulsbreddemodulering

lbf ..... Pound-force, britisk enhet for kraft

Figurer

Tabeller

Kapittel 1

Innledning

1.1 Bakgrunn

Den maritime næringen har alltid stått sterkt i Norge, og spesielt på nordvestlandet. Verftsindustrien sysselsetter svært mange mennesker, og er ofte hjørnesteinsbedrifter i lokalsamfunnene. I tillegg lever mange mennesker av havbruk og oppdrett. Det er et stort fokus på effektivisering og reduksjon av kostnader innen alle næringer, og særlig innen offshore og maritim næring. Robotisering og automatisering kan være en viktig bidragsyter for å holde norsk maritim industri og næringsliv konkurransedyktig inn i fremtiden. Autonome fartøyer og automatisering av maritime operasjoner er et felt med stor forskningsaktivitet, og Norge med NTNU i spissen har gode muligheter for være ledende i utviklingen.

1.2 Introduksjon

NTNU i Ålesund ønsker å utvikle en USV, et ubemannet overflatefartøy som kan fungere som instrument og utstyrsplattform. Et slikt fartøy kan ha mange bruksområder, som for eksempel inspeksjon av oppdrettsanlegg, overvåking av miljø i fjorder, autonom forflytning av utstyr med mer. Fartøyet skal ha elektrisk fremdriftssystem og dynamisk posisjonering. Dette prosjektet tar sikte på utredning av et konsept for et slikt fartøy, som er ment som basis for senere prosjekter ved NTNU i Ålesund. Fokusområder under utredningen er på design av fremdrift- og dynamisk posisjoneringssystem, kraftsystem, kontrollsystem og instrumentering. Under vurdering av ulike løsninger er det lagt vekt på fleksibilitet og lav pris. I tillegg er det utviklet en prototype for uttesting av mulige løsninger for fremdriftssystem, kontrollsystem og dynamisk posisjonering. Oppgaven er gitt av NTNU i Ålesund.

1.3 Problemstilling

Problemstillingen for dette prosjektet kan deles i to deler. Den ene delen går på utredning av et konsept som danner basis for videre utvikling i senere prosjekter. Den andre delen går på bygging av en prototype for uttesting av mulige løsninger for fremdriftssystem, kontrollsystem, instrumentering og dynamisk posisjonering.

Problemstillinger:

- Utredning av konsept for en USV som basis for senere prosjekter.
- Bygge en prototype for uttesting av fremdriftssystem, kontrollsystem og dynamisk posisjonering.

1.4 Kravspesifikasjoner

En utviklet prototype skal tilfredsstille følgende kriterier:

- Prototypen kan ligge i fast posisjon med konstant peiling (innenfor nøyaktigheten til GPS) ved hjelp av dynamisk posisjonering.
- Prototypen kan kjøres manuelt til ny posisjon og legges i ny fast posisjon med konstant peiling.

1.5 Rapportinnhold

Denne rapporten er inndelt som følgende.

Kapittel 2 - Teoretisk Grunnlag: Inneholder en oppsummering av teoretisk bakgrunn nødvendig for vurderinger og valg senere i rapporten.

Kapittel 3 - Materialer og Metode: Inneholder en beskrivelse av materialer og utviklingsmetodikk som er benyttet i prosjektet, og en beskrivelse av prosjektets organisering. I dette kapittelet er det også en utredning av viktige faktorer og moment som må vurderes ved bygging av et ubemannet overflatefartøy. Til slutt i kapittelet kommer en beskrivelse av testoppsettet for prototypen.

Kapittel 4 - Resultater: Inneholder en beskrivelse av den utviklede prototypen, og materialer som inngår i denne. Dette kapittelet inneholder også en beskrivelse av dynamisk posisjonering og thrusterallokering som er utviklet til prototypen, samt en beskrivelse av programvaren som er utviklet. Til slutt presenteres resultater fra testing.

Kapittel 5 - Drøfting: Inneholder en kritisk drøfting av resultatene fra kapittel 4, og en drøfting av hva som skiller den utviklede prototypen fra en fullstendig løsning.

Kapittel 6 - Konklusjon: Trekker en konklusjon over prosjektet i sin helhet, og besvarer problemstillingen.

Kapittel 2

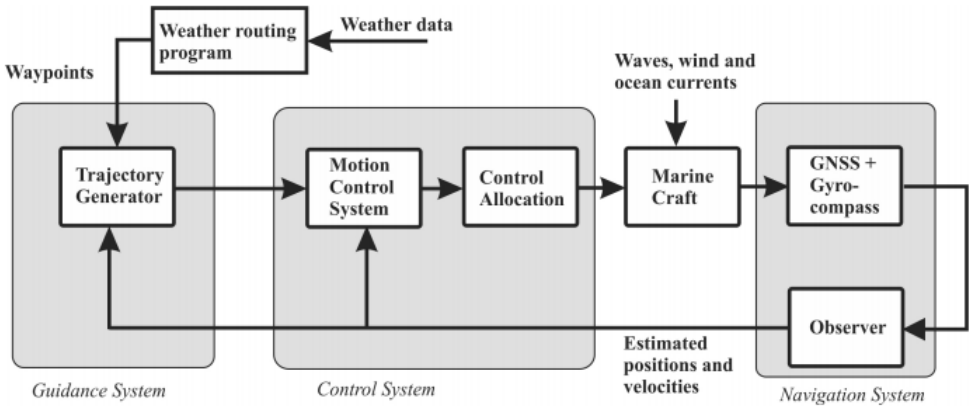
Teoretisk Grunnlag

2.1 Ubemannede Overflatefartøy

Et ubemannet overflatefartøy er et maritimt fartøy som opererer autonomt eller ved fjernstyring uten mannskap. De brukes både militært og i det sivile, til overvåkning og forskning.

2.2 Dynamisk posisjonering

Dynamisk posisjonering (DP) er et konsept brukt i maritim forbindelse. Det lar et fartøy holde en konstant posisjon og peiling ved bruk av egne framdriftssystemer og sensorer. Et robust DP-system, som i figur 2.1 hentet fra [1], benytter en datamaskin som samler informasjon om fartøyets bevegelser fra sensorer om bord. Med disse dataene i kombinasjon med en matematisk modell av fartøyet, som kalles en observer, kan datamaskinen forutse hvilken retning det vil drifte. Kontrollsystemet vil da kunne bestemme hvor stor kraft og hvilken retning hver enkel thruster/propeller må ha for kompensere for de ytre kreftene som påvirker fartøyets posisjon og peiling [2].



Figur 2.1: Kontrollsystem [1]

Figur 2.1 består av tre ledd: *Guidance System*, *Control System* og *Navigation System*. Et DP-system består av de to siste. *Navigation System* består av GPS, IMU og en observer. De bestemmer posisjon og peiling til fartøyet og oppdaterer den matematiske modellen. *Control System* bestemmer de nødvendige kontrollkreftene

og -momentene, ved hjelp av output fra *Navigation System*-blokken, for å tilfredsstille et gitt kontrolloppdrag. Eksempler på kontrolloppdrag er bl.a:

- Minumum energi
- Settpunkt regulering
- Baneføլging

Dynamisk posisjonering kategoriseres etter disse klassene [3]:

- Klasse 1 - Tap av posisjon ved en enkelt feil på aktiv komponent
  Ingen redundans
- Klasse 2 - Tap av posisjone skjer IKKE ved en enkelt feil på aktiv komponent
  Oppnådd gjennom redundans
- Klasse 3 - Klasse 2 + inkluderer bl.a druknede/brente avdelinger
  Oppnådd gjennom redundans og inndeling av avdelinger

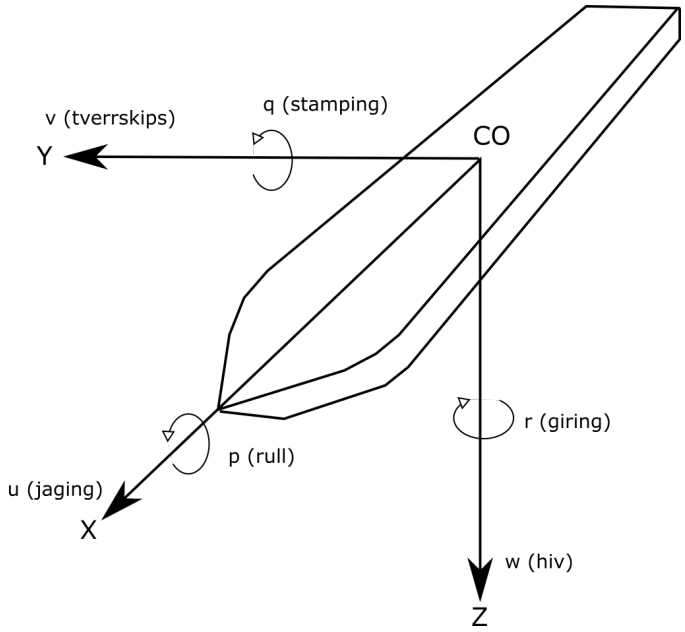
Sammenligner man DP med tradisjonell ankring gir det flere fordeler. Et DP-system kan benyttes der ankring ikke er mulig, f.eks i dypt hav eller områder der strømkabler og rør dekker havbunnen. Det er også enkelt å manøvrere og raskt å starte opp. Baksiden er kompleksiteten og oppstartskostnaden. Det kreves ekstra thrustere og generatorer, samt en robust regulator og vedlikehold av det mekaniske utstyret. Det er også fare for såkalte *blackouter*, der skipet er uten strøm. Utviklingen av DP startet tidlig på 1960-tallet og det første norske DP-system ble montert på *Seaway Eagle* i 1977 [4].

2.3 Eulervinkler

Orienteringen til et legeme i det tredimensjonale rommet defineres med tre rotasjoner, vanligvis en rotasjon om hver av aksene i koordinatsystemet som benyttes som referanse. For et skip er rotasjon om x-aksen parametrisert med ϕ, rotasjon om y-aksen er parametrisert med θ, og rotasjon om z-aksen parametriseres med ψ. Enhver orientering av et legeme i rommet kan dermed parametriseres med disse tre vinklene [5]. Disse vinklene kalles Eulervinkler, etter den sveitsiske matematikeren Leonhard Euler.

2.4 Bevegelsesvariabler

Et skip som beveger seg i 6 frihetsgrader trenger 6 uavhengige variabler for å bestemme posisjon og orientering. De første tre koordinatene, og deres tidsderiverte, beskriver posisjonen og translasjon langs x-, y- og z-aksene, mens de tre siste koordinatene, og deres tidsderiverte, beskriver orientering og roterende bevegelse. For marine fartøy er bevegelseskoordinatene definert som jaging, tverrskips, hiv, rull, stamping og giring. Dette er illustrert i figur 2.2.



Figur 2.2: Koordinatakser og bevegelsesvariabler.

I denne rapporten vil det bli benyttet SNAME-notasjon (Society of Naval Architects and Marine Engineers) beskrevet i [6], og gjentatt i tabell 2.1.

Frihetsgrad	Bevegelse	Krefter og Momenter	Lineær/vinkelhastighet	Posisjon/eulervinkler
1	Translasjon langs x-aksen (jaging)	X	u	x
2	Translasjon langs y-aksen (tverrskips)	Y	v	y
3	Translasjon langs z-aksen (hiving)	Z	w	z
4	Rotasjon om x-aksen (rulling)	K	p	ϕ
5	Rotasjon om y-aksen (stamping)	M	q	θ
6	Rotasjon om z-aksen (giring)	N	r	ψ

Tabell 2.1: SNAME-notasjon.

2.5 Referansesystemer

For gaidings- og navigasjonsformål er det hensiktsmessig å definere flere ulike koordinatsystemer.

2.5.1 Jord-sentrerte koordinatsystemer

ECI: Earth-centered inertial frame, jord-sentrert treghetsramme $\{i\} = \{x_i, y_i, z_i\}$ er et referansesystem med origo i senter av jorda, og som *ikke* roterer med jordkloden. Det er et ikke-akselererende referansesystem, slik at Newtons bevegelseslover gjelder.

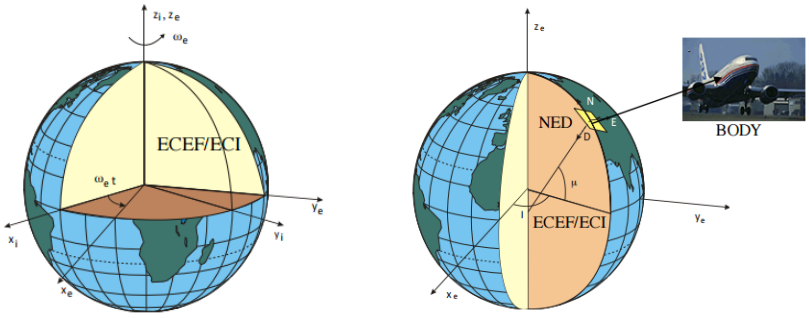
ECEF: Earth-centered earth fixed reference frame, jord-sentrert jordfast koordinatsystem $\{e\} = \{x_e, y_e, z_e\}$ har i likhet med ECI origo i senter av jorda, men systemet roterer relativt til aksene i ECI, med samme vinkelhastighet som jorda. Vinkelhastigheten er $\omega_e = 7,2921 \cdot 10^{-5}$ rad/s, og for marine fartøy som beveger seg i lav hastighet kan jordrotasjonen ignoreres slik at $\{e\}$ kan betraktes som et treghetssystem [7].

2.5.2 Geografiske koordinatsystemer

NED: NED står for *North-East-Down*, Nord-Øst-Ned, og er et koordinatsystem $\{n\} = \{x_n, y_n, z_n\}$ hvor x- og y-aksen danner et tangentplan på jordoverflaten som beveger seg med fartøyet. X-aksen peker alltid mot nord, og y-aksen peker alltid mot øst. Z-aksen peker nedover normal på jordoverflaten. Origo ligger normalt i senter på fartøyet. Posisjonen til koordinatsystemet $\{n\}$ i forhold til $\{e\}$ bestemmes av to vinkler l og μ , som er henholdsvis lengde- og breddegrad. For fartøy som opererer i et lokalt avgrenset område med tilnærmet konstant lengde- og breddegrad, kan man anta at $\{n\}$ er et treghetssystem, slik at Newtons lover gjelder [7].

BODY: Body-koordinatsystemet $\{b\} = \{x_b, y_b, z_b\}$ er et koordinatsystem hvor aksene følger fartøyet, som illustrert i figur 2.2. Posisjonen og orienteringen av fartøyet beskrives relativt til treghetssystemet som benyttes som referanse, vanligvis $\{n\}$ for lokaliserte navigasjonsformål, mens lineære hastigheter og vinkelhastigheter beskrives i $\{b\}$. Origo O_b velges slik at det sammenfaller med et punkt midtskips i vannlinjen. Dette punktet blir referert til som CO [7].

Koordinatsystemene er illustrert i figur 2.3, som er hentet fra [1].



(a) ECEF og ECI koordinatsystemer (b) NED og BODY koordinatsystemer

Figur 2.3: ECEF, ECI, NED og BODY koordinatsystemer

2.5.3 Konvertering av geodetiske koordinater til NED koordinater

Utledningen fra geodetiske koordinater starter med å finne små endringer i breddegrad $d\mu$ og lengdegrad dl ved å ta differansene mellom ny verdi μ og l, og referanseverdier μ_0 og l_0.

$$d\mu = \mu - \mu_0 \tag{2.1}$$

$$dl = l - l_0 \tag{2.2}$$

For å konvertere til North-East koordinater brukes krumningsradien til "prime vertical" (R_N) og "prime meridian" (R_M). R_N og R_M er definert som følgende [8]:

$$R_N = \frac{R}{\sqrt{1 - (2f - f^2)\sin^2(\mu_0)}} \tag{2.3}$$

$$R_M = R_N \frac{1 - (2f - f^2)}{1 - (2f - f^2)\sin^2(\mu_0)} \tag{2.4}$$

der R er jordens radius ved ekvator og f er jordens flattrykhet. Små endringer i North (dN) og East (dE) er beregnet utifra små endringer i nord og øst posisjon ved:

$$dN = \frac{d\mu}{\arctan(\frac{1}{R_M})} \tag{2.5}$$

$$dE = \frac{dl}{\arctan(\frac{1}{R_N \cos\mu_0})} \tag{2.6}$$

2.6 Kinematikk

2.6.1 Vektornotasjon

Ved å definere følgende vektornotasjon, hentet fra [7]

$\mathbf{v}^e_{b/n}$ = lineær hastighet til punktet O_b med hensyn til $\{n\}$ uttrykt i $\{e\}$
$\boldsymbol{\omega}^b_{n/e}$ = vinkelhastigheten til $\{n\}$ med hensyn til $\{e\}$ uttrykt i $\{b\}$
$\mathbf{f}^b_b$ = kraft gjennom O_b uttrykt i $\{n\}$
$\mathbf{m}^b_b$ = moment om punktet O_b uttrykt i $\{n\}$
$\boldsymbol{\Theta}_{nb}$ = Eulervinkler mellom $\{n\}$ og $\{b\}$
$\mathbf{p}^n_{b/n}$ = Posisjon til punktet O_b med hensyn til $\{n\}$ uttrykt i $\{n\}$

så kan variablene i tabell 2.1 uttrykkes i en vektoriell setting som følgende

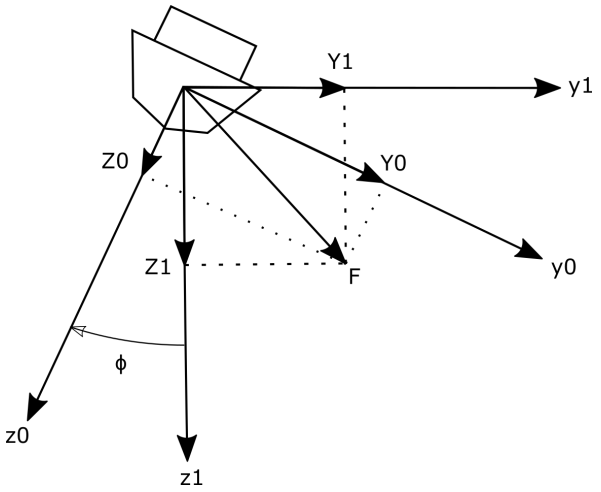
$$\boldsymbol{\eta} = \begin{bmatrix} \mathbf{p}^n_{b/n} \\ \boldsymbol{\Theta}_{nb} \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ \phi \\ \theta \\ \psi \end{bmatrix}, \boldsymbol{\nu} = \begin{bmatrix} \mathbf{v}^b_{b/n} \\ \boldsymbol{\omega}^b_{b/n} \end{bmatrix} = \begin{bmatrix} u \\ v \\ w \\ p \\ q \\ r \end{bmatrix} \text{ og } \boldsymbol{\tau} = \begin{bmatrix} \mathbf{f}^b_b \\ \mathbf{m}^b_b \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \\ K \\ M \\ N \end{bmatrix}.$$

2.6.2 Transformasjon mellom BODY og NED

Eulervinklene om x- (ϕ), y- (θ) og z-aksen (ψ) kan brukes for å dekomponere BODY-vektorer (f. eks. kraft, hastighet) i NED-koordinatsystemet. $\mathbf{R}^n_b(\boldsymbol{\Theta}_{nb})$ er Eulervinkel rotasjonsmatrisen fra $\{b\}$ til $\{n\}$. Da kan man finne kraftvektoren som virker på skipet i NED-koordinater ved å premultiplisere kraftvektoren eller hastighetsvektoren i BODY-koordinater med rotasjonsmatrisen, $\mathbf{f}^n_b = \mathbf{R}^n_b(\boldsymbol{\Theta}_{nb})\mathbf{f}^b_b$ [7].

2.6.3 Hovedrotasjoner

Rotasjonsmatrisen $\mathbf{R}^n_b(\boldsymbol{\Theta}_{nb})$ kan finnes ved å gjennomføre tre rotasjoner, en om hver akse. Dette gir tre hovedrotasjonsmatriser (rotasjon om en akse), som til slutt kan samles i en rotasjonsmatrise.
Rotasjon om x-aksen (rull) er illustrert i figur 2.4.



Figur 2.4: Rotasjon om x-aksen

Fra figur 2.4 ser man at

$$X_1 = X_0 \tag{2.7}$$

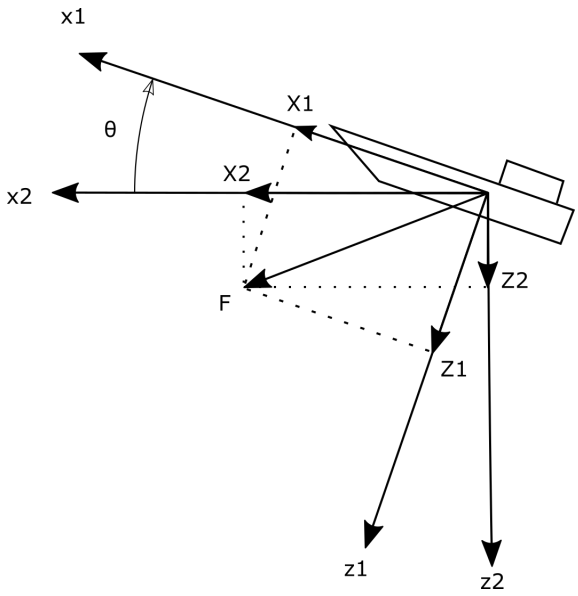
$$Y_1 = \cos(\phi) \cdot Y_0 - \sin(\phi) \cdot Z_0 \tag{2.8}$$

$$Z_1 = \sin(\phi) \cdot Y_0 + \cos(\phi) \cdot Z_0 \tag{2.9}$$

Dette resultatet kan samles i en matrise slik at

$$\begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\phi & -s\phi \\ 0 & s\phi & c\phi \end{bmatrix} \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \end{bmatrix} = \mathbf{R}_{x,\phi}\mathbf{f}_b^b \tag{2.10}$$

Rotasjon om y-aksen er illustrert i figur 2.5.



Figur 2.5: Rotasjon om y-aksen

En rotasjon om y-aksen gir følgende sammenheng mellom BODY- og NED-koordinatene

$$X_2 = \cos(\theta) \cdot X_1 + \sin(\theta) \cdot Z_1 \tag{2.11}$$

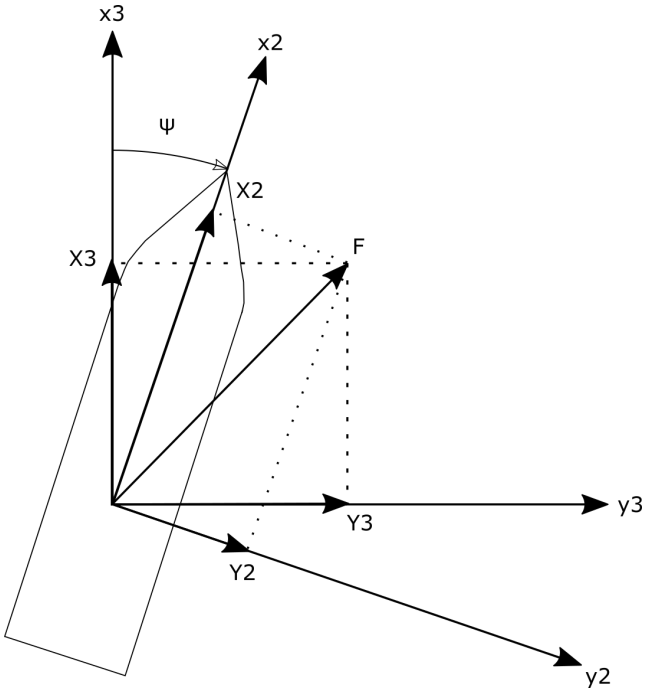
$$Y_2 = Y_1 \tag{2.12}$$

$$Z_2 = -\sin(\theta) \cdot X_1 + \cos(\theta) \cdot Z_1 \tag{2.13}$$

Samlet i en matrise blir dette

$$\begin{bmatrix} X_2 \\ Y_2 \\ Z_2 \end{bmatrix} = \begin{bmatrix} c\theta & 0 & s\theta \\ 0 & 1 & 0 \\ -s\theta & 0 & c\theta \end{bmatrix} \begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \end{bmatrix} = \mathbf{R}_{y,\theta}\mathbf{R}_{x,\phi}\mathbf{f}_b^b \tag{2.14}$$

Rotasjon om z-aksen er illustrert i figur 2.6.



Figur 2.6: Rotasjon om z-aksen

Denne rotasjonen gir følgende sammenheng mellom BODY- og NED-koordinater

$$X_3 = \cos(\psi) \cdot X_2 - \sin(\psi) \cdot Y_2 \tag{2.15}$$

$$Y_3 = \sin(\psi) \cdot X_2 + \cos(\psi) \cdot Y_2 \tag{2.16}$$

$$Z_3 = Z_2 \tag{2.17}$$

På matriseform blir dette

$$\begin{bmatrix} X_3 \\ Y_3 \\ Z_3 \end{bmatrix} = \begin{bmatrix} c\psi & -s\psi & 0 \\ s\psi & c\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_2 \\ Y_2 \\ Z_2 \end{bmatrix} = \mathbf{R}_{z,\psi}\mathbf{R}_{y,\theta}\mathbf{R}_{x,\phi}\mathbf{f}_b^b \tag{2.18}$$

Rotasjonsmatrisen som går fra BODY-koordinater til NED-koordinater er dermed gitt av matriseproduktet

$$\mathbf{R}_b^n(\mathbf{\Theta}_{nb}) = \mathbf{R}_{z,\psi}\mathbf{R}_{y,\theta}\mathbf{R}_{x,\phi} = \begin{bmatrix} c\psi c\theta & -s\psi c\theta + c\psi s\theta s\phi & s\psi s\phi + c\psi c\phi s\theta \\ s\psi c\theta & c\psi c\phi + s\phi s\theta s\psi & -c\psi s\phi + s\theta s\psi c\phi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{bmatrix} \tag{2.19}$$

En rotasjonsmatrise, $\mathbf{R} \in \mathbb{R}^{3\times3}$ har den egenskapen at $\mathbf{R}\mathbf{R}^T = \mathbf{R}^T\mathbf{R} = \mathbf{I}$ og det $\mathbf{R} = 1$ [7]. Dette betyr at transformasjonen fra NED-koordinater til BODY-koordinater er gitt ved

$$\mathbf{R}_n^b(\mathbf{\Theta}_{nb}) = \mathbf{R}_b^n(\mathbf{\Theta}_{nb})^{-1} = \mathbf{R}_b^n(\mathbf{\Theta}_{nb})^T = \mathbf{R}_{x,\phi}^T\mathbf{R}_{y,\theta}^T\mathbf{R}_{z,\psi}^T \tag{2.20}$$

Et fartøys hastighet i BODY-koordinater kan nå representeres i NED-koordinater gjennom transformasjonen

$$\dot{\mathbf{p}}^n_{b/n} = \mathbf{R}^n_b(\mathbf{\Theta}_{nb})\mathbf{v}^b_{b/n} \tag{2.21}$$

hvor $\mathbf{p}^n_{b/n}$ er fartøyets posisjonsvektor i NED-koordinater. Den inverse transformasjonen er gitt ved

$$\mathbf{v}^b_{b/n} = \mathbf{R}^n_b(\mathbf{\Theta}_{nb})^T\dot{\mathbf{p}}^n_{b/n} = \mathbf{R}^b_n(\mathbf{\Theta}_{nb})\dot{\mathbf{p}}^n_{b/n} \tag{2.22}$$

2.6.4 Kinematikk i tre frihetsgrader

For overflateskip er det tilstrekkelig å beskrive bevegelsene til skipet i tre frihetsgrader (jaging, sideveis og giring). Dette gjør at kinematikken kan forenkles. Forenklingene er basert på antagelsen at ϕ og θ er små [7]. Da blir rotasjonsmatrisen

$$\mathbf{R}^n_b(\mathbf{\Theta}_{nb}) = \mathbf{R}_{z,\psi}\mathbf{R}_{y,\theta}\mathbf{R}_{x,\phi} \approx \mathbf{R}_{z,\psi} \tag{2.23}$$

Siden ϕ og θ antas å være små, kan man også anta at $\dot{\psi} = r$ (giring). Ved å overse elementene som korresponderer til hiv, rull og stamping, får vi kinematikklikningen $\dot{\boldsymbol{\eta}} = \mathbf{R}(\psi)\mathbf{v}$, hvor $\mathbf{R}(\psi) := \mathbf{R}_{z,\psi}$ og $\mathbf{v} = \begin{bmatrix} u, & v, & r \end{bmatrix}^T$ og $\boldsymbol{\eta} = \begin{bmatrix} N, & E, & \psi \end{bmatrix}^T$.

Det vil si

$$\begin{bmatrix} \dot{N} \\ \dot{E} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} c\psi & -s\psi & 0 \\ s\psi & c\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ r \end{bmatrix} \tag{2.24}$$

2.7 Dynamisk modell av fartøy

Matematiske modeller for marine fartøy finnes i flere variasjoner, med varierende kompleksitet [7]. Modeller for marine fartøyer er ulineære på grunn av at de hydrodynamiske kreftene er en funksjon av hastighet, corioliseffekten, rotasjon mellom BODY-koordinater og globale koordinater, samt oppdriftskrefter. Den mest komplekse modellen i 6 frihetsgrader på vektoriell form er uttrykt i likn. (2.25) og likn. (2.26), med samme vektornotasjon som i kapittel 2.6.1. $\boldsymbol{\nu}_r$ er fartøyets hastighetsvektor relativt til strømmingen i vannet, $\boldsymbol{\nu}_r = \boldsymbol{\nu} - \boldsymbol{\nu}_c$ hvor $\boldsymbol{\nu}_c$ er hastigheten til vannstrømmen. En komplett utledning av disse likningene kan finnes i læreboken *Handbook of Marine Craft Hydrodynamics and Motion Control* [7].

$$\dot{\boldsymbol{\eta}} = \mathbf{J}_\Theta(\boldsymbol{\eta})\boldsymbol{\nu} \tag{2.25}$$

$$\underbrace{\mathbf{M}_{RB}\dot{\boldsymbol{\nu}} + \mathbf{C}_{RB}(\boldsymbol{\nu})\boldsymbol{\nu}}_{stivt-legeme\ krefter} + \underbrace{\mathbf{M}_A\dot{\boldsymbol{\nu}}_r + \mathbf{C}_A(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r + \mathbf{D}(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r}_{hydrodynamiske\ krefter} + \underbrace{\mathbf{g}(\boldsymbol{\eta}) + \mathbf{g}_0}_{hydrostatiske\ krefter} = \boldsymbol{\tau} + \boldsymbol{\tau}_{vind} + \boldsymbol{\tau}_{bølge} \tag{2.26}$$

På grunn av kompleksiteten i disse likningene, benyttes de i hovedsak til simulering av systemets dynamikk. En forenkling av denne modellen er nødvendig for kontrollsystemdesign [7].

2.7.1 Stivt-legeme krefter

For DP-applikasjoner er kun tre av frihetsgradene av interesse (jaging, sideveis og giring), siden disse tre frihetsgradene kan kontrolleres ved hjelp av thrustere. Første del av likn. (2.26) beskriver stivt-legeme kreftene som virker på fartøyet, utledet fra Newtons lover.

$$\mathbf{M}_{RB}\dot{\boldsymbol{\nu}} + \mathbf{C}_{RB}(\boldsymbol{\nu})\boldsymbol{\nu} = \boldsymbol{\tau}_{RB} \tag{2.27}$$

$\mathbf{M}_{RB}$ er stivt-legeme massematrisen, og $\mathbf{C}_{RB}(\boldsymbol{\nu})$ er coriolis-sentripetalmatrisen som kommer av at BODY-koordinatsystemet roterer i NED-koordinatsystemet. Vektorene $\boldsymbol{\nu} = \begin{bmatrix} u & v & r \end{bmatrix}^T$ og $\boldsymbol{\tau}_{RB} = \begin{bmatrix} X & Y & N \end{bmatrix}^T$ er representert i BODY-koordinater. Matrisen $\mathbf{M}_{RB}$ er definert i likning likn. (2.28).

$$\mathbf{M}_{RB} = \begin{bmatrix} m & 0 & -my_g \\ 0 & m & mx_g \\ -my_g & mx_g & I_z \end{bmatrix} \tag{2.28}$$

hvor m er fartøyets masse, x_g og y_g er avstanden fra O_b til massesenteret i henholdsvis x- og y-retning. I_z er treghetsmomentet til fartøyet om z-aksen, og kan beregnes med metoder beskrevet i [9]. I praksis vil massesenteret ligge på x-aksen slik at $y_g = 0$. Da kan matrisen $\mathbf{M}_{RB}$ skrives som vist i likn. (2.29).

$$\mathbf{M}_{RB} = \begin{bmatrix} m & 0 & 0 \\ 0 & m & mx_g \\ 0 & mx_g & I_z \end{bmatrix} \tag{2.29}$$

Matrisen $\mathbf{C}_{RB}(\boldsymbol{\nu})$, utledet i [7], kan skrives som 2.30.

$$\mathbf{C}_{RB}(\boldsymbol{\nu}) = \begin{bmatrix} 0 & 0 & -m(x_g r + v) \\ 0 & 0 & mu \\ m(x_g r + v) & -mu & 0 \end{bmatrix} \tag{2.30}$$

Ut fra dette ser man at bevegelse langs x-aksen (jaging) er dekoplet og uavhengig av giring og sideveis bevegelse.

2.7.2 Hydrodynamiske krefter

De hydrodynamiske kreftene i likn. (2.26) stammer fra tillagt masse, coriolis-sentripetalkrefter på grunn av rotasjon av BODY-koordinatsystemet relativt til NED-koordinatene, samt viskøs demping. Et fartøy i bevegelse vil føre til bevegelse i vannet rundt fartøyet, siden vannet må flytte seg unna foran fartsretningen, og samle seg igjen bak fartsretningen. Dette tilfører vannet kinetisk energi, som fører til et ekstra masse-ledd og et ekstra coriolis-sentripetalledd i bevegelseslikningene.

$$\mathbf{M}_A\dot{\boldsymbol{\nu}}_r + \mathbf{C}_A(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r + \mathbf{D}(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r = \boldsymbol{\tau}_{hyd} \tag{2.31}$$

Matrisen $\mathbf{M_A}$, med antagelsen at fartøyet har xz-plan symmetri som gjelder for de fleste overflatefartøy, skrives som i likning likn. (2.32).

$$\mathbf{M}_A = -\begin{bmatrix} X_{\dot{u}} & 0 & 0 \\ 0 & Y_{\dot{v}} & Y_{\dot{r}} \\ 0 & Y_{\dot{r}} & N_{\dot{r}} \end{bmatrix} \tag{2.32}$$

Den hydrodynamiske tillagt-massekraften Y langs y-aksen grunnet en akselerasjon $\dot{v}$ i Y-retningen skrives $Y = -Y_{\dot{v}}v$, $Y_{\dot{v}} := \frac{\partial Y}{\partial \dot{v}}$. Koeffisientene $X_{\dot{u}}$, $Y_{\dot{v}}$, ..., $N_{\dot{r}}$ finnes gjennom testing og systemidentifikasjon. En metodikk for å finne disse koeffisientene er beskrevet i [10]. Matrisen $\mathbf{C}_A(\boldsymbol{\nu}_r)$ kan skrives som vist i likn. (2.33) [7].

$$\mathbf{C}_A(\boldsymbol{\nu}_r) = \begin{bmatrix} 0 & 0 & Y_{\dot{v}}v_r + Y_{\dot{r}}r \\ 0 & 0 & -X_{\dot{u}}u_r \\ -Y_{\dot{v}}v_r & X_{\dot{u}}u_r & 0 \end{bmatrix} \tag{2.33}$$

Hydrodynamisk demping forårsakes av potensiell demping, friksjon, bølgedriftdemping, demping pga. hvirvelstrømmer og hydrodynamisk løft [7]. Alle disse effektene bidrar med både lineære og kvadratiske ledd. Det er vanlig å dele opp total hydrodynamisk demping som $\mathbf{D}(\boldsymbol{\nu}_r) = \mathbf{D} + \mathbf{D}_n(\boldsymbol{\nu}_r)$ hvor $\mathbf{D}$ er lineær demping og $\mathbf{D}_n(\boldsymbol{\nu}_r)$ er ulineær demping. Matrisen $\mathbf{D}$ er utskrevet i likn. (2.34).

$$\mathbf{D} = -\begin{bmatrix} X_u & 0 & 0 \\ 0 & Y_v & Y_r \\ 0 & N_v & N_r \end{bmatrix} \tag{2.34}$$

Kraften er nå avhengig av hastighet og ikke akselerasjon. Ulineær demping kan uttrykkes som i likn. (2.35).

$$\mathbf{D}_n(\boldsymbol{\nu}_r) = -\begin{bmatrix} X_{|u|u}|u_r| & 0 & 0 \\ 0 & Y_{|v|v}|v_r| + Y_{|r|v}|r| & Y_{|v|r}|v_r| + Y_{|r|r}|r| \\ 0 & N_{|v|v}|v_r| + N_{|r|v} & N_{|v|r}|v_r| + N_{|r|r}|r| \end{bmatrix} \tag{2.35}$$

Alle parametere i denne seksjonen finnes normalt ved systemidentifikasjon, hvor spesifikke manøvere gjennomføres. Dette er beskrevet i [10]. Vektorene $\mathbf{g}(\boldsymbol{\eta})$ og $\mathbf{g}_0$ kan ignoreres for 3 frihetsgrader [7].

2.7.3 Lineær DP-modell i 3 frihetsgrader

En linearisert modell av fartøyet i tre frihetsgrader kan utledes fra den ulineære modellen. Den lineariserte modellen er basert på noen antagelser:

- Hastigheten er lav (< 2 m/s)
- Fartøyet er symmetrisk om xz-planet
- ϕ og θ er små

Siden antakelsen er at hastigheten er lav, kan de ulineære matrisene $\mathbf{C}_{RB}(\boldsymbol{\nu})$, $\mathbf{C}_A(\boldsymbol{\nu})$ og $\mathbf{D}_n(\boldsymbol{\nu})$ lineariseres rundt $\boldsymbol{\nu} = \mathbf{0}$ og $\phi = \theta = 0$. Siden $\mathbf{C}_{RB}(\mathbf{0}) = \mathbf{0}$, $\mathbf{C}_A(\mathbf{0}) = \mathbf{0}$ og $\mathbf{D}_n(\mathbf{0}) = \mathbf{0}$ så elimineres de fra likningene. Den relative hastighetsvektoren $\boldsymbol{\nu}_r$ kan fjernes hvis havstrømmingen kompenseres med integralvirkning i regulatoren. En måte å gjøre dette på er å behandle havstrømmer som en sakte varierende bias-vektor $\mathbf{b}$ uttrykt i {n} [7]. Den lineære modellen uttrykkes vanligvis i fartøyparallelle koordinater, hvor NED-koordinatsystemet roteres slik at N er parallell med x-aksen i BODY-koordinater og E er parallell med y-aksen i BODY-koordinater. Dette gjøres med transformasjonen $\boldsymbol{\eta}_p = \mathbf{R}^T(\psi)\boldsymbol{\eta}$. Den lineariserte modellen kan dermed uttrykkes som i likn. (2.36), likn. (2.37) og likn. (2.38).

$$\dot{\boldsymbol{\eta}}_p = \boldsymbol{\nu} \tag{2.36}$$

$$(\mathbf{M}_{RB} + \mathbf{M}_A)\dot{\boldsymbol{\nu}} + \mathbf{D}\boldsymbol{\nu} = \mathbf{R}^T(\psi)\mathbf{b} + \boldsymbol{\tau} + \boldsymbol{\tau}_{\text{vind}} + \boldsymbol{\tau}_{\text{bølge}} \tag{2.37}$$

$$\dot{\mathbf{b}} = \mathbf{0} \tag{2.38}$$

hvor

$$\boldsymbol{\tau} = \mathbf{T}\mathbf{u} \tag{2.39}$$

Denne modellen er egnet for kontrolldesign, hvor tilbakekopling kompenserer for feil som skyldes modellusikkerhet. Matrisen $\mathbf{T}$ beskriver thrusterkonfigurasjonen, og $\mathbf{u}$ er kontrollinputvektoren. Posisjonsreferansesignalene $\boldsymbol{\eta}$ transformeres til fartøyparallelle koordinater i hvert tidsskritt [7].

2.8 Matematisk optimalisering

Et matematisk optimaliseringsproblem er et problem på den generelle formen

$$\begin{aligned} &\text{minimer} && f_0(x) \\ &\text{med bibetingelse} && f_i(x) \leq b_i, \quad i = 1, \ldots, m \end{aligned} \tag{2.40}$$

hvor $x = (x_1, x_2, \ldots, x_n)$ er variabelen som skal optimeres, $f_0 : \mathbb{R}^n \to \mathbb{R}$ er objektivfunksjonen, $f_i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, \ldots, m$ er ulikhetsbegrensningsfunksjonene, og konstantene $b_1, \ldots, b_m$ er grensene for begrensningsfunksjonene.

Matematisk optimalisering kan benyttes til å løse mange praktiske problemer. For eksempel innen aksjeinvestering kan elementet x_i være beløpet investert i aksje nummer i, og funksjonen f_0 beskriver samlet risiko for hele investeringen. Ulikhetsbegrensningen kan være begrensing i budsjett, eller et minimumskrav til avkastning for hele investeringen. Optimaliseringen består da i å velge investeringsbeløp $x_1, \ldots, x_n$ slik at risiko ved investeringen blir minimert [11].

2.8.1 Konveks optimalisering

Et konvekst optimaliseringsproblem er et problem der objektivfunksjonen og begrensingsfunksjonene tilfredsstiller ulikheten

$$f_i(\alpha x + \beta y) \leq \alpha f_i(x) + \beta f_i(y) \;\forall\; x, y \in \mathbb{R}^n \text{ og } \forall\; \alpha, \beta \in \mathbb{R} \text{ med } \alpha + \beta = 1,\ \alpha \geq 0,\ \beta \geq 0 \tag{2.41}$$

Optimaliseringsproblemet sies da å være konvekst. Det fins ikke en generell analytisk formel for å løse konvekse optimaliseringsproblemer, men det fins effektive algoritmer som kan løse problemene [11].

En spesiell form av konveks optimalisering som er relevant for dette prosjektet er problemet på formen

$$\begin{aligned} \text{minimer} \quad & f_0(x) \\ \text{med bibetingelser} \quad & f_i(x) \leq b_i, \quad i = 1, \ldots, m \\ & Ax = b \end{aligned} \tag{2.42}$$

hvor A er en matrise $A \in \mathbb{R}^{p \times n}$, og med rang $p < n$. Et slikt problem kalles et kvadratisk programmeringsproblem, og kan løses med indrepunkts-algoritmer, som er beskrevet i [11].

2.8.2 Lagranges multiplikatormetode

Lagranges multiplikatormetode kan benyttes til å finne løsninger på problemer av formen

$$\begin{aligned} \text{minimer} \quad & f(x, y) \\ \text{med bibetingelse} \quad & g(x, y) = 0 \end{aligned} \tag{2.43}$$

Hvis både f og g oppfyller visse betingelser, beskrevet i [12], så fins det et tall λ_0 slik at (x_o, y_o, λ_0) er et kritisk punkt til Lagrange funksjonen

$$L(x, y, \lambda) = f(x, y) + \lambda g(x, y) \tag{2.44}$$

Med andre ord, så kan minimumsverdier til funksjonen $f(x, y)$ finnes ved å finne ekstremalpunkt til Lagrange-funksjonen. Dette gjøres ved å sette de partielt deriverte til Lagrange-funksjonen lik null, og løse for x og y [12].

$$\begin{aligned} \frac{\partial L}{\partial x} &= 0 \\ \frac{\partial L}{\partial y} &= 0 \\ \frac{\partial L}{\partial \lambda} &= g(x, y) = 0 \end{aligned} \tag{2.45}$$

2.9 Froudes tall

Froudes tall benyttes for å klassifisere fartøy basert på deres maksimale hastighet. Tallet er definert som

$$F_n := \frac{U}{\sqrt{gL}} \tag{2.46}$$

hvor U er fartøyets hastighet, g er gravitasjonsakselerasjonen og L er fartøyets lengde under vannflaten. Basert på fartøyets Froude-tall, kan følgende man gjøre følgende klassifiseringer [13]:

Når $F_n < 0.4$ så dominerer oppdriftskrefter over hydrodynamiske krefter. Fartøyer med $F_n < 0.4$ kalles deplasementfartøy.

Når $0.4 - 0.5 < F_n < 1.0 - 1.2$ så er ikke oppdriftskreftene dominante over de hydrodynamiske kreftene ved maksimal hastighet. Slike fartøy kalles semi-deplasementfartøy.

Når $F_n > 1.0 - 1.2$ så dominerer de hydrodynamiske kreftene over oppdriftskraften ved maksimal hastighet. Vekten av fartøyet bæres da av de hydrodynamiske kreftene. Slike fartøy kalles planende fartøy.

2.10 Vannmotstand mot skrog

Et fartøys motstand gjennom vannet består av flere deler. Et skip som beveger seg gjennom vannet vil generere et bølgemønster som beveger seg bort fra skroget, samt at det bygger seg opp en region med turbulent strømning langs skroget og som strekker seg ut som kjølvann bak skroget. Begge disse egenskapene ved vannstrømmingen absorberer energi fra skroget mens det beveger seg gjennom vannet, og vil med det yte motstand mot skipets bevegelse. Motstandskraften forplanter seg i skroget som en fordeling av trykk og skjærkrefter over skroget. Skjærkreftene oppstår på grunn av viskositeten til vannet. Motstanden kan dermed brytes ned til kreftene på følgende måte [14]:

Figur 2.7: Krefter som virker på et skip i bevegelse

1. Friksjonsmotstand.
   Alle tangentielle skjærkrefter, τ på figur 2.7, kan summeres over hele skroget for å finne total friksjonsmotstand.

2. Trykkmotstand.
   Trykkmotstanden, P på figur 2.7, kan summeres over hele skroget for å finne total trykkmotstand. Friksjonsmotstanden oppstår kun på grunn av vannets viskositet, mens trykkmotstanden oppstår delvis på grunn av viskositeten, og delvis på grunn av at skroget danner bølger.

Det er også vanlig å dele motstanden inn i total viskøs motstand og total bølgemotstand.

Total viskøs motstand: Bernoullis teorem sier at $\frac{P}{\rho g} + \frac{V^2}{2g} + h = K$ og i fravær av viskøse krefter og med konstant væsketetthet og stasjonær strømning er K konstant [15]. Siden tap av trykk i den strømmende væsken skyldes viskøse krefter, vil man kunne måle den totale viskøse motstanden ved å måle trykkfallet i den strømmende væsken i kjølvannet til skipet.

Total bølgemotstand: Bølgemønsteret som skroget genererer kan måles og brytes ned til komponentbølgene det består av. Energien som kreves for å drive hver enkelt komponent kan beregnes, og dermed har man et mål for den totale bølgemotstanden.

2.11 Linux

Linux er et gratis, åpen-kildekode operativsystem som i starten ble utviklet av en finsk student ved navn Linus Thorvalds på begynnelsen av 1990-tallet. Operativsystemet vokste i popularitet i takt med veksten i funksjonalitet, og etterhvert tiltrakk prosjektet seg mange folk som bidro til utviklingen av systemet, under Thorvalds tilsyn. Linux har vært i kontinuerlig utvikling siden unnfangelsen, og finnes i dag i mange

forskjellige varianter. Linux (og varianter av Linux) er det dominerende operativsystemet på høyytelses arbeidsmaskiner og servere, men har også mange andre bruksområder. Android operativsystemet til smarttelefoner er for eksempel basert på Linux [16].

2.12 Programmering i Java

Java er et objekt-orientert programmeringsspråk utviklet av James Gosling ved Sun Microsystems, som nå er Oracle. Et objekt-orientert kodespråk består av *klasser* som inneholder data. Disse dataene er karakteristiske til hva klassen representerer. F.eks en klasse som heter *Person* vil gjerne ha datafelt som *navn*, *alder*, *nasjonalitet* etc. En slik klasse vil også inneholde metoder som henter eller manipulerer disse datafeltene. I en applikasjon kan det lages flere instanser av en slik klasse og man kan enkelt referere til disse. En av styrkene til Java er at det kan kjøres på alle plattformer som støtter Java Virtual Machine (JVM).
Java har mye av sin syntaks fra to andre programmeringsspråk, C og C++, men det er noe enklere å lære [17].

2.12.1 Tråd

En tråd er en uavhengig programenhet som eksisterer i en prosess [16]. En prosess i Java er en uavhengig sekvensielt aktivitet som kjører sin egen JVM. Hver prosess har minst en tråd [18]. En tråd kan ha følgende seks tilstander:

- NEW
  Tråd har blit opprettet men ikke startet
- RUNNABLE
  Tråd utfører run() i JVM
- BLOCKED
  Tråd venter på å få monitor lås
- WAITING
  Tråd venter på ubestemt tid på en bestemt handling fra en annen tråd
- TIMED_WAITING
  Samme som WAITING men bare i en spesifisert tid
- TERMINATED
  Tråd har gått ut av run()

En tråd kan bare ha en av disse tilstandene i gangen og gjelder bare i JVM [19].

2.12.2 Flertrådet programmering

Flertrådet programmering er støttet av Java og JVM. Det gjør at et program kan kjøre flere tråder samtidig. Et sekvensielt program trenger tilgang til en kjerne i CPUen for å kjøre. I et flertrådet program kan flere tråder dele på tilgangen til CPUen. Hvilken tråd som kjører er bestemt av en Scheduler [20].
En tråd kan potensielt ha tilgang til hvert objekt i programmet. Det er derfor viktig å sørge for å synkronisere tilgangen til objektene for å unngå at data blir korrupt. Verdier kan bli korrupte ved at en tråd forsøker å lese en verdi som en annen tråd endrer samtidig. Dette løses med bruk av synchronized nøkkelordet. Hvert objekt i Java har en monitor. Dersom en tråd kaller på en metode som har synchronized nøkkelordet i

metodesignaturen, vil den tråden få monitorlåsen. Ingen andre tråder har tilgang til andre synkroniserte metoder i objektet så lenge låsholderen er inne i metoden. Tråden med tilgang kan da trygt gjøre endringer i objektet uten å bekymre seg for at andre tråder leser verdier som ikke er ferdig skrevet til minnet.

2.12.3 Thread-klassen

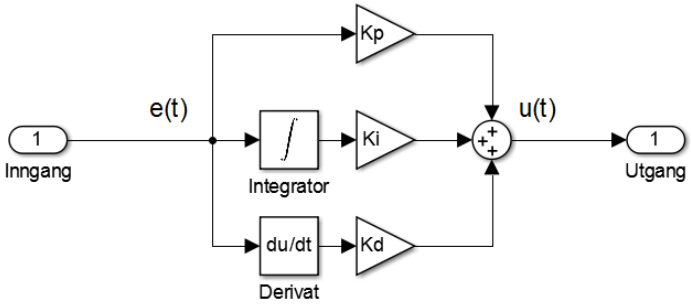
Thread-klassen tilbyr mekanismer for å opprette tråder. Thread implementerer grensesnittet Runnable. Underklasser av Thread må overskrive `run()` metoden fra Runnable og kalle `start()` for å utføre den. Tråder har prioritet i stigende rekkefølge fra 1 til 10. Tråder som ikke blir tildelt prioritet får normalprioritet 5. Når en tråd har stoppet, ved f.eks å la tråden gå ut av `run()`, kan den ikke startes igjen. En ny instanse av tråden kan lages og startes istedet [21].

2.12.4 Timertask-klassen

TimerTask klassen er abstrakt klasse som også implementerer Runnable [22]. Abstrakte klasser kan ikke bli instansiert [23]. Underklasser av TimerTask brukes av en Timer fra java.util.Timer for å bli utført en gang eller med bestemte intervaller. Timer bruker metoder som `schedule(Task,delay,period)` og `scheduleAtFixedRate(Task,delay,period)` [24]. Der task er underklassen av TimerTask, delay er start tiden etter utførelse i ms, og period er periodetiden i ms.

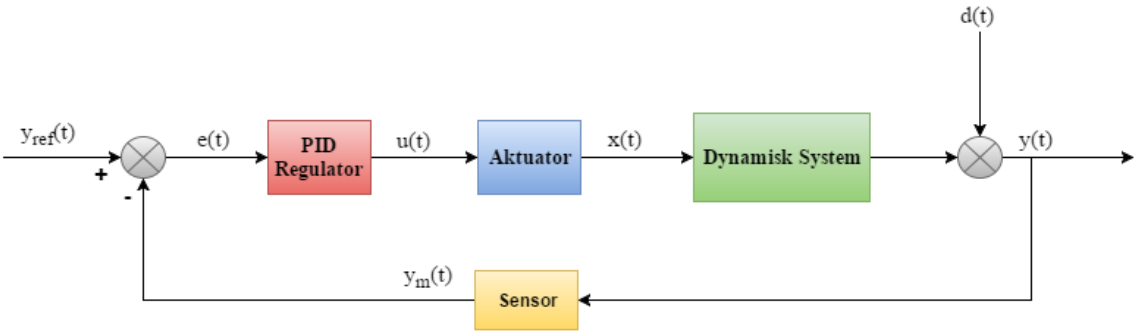
2.13 PID Regulator

PID er en metode bruk til å regulere et dynamisk system. *P* står for proporsjonal fordi den produserer en utgang som er proporsjonal med det nåværende avviket. *I* står for integrasjon fordi den produserer en utgang proporsjonalt med summen til avviket over tid. *D* står for derivasjon fordi den produserer en utgang proporsjonal med endringen i avviket over tid. Summen av P-, I- og D-leddene vil være pådraget til aktuatorene i systemet.



Figur 2.8: Kontinuerlig PID regulator

2.13.1 Tilbakekoblingssystem



Figur 2.9: Reguleringssløyfe m/ negativ tilbakekobling

Et tilbakekoblingssystem er et system der utgangen blir målt og sendt tilbake for å bli brukt som inngang. Dette er illustrert i figur 2.9. Vi har to typer tilbakekobling, positiv og negativ. Positiv tilbakekobling medfører ustabilitet, negativ tilbakekobling benyttes for reguleringsformål [25]. Ved negativ tilbakekobling trekkes utgangen fra settpunktet, slik at avviket mellom utgang og settpunkt blir benyttet som inngang til regulatoren. Eksempel på et tilbakekoblet system kan være *cruise control* (fartsholder) på en bil. Der vil ønsket hastighet være $y_{ref}(t)$ i km/t, $y(t)$ vil være vinkelhastigheten til hjulene i rad/s. $y_m(t)$ vil konvertere rad/s til km/t, bilens translatoriske hastighet. I en negativ tilbakekobling vil denne bli trukket fra referansen og vi vil få et eventuelt avvik $e(t)$. Dette avviket blir inngangen til PID regulatoren, som vil beregne $u(t)$ som blir pådraget til aktuatorene i systemet.

2.13.2 Kontinuerlig PID

Den tidskontinuerlige PID regulator er definert som [26]

$$u(t) = \underbrace{K_p e(t)}_{\mathbf{P}} + \underbrace{K_i \int_0^t e(\tau)\mathrm{d}\tau}_{\mathbf{I}} + \underbrace{K_d \frac{de(t)}{dt}}_{\mathbf{D}} \tag{2.47}$$

der $u(t)$ og $e(t)$, som nevnt i 2.13.1, er PID regulatorens utgang og avvik. K_p, K_i og K_d er henholdsvis forsterkningene til P-, I- og D-leddet.

2.13.3 Diskret PID

Fra den tidskontinuerlige PID fra likn. (2.47) har vi tilsvarende tidsdiskrete PID algoritme:

$$u[n] = \underbrace{K_p e[n]}_{\mathbf{P}} + \underbrace{K_i T_s \sum_{k=0}^{n} e[k]}_{\mathbf{I}} + \underbrace{\frac{K_d}{T_s}\Big[e[n] - e[n-1]\Big]}_{\mathbf{D}} \text{ der } e[n] = 0 \;\forall\; n \leq 0 \tag{2.48}$$

n er sample nummer n i algoritmen og T_s er tiden mellom hver sample i sekund.

Dersom K_i i likn. (2.48) flyttes inn i summasjons-tegnet i I-leddet får vi

$$T_s \sum_{k=0}^{n} K_i e[k] = T_s \Big(K_i e[0] + K_i e[1] + ... + K_i e[n] \Big) \tag{2.49}$$

Dette er tilsynelatende likt likn. (2.48), men dersom K_i endres underveis vil ikke de tidligere samplene bli påvirket. Da unngås et uønsket hopp i utgangen fra regulatoren [27].

2.13.4 Ziegler-Nichols lukket-sløyfe-metode

Ziegler-Nichols lukket-sløyfe-metode er en metode for å bestemme parameterene i en PID-kontroller via eksperimenter, enten på simulator eller på det fysiske systemet.
Metoden er basert på kravet om at systemet skal reguleres så hurtig som mulig, og samtidig beholde akseptabel stabilitet i systemet. Ziegler og Nichols definerte akseptabel stabilitet som at amplituden på svingningene i systemresponsen etter et sprang i settpunktet avtar med en faktor på fire, som betyr at amplituden på den andre toppen i responsen er rundt 1/4 av amplituden av den første toppen. Metoden for å oppnå en opptreden i systemet som er nær disse kravene er beskrevet under [26].

1. Sørg for at regulatorene er i manuell modus.
2. Før prosessen til nominelt arbeidspunkt ved å justere kontrollinngangen manuelt til prosessvariabelen har nådd arbeidspunktet.
3. Sørg for at PID-regulatoren er en ren proporsjonalregulator med $K_p = 0$.
4. Sett kontrolleren i automatisk modus.
5. Øk K_p til prosessen har stående svingninger i responsen etter et sprang i settpunktet.
6. Noter verdien til K_p etter steg 5. Denne verdien kalles *kritisk forsterkning* K_{pk}. Noter i tillegg periodetiden til de stående svingningene. Denne periodetiden kalles *kritisk periodetid* T_k.
7. Avhengig av type regulator, sett parameterene i regulatoren i henhold til formlene oppgitt i figur 2.10.

	K_p	T_i	T_d
P-regulator	$0{,}5K_{pk}$	∞	0
PI-regulator	$0{,}45K_{pk}$	$T_k/1{,}2$	0
PID-regulator	$0{,}6K_{pk}$	$T_k/2$	$T_k/8$

Figur 2.10: Formler for kontrollerparameter.

En av svakhetene med å bruke Ziegler-Nichols metode er at den forårsaker overskyting i tillegg til dempede svingninger i responsen.

2.14 GPS

GPS er et satellittbasert posisjoneringssystem drevet av det amerikanske forsvarsdepartementet. Det grunnleggende prinsippet for denne type posisjonering er måling av reisetiden til signalet fra satellitt til mottaker. En mottaker trenger signaler fra minst fire satellitter for å bestemme mottakerens posisjon og tid, dette løses vha et likningssystem med fire ukjente. En ordinær GPS tilkobling gir nøyaktighet på 5-10m.

Et relativt nytt system for å forbedre nøyaktigheten i GPSer er bruk av landsbaserte stasjoner som korrigerer feilene i målingene. Disse landbaserte stasjonene er plassert på kjente posisjoner og prinsippet kalles *Differensiell GPS*. Med slike korrigeringer kan man måle posisjonen med ca en meters nøyaktighet. I Europa er dette kjent som EGNOS (European Geostationary Navigation Overlay Service) og i USA, WAAS (Wide Area Augumented System). EGNOS ble erklært operativt i 2009 [28].

Det tok 33 år før GPS ble erklært operativt i 1993, og består nå av 24 satellitter. Opprinnelig var det utviklet kun for militært bruk, men pga pris, størrelse og tilgjengelighet opptar sivile omtrent 90% av GPS-bruken i verden [29].

2.15 IMU

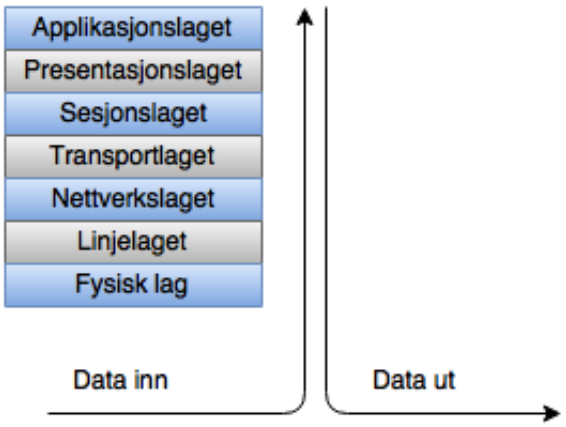
En Inertial Measurment Unit eller IMU oppdager akselerasjon, rotasjon og endringer i magnetiske felt. Det er en kombinasjon av akselerometer, gyroskop og magnetometer. Hver enhet registrerer endringer i tre akser som gir en IMU 9 frihetsgrader. Oppfinneren og patentinnehaveren er Melvin M. Morrison [30]. I dag brukes IMUer mye til treghetsnavigasjon i fly, droner, missiler og maritime fartøy. En datamaskin samler IMU-data og benytter de til å navigere [31].

2.16 Kommunikasjonsprotokoller

En kommunikasjonsprotokoll beskriver hvordan to eller flere enheter skal kommunisere med hverandre over en dataforbindelse. En protokoll beskriver i detalj hvilke data og i hvilken rekkefølge data skal sendes, samt hva som skal utføres på sender og/eller mottakersiden av en melding [32].

2.16.1 OSI-modellen

OSI står for Open Systems Interconnection, og er en lagdelt modell for nettverkskommunikasjon. I modellen tilfører hvert lag en tjeneste til laget over. Modellen består av 7 lag, og er vist i figur 2.11 [32].



Figur 2.11: OSI-modellen.

2.16.2 TCP

Transport Control Protocol eller TCP opererer på transportlaget i OSI-modellen. Den er internettets pålitelige transportprotokoll. Den sender og mottar data via sockets, som er forbindelsen mellom transportlaget og nettverkslaget i OSI-modellen. Den er pålitelig fordi den kontrollerer at alle pakker som blir sendt fra en prosess via socket blir mottatt i den andre enden. Dette gjør den ved at mottaker sender såkalte ACK (acknowledge)-segmenter tilbake. Dersom sender ikke får ACK-segmentet sendes pakkene som mangler på nytt. Kommunikasjon mellom to prosesser må også utføre en "three-way-handshake" for å forsikre seg om at forbindelsen er opprettet. En TCP-forbindelse forblir åpen helt til klienten som initierte forbindelsen velger å lukke den [32].

2.16.3 Internet Protocol

Internet Protocol, normalt forkortet med IP, er en kommunikasjonsprotokoll som opererer i nettverkslaget i OSI-modellen. Denne protokollen sørger for adressering og ruting av datapakker i internettet. Alle nettverksenheter som PCer og mobiltelefoner tildeles en unik IP-adresse når de er tilkoblet internettet. Denne protokollen sammen med rutingprotokoller i rutere sørger for at datapakkene ankommer riktig enhet uansett hvor den måtte befinne seg i verden. Et IP-datagram legger til felt før og etter datapakkene som ligger over i OSI-modellen, som for eksempel TCP-rammer. IP-protokollen fins i to versjoner, IPv4 og IPv6. Hovedforskjellen mellom versjonene er størrelsen på adressefeltet. I IPv4 benyttes 32 bit til adressering, mens IPv6 benytter 128 bit til adressering. Denne endringen kom på grunn av at man begynte å gå tom for IPv4-adresser [32].

2.16.4 USB

USB står for Universal Serial Bus og er standard innen overføring av data mellom datamaskiner og periferienheter (mus, tastatur, joystick etc). Den erstattet de serielle og parallelle utgangene på PCene og gjorde det lettere å installere nytt periferiutstyr. USB 2.0, fra år 2000, har maksimal overføringshastighet på 480 Mbps mens den nye USB 3.0, fra 2010, er ti ganger så rask [33].

2.16.5 WiFi

WiFi eller IEEE 802.11 wireless LAN er en trådløs kommunikasjonsprotokoll for overføring av datapakker mellom aksesspunkt og klient. Wifi bruker frekvensbåndene 2.4- og 5GHz, der 802.11b/a/g bruker det første og -n/ac kan bruke begge. Et slikt trådløs aksesspunkt bør også beskyttes ved bruk av forskjellige krypteringsmekanismer. Den lavest rangerte WEP (Wired Equivalent Privacy) eller den høyst rangerte WPA2 (Wi-Fi Protected Access) + AES (Advanced Encryption Standard). Den første versjonen av 802.11 standarden kom i 1997 og hadde maks hastighet på 2 Mbps. I dag tillater 802.11ac opp mot 1 Gbps [32].

2.17 Versjonskontroll

Versjonskontroll er et system som håndterer endringer i filer i prosjekter over tid, slik at man kan gå tilbake til tidligere versjoner av en fil om ønskelig. Versjonskontroll er mye brukt innen programvareutvikling, men det er i prinsippet ingen begrensning for hvilke typer filer som kan kontrolleres.

Versjonskontroll kan løses på tre ulike måter; Lokal versjonskontroll, sentralisert versjonskontroll, og distribuert versjonskontroll. Lokal versjonskontroll lagrer versjoner av prosjektet lokalt på brukerens datamaskin. Sentralisert versjonskontroll benytter en server som lagrer alle filene i prosjektet som er under versjonskontroll. Brukere som skal arbeide på en fil sjekker ut kun den ene filen. Når brukeren er ferdig, lastes filen opp til serveren igjen. Distribuert versjonskontroll lagrer også alle filer på en sentral server, men brukere som skal arbeide på en fil kloner nå hele prosjektet som ligger på serveren. Dermed har brukere som arbeider på filer en fullstendig kopi av alle filer som er lagret på serveren, og om serveren skulle få problemer så har brukeren en fullstendig versjon av prosjektet på sin datamaskin. [34].

2.17.1 Git

Git er et distribuert versjonskontrollsystem som fikk sin unnfangelse fra utviklingsmiljøet som utviklet Linux-kjernen i 2005. Git lagrer data som øyeblikksbilder, i stedet for i en liste over filbaserte endringer, som i andre systemer. Git behandler prosjektet som et miniatyr filsystem, og hver gang brukeren lagrer sine endringer tar Git et øyeblikksbilde av filsystemet der og da, og lagrer en referanse til øyeblikksbildet. Denne løsningen gjør at det er veldig lett å gå tilbake til tidligere versjoner av prosjektet [34]. For å benytte Git må programvaren installeres på brukerens datamaskin.

2.17.2 Bitbucket

BitBucket er en web-basert tjeneste for versjonskontroll, eid av Atlassian, som støtter Git. Gratisversjonen av Bitbucket lar opp til fem personer samarbeide i et prosjekt, og har ingen øvre grense for antall prosjekter en bruker kan opprette. Gratisversjonen av BitBucket tillater at prosjektene er private, i motsetning til andre populære løsninger som GitHub [35].

2.18 Batteriteknologier

Det finnes mange typer batteri på markedet. Dette kapittelet er en kort oppsummering av de mest brukte batteritypene. Tabell 2.2, hentet fra [36], sammenligner og viser nøkkelegenskapene til de forskjellige batteritypene.

2.18.1 Blyakkumulator

I ladet tilstand består den positive elektroden i en blyakkumulator av blyoksid (PbO2) og den negative består av porøst bly (blysvamp). Elektrolytten er svovelsyre. Ved utladning reduseres blydioksid på positiv pol ved at oksygenioner går ut i elektrolytten og forbinder seg med hydrogenioner til vann. Sulfationer i svovelsyren går til begge elektroder og danner blysulfat. Elektrolytten deltar i prosessen og går over til vann. Syreinnholdet og dermed densiteten i elektrolytten blir derfor et mål på batteriets ladetilstand [37].

2.18.2 Nikkel-kadmium (NiCd)

Nikkel-kadmium (NiCd) batteriet har nikkeloksid som positiv elektrodemasse og kadmiumhydroksid som negativ elektrodemasse. Elektrolytt er kaliumhydroksid. NiCd-batteriet har flere fordeler i forhold til blyakkumulatoren. Det er mindre og lettere, og det holder mer konstant spenning ved utladning. I tillegg beholdes kapasiteten og belastningsegenskapene bedre ved lave temperaturer [37].

2.18.3 Nikkel-metallhydrid (NiMH)

Nikkel-metallhydrid (NiMH) batteriet har nikkeloksid som positiv elektrode og en høyporøs hydrogenabsorberende legering av metallhydrider som negativ elektrode. Det er en videreutvikling av NiCd batterier, med bedre egenskaper. Det har mange av de samme, men til dels bedre, egenskaper enn NiCd-batteriet [37].

2.18.4 Litium-ione (Li-ion)

Litium-ione batteri består av en positiv elektrode av litium-koboltoksid, en separator og en negativ elektrode av karbon, og organisk elektrolytt som iontransportør. Det har mer eller mindre erstattet de nikkelbaserte batteritypene. Det tåler godt klattlading og er svært stabilt med tanke på selvutladning [37].

2.18.5 Litium-polymer (LPB)

Litium-polymer batteri er en variant av Li-ion med en fast polymer elektrolytt. Ellers har det veldig like egenskaper som Li-ion [37].

2.18.6 C-rate

C-rate er et mål som forteller ved hvilken strøm et batteri lades eller utlades. Ved en rate på 1C vil et batteri på 1Ah utlades på 1 time. C-rate kan defineres slik: C-rate=kapasitet i Ah/(ut)ladetid [38].

	NiCd	NiMH	Blyakkumulator	Li-ione	LPB
Gravimetrisk energitetthet (Wh/kg)	45-80	60-120	30-50	110-160	100-130
Indre motstand (mΩ)	100 til 200 6V pakke	200 til 300 6V pakke	<100 12V pakke	150 til 250 7,2V pakke	200 til 300 7,2V pakke
Levetid antall sykluser (til 80% av opprinnelig kapasitet)	1500	300 til 500	200 til 300	500 til 1000	300 til 500
Hurtigladetid	1h	2-4h	8-16h	2-4h	2-3h
Toleranse for overladning	Middels	Lav	Høy	Veldig lav	Lav
Cellespenning	1,25V	1,25V	2V	3,6V	1,5V
Laststrøm -peak -beste resultat	20C 1C	5C 0,5C eller lavere	5C 0,2C	>2C 1C eller lavere	>2C 1C eller lavere
Driftstemperaturområde	-40 til 60°C	-20 til 60°C	-20 til 60°C	-20 til 60°C	0 til 60°C
Vedlikeholdsintervall	30 til 60 dager	60 til 90 dager	3 til 6 mnd	Ikke påkrevd	Ikke påkrevd

Tabell 2.2: Sammenligning av forskjellige batteriteknologier [36]

Kapittel 3

Materialer og Metode

3.1 Data

3.1.1 NMEA 0183 Standarden

Posisjonsdata for prototypen leses fra GPS i form av NMEA-setninger. National Marine Electronics Association (NMEA) har definert en standard for lesing av GPS data kalt NMEA-setninger. Det finnes flere typer NMEA-setninger. Felles for alle er at de starter med $, etterfulgt av hvilken type det er, f.eks GPRMC, GPGGA eller GPGSV. Hver type har noen felt til felles, f.eks klokkeslett eller posisjon, mens resten er ulike. Nøyaktigheten i dataene fra GPS-en avhenger av flere faktorer. De faktorene som kan kontrolleres er valg av modul. For å få best mulig nøyaktighet bør man velge en modul med støtte for EGNOS/differensiell GPS, som er et landbasert tilleggsystem for å korrigere feil og gi langt bedre nøyaktighet. Videre bør målingene skje utendørs da GPS signaler har problemer med å gjennomtrenge tykke tak. Faktorer som ikke er kontrollerbare er atmosfæriske effekter og skytetthet. Disse usikkerhetene kan minskes med bruk av DGPS. To eksempler og beskrivelser på NMEA setninger er vist i tabell 3.1 og 3.2:

```
$GPRMC,123519,A,4807.038,N,01131.000,E,022.4,084.4,190416,003.1,W*6A
$GPGGA,123519,4807.038,N,01131.000,E,2,08,0.9,545.4,M,46.9,M,,*47
```

Nummer	Feltnavn	Eksempel Data	Beskrivelse
1	Type	$GPRMC	Recommended minimum specific GPS/Transit data
2	Klokkeslett	123519	12:35:19 i posisjonens tidssone
3	Tilkobling	A	A-Aktiv/V-Void
4	Breddegrad	4807.038,N	48 grader 7.038 min nord
5	Lengdegrad	01131.000,W	011 grader 31.000 min vest
6	Fart	022.4	Fart målt i knop
7	Kurs	084.4	Bevegelseskurs 0-359
8	Dato	190416	Dato ved oppnådd fix
9	Tillegg	3.1	Magnetisk variasjon(grader)
10	Sjekksum	W*6A	Obligatorisk sjekksum

Tabell 3.1: GPRMC tabell

Nummer	Feltnavn	Eksempel Data	Beskrivelse
1	Type	$GPGGA	Global Positioning System Fix Data
2	Klokkeslett	123519	12:35:19 i posisjonens tidssone
3	Breddegrad	4807.038,N	48 grader 7.038 min nord
5	Lengdegrad	01131.000,W	011 grader 31.000 min vest
6	Fix kvalitet	2	0=ingen,1=GPS,2=DGPS
7	Antall satellitter	8	Satellitter som sender til GPS-en
8	Horisontal svekking av presisjon	0.9 grader	Relativ horisontal treffsikkerhet
9	Høyde	545.4,M	545.4m over gjennomsnittlig havnivå
10	Høyde over WGS84	-34,M	-34m
11	DGPS tillegg	ingen ID	DGPS referansestasjon ID
12	Sjekksum	*47	Obligatorisk sjekksum

Tabell 3.2: GPGGA tabell

3.1.2 Kompasspeiling

For å måle peilingen til prototypen brukes det kombinert magnetometer og gyroskop. Magnetometeret måler magnetiske felt og kan dermed gi peiling relativt til den magnetiske nordpol. Men slike magnetometer er utsatt for elektrisk støy og må kalibreres i den omgivelsen det skal brukes. Det reagerer også dårlig på raske bevegelser. Gyroskop måler raskt endringer i vinkelposisjon og hastighet, men er avhengig av å starte i en kjent posisjon. I tillegg vil et gyroskop drifte over tid. Sammen vil disse kunne gi en nøyaktig peiling for prototypen.

3.1.3 Målinger i NED koordinatsystemet

Målinger er generert i applikasjonen ombord i prototypen og på klientsiden under kjøring av DP-systemet. Det måles en x- og y-posisjon ved hjelp av GPS-en og en algoritme for konvertering til NED-koordinatsystemet, beskrevet matematisk i kapittel 2.5.3, samt en peiling relativt til den magnetiske nordpol. Forflytning i NED gjelder for små endringer i lengde- og breddegrader, veldig store forflytninger vil føre til unøyaktige målinger.

3.2 Prosjektorganisering

Prosjektgruppen har vært organisert med en prosjektleder og en sekretær. Prosjektleder sine ansvarsområder har vært å oppdatere fremdriften til prosjektet i Gantt-diagrammet og kalle inn til og lede møter med styringsgruppen. Sekretærens oppgaver har vært å reservere møterom for prosjektmøte, skrive og distribuere møtereferat, samt skrive fremdriftsrapport før hvert møte med styringsgruppen. Selv om gruppen har hatt en prosjektleder har det blitt praktisert flat ledelsesstruktur i prosjektgjennomføringen, hvor avgjørelser har blitt tatt i fellesskap.

Gjennom hele prosjektet har det blitt holdt møter med styringsgruppen ca. hver 14. dag, avhengig av passende tidspunkt for veilederene. Under møtene har det blitt diskutert fremdrift for prosjektet, gruppen har presentert sine forslag til løsninger på problemstillinger, og styringsgruppen har gitt veiledning, tips og forslag til løsninger. Under forprosjekteringen ble det utarbeidet en prosjektplan med utgangspunkt i den gitte oppgaven. Prosjektplanen ble laget ut fra fossefallsmetoden, og et Gantt-diagram basert på prosjektplanen har vært styrende for prosjektgjennomføringen. Microsofts skylagringstjeneste OneDrive har blitt benyttet for lagring og deling av alle relevante filer og dokumenter gjennom prosjektet.

3.3 Programvare

Programvare som er benyttet i prosjektet er listet opp under.

- NetBeans 8.1 IDE
- Arduino IDE
- SolidWorks
- Google Earth Pro
- Inkscape 0.91
- Draw.io

- ShareLaTeX
- MatLab
- GeoGebra
- Git

NetBeans 8.1 IDE er benyttet til programvareutvikling, og er et gratis utviklingsmiljø som er utviklet primært for Java. NetBeans støtter mange forskjellige versjonskontrollsystemer. NetBeans er kompatibelt med alle OS som støtter JVM.

Arduino IDE er benyttet til å utvikle programvare til Ardino. Arduino IDE brukes for å programmere Arduino/Genuino mikrokontrollere via USB i en forenklet versjon av C-språket [39].

SolidWorks er et 3D-modelleringsprogram for Windows, utviklet av Dassault Systèmes. SolidWorks kan benyttes til blant annet utvikling, analysering, maskinering og simulering av både enkle og komplekse komponenter/systemer [40]. SolidWorks er benyttet for design av 3D-modell av jollen og adaptere for innfesting av thrustere i rør.

Google Earth Pro er en interaktiv globus, som kan vise posisjoner utifra NMEA-setninger og andre lister med data [41]. Google Earth Pro er brukt for å visualisere posisjonen til USV fra avleste NMEA-setninger fra GPS.

Inkscape 0.91 er et gratis redigeringsprogram for vektorgrafikk, og støttes av Windows, Mac og Linux [42]. Inkscape er benyttet til tegning av figurer.

Draw.io er et gratis online program for tegning av diagrammer og lignende [43]. Draw.io er brukt til å tegne klassediagrammer og figurer.

ShareLaTeX er et online LaTeX redigeringsprogram som tillater samarbeid i sanntid i samme prosjekt. Dokumenter kan kompileres til pdf-format gjennom nettleseren [44]. Prosjektrapporten ble skrevet i ShareLaTeX.

MatLab er et program for numeriske beregninger og simulering utviklet av MathWorks. Det støtter scripting i et eget høynivå programmeringsspråk, samt flere interaktive applikasjoner og metoder [45]. MatLab er benyttet for behandling av NED-data fra testing av DP.

GeoGebra er et interaktivt geometriprogram som benyttes til undervisningsformål [46]. I prosjektet er GeoGebra benyttet til polynomtilpasning av skyvkraft/PWM-data for thrustere.

Git ble benyttet for versjonskontroll av programvare under utvikling.

3.4 Programvareutvikling

Programvaren som har blitt utviklet i prosjektet er skrevet i enten Java-språket eller Arduino C. Java ble brukt som programmeringsspråk på grunn av at det er et kjent språk for alle prosjektmedlemmene, og har innebygde mekanismer for sanntidsprogrammering. Det ble opprettet to ulike prosjekt i NetBeans IDE, ett for klientsiden og ett for serversiden (USV siden). Det ble opprettet ett prosjekt i Bitbucket for hvert NetBeans-prosjekt, og Git ble brukt for versjonskontroll. Koden som ble utviklet til Arduino ble lagret sammen med kildekoden på serversiden.

Programvaren ble utviklet ved at gruppen laget et overordnet bilde av programmeringsmessige utfordringer som må løses, og laget forslag til klasser i Java ut fra dette. Deretter ble oppgaver fordelt på gruppemedlemmene delvis ut fra egne ønsker, og delvis ut fra kompetanse.

Problemene som måtte løses kan deles opp som følgende

- Utvikling av grafisk brukergrensesnitt
- Kommunikasjon mellom klient og server
- Lese og sortere data fra sensorer
- Beregne nødvendig bevegelse basert på mottatt data
- Generere kontrollsignal til thrustere, basert på ønsket bevegelse

Under utviklingen av programvaren ble det lagt vekt på at alle delte variabler og objekter skal være trådsikre, i henhold til sanntidsprogrammeringsformalismer. For kommunikasjon mellom tråder ble det benyttet delt-variabel synkronisering. Tråder som kommuniserer har tilgang på samme objekt/variabel, og synkroniserte metoder sørger for at kun èn tråd har lese- eller skrivetilgang til enhver tid. Generelt under utviklingen ble det lagt vekt på at hver klasse skal ha et klart definert og avgrenset ansvarsområde. Klassene er utviklet for å være minst mulig avhengig av hverandre, såkalt lav kobling. Programmet som kjører ombord i prototypen er kompilert til en .jar-fil og et oppstartsskript sørger for at programmet starter automatisk under oppstart av prototypen.

3.4.1 Bibliotek

Tredjepartsbibliotek for Java som er benyttet i prosjektet er:

- JOptimizer
- Apache Commons Math
- jInput
- jSerialComm
- jFreeChart

JOptimizer og Apache Commons Math benyttes til å løse optimaliseringsproblemet i thrusterallokeringen. JInput benyttes til å lese inputdata fra joystick. JSerialComm benyttes for å lese serielldata fra USB i Java. JFreeChart benyttes for å tegne kurver basert på NED-data i brukergrensesnittet.

Bilbliotek som er benyttet for Arduino er:

- SoftwareSerial
- Adafruit GPS Library
- Servo

SoftwareSerial er brukt for å kunne benytte PWM-pinner på Arduinoen til seriell kommunikasjon. Adafruit GPS Library er brukt for å sende og motta data fra Adafruit Ultimate GPS via SoftwareSerial-pinnene. Servo biblioteket ble brukt til å sende PWM-signal til motorkontrollerene som styrer thrusterene.

3.4.2 Java Marine API

Java Marine API brukes for å bearbeide NMEA-setningene fra Adafruit GPS-en. Det sikter på å gjøre data fra marine enheter (som GPS) lett tilgjengelig. Den støtter 23 typer setninger, deriblant GPRMC og GPGGA. Biblioteket konverterer posisjonen i NMEA setningen fra grader og desimalminutt til desimalgrader.

3.4.3 PMTK-pakker

PMTK-pakker benyttes for å sende såkalte kommandopakker til GPS-modulen via en seriell tilkopling. Disse datapakkene har en fast oppbygging, som er spesifisert i [47]. Pakkelengden er maksimalt 255 byte. Oppbyggen av pakken er som vist i tabell 3.3.

Innledning	Sender ID	Pakketype	Datafelt	*	CHK1	CHK2	<CR>	<LF>

Tabell 3.3: PMTK pakkeformat

En beskrivelse av hvert felt står i tabell 3.4.

Felt	**Lengde**	**Type**	**Beskrivelse/merknad**
Innledning	1 byte	char	”$”
Sender ID	4 byte	String	”PMTK”
Pakketype	3 byte	String	Fra ”000” til ”999”
Datafelt	Variabel		Et ”,”-tegn må komme før datafeltet
*	1 byte	char	Markerer enden av datafeltet
CHK1, CHK2	2 byte	String	Sjekksum av datafeltet
<CR>, <LF>	2 byte	Binær data	Markerer slutten på datapakken

Tabell 3.4: Datafelt i PMTK-pakke

3.5 Materialer

3.5.1 Joystick

Logitech Wingman joystick med USB tilkobling. I prosjektet leses bevegelser i X-og Y-aksen samt rotasjon, som benyttes for å styre USV-en i manuell modus.

Figur 3.1: WingMan Joystick

3.5.2 TP-Link WL-Router TL-WR841N

Wifi-router med 300 Mbps båndbredde, som er brukt til å rute datapakkene mellom klientmaskinen og USV. Den benytter 802.11n standarden.

Figur 3.2: TP-Link WL-Router

3.5.3 Multicom N250JU Bærbar PC

Multicom N250JU bærbar PC er benyttet i prosjektet for å kjøre klientapplikasjonen. Spesifikasjoner:

- Intel(R) Core(TM) i7-6500U CPU 2.60GHz
- 8.00 GB RAM
- Intel HD Graphics 520
- Windows 10 Education x64
- 250GB SSD

3.6 Konseptutredning

Dette delkapittelet har fokus på designkrav og funksjonalitet på et tenkt konsept på 8-10 fot med enkelt skrog. Delkapittelet fokuserer ikke på regulering og dynamisk posisjonering, som blir utredet ellers i rapporten.

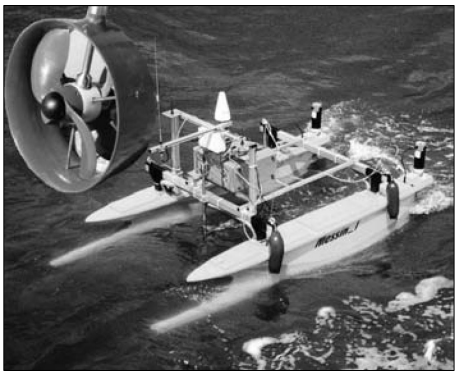
Ved utvikling av en USV må systemet deles opp i flere delsystem. Deretter settes ønskede krav til delsystemene, og tilslutt ser man om spesifikasjonene mellom delsystemene går overens og man inngår kompromiss.

De største delsystema til en USV kan inndeles i

- Skrog
- Motor og propellsystem
- Energi/drivstoff system
- Kontroll- og reguleringssystem med sensorer, motorstyring, navigasjon, DP osv.
- Kommunikasjon mellom operatør og USV
- Annet påmontert utstyr

3.6.1 Skrog

Å utvikle dedikerte skrog for en USV kan være dyrt, derfor brukes ofte etablerte skrog [48]. Valget av skrog kommer an på hvilke typer oppdrag den skal utføre og hva slags sjø den skal operere i. De tre mest brukte typene skrog er fortrengningsskrog, gummibåt/RIB og katamaran. Disse typene har alle sine fordeler og ulemper. En katamaran, vist i figur 3.3 hentet fra [48], vil bli mye påvirket av strøm, men mindre av vind. Dette fordi at et slikt skrog har høy vannmotstand sideveien. I dynamisk posisjonering vil katamaranen trenge høy kraft fra sidethrusterene på grunn av motstanden i vannet. Til fremdrift vil katamaranen ha lite motstand i vannet og vil derfor trenge relativt lav kraft til dette. En gummibåt/RIB, som vist i figur 3.4 hentet fra [48], vil flyte lett på vannet og vil derfor være mer påvirket av vind, men mindre av strøm. En slik type båt vil derfor trenge mindre kraft på sidethrusterene. Siden en slik båt flyter lett vil også denne trenge relativt lav kraft til fremdrift. Et fortrengningsskrog, vist på figur 3.5, vil kreve sidethrustere som er sterkere enn for gummibåten, men litt svakere enn for katamaranen. Fortrengningsskrog vil ha en merkbar høyere motstand i fartsretningen, selv om dette kommer an på hvor rundt skroget er, og vil derfor være mindre energieffektiv ved vanlig kjøring. Den største fordelen med jolleskroget er at det har stor plass til utstyr og last. Denne typen skrog er derfor et godt alternativ for en allsidig USV som eventuelt skal ombygges, eller settes mer utstyr på, i fremtiden.



Figur 3.3: Katamaran [48]

Figur 3.4: Rib [48]

Figur 3.5: Fortrengningsskrog

3.6.2 Thrusterbehov

Noe av det første som må gjøres etter at type skrog er bestemt er plassering av eventuelle thrustere. Thrusterenes plassering og antall vil bestemme antall frihetsgrader båten kan styres i. Ved dynamisk posisjonering ønskes regulering i jaging, tverrskips og giring. I praksis betyr det å nærme seg en posisjon, mens man holder en gitt peiling. Båten må derfor ha kraft til å gå framover og bakover (jaging), sideveien

(tverrskips) og rotasjon (giring). Man må minst ha like mange aktuatorer som antallet frihetsgrader som kontrolleres [7]. Ved vanlig kjøring vil båten være avhengig i å ha en framdrift som kan svinges. Dette utgjør at båten kan justere retningen sin uten hjelp av ekstra kraft fra andre thrustere. Når båten står stille i fartsretningen er den også avhengig av at den har motorkraft som kan flytte den sideveien. Denne siste egenskapen gjør at båten kan holde seg i eller justere seg mot en ønsket referanseposisjon i to frihetsgrader (jaging og tverrskips). For å oppnå den siste frihetsgraden giring (peiling), må den driften som gjør at båten går sideveien også kunne svinge den. For å oppnå dette kan det brukes to toveis-sidethrustere der den ene står framme og den andre står bak. Dersom momentet om massesenteret i lengderetningen til båten er like stort for begge side-thrusterene vil båten rotere uten å miste sin nåværende posisjon.

3.6.3 Valg av type thruster

Til fremdrift kan det velges mellom:

- Tunnelthruster(e)
- Roterende Azimuth thruster(e)
- Vannjet

Til sideforskyving kan det velges mellom:

- Tunnelthruster(e)
- Roterende Azimuth thruster(e)
- Eller en kombinasjon av disse to

Det enkleste thruster-oppsettet er å bruke statiske (faste, ikke-roterbare) thrustere [49]. For å styre båten i de tre frihetsgradene jaging, tverrskips og giring kan et typisk oppsett være; en thruster til fremdrift, en thruster til sideforskyving bak på skroget, og en thruster til sideforskyving fremme på skroget. Fordelen med dette oppsettet er at hver thruster har en bestemt oppgave og utgjør visse moment i forhold til skroget i sine respektive retninger. I forhold til dynamisk posisjonering er thrusterene alltid klar for sitt respektive arbeid ved behov. Ulempen ved et slikt oppsett er at thrusterene ikke kan turnes ved vanlig kjøring. Derfor må det monteres en form for ror på skroget for å få turning ved vanlig kjøring. For effektiv bruk av tilgjengelig motorkraft kunne to azimuth-thrustere erstattet de statiske thrusterene for fremdrift og sidedrift bak. Fordelen med en azimuth-thruster er at denne kan roteres +-90 grader. Dette betyr at azimuthene brukes til fremdrift samt å gjøre den jobben som tidligere sidethrusteren hadde i den dynamiske posisjoneringen. Ved en slik løsning er det dermed ikke behov for et ror til å turne siden azimuthene kan justere headingen til båten ved vanlig kjøring [50]. I denne oppgaven tar vi for oss utvikling av et ubemannet fartøy som er relativt saktegående og energieffektivt. Dette utelukker vannjet, da denne type fremdrift ikke er effektivt på hastigheter fra rundt 30 knop og nedover [14].

3.6.4 Plassering av sidethrustere

Ved plassering av sidethrustere må det taes hensyn til momentet om båten sitt massesenter og at thrusteren ikke suger luft. Momentet til hver thruster blir den effektive kraften som thrusteren leverer multiplisert med avstanden fra båten sitt massesenter i lengderetningen [15]. Jo lenger en thruster står fra massesenteret jo større blir momentet om aksen som massesenteret utgjør. En tilsynelatende svak sidethruster kan derfor likevel bidra til å svinge båten om z-aksen ved å utnytte dette prinsippet. Når båten ligger i dynamisk posisjonering i en bestemt posisjon og det prøves å endre båten sin heading ønskes det at fartøyet skal

manøvreres rundt massesenteraksen. Forutsetningen er at momentet om aksen er like stort fra begge sidethrusterene. Kraften på sidethrusterene kan altså være ulik, så lenge de potensielt kan utgjøre det samme momentet om massesenteraksen.

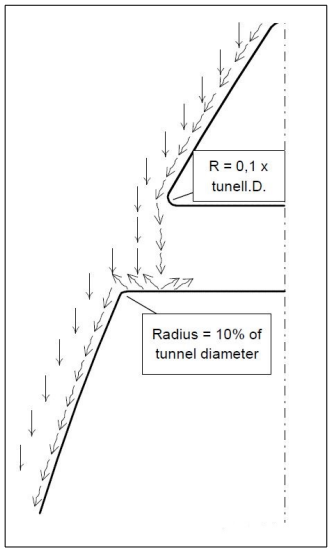
En thruster som suger luft vil miste kraft og er ikke pålitelig. Det er derfor ønskelig at thrusteren ligger dypest mulig i sjøen. Vanntrykket er også høyere jo lengre under overflaten thrusteren ligger. Økt vanntrykk gjør at kraften fra thrusteren blir mer stabil [51].

3.6.5 Lengde på thrustertunnel

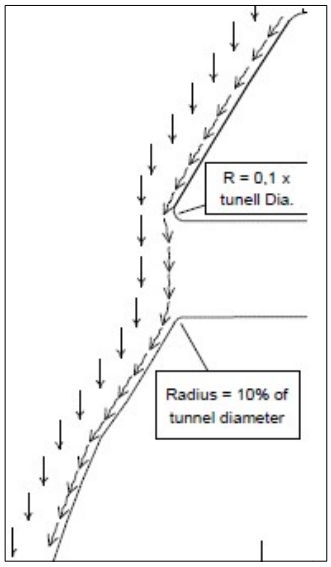
Ved plassering av thrustertunneler er en viktig faktor hvor lange tunnelene blir i forhold til thrusterdiameteren, det vil si diameteren til propellen. Side-Power anbefaler tuneller som er 2 til 4 ganger thrusterdiameteren, og tunneler som er lengre enn 6 til 7 ganger thrusterdiameteren bør unngås [51]. Dette er på grunn av at friksjonen i tunnelen vil bremse flyten til vannet på vei gjennom tunnelen og vil derfor hele tiden jobbe mot skyvkraften til thrusteren.

3.6.6 Reduksjon av drag skapt av sidethrustere

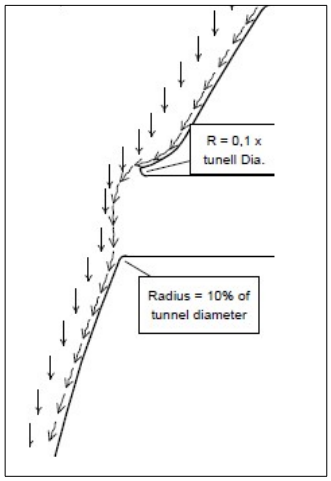
Sidethrustere som ligger under vann vil ved vanlig kjøring skape drag slik som vist på figur 3.6. For å redusere draget kan det lages et innsving i skroget bak tunnelen som på figur 3.7. Innsvinget vil gjøre at vannet ikke har en direkte kant å treffe og draget blir derfor kraftig redusert. I figur 3.8 demonstreres et alternativ der en spoiler foran thrusteren leder bort vannstrømmen fra kanten på thrustertunnelen.



Figur 3.6: Tunnel som ikke tar hensyn til drag [51].



Figur 3.7: Innsving etter tunnel som gir vannet bedre flyt forbi åpningen [51].



Figur 3.8: En alternativ løsning med en vannspoiler foran tunnelen [51].

3.6.7 Utforming av tunnelåpninger

Tunnelåpningen bør avrundes langs og oppover skroget. Dette reduserer turbulens/kavitasjon i røret samt at det skaper et undertrykk langs den siden av skroget som thrusteren tar vann fra. Dette undertrykket kan utgjøre opptil 40% av den totale effektive kraften levert fra thrusteren [51]. Dersom det brukes en skarp tunnelåpning vil det oppstå turbulens/kavitasjon. Denne turbulensen/kavitasjonen vil dermed oppstå i de ytre delene av tunnelen og den effektive tunneldiameteren vil dermed bli redusert. Denne effekten vil redusere ytelsen til thrusteren og føre til unødvendig støy. Avrunda tunnelåpninger vil derfor gi fartøyet lenger driftstid per energienhet samt høyere tilgjengelig effektiv kraft. Høyere effektiv kraft er ønskelig siden fartøyet da vil

levere høyere ytelse og kan brukes i vanskeligere værforhold. For en USV vil det også være viktig å beskytte thrustertunnelene mot inntrengning av fremmedlegemer. En USV kan for eksempel operere i nærheten av oppdrettsanlegg, hvor det er fare for at tau eller garnnot blir sugd inn i tunnelene. Beskyttelse av thrustertunnelene har også et sikkerhetsmessig aspekt. Det bør, så langt det lar seg gjøre, være umulig for mennesker å skade seg på roterende deler. Tunnelåpningene kan for eksempel beskyttes med gitter eller netting. Det er viktig at beskyttelsen ikke reduserer vannflyten i nevneverdig grad.

3.6.8 Beregning av effektbehov

Framgangsmåte for å beregne effektbehov for framdriftssystem, hentet fra læreboken *Ship Resistance and Propulsion* [14]:

1. Beregn total stille vanns motstand R_T ved hastighet V
2. Aktiv effekt $P_E = R_T \times V$ (kraft som kreves for å oppnå V med motstand R_T)
3. Beregn kvasi-propulsiv koeffisient η_D.
4. Levert effekt til propell $P_D = \frac{P_E}{\eta_D}$.
5. Beregn modell-skip korrelasjonsfaktor SCF.
6. Korrigert levert effekt: $P_{D\text{skip}} = P_{D\text{modell}} \times SCF$
7. Overføringstap η_T
8. Generert effekt $P_S = \frac{P_{D\text{skip}}}{\eta_T}$
9. Margin, typisk 15-30% for å kompensere for groing på skroget, vær og lignende
10. Total installert effekt $P_I = \frac{P_E}{\eta_D} \times SCF \times \frac{1}{\eta_T} + \text{margin}$

Problemet kan deles inn i tre deler; estimering av aktiv effekt, estimering av kvasipropulsiv koeffisient, og estimering av marginer.

For å beregne aktiv effekt trenger man et mål for total stille vanns motstand for fartøyet ved den gitte hastigheten. Beregning av total stille vanns motstand utføres normalt ved hjelp av en skalamodell i et testbasseng [14]. Det er mulig å beregne motstanden til et gitt skrog analytisk med avanserte datamodeller (CFD). Slike modeller vil kunne gi et godt innblikk i hvordan kreftene virker på skroget, men det krever en nøyaktig modell av skroget og kraftige datamaskiner.

En prosedyre for å måle skrogets motstand i vannet finnes i [52], International Towing Tank Conference – Recommended Procedures 7.5-02-02-01 Resistance Test. Denne prosedyren tar for seg testing av modeller i en tauetank, og kan benyttes om det utviklede skroget til USV skal testes i basseng. Det finnes i tillegg empiriske metoder, hvor resultater fra mange modellforsøk og fullskalaforsøk på motstand er samlet og brukt til å lage formler som estimerer motstanden på lignende konstruksjoner. En mye brukt empirisk metode er Holtrops metode [53].

Merk at alle disse metodene krever enten en fysisk modell av skipet eller at skrogets fysiske dimensjoner er kjente. Holtrops metode krever i tillegg at skroget er likt skrogene som modellen er basert på.

Kvasi-propulsiv koeffisient η_D avhenger hovedsaklig av thrusteren eller propellens effektivitet, og er et mål på effekttap i konvertering av roterende kraft til skyvkraft. Metoder for å beregne η_D kan finnes i [14].

Overføringstapet η_T er et mål på effekttap i kraftoverføringssystemet. Dette tapet avhenger av hvordan framdriftssystemet er designet. For en USV som er drevet av batterier vil dette typisk være tap i kabler og frekvensomformere, og elektromotorenes effektfaktor.

Modell-skip korrelasjonsfaktoren korrigerer for forskjeller mellom modellforutsigelser for levert effekt og resultater fra sjøtesting [14].
For å kunne estimere dimensjoner på thrustere og batteripakke kreves det altså at skroget er designet, og motstanden er kjent.

Strømmer i norske fjorder er normalt under 0,5 m/s, men kan i visse tilfeller komme opp i flere m/s, spesielt i trange sund [54]. Hvis vi antar at under normale operasjonsforhold til USV vil strømningsfarten være under 0,5 m/s (≈1 knop), vil en masjfart på 5 knop (=2,57 m/s) være høy nok til at USV kan forflytte seg med god margin for strømningsforhold. Et fartøy med 2 meter lengde ved vannlinjen vil med masjfart 2,57 m/s ha Froude tall $F_n = \frac{V}{\sqrt{gL}} = \frac{2{,}57}{\sqrt{9.81 \cdot 2}} = 0.58$, og vil klassifiseres som et semi-deplasementfartøy.

3.6.9 Valg av kraftforsyning

I spesifikasjonskravene er det stilt krav til at USV-en skal drives med batteri. Dimensjonering av batteripakke til en ferdigutviklet USV krever at fremdriftssystemet er dimensjonert, som igjen krever at skroget et designet og motstanden i vannet er kjent. Likevel kan man ta stilling til hvilken batteriteknologi som er mest egnet til det bruksmønsteret og -miljøet en USV sannsynligvis vil utsettes for.

For en USV vil det være viktig at batteriet er miljøvennlig i tilfelle havari eller uhell. Energitettheten bør være høyest mulig for å redusere vekt, og det må kunne levere nok strøm til å drive fremdriftssystemet. Vedlikeholdsbehovet bør være lavest mulig, og ladetiden bør også være relativt lav for å maksimere bruksvinduet. Lav pris er også et kriterie. Ved å se på batteriegenskapene som er vist i tabell 2.2, kan man gjøre noen seg vurderinger.

Man ser at NiCd har størst levetid, kortest ladetid og høyest laststrøm, men krever mest vedlikehold. I tillegg inneholder NiCd-batterier giftige metaller, forurensing til sjø er en potensiell risiko.

NiMH har mer kapasitet enn NiCd, og inneholder mindre miljøgifter enn NiCd. Men også NiMH krever vedlikehold (full utladning) ofte, og har ganske kort levetid. I tillegg er utladestrømmen i NiMH-batterier lavere enn for NiCd. Ladetiden er også lengre.

Blyakkumulatoren er det billigste batteriet, og har lite selvutladning. Men energitettheten er den laveste av alle typer, og utladestrømmen er lavest av alle. Ladetiden er lang. Antall utladesykluser er også begrenset. I tillegg inneholder blyakkumulatoren miljøgifter.

Li-ionebatterier har høyest energitetthet, og et høyt antall utladesykluser. Selvutladingen er relativt lav. Utladestrømmen er forholdsvis høy, men peak strømmen er begrenset i forhold til blyakkumulatoren eller nikkelbaserte batterier. Li-ion batterier krever ikke spesielt vedlikehold. Li-ion batterier er kostbare, og er sårbare for overlading.

LPB batterier har mye av de samme egenskapene som Li-ion, men litt lavere energitetthet og levetid, samt økt motstand mot overlading. LPB tåler ikke lave temperaturer i samme grad som Li-ion.

Litium-ionebatteriet er det batteriet som tilfredsstiller flest av kriteriene. Det er det dyreste av alle, men levetiden og reduserte vedlikeholdskostnader vil tjene inn noe av den ekstra kostnaden. Det har høyest

energitetthet, kan levere medium utladestrøm (sammenliknet med de andre typene), miljøvennlig, tåler temperaturer ned mot -20°C, og har lengst levetid av alle typene.

3.6.10 Valg av kontrollsystemplattform

Ved valg av kontrollsystemplattform for en USV bør plattformen oppfylle visse kriterier og krav for funksjonalitet. I dette prosjektet har følgende kriterier blitt vektlagt:

- God mulighet for trådløs kommunikasjon
- Lavt strømforbruk
- Lav vekt
- Lav pris
- Robust og fleksibel
- Gode muligheter for utvidelse av funksjonalitet, som for eksempel autopilot, bildebehandling

Trådløs kommunikasjon er nødvendig for å kunne sende kommandoer og avlese data fra en USV. Lav vekt er viktig for et lite ubemannet overflatefartøy for å kunne maksimere nyttelasten. Siden USV-en kommer til å drives primært med batterier, er det viktig å minimere strømforbruket. Siden det er sannsynlig at USV-en kommer til å videreutvikles i senere prosjekter er det viktig at plattformen er fleksibel og robust, og at det er gode muligheter for å legge til ny funksjonalitet.

Det fins flere løsninger som kan tilfredsstille flere av kriteriene, som for eksempel PLS (programmerbar logisk styring) eller industri-PC, men slike løsninger medfører ofte en vesentlig kostnad. En kandidat som tilfredsstiller alle kravene, lav pris inkludert, er SBC, også kalt mini-PC. Disse er i prinsippet en PC i miniatyr, som har full støtte for nettverksprotokoller som TCP, IP og WiFi. De har liten formfaktor, størrelsen er sammenliknbar med en mobiltelefon, og hardwaren er ofte basert på hardware utviklet for mobiltelefoner. De kjører ofte operativsystemer som er basert på Linux-kjernen, og støtter mange forskjellige programmeringsspråk [55]. På grunn av god regnekraft og mye minne er det gode muligheter for utvidelse av funksjonalitet, som for eksempel bildeprosessering.

3.6.11 Valg av instrumenter

Siden utvikling av dynamisk posisjonering er ett av målene med prosjektoppgaven, må USV-en ha mulighet for å kunne bestemme sin geografiske posisjon og peiling. GPS er den åpenbare kandidaten for posisjonsmåling. GPS har relativt dårlig nøyaktighet (rundt 5-10 meter), så differensiell GPS er en nødvendighet. Med differensiell GPS kan posisjonen bestemmes med ca 1 meter nøyaktighet. For å bestemme fartøyets peiling kan det benyttes kompass, eller gyroskop, eller en kombinasjon av disse. IMU-er som benytter gyroskop sammen med magnetisk kompass gir god nøyaktighet, og har i tillegg mulighet for å lese av akselerasjon. Akselerasjonsdata kan for eksempel benyttes for å estimere bølgehøyde og frekvens, eller benyttes i kombinasjon med GPS for å oppnå mer nøyaktig posisjonsdata [7]. Vindmålinger er også nyttig for en USV, da enkelte DP-løsninger benytter måling av vindretning og hastighet sammen med en modell av vindens virkninger på skipet direkte, for å kompensere for vindens påvirkninger før de forårsaker at fartøyet driver av posisjonen. Dette kalles foroverkobling [56]. Avlesning av vindhastighet og retning kan også være nyttig for en operatør.

3.7 Testoppsett

For å teste prototypen i sjøen, trenger man en PC med klientapplikasjonen, en trådløs WiFi-ruter, og en joystick som er tilkoblet PCen via USB. Joysticken kan i prinsippet være av hvilken som helst type, så lenge den kan beveges i x- og y-retning, samt roteres om z-aksen. For at datamaskinen ombord i USV-en skal koble til WiFi-nettverket er man avhengig av at den er satt opp til å koble seg til det aktuelle nettverket automatisk. Datamaskinen ombord tildeles en IP-adresse av ruteren, denne IP-adressen er nødt til å være den samme som i `Client`-klassen i klientapplikasjonen. Stegene som må gjennomføres for å koble seg til USV er:

1. Slå på datamaskinen ombord, vent til den har startet opp (ca 15 sekunder)
2. Sørg for at PC er tilkoblet det trådløse nettverket
3. Start klientapplikasjonen
4. Klikk på knappen *Connect* i brukergrensesnittet
5. Når feltet *Connected to USV:* viser status *true* er man tilkoblet

Et script på PCen ombord i USV-en starter opp serverapplikasjonen automatisk under oppstart. Når man er tilkoblet kan man overvåke status på USV, samt bestemme modus slik som fjernstyring og DP gjennom brukergrensesnittet, som er beskrevet under resultater.

Kapittel 4

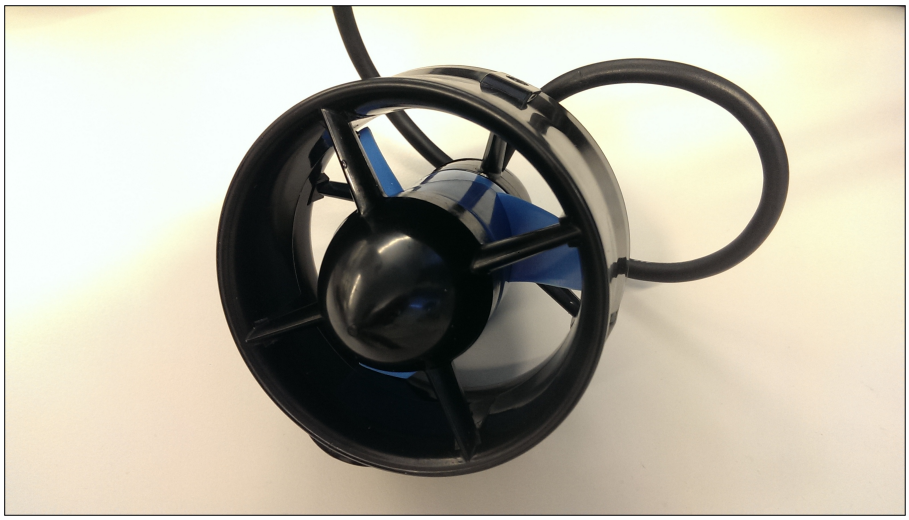
Resultater

4.1 Prototypen

I dette delkapittelet blir byggingen av prototypen basert på et Pioner 8 Mini skrog gjennomgått, og valgt utstyr beskrives. Fokus under byggingen har vært på funksjonalitet og lavest mulig kostnad. Målet med prototypen er test av DP. Pioner 8 mini ble i hovedsak valgt fordi denne modellen har en praktisk størrelse og er laget av materialet polyetylen (PE). Siden den er laget av polyetylen kunne polyetylen-rør plastsveises på skroget og brukes som thrustertuneller. Størrelsen på plastrørene er bestemt etter størrelsen på T200 thrusterene fra Blue Robotics. Dette er rimelige thrustere som yter 35N på 12V. Jolla har fire slike thrustere, to til fremdrift og to sidethrustere, en fremme og en bak.

4.1.1 Thrustere

T200-thrustere fra BlueRobotics ble valgt som thrustere for prosjektet. Thrusterene var den første store komponenten som ble kjøpt til utviklingen av prototypen. Grunnen til dette er at thrusterene har såpass lang leveringstid, fire til seks uker, at de måtte bestilles tidligst mulig. Ideelt skulle skroget vært kjøpt før thrustere siden en da lettere kan bruke skroget til å regne ut hvor stor kraft som behøves for å utføre den planlagte testen. T200-modellen ble valgt fordi den er laget spesifikt til bruk på små ROVer eller USVer. Et bilde av T200 thrusteren er vist i figur 4.1. Motoren er børsteløs og vannkjølt, og kan drives med spenning i området 12 til 16 volt DC. Den yter 35N på 12V rett ut i fra boksen, som ble regnet som nok for testing av DP på ei jolle på 8 til 10 fot. Ved å ha to thrustere til fremdrift og to til sidedrift har jolla i teorien 70 N tilgjengelig kraft uansett hvilken vei den skal. Thrusterene fra Blue Robotics styres ved hjelp av et 5V puls-bredde-signal, som går til en elektronisk motor kontroller (ESC) som følger med thrusteren fra en mikrokontroller som f.eks. Arduino. Thrusterene trenger derfor ikke å tilpasses eller ombygges og fungerer dermed rett fra boksen ved levering. En svakhet med T200-thrusterene er at det er ingen mulighet for tilbakemelding fra motorkontrolleren eller ECS-en. Man vet dermed ikke om thrusteren roterer eller ei når den gis en styrekommando. Alternativt kunne elektrisk påhengsmotor blitt brukt som motor for fremdrift. Motorene fra Endura serien til MINN KOTA har skyvekraft fra 14kg til 18kg og ligger i prisklassen fra 1900 NOK til 3500 NOK per stykk [57]. Motorene tåler å stå i og brukes i sjø over tid og har høy nok kraft til at jolla kunne brukes til andre ting enn å ligge i DP. På disse motorene følger ikke ESC med og dette måtte derfor implementeres av gruppen selv. På grunn av tidsmangel ble dette alternativet valgt bort.

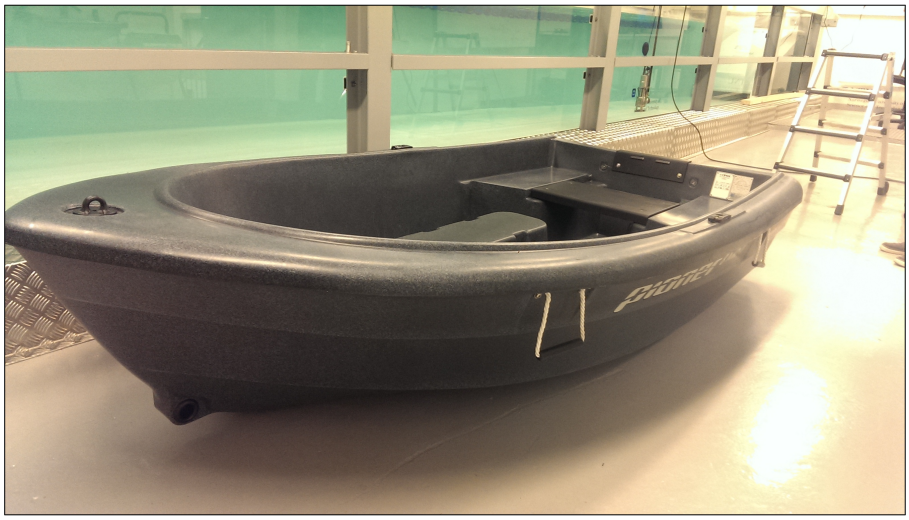


Figur 4.1: Original T200 thruster med medfølgende tunnel m/ feste.

4.1.2 Jolle

Pioner 8 Mini

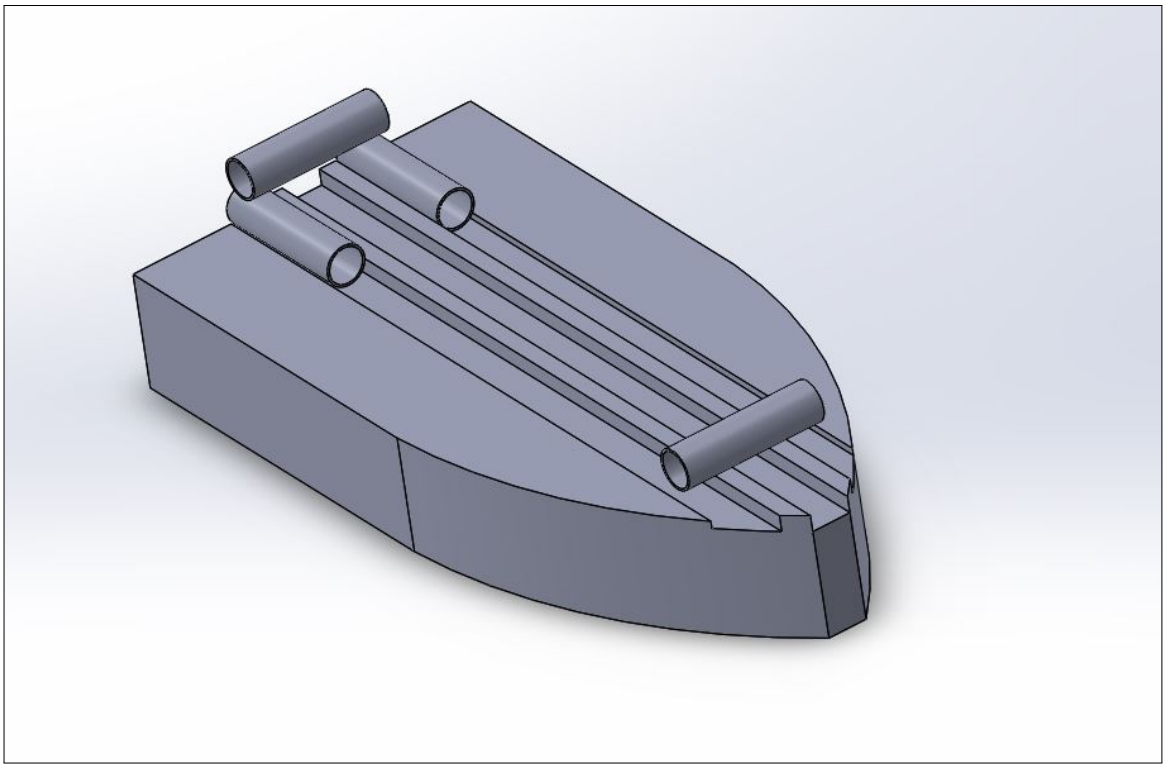
Pioner 8 mini er en jolle laget for fritidsbruk av Pioner og rommer to voksne personer. Et bilde av jollen er vist i figur 4.2. Skroget er todelt med et indre og ytre skrog av plast(polyetylen). Den veier 54 kg og dimensjonene er 242×132cm (lengde×bredde). Jollen har en maksimal nyttelast på 195 kg i henhold til merkeskilt.



Figur 4.2: Pioner 8 Mini Dark Line

Denne ble valgt som prototype på prosjektet. Jollen er ganske flat i bunnen og flyter lett i vannet. Valget av denne jollen fikk sitt utfall på grunn av flere faktorer. Det første store spørsmålet ved valg av prototype var

om det skulles bygges en jolle selv eller om det skulle kjøpes et typisk etablert skrog. På grunn av tiden til rådighet samt gruppens kunnskapsmangel innen båt og båtbygging falt valget på å kjøpe et etablert skrog. Det ble derfor søkt etter etablerte skrog på oppunder 10 fot. Det neste som måtte velges var hva slags materiale prototypen burde være laget av. De åpenbare alternativene var glassfiber eller plasttypen polyetylen. Valget falt på polyetylen siden dette er et materiale som kan plastsveises i motsetning til glassfiber som må sparkles eller limes. Dersom thrustertunneler skulle innfelles i skroget hadde det vært mye lettere å tette skroget med plastsveis kontra sparkel eller lim. Derfor ble det søkt etter joller bygd i polyetylen. Siden båtmarkedet er relativt dårlig tidlig på året var det vanskelig å finne et relevant skrog på markedet til bruktpris. Søkene på bruktmarkedet gjorde gruppen oppmerksom på Pioner joller som er bygd i polyetylen. Pioner leverer to passende modeller for prosjektet, en på åtte fot og en på ti fot. Det ble så undersøkt hos lokale båtleverandører etter jollemodellene fra Pioner. Ferd Båt AS kunne levere en Pioner 8 Mini til innkjøpspris på 8950 NOK, som er en veldig bra pris. Prisen på 10-fots modellen hadde blitt noe dyrere, og vurderingen var at 8-foteren er stor nok for prototypen. En illustrasjon av planlagt plassering av thrustertunneler er vist i figur 4.3.

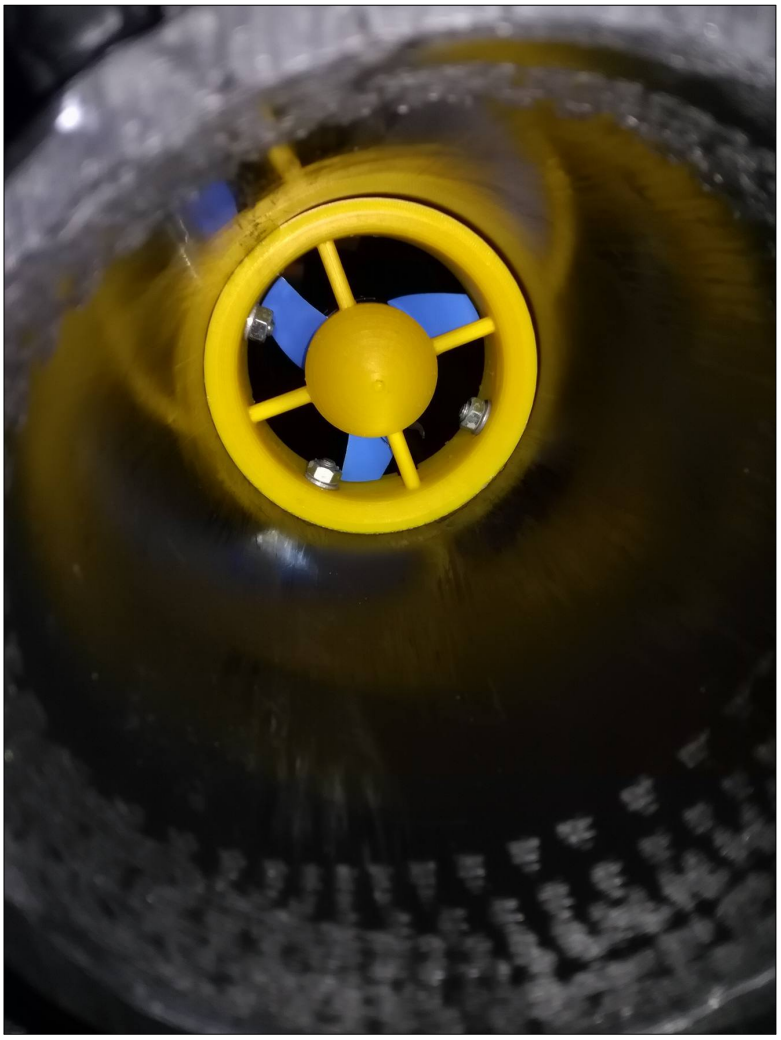


Figur 4.3: Modell av thrusterplassering på utsiden av skroget.

4.1.3 Valg og utforming av rør til thrustertunneler

Thrustertunnelene ble laget i PE100 rør som har ytre diameter på 110mm og indre diameter på 97mm. Propelldiameteren på T200 thrusterene er på 87mm. Det ble derfor valgt en indre diameter på røret på 97mm for å ha 10mm å bygge adapter på. Adapteret er til å montere thrusterene i røra. Røra ble kappet på 400mm som er $400/87 = 4.6$ ganger thrusterdiameteren som er innenfor anbefalingene til Side Power adressert i

metodekapittelet. Denne lengden var også optimal i forhold til utformingen av bunnen på jollen. Thruster montert i rør er vist i figur 4.4.

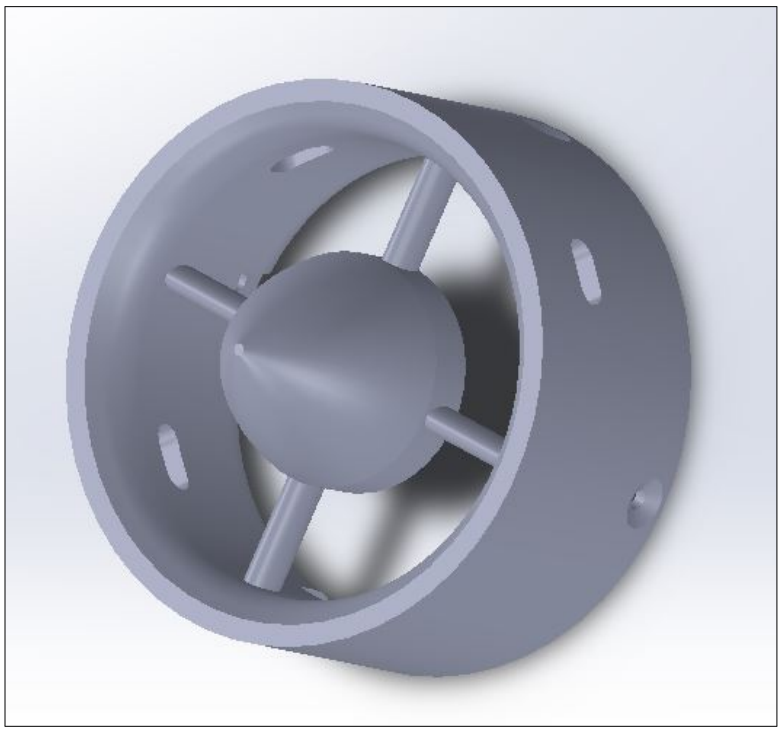


Figur 4.4: Thruster montert i påsveist tunnel

4.1.4 Plassering av thrustertunnelene

Siden skroget er såpass flatt under som det er, ble thrusterene montert på utsiden. Dette fordi at tunnelene hadde blitt like lange som bredden av båten om de skulle innfelles, og på grunn av det faktum at de sannsynligvis hadde sugd luft på grunn av at jolla flyter veldig lett. Figur 4.3 illustrer plasseringen av thrusterene på skroget. Sidethrusterene er plassert lengst mulig fra massesenteret for å få høyest mulig tilgjengelig moment ved rotasjon om z-aksen. Figur 4.3 viser også hvordan sporene som går på langs utnyttes til å plassere fremdriftstunnelene. Tunnelene til fremdrift lager derfor veldig lite drag i vannet og har av den grunn en god og kompakt plassering. Det samme kan ikke sies om sidethrusterene som begge vil bremse skroget i vannet. Dette fører til at jolla oppnår relativt lav fart ved vanlig kjøring, men dette har nesten ingen

negativ virkning ved kjøring i DP-modus.

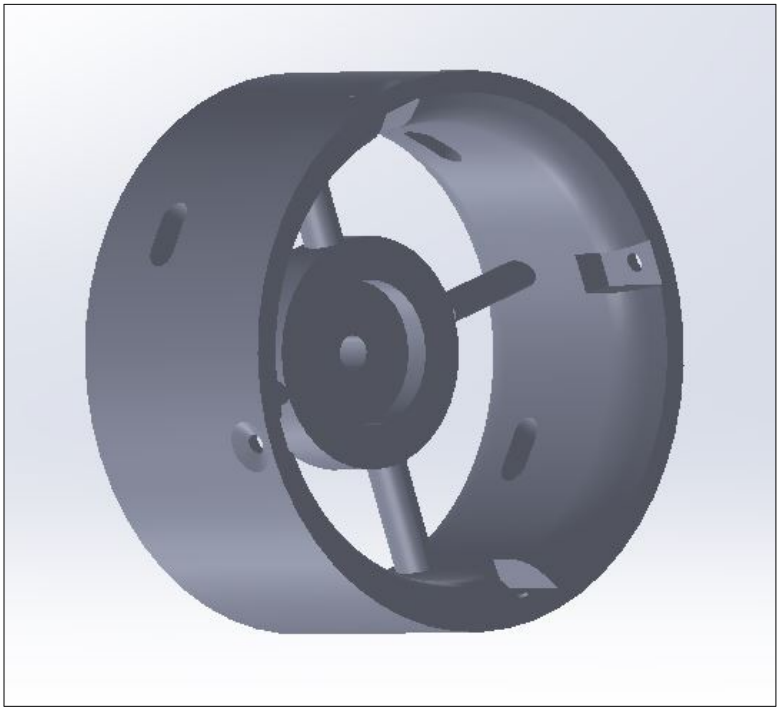


Figur 4.5: 3D-modell av adapter mellom thruster og thrustertunnelene, sett fra framsiden

4.1.5 Montering av tunneler og thrustere

Tunnelene ble plastsveist og montert som på figur 4.3 av Steinsvik AS på Straumsgjerde i Sykkylven. Resulatet kan sees på figur 4.6. De trakk seg litt på grunn av varmen uten å skape problem for videre montering. Dette kan sees på figur 4.4, som skaper gliset mellom adapteret og røret. For å få montert thrusterene i de påsveiste rørene måtte det lages et adapter mellom hver thruster og hvert rør. 3D-modellen av adapteret er vist i figur 4.5 og figur 4.7. Adapteret ble 3D-printet i plasttypen PLA. PLA tåler ikke saltvann spesielt godt, det ble derfor forsøkt å printe ut i ABS som skal tåle saltvann. Dette viste seg å være vanskelig og printen ble et mye dårlige resultat enn når PLA brukes. Siden prototypen skulle brukes i saltvann i relativt kort tid ble derfor de ferdige adapterene printet i PLA. Den lille tunnelen, vist på figur 4.1, som fulgte med T200 thrusterene ble brukt som utgangspunkt for designet. 3D-modellering ble utført i Solidworks og til 3D-print ble en Makerbot Ultimaker 2+ brukt.

Figur 4.6: Tunellplassering på Prototype

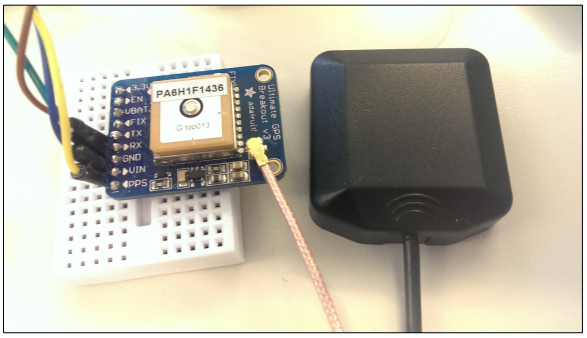


Figur 4.7: 3D-modell av adapter mellom thruster og thrustertunnelene, sett fra baksiden

4.1.6 Instrumenter

Adafruit Ultimate GPS m/ekstern Antenne

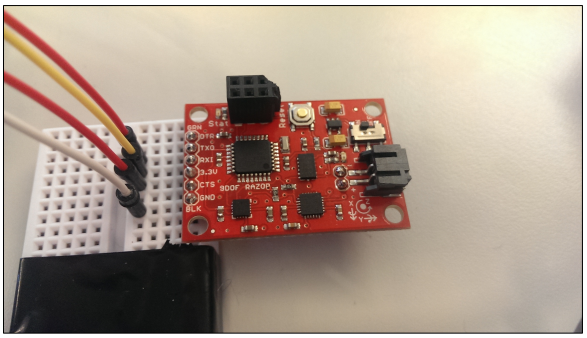
Som nevnt i kapittel 3.6.11 trenger USV-en en GPS for å bestemme sin geografiske posisjon. Instrumenteringsplattformen om bord i prototypen tar i bruk GPS med støtte for EGNOS (DGPS), for å bestemme posisjonen til prototypen. Adafruit Ultimate GPS var tilgjengelig ved universitetets laboratorium, og kan enkelt kobles til en Arduino mikrokontroller. Adafruit Ultimate GPS er en enhet som kan lese sin posisjon på jordkloden ved hjelp av satellitter i verdensrommet. Den har kan oppdatere posisjon ved 5Hz og kan følge 22 satellitter på 66 kanaler. Den har innebygget antenne og kan enkelt kobles til en Arduino mikrokontroller med seriell dataoverføring, og trenger 3-5V inngangspenning. I tillegg støtter den differensiell GPS som EGNOS (Europa) og WAAS (USA) som er et landbasert system for å korrigere feil i posisjon og gi bedre nøyaktighet (inntil ±1m) [58]. En ekstern antenne er koblet til for høyere sensitivitet. Man kan endre innstillingene i GPS-modulen ved å sende såkalte PMTK-pakker. Oppbyggingen av PMTK-pakkene er beskrevet i 3.4.3. Adafruit Ultimate GPS er vist i figur 4.8.



Figur 4.8: Adafruit Ultimate GPS

Razor 9-DOF IMU

En IMU benyttes å bestemme peilingen til USV-en i relativ til den magnetiske nordpol. Det var nødvendig å bruke en IMU fordi et gyroskop alene vil drifte over tid [59]. Gruppen valgte nettopp denne grunnet gode anmeldelser på nett og Arduino kompatibilitet. Razors 9DOF IMU består av tre typer sensorer. Et akselerometer som måler krefter, et gyroskop som måler rotasjoner og et magnetometer. IMU-en har egen ATmega 328 prosessor om bord, og er programmert til å sende seriell data til Arduino.



Figur 4.9: Razor 9-DOF IMU

Sparkfun Weather Meters

Det ble forsøkt å benytte en værstasjon fra Sparkfun til å måle vindstyrke og vindretning. Værstasjonen inneholder ingen aktiv elektronikk og må derfor forsynes av f.eks en Arduino. Vindstyrkemåleren inneholder en bryter som lukker seg for hver halve omdreining, og ikke hver omdreining som står i brukermanualen. Hvert klikk per sekund tilsvarer 0.33 m/s vindfart. Tre klikk per sekund tilsvarer dermed en vindfart på 1 m/s. Problemet med styrkemåleren er at den kan oppgi gjennomsnittlig vindfart over et tidsintervall, for eksempel siste tre sekund, mens momentan vindfart blir unøyaktig. Dersom en løsning med enkoder hadde blitt brukt istedet for antall lukkinger av en bryter, kunne momentan vindfart ha blitt oppdrevet. Momentan vindfart kunne inngått i reguleringen av dynamisk posisjonering, gjennom en foroverkobling.

Figur 4.10 viser hvilken spenning som tilsvarer hvilke retninger, gitt en spenning på 5V og resistans på 10kΩ på sensoren som måler vindretningen. Selv om den lave oppløsningen på 22.5 grader ikke er et problem i seg selv, viste det seg ved testing at 22.5 graders områdene er veldig ujevne i forhold til hverandre. Noen av stegene er i praksis opp til 30 grader mens noen er ned i 7 grader. Dette er store avvik fra 22.5 grader og vindretningsmåleren er derfor ikke egnet for hverken å måle værdata eller brukes til regulering av dynamisk posisjonering. Den ble derfor ikke implementert i prototypen.

Direction (Degrees)	Resistance (Ohms)	Voltage (V=5v, R=10k)
0	33k	3.84v
22.5	6.57k	1.98v
45	8.2k	2.25v
67.5	891	0.41v
90	1k	0.45v
112.5	688	0.32v
135	2.2k	0.90v
157.5	1.41k	0.62v
180	3.9k	1.40v
202.5	3.14k	1.19v
225	16k	3.08v
247.5	14.12k	2.93v
270	120k	4.62v
292.5	42.12k	4.04v
315	64.9k	4.78v
337.5	21.88k	3.43v

Figur 4.10: Tabell for retning, motstand og spenning for vindretningsmåler

SparkFun Atmospheric Sensor BME280

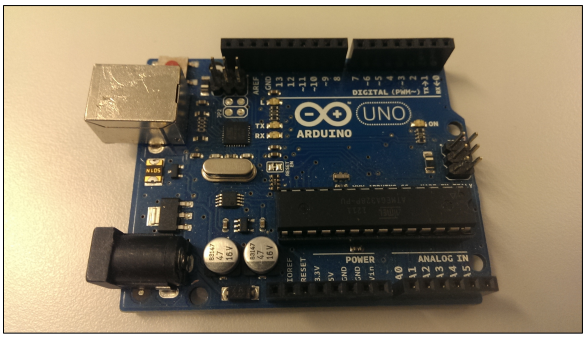
SparkFun Atmospheric Sensor måler atmosfærisk trykk fra 30kPA til 110kPa, samt relativ luftfuktighet og temperatur. Denne kommuniserer med Arduino via I^2C-protokollen. Det var planlagt å koble værstasjonen til samme Arduino, men på grunn av lav kvalitet på værstasjonen ble ikke denne installert.

4.1.7 Plassering av instrumenter

Ved plassering av instrumenter var det viktig at den eksterne antennen til GPS-en ble plassert så nært massesenteret som mulig og med fri sikt til himmelen. Det var også viktig at positiv x-akse til IMU rettes parallelt med lengderetningen til båten. Dette gjorde at posisjonen til USV-en ikke flyttet seg ved rotasjon og at peilingen til baugen ble relativ til den magnetiske nordpol. IMU er montert i samme kapsling som resterende kontrollsystem.

4.1.8 Mikrokontrollere og I/O

Arduino Uno er en mikrokontroller som bruker ATmega328P brikken utviklet av Atmel. Den har 14 digitale IO, der seks har støtte for puls-bredde-modulering, og seks analoge innganger. Brikken har klokkehastighet på 16MHz. Dimensjonene er 69x53mm. Inngangsspenning er 7-12V. Arduino ble valgt å bruke til I/O dels på grunn av at universitetet har Arduinoer tilgjengelig til utlån for prosjekter, og dels på grunn av at gruppemedlemmene har god erfaring med programmering og bruk av Arduinoer gjennom tidligere emner i studiet. Dermed kunne utviklingstiden minimeres.



Figur 4.11: Arduino Uno

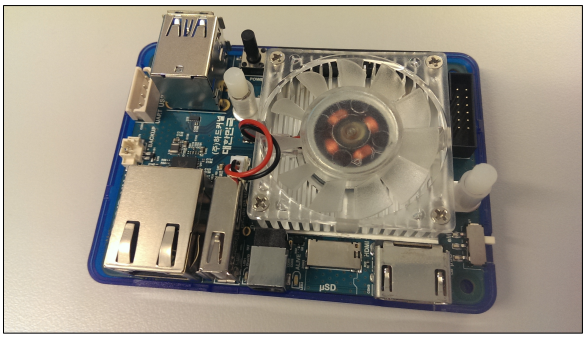
Til hver sensor er det tildelt en Arduino mikrokontroller. Det er montert tilsammen fire stk. der GPS, IMU, vær/vind/temperatur og thrusterstyring er tildelt hver sin mikrokontroller. Mikrokontrolleren for vær/vind/temperatur brukes bare til avlesning av temperatur. Grunnen til at gruppen valgte å bruke fire forskjellige Arduinoer var for å minimere kodelinjer per mikrokontroller. Når kode skulle utvikles var det lettere for gruppen å ha hver sin sensor å ta ansvar for. Det var også enklere å feilsøke i ettertid. Ved utvikling av f.eks GPS-lesingingsalgoritmer i server-applikasjonen om bord i prototypen trenger ikke alle sensorene å kobles til mikrokontrolleren. I tillegg er det, i følge GPS-produsenten, bedre med kortest mulig syklustid for lesing av GPS data på mikrokontrolleren for ikke å motta korrupte NMEA setninger [58]. En egen Arduino for GPS-modulen sørger for dette.

4.1.9 Kontrollsystem

Som beskrevet i kapittel 3.6.10 er en SBC (single board computer) et godt valg som kontrollsystemplattform. En slik løsning tilbyr svært god regnekraft i forhold til pris og størrelse. Siden universitetet har Odroid XU4 tilgjengelig for bruk i prosjekter, har en slik blitt tatt i bruk til prototypen.

Odroid-XU4

Odroid-XU4 er en liten og relativt kraftig datamaskin som kjører Linux Ubuntu OS. Den har blant annet en 8-kjerners Cortex A7 CPU, 2GB LPDDR RAM, 4 USB porter, en HDMI utgang og støtte for USB WIFI-antenne. Dimensjonene er 82x58x22mm. Inngangsspenning er 5V med et maksimalt effektforbruk på 20W [60].



Figur 4.12: Odroid-XU4

En svakhet med Odroid XU4 er at det er begrenset med muligheter for I/O. For å løse dette problemet på enklest mulig måte har det derfor blitt brukt mikrokontrollere til I/O, og disse er tilkoblet Odroid XU4 via USB. Siden prosjektet benytter 4 mikrokontrollere, i tillegg til at Odroiden bruker en USB-port til WiFi-antenne, er det koblet en USB-hub mellom Odroid og mikrokontrollere.

4.1.10 Kontrollsystemkapsling

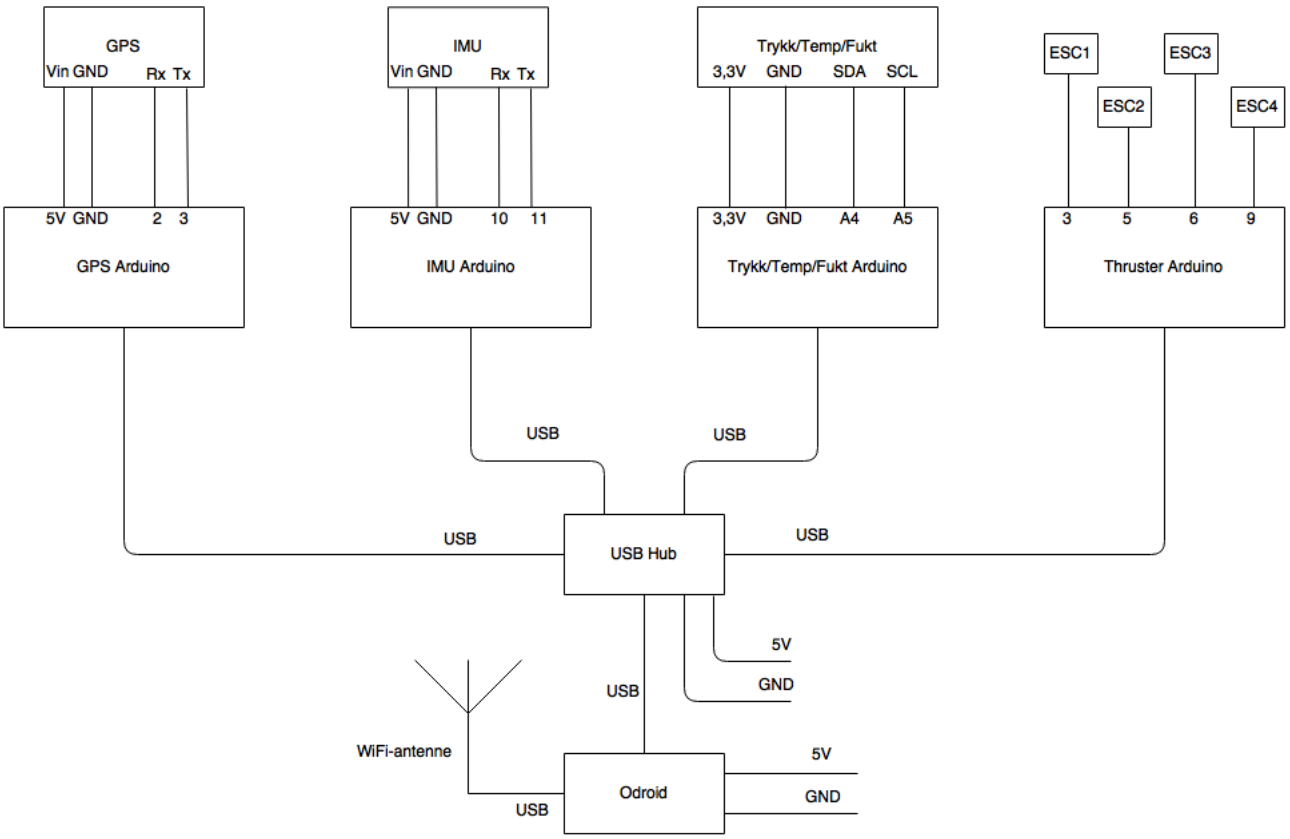
Odroid, USB-hub, mikrokontrollere og instrumenter er montert i en kapsling med kapslingsgrad IP66/67, som tilsier at den er støvtett og tåler å spyles fra alle retninger, samt midlertidig nedsenkning i vann (1 meter i 30 minutter). Kapslingen er av typen Fibox EKOE 130 G. Kabelgjennomføringer er utført med M20-nipler med kapslingsgrad IP68, som tilsier at de tåler kontinuerlig nedsenkning i vann. I bunnen av kapslingen er det skrudd fast en plate i pleksiglass. I denne platen er Odroid og USB-hub festet med borrelås, og mikrokontrollerene er skrudd fast i platen. På grunn av liten plass ble mikrokontrollerene montert i høyden, to oppå hverandre. Figur 4.13 viser montasjen i kapslingen.

Figur 4.13: Kontrollsystem

Referert til tallene i figur 4.13 er:

1. Odroid XU4 med WiFi-antenne
2. Adafruit Ultimate GPS
3. Razor 9 DOF IMU
4. Arduino mikrokontrollere
5. Temperatur/trykk/fukt sensor
6. Rekkeklemmer for PWM-signal til thrustere
7. Rekkeklemmer for 5VDC og 0VDC
8. USB hub

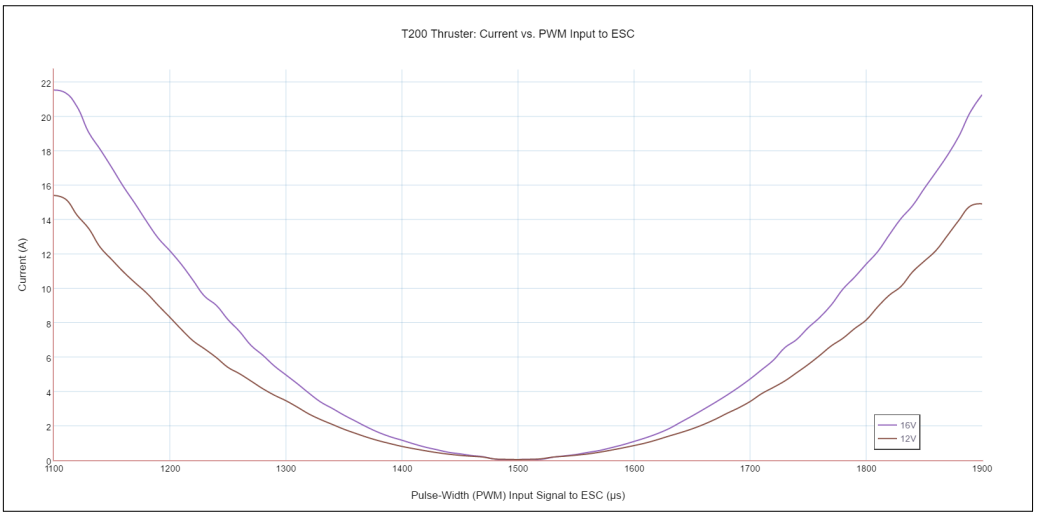
USB-kablene mellom Arduinoene og USB-huben er veldig lange. Disse ble brukt på grunn av at de var tilgjengelige ved universitetet, og medførte ingen ekstrakostnad for prosjektet. Et enkelt koblingsskjema for kontrollsystemet er vist i figur 4.14.



Figur 4.14: Koblingsskjema for kontrollsystem

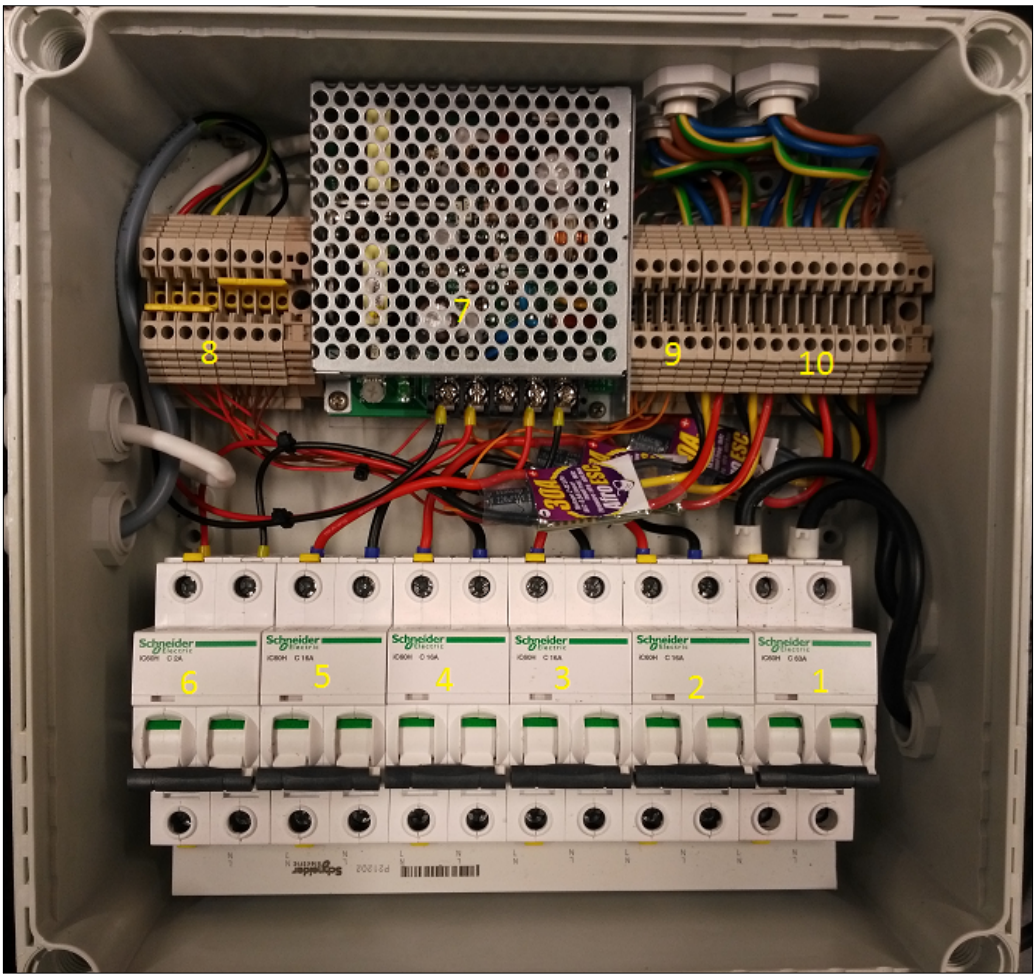
4.1.11 Kraftfordeling

Den største kraftforbrukeren i prosjektet er thrusterene. I følge dokumentasjonen til produsenten har motorene maksimalt strømtrekk på 25 Ampere ved 16 Volt [61]. I prototypen er installert spenning 12 Volt, som gir et mindre strømtrekk. I figur 4.15 vises kurven over strømtrekk ved en gitt pulsbredde for motorene, som er oppgitt av produsenten. Som man kan lese av på kurven vil strømtrekket ligge like oppunder 16 A ved maksimalt pådrag.



Figur 4.15: Strøm vs. PWM [61]

Basert på dette er det installert 16 A sikringer som kortslutningsvern for hver thruster. Siden Odroid og USB-hub drives med 5 V DC, ble det innkjøpt en 5 V 25W DC/DC-omformer fra produsenten Mean Well, for å drive kontrollsystemet. Denne konverterer en inngangsspenning i området 9,2-18 V DC til 5 V DC. Maksimal belastning for omformeren er 25 watt. 25 watt ved 12 Volt gir en strøm på $I = \frac{P}{U} = \frac{25}{12} \approx 2,1$ A. Omformeren er derfor sikret mot overbelastning med en 2 A sikring. Maksimalt strømtrekk for hele systemet blir da $16 \cdot 4 + 2 = 66$ A. Man kan anta at tilfeller hvor alle thrustere kjører med maksimal belastning er sjeldne. Sikringen mellom batteriet og resterende system er dimensjonert til 63 A, som er nærmeste standard størrelse. Selv om man kan risikere at strømtrekket i enkelte tilfeller kan overstige 63 A, har ikke dette vært et problem i praksis. Hovedstrømskjema er vedlagt i vedlegg 7. Montasje i kapslingen er illustert i figur 4.16.



Figur 4.16: Kraftfordelingsboks

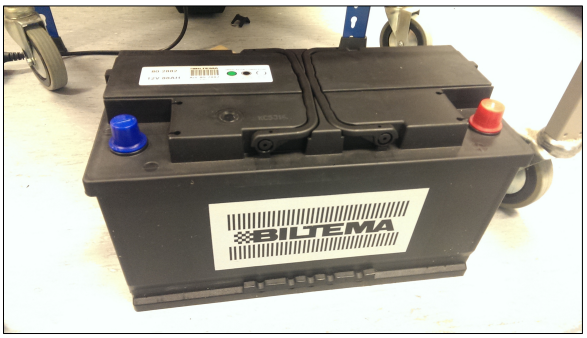
Referert til tallene i figur 4.16 er

1. Hovedsikring, 63A
2. Sikring til thruster 4, 16A
3. Sikring til thruster 3, 16A
4. Sikring til thruster 2, 16A
5. Sikring til thruster 1, 16A
6. Sikring til kontrollsystem, 2A
7. 25 W DC/DC omformer
8. Rekkeklemmer 5V DC/GND

9. Rekkeklemmer for PWM-signal til thrustere
10. Rekkeklemmer for thrustermotorer

4.1.12 Batteri

I kapittel 3.6.9 kom det frem at et litium-ionebatteri har de beste egenskapene i forhold til kriteriene som ble listet opp. På grunn av at et slikt batteri ville medført en veldig stor utgiftspost for prototypen, så ble det i stedet innkjøpt en 12 V blyakkumulator fra Biltema. Et uformelt prissøk på nett viser at en blyakkumulator koster ca. 1/10 av prisen på et litium-ionebatteri med sammenlignbar kapasitet. Det innkjøpte batteriet har en ladning på 88 Amperetimer, og dette viste seg å være nok til en hel dags testing av prototypen. Mellom testdagene ble batteriet ladet med en standard bilbatterilader.



Figur 4.17: Biltema Bilbatteri

4.1.13 Montering av utstyr i prototype

Alt av utstyr ombord i prototypen har blitt festet med borrelåsbånd. Borrelåsbånd ble brukt for at det skal være enkelt å montere og demontere utstyr for vedlikehold og lignende, og har relativt god festestyrke. Ved å bruke borrelås unngår man også større inngripen i skroget. Hvis prototypen skal bygges om, eller det skal brukes annet utstyr i fremtiden er borrelåsen relativt enkel å fjerne. Det tyngste utstyret ombord er batteriet, som veier 22,4 kg. Dette er forsøkt plassert nærmest mulig massesenteret, langs midtlinjen. På hver side av batteriet står kapslingene for kontrollsystemet og for kraftforsyningen. Figur 4.18 illustrerer dette.

Figur 4.18: Plassering av batteri og kapslinger

Kapslingen til venstre inneholder kontrollsystemet, mens den til høyre inneholder sikringer og omformer.

Kablene som følger med T200 thrusterene er relativt korte. For å nå fram til kapslingen med sikringene, måtte disse skjøtes. Dette ble gjort med fire koblingsbokser, en til hver thruster. Kabler mellom skjøtebokser og sikringer er av type PFXP 3×2,5 mm^2. Boksene er av type Hensel D9125, med kapslingsgrad IP65, som tilsier at de er støvtette og tåler spyling fra alle kanter. Figur 4.19 illustrerer plasseringen av disse boksene.

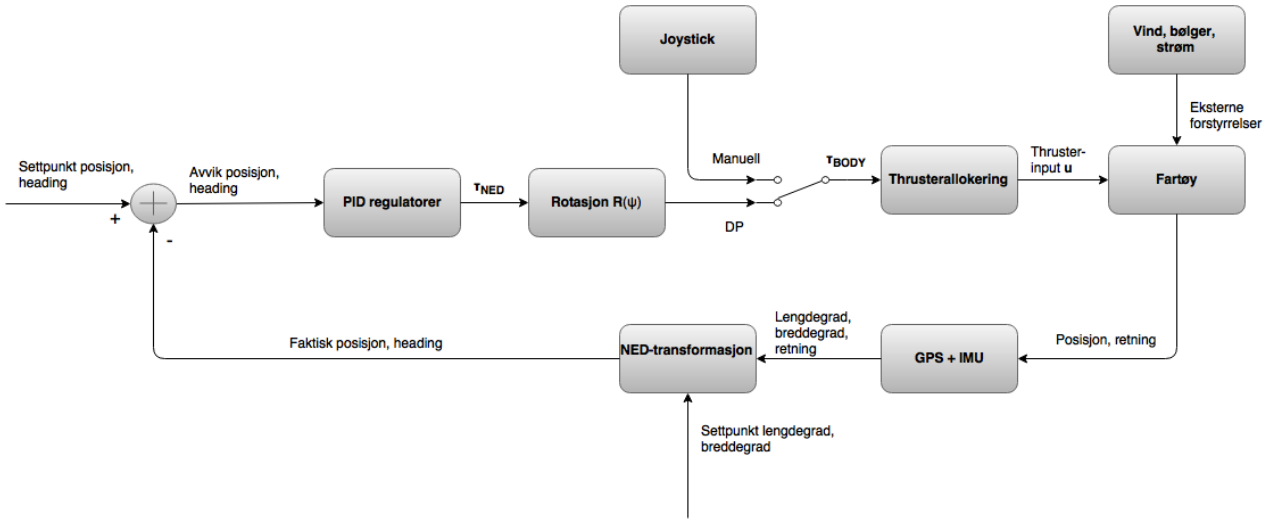
(a) Bokser for hekkthrustere (b) Boks for baugthruster

Figur 4.19: Bokser for skjøting av thrusterkabler

4.2 Dynamisk Posisjonering

Som vist i kapittel 2.7.3 er det koeffisienter i den matematiske modellen som krever systemidentifikasjon for å kunne estimeres. På grunn av mangel på tid for gjennomføring av slike forsøk har dette ikke blitt utført i dette prosjektet. Siden en matematisk modell ikke er tilgjengelig for regulatordesign, ble det i stedet benyttet 3 uavhengige PID-regulatorer, en for hver frihetsgrad som kontrolleres, i likhet med de første DP-systemene som ble utviklet på 1960- og 1970-tallet [7]. Denne vurderingen bygger på antagelsen om at systemet er stabilt, og

at koblingen mellom sideveis bevegelse og giring er av liten betydning. Som vist i kapittel 2.7 er jaging dekoplet fra sideveis bevegelse og giring. Rent intuitivt gir det mening at systemet er stabilt, siden bevegelse gjennom vann fører til positiv demping i systemet. Algoritmen som benyttes i PID-regulatorene er basert på likn. (2.49) i kapittel 2.13.3. Et oversiktsbilde over DP-systemet som er utviklet i prosjektet er vist i figur 4.20.



Figur 4.20: DP-system

Ideelt burde det vært en indre tilbakekoblingssløyfe til hver thruster også, for å regulere skyvkraften til hver enkelt thruster. Siden det ikke er mulig å lese tilbake noe signal fra thrusterene så ble ikke dette implementert.

Når fartøyet er i DP-modus så settes et punkt i geodetiske koordinater som refeanse. Det er dette punktet USV-en vil forsøke å holde seg i. En peiling i grader settes som referansepeiling. Peiling direkte nord er 0°, og øker med urviseren. GPS-koordinatet til USV-en konverteres til koordinater i NED-systemet med likningene beskrevet i kapittel 2.5.3. Referansepunktet blir origo i NED-systemet. PID-regulatorene beregner nødvendig kraft som må virke på USV via thrusterene i hver retning (nord, øst), samt moment om z-aksen. For at thrusterallokeringen skal kunne bruke denne kraftvektoren må den transformeres til BODY-koordinater med rotasjonsmatrisen som er beskrevet i likn. (2.24). Thrusterallokeringen beregner nødvendig pådrag til hver enkelt thruster basert på utgangene fra PID-regulatorene. Algoritmen for dynamisk posisjonering utfører følgende trinn:

1. Les av geodetiske koordinater fra GPS
2. Les av peiling fra IMU
3. Transformer geodetiske koordinater til NED-koordinater
4. Beregn avvik fra ønsket peiling ved å trekke faktisk peiling fra ønsket peiling
5. Gjennomfør PID-beregning for hver akse (N, E, peiling)
6. Transformer kraftvektoren fra PID-regulatorene til BODY-koordinater med rotasjonsmatrise $\mathbf{R}(\psi)$
7. Gjennomfør thrusterallokering med vektoren fra trinn 6, og finn pådrag til hver thruster

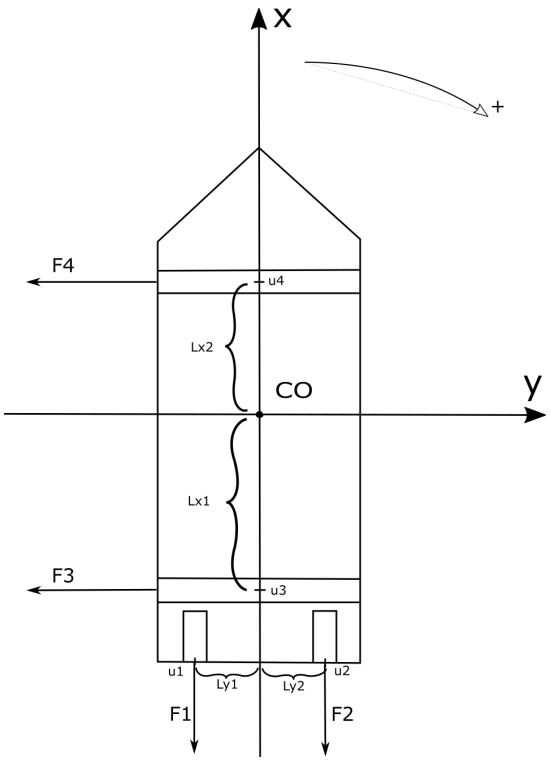
8. Gi styringskommando til hver enkelt thruster basert på resultatet fra trinn 7
9. Gjenta fra trinn 1

Inngangsverdiene til PID-regulatorene som regulerer posisjonen i nordlig og østlig retning blir dermed avvik i meter fra referansepunktet. Inngangsverdien til PID-regulatoren som regulerer peilingen er avviket i grader fra referansepeilingen. Ved å benytte integratorvirkning i regulatorene kan dette avviket styres til 0 over tid [7][25]. Siden en modell av USV-en ikke er tilgjengelig, kan parameterene i hver PID-regulator (K_p, K_i og K_d) finnes ved testing, for eksempel med Ziegler-Nichols metode beskrevet i kapittel 2.13.4.

I manuell modus sendes avleste verdier fra joystick direkte til thrusterallokeringsalgoritmen. Avleste verdier fra joystick kan dermed sees på som kraft i henholdsvis x- og y-retning, og vridningen til joysticken blir momentet om z-aksen.

4.3 Thrusterallokering

Thrusterallokering består i å finne pådraget til hver thruster basert på en ønsket kraft i henholdsvis x- og y-retning, samt et ønsket moment om z-aksen. Den ønskede kraften og det ønskede momentet er input til thrusterallokatoren, og er generert i DP-regulatoren, eller fra joysticken. Konfigurasjonen til thrusterene på jollen er illustrert i figur 4.21.



Figur 4.21: Thrusterkonfigurasjon

4.3.1 Thrusterkonfigurasjon

Input til thrusterallokeringen er kraftvektoren $\boldsymbol{\tau}_{ref} = \begin{bmatrix} X & Y & N \end{bmatrix}^T$, hvor X er ønsket skyvkraft i x-retning, Y er ønsket skyvkraft i y-retning, og N er ønsket moment om z-aksen. I et høyrehånds koordinatsystem vil positivt moment om z-aksen virke med klokka i figur 4.21. Kontrollkraften gitt av en enkelt thruster er $F = u$. Kreftene generert av hver thruster samles i en vektor $\mathbf{u} = \begin{bmatrix} u_1, u_2, u_3, u_4 \end{bmatrix}$. Thrusterenes krefter og momenter kan relateres til kontrollkraften $\boldsymbol{\tau}_{ref}$ gjennom likningen

$$\boldsymbol{\tau}_{ref} = \mathbf{Tu} \tag{4.1}$$

hvor $\mathbf{T}$ er en matrise som beskriver konfigureringen til thrusterene på fartøyet.
For u_1 har vi

$$\boldsymbol{\tau} = \begin{bmatrix} 1 \\ 0 \\ L_{y1} \end{bmatrix} u_1 \tag{4.2}$$

For u_2 har vi

$$\boldsymbol{\tau} = \begin{bmatrix} 1 \\ 0 \\ -L_{y2} \end{bmatrix} u_2 \tag{4.3}$$

For u_3 har vi

$$\boldsymbol{\tau} = \begin{bmatrix} 0 \\ 1 \\ -L_{x1} \end{bmatrix} u_3 \tag{4.4}$$

Og for u_4 har vi

$$\boldsymbol{\tau} = \begin{bmatrix} 0 \\ 1 \\ L_{x2} \end{bmatrix} u_4 \tag{4.5}$$

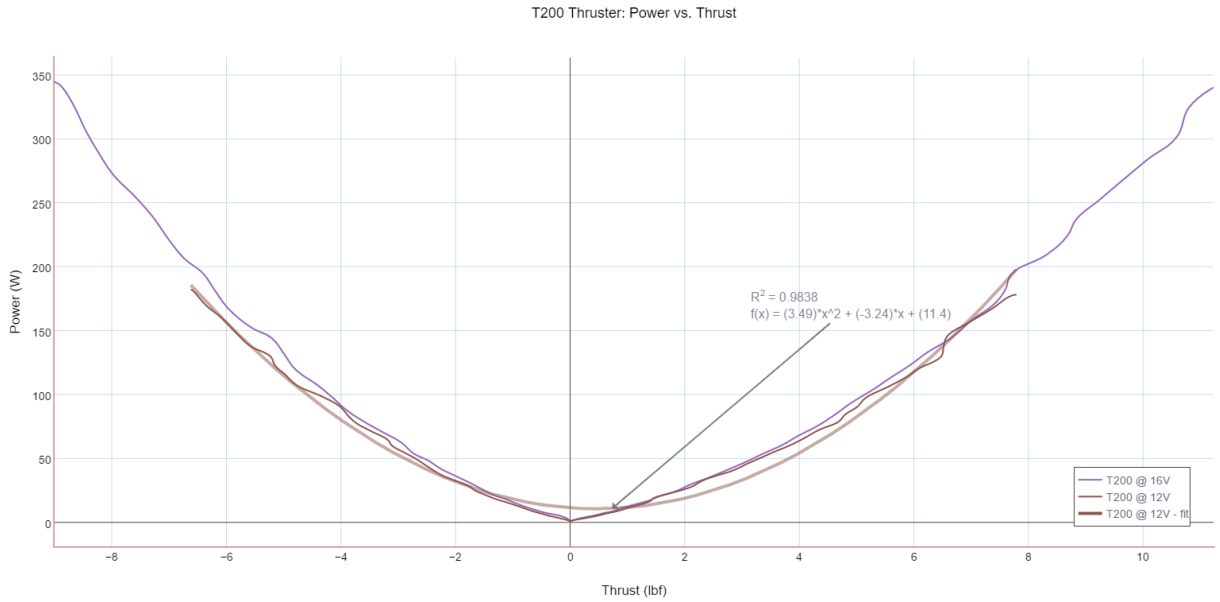
Dette gir følgende thrusterkonfigureringssystem

$$\boldsymbol{\tau}_{ref} = \begin{bmatrix} X \\ Y \\ N \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ L_{y1} & -L_{y2} & -L_{x1} & L_{x2} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \mathbf{Tu} \tag{4.6}$$

Allokeringsproblemet består i å finne verdier for $\mathbf{u}$ som tilfredsstiller likn. (4.6). Målte verdier for avstandene er $L_{x1} = 0,8$ meter, $L_{x2} = 0,87$ meter, og $L_{y1} = L_{y2} = 0,2$ meter. Massesenteret til prototypen ble funnet med vektstangprinsippet. Balansepunktet på den langsgående aksen vil være massesenteret i x-retning på figur 4.21.

4.3.2 Effekt versus skyvkraft

Produsenten av thrusterene, BlueRobotics, oppgir på sine nettsider diverse kurver for thrusterene. Blandt disse finner man en kurve for effekt versus skyvkraft. Denne er illustrert i figur 4.22, med en inntegnet tilpasset kurve for 12 V forsyningsspenning. Figuren er hentet fra [61].



Figur 4.22: T200 effekt vs. kraft

Som det framkommer av den tilpassede kurven i figur 4.22, så er effektforbruket tilnærmet proporsjonalt med konstant ganger kvadratet av skyvkraft, det vil si $P(u) \approx w \cdot u^2$, hvor w er en konstant. I figuren er skyvkraft oppgitt som lbf, men en konvertering til Newton medfører bare en skalering av konstanten w. Dette kan skrives samlet for alle fire thrusterene som

$$P(\mathbf{u}) \approx w\|\mathbf{u}\|^2 = \mathbf{u}^T\mathbf{W}\mathbf{u} \quad \text{hvor} \quad \mathbf{W} = \begin{bmatrix} w_1 & 0 & 0 & 0 \\ 0 & w_2 & 0 & 0 \\ 0 & 0 & w_3 & 0 \\ 0 & 0 & 0 & w_4 \end{bmatrix} \quad \text{og} \quad \mathbf{u} = \begin{bmatrix} u_1 & u_2 & u_3 & u_4 \end{bmatrix} \tag{4.7}$$

For å minimere effektforbruket, skal derfor likning 4.7 minimeres. Siden alle fire thrustere er like, velges matrisen $\mathbf{W}$ som identitetsmatrisen [7].

4.3.3 Løsning med Lagranges multiplikatormetode

Minimaliseringsproblemet er formulert som

$$\begin{aligned} \underset{\mathbf{u}}{\text{minimer}} \quad & \mathbf{u}^T\mathbf{W}\mathbf{u} \\ \text{med bibetingelse} \quad & \mathbf{T}\mathbf{u} = \boldsymbol{\tau}_{ref} \\ \text{hvor} \quad & \mathbf{W} = \mathbf{I}_{4\times 4} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{4.8}$$

Dette problemet løses med Lagranges multiplikatormetode, som beskrevet i kapittel 2.8.2. Lagrange-funksjonen er

$$L(\mathbf{u}, \boldsymbol{\lambda}) = \mathbf{u}^T\mathbf{W}\mathbf{u} + \boldsymbol{\lambda}^T(\boldsymbol{\tau}_{ref} - \mathbf{T}\mathbf{u}) \tag{4.9}$$

hvor $\boldsymbol{\lambda}$ er en vektor med Lagrange-multiplikatorer. Derivasjon mht. $\mathbf{u}$ gir

$$\begin{aligned} \frac{\partial L}{\partial \mathbf{u}} &= 2\mathbf{W}\mathbf{u} - \mathbf{T}^T\boldsymbol{\lambda} = 0 \\ &\Downarrow \\ \mathbf{u} &= \frac{1}{2}\mathbf{W}^{-1}\mathbf{T}^T\boldsymbol{\lambda} \end{aligned} \tag{4.10}$$

Siden $\mathbf{T}\mathbf{W}^{-1}\mathbf{T}^T$ har invers får vi

$$\begin{aligned} \boldsymbol{\tau}_{ref} = \mathbf{T}\mathbf{u} &= \frac{1}{2}\mathbf{T}\mathbf{W}^{-1}\mathbf{T}^T\boldsymbol{\lambda} \\ &\Downarrow \\ \boldsymbol{\lambda} &= 2(\mathbf{T}\mathbf{W}^{-1}\mathbf{T}^T)^{-1}\boldsymbol{\tau}_{ref} \end{aligned} \tag{4.11}$$

Innsatt i likn. (4.10) gir dette løsningen for $\mathbf{u}$ som

$$\mathbf{u} = \mathbf{W}^{-1}\mathbf{T}^T(\mathbf{T}\mathbf{W}^{-1}\mathbf{T}^T)^{-1}\boldsymbol{\tau}_{ref} \tag{4.12}$$

Siden $\mathbf{W}$ er identitetsmatrisen reduseres problemet til

$$\mathbf{u} = \mathbf{T}^T(\mathbf{T}\mathbf{T}^T)^{-1}\boldsymbol{\tau}_{ref} \tag{4.13}$$

Likn. (4.13) gir alltid en løsning for $\mathbf{u}$ for en hvilken som helst $\boldsymbol{\tau}_{ref}$. Et problem med denne løsningen er at den ikke tar hensyn til metning av thrusterene. Det vil si at en løsning for kraftvektoren $\mathbf{u}$ kan gi verdier som thrusterene aldri kan oppnå. Derfor ble denne løsningen forkastet til fordel for et konvekst optimaliseringsproblem, som også tar hensyn til arbeidsområdet til hver enkelt thruster.

4.3.4 Løsning med kvadratisk programmering og JOptimizer

For å ta hensyn til arbeidsområdet til thrusterene, må optimaliseringsproblemet reformuleres. Problemformuleringen som er benyttet for å løse allokeringsproblemet tar utgangspunkt i den presentert i delkapittel IV i [49], *Linear Quadratic Constrained Control Allocation*. Problemformuleringen er

$$\begin{aligned} \underset{\mathbf{u},\mathbf{s}}{\text{minimer}} \quad & \mathbf{u}^T\mathbf{W}\mathbf{u} + \mathbf{s}^T\mathbf{Q}\mathbf{s} \\ \text{med bibetingelser} \quad & \mathbf{T}\mathbf{u} = \boldsymbol{\tau}_{ref} + \mathbf{s} \\ & \mathbf{u}_{min} \leq \mathbf{u} \leq \mathbf{u}_{max} \end{aligned} \tag{4.14}$$

Her er $\mathbf{s}$ en slakkvariabel som er innført for å ta hensyn til de tilfeller hvor $\boldsymbol{\tau}_{ref}$ ikke kan nåes av $\mathbf{Tu}$. Betingelsen $\mathbf{u}_{min} \leq \mathbf{u} \leq \mathbf{u}_{max}$ sørger for at thrusterene ikke går i metning. $\mathbf{u}_{min}$ og $\mathbf{u}_{max}$ bestemmer henholdsvis minimums- og maksimumsverdier som thrusterene kan oppnå. Ved å velge $\mathbf{Q} \gg \mathbf{W} > \mathbf{0}$ sørges det for at slakkvariabelen blir minst mulig, og kraften $\mathbf{Tu}$ blir mest mulig nøyaktig [49]. Ved å definere

$$\begin{aligned} \mathbf{p} &= \begin{bmatrix} \boldsymbol{\tau}_{ref}^T & \mathbf{u}_{min}^T & \mathbf{u}_{max}^T \end{bmatrix}^T \\ \text{og} \quad \mathbf{z} &= \begin{bmatrix} \mathbf{u}^T & \mathbf{s}^T \end{bmatrix}^T \end{aligned} \tag{4.15}$$

så kan problemet reformuleres til formen som er vist i likn. (2.42) i delkapittel 2.8.1.

$$\begin{aligned} \underset{\mathbf{z}}{\text{minimer}} \quad & \mathbf{z}^T\boldsymbol{\Phi}\mathbf{z} \\ \text{med bibetingelser} \quad & \mathbf{A}_1\mathbf{z} = \mathbf{C}_1\mathbf{p} \\ & \mathbf{A}_2\mathbf{z} \leq \mathbf{C}_2\mathbf{p} \end{aligned} \tag{4.16}$$

hvor

$$\begin{aligned} \boldsymbol{\Phi} &= \begin{bmatrix} \mathbf{W} & \mathbf{0}_{4\times3} \\ \mathbf{0}_{3\times4} & \mathbf{Q} \end{bmatrix} \\ \mathbf{A}_1 &= \begin{bmatrix} \mathbf{T} & -\mathbf{I}_{3\times3} \end{bmatrix} \\ \mathbf{C}_1 &= \begin{bmatrix} \mathbf{I}_{3\times3} & \mathbf{0}_{3\times8} \end{bmatrix} \\ \mathbf{A}_2 &= \begin{bmatrix} -\mathbf{I}_{4\times4} & \mathbf{0}_{4\times3} \\ \mathbf{I}_{4\times4} & \mathbf{0}_{4\times3} \end{bmatrix} \\ \mathbf{C}_2 &= \begin{bmatrix} \mathbf{0}_{4\times3} & -\mathbf{I}_{4\times4} & \mathbf{0}_{4\times4} \\ \mathbf{0}_{4\times3} & \mathbf{0}_{4\times4} & \mathbf{I}_{4\times4} \end{bmatrix} \end{aligned} \tag{4.17}$$

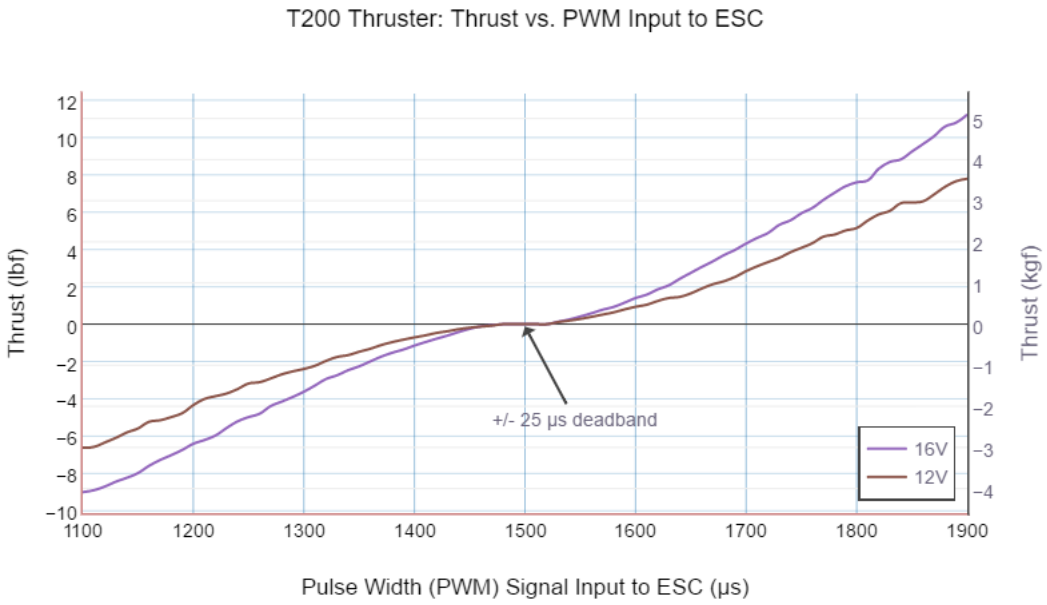
Åpen-kilkekode biblioteket JOptimizer for java er benyttet for å løse dette optimaliseringsproblemet. I klassen `ThrustAllocator` settes disse matrisene opp i konstruktøren. Deretter lages et

`PDQuadraticMultivariateRealFunction`-objekt, som er ojektivfunksjonen som skal minimeres. Dette objektet tar inn matrisen $\mathbf{\Phi}$ som parameter i konstruktøren. Et array av `ConvexMultivariateRealFunction`-objekter som bestemmer ulikhetsbegrensningene initialiseres også i konstruktøren. Til slutt initialiseres det et objekt av klassen `JOptimizer`, som tar seg av selve optimaliseringen. Dette objektet benytter en primal-dual interior-pointalgoritme for å løse det kvadratiske programmeringsproblemet [62]. En beskrivelse av denne algoritmen kan finnes i [11].

For å beregne vektoren $\mathbf{u}$ for en gitt $\boldsymbol{\tau}_{ref}$ kalles metoden `calculateOutput` i `ThrustAllocator`-klassen. Denne metoden returnerer et array, dimensjon 4, av datatypen `double`, hvor elementene i arrayet er kraften som hver enkelt thruster skal generere.

4.3.5 Konvertering fra Newton til PWM

Som nevnt i kapittel 4.3.2 har BlueRobotics diverse kurver for thrusterene. Figur 4.23 viser forholdet mellom styresignalet (PWM) inn til ESC og skyvkraften (Kgf) hver enkelt thruster skal gi. Figuren er hentet fra [61].



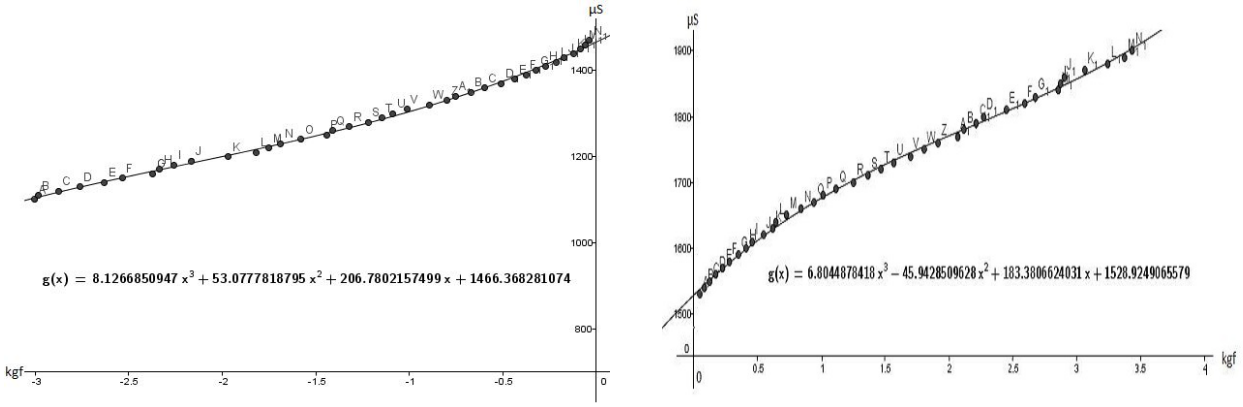
Figur 4.23: Kurve av Kgf mot PWM

Det ble utført polynomregresjon for begge sider av 1500 μs på PWM-aksen hver for seg, som resulterte i følgende to ligninger der $y_p(x)$ er funksjon for positivt rettet kraft og $y_n(x)$ er funksjon for negativt rettet kraft. Her er x kgf og y tilsvarende PWM-verdi. Regresjonen er vist grafisk i figur 4.24.

$$y_p(x) = 6.8045x^3 - 45.9428x^2 + 183.3807x + 1528.9250 \tag{4.18}$$

$$y_n(x) = 8.1267x^3 - 53.0778x^2 + 206.7802x + 1466.3682 \tag{4.19}$$

Input til likningene er i praksis Newton, her er x i Kgf. Inputen (Newton) må derfor multipliseres med $\frac{1}{g}$ for å gi riktig PWM verdi. Det er ikke noen form for datagenerert tilbakemelding for å sjekke om den individuelle thrusteren gir den skyvkraften den skal.



(a) Regresjon for negativ rettet kraft

(b) Regresjon for positiv rettet kraft

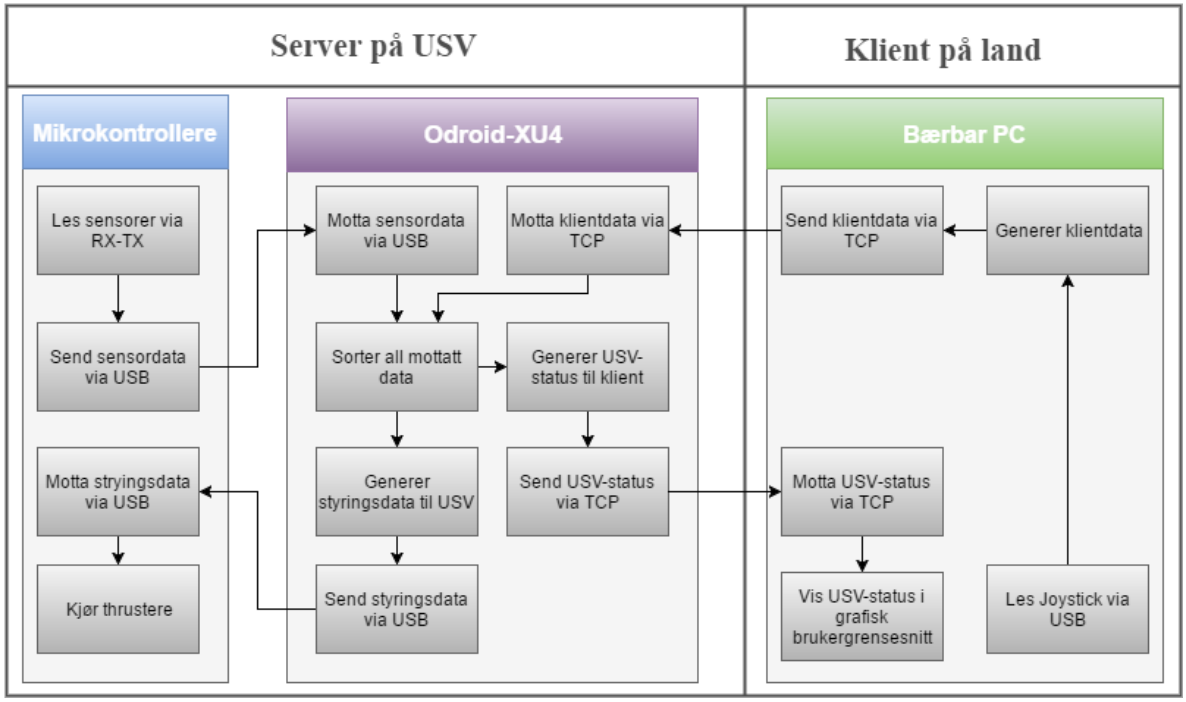
Figur 4.24: Regresjon for skyvkraft til PWM i Geogebra

4.4 Programvareløsning

For å utvikle programvare som løser den gitte oppgaven, ble problemet delt opp i mindre delproblemer, som beskrevet i kapittel 3.4. Løsningen på disse problemene beskrives i dette kapittelet. Klassediagrammer for klient- og serverside er vedlagt i vedlegg 8.

4.4.1 Dataflytskjema

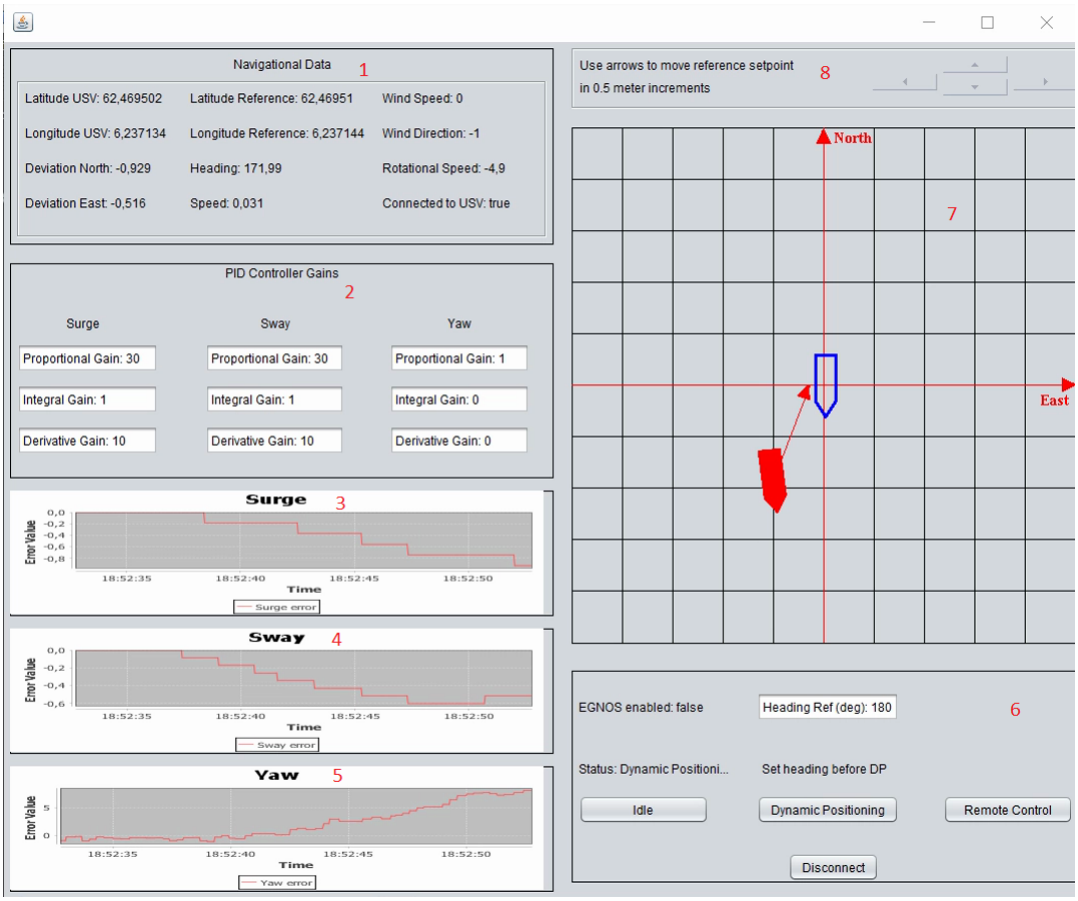
Figur 4.25 viser dataflyten i- og mellom klienten, server og tilhørende komponenter.



Figur 4.25: Dataflytskjema

4.4.2 Grafisk brukergrensesnitt

For å kunne motta og avlese data mottatt fra USV, samt generere kommandoer til USV, har det blitt utviklet et grafisk brukergrensesnitt i Java, som benytter Javas innebygde Swing-bibliotek til generering av komponenter. Under utvikling av brukergrensesnittet har fokuset vært på å lage et grensesnitt som er informativt og funksjonelt, fremfor estetisk tilfredsstillende.



Figur 4.26: Grafisk brukergrensesnitt

I figur 4.26 vises brukergrensesnittet i et tilfelle hvor USV står i DP-modus. De forskjellige komponentene i grensesnittet, med referanse til tallene inntegnet i figuren, er

1. Navigasjonsdata som GPS-posisjon, heading, tilkoblingsstatus m.m.
2. Felt for å lese av samt sette nye forsterkninger til PID-regulatorene
3. Kurve som viser avvik fra settpunkt for N-aksen (i NED-koordinater)
4. Kurve som viser avvik fra settpunkt for E-aksen (i NED-koordinater)
5. Kurve som viser avvik fra settpunkt for retningen til USV (giring)
6. Felt for å bestemme settpunkt for retning, koble til/fra USV, samt bestemme status til USV
7. Koordinatsystem som viser avvik fra settpunkt for fartøyet, samt retning til fartøyet i NED-systemet
8. Felt for å flytte settpunktet for DP-algoritmen, i 0,5 meters inkrement

I koordinatsystemet er den blå skipsboksen referansepunktet til USV-en. Den røde skipsboksen er den faktiske posisjonen til USV, som er beregnet ut fra data avlest fra GPS-en. Den røde pilen representerer kraften som er generert fra PID-regulatorene. Denne peker i retningen som USV-en vil forflytte seg, og gjør at brukeren kan følge med på hvordan DP-reguleringen oppfører seg. Hver rute i koordinatsystemet er 0,5×0,5 meter. Skipsboksene representerer ikke reell størrelse på USV-en. Koordinatsystemet er implementert som en egen klasse, `CoordinateSystem`, som arver fra `JPanel`-klassen i Swing-biblioteket. Selve brukergrensesnittet er implementert som Java-klassen `GUI`.

4.4.3 CoordinateSystem-klassen

CoordinateSystem-klassen har en metode kalt `dataUpdated` som tar inn en vektor med float-verdier. Denne metoden kalles i GUI-klassen fra metoden `updateNavDataFields`. GUI-en sender retningsreferansen, faktisk retning, avvik i henholdsvis x- og y-retning for USV-en, samt kraft generert fra PID-regulatorene i x- og y-retning som en vektor. Metoden `dataUpdated` lagrer så disse verdiene, og tegner koordinatsystemet på nytt. Basert på mottatt data vil pilen og figurene forflyttes og roteres i koordinatsystemet før de tegnes på nytt, ved å benytte tre `AffineTransform`-objekter, som gjennomfører en lineær transformasjon på figurene.

4.4.4 Klient-Server kommunikasjon

Kommunikasjon mellom klient (bærbar PC) og server (Odroid XU4) skjer gjennom en åpen TCP-forbindelse over WiFi. Grunnen til at TCP er brukt er at TCP garanterer at sendte data faktisk kommer frem, samt at en TCP-forbindelse vil forbli åpen helt til klienten lukker tilkoblingen. På klientsiden etableres forbindelsen i `Client`-klassen, ved at denne lager et nytt Socket-objekt som den skriver til og leser fra når brukeren klikker på knappen *Connect* i det grafiske brukergrensesnittet. Denne forbindelsen forblir åpen helt til brukeren klikker på *Connect* knappen igjen (som da vil ha byttet tekst til *Disconnect*). Ved å benytte systemklokken sammen med en variabel som bestemmer ventetiden styres kommunikasjonen slik at data sendes fra klienten hvert 100 millisekund. Denne begrensningen er lagt inn for å hindre at data sendes hver eneste gang klient-tråden får kjøretid på prosessoren, siden dette er både unødvendig og fører til treg oppdatering av brukergrensesnittet. På serversiden blir datapakken mottatt og tolket. Serveren sender deretter tilbake relevante data som GPS posisjon, hastighet, retning med mer. Data som sendes frem og tilbake sendes som String-objekter. Hvert datafelt skilles med et mellomrom, og på andre siden deles stringen opp ved hvert mellomrom. Data konverteres så til relevant datatype og lagres. Javas innebygde metoder for konvertering av primitive datatyper til og fra String-objekter gjør at denne løsningen er veldig lett å implementere. På klientsiden lagres all mottatt data i et eget objekt av typen `DataStorage`. Brukergrensesnittet har tilgang på dette objektet, og henter all relevant data fra dette objektet. Geodetiske koordinater sendes fra prototypen til klienten som datatypen `double`. Dette var nødvendig for å oppnå tilstrekkelig oppløsning på lengde- og breddegrad. Resterende verdier sendes som datatypen `float`, som er tilstrekkelig oppløsning for verdier som PID-forsterkning, peiling, hastighet m.m. For å vite om differensiell GPS er tilgjengelig sendes det en variabel av typen `int` som leses fra GPS-en. Denne variabelen har verdi 1 om GPS har vanlig GPS-fiks, og verdi 2 om differensiell GPS er tilgjengelig.

4.4.5 Serverapplikasjon

I programmet som kjører på Odroid ombord i USV styres logikken av `Application`-klassen. I det klienten kobler til, vil `Application` starte alle andre tråder. `Server`-klassen mottar data fra klienten på land, og setter relevante verdier i `Application`. Basert på klientens input, vil `Application` starte DP, fjernstyring, eller ligge i ro (*idle*-funksjon). Når klienten velger å koble fra, vil `Application` stoppe alle kjørende tråder, og vente på ny forbindelse.

4.4.6 Lesing av sensorer ombord i USV

GPS, IMU og vær/vind/temperatursensor sender dataene sine via hver sin mikrokontroller. Sensorene sender data over en seriell forbindelse, og SoftwareSerial-biblioteket til Arduino, beskrevet i 3.4.1, er benyttet for å lese data. I server-applikasjonen har hver sensor sin egen tråd med hver sin instanse av `SerialCommunication`-klassen som henter og lagrer dataene i sine respektive variabler. Det er implementert en `SerialPortDataListener` fra biblioteket jSerialComm for å lese data fra seriellportene via USB.

4.4.7 Behandling av sensordata om bord i USV

Kommunikasjonen mellom trådene og delte objekter er alltid via synkroniserte metoder. To NMEA-setninger leses av `GPSReader`-tråden og konverteres til et `GPSPosition`-objekt. Dette posisjonsobjektet blir lagret i det delte objektet `GPSPositionStorageBox`. `Application`-tråden henter dette og i tillegg heading fra `IMUreader`-tråden før det blir sendt til GUI-en via TCP socketen. Posisjonen har en egen lagringsboks fordi Java Marine API som tolker NMEA-setningene lagrer posisjonen i et eget objekt som er tilgjengelig for alle andre klasser. Dette objektet er ikke trådsikkert, så ved å utnytte det faktum at java sender objekter basert på verdi og ikke referanse blir dette objektet lagret i `GPSPositionStorageBox` umiddelbart etter hver NMEA-setning er lest og tolket. Dermed vil det eksistere en kopi av `GPSPosition`-objektet i lagringsboksen som er beskyttet med synkroniserte metoder og er dermed trådsikkert.
I DP-modus konverteres den geodetiske posisjonen til Nord-Øst koordinatsystemet. Metode for konvertering til nord øst koordinater fra kapittel (2.5.3) er vist under.

```
public double[] getNorthEastCoordinates(double latBody, double lonBody,
        double latRef, double lonRef) {
    double dMy = latBody - latRef;
    double dL = lonBody - lonRef;

    double rN = (R/(sqrt(1-(2*f-f*f)*sin2Pow(latRef))));
    double rM = rN * ((1 - (2 * f - f * f)) / (1 -(2*f-f*f)*sin2Pow(latRef)));
    double dN = (dMy / atan(1 / rM));
    double dE = (dL / atan(1 / (rN * cos(latRef))));
    return new double[]{dN,dE};
}
```

`latBody` og `lonBody` er båtens geodetiske posisjon i radianer, `latRef` og `lonRef` er det geodetiske referansepunktet (i radianer) til den dynamiske posisjoneringen, `R` er radius til jorden ved ekvator og `f` er jordens flattrykning. Metoden `sin2pow(double)` er sinusfunksjon kvadrert. Avvik i nord og øst er hhv `dN` og `dE`.

4.4.8 Generering av manuell styringskommando

Når USV-en opererer i fjernstyrings-modus genereres styringskommandoene av en joystick på klientsiden av applikasjonen. `JoystickReader` leser en posisjon i x og y og vridning om z-aksen og transformerer det til kreftene X og Y, og moment N. `Client` sender dataene til USV-en via en TCP Socket. I DP-modus genereres styringskommandoene X, Y og N fra PID regulatorene i `DynamicPositioning` på USV-en. Disse styringskommandoene blir input til thrusterallokeringen.

4.4.9 Dynamisk Posisjonering

Når DP blir aktivert lagres den nåværende geodetiske posisjonen (breddegrad og lengdegrad) i `GPSReader`-tråden. Dette blir referansen til posisjoneringen. Nye posisjoner lagres i `GPSPositionStorageBox`. De nye posisjonene og referansen blir konvertert til N- og E-verdier i NED med referansen som nullpunkt ved hjelp av `NEDtransform`-klassen. Verdiene for avvik i N og E blir deretter lagret i `NorthEastPositionStorageBox`. Tråden `DynamicPositioning` leser de samt verdier for peiling fra `IMUReader` og gjør de tilgjengelige for regulatorene som genererer styringskommandoer.
Når et referansepunkt for posisjoneringen er lagret kan det endres med piltast-knappene i det grafiske brukergrensesnittet. Et klikk tilsvarer å flytte referansepunktet med 0.5m i retning pilen og man vil se visuelt at referansepunktet blir flyttet. Peilingsreferansen bestemmes også i GUI-en. Et tall mellom 0 og 360 tastes inn og referansepunktet endres, dette kan også sees ved at referanseboksen roteres.

4.4.10 PID Regulering

Når USV-en er i DP-modus reguleres posisjonen og peilingen i NED-koordinatsystemet ved bruk av en diskret PID regulator for hver av de tre aktuelle bevegelsesaksene. Klassen `DynamicPositioning` oppretter alle tre regulatorene og henter, som beskrevet i 4.4.9, verdiene i nord og øst fra `NorthEastPositionStorageBox`-klassen og peiling fra `IMUReader`-klassen. `DynamicPositioning` arver fra `java.util.Timertask` for å sørge for at regulatorene kjører med fast syklustid. I hver syklus kalles `computeOutput(...)` i hvert av de tre `PIDController` objektene. Denne metoden utfører den diskrete PID algoritmen med modifikasjonen i likn. (2.49) fra kapittel 2.13.3 før den returnerer en flytverdi for en kreftene X og Y i x- og y-retning og et moment N om z-aksen. Utdrag fra `computeOutput(...)`-metoden (her med også T_s inne i summasjonstegnet):

```
//utdrag fra computeOutput(...)
error = referenceVariable - newInput;
//integrasjons leddet
integralTerm += Ki*error * cycleTimeInSeconds;
//derivasjons leddet
dError = (error - lasterror)/cycleTimeInSeconds;

//PID algoritme P + I + D
outputVariable = Kp * error + integralTerm + Kd * dError;

lastError = error;
return outputVariable;
```

Der `Kp`, `Ki` og `Kd` er henholdsvis proporsjonal-, integrasjons- og derivasjonsforsterkningen, og `error` er nåværende avvik. `integralTerm` og `dError` er henholdsvis den integrerte (multiplisert med `Ki`) og deriverte av avviket over periodetid-intervallet.

Siden thrusterallokeringen bruker BODY-koordinatsystemet må vektoren $\tau_{ref} = \begin{bmatrix} X & Y & N \end{bmatrix}^T$ fra PID regulatorene transformeres fra NED til BODY. Dette gjøres ved å multiplisere den med den transponerte av rotasjonsmatrisen $\mathbf{R}_{z,\psi}$ fra kapittel 2.6.4.

Syklustiden til `DynamicPositioning` er satt til 200 ms vha `scheduleAtFixedRate(...)` fra java.util.Timer. Metoden tar inn et `TimerTask` objekt, syklustid (ms) og starttid (ms). Dette sørger for at hver PID regulator

kjører når den skal og i bestemte intervaller. Syklustiden er bestemt utifra oppdateringsfrekvensen til posisjon på GPS-modulen som er 5Hz , som beskrevet i kapittel 4.1.6.
Ved regulering av peilingen må regulatoren ta høyde for maks- og minimumsinngang. Dette for å sørge for at båten roterer den korteste veien til referansevinkelen. Når faktisk verdi er 5 ° og referanseverdi er 355 °, skal båten rotere -10 ° i stedet for +350 °. Følgende kodesnutt, med min- og maksverdi henholdsvis 0 og 360, sørger for dette.

```
if (continuous) {
    if (Math.abs(error) > 180) {
        if (error > 0) {
        error = error - 360.0f;
        } else {
        error = error + 360.0f;
        }
    }
}
```

Her er `error` er det målte avviket og `continuous` er den logiske betingelsen som forteller om inngangsverdien kan rulle over. I prosjektet gjelder dette bare for peilingsaksen.

4.4.11 Kontroll av thrustere

I DP-modus styres thrusterene av klassen `DynamicPositioning`. Etter at PID-regulatorene har returnert verdier for kraft i nordlig og østlig retning samt moment om z-aksen, transformeres denne kraften til BODY-koordinater ved bruk av rotasjonsmatrisen $\mathbf{R}(\psi)^T$. Deretter gjennomføres thrusterallokeringen med metoden `calculateOutput(XYNtransformed)`, hvor `XYNtransformed` er den transformerte kraftvektoren. Denne metoden returnerer så en vektor med pådragsverdiene til hver thruster. Pådragene settes ved at den kaller metoden `setThrustForAll(double[4])` og `writeThrust()` i en instanse av `ThrustWriter`-klassen. Parameteret i `setThrustForAll` er et double array som inneholder krefter i newton som hver thruster skal gi. Disse kreftene er bestemt av thrusterallokeringen. Verdiene blir konvertert til tilsvarende puls-bredde-verdi ved bruk av likn. (4.18) og likn. (4.19). Puls-bredde-verdiene ligger i intervallet 1100-1900 μs. `writeThrust()` i `ThrustWriter`-klassen kaller metoden `writeThrustMicros(...)`, i dens tildelte `SerialConnection` objekt, som skriver verdiene over seriellporten.
Klassen `RemoteOperation` bruker samme instanse av `ThrustWriter`-klassen som `DynamicPositioning`. Den kaller de samme metodene for å kjøre thrusterene. Forskjellen er at kraftvektoren blir generert av joystick istedet for regulatorene.

Styring med Arduino

Arduinoen bruker fire pinner som støtter puls-bredde-modulering til å skrive verdiene til de fire thrusterene. Metoden `readStringUntil(char)` fra `Serial` objektet leser data fra den serielle forbindelsen til server-applikasjonen. Når `char` parameteret blir oppdaget lagres dataene i en String variabel. Dette skjer fire ganger til alle PWM verdier er hentet. Deretter blir de omgjort til fire primitive `int` og `writeMicroseconds(int)` fra Servo biblioteket skriver verdiene til den elektroniske motorkontrolleren.

Det var nødvendig å legge inn en 30-sekunders forsinkelse i `setup()`-funksjonen i Arduinokoden, for deretter å tømme seriellbufferen, under oppstart. Dette skyldes trolig at Odroid scanner seriellportene etter at den har startet opp, som blir tolket som en styrekommando i Arduinoen. Dette førte til at thrusterene startet å kjøre kort tid etter oppstart av Odroiden. Ventefunksjonen etterfulgt av en tømming av seriellbufferen omgikk dette problemet.

4.4.12 Oppstartsskript

Det ble laget et oppstartsskript på Odroiden som startet server-applikasjonen. Dette gjorde det enkelt koble seg til prototypen, da applikasjonen på USV starter opp ved oppstart av Odroiden. Skriptet er en .sh-fil (shell script) og legges i /etc/init.d/ på Odroiden og ser slik ut:

```
#!/bin/bash
cd /home/odroid/Desktop/BachelorUSV2016/GIT/usvapplication/dist/
java -jar USVProsjekt.jar &
```

Den andre linjen er filbanen og tredje linje starter jar filen. Etter dette må denne kommandoen utføres i terminalvinduet:

```
update-rc.d USVstartup.sh defaults
```

Der USVstartip.sh er oppstartsskriptet som nå er lagt til i oppstartssekvensen.

4.5 Resultater fra testing

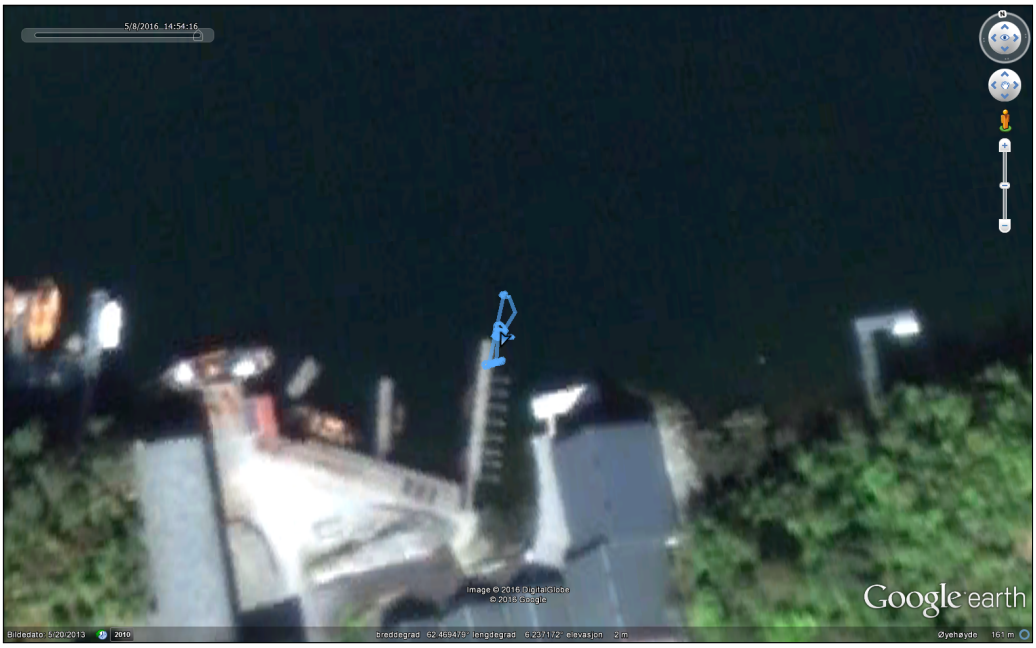
4.5.1 GPS posisjonsdata

Under testing av dynamisk posisjonering, oppstod det problemer med at GPS-posisjonsdata som ble lest fra GPS sluttet å oppdatere seg når prototypen stod helt i ro i noen sekunder. Andre GPS-data, som hastighet og retning, fortsatte å oppdatere seg. Antagelsen var derfor at det er en innstilling i selve GPS-modulen som bestemmer når posisjonsdata skal oppdateres eller ikke.

Leverandøren av GPS-modulen spesifiserer en rekke ulike såkalte PMTK-pakker i sin dokumentasjon, som er tilgjengelig på Adafruits nettsider [47]. Oppbygging av pakkene er forklart i kapittel 3.4.3. Pakketype 386 PMTK_SET_Nav Speed threshold benyttes i følge dokumentasjonen til å sette en terskelverdi på hastighet i GPS-modulen, som bestemmer hvor stor hastighet GPS-en må ha for at den skal oppdatere posisjonen sin automatisk. Terskelverdien kan settes til fast definerte verdier: 0, 0.2, 0.4, 0.6, 0.8, 1.0, 1.5 og 2.0 m/s. Ved å sende strengen "$PMTK386,0*23" gjennom Adafruit GPS bibliotekets funksjon `GPS.sendCommand()` settes terskelverdien til 0. Ved å lese tilbake verdien fra GPS-modulen med kommandoen 447 PMTK_Q_Nav_Threshold ("$PMTK447*35") ble det verifisert at terskelverdien var satt til 0 m/s. Likevel frøs posisjonsdataene når prototypen lå i ro i noen få sekunder. For å få oppdateringer igjen måtte GPS-en få litt hastighet. Basert på testingen ser det ut til at hastigheten måtte passere ca. 0,2 m/s før GPS-posisjonen ble oppdatert. Dette medførte problemer under testing, ved at prototypen posisjonerte seg til referansepunktet og lå helt stille, som igjen førte til at GPS-posisjonen sluttet å oppdatere seg. Dette førte til at prototypen drev sakte av posisjonen, uten at dette ble fanget opp av programvaren.

For å være sikker på at problemet ikke var defekt GPS-modul, ble det prøvd med to forskjellige GPS-moduler. Resultatet var identisk, med at begge frøs posisjonsdata etter å ha ligget i ro i få sekunder. Så lenge prototypen hadde litt bevegelse hele tiden ble posisjonen oppdatert.

4.5.2 Dynamisk Posisjonering

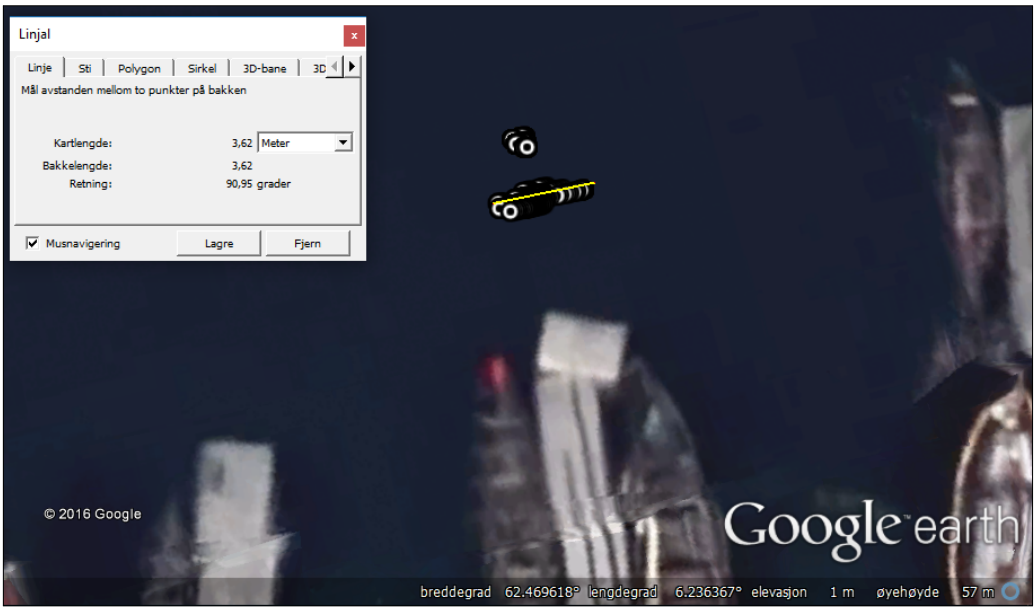


Figur 4.27: Posisjonssamling i Google Earth fra NMEA setninger

Figur 4.27 viser samling av GPS data, med EGNOS kontakt, fra NMEA setninger i Google Earth under testing av dynamisk posisjonering. I denne testen er ikke thrusterene i bruk, båten blir holdt fast i kaien i kort tid før den blir dratt videre. Dette ble gjort for å logge posisjonssamlingen fra GPS-en når båten står helt stille. GPS-en brukte mellom 0 og 5 minutter på å få EGNOS kontakt og det ga en nøyaktighet på ca 1-2m. Den blå linjen er hvor båten har vært og pilen er hvor den står (retningen er ikke nødvendigvis peilingen). Den midtre samlingen er utgangspunktet og kan lastes ned fra OneDrive-link [63].

Figur 4.28: Posisjonssamling i Google Earth med EGNOS

Figur 4.28 viser en samling fra en test av DP med EGNOS. De svarte og blå punktene er logget posisjoner mens den gule streken er lengdemåling på 1.59m.

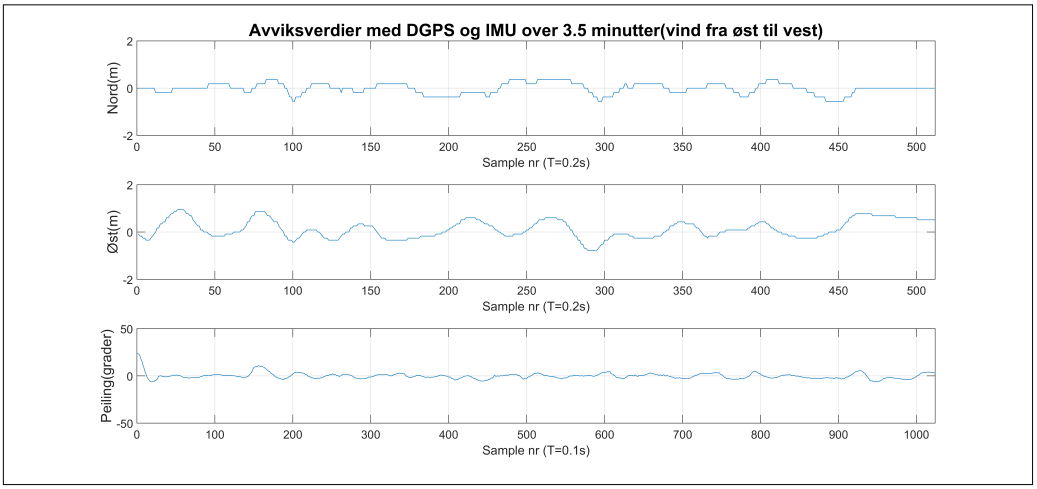


Figur 4.29: Posisjonssamling i Google Earth uten EGNOS

Figur 4.29 viser en samling fra en annen test av DP uten EGNOS. De svarte og hvite punktene er logget posisjoner mens den gule streken er lengdemåling på 3.62m.

4.5.3 PID regulering

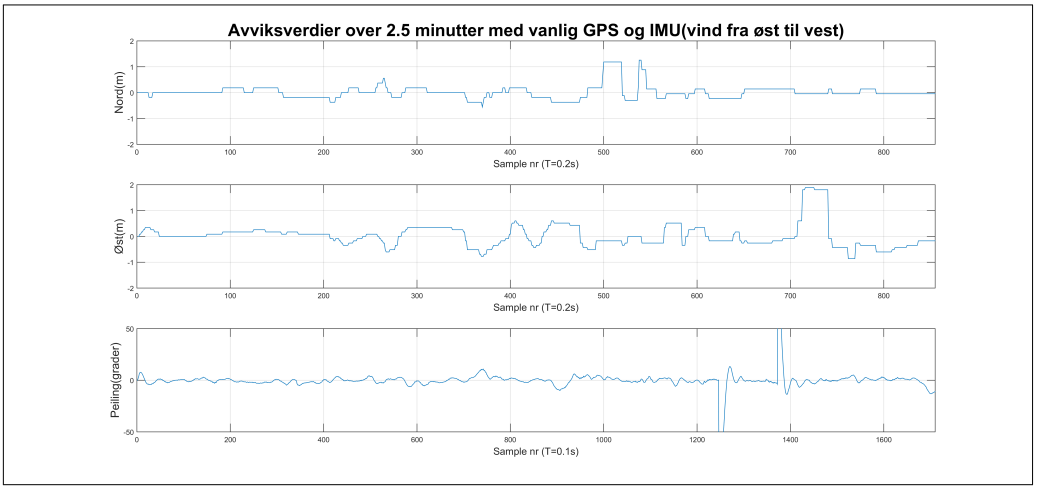
Figurene under viser tre plot av avviksverdier. Plottene viser avvik i nord, øst og peilingen der referanseverdien er 0. Målingene er tatt under tester av dynamisk posisjonering med ulike forsterkninger og er de samme målingene som de to siste fra kapittel 4.5.2. Et negativt avvik for nord eller øst betyr at prototypen beveget seg i nordlig eller østlig retning. X-aksen er sample nummer og y-aksen er avvik i meter for translasjon og grader for rotasjon.
Figuren under viser en test utført med forsterkningskoeffisienter $K_p = 30$, $K_i = 1$ og $K_d = 10$ for x- og y-verdiene og $K_p = 1$ for peiling. Antall samples n er 1024 for posisjon og $2n$ for peilingen. Periodetiden T_1 for posisjonsmåling er 0.2 sekunder og periodetiden T_2 for peilingen er 0.1 sekunder (siste plot).



Figur 4.30: Målinger i NED-koordinatsystemet med DGPS

Testendata i figur 4.30 viser at høyeste og laveste avvik i nord er målt til **0.372m** og **-0.554m**. I øst er det **0.946m** og **-0.774m**. For peiling var det **10.81°** (sett vekk i fra initial vinkelen) og **-6.21°**. I målingene tilsvarer 100 samples 20 sekunder for posisjon og 10 sekunder for peiling.

Siste test beholder samme forsterkningskoeffisienter, men testen ble utført uten bruk av DGPS. Dette fordi konktakt ikke ble oppnådd på dette tidspunktet. Antall samples n er 855 for posisjonsmåling og $2n$ for peiling.



Figur 4.31: Målinger i NED-koordinatsystemet uten DGPS

Testdata i figur 4.31 viser at høyeste- og laveste avvik i nord hhv **1.257m** og **-0.557m**. I øst er det **1.891m** og **-0.859m**. For peiling er det **13.37**° og **-13.59**°. De to toppene mellom sample nummer 1200 og 1400 er endringer av referansevinkel.
Under begge testene må det nevnes at det var relativt sterk strøm fra vest. Dette førte til at jollen aldri stod helt stille, som igjen førte til at GPS-posisjonen ikke frøs.

4.5.4 Hastighet- og krafttester

Hastighetsmålingen er utført ved å lese av verdier fra GPS under kjøring i manuell modus.
Rotasjonshastigheten er beregnet ved numerisk derivasjon av vinkelposisjonen avlest fra IMU, med formelen $\dot{\psi} = (\psi - \psi_{\text{forrige}})/\text{sampletid}$, hvor sampletid er tiden mellom avlesninger av vinkelposisjon. Kraft er målt med fjærvekt og manuell kjøring av thrustere i den aktuelle retningen. Det var noe vanskelig å lese av stabile verdier fra fjærvekten, så usikkerheten i kraftmålingen er betydelig. Resultater er vist i tabell 4.1.

Akse	Toppfart	Skyvkraft
Jaging (fremover)	0,6(m/s)	40(N)
Jaging (bakover)	0,5(m/s)	25(N)
Tverrskips styrbord	0,48(m/s)	35(N)
Tverrskips babord	0,56(m/s)	40(N)
Giring	±40(°/s)	-

Tabell 4.1: Hastighet- og krafttest tabell

Som tabellen viser, var det ikke stor forskjell i maksimal hastighet sideveis og fremover/bakover. Rotasjonshastigheten på $\pm 40°/s$ medførte at prototypen var veldig responsiv om rotasjonsaksen.

Kapittel 5

Drøfting

5.1 Resultater fra testing

5.1.1 Prototypens oppførsel i sjøen

Testing av prototypen viser at den valgte løsningen med fire fastmonterte thrustere fører til at USVen er svært manøvrerbar, som verdiene i tabell 4.1 viser. Det er små forskjeller i hastighet ved forflytning sideveis og fremover/bakover, og den roterer veldig hurtig. Svinging i fart var heller ikke et problem med fire fastmonterte thrustere, men det er usikkert hvordan en slik løsning vil virke i høyere hastighet. Løsningen med at verdier fra joystick og PID-regulatorene ble behandlet som krefter og moment i thrusterallokeringen virket veldig bra, og gjorde at prototypen var intuitivt og lett å fjernstyre. Thrusterallokeringen bidrar også til å redusere effektforbruket, med at funksjonen som minimeres er proporsjonal med thrusterenes effektforbruk, som beskrevet i kapittel 4.3.2. Det at thrustertunnelene ligger utenpå skroget i kombinasjon med at T200-thrusterene er relativt svake gjorde at hastigheten til prototypen ble ganske lav. Som diskutert i kapittel 3.6.8 kan strømhastigheten i norske fjorder overstige 0,5 m/s, som er omtrent maksimalhastigheten til prototypen.

5.1.2 Dynamisk Posisjonering

Med et dynamisk posisjoneringssystem bestående av tre PID-regulatorer og grovjusterte forsterkningskoeffisienter, klarer prototypen å beholde sin posisjon og peiling innenfor den oppgitte nøyaktigheten for differensiell GPS, som vist i figur 4.30. Dette gjaldt selv om det var relativ sterk strøm og/eller vind i sundet. Regulatorjusteringene er gjort under testing, ved å prøve seg litt frem med forskjellige forsterkningskoeffisienter og å se på båtens oppførsel fysisk og i brukergrensesnittet. Ytterlige justeringer av koeffisientene i PID-algoritmene bør føre til bedre posisjonering.

Opprinnelig var planen å benytte Ziegler-Nichols metode for innstilling av regulatorene, som beskrevet i kapittel 2.13.4, men på grunn av problemene med GPS-posisjoneringen som nevnt i kapittel 4.5.1 var det vanskelig å få til. Mye tid gikk med på å prøve å løse problemet med frysing av GPS-data, uten at en løsning ble funnet. Basert på oppnådde posisjoneringsresultat fra testing når GPS-posisjonen ikke frøs, mener gruppen at den valgte løsningen for dynamisk posisjonering tilfredsstiller kravet om posisjonering innenfor GPS-nøyaktigheten så lenge GPS-en oppdaterer posisjonsdata kontinuerlig. En annen GPS er nødvendig for at posisjonsdata ikke skal fryse.

Alternativt hadde det vært mulig å gjennomføre systemidentifikasjon for å estimere koeffisientene i likn. (2.36), likn. (2.37) og likn. (2.38) i kapittel 2.7.3, og designet en kontroller basert på en modell av prototypen. Hovedgrunnen til at dette ikke ble gjennomført er at oppgaven ble veldig omfattende, med innkjøp av utstyr, bygging av prototype, programmering og testing, slik at tiden ikke strakk til.

5.2 Programvareløsning

Serverapplikasjonen er ganske stor og ble utviklet på relativ kort tid. I et slikt stort program er det alltid et stort forbedringspotensial. Blant annet når referansepunkt endres i NED etter DP har startet vil ikke referanse bredde- og lengdegrad følge etter. Båten vil justere seg riktig, men tekstfeltene i GUI viser det ikke. En annen sak er at de tre `SerialPortDataListeners` i `SerialConnection` trenger en *Thread.sleep(int)*, som setter tråden i tilstanden TIMED_WAITING, for å få tid til å samle opp all data for å opprette en fullstendig streng. En bedre løsning ville vært unngå å at tråden sover og heller finne en metode for å lagre strengen når slutten på setningen er oppdaget.

Brukergrensesnittet virket godt, og visuell informasjon om posisjon av fartøy i tillegg til kraftvektoren fra regulatorene var verdifull informasjon under testingen. Mulighet for å endre forsterkningsverdiene i regulatorene fra grensesnittet var helt nødvendig for å kunne teste dynamisk posisjonering. Vår erfaring er at trendplottene som tegnes godt kunne vært sløyfet. Disse ga lite relevant informasjon som ikke kan leses direkte fra koordinatsystemet, og førte til unødig støy i brukergrensesnittet.

Kommunikasjon mellom klient og server virket problemfritt under hele testperioden, løsningen med å sende tekststrenger over en åpen TCP-forbindelse var stabil.

5.3 Motstandstest

For dimensjonering av fremdriftssystemet hadde det vært nyttig med en motstandstest. Vi har resultater for kraft, og resultater for hastighet i alle retninger, men en beregning av motstanden med disse dataene hadde gitt lite relevant informasjon for estimering av motstand ved en annen hastighet. Grunnen til dette er at vannmotstanden mot skroget ikke er proporsjonal med hastigheten. Som vist i kapittel 2.7.2 og i kapittel 2.10 i teoretisk grunnlag, består de hydrodynamiske kreftene av både lineære og kvadratiske ledd. Det er dermed ikke rett fram å ekstrapolere motstanden ved en gitt hastighet til en annen hastighet uten å vite de lineære og kvadratiske koeffisientene. En motstandstest i tauetank hadde vært veldig nyttig, men tauetanken ved NTNU i Ålesund er ikke dimensjonert for et fartøy på størrelse med en Pioner jolle. Med en motstandstest kunne man benyttet metodene som vist i kapittel 3.6.8 for dimensjonering av fremdriftssystemet.

5.4 Utfordringer som må utbedres

5.4.1 Utfordringer med T200 thrustere

T200-thrusterene fra BlueRobotics gir ingen tilbakemeldinger om tilstand. Man får derfor ingen diagnose fra thrusterene om verken fart eller om thrusterene kjører i det hele tatt. Thrustere bør ideelt kunne reguleres, med tilbakemelding i form av turtall, moment eller lignende. Når vi under testing fikk havari på to thrustere kunne ikke dette identifiseres på en annen måte enn at det ble observert at de ikke kjørte. Havariet skyldes brudd i ene statorviklingen på begge thrusterene. BlueRobotics har sendt nye statorer med forsterkede gjennomføringer.

5.4.2 Bedre utstyr til å måle værdata

Som forklart i kapittel 4.1.6 var ikke værstasjonen "Weather Meters" fra Sparkfun egnet til å måle værdata. Særlig vindretningsmåleren var skuffende i forhold til oppgitt spesifikasjon. Det må derfor finnes en værstasjon

med bedre spesifikasjoner. Dersom tilnærmet momentan vindfart kan måles kan en værstasjon brukes i en foroverkobling i dynamisk posisjonering siden vinden varierer stort innenfor et tidsintervall på noen få sekunder.

5.4.3 Bedre GPS

Som nevnt i kapittel 5.1.2 er frysing av GPS-posisjonsdata et stort problem under dynamisk posisjonering. Vår vurdering er at GPS-modulen fra Adafruit ikke er egnet til formålet. Siden GPS-data som er brukt i prosjektet er basert på NMEA 0183-standarden, bør det være enkelt å implementere en annen GPS i programvaren.

5.5 Mulige bruksområder for autonome ubemannede fartøy

Testing har vist at en USV med fire fastmonterte thrustere er svært manøvrerbar. Et slikt fartøy kan benyttes til manøvrering i trange områder, som for eksempel rundt merder i oppdrettsanlegg. Et mulig bruksområde kan for eksempel være å utstyre fartøyet med en liten ROV, og navigere fartøyet tett inn til en oppdrettsmerd. Ved å sette fartøyet i dynamisk posisjonering, for så å senke ned en ROV har man en god og mobil plattform som kan benyttes til inspeksjonsoppgaver, eller mindre reparasjonsarbeid. På grunn av manøvreringsdyktigheten kan det også benyttes til å frakte utstyr mellom merder i oppdrettsanlegget, eller mellom oppdrettsanlegget og andre skip.

Et annet mulig bruksområde kan være å utstyre fartøyet med autopilot, og la det gå i en forhåndsdefinert rute. Ved å utstyre fartøyet med instrumenter for analyse av vannprøver kan det benyttes for å overvåke miljøet i norske fjorder.

5.6 Det som skiller prototypen fra en fungerende løsning

5.6.1 Innstøpte tunneler

Prototypen er bygd med påsveiste thrustere på utsiden av skroget. Dette fungerer i DP-modus, siden vannmotstanden skapt av de påsveiste thrusterene er lav ettersom hastigheten i DP er lav. For at prototypen skal tas videre til å kunne brukes til vanlig kjøring, som for eksempel i en slags autopilot-modus, må thrusterene, og da spesielt sidethrusterene, innfelles i skroget. Skroget på Pioner 8 Mini er veldig flatt. Det nye skroget må være rundere i bunnen for å tilfredstille anbefalingene om thrustertunnel-lengder i forhold til diameter på thruster. En rundere bunn vil også hjelpe til med å få thrusterene dypere i vannet slik at risikoen for at de suger luft er lavere. Skroget vil da få høyere vannmotstand sideveien på grunn av det dypere skroget, sidethrusterene må derfor sannsynligvis være sterkere enn prototypen sine T200 sidethrustere. T200 fungerte derimot fint som sidethrustere på prototypen med god margin.

5.6.2 Thrustere

Med et dypere skrog, som kommer av behovet for innstøpte thrustere, må thrustertunnelene til fremdrift flyttes. Dersom det fortsatt ønskes å bruke to thruster til fremdrift kan tunnelene ligge i en bredere posisjon enn hos prototypen, der de ligger med 20 centimeter avstand fra senterlinjen til skroget. Fordelen med den implementerte løsningen er at disse tunnelene ikke vil lage noe merkbar motstand i vannet ved vanlig kjøring siden de ligger med vannstrømmen. Problemet med denne løsningen er at disse thrusterene vil bidra til større

drag når båten skal manøvrere sideveien med sidethrusterene f.eks. i DP-modus. Dette vil da kreve ekstra kraft til sidethrusterene. Et annet mulig problem med den implementerte løsningen er at det kan bli vanskelig å svinge i større fart. Dette er ikke utprøvd med prototypen på grunn av dens lave topphastighet. En mulighet kan være å montere ror bak thrusterrørene. Et godt annet alternativ kan være å bruke etablerte elektriske påhengere med en type servostyring i stedet for ror. Disse er mye sterkere enn T200-thrusterene til prototypen og er laget for å brukes i både ferskvann og saltvann. En utfordring med denne løsningen er at thrusterallokeringen til prototypen er laget for fastmonterte thrustere og må derfor modifiseres for å tilpasses en slik løsning. En måte å løse dette på kan være å låse påhengeren i en gitt servoposisjon ved kjøring i DP, slik at den oppfører seg som en fast thruster i DP-kjøring.

Det finnes lite thrustere på markedet som kan passe på båter på 8-10 fot. Det er et stort hopp fra BlueRobotics sine T-200 thrustere til thrustere som typisk brukes til daycruisere, cabincruisere og andre større båter på over 20 fot. Selv om det er mulig å kjøre slike thrustere med lavere kraft så er også vekt et problem. Det er vanskelig å finne thrustere som veier mindre enn 25 kg, som er veldig mye for et fartøy på oppunder 10 fot.

T200-thrusterene bør i følge hjemmesiden til BlueRobotics spyles med ferskvann etter bruk i saltvann. Dette er ikke ideelt for en ubemannet båt som skal brukes i saltvann over lengre tid. På grunn av disse svakhetene bør det finnes et produkt av høyere kvalitet samt med litt høyere kraft til å erstatte T200 til videre utvikling.

5.6.3 Kommunikasjon mellom operatør og USV

Til testing av prototypen brukes WiFi til trådløs kommunikasjon av data. Konseptuelt må dette erstattes med en form for kommunikasjon enten over satellitt eller over 4G-nett slik at fartøyet kan kommuniseres med og overvåkes over lengre avstander. Dersom videostrømming skal implementeres er 4G et godt alternativ siden det er det samme nettet vi bruker på telefonene våre til daglig bruk, og er av erfaring stabilt og raskt.

5.6.4 Autopilot

Den endelige løsningen bør ha en form for autopilot for navigering. USV-en må kunne bevege seg langs bestemte baner, over både korte og lange avstander. Kartplotter kan være en løsning for å generere en slik bane. Algoritmene for flat jordnavigering for NED-koordinatsystemet brukt i prosjektet vil ikke være nøyaktig nok over lengre avstander.

5.6.5 Utstyr for diverse oppdrag

For drift i lengre tid kan det være hensiktsmessig å installere et diesel-/bensinaggregat og/eller solcellepaneler for å lade batteriene om bord. Det kan også installeres en vinsj for lasting og nedsenking av en liten ROV for inspeksjon av marine anlegg. Lasterommene kan også brukes til lagring av prøver av det marine miljøet den opererer i. For oppdrag i mørket er det kritisk å ha en navigasjonsradar. En lading- og oppbevaringsplass for drone vil gi kunne gi konseptet overvåkningsmuligheter under vann, på vann og i luften.

5.7 Erfaringer fra prosjektet

5.7.1 Arbeidsfordeling

Gruppemedlemmene har forskjellige bakgrunner fra blant annet mekaniske fag og elektrofag. De forskjellige bakgrunnene har derfor blitt aktivt brukt i arbeidsfordelingen for å utnytte hvert medlem sin spisskompetanse. Dette har spart oss mye tid og gitt oss muligheten til å gjennomføre et så stort og tverrfaglig prosjekt.

5.7.2 Fossefall - Ganttdiagram

Det ble brukt Gantt-diagram som aktivitetsplanlegger for prosjektet. Gruppen prøvde å estimere hvor lang tid hver aktivitet kom til å ta og satte opp diagrammet tidlig i planleggingsfasen. Selv om det ble mange aktiviteter stemte tidsestimatene veldig bra for prosjektet i sin helhet. Noen av aktivitetene tok lenger tid enn planlagt, men dette ble kompensert med at andre aktiviteter tok kortere tid enn planlagt.

5.7.3 Risiko

For å gjennomføre prosjektet var det viktig å adressere eventuelle risikoer som kunne hindre prosjektet, både underveis og med tanke på resultatet. Det ble derfor laget en risikorapport i Excel, som var et levende dokument gjennom prosjektperioden. Risikorapporten ligger som vedlegg til hovedrapporten.

5.7.4 Ekstern hjelp

Der gruppen har møtt utfordringer har ekstern hjelp blitt benyttet for å spare tid. Dette har særlig skjedd på felt og områder som gruppen i utgangspunktet har lite kompetanse om, som ikke er direkte relevante til fagområdet vårt. Fablaben ved NTNU i Ålesund bidro sterkt med sin kompetanse innen 3D-printing av adaptere til montering av thrustere i thrustertunneler. For sveising av thrustertunnellene ble Steinsvik Group A/S ved Straumsgjerde i Sykkylven brukt. Ellers har Jostein Berge på Skipsdesign avdelingen ved NTNU i Ålesund bidratt med tips angående materialvalg på prototype og tips om at Steinsvik Group A/S har plastsveis som et av sine spesialområder.

5.7.5 Versjonskontroll ved programvareutvikling

Versjonskontroll med bruk av Bitbucket og Git var høyst nødvendig for dette prosjektet. Alternativet til en slik versjonskontrolltjeneste ville vært å dele en mappe på OneDrive for deretter å importere manuelt hver gang det skulle gjøres arbeid på applikasjonen. Dette ville begrense programmerere til kun en i gangen. Med Git/Bitbucket kunne alle jobbe samtidig. Dette gjorde også at siste versjon av kildekoden enkelt kunne lastes ned på Odroiden før testing på sjø, og det var enkelt å spore endringer og gå tilbake til tidligere versjoner om nødvending.

5.8 Utfordringer og gråsoner med ubemannede overflatefartøy

En stor utfordring med autonome overflatefartøy er hvordan fartøyet skal håndtere og oppføre seg ovenfor andre fartøy. Det er viktig å unngå kollisjon med båter og andre hindringer siden dette kan føre til både personskader og materielle skader [64]. Uavhengig om et ubemannet fartøy opererer i høy eller lav fart vil de samme reglene gjelde for et ubemannet fartøy som for andre menneskestyrte båter angående vikeplikt og plassering i sjø. Et ubemannet fartøy er derfor avhengig av menneskelig overvåkning og styring. Den mest naturlige måten å overvåke et slikt fartøy på er videooverføring. Dette fører til det neste problemet. Hvordan skal et ubemannet fartøy oppføre seg dersom den mister kommunikasjonen med operatøren? Den kunne eventuelt kastet ut en dregg slik at den holdt seg i sist kjente posisjon, og operatøren kan lokalisere og hente den. Problemet med dette er at dybden gjerne er for stor til å bruke dregg og den vil lett kunne slites ved værforandringer dersom den ligger over lengre tid. Dette begrenser bruksområdet for et ubemannet fartøy til kjente farvann der operatøren relativt fort kan nå fartøyet fysisk. System for å unngå kollisjon med andre båter er fremdeles utilregnelige og kostbare [48]. Kostnaden kommer i form av sensorer og dyr utvikling. Dersom et ubemannet fartøy skal kjøre i autonom modus uten assistanse må dette være i et relativt kjent

miljø som for eksempel rundt et oppdrettsanlegg. I et slikt bruksområde kan det lages lettere smarte løsninger som detekterer og forhindrer eventuelle kollisjoner som passer den spesifikke jobben fartøyet skal utføre. Dette punktet går utenfor fokuset til denne oppgaven.

Mye av de samme problemstillingene og mulighetene ligger på større fartøy. Å gjøre store fartøy ubemannede vil ta jobben fra sjøfolk. Samtidig vil det åpne for at sjøfolk kan ta nye jobber som landbaserte skipsoperatører. Å flytte jobbene på land vil ikke bare være lønnsomt, det vil også øke sikkerheten. Det finnes fortsatt ikke klare nok rammeverk og lovverk for å gjennomføre dette, men det blir spekuleres om at ubemannede skip vil komme på markedet i løpet av 10 år [65].

Kapittel 6

Konklusjon

I kravspesifikasjonene var det stilt krav om at prototypen skal kunne ligge i fast posisjon med konstant peiling (innenfor nøyaktigheten til GPS) ved hjelp av dynamisk posisjonering. Siden differensiell GPS har en oppgitt nøyaktighet på ± 1 meter, viser resultatene fra testingen, som er presentert i kapittel 4.5.3, at løsningen med tre PID-regulatorer og GPS som posisjonsreferanse kan tilfredsstille dette kravet. Grunnet problemer med oppdateringer av GPS-posisjon vil det være nødvendig med en annen GPS enn den implementerte i en endelig løsning. Kravet om å kjøre prototypen til ny posisjon og legge den i ny fast posisjon med konstant peiling er møtt ved at prototypen kan styres med joystick i manuell modus, og dynamisk posisjonering kan aktiveres og deaktiveres fra et grafisk brukergrensesnitt. Alternativt kan settpunkt endres stegvis i DP-modus, og fartøyet forflytter seg til ny posisjon automatisk.

Viktige designkriterier ved utvikling av en USV er presentert i metodekapittelet, og den utviklede prototypen har gitt erfaringer vedrørende løsninger som kan bidra i den videre utviklingen av et ubemannet overflatefartøy. Prototypen som er utviklet var i hovedsak bygd for uttesting av dynamisk posisjonering og øvrig kontrollsystem, men den har også gitt erfaringer rundt design av ubemannede overflatefartøy generelt.

De viktigste erfaringene er at tunneller for sidethrustere bør innfelles i skroget, i stedet for den implementerte løsningen med utenpåliggende rør, siden utenpåliggende rør fører til stor motstand i vannet. Det vil også være svært nyttig å gjennomføre en motstandstest i slepetank for dimensjonering av fremdriftssystemet, da de innkjøpte thrusterene er for svake for annet enn dynamisk posisjonering. Den implementerte løsningen med fire fastmonterte thrustere førte til at prototypen ble svært manøvrerbar, men det er knyttet usikkerhet til hvordan den er å styre i større hastigheter.

Prototypen viser også at en mini-PC som Odroid kan fylle rollen som kontrollsystemplattform, og en løsning for dynamisk posisjonering kan implementeres i Java ved hjelp av innebygde mekanismer for sanntidsprogrammering og eksterne biblioteker. Spesielt løsningen av thrusterallokeringsproblemet ved bruk av kvadratisk programmering og JOptimizer har potensiale til å kunne benyttes videre.

Gruppen opplever prosjektet som en suksess, og selv om det til tider har blitt lange dager og arbeidsmengden føltes uoverkommelig, var alt slitet verdt det når man endelig fikk se resultatet på sjøen. Prosjektet har gitt gruppemedlemmene god erfaring i planlegging og gjennomføring av prosjekter, og har knyttet sammen mange av de ulike emnene gjennom tre års studier til en helhet. Prosjektet har også gitt gruppen mye ny kunnskap, ikke bare innen automasjonsfaget, men også kunnskap som man kanskje ellers ikke ville oppnådd, på grunn av prosjektets tverrfaglighet.

6.1 Videre arbeid

Det er mye man kan tak i videre.

- Et annet skrog bør anskaffes eller utvikles på grunn av vanskeligheten med å innfelle thrustertunneller i Pioner jollen.
- Andre fremdriftsløsninger bør vurderes, på grunn av lite skyvkraft i T200-thrusterene.
- En slepetest vil være nyttig i dimensjonering av fremdriftssystemet. Metodene vist i kapittel 3.6.8 kan benyttes for dimensjonering.
- Annen GPS-løsning må implementeres.
- Innstilling av DP-regulatorparametere ved hjelp av Ziegler-Nichols metode, eller andre metoder. Alternativt kan det gjennomføres systemidentifikasjon som beskrevet i [9] og i [10], og benytte modellen i kapittel 2.7.3 for utvikling av ny DP-regulator.
- Uvikle autopilotløsning for autonom forflytning.
- Utvikle regulatorer for å regulere kraften fra hver thruster.

Kapittel 7

Referanser

[1] *Lecture Notes TTK4190 Guidance and Control of Vehicles*. 2015. URL: http://www.fossen.biz/wiley/Ch2.pdf.

[2] A.J. Sørensen. «A survey of dynamic positioning control systems». I: *Annual Reviews in Control* 35 (2011), s. 123–136.

[3] *SlideShare, DP-Klasser*. 2016. URL: http://www.slideshare.net/LREnergy/dp-pm-awareness-and-training.

[4] *Dynamisk Posisjonering, Store Norske Leksikon*. 2009. URL: https://snl.no/dynamisk_posisjonering.

[5] D. S. Bernstein. *Geometry, Kinematics, Statics, and Dynamics*. Princeton University Press, 2012.

[6] The Society of Naval Architects og Marine Engineers. *Nomenclature for Treating the Motion of a Submerged Body Through a Fluid*. Tekn. rapp. April 1950.

[7] T. I. Fossen. *Handbook of Marine Craft Hydrodynamics and Motion Control*. John Wiley & Sons, Ltd, 2011.

[8] *Estimate flat earth position from geodetic latitude, longitude and altitude, MathWorks*. 2011. URL: http://se.mathworks.com/help/aerotbx/ug/lla2flat.html?requestedDomain=www.mathworks.com.

[9] R. Aasen og B. Hays. «Method for Finding Min and Max Values of Error Range for Calculation of Moment of Inertia». I: *69th Annual Conference Of Society of Allied Weight Engineers, Inc* 3504 (2010).

[10] H.K. Yoon og K.P. Rhee. «Identification of hydrodynamic coefficients in ship maneuvering equations of motion by Estimation-Before-Modeling technique». I: *Ocean Engineering* 30.18 (2003), s. 2379–2404.

[11] S. Boyd og L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2009.

[12] R. A. Adams og C. Essex. *Calculus: A Complete Course*. Pearson, 2013.

[13] O. M. Faltinsen. *Hydrodynamics of High-Speed Marine Vehicles*. Cambridge University Press, 2005.

[14] Anthony F. Mollard, Stephen R. Turnock og Dominic A. Hudson. *Ship Resistance and Propulsion*. Cambridge University Press, 2013.

[15] R. P. Feynman, R. B. Leighton og M. Sands. *The Feynman Lectures on Physics, Vol. II*. Addison-Wesley, 1964.

[16] A.S. Tanenbaum og H. Bos. *Modern Operating Systems*. Pearson, 2015.

[17] *Java, TechTarget*. 2007. URL: http://searchsoa.techtarget.com/definition/Java.

[18] *Processes and Threads*. 2000. URL: https://docs.oracle.com/javase/tutorial/essential/concurrency/procthread.html.

[19] *Thread State*. 2000. URL: https://docs.oracle.com/javase/7/docs/api/java/lang/Thread.State.html.

[20] A. Wellings. *Concurrent and Real-time Programming in Java*. John Wiley & Sons, Ltd, 2014.

[21] *Thread-klasse, Oracle*. 1993. URL: https://docs.oracle.com/javase/7/docs/api/java/lang/Thread.html.

[22] *TimerTask-klasse, Oracle*. 1993. URL: https://docs.oracle.com/javase/7/docs/api/java/util/TimerTask.html.

[23] D. J. Barnes og M. Kölling. *Objects First With Java*. 5. utgave. Pearson, 2012.

[24] *TimerTask-klasse, Oracle*. 1993. URL: https://docs.oracle.com/javase/7/docs/api/java/util/Timer.html.

[25] N. S. Nise. *Control Systems Engineering*. Sixth edit. John Wiley & Sons, 2009.

[26] F. A. Haugen. *Reguleringsteknikk*. 2. utgave. Fagbokforlaget, 2014.

[27] *Project Blog, Improving the Beginner's PID*. 2007. URL: http://brettbeauregard.com/blog/2011/04/improving-the-beginner\%E2\%80\%99s-pid-tuning-changes/.

[28] *EGNOS, European Space Agency*. 2013. URL: http://www.esa.int/Our_Activities/Navigation/The_present_-_EGNOS/What_is_EGNOS.

[29] *GPS, Store Norske Leksikon*. 2015. URL: https://snl.no/GPS.

[30] *United States Patent: 4711125*. 1987. URL: https://docs.google.com/viewer?url=patentimages.storage.googleapis.com/pdfs/US4711125.pdf.

[31] University of Cambridge. *An introduction to inertial navigation*. Tekn. rapp. Number 696.

[32] J. F. Kurose og K. W. Ross. *Computer Networking A Top-Down Approach*. Sixth edit. Pearson, 2013.

[33] *USB, Store Norske Leksikon*. 1987. URL: https://snl.no/USB\%252FIT.

[34] S. Chacon og B. Straub. *Pro Git*. Apress, 2014.

[35] *Bitbucket*. 2016. URL: https://bitbucket.org/.

[36] *What's the best battery, Battery University*. 2016. URL: http://batteryuniversity.com/learn/article/whats_the_best_battery/.

[37] D. Linden og T.B. Reddy. *Handbook Of Batteries*. McGraw-Hill, 2001.

[38] K. Young mfl. «Electric Vehicle Battery Technologies». I: *Electric Vehicle Integration into Modern Power Networks*. Red. av R Garcia-Valle og J.A. Pecas-Lopes. Springer, 2013. Kap. 2, s. 15–57.

[39] *Estimate flat earth position from geodetic latitude, longitude and altitude, MathWorks*. 2011. URL: https://www.arduino.cc/en/Main/ArduinoBoardUno.

[40] *Solidworks Software*. 2016. URL: http://www.solidworks.no/sw/6453_NOR_HTML.htm.

[41] *Google Earth*. 2016. URL: http://www.google.com/intl/no/earth/.

[42] *About Inkscape*.

[43] *Draw.io Online User Manual*. 2016. URL: https://support.draw.io/display/DO/Draw.io+Online+User+Manual.

[44] *ShareLaTeX Documentation*. 2016. URL: https://www.sharelatex.com/learn.

[45] *About Matlab*. 2015. URL: www.mathworks.com/products/matlab.

[46] *GeoGebra Brukermanual*. 2016. URL: http://www.geogebra.org/manual/en/Manual.

[47] *GlobalTop PMTK command packet*. 2012. URL: https://cdn-shop.adafruit.com/datasheets/PMTK_A11.pdf.

[48] S.J. Corfield og J.M. Young. «Advances in Unmanned Marine Vehicles». I: *IET Control Engineering Series* 69.2008 (2008), s. 313–317.

[49] T. I. Fossen og T. A. Johansen. «A Survey of Control Allocation Methods for Ships and Underwater Vehicles». I: *2006 14th Mediterranean Conference on Control and Automation*. Red. av Giuseppe Conte og Marcello Napoletano, s. 1–6.

[50] *Methods of Propulsion: Azimuth Thrusters*. 2009. URL: http://www.brighthubengineering.com/naval-architecture/36176-methods-of-propulsion-azimuth-thrusters/.

[51] Sleipner Motor AS. *Tunnel Installation Guide*. Tekn. rapp. May 2016.

[52] International Towing Tank Conference. *Testing and Extrapolation Methods, Resistance, Resistance Test*. Tekn. rapp. 2002.

[53] J. Holtrop og G.G.J. Mennen. «An Approximate Power Prediction Method». I: *International Shipbuilding Progress* 29.335 (1982), s. 166–170.

[54] L. Asplin, A. D. Sandvik og J. Albretsen. «Strøm i fjorder - Strømkatalog og smittespredning». I: *Havforskningsnytt* 9.2011 (2011), s. 1–2.

[55] P. Barry og P. Crowley. *Modern Embedded Computing*. Morgan Kaufmann, 2012.

[56] R.I. Stephens. «Wind feedforward: blowing away the myths». I: *Marine Technology Society Dynamic Positioning Conference* (2011).

[57] *MINN KOTA Endura 30 C2*. 2016. URL: https://www.marineshop.no/Styring-Motor/Elektrisk-p\%C3\%A5hengsmotor/MINN-KOTA-Endura-30-C2-Skyvekraft-14Kg-Stamme-76cm114946-p0000164238.

[58] *Adafruit Ultimate GPS*. 2016. URL: https://cdn-learn.adafruit.com/downloads/pdf/adafruit-ultimate-gps.pdf.

[59] J. Borenstein, L. Ojeda og S. Kwanmuang. «Heuristic Reduction of Gyro Drift for Personnel Tracking Systems». I: *Journal of Navigation* 62 (01 jan. 2009), s. 41–58. ISSN: 1469-7785. DOI: 10.1017/S0373463308005043. URL: http://journals.cambridge.org/article_S0373463308005043.

[60] *Odroid XU4 User Manual*. 2015. URL: http://magazine.odroid.com/wp-content/uploads/odroid-xu4-user-manual.pdf.

[61] *T200 Thruster Documentation*. 2016. URL: http://docs.bluerobotics.com/thrusters/t200/#d-model.

[62] *Primal-dual interior-point method*. 2016. URL: http://www.joptimizer.com/primalDualMethod.html.

[63] *Onedrive Link to NMEA data*. 2016. URL: https://onedrive.live.com/redir?resid=9055B8486A67FE09!9987&authkey=!ADG-IoW8CMbaxNE&ithint=file\%2cnmea.

[64] M. R. Benjamin mfl. «Protocol-Based COLREGS Collision Avoidance Navigation Between Unmanned Marine Surface Craft». I: *Journal of Field Robotics* 23.5 (2006), s. 1–1.

[65] *- Norge bør bli først på ubemannede skip*. 2015. URL: http://www.sintef.no/siste-nytt/-norge-bor-bli-forst-pa-ubemannede-skip.

Kapittel 8

Vedlegg

Vedlegg 1 Forprosjektrapport
Vedlegg 2 Gantt-diagram
Vedlegg 3 Fremdriftsrapporter 1 til 6
Vedlegg 4 Møtereferater 1 til 7
Vedlegg 5 Risikorapport
Vedlegg 6 Komponentliste prototype
Vedlegg 7 Hovedstrømskjema
Vedlegg 8 Klassediagrammer
Vedlegg 9 Kildekode klientapplikasjon
Vedlegg 10 Kildekode serverapplikasjon
Vedlegg 11 Kildekode Arduino

Vedlegg 1
Forprosjektrapport

FORPROSJEKT - RAPPORT

FOR BACHELOROPPGAVE

NTNU
Kunnskap for en bedre verden

TITTEL:
USV – Unmanned Surface Vessel

KANDIDATNUMMER(E):			
Albert Havnegjerde **Vegard Kamsvåg** **Sveinung Liavaag**			
DATO: **29.01.2016**	EMNEKODE: **IE303612**	EMNE: **Bacheloroppgave**	DOKUMENT TILGANG: - Åpen
STUDIUM: **BACHELOR I INGENIØRFAG, AUTOMATISERINGSTEKNIKK**		ANT SIDER/VEDLEGG: 10/3	BIBL. NR: - Ikke i bruk -

OPPDRAGSGIVER(E)/VEILEDER(E):
NTNU i Ålesund v/ Ottar L. Osen, Rune Volden og Ibrahim Hameed

OPPGAVE/SAMMENDRAG:

NTNU i Ålesund ønsker å utvikle en USV, et ubemannet overflatefartøy som kan fungere som instrument og utstyrsplattform. Fartøyet skal ha dynamisk posisjonering og elektrisk framdriftssystem. Denne oppgaven er gitt som en bacheloroppgave til studenter ved avdeling for ingeniør og realfag, automatiseringsteknikk. Denne forprosjektrapporten er en prosjektbeskrivelse av bacheloroppgaven.

Bacheloroppgaven skal utrede et konsept for et slikt fartøy, med særlig fokus på design av framdrift- og DP-system, kraftsystem, kontrollsystem og instrumentering. Viktige designkriterier vil være vekt, kostnad og fleksibilitet. Dette skal danne et grunnlag for videreutvikling av fartøyet ved senere prosjekter.

I tillegg skal det utvikles en prototype for uttesting av DP og kontrollsystemet, som i størst mulig grad skal representere konseptet.

Denne oppgaven er en eksamensbesvarelse utført av student(er) ved NTNU 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

INNHOLD

1 INNLEDNING

Gruppen har valgt oppgaven USV - Unmanned Surface Vessel, som går ut på å utvikle et konsept for en rimelig, ubemannet båt som kan fungere som instrument og utstyrsplattform. Båten skal ha elektrisk framdrift og dynamisk posisjonering. Det skal også utvikles en billig prototype for uttesting av kontrollalgoritmer og instrumentering. Et tenkt bruksområde for en slik båt kan for eksempel være inspeksjon av oppdrettsanlegg. Oppgaven er gitt av NTNU i Ålesund.

2 BEGREPER

USV - Unmanned Surface Vessel, ubemannet overflatefartøy

DP - Dynamisk Posisjonering, et system for å holde et skip i konstant posisjon og retning

GPS - Globalt Posisjoneringssystem, et system for å finne posisjon og hastighet.

IDE - Integrated Development Environment, utviklingsmiljø

3 PROSJEKTORGANISASJON

3.1 Prosjektgruppe

Studentnummer(e)
130721 - Sveinung Liavaag, Prosjektleder 130160 - Vegard Kamsvåg, Sekretær 130722 - Albert Havnegjerde

Tabell: Studentnummer(e) for alle i gruppen som leverer oppgaven for bedømmelse i faget ID 302906

3.1.1 Oppgaver for prosjektgruppen - organisering

Alle i prosjektgruppen har likt ansvar for gjennomføring av prosjektet. Alle gruppemedlemmer har ansvar for å holde prosjektleder oppdatert på framdrift og eventuelle avvik.

3.1.2 Oppgaver for prosjektleder

- Oppdatere Gantt-diagram
- Møteinnkalling med agenda
- Lede møter med styringsgruppen

3.1.3 Oppgaver for sekretær

- Reservasjon av møterom for prosjektmøte
- Skrive og distribuere møtereferat
- Skrive framdriftsrapport

3.1.4 Oppgaver for øvrige medlem(mer)

- Bistå prosjektleder og sekretær

3.2 Styringsgruppe (veileder og kontaktperson oppdragsgiver)

Styringsgruppen består av Ottar Osen og Ibrahim Hameed fra NTNU i Ålesund, og Rune Volden fra Ulstein Power&Control.

4 AVTALER

4.1 Avtale med oppdragsgiver

Oppgaven går ut på utvikling av et forslag til konsept på en USV, et ubemannet fartøy, som kan navigere selv ved hjelp av DP og tilhørende teknologi. Fokuset er å utvikle et konsept for en USV som kan utvikles videre ved senere prosjekter. Gruppen skal ta stilling til spørsmål som valg av framdriftssystem og plassering av thrustere, kraftforsyningsbehov, instrumentering, kontrollsystem, kommunikasjon og utstyrslayout. Viktige designkriterier for konseptet vil være vekt, kostnad og fleksibilitet. Gruppen skal også utvikle en fungerende prototype som representerer konseptet angående instrumentering, plassering av thrustere, og DP-system. I tillegg vil det bli sett på forskjellige bruksområder for et slikt system og forslag til hvordan konseptet kan tilpasses disse behovene.

4.2 Arbeidssted og ressurser

Prosjektet skal gjennomføres ved NTNU i Ålesund. Her jobber to av tre veiledere, som dermed vil være tilgjengelige på på relativ kort tid ved forespørsel. Den siste veilederen er ansatt ved Ulstein Power & Control og vil derfor være tilgjengelig nokså fort. Veileder skal undersøke om arbeidssted til testing og oppbevaring av prototype kan være på Sunnmøre Museum, utenfor høgskolen. Møter med styringsgruppen skal foregå annenhver mandag.

4.3 Gruppenormer – samarbeidsregler – holdninger

Gruppen er enige om å ha en kjernetid fra 10.00 til 16.00 hver ukedag der det samarbeides om oppgaven. Arbeid utenfor dette tidsrommet anses som nødvendig, men alle gruppemedlemmer skal være tilstede ved NTNU i Ålesund innenfor dette tidsrommet. Dette er for å få en god og jevn kontinuitet i arbeidet, og sikre godt samarbeid innad i gruppen.

Gruppemedlemmene skal behandle hverandre med respekt, og alle skal ha mulighet til å fremme sine

synspunkt. Alle gruppemedlemmer skal være ærlige og nøyaktige i sitt arbeid, og være punktlige med hensyn til avtaler.

5 PROSJEKTBESKRIVELSE

5.1 Problemstilling - målsetting – hensikt

Problemstillingen kan deles i to deler. En del består av konseptutredning av en USV som skal danne basis for videreutvikling ved senere prosjekter. Den andre delen består av utvikling av et DP-system som skal testes i en prototype som skal representere konseptmodellen best mulig.

Problemstillinger:

- Hvordan kan vi utrede et konsept for en USV med dynamisk posisjonering som kan danne en basis for framtidige prosjekter?
- Kan vi bygge en prototype for uttesting av framdriftssystem og kontrollalgoritmer?

Effektmål:

- Levere et konsept som med letthet kan bygges videre på av andre studenter under senere prosjekter

Resultatmål:

- Lage forslag til konsept USV der spørsmål som valg av utforming, energiforbruk, rekkevidde, kommunikasjon osv. blir vurdert.
- Utvikle DP-algoritme.
- Bygge en fungerende prototype for uttesting.

Prosessmål:

- Lære å bruke prosjekt som arbeidsform.
- Integrere tidligere ervervet kunnskap og tilegne seg ny kunnskap til løsning av problemstillingen.

5.2 Krav til løsning eller prosjektresultat – spesifikasjon

Gruppen skal utrede et konsept for en USV, hvor det skal taes stilling til plassering og størrelse på framdriftssystem, nødvendig instrumentering, kraftbehov, kontrollsystem og kommunikasjonsløsning. Det skal utvikles en 3D-modell med forslag til utforming og plassering av utstyr.

Det skal bygges en rimelig prototype som primært skal benyttes til å teste nøyaktigheten til DP-systemet.

Etter diskusjon med styringsgruppen/oppdragsgiver anses prosjektet fullført dersom: Prototypen ligger i fast posisjon med konstant heading (innenfor nøyaktigheten til GPS-systemet) styrt av DP. Båten kjøres manuelt til ny posisjon og legger seg i ny fast posisjon med konstant heading.

5.3 Planlagt framgangsmåte(r) for utviklingsarbeidet – metode(r)

Gruppen vil jobbe dynamisk mot Gantt-diagrammet. Aktivitetene som må bli tidligst ferdig blir prioritert først. Hver person vil arbeide med flere aktiviteter samtidig der aktivitene til enhver tid vil ha forskjellig prioritering etter hvilke som må bli ferdig først.

Gruppemedlemmet som står som ansvarlig for en aktivitet har fullt ansvar for at denne aktiviteten blir ferdig. Hver aktivitet vil også ha en medansvarlig. Utgangspunktet er at disse to skal samarbeide om aktiviteten slik at to og to jobber sammen med hver aktivitet.

Dersom det oppstår problemer med å fullføre en aktivitet skal prosjektleder informeres slik at et møte kan holdes. Agendaen for dette møtet skal være hvordan og hvilke ressurser som skal til for at problemet kan løses.

Det er vanskelig å estimere hvor lang tid og hvor store ressurser hver aktivitet vil ta. Derfor brukes en dynamisk tilnærming som legger til rette for at vi kan oppdatere planen fortløpende. Dette fordi at aktiviteter kan ta mer eller mindre tidsressurser enn planlagt.

5.4 Informasjonsinnsamling – utført og planlagt

I løpet av forprosjektet har gruppen anskaffet relevant litteratur om emnet dynamisk posisjonering. Det fins mye litteratur og mange forskjellige løsninger innenfor emnet dynamisk posisjonering, og mange løsninger er forholdsvis avanserte. En viktig del av prosjektet vil være å finne en løsning som er overkommelig å implementere innenfor en bacheloroppgave.

I tillegg vil det være nødvendig å studere tilgjengelig litteratur for design av fremdriftssystemer for skip med DP. Gruppen har ingen erfaring innenfor dette området fra før.

Gruppen har også brukt tid i løpet av forprosjektet på å søke på internett etter tilgjengelig egnet utstyr til å bygge prototypen.

5.5 Vurdering – analyse av risiko

Målsettingen med utvikling av en prototype baserer seg på at det er mulig å anskaffe utstyr til bygging av prototypen innenfor tilgjengelig budsjett.

Suksessfaktorer:

- Klare, realistiske mål
- Detaljert, oppdatert plan
- God kommunikasjon og samarbeid innad i gruppen
- God kommunikasjon mellom prosjektgruppen og styringsgruppen

Trusler mot suksess:

- For optimistisk tidsplan
- Dårlig kommunikasjon og samarbeid innad i gruppen
- Dårlig rolle/oppgavefordeling
- Mangel på støtte og veiledning fra styringsgruppen
- Svakt definerte suksesskriterier

5.6 Hovedaktiviteter i videre arbeid

A. Litteraturstudie dynamisk posisjonering, maritime fremdriftssystemer.
B. Hovedaktivitet: Utredning av konsept USV
C. Hovedaktivitet: Utstyrsanskaffelse
D. Hovedaktivitet: Designe 3D konseptmodell med utstyrsplassering
E. Hovedaktivitet: Modellering av dynamikk
F. Hovedaktivitet: Utvikling av DP system
G. Hovedaktivitet: Systemutvikling
H. Hovedaktivitet: Bygge prototype
I. Hovedaktivitet: Testing av prototype
J. Hovedaktivitet: Fullføre sluttrapport

For detaljert aktivitetsplan med tidsramme og ansvarlig person refereres det til vedlagt Gantt-diagram.

5.7 Framdriftsplan – styring av prosjektet

5.7.1 Hovedplan

Prosjektet sine hovedaktiviteter skal utføres med utgangspunkt i Gantt-diagrammet. I Gantt-diagram er det sett en estimert startdato og sluttdato for hver aktivitet. Diagrammet sier også hvem som har ansvar for at aktiviteten blir fullført. Aktivitetene er delt inn i et hovednivå og et undernivå siden aktiviteter ofte kan deles inn i flere underaktiviteter. Fullført hovedaktivitet er milepæl i prosjektet.

5.7.2 Styringshjelpemidler

Gruppen bruker GanttProject. Denne programvaren gjør det lettere å holde oversikt over aktivitene som skal utføres, hvem som har hovedansvaret, varigheten på aktivitene, milepæler og kostnader.

Til prosjektrapporten bruker gruppen ShareLaTeX. Dette er et profesjonelt tekstredigeringsprogram som kjøres i nettleseren. Med ShareLatex kan flere personer med tilgang endre og kompilere rapporten samtidig.

5.7.3 Utviklingshjelpemidler

Til utvikling og simulering av DP systemet vil gruppen ha behov for Matlab og Simulink, samt MSS (Marine Systems Simulator) toolbox til Matlab. Til design av en konseptmodell vil gruppen benytte programvaren SolidWorks.
For programmering av sensorer og aktuatorer, som er tilkoblet Arduino mikrokontrollere, vil vi bruke både Arduino's egen IDE og Atmel Studio 6.2. Sistnevnte har en plugin for Arduino-kort, kodefullføring og kompileringsfeil blir oppdaget i sanntid.
PCen ombord vil være en Odroid XU-4 som sannsynligvis vil kjøre reguleringen i en Java applikasjon. Til denne bruker vi NetBeans IDE.

5.7.4 Intern kontroll – evaluering

Prosjektleder har overordnet ansvar for at Gantt-diagrammet oppdateres minst en gang i uka. En slik kontroll vil hjelpe gruppa å holde oversikt over flyten i prosjektet. I tillegg ved samtaler på en daglig basis, siden gruppemedlemmene vil være tilgjengelige for hverandre på hverdager.

For at et delmål skal regnes som fullført må det vere enighet om at det minst er godt nok til å brukes til eventuelle senere delmål.

5.8 Beslutninger – beslutningsprosess

Beslutningene og rammevilkårene for prosjektet, som er bestemt under forprosjektet, er bestemt i de to første møtene med styringsgruppen. Desse møta var 11.01.16 og 25.01.16. Møtereferatene fra desse to møta ligger som vedlegg.

Signifikante endringer i fremdriftsplan skal legges frem for styringsgruppen. Store beslutninger om endringer blir bestemt på styringsmøte med prosjektgruppen og styringsgruppen tilstede.

6 DOKUMENTASJON

6.1 Rapporter og tekniske dokumenter

Prosjektleder skal sende en møteinnkalling til styringsgruppen. Innkallingen skal inneholde en agenda for møtet, en framdriftsrapport som beskriver planlagt og utført arbeid forrige periode i tillegg til planlagte aktiviteter neste periode. Møteinnkallingen skal sendes senest 24 timer før møtet.

Møtereferat etter møte med styringsgruppen skal skrives samme dag som møtet var. Referatet skal inneholde nøkkelpunkta i møtet samt hva det ble enighet om på møtet. Sekretær skriver møtereferat.

Alle dokumenter blir sikkerhetskopiert på internett. Gruppen benytter Microsoft OneDrive for sikkerhetskopiering.

7 PLANLAGTE MØTER OG RAPPORTER

7.1 Møter

7.1.1 Møter med styringsgruppen

- Oppstartmøte mandag 11.01.16 klokka 09.00 med prosjektgruppa og veileder Ottar L. Osen.
- Planlagt møte annenhver mandag fra og med 25.01.16. Hensikten med disse er å informere styringsgruppen om fremdriften til prosjektet og i tillegg gi veiledning til prosjektgruppen.
- Sekretær distribuerer et møtereferat til alle deltagere så raskt som mulig etter møtet.

7.1.2 Prosjektmøter

Prosjektgruppen ser ikke behov for å avtale interne prosjektmøter, da hele gruppen kommer til å jobbe tett sammen gjennom store deler av prosjektet.

Dersom det oppstår problem med en eller flere aktiviteter kan det bli aktuelt med slike møter.

7.2 Periodiske rapporter

7.2.1 Framdriftsrapporter (inkl. milepæl)

Før hvert møte med styringsgruppen skal det utarbeides en framdriftsrapport som viser planlagte og faktisk utførte aktiviteter i foregående toukersperiode, beskrivelse av eventuelle endringer eller avvik fra plan, samt fokus og planlagte aktiviteter for neste toukersperiode. Denne framdriftsrapporten skal distribueres til styringsgruppen minimum 24 timer før møtet. Oppdatert fremdriftsplan (Gantt-diagram) legges ved møteinnkalling. Et møtereferat skal leveres til styringsguppe i etterkant av hvert møte.

8 PLANLAGT AVVIKSBEHANDLING

Dersom et avvik på en bestemt oppgave oppstår som den ansvarlige personen for aktiviteten ikke klarer å løse må prosjektleder varsles. Det må da bli arrangert et internt gruppemøte der disse spørsmålene må stilles.

- Kan avviket løses ved hjelp av mer ressurser?
- Kan avviket unngås?
- Kan avviket løses ved hjelp av ekstern hjelp?
- Kan arbeidsoppgaven bli byttet ut slik at avviket kan bli unngått?

Styringsgruppen skal informeres etter at prosjektgruppen har evaluert. Slik kan styringsgruppen komme med innspill før endelig avgjørelse taes.

9 UTSTYRSBEHOV/FORUTSETNINGER FOR GJENNOMFØRING

Gruppen må finne og kjøpe inn alt av hardware til båt, i tillegg til prototype-båten. Dette betyr at vi må finne passende motor, thrustere, sensorer, mikrokontrollere, liten datamaskin, batteri etc.
Båten og utstyret må også oppbevares et sted under prosjektet. Veileder skal spørre om vi kan jobbe ved Sunnmøre Museum ved uttesting. Utstyr som en vanligvis ikke har tilgang til som er essensielt for gjennomføring er følgende.

- Båt med en lengde på ca 8 fot
- Fremdriftssystem til båt (thrustere, motorkontollere etc.)
- Instrumentering til DP-systemet
- Lett tilgjengelig oppbevarings- og testområde

10 REFERANSER

VEDLEGG

Vedlegg 1	Gantt-diagram
Vedlegg 2	Møtereferat 11.01.2016
Vedlegg 3	Møtereferat 25.01.2016

Vedlegg 2
Gantt-diagram

USV - Unmanned Surface Vessel

23.mai.2016

NTNU i Ålesund

http://

Prosjektleder	Sveinung Liavaag
Prosjekt start/slutt datoer	05.jan.2016 - 28.mai.2016
Avsluttet	99%
Oppgave	45
Deltakere	3

Bachelorprosjekt 2016

Oppgave

Deltakere	Navn	Startdato	Sluttdato	Koordinator
	Forprosjektrapport	05.01.16	29.01.16	
Vegard Kamsvåg, Albert Havnegjerde	A1 Litteraturstudie dynamisk posisjonering, maritime fremdriftssystemer	11.01.16	05.02.16	Vegard Kamsvåg
	A2 Hovedaktivitet: Utredning av konsept USV	01.02.16	19.02.16	
Sveinung Liavaag, Vegard Kamsvåg	A2.1 Framdriftssystem: Plassering av thrustere på skrog	01.02.16	05.02.16	Sveinung Liavaag
Sveinung Liavaag, Vegard Kamsvåg	A2.2 Framdriftssystem: Dimensjonering av thrustere	08.02.16	12.02.16	Sveinung Liavaag
Sveinung Liavaag, Vegard Kamsvåg	A2.3 Framdriftssystem: Vurdere kraftbehov	15.02.16	16.02.16	Sveinung Liavaag
Sveinung Liavaag, Vegard Kamsvåg	A2.4 Framdriftssystem: Forslag til komponentliste	15.02.16	19.02.16	Sveinung Liavaag
Sveinung Liavaag, Albert Havnegjerde	A2.5 Instrumentering: Bestemme nødvendig instrumentering til USV	01.02.16	05.02.16	Albert Havnegjerde
Sveinung Liavaag, Albert Havnegjerde	A2.6 Instrumentering: Vurdere plassering av instrumentering på USV	08.02.16	12.02.16	Albert Havnegjerde
Sveinung Liavaag, Albert Havnegjerde	A2.7 Instrumentering: Forslag til komponentliste	15.02.16	17.02.16	Albert Havnegjerde
Vegard Kamsvåg, Albert Havnegjerde	A2.8 Kontrollsystem: Vurdere kontrollsystemløsning (plattform, protokollstøtte, utviklingsmiljø)	01.02.16	05.02.16	Vegard Kamsvåg
Vegard Kamsvåg, Albert Havnegjerde	A2.9 Kontrollsystem: Kommunikasjonsløsning	08.02.16	12.02.16	Vegard Kamsvåg
Vegard Kamsvåg, Albert Havnegjerde	A2.10 Kontrollsystem: Vurdere løsning for fjernstyring 3DOF	15.02.16	17.02.16	Vegard Kamsvåg
Vegard Kamsvåg, Albert Havnegjerde	A2.11 Kontrollsystem: Forslag til komponentliste	15.02.16	17.02.16	Vegard Kamsvåg
	A3 Hovedaktivitet: Utstyrsanskaffelse	17.02.16	04.03.16	
Sveinung Liavaag, Vegard Kamsvåg	A3.1 Bestille nødvendig utstyr	17.02.16	19.02.16	Sveinung Liavaag

Oppgave

Deltakere	Navn	Startdato	Sluttdato	Koordinator
Vegard Kamsvåg, Albert Havnegjerde	A3.2 Anskaffe båt	22.02.16	04.03.16	Albert Havnegjerde
Vegard Kamsvåg, Albert Havnegjerde	A4 Hovedaktivitet: Designe 3D konseptmodell	22.02.16	04.03.16	Albert Havnegjerde
	A5 Hovedaktivitet: Hovedprogram til prototype med instrumentering	08.02.16	21.03.16	
Vegard Kamsvåg, Albert Havnegjerde	A5.1 Utvikle klient/server-hovudprogram for prototype	29.02.16	21.03.16	Albert Havnegjerde
	A5.2 Testing og implementering av instrumentering til prototype	08.02.16	07.03.16	
Vegard Kamsvåg, Albert Havnegjerde	A5.2.1 GPS	08.02.16	22.02.16	Albert Havnegjerde
Sveinung Liavaag, Albert Havnegjerde	A5.2.2 IMU	22.02.16	07.03.16	Sveinung Liavaag
Sveinung Liavaag, Vegard Kamsvåg	A5.2.3 Vindmåler (retning og styrke)	29.02.16	07.03.16	Vegard Kamsvåg
Sveinung Liavaag, Albert Havnegjerde	A5.2.4 Lufttrykks- og temperaturmåler	29.02.16	07.03.16	Sveinung Liavaag
Vegard Kamsvåg, Albert Havnegjerde	A5.3 Programmere grafisk brukergrensesnitt	14.03.16	21.03.16	Albert Havnegjerde
	A6 Hovedaktivitet: Modellering	18.02.16	26.02.16	
Vegard Kamsvåg, Albert Havnegjerde	A6.1 Matematisk modell av båtens dynamikk	18.02.16	26.02.16	Vegard Kamsvåg
Sveinung Liavaag, Vegard Kamsvåg	A6.2 Modellering av sjøkrefter	22.02.16	26.02.16	Sveinung Liavaag
	A7 Hovedaktivitet: DP system	29.02.16	25.03.16	
Sveinung Liavaag, Vegard Kamsvåg	A7.1 Algoritme for thrusterallokering	29.02.16	11.03.16	Vegard Kamsvåg
Sveinung Liavaag, Albert Havnegjerde	A7.2 Designe regulator(er)	07.03.16	18.03.16	Albert Havnegjerde

Oppgave

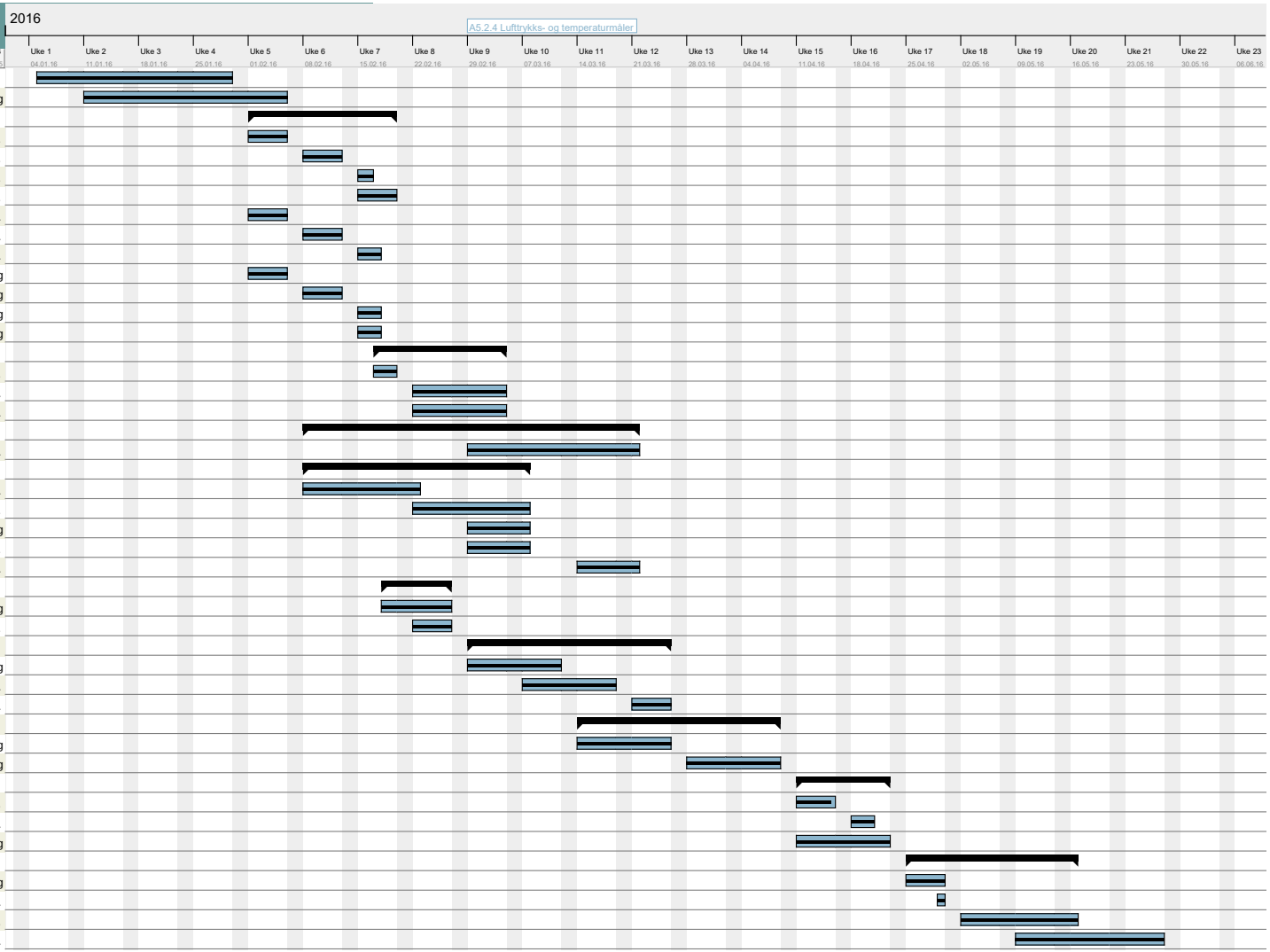
Deltakere	Navn	Startdato	Sluttdato	Koordinator
Sveinung Liavaag, Albert Havnegjerde	A7.3 Utvikle DP algoritme	21.03.16	25.03.16	Albert Havnegjerde
	A8 Hovedaktivitet: Videre implementering i hovedprogram	14.03.16	08.04.16	
Vegard Kamsvåg, Albert Havnegjerde	A8.1 Programmering av thrusterkontroll	14.03.16	25.03.16	Vegard Kamsvåg
Vegard Kamsvåg, Albert Havnegjerde	A8.2 Programmere fjernstyringsløsning	28.03.16	08.04.16	Vegard Kamsvåg
	A9 Hovedaktivitet: Bygge prototype	11.04.16	22.04.16	
Sveinung Liavaag, Vegard Kamsvåg	A9.1 Sammenstilling av kontrollsystem	11.04.16	15.04.16	Sveinung Liavaag
Sveinung Liavaag, Albert Havnegjerde	A9.2 Tørrtest av kontrollsystem og thrustere	18.04.16	20.04.16	Albert Havnegjerde
Vegard Kamsvåg, Albert Havnegjerde	A9.3 Montering av utstyr på båt	11.04.16	22.04.16	Vegard Kamsvåg
	A10 Hovedaktivitet: Testing av prototype	25.04.16	16.05.16	
Sveinung Liavaag, Vegard Kamsvåg	A10.1 Test av fjernstyring	25.04.16	29.04.16	Vegard Kamsvåg
Vegard Kamsvåg, Albert Havnegjerde	A10.2 Test av thrustere og prototypens oppførsel i sjø	29.04.16	29.04.16	Albert Havnegjerde
Sveinung Liavaag, Albert Havnegjerde	A10.3 Test av dynamisk posisjonering	02.05.16	16.05.16	Sveinung Liavaag
Vegard Kamsvåg, Albert Havnegjerde	A11 Hovedaktivitet: Fullføre sluttrapport	09.05.16	27.05.16	Albert Havnegjerde

Deltakere

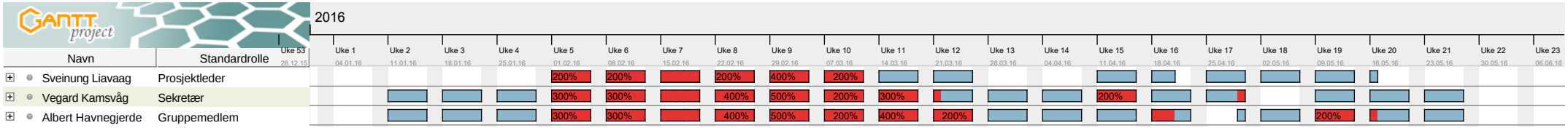
Navn	Standardrolle
Sveinung Liavaag	Prosjektleder
Vegard Kamsvåg	Sekretær
Albert Havnegjerde	Gruppemedlem

Ganttskjema

Deltakere	Navn	Startdato	Sluttdato	Koordinator
	Forprosjektrapport	05.01.16	29.01.16	
Vegard Kamsvåg,Albe...	A1 Litteraturstudie dynamisk posisjonering, maritime...	11.01.16	05.02.16	Vegard Kamsvåg
	A2 Hovedaktivitet: Utredning av konsept USV	01.02.16	19.02.16	
Sveinung Liavaag,Veg...	A2.1 Framdriftssystem: Plassering av thrustere på...	01.02.16	05.02.16	Sveinung Liava...
Sveinung Liavaag,Veg...	A2.2 Framdriftssystem: Dimensjonering av thrust...	08.02.16	12.02.16	Sveinung Liava...
Sveinung Liavaag,Veg...	A2.3 Framdriftssystem: Vurdere kraftbehov	15.02.16	16.02.16	Sveinung Liava...
Sveinung Liavaag,Veg...	A2.4 Framdriftssystem: Forslag til komponentliste	15.02.16	19.02.16	Sveinung Liava...
Sveinung Liavaag,Alb...	A2.5 Instrumentering: Bestemme nødvendig instr...	01.02.16	05.02.16	Albert Havnegj...
Sveinung Liavaag,Alb...	A2.6 Instrumentering: Vurdere plassering av instr...	08.02.16	12.02.16	Albert Havnegj...
Sveinung Liavaag,Alb...	A2.7 Instrumentering: Forslag til komponentliste	15.02.16	17.02.16	Albert Havnegj...
Vegard Kamsvåg,Albe...	A2.8 Kontrollsystem: Vurdere kontrollsystemløsni...	01.02.16	05.02.16	Vegard Kamsvåg
Vegard Kamsvåg,Albe...	A2.9 Kontrollsystem: Kommunikasjonsløsning	08.02.16	12.02.16	Vegard Kamsvåg
Vegard Kamsvåg,Albe...	A2.10 Kontrollsystem: Vurdere løsning for fjernst...	15.02.16	17.02.16	Vegard Kamsvåg
Vegard Kamsvåg,Albe...	A2.11 Kontrollsystem: Forslag til komponentliste	15.02.16	17.02.16	Vegard Kamsvåg
	A3 Hovedaktivitet: Utstyrsanskaffelse	17.02.16	04.03.16	
Sveinung Liavaag,Veg...	A3.1 Bestille nødvendig utstyr	17.02.16	19.02.16	Sveinung Liava...
Vegard Kamsvåg,Albe...	A3.2 Anskaffe båt	22.02.16	04.03.16	Albert Havnegj...
Vegard Kamsvåg,Albe...	A4 Hovedaktivitet: Designe 3D konseptmodell	22.02.16	04.03.16	Albert Havnegj...
	A5 Hovedaktivitet: Hovedprogram til prototype med...	08.02.16	21.03.16	
Vegard Kamsvåg,Albe...	A5.1 Utvikle klient/server-hovudprogram for pro...	29.02.16	21.03.16	Albert Havnegj...
	A5.2 Testing og implementering av instrumenteri...	08.02.16	07.03.16	
Vegard Kamsvåg,Albe...	A5.2.1 GPS	08.02.16	22.02.16	Albert Havnegj...
Sveinung Liavaag,Alb...	A5.2.2 IMU	22.02.16	07.03.16	Sveinung Liava...
Sveinung Liavaag,Veg...	A5.2.3 Vindmåler (retning og styrke)	29.02.16	07.03.16	Vegard Kamsvåg
Sveinung Liavaag,Alb...	A5.2.4 Lufttrykks- og temperaturmåler	29.02.16	07.03.16	Sveinung Liava...
Vegard Kamsvåg,Albe...	A5.3 Programmere grafisk brukergrensesnitt	14.03.16	21.03.16	Albert Havnegj...
	A6 Hovedaktivitet: Modellering	18.02.16	26.02.16	
Vegard Kamsvåg,Albe...	A6.1 Matematisk modell av båtens dynamikk	18.02.16	26.02.16	Vegard Kamsvåg
Sveinung Liavaag,Veg...	A6.2 Modellering av sjøkrefter	22.02.16	26.02.16	Sveinung Liava...
	A7 Hovedaktivitet: DP system	29.02.16	25.03.16	
Sveinung Liavaag,Veg...	A7.1 Algoritme for thrusterallokering	29.02.16	11.03.16	Vegard Kamsvåg
Sveinung Liavaag,Alb...	A7.2 Designe regulator(er)	07.03.16	18.03.16	Albert Havnegj...
Sveinung Liavaag,Alb...	A7.3 Utvikle DP algoritme	21.03.16	25.03.16	Albert Havnegj...
	A8 Hovedaktivitet: Videre implementering i hovedpr...	14.03.16	08.04.16	
Vegard Kamsvåg,Albe...	A8.1 Programmering av thrusterkontroll	14.03.16	25.03.16	Vegard Kamsvåg
Vegard Kamsvåg,Albe...	A8.2 Programmere fjernstyringsløsning	28.03.16	08.04.16	Vegard Kamsvåg
	A9 Hovedaktivitet: Bygge prototype	11.04.16	22.04.16	
Sveinung Liavaag,Veg...	A9.1 Sammenstilling av kontrollsystem	11.04.16	15.04.16	Sveinung Liava...
Sveinung Liavaag,Alb...	A9.2 Tørrtest av kontrollsystem og thrustere	18.04.16	20.04.16	Albert Havnegj...
Vegard Kamsvåg,Albe...	A9.3 Montering av utstyr på båt	11.04.16	22.04.16	Vegard Kamsvåg
	A10 Hovedaktivitet: Testing av prototype	25.04.16	16.05.16	
Sveinung Liavaag,Veg...	A10.1 Test av fjernstyring	25.04.16	29.04.16	Vegard Kamsvåg
Vegard Kamsvåg,Albe...	A10.2 Test av thrustere og prototypens oppførsel...	29.04.16	29.04.16	Albert Havnegj...
Sveinung Liavaag,Alb...	A10.3 Test av dynamisk posisjonering	02.05.16	16.05.16	Sveinung Liava...
Vegard Kamsvåg,Albe...	A11 Hovedaktivitet: Fullføre sluttrapport	09.05.16	27.05.16	Albert Havnegj...



Personelloversikt



Vedlegg 3
Fremdriftsrapporter 1-6

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

Hovedhensikt / fokus for arbeidet i denne perioden

Denne perioden har vært første prosjektperiode etter oppstartsmøte den 11.01.16. Fokus for arbeidet i denne perioden har vært å jobbe med forprosjektrapporten, samt få en oversikt over oppgaven som skal utføres.

Planlagte aktiviteter i denne perioden

Planlagte aktiviteter for denne perioden var:

1. Komme i gang med forprosjektrapport
2. Lage utkast til framdriftsplan (Gantt diagram)
3. Avgrense og spesifisere oppgaven
4. Lage mal for prosjektrapport i ShareLaTeX
5. Innhente relevant litteratur for å kunne løse oppgaven

Virkelig gjennomførte aktiviteter i denne perioden

1. Forprosjektrapporten er godt i gang, og blir ferdig neste uke.

2. Utkast til Gantt-diagram er laget, med forslag til aktiviteter, ansvarlig person og medhjelper, samt varighet for aktivitet. Dette vil spesifiseres ytterligere innen forprosjektet er ferdig.

3. Mal for rapport er opprettet i ShareLaTeX, alle gruppemedlemmer har redigeringsrett i dokumentet.

4. Gruppen har anskaffet relevante lærebøker og artikler/avhandlinger innenfor emnet dynamisk posisjonering og ulineære systemer. Vi har brukt en del tid på å få oversikt over emnet, og hvilke mulige løsninger som finnes. Vi har anskaffet Marine Systems Simulator toolbox til Matlab/Simulink, som inneholder en del ferdige funksjoner/blokker for utvikling av DP-systemer. Litt tid har gått med på å gjøre seg kjent med denne. Læreboken *Handbook of Marine Craft Hydrodynamics and Motion Control* refererer til denne toolboxen, og vi tror den kan bli nyttig i det videre arbeidet.

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

Avvik på planlagt aktivitet 3. Oppgaven trenger ytterligere avgrensing og spesifisering, spesielt med tanke på spesifikasjonskrav til prosjektresultatet. Gjøres i samarbeid med styringsgruppen.

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

-

Hovederfaring fra denne perioden

Fått oversikt over oppgaven, og mengden arbeid som kreves for å gjennomføre den.

Hovedhensikt/fokus neste periode

Fokus neste periode er å fullføre forprosjektet, og komme i gang med utføring. Spesielt fokus på utstyrsbehov og plassering av framdriftssystem.

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)	Antall timer denne per. (fra logg)	Prosjektgruppe (navn)	Dato

Planlagte aktiviteter neste periode

Fullføre A1 – Litteraturstudie DP system - Alle
A21- Framdriftssystem – Sveinung
A211 Plassering av thrustere på skrog
A212 Vurdere kraftbehov
A22 – Instrumentering – Albert
A221 Bestemme nødvendig instrumentering for å løse oppgaven
A23 – Kontrollsystem – Vegard
A231 Kontrollsystemløsning (protokollstøtte, regnekraft, utviklingsmiljø)
A232 Fjernstyringsløsning 3DOF

Annet

-

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

-

Godkjenning/signatur gruppeleder	Signatur øvrige gruppedeltakere
Sveinung Liavaag - signatur	Vegard Kamsvåg – signatur Albert Havnegjerde - signatur

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

Hovedhensikt / fokus for arbeidet i denne perioden

Fokus for arbeidet denne perioden har vært å fullføre og levere forprosjektrapport, samt undersøking av mulige løsninger for framdriftssystem, kontrollsystem og instrumentering, både med tanke på konseptet og prototypen.

Planlagte aktiviteter i denne perioden

Forprosjektrapport
A1: Litteraturstudie dynamisk posisjonering, maritime framdriftssystemer.

A2.1: Plassering av thrustere på skrog. Hvor skal thrustere plasseres, utforming av thrustertunnel. Både med tanke på normal framdrift og DP. Ansvarlig: Sveinung, medhjelper: Vegard.

A2.5: Bestemme nødvendig intrumentering til USV. Hvilken instrumentering er nødvendig med hensyn på dynamisk posisjonering, hvilken instrumentering er fornuftig å ha med i konseptet. Hvordan instrumenteringen kan tilkobles resterende kontrollsystem. Ansvarlig: Albert, medhjelper: Sveinung.

A2.8: Vurdere kontrollsystemløsning (plattform, protokollstøtte, utviklingsmiljø). Valg av kontrollsystemplattform, hvilke kommunikasjonsprotokoller må støttes, hvilket utviklingsmiljø skal kontrollsystemet utvikles i. Ansvarlig: Vegard, medhjelper: Albert.

Virkelig gjennomførte aktiviteter i denne perioden

Forprosjektrapport.
A1
A2.5
A2.8
A3.1: Bestille nødvendig utstyr

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

A3.1: Bestillingsliste på utstyr til prototypen (som ikke allerede er tilgjengelig ved NTNU i Ålesund) er sendt til labingeniør Anders Sætersmoen.

A2.1: Plassering av thrustere på skrog. Gruppen planlegger å kontakte Jostein Berge eller Arne Jan Sollied ved AMO for faglig input, da erfaring med slik design er manglende. Aktivitet er påbegynt, men ikke avsluttet.

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

-

Hovederfaring fra denne perioden

- Gruppen planlegger å benytte SBC (single board computer) Odroid XU4 som kontrollsystemsplattform. Programvare vil utvikles i java, siden gruppen har god kjennskap til og erfaring med java.

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

- Til I/O (instrumentering, motorkontroll) vil det bli benyttet Arduino, med seriell kommunikasjon til Odroid. Differensiell GPS er anskaffet, testing mot Arduino pågår.
- Vi vil bruke BlueRobotics T200 thrustere til prototypen. Disse er enkle å styre med PWM signal fra Arduino.

Hovedhensikt/fokus neste periode

Fokus for neste periode er å bli ferdig med utredningen av konsept USV framdriftssystem, kraftbehov, kontrollsystem, fjernstyring og instrumentering, og sette opp et forslag til komponentliste for konseptmodellen.
Anskaffing av båt til prototypen vil også være fokus, vi må undersøke bruktmarkedet i nærområdet, eventuelt kontakte forhandlere/produsenter og forespørre pris på ny jolle i egnet størrelse.
I tillegg er det planlagt å starte med modellering av båt neste periode.

Planlagte aktiviteter neste periode

A2.2 Framdriftssystem: Dimensjonering av thrustere til konsept USV. Ansvarlig: Sveinung, medhjelper: Vegard

A2.3 Framdriftssystem: Vurdere kraftbehov. Vurdere batterikapasitetsbehov med basis i valgte dimensjoner/komponenter, med tanke på driftstid mellom ladinger, vekt etc. Ansvarlig: Sveinung, medhjelper: Vegard.

A2.4 Framdriftssystem: Forslag til komponentliste til konsept USV. Ansvarlig: Sveinung, medhjelper: Vegard.

A2.6 Instrumentering: Plassering av instrumentering på USV. Ansvarlig: Albert, medhjelper: Sveinung

A2.7 Instrumentering: Forslag til komponentliste. Ansvarlig: Albert, medhjelper: Sveinung.

A2.9 Kontrollsystem: Kommunikasjonsløsning. Løsning for å kommunisere med en «basestasjon» (pc) på land, med mulighet for fjernstyring. Ansvarlig: Vegard, medhjelper: Albert.

A2.10 Kontrollsystem: Vurdere løsning for fjernstyring 3DOF. Løsning for å fjernstyre båt fra land, i tre frihetsgrader. Ansvarlig: Vegard, medhjelper: Albert.

A2.11 Kontrollsystem: Forslag til komponentliste. Ansvarlig: Vegard, medhjelper: Albert.

A3.2 Anskaffe båt for prototype. Ansvarlig: Albert, medhjelper: Vegard.

A5.1 Matematisk modell av båtens dynamikk. Starte med modellering av båt, med tanke på utvikling av DP kontroller. Ansvarlig: Vegard, medhjelper: Albert.

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

Annet
-

Ønske om /behov for veiledning, tema i undervisningen – drøfting ellers
Gruppen trenger faglig input for prosjektering av framdriftssystem. Fagpersoner ved NTNU i Ålesund er identifisert, gruppen tar kontakt snarlig.

Godkjenning/signatur gruppeleder	Signatur øvrige gruppedeltakere
Sveinung Liavaag - signatur	Vegard Kamsvåg – sekretær Albert Havnegjerde – gruppemedlem

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

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

Hovedhensikt / fokus for arbeidet i denne perioden

Konseptutredning, litteraturstudie og implementering av instrumentering

Planlagte aktiviteter i denne perioden

Fullføre konseptutredning(framdriftssystem, instrumentering og kontrollsystem).
Anskaffe jolle til prototype.
Matematisk modell av båtens dynamikk.

Virkelig gjennomførte aktiviteter i denne perioden

Utvikle applikasjon for GPS i Java og for Arduino og testet på Odroid.
Fullføre konseptutredning (framdriftssystem, instrumentering og kontrollsystem)..

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

Matematisk modell av båtens dynamikk: Ikke påbegynt, starter 22.02. Skulle startet 18.02
Anskaffe jolle: Søkt på annonsenettsteder og kjøp/salg sider i sosiale medier. Egnet jolle til prototype (pioner 8 mini) ligger rundt 10 000 NOK i pris.

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

Hovederfaring fra denne perioden

Modbus TCP mellom Odroid(Slave) og PC(Master) virket ikke, dersom rollene ble byttet virket det. Vanlig klient-server TCP socket var feilfritt.

GPS applikasjonen gjør en breddegrad og en lengdegrad til koordinater i xyz gitt av en bredde- og lengdegradreferanse.

Hovedhensikt/fokus neste periode

Fortsette med implementering av instrumentering.
Erverve seg nok ferdigheter til å tegne i solidworks.
Modellering.

Planlagte aktiviteter neste periode

Uvikle applikasjon for IMUen i Java og for Arduino
Utstyrsplassering i en 3D modell for konseptet.
Matematisk modell av prototypens dynamikk
Anskaffe jolle

Annet

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

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

Ønske om /behov for veiledning, tema i undervisningen – drøfting ellers	
Godkjenning/signatur gruppeleder Sveinung Liavaag - signatur	Signatur øvrige gruppedeltakere Vegard Kamsvåg – sekretær Albert Havnegjerde – gruppemedlem

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

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

Hovedhensikt / fokus for arbeidet i denne perioden

Fortsette med implementering av instrumentering.
Erverve seg nok ferdigheter til å tegne i solidworks.
Modellering.

Planlagte aktiviteter i denne perioden

Uvikle applikasjon for IMUen i Java og for Arduino
Utstyrsplassering i en 3D modell for konseptet.
Matematisk modell av prototypens dynamikk
Algoritme for thrusterallokering.
Anskaffe jolle

Virkelig gjennomførte aktiviteter i denne perioden

Uvikle applikasjon for IMUen i Java og for Arduino
Algoritme for thrusterallokering - pågår
Matematisk modell av prototypens dynamikk
Anskaffe jolle

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

Designe 3D konseptmodell med utstyrsplassering: Planlagt ferdig 06.03. Trenger større kunnskap i Solidworks.

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

Hovederfaring fra denne perioden

Den matematiske modellen av båtens dynamikk er utleda, selv om det er mange ukjente koeffisienter som bare kan finnes ut gjennom omfattende testing av prototypen.

Den kjøpte værstasjonen som måler vindretning og vindstyrke har alt for lav presisjon på korte tisdintervall og kan derfor trolig ikke brukes til annet enn å logge værdata.

Har nå fått prototype-båt og mangler i hovedsak berre thrusterer.
Ser at behovet for å få thrusterene begynner å nærme seg selv om det fortsatt er nok å ta tak i.

Hovedhensikt/fokus neste periode

Erverve seg nok ferdigheter til å tegne i solidworks.
Dersom vi får thrusterene: Testing av thrustere
Fokus på DP-algoritme

Planlagte aktiviteter neste periode

Programmere grafisk brukergrensesnitt
Algoritme for thrusterallokering
Designe regulator(er)

Annet

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

Ønske om /behov for veiledning, tema i undervisningen – drøfting ellers	
Godkjenning/signatur gruppeleder Sveinung Liavaag - sign	Signatur øvrige gruppedeltakere Albert Havnegjerde Vegard Kamsvåg

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)	Antall timer denne per. (fra logg)	Prosjektgruppe (navn)	Dato

Hovedhensikt / fokus for arbeidet i denne perioden

Erverve seg nok ferdigheter til å tegne i solidworks.
Dersom vi får thrusterene: Testing av thrustere
Utvikling av DP-algoritme og thrusterkontroll
Programmering av kontrollsystem

Planlagte aktiviteter i denne perioden

Programmere grafisk brukergrensesnitt
Algoritme for thrusterallokering
Designe regulator(er)
Utvikle DP algoritme
Programmere grafisk brukergrensesnitt
Programmering av thrusterkontroll
Programmere fjernstyringsløsning
Utvikle klient-server hovedprogram for prototypen

Virkelig gjennomførte aktiviteter i denne perioden

Programmere grafisk brukergrensesnitt
Algoritme for thrusterallokering
Designe regulator(er)
Utvikle DP algoritme
Programmere grafisk brukergrensesnitt
Programmering av thrusterkontroll
Programmere fjernstyringsløsning

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

Utvikle klient-server hovedprogram for prototypen pågår, 90% ferdig.
3D-modell i SolidWorks har blitt nedprioritert pga. andre aktiviteter har blitt vurdert som viktigere i nåværende fase av prosjektet.

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

Ønsker å flytte tegning av 3D-modell til fullføring av sluttrapport. Gruppen anser det som viktigere å bli ferdig med prototypen først.

Hovederfaring fra denne periode

DP-kontrolleren vil bestå av 3 PID-regulatorer, hvor vi antar at aksene er dekoplet. Dette fordi en fullstendig matematisk modell av jolla med tanke på utvikling av regulator(er) vil kreve omfattende testing og systemidentifikasjon, som det ikke er tid til.

På grunn av skrogets utforming vurderer vi det som mest hensiktsmessig å ha thrustertunnellene liggende utenpå skroget. Bunnen er nesten helt flat, som gjør at tunnellene ville blitt veldig lange om de skal gå gjennom skroget, samt at jolla må ligge veldig lavt i vannet for at tunellene skal være helt nedsenket. Ved at tunellene ligger utenpå minimerer vi også risikoen for at det skal lekke vann inn i dobbeltskroget.

Har fått thrustere, og har testkjørt en i bassenget i lab-bygget. Det var problemfritt å kople opp og kjøre.

Bruker Bitbucket og Git for versjonskontroll av software, dette har fungert bra.

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)	Antall timer denne per. (fra logg)	Prosjektgruppe (navn)	Dato

Hovedhensikt/fokus neste periode
Hovedfokus neste periode er å bygge prototypen.
Planlagte aktiviteter neste periode
Utvikle klient-server hovedprogram for prototypen (ferdigstilling) Sammenstilling av kontrollsystem Test av kontrollsystem og thrustere Montering av utstyr på båt
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	Antall møter denne periode 1).	Firma - Oppdragsgiver Høgskolen i Ålesund /	Side 1 av 2
Rapport fra prosess Framdriftsrapport	Periode/uke(r)	Antall timer denne per. (fra logg)	Prosjektgruppe (navn)	Dato

Hovedhensikt / fokus for arbeidet i denne perioden

Hovedfokus denne perioden har vært å bygge prototypen.

Planlagte aktiviteter i denne perioden

Utvikle klient-server hovedprogram for prototypen (ferdigstilling)
Sammenstilling av kontrollsystem
Test av kontrollsystem og thrustere
Montering av utstyr på båt

Virkelig gjennomførte aktiviteter i denne perioden

Utvikle klient-server hovedprogram for prototypen (ferdigstilling)
Sammenstilling av kontrollsystem
Montering av utstyr på båt
Jobbet med prosjektrapport

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

På grunn av thrusterenes utforming har det blitt 3D-printa festeanordning for innfesting av thrustere i rør. Dette tok litt tid, så det har blitt jobbet en del med rapportskriving for å kunne hente inn tapt tid. Rørene er nå sveiset fast i skroget, og vi er klar for testing.

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

Hovederfaring fra denne perioden

For å feste thrusterene i rør har det blitt 3D-printet festeanording som passer inn i røret som ble innkjøpt. Denne løsningen ser ut til å fungere bra så langt.

Hadde jolla i slepetanken i kjelleren på lab-bygget for å sjekke fribord samt få et grovt estimat på motstand i vannet. Laveste punkt på kjølen stikker ca 10 cm dypt uten last om bord.

Hovedhensikt/fokus neste periode

Testing av prototype
Fullføre sluttrapport

Planlagte aktiviteter neste periode

Test av fjernstyring
Test av thrustere og prototypens oppførsel i sjø
Test av dynamisk posisjonering
Designe 3D-konseptmodell
Fullføre sluttrapport

Annet

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)	Antall timer denne per. (fra logg)	Prosjektgruppe (navn)	Dato

Ønske om /behov for veiledning, tema i undervisningen – drøfting ellers	
Godkjenning/signatur gruppeleder Sveinung Liavaag	Signatur øvrige gruppedeltakere Vegard Kamsvåg Albert Havnegjerde

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

Vedlegg 4
Møtereferater 1-7

Møtereferat

Dato: 11.01.2016

Sted: Ottar L. Osens kontor

Deltakere: Vegard Kamsvåg, Sveinung Liavaag, Albert Havnegjerde, Ottar L. Osen

Møtets formål: Oppstartsmøte hovedprosjekt

Oppstartsmøte for hovedprosjekt USV – Unmanned Surface Vessel ble holdt mandag 11. januar 2016 på Ottar L. Osens kontor. Møtet startet med at Ottar L. Osen presenterte sin visjon om prosjektet, og potensielle bruksområder for et ubemannet overflatefartøy. Siden prosjektet er for stort for ett bachelorprosjekt, og er planlagt å gå over flere år, fikk gruppen komme med ønske om hva vi vil fokusere på.

Gruppen ytret ønske om å fokusere på utvikling av et konsept for framdrift og manøvrering ment for input til framtidige prosjekter, samt utvikling av en løsning for dynamisk posisjonering av fartøyet. I tillegg ønsker gruppen å utvikle en billig, enkel prototype for testing av instrumentering og algoritmer for dynamisk posisjonering. Ottar var enig i at dette vil være et bra fokusområde for prosjektet.

I tiden fram til neste statusmøte vil gruppen jobbe med forprosjektrapporten, og undersøking av potensielle løsninger for framdrift, dynamisk posisjonering og instrumentering, både med tanke på det ferdige produktet samt prototypen som skal utvikles. Gruppen skal også forsøke å finne en egnet jolle for innkjøp, til utvikling av prototypen.

Neste statusmøte er planlagt den 25.01.16 kl. 09.00, på Ottars kontor.

Møtereferat

Fremdriftsmøte for prosjekt USV – Unmanned Surface Vessel den 25. januar 2016 kl 1100

Sted: A433

Deltakere: Sveinung Liavaag, Albert Havnegjerde, Vegard Kamsvåg, Ottar L. Osen, Rune Volden

Saker:

1. **Konseptmodell – hva skal gruppen ta stilling til?**
   Gruppen ønsket å få klarhet i hva det forventes at vi skal utrede med tanke på konseptmodellen. Spesielt med tanke på at gruppen ikke har bakgrunn innen hydrodynamikk eller andre maritime fag. Styringsgruppen mente at det bør som et minimum taes stilling til plassering av thrustere, dimensjonering av thrustere (størrelse, skyvkraft), og for annet utstyr som kraftforsyning, kontrollsystem, instrumentering og lignende bør det taes hensyn til vekt og størrelse, samt plassering i skipet med tanke på stabilitet.
   Styringsgruppen fremhevet også viktigheten av fleksibilitet i konseptet, da forutsetningene for konseptet kan endre seg i tiden mellom dette prosjektets ferdigstilling og en eventuell videreføring av USV-prosjektet.

2. **Krav til løsning eller resultat – spesifikasjon**
   Som en del av forprosjektet skal det spesifiseres hvilke krav som gjelder for det endelige resultatet.
   For prototypen ble det spesifisert:
   - Kunne ligge i ro på DP med konstant heading, innenfor nøyaktigheten gitt av oppløsningen på GPS (typisk +-1 meter).
   - Gå til ny posisjon og gå over i DP med ny heading.
3. **Annet**
   Styringsgruppen påpekte viktigheten av en helhetlig og grundig rapport, siden rapporten er det viktigste vurderingskriteriet for sensor. Det kom fram mange nyttige tips til rapportskrivingen som prosjektgruppen tar med seg videre i arbeidet.
   Styringsgruppen påpekte også at det er viktig at komponenter som behøves til prototypen og som kan ha lang leveringstid spesifiseres tidlig, slik at dette kan bestilles i god tid før arbeid med prototyping kommer i gang. Dette vil spesielt gjelde thrustere og skrog for prototypen.

NTNU i Ålesund 25. januar 2016

Vegard Kamsvåg, sekretær

Møtereferat

Fremdriftsmøte for prosjekt USV – Unmanned Surface Vessel den 9. februar 2016 kl 09.00

Sted: B434

Deltakere: Sveinung Liavaag, Albert Havnegjerde, Vegard Kamsvåg, Rune Volden, Ottar L. Osen

Saksliste:

1. **Gå gjennom framdriftsrapport**
   Ønske fra styringsgruppen at framdriftsrapporten beskriver i hovedtrekk hva som er gjort foregående periode, og hva som er planlagt til neste periode, i stedet for direkte avskriving av aktiviteter fra Gantt-diagram.

2. **Gantt-diagram**
   Styringsgruppen ønsker at Gantt-diagram skal brytes ned i flere aktiviteter til neste møte med styringsgruppen, siden gruppen nå har fått større oversikt over oppgaven.
   Anbefaling fra Ottar at gruppen vurderer å benytte Asana som prosjektstyringsverktøy.

3. **Forprosjektrapport**
   Gruppen bør utarbeide en risikoliste, med risikoreduserende tiltak, som oppdateres underveis. Gruppen skal lage en risiko-matrise, hvor aksene beskriver sannsynlighet og konsekvens. Slik kan vi få en visuell indikasjon på risikoene, før og etter risikoreduserende tiltak. Utarbeides før neste møte med styringsgruppen.

4. **Eventuelt**
   Styringsgruppen framhevet viktigheten i å finne enkleste løsninger, samt viktigheten i å ta tak i problemer som gruppen støter på tidligst mulig.

   Til aktivitet A2.1 Plassering av thrustere på skrog anbefaler styringsgruppen at fastmonterte thrustere vil være lettest å implementere på prototypen.

   Gruppen må ta en vurdering på om Odroid eller Raspberry Pi skal benyttes som kontrollplattform på USV. Må vurdere hva som er best, eventuelt godt nok til formålet.

   Til neste møte skal gruppen dokumentere arbeid som er utført til nå, og plan i hovedtrekk videre.

Møtereferat

Fremdriftsmøte for prosjekt USV – Unmanned Surface Vessel den 22. februar 2016 kl 14.00

Sted: F420

Deltakere: Sveinung Liavaag, Albert Havnegjerde, Vegard Kamsvåg, Rune Volden, Ottar L. Osen

Saksliste:

1. **Gå gjennom risikorapport**
   Malen som gruppen benytter til risikorapporten er bra, men innholdet er litt mangelfullt enda. Dokumentet vil brukes aktivt gjennom prosjektet, og benyttes som innspill til møter og lignende. Styringsgruppen påpekte at risikovurdering er et godt verktøy ikke bare til identifisering av risiko i prosjektet, men også til løsing av problemer underveis i prosjektet, da man tvinges til å tenke grundig gjennom hva som kan gå galt, og hvordan man kan unngå dette.

2. **Gå gjennom framdriftsrapport**
   Jamod for Modbus kommunikasjon med odroid som slave virker ikke, uten at gruppen kan identifisere grunnen til dette problemet. Gruppen er uansett ikke nødt til å bruke Modbus til dette prosjektet, man må vurdere funksjonalitet opp mot tidsbruk. Gruppen kan for eksempel lage en egen protokoll, og benytte socketprogrammering i java for sending og mottak av data.

   Gruppen må ta stilling til hva som skal skje om båten mister kontakt med PC på land. Bør ha en form for overvåking av kommunikasjonslinken(e), og ta aksjon om den faller bort.

   GPS er testet både på arduino og odroid, og metoder for koordinat-transformasjon er implementert. Arbeid med implementasjon av IMU pågår.

   3D-modellering skal starte denne perioden. Pål Steffen Kleppe er ressursperson på Siemens NX om gruppen skal benytte denne programvaren.

3. **Diskutere kjøp av båt**
   Pris for Pioner 8 mini, som kan være en egnet båt for prototype, koster fra 9500 (brukt) til 11300 (ny) NOK. Styringsgruppen mener at det skal være overkommelig å kjøpe inn, og gruppen skal ha fokus på anskaffelse av båt denne uken. Gruppen skal følge med bruktmarkedet, samt innhente pris hos Ferd (båtforhandler i Skarbøvik).

   Når en prototype er på plass, får gruppen behov for hjelp til å sveise plast. Ottar tipset om at Jostein Berge ved AMO kan være behjelpelig med kontakter som kan bistå.

4. **Eventuelt**
   Gruppen kan snakke med Arne Jan Sollied om å få bruke slepetanken i kjelleren på lab-bygget for testing av motstand og skyvkraft på prototypen.

Gruppen skal spørre prosjektgruppen som har anskaffet BlueRobotics T100-thrustere om å få låne en thruster for testformål. Dette for å være sikker på at styring av thrustere ikke blir et problem senere, dersom det skulle vise seg at thrustere som er bestilt til dette prosjektet kommer senere enn forventet.

Styringsgruppen lurte på om prosjektgruppen har vurdert batteri/kraftbehov til prototypen. Gruppen mener at litium-ione batteri har de beste egenskapene for formålet, men disse batteriene er forholdsvis kostbare. Til konseptet vil gruppen foreslå litium-ione batteri, men til prototypen vil det bli brukt en billigere løsning som for eksempel et bilbatteri. Pris på 12V 100Ah batteri fra biltema er 1100 kr, og dette vil antagelig holde for prototypen.

Møtereferat

Fremdriftsmøte for prosjekt USV – Unmanned Surface Vessel den 7. mars 2016 kl 09.00

Sted: F424

Deltakere: Sveinung Liavaag, Albert Havnegjerde, Vegard Kamsvåg, Rune Volden, Ottar L. Osen

Saksliste:

1. **Diskutere framdriftsrapport**
   Båt (Pioner 8 mini) er anskaffet, og ligger utenfor tunglabben i kjelleren på lab-bygget.
   3D-modell av konseptbåten er ikke ferdig enda, vil pågå i tiden fremover. Denne vil ikke bli veldig detaljert, men vil forsøke å få fram layout og plassering av thrustere og lignende.

   DP-modellen gruppen har utviklet inneholder en del ukjente koeffisienter som må finnes via systemidentifikasjon, om regulator skal utvikles med modellen som utgangspunkt.
   Ottar foreslo å starte med en ren proporsjonalregulator, eventuelt legge til integratorvirkning om nødvendig, for å teste om det er godt nok. Hvis ikke fins det «enkle» systemidentifikasjonsmetoder som kan forsøkes.

   Gruppen ligger ellers greit an i forhold til planen, bortsett fra litt etterslep på aktiviteten for 3D-modellering. Dette etterslepet vil ikke ha noe å si for videre framdrift.

2. **Diskutere risikorapport**
   Risiko relatert til innkjøp av båt er nå eliminert, siden båt er innkjøpt. Risiko relatert til dårlig/feil prosjektplan er redusert på grunn av revidert Gantt-diagram.
   Til neste møte kan gruppen behandle risiko relatert til tidsrammen for prosjektet på nytt, da vi har stadig bedre oversikt over gjenstående arbeid.

3. **Eventuelt**
   Thrustere til prototypen er enda ikke ankommet. Gruppen skal purre leverandør.

   Nå som båt er innkjøpt, kan gruppen begynne å planlegge montering av thrustere på skroget. Man må finne ut hva som er den beste løsningen, om det skal borres inn tuneller eller om tunellene kan ligge utenpå skroget. Gruppen skal også finne ut fribord på jolla med utstyr om bord, og gjøre en enkel beregning på maks lastevekt.

   Ottar informerte om at det er en vinylkutter på L101 som vi kan bruke til å lage logoer til prototypen.

Møtereferat

Fremdriftsmøte for prosjekt USV – Unmanned Surface Vessel den 4. april 2016 kl 10.00

Sted: B431

Deltakere: Sveinung Liavaag, Albert Havnegjerde, Vegard Kamsvåg, Rune Volden, Ottar L. Osen

Saksliste:

1. **Diskutere framdrift**
   Gruppen ligger bra an i forhold til planlagt framdrift, bortsett fra 3D-modelleringen, som foreløpig har blitt nedprioritert. Veilederene påpekte at en slik modell vil være verdifull i sluttrapporten, og at det fins kompetanse ved NTNU i Ålesund som kan hjelpe gruppen ved behov. Gruppen bør prøve å få dette til.

2. **Diskutere bygging av prototype**
   Bygging av prototypen er neste trinn i prosjektet. På grunn av skrogets utforming vil thrusterene ligge utenpå skroget. Det må kjøpes inn rør for innmontering av thrustere, og disse må sveises på skroget.

   Til kontrollsystemet må det kjøpes inn en kapsling til kontrollsystemet, som har mulighet for å sette inn PG-nipler. Det må også anskaffes koblingsbokser til motorkablene, siden disse er relativt korte og må skjøtes. Gruppen skal til Berggård-Amundsen og kjøpe inn nødvendige deler.

   Det må også anskaffes et 12V bilbatteri som kraftforsyning. Dette kan kjøpes på biltema.
   En 12-5V DC/DC omformer må anskaffes, siden Odroid og Arduino trenger 5V. Kan kjøpes billig på Elfa.

   Lageret på L160 skal brukes som «verksted» for å bygge prototypen.

3. **Eventuelt**
   Gruppen foreslo å benytte bassenget i kjelleren på lab-bygget til en første test av prototypen, for å kunne teste i kontrollerte omgivelser. Må snakke med André og Arne Jan om tillatelse. Samtidig kan vi prøve å dra jolla i bassenget med en fjærvekt, for å få et grovt estimat av motstanden i vannet.

   Til sluttrapporten må det utarbeides en komplett materialliste for prosjektet.

Møtereferat

Fremdriftsmøte for prosjekt USV – Unmanned Surface Vessel den 28. april 2016 kl 09.00

Sted: F420

Deltakere: Sveinung Liavaag, Albert Havnegjerde, Vegard Kamsvåg, Ottar L. Osen

Saksliste:

1. **Diskutere framdrift**
   Rør er nå sveisa fast i skroget, og utstyr er i ferd med å komme på plass om bord. Har tatt litt tid å 3D-printe innfestningsanordninger for thrusterene, slik at de skal passe inn i røra. Dette er ferdig, og thrusterene er innmontert. Gruppen har også hatt jolla i slepetanken i kjelleren på lab-bygget, hovedsakelig for å sjekke dypgangen til jolla med tanke på plassering av thruster-rør. Neste steg i prosjektet er testing på sjø.

2. **Sted for å teste prototype på sjø**
   Gruppen trenger et egnet sted for å teste prototypen i sjøen. Ottar ringte til Sunnmøre museum for å spørre om gruppen kan jobbe med testing fra deres brygge. Gruppen får beskjed iløpet av dagen om dette er greit (29.04: Tilbakemelding mottatt om at dette var ok).

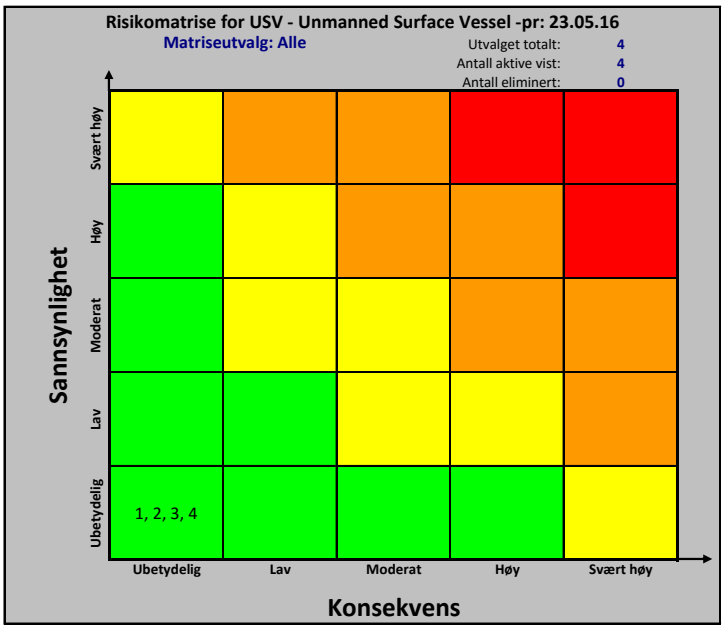
3. **Eventuelt**
   Ottar foreslo å ha jolla i slepetanken igjen nå som alt utstyret er montert, og kjøre en enkel test på skyvkraften til thrusterene ved å bruke fjærvekt(er). Gruppen var enig i at dette er en god ide, og skal spørre ansvarlig for tanken om tillatelse.

   Gruppen har også fått forespørsel fra andre ansatte ved NTNU som er interesserte i prosjektet om det er mulig å få video fra testingen. De nevnte at det fins videolink-utstyr ved NTNU som kan sende video trådløst. Ottar sa at Anders visste om dette utstyret, og gruppen kan låne det av han (29.04: Anders har utstyret liggende, og gruppen kan få låne det).

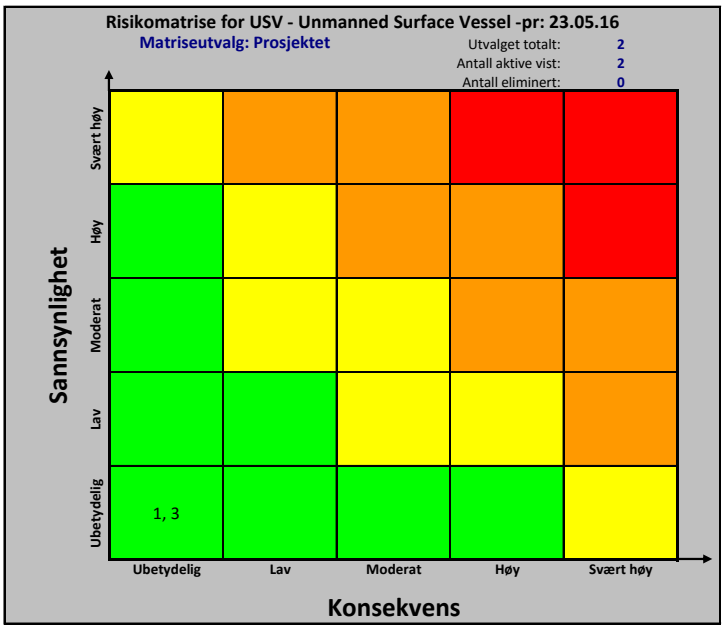
Vedlegg 5
Risikorapport

Risiko og usikkerhetslogg for: USV - Unmanned Surface Vessel

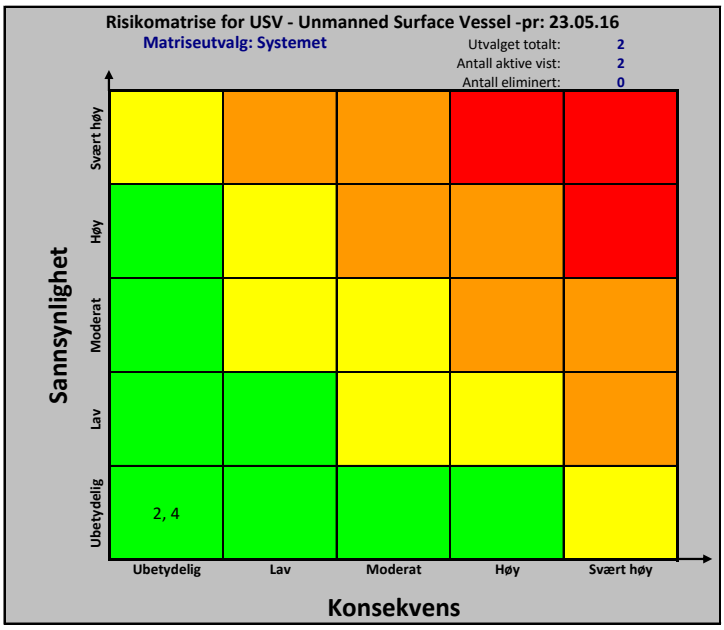
Risikoområde			Mulig konsekvens					Forslag til risikoreduserende tiltak						
Nr	Dato identifisert: Risikobeskrivelse	S	K	E	Dato analysert: Konsekvensbeskrivelse	Risikotype	Dato Kan inntreffe	Dato tiltaksvurdert: Tiltaksbeskrivelse	Detalj-plan	Oppfølging Part	Frist dato:	Ansvar	Dato: Status kommentar:	Status
1	15.02.16: For kort tidsramme for prosjekt	1	1	1	15.02.16: Prosjekt ikke fullført ihht. avtale.	Prosjektet		15.02.16: Jobbe lengre dager, redusere omfang av aktiviteter i samråd med styringsgruppe når evt problemer oppstår	T-1			SL		**Tiltak uført**
4	15.02.16: Utvikling av DP for krevende	1	1	1	15.02.16: Løsning for DP ikke tilfredsstillende ihht. avtale	Systemet		15.02.16: Innhente faglig støtte, velge enkleste løsning	T-4			SL		**Tiltak uført**
3	15.02.16: Uklar prosjektplan	1	1	1	15.02.16: Aktiviteter blir oversett/glemt/ikke utført	Prosjektet		15.02.16: Utarbeide en grundig og klar prosjektplan	T-3			SL		**Tiltak uført**
2	15.02.16: Ikke mulig å anskaffe båt til prototype innenfor budsjett	1	1	1	15.02.16: Får ikke levert/demonstrert prototype	Systemet		15.02.16: Sosiale medier benyttet for å finne båt på bruktmarkedet. Hvis båt ikke kan anskaffes, skifte fokus til å simulere en DP løsning, eller øke budsjett.	T-2			SL		**Tiltak uført**



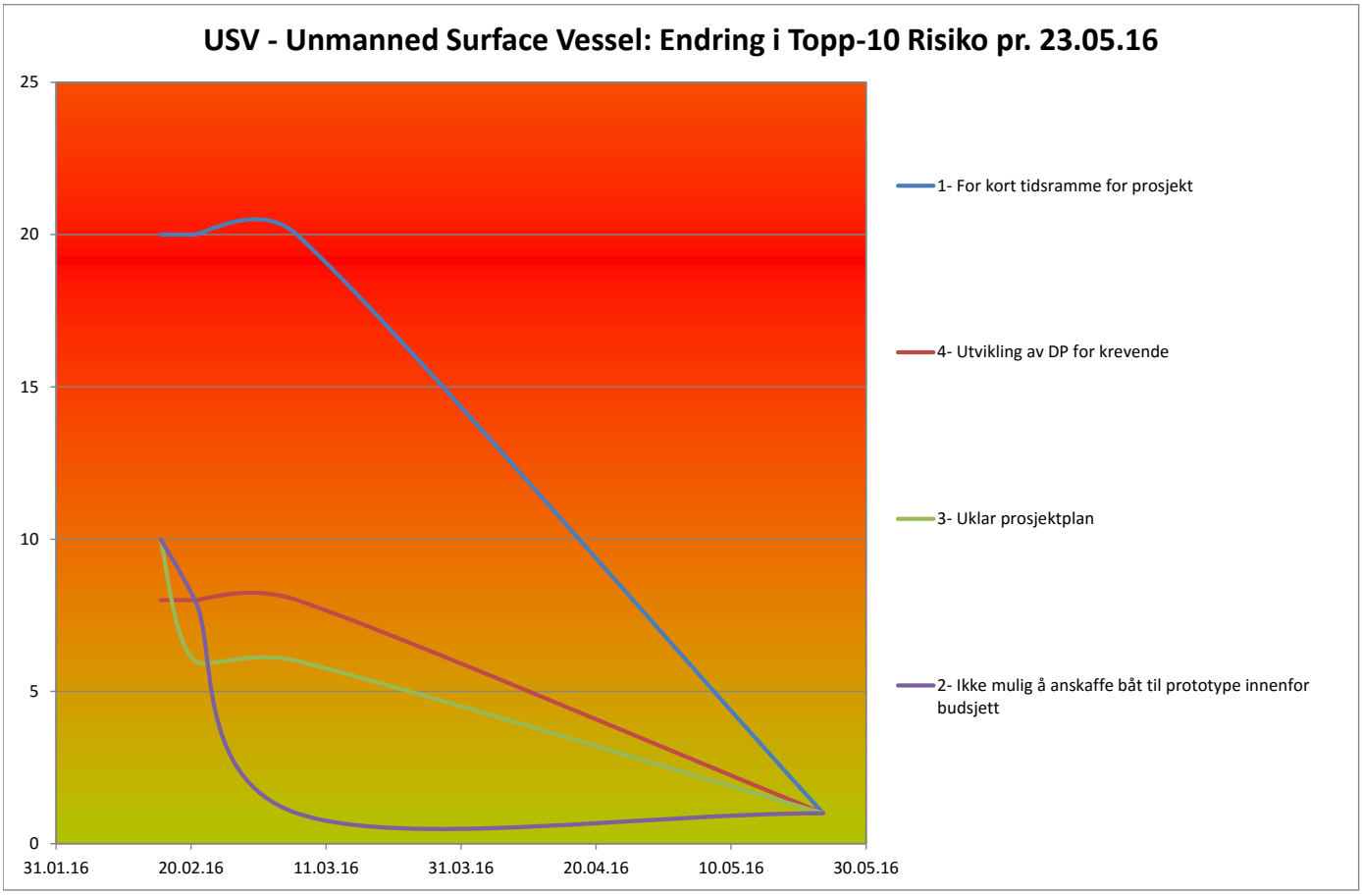
Forklaring: Betydningen av "Konsekvens"		
Verdi		Beskrivelse
1	Ubetydelig	- Ingen nevneverdig forsinkelse - Ingen økonomiske merkostnader - Ubetydelig skade som raskt lar seg rette - Spor av misnøye blant enkelte personer
2	Lav	- Mindre forsinkelser - Lave økonomiske merkostnader - Liten skade som raskt lar seg rette - Misnøye oppstår i enkelte grupper
3	Moderat	- Moderate forsinkelser - Moderate økonomiske merkostnader innenfor budsjett - Moderat skade som kan rettes innenfor en akseptable tid - Moderat misnøye medfører noe fravær i anvendelse / deltagelse
4	Høy	- Lengre forsinkelser - Høy økonomisk merkostnad som går ut over budsjett - Langvarig skade som krever lang tid å rette - Stor grad av misnøy som medfører liten grad av anvendelse / deltagelse
5	Svært høy	- Svært lang forsinkelse eller full stopp - Svært høye økonomiske merkostnader - Stor skade som vanskelig lar seg rette - Svært høy misnøye og ingen anvendelse / deltagelse



Forklaring: Betydningen av "Konsekvens"		
Verdi		Beskrivelse
1	Ubetydelig	- Ingen nevneverdig forsinkelse - Ingen økonomiske merkostnader - Ubetydelig skade som raskt lar seg rette - Spor av misnøye blant enkelte personer
2	Lav	- Mindre forsinkelser - Lave økonomiske merkostnader - Liten skade som raskt lar seg rette - Misnøye oppstår i enkelte grupper
3	Moderat	- Moderate forsinkelser - Moderate økonomiske merkostnader innenfor budsjett - Moderat skade som kan rettes innenfor en akseptable tid - Moderat misnøye medfører noe fravær i anvendelse / deltagelse
4	Høy	- Lengre forsinkelser - Høy økonomisk merkostnad som går ut over budsjett - Langvarig skade som krever lang tid å rette - Stor grad av misnøy som medfører liten grad av anvendelse / deltagelse
5	Svært høy	- Svært lang forsinkelse eller full stopp - Svært høye økonomiske merkostnader - Stor skade som vanskelig lar seg rette - Svært høy misnøye og ingen anvendelse / deltagelse



Forklaring: Betydningen av "Konsekvens"		
Verdi		Beskrivelse
1	Ubetydelig	- Ingen nevneverdig forsinkelse - Ingen økonomiske merkostnader - Ubetydelig skade som raskt lar seg rette - Spor av misnøye blant enkelte personer
2	Lav	- Mindre forsinkelser - Lave økonomiske merkostnader - Liten skade som raskt lar seg rette - Misnøye oppstår i enkelte grupper
3	Moderat	- Moderate forsinkelser - Moderate økonomiske merkostnader innenfor budsjett - Moderat skade som kan rettes innenfor en akseptable tid - Moderat misnøye medfører noe fravær i anvendelse / deltagelse
4	Høy	- Lengre forsinkelser - Høy økonomisk merkostnad som går ut over budsjett - Langvarig skade som krever lang tid å rette - Stor grad av misnøy som medfører liten grad av anvendelse / deltagelse
5	Svært høy	- Svært lang forsinkelse eller full stopp - Svært høye økonomiske merkostnader - Stor skade som vanskelig lar seg rette - Svært høy misnøye og ingen anvendelse / deltagelse

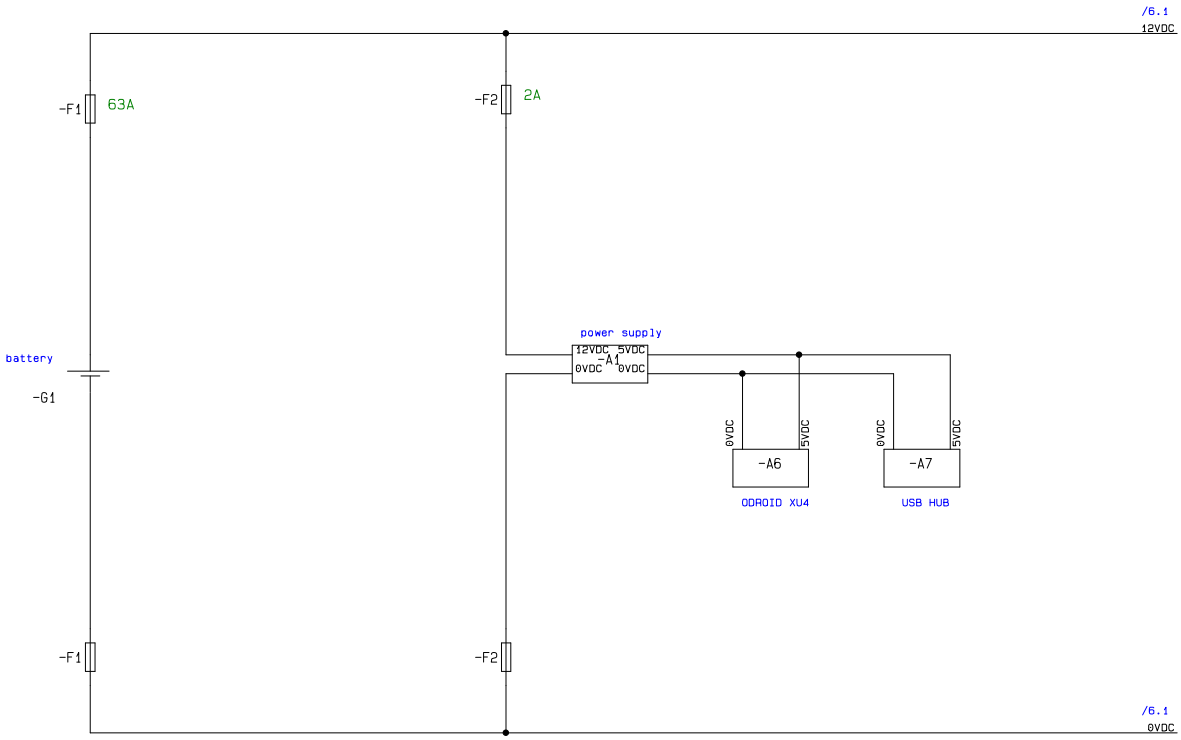

USV - Unmanned Surface Vessel: Endring i Topp-10 Risiko pr. 23.05.16
25
20
15
10
5
0
31.01.16
20.02.16
11.03.16
31.03.16
20.04.16
10.05.16
30.05.16
1- For kort tidsramme for prosjekt
4- Utvikling av DP for krevende
3- Uklar prosjektplan
2- Ikke mulig å anskaffe båt til prototype innenfor budsjett

RiskNr	Risikobeskrivelse	Risiko relatert til	Opprettet	Sist endret	Effekt nå	Status		Aksjoner iverksatt	Aksjoner avsluttet
1	For kort tidsramme for prosjekt	Prosjektet	15.02.2016	23.05.2016	1	Tiltak uført		1	1
2	Ikke mulig å anskaffe båt til prototype innenfor budsjett	Systemet	15.02.2016	06.03.2016	1	Tiltak uført		3	3
3	Uklar prosjektplan	Prosjektet	15.02.2016	23.05.2016	1	Tiltak uført		2	2
4	Utvikling av DP for krevende	Systemet	15.02.2016	23.05.2016	1	Tiltak uført		1	1

Vedlegg 6
Komponentliste prototype

Utstyr	Antall
Pioner 8 mini jolle	1 stk.
BlueRobotics T200 thruster m/ESC	4 stk.
Icy Box 4-port USB hub IB-AC611	1 stk.
Odroid XU4	1 stk.
Realtek RTL8188CUS-GR WLAN USB antenne	1 stk.
Arduino UNO m/USB kabel	4 stk.
Razor 9 DOF IMU	1 stk.
Sparkfun Atmospheric Sensor Breakout - BME280	1 stk.
Mean Well SD-25A-5 DC/DC-omformer	1 stk.
DIN-skinne	0,3 m.
Schneider Electric iC60H C 2A	1 stk.
Schneider Electric iC60H C 16A	4 stk.
Schneider Electric iC60H C 63A	1 stk.
Samleskinne, 2P pinne 16 mm^2	0,2 m.
Fibox EKOE 130 G kapsling	2 stk.
Fibox EKO 30-G lokk	2 stk.
Hensel D9125 koblingsboks	4 stk.
M20 nippel	19 stk.
Weidmüller WDU 2.5 rekkeklemme	33 stk.
Weidmüller WAP WDU 2.5N endeplate	1 stk.
Weidmüller WEW 35/2 endestopper	2 stk.
Biltema 12V 88 Ah batteri	1 stk.
Biltema batteripolsko	2 stk.
Ölflex 3x0,75 mm^2	1,5 m.
PFXP 3x2,5 mm^2	5 m.
Systimax gigaSpeed XL Cat 6 kabel	1,5 m.
Endehylse 16 mm^2	2 stk.
Endehylse 2,5 mm^2	12 stk.
Endehylse 1,5 mm^2	12 stk.
Endehylse 1 mm^2	4 stk.
Endehylse 0,75 mm^2	6 stk.
Biltema borrelåsbånd 5 cm	2 m.
PE100 rør	1,6 m.

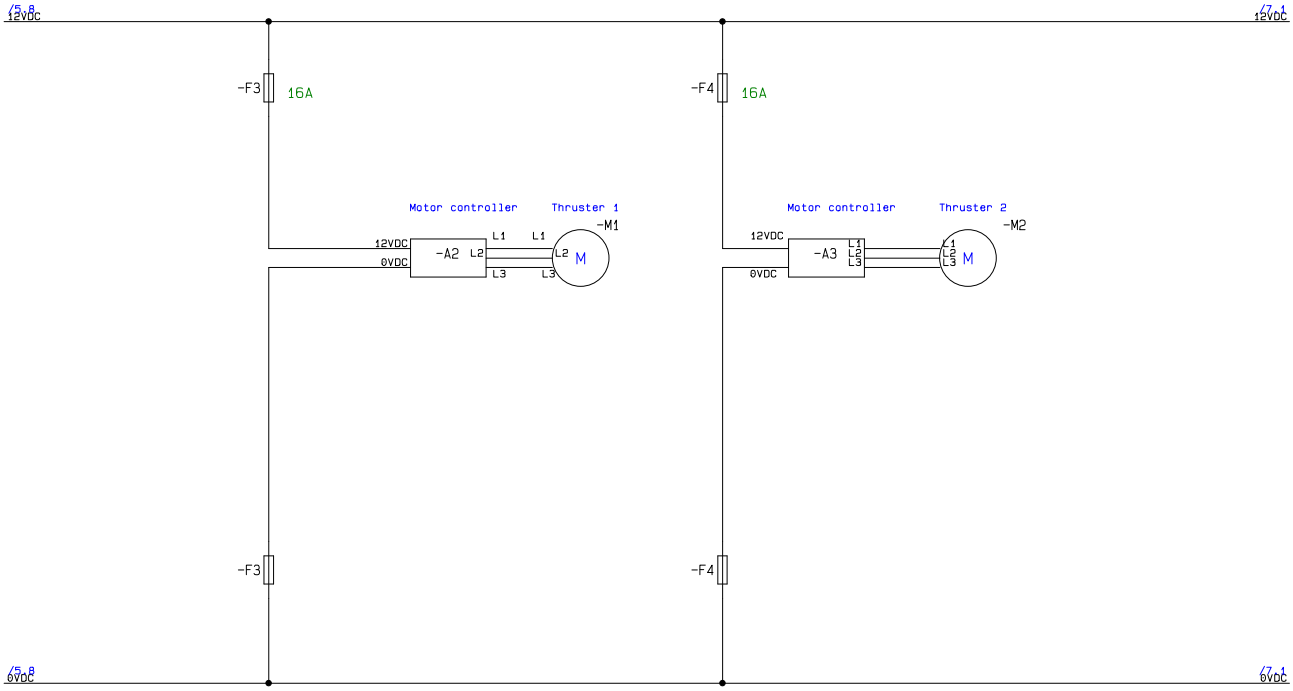
Vedlegg 7
Hovedstrømskjema



PC|SCHEMATIC Automation

NTNU
Kunnskap for en bedre verden

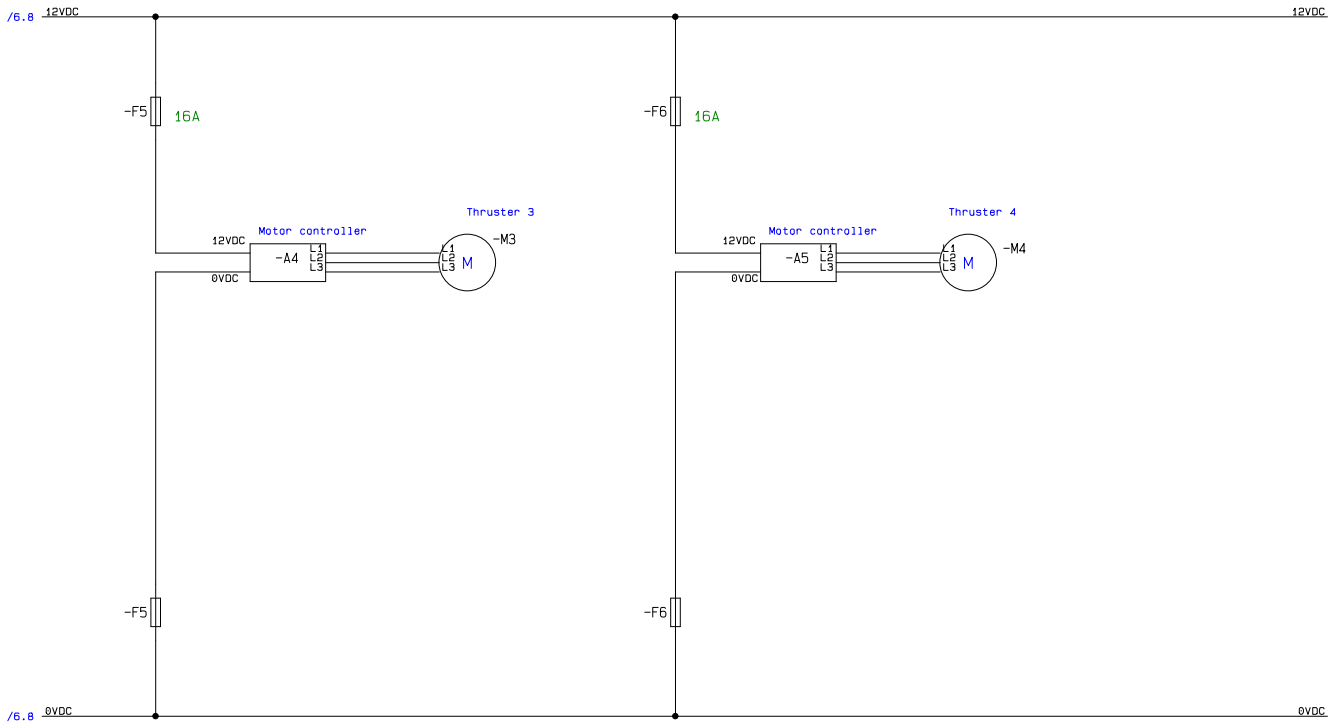
Project title:USV	**Project no.:**	**Project rev.:**	**Page**	**5**
Customer: NTNU i Ålesund	DCC:		Scale:	1:1
Page title:	Drawing no.:	Page rev.:	Previous page:	4
Filename: Hovedstrømsskjema	Constructor (project/page)	Last printed: 23.05.2016	Next page:	6
Page ref.:	Appr. (date/sign.)	Last correction: 22.05.2016	Number of pages:	20



PC|SCHEMATIC Automation

NTNU
Kunnskap for en bedre verden

Project title:USV	**Project no.:**	**Project rev.:**	**Page**	**6**
Customer: NTNU i Ålesund	DCC:		Scale:	1:1
Page title:	Drawing no.:	Page rev.:	Previous page:	5
Filename: Hovedstrømsskjema	Constructor (project/page)	Last printed: 23.05.2016	Next page:	7
Page ref.:	Appr. (date/sign.)	Last correction: 23.05.2016	Number of pages:	20

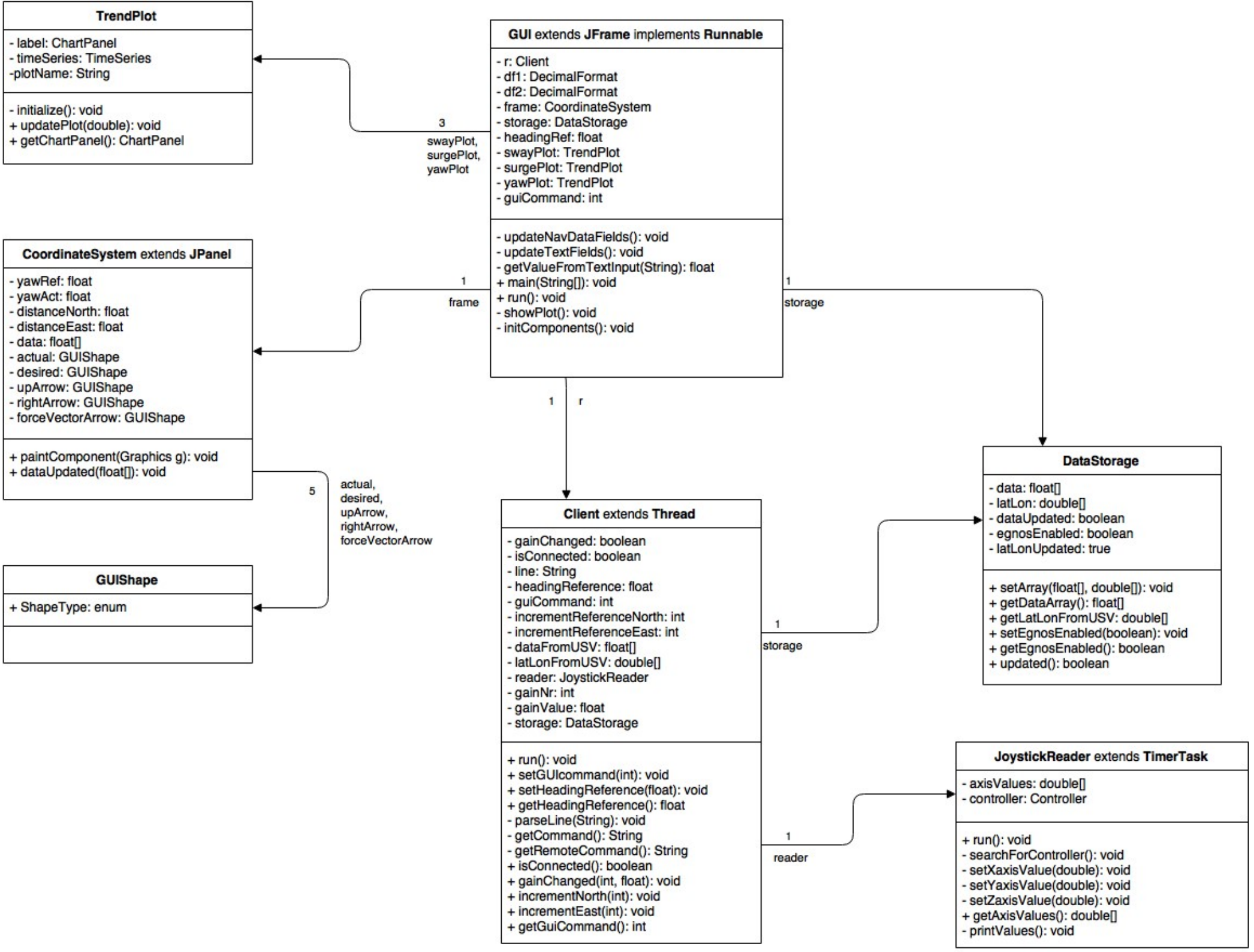


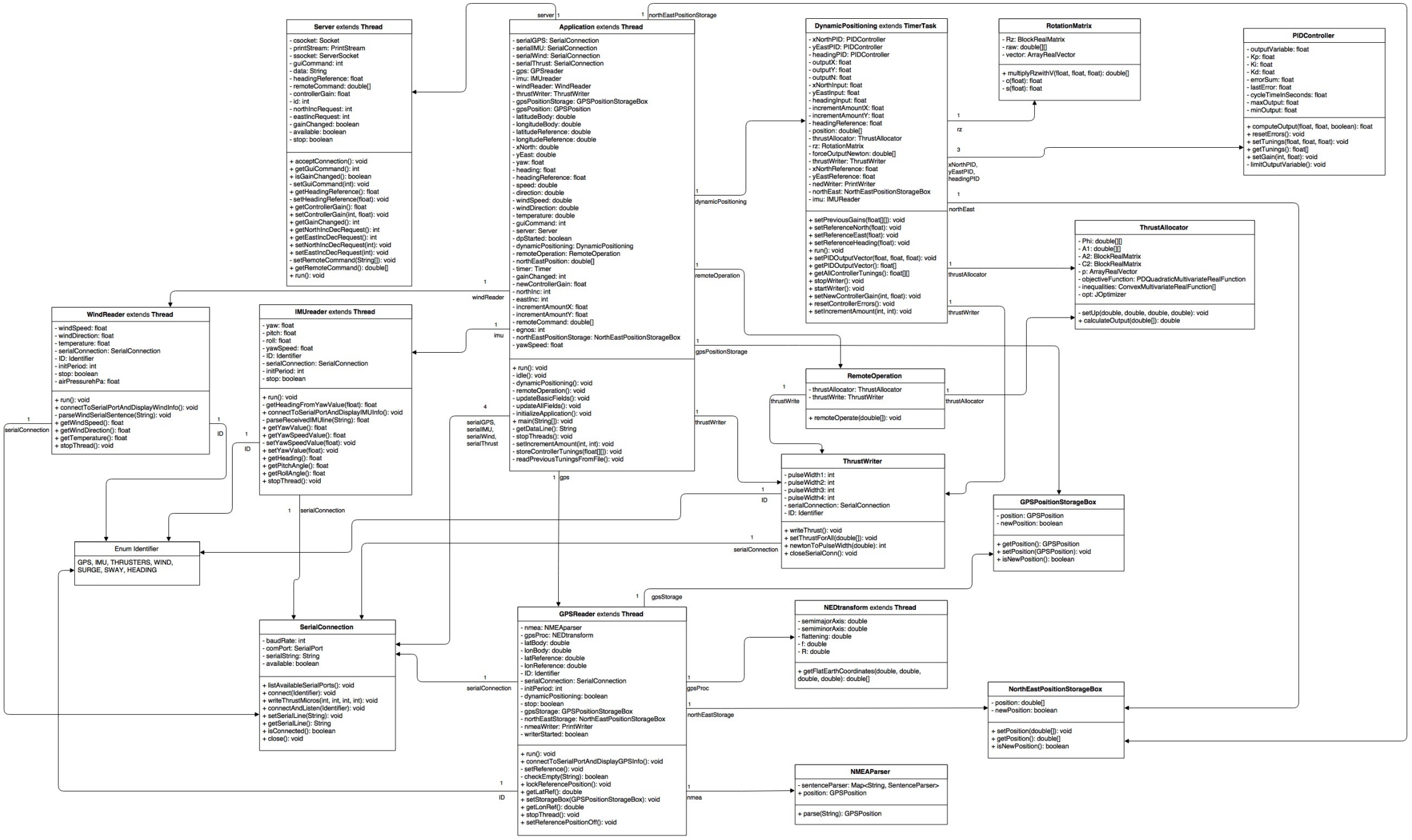
PC|SCHEMATIC Automation

NTNU
Kunnskap for en bedre verden

Project title:USV	**Project no.:**	**Project rev.:**	**Page** 7
Customer: NTNU i Ålesund	DCC:		Scale: 1:1
Page title:	Drawing no.:	Page rev.:	Previous page: 6
Filename: Hovedstrømsskjema	Constructor (project/page)	Last printed: 23.05.2016	Next page: 9
Page ref.:	Appr. (date/sign.)	Last correction: 23.05.2016	Number of pages: 20

Vedlegg 8
Klassediagrammer


TrendPlot
- label: ChartPanel
- timeSeries: TimeSeries
-plotName: String
- initialize(): void
+ updatePlot(double): void
+ getChartPanel(): ChartPanel
3
swayPlot,
surgePlot,
yawPlot
GUI extends JFrame implements Runnable
- r: Client
- df1: DecimalFormat
- df2: DecimalFormat
- frame: CoordinateSystem
- storage: DataStorage
- headingRef: float
- swayPlot: TrendPlot
- surgePlot: TrendPlot
- yawPlot: TrendPlot
- guiCommand: int
- updateNavDataFields(): void
- updateTextFields(): void
- getValueFromTextInput(String): float
+ main(String[]): void
+ run(): void
- showPlot(): void
- initComponents(): void
CoordinateSystem extends JPanel
- yawRef: float
- yawAct: float
- distanceNorth: float
- distanceEast: float
- data: float[]
- actual: GUIShape
- desired: GUIShape
- upArrow: GUIShape
- rightArrow: GUIShape
- forceVectorArrow: GUIShape
+ paintComponent(Graphics g): void
+ dataUpdated(float[]): void
1
frame
1
storage
1
r
5
actual,
desired,
upArrow,
rightArrow,
forceVectorArrow
GUIShape
+ ShapeType: enum
DataStorage
- data: float[]
- latLon: double[]
- dataUpdated: boolean
- egnosEnabled: boolean
- latLonUpdated: true
+ setArray(float[], double[]): void
+ getDataArray(): float[]
+ getLatLonFromUSV: double[]
+ setEgnosEnabled(boolean): void
+ getEgnosEnabled(): boolean
+ updated(): boolean
Client extends Thread
- gainChanged: boolean
- isConnected: boolean
- line: String
- headingReference: float
- guiCommand: int
- incrementReferenceNorth: int
- incrementReferenceEast: int
- dataFromUSV: float[]
- latLonFromUSV: double[]
- reader: JoystickReader
- gainNr: int
- gainValue: float
- storage: DataStorage
+ run(): void
+ setGUIcommand(int): void
+ setHeadingReference(float): void
+ getHeadingReference(): float
- parseLine(String): void
- getCommand(): String
- getRemoteCommand(): String
+ isConnected(): boolean
+ gainChanged(int, float): void
+ incrementNorth(int): void
+ incrementEast(int): void
+ getGuiCommand(): int
1
storage
1
reader
JoystickReader extends TimerTask
- axisValues: double[]
- controller: Controller
+ run(): void
- searchForController(): void
- setXaxisValue(double): void
- setYaxisValue(double): void
- setZaxisValue(double): void
+ getAxisValues(): double[]
- printValues(): void


Server extends Thread
- csocket: Socket
- printStream: PrintStream
- ssocket: ServerSocket
- guiCommand: int
- data: String
- headingReference: float
- remoteCommand: double[]
- controllerGain: float
- id: int
- northIncRequest: int
- eastIncRequest: int
- gainChanged: boolean
- available: boolean
- stop: boolean
+ acceptConnection(): void
+ getGuiCommand(): int
+ isGainChanged(): boolean
- setGuiCommand(int): void
+ getHeadingReference(): float
- setHeadingReference(float): void
+ getControllerGain(): float
+ setControllerGain(int, float): void
+ getGainChanged(): int
+ getNorthIncDecRequest(): int
+ getEastIncDecRequest(): int
+ setNorthIncDecRequest(int): void
+ setEastIncDecRequest(int): void
- setRemoteCommand(String[]): void
+ getRemoteCommand(): double[]
+ run(): void
server
1
Application extends Thread
- serialGPS: SerialConnection
- serialIMU: SerialConnection
- serialWind: SerialConnection
- serialThrust: SerialConnection
- gps: GPSreader
- imu: IMUreader
- windReader: WindReader
- thrustWriter: ThrustWriter
- gpsPositionStorage: GPSPositionStorageBox
- gpsPosition: GPSPosition
- latitudeBody: double
- longitudeBody: double
- latitudeReference: double
- longitudeReference: double
- xNorth: double
- yEast: double
- yaw: float
- heading: float
- headingReference: float
- speed: double
- direction: double
- windSpeed: double
- windDirection: double
- temperature: double
- guiCommand: int
- server: Server
- dpStarted: boolean
- dynamicPositioning: DynamicPositioning
- remoteOperation: RemoteOperation
- northEastPosition: double[]
- timer: Timer
- gainChanged: int
- newControllerGain: float
- northInc: int
- eastInc: int
- incrementAmountX: float
- incrementAmountY: float
- remoteCommand: double[]
- egnos: int
- northEastPositionStorage: NorthEastPositionStorageBox
- yawSpeed: float
+ run(): void
- idle(): void
- dynamicPositioning(): void
- remoteOperation(): void
- updateBasicFields(): void
- updateAllFields(): void
- initializeApplication(): void
+ main(String[]): void
- getDataLine(): String
- stopThreads(): void
- setIncrementAmount(int, int): void
- storeControllerTunings(float[][]): void
- readPreviousTuningsFromFile(): void
northEastPositionStorage
1
1
DynamicPositioning extends TimerTask
- xNorthPID: PIDController
- yEastPID: PIDController
- headingPID: PIDController
- outputX: float
- outputY: float
- outputN: float
- xNorthInput: float
- yEastInput: float
- headingInput: float
- incrementAmountX: float
- incrementAmountY: float
- headingReference: float
- position: double[]
- thrustAllocator: ThrustAllocator
- rz: RotationMatrix
- forceOutputNewton: double[]
- thrustWriter: ThrustWriter
- xNorthReference: float
- yEastReference: float
- nedWriter: PrintWriter
- northEast: NorthEastPositionStorageBox
- imu: IMUReader
+ setPreviousGains(float[][]): void
+ setReferenceNorth(float): void
+ setReferenceEast(float): void
+ setReferenceHeading(float): void
+ run(): void
+ setPIDOutputVector(float, float, float): void
+ getPIDOutputVector(): float[]
+ getAllControllerTunings(): float[][]
+ stopWriter(): void
+ startWriter(): void
+ setNewControllerGain(int, float): void
+ resetControllerErrors(): void
+ setIncrementAmount(int, int): void
dynamicPositioning
1
RotationMatrix
- Rz: BlockRealMatrix
- raw: double[][]
- vector: ArrayRealVector
+ multiplyRzwithV(float, float, float): double[]
- c(float): float
- s(float): float
1
rz
PIDController
- outputVariable: float
- Kp: float
- Ki: float
- Kd: float
- errorSum: float
- lastError: float
- cycleTimeInSeconds: float
- maxOutput: float
- minOutput: float
+ computeOutput(float, float, boolean): float
+ resetErrors(): void
+ setTunings(float, float, float): void
+ getTunings(): float[]
+ setGain(int, float): void
- limitOutputVariable(): void
3
xNorthPID,
yEastPID,
headingPID
1
northEast
1
thrustAllocator
ThrustAllocator
- Phi: double[][]
- A1: double[][]
- A2: BlockRealMatrix
- C2: BlockRealMatrix
- p: ArrayRealVector
- objectiveFunction: PDQuadraticMultivariateRealFunction
- inequalities: ConvexMultivariateRealFunction[]
- opt: JOptimizer
- setUp(double, double, double, double): void
+ calculateOutput(double[]): double
1
thrustWriter
remoteOperation
1
gpsPositionStorage
1
RemoteOperation
- thrustAllocator: ThrustAllocator
- thrustWrite: ThrustWriter
+ remoteOperate(double[]): void
1
thrustWrite
1
thrustAllocator
1
thrustWriter
ThrustWriter
- pulseWidth1: int
- pulseWidth2: int
- pulseWidth3: int
- pulseWidth4: int
- serialConnection: SerialConnection
- ID: Identifier
+ writeThrust(): void
+ setThrustForAll(double[]): void
+ newtonToPulseWidth(double): int
+ closeSerialConn(): void
1
ID
1
serialConnection
GPSPositionStorageBox
- position: GPSPosition
- newPosition: boolean
+ getPosition(): GPSPosition
+ setPosition(GPSPosition): void
+ isNewPosition(): boolean
WindReader extends Thread
- windSpeed: float
- windDirection: float
- temperature: float
- serialConnection: SerialConnection
- ID: Identifier
- initPeriod: int
- stop: boolean
- airPressurehPa: float
+ run(): void
+ connectToSerialPortAndDisplayWindInfo(): void
- parseWindSerialSentence(String): void
+ getWindSpeed(): float
+ getWindDirection(): float
+ getTemperature(): float
+ stopThread(): void
windReader
1
1
serialConnection
1
ID
IMUreader extends Thread
- yaw: float
- pitch: float
- roll: float
- yawSpeed: float
- ID: Identifier
- serialConnection: SerialConnection
- initPeriod: int
- stop: boolean
+ run(): void
- getHeadingFromYawValue(float): float
+ connectToSerialPortAndDisplayIMUInfo(): void
- parseReceivedIMUline(String): float
+ getYawValue(): float
+ getYawSpeedValue(): float
- setYawSpeedValue(float): void
+ setYawValue(float): void
+ getHeading(): float
+ getPitchAngle(): float
+ getRollAngle(): float
+ stopThread(): void
imu
1
1
ID
4
serialGPS,
serialIMU,
serialWind,
serialThrust
1
serialConnection
Enum Identifier
GPS, IMU, THRUSTERS, WIND,
SURGE, SWAY, HEADING
SerialConnection
- baudRate: int
- comPort: SerialPort
- serialString: String
- available: boolean
+ listAvailableSerialPorts(): void
+ connect(Identifier): void
+ writeThrustMicros(int, int, int, int): void
+ connectAndListen(Identifier): void
+ setSerialLine(String): void
+ getSerialLine(): String
+ isConnected(): boolean
+ close(): void
1
serialConnection
1
gps
1
gpsStorage
GPSReader extends Thread
- nmea: NMEAparser
- gpsProc: NEDtransform
- latBody: double
- lonBody: double
- latReference: double
- lonReference: double
- ID: Identifier
- serialConnection: SerialConnection
- initPeriod: int
- dynamicPositioning: boolean
- stop: boolean
- gpsStorage: GPSPositionStorageBox
- northEastStorage: NorthEastPositionStorageBox
- nmeaWriter: PrintWriter
- writerStarted: boolean
+ run(): void
+ connectToSerialPortAndDisplayGPSInfo(): void
- setReference(): void
- checkEmpty(String): boolean
+ lockReferencePosition(): void
+ getLatRef(): double
+ setStorageBox(GPSPositionStorageBox): void
+ getLonRef(): double
+ stopThread(): void
+ setReferencePositionOff(): void
1
ID
1
gpsProc
1
northEastStorage
1
nmea
NEDtransform extends Thread
- semimajorAxis: double
- semiminorAxis: double
- flattening: double
- f: double
- R: double
+ getFlatEarthCoordinates(double, double, double, double): double[]
NorthEastPositionStorageBox
- position: double[]
- newPosition: boolean
+ setPosition(double[]): void
+ getPosition(): double[]
+ isNewPosition(): boolean
NMEAParser
- sentenceParser: Map<String, SentenceParser>
+ position: GPSPosition
+ parse(String): GPSPosition

Vedlegg 9
Kildekode klientapplikasjon

```
00001: package application;
00002:
00003: import java.io.BufferedReader;
00004: import java.io.File;
00005: import java.io.FileWriter;
00006: import java.io.IOException;
00007: import java.io.InputStreamReader;
00008: import java.io.PrintStream;
00009: import java.io.PrintWriter;
00010: import java.net.Socket;
00011: import java.util.Date;
00012:
00013: /**
00014:  *
00015:  * @author Albert
00016:  * klasse for Klient-serverkommunikasjon mellom USV og land
00017:  *
00018:  */
00019: public class Client extends Thread {
00020:
00021:     private boolean gainChanged;
00022:     private boolean isConnected;
00023:     private String line;
00024:
00025:     private float headingReference;
00026:
00027:     private int guiCommand;
00028:     private int incrementReferenceNorth;
00029:     private int incrementReferenceEast;
00030:
00031:     private float[] dataFromUSV;
00032:     private double[] latLonFromUSV;
00033:
```

```
00034:     private JoystickReader reader;
00035:     private int gainNr;
00036:     private float gainValue;
00037:     private DataStorage storage;
00038:     private PrintWriter nedWriter;
00039:     private boolean writing;
00040:
00041:     public Client(JoystickReader reader, DataStorage storage) {
00042:         this.reader = reader;
00043:         this.storage = storage;
00044:         isConnected = gainChanged = writing = false;
00045:         incrementReferenceNorth = incrementReferenceEast = 0;
00046:         guiCommand = 0;
00047:         latLonFromUSV = new double[2];
00048:         dataFromUSV = new float[23];
00049:     }
00050:
00051:     @Override
00052:     public void run() {
00053:         while (true) {
00054:             System.out.println("GUI Command: " +
00055:                     guiCommand + " in FIRST While-loop in Reader");
00056:             // Bruker trykker connect
00057:             if (guiCommand == 3) {
00058:                 try {
00059:                     // Opprett ny socket
00060:                     Socket clientSocket = new Socket("192.168.0.101", 2345);
00061:
00062:                     BufferedReader inFromServer;
00063:                     PrintStream pstream = new PrintStream(clientSocket.
00064:                             getOutputStream());
00065:                     long lastTime = 0;
00066:                     long delay = 100;
```

```
00067:                     while (!clientSocket.isClosed()) {
00068:                         // Send data med 100 ms mellomrom
00069:                         if (lastTime + delay < System.currentTimeMillis()) {
00070:                             System.out.println("GUI Command: "
00071:                                     + guiCommand
00072:                                     + " in SECOND While-loop in Reader");
00073:                             isConnected = true;
00074:                             // Hvis fjernstyring
00075:                             if (guiCommand == 2) {
00076:                                 pstream.println(getRemoteCommand());
00077:                             } else {
00078:                                 pstream.println(getCommand());
00079:                             }
00080:                             inFromServer =
00081:                                     new BufferedReader(new InputStreamReader
00082:                                             (clientSocket.getInputStream()));
00083:                             // Les data fra USV
00084:                             line = inFromServer.readLine();
00085:                             if (line != null) {
00086:                                 if (!line.isEmpty()) {
00087:                                     parseLine(line);
00088:                                 }
00089:                             }
00090:                             if (guiCommand == 1) {
00091:                                 if (!writing) startWriter();
00092:                                 storeDataFromUSV();
00093:                             }
00094:                             else stopWriter();
00095:                             if (guiCommand == 4) {
00096:
00097:                                 pstream.println("" + 4 + " " + 0 + " " + 0 + " "
00098:                                         + 0f + " " + 0 + " " + 0);
00099:                                 pstream.close();
```

```
00100:                                 clientSocket.close();
00101:                                 System.out.println("Socket CLOSED");
00102:                             }
00103:
00104:                             lastTime = System.currentTimeMillis();
00105:                         }
00106:
00107:                     }
00108:
00109:                     isConnected = false;
00110:                 } catch (IOException ex) {
00111:                     System.out.println("IOexeption");
00112:                 }
00113:             }
00114:         }
00115:     }
00116:
00117:     public synchronized void setGUIcommand(int i) {
00118:         guiCommand = i;
00119:     }
00120:
00121:     public synchronized void setHeadingReference(float reference) {
00122:         headingReference = reference;
00123:     }
00124:
00125:     public synchronized float getHeadingReference() {
00126:         return headingReference;
00127:     }
00128:
00129:
00130:    /*
00131:     Metode for parsing av datalinje fra USV
00132:     */
```

```
00133:     private synchronized void parseLine(String line) {
00134:
00135:         String[] lineData = line.split(" ");
00136:         if (lineData.length > 24) {
00137:             // breddegrad USV
00138:             latLonFromUSV[0] = Double.parseDouble(lineData[1]);
00139:             // lengdegrad USV
00140:             latLonFromUSV[1] = Double.parseDouble(lineData[3]);
00141:             // avstand nord
00142:             dataFromUSV[0] = Float.parseFloat(lineData[5]);
00143:             // avstand Ã¸st
00144:             dataFromUSV[1] = Float.parseFloat(lineData[7]);
00145:             // peiling (grader)
00146:             dataFromUSV[2] = Float.parseFloat(lineData[9]);
00147:             // hastighet
00148:             dataFromUSV[3] = Float.parseFloat(lineData[11]);
00149:             // hastighetsretning
00150:             dataFromUSV[4] = Float.parseFloat(lineData[13]);
00151:             // vindhastighet (ikke i bruk)
00152:             dataFromUSV[5] = Float.parseFloat(lineData[15]);
00153:             // vindretning (ikke i bruk)
00154:             dataFromUSV[6] = Float.parseFloat(lineData[17]);
00155:             // temperatur (i kapsling)
00156:             dataFromUSV[7] = Float.parseFloat(lineData[19]);
00157:             // breddegrad referanse
00158:             dataFromUSV[8] = Float.parseFloat(lineData[21]);
00159:             // lengdegrad referanse
00160:             dataFromUSV[9] = Float.parseFloat(lineData[23]);
00161:             // PID gain: jaging (kp ki kd) sidevis (kp ki kd) giring (kp ki kd)
00162:
00163:             for (int i = 0; i < 9; i++) {
00164:                 dataFromUSV[i + 10] = Float.parseFloat(lineData[i + 24]);
00165:             }
```

```
00166:             // kraft: X, Y, N, vinkelhastighet
00167:             for (int i = 0; i < 4; i++) {
00168:                 dataFromUSV[i + 19] = Float.parseFloat(lineData[i + 34]);
00169:             }
00170:             // Lagre mottatt data
00171:             storage.setArray(dataFromUSV, latLonFromUSV);
00172:             int egnosEnabled = Integer.parseInt(lineData[33]);
00173: 
00174:             //System.out.println("EGNOS: " + egnosEnabled);
00175:             // Sjekk om DGPS
00176:             if (egnosEnabled == 2) {
00177:                 storage.setEgnosEnabled(true);
00178:             } else {
00179:                 storage.setEgnosEnabled(false);
00180:             }
00181:         }
00182:     }
00183: 
00184:     private synchronized String getCommand() {
00185:         // data format: gui kommando, nummer pÃ¥ regulator gain som skal tunes,
00186:         // ny heading referanse til dp-kontroller, verdi pÃ¥ gain som skal
00187:         // endres, antall halvmeter referansen flyttes nordover,
00188:         // antall halvmeter referansen flyttes Ã¸stover
00189:         // (-1 vil da vÃ¦re hhv 0,5 m sÃ¸r eller vest)
00190:         int a = incrementReferenceNorth;
00191:         int b = incrementReferenceEast;
00192:         incrementReferenceNorth = 0;
00193:         incrementReferenceEast = 0;
00194:         if (!gainChanged) {
00195:             return guiCommand + " " + 0 + " " + headingReference + " "
00196:                     + 0f + " " + a + " " + b;
00197:         } else {
00198:             // Forandret parameter i PID regulering
```

```
00199:             gainChanged = false;
00200:             return guiCommand + " " + gainNr + " " + headingReference + " "
00201:                     + gainValue + " " + a + " " + b;
00202:         }
00203:     }
00204:     // Les data fra joystick og send til USV
00205:     private synchronized String getRemoteCommand() {
00206:         double axisValues[] = reader.getAxisValues();
00207:         return guiCommand + " " + 0 + " X-Y-Yaw: "
00208:                 + axisValues[0] + " " + axisValues[1]
00209:                 + " " + axisValues[2];
00210:
00211:     }
00212:
00213:     public boolean isConnected() {
00214:         return isConnected;
00215:     }
00216:
00217:     // Sett ny forsterkning til PID regulator
00218:     public synchronized void gainChanged(int gainNr, float value) {
00219:         this.gainNr = gainNr;
00220:         gainValue = value;
00221:         gainChanged = true;
00222:     }
00223:
00224:     // Flytt DP-settpunkt nord
00225:     public synchronized void incrementNorth(int increment) {
00226:         incrementReferenceNorth += increment;
00227:     }
00228:
00229:     // Flytt DP-settpunkt Ã¸st
00230:     public synchronized void incrementEast(int increment) {
00231:         incrementReferenceEast += increment;
```

```
00232:     }
00233:
00234:     int getGuiCommand() {
00235:         return guiCommand;
00236:     }
00237:
00238:     // Lagre NED-data til fil
00239:     private void storeDataFromUSV() {
00240:         nedWriter.println(dataFromUSV[0] + " " + dataFromUSV[1] + " "
00241:                 + (headingReference - dataFromUSV[2]) + " "
00242:                 + latLonFromUSV[0] + " " + latLonFromUSV[1]);
00243:     }
00244:
00245:
00246:     // Start filskriver
00247:     private void startWriter() {
00248:         File log = new File("NED_Data.txt");
00249:
00250:         try {
00251:             log.createNewFile();
00252:             nedWriter = new PrintWriter(new FileWriter(log, true));
00253:             nedWriter.println(new Date().toString());
00254:             writing = true;
00255:         } catch (IOException ex) {
00256:         }
00257:     }
00258:
00259:     // Stopp filskriver
00260:     private void stopWriter() {
00261:         if (nedWriter != null) {
00262:             nedWriter.close();
00263:             writing = false;
00264:         }
```

```
00265:     }
00266: }
```

```
00001: /*
00002:  * To change this license header, choose License Headers in Project Properties.
00003:  * To change this template file, choose Tools | Templates
00004:  * and open the template in the editor.
00005:  */
00006: package application;
00007:
00008: import java.awt.*;
00009: import java.awt.geom.AffineTransform;
00010: import java.awt.geom.Path2D;
00011: import static java.lang.Math.atan;
00012:
00013: import javax.swing.JPanel;
00014: import javax.swing.SwingUtilities;
00015:
00016: /**
00017:  *
00018:  * @author vegard
00019:  * Klasse for koordinatsystemet. Tegner alle komponenter, og flytter de rundt
00020:  * basert pÃ¥ data fra USV
00021:  */
00022: public class CoordinateSystem extends JPanel {
00023:
00024:     private float yawRef;
00025:     private float yawAct;
00026:     private float distanceNorth;
00027:     private float distanceEast;
00028:     private float[] data;
00029:     private GUIShape actual;
00030:     private GUIShape desired;
00031:     private GUIShape upArrow;
00032:     private GUIShape rightArrow;
00033:     private GUIShape forceVectorArrow;
```

```
00034:
00035:     public CoordinateSystem(int width, int height) {
00036:         setSize(width, height);
00037:         actual = new GUIShape(GUIShape.ShapeType.BOAT);
00038:         desired = new GUIShape(GUIShape.ShapeType.BOAT);
00039:         upArrow = new GUIShape(GUIShape.ShapeType.UP_ARROW);
00040:         rightArrow = new GUIShape(GUIShape.ShapeType.RIGHT_ARROW);
00041:         forceVectorArrow = new GUIShape(GUIShape.ShapeType.UP_ARROW);
00042:     }
00043:
00044:     @Override
00045:     public void paintComponent(Graphics g) {
00046: //        long time1 = System.currentTimeMillis();
00047:         super.paintComponent(g);
00048:         Graphics2D g2d = (Graphics2D) g.create();
00049:         // Tegner koordinatsystem fÃ¸rst
00050:         int i;
00051:
00052:         int height = this.getBounds().height;
00053:         int width = this.getBounds().width;
00054:         // tegn radene
00055:
00056:         int rowHt = (height / 10);
00057:         for (i = 0; i < 10; i++) {
00058:             g.drawLine(0, i * rowHt, width, i * rowHt);
00059:         }
00060:         g.drawLine(0, 501, 501, 501);
00061:         // tegn kolonnene
00062:         int rowWid = (width / 10);
00063:         for (i = 0; i < 10; i++) {
00064:             g.drawLine(i * rowWid, 0, i * rowWid, height);
00065:         }
00066:         g.drawLine(501, 0, 501, 500);
```

```
00067:         // Sett navn pÃ¥ aksene
00068:         g.setFont(new Font("Serif", Font.BOLD, 15));
00069:         g.setColor(Color.red);
00070:
00071:         g.drawLine(250, 0, 250, 500);
00072:         g.drawLine(0, 250, 500, 250);
00073:         g.drawChars(new char[]{'N', 'o', 'r', 't', 'h'}, 0, 5, 260, 15);
00074:         g.drawChars(new char[]{'E', 'a', 's', 't'}, 0, 4, 465, 270);
00075:
00076:         yawRef = data[0];
00077:         yawAct = data[1];
00078:         distanceNorth = data[2]*100;
00079:         distanceEast = data[3]*100;
00080:         float forceX = data[4]*3;
00081:         float forceY = data[5]*3;
00082:
00083:
00084:         // Trenger tre lineÃ¦re transformasjoner
00085:         AffineTransform at1 = new AffineTransform();
00086:         AffineTransform at2 = new AffineTransform();
00087:         AffineTransform at3 = new AffineTransform();
00088:
00089:         // LineÃ¦r transformasjon av "skipsboksene"
00090:         at1.translate(((width / 2) + (distanceEast - 10)),
00091:                 ((height / 2) - (distanceNorth + 30)));
00092:         // Tegn kraftvektoren fra PIDene
00093:         g.setColor(Color.RED);
00094:
00095:         // parameter i drawline: x1,y1,x2,y2
00096:         g.drawLine((int) ((width / 2) + distanceEast),
00097:                 (int) ((height / 2) - distanceNorth),
00098:                 (int) ((width / 2) + distanceEast + forceY),
00099:                 (int) ((height / 2) - distanceNorth - forceX));
```

```
00100:
00101:         // Translasjon av pilen
00102:         at3.translate(distanceEast + forceY,
00103:                 (height/2)-(distanceNorth + forceX)-3);
00104:         float arrowDir = (float) atan(forceX/forceY);
00105:
00106:         // atan() returnerer -pi/2 til pi/2
00107:         // sjekk om rotasjonen er utenfor dette omrÃ¥det
00108:         // Pilen roterer om spissen
00109:         if(forceY < 0f) {
00110:             // Pilen starter med vinkel pi/2
00111:         at3.rotate(-arrowDir + 3*Math.PI/2,250,4);
00112:         }
00113:         else at3.rotate(-arrowDir+Math.PI/2, 250, 4);
00114:
00115:         Shape force = new Path2D.Float(forceVectorArrow, at3);
00116:
00117: //         Rotasjon om senter
00118:         at1.rotate(Math.toRadians(yawAct), 10, 30);
00119:         Shape actShape = new Path2D.Float(actual, at1);
00120:
00121:         at2.translate((width / 2) - 10, (height / 2) - 30);
00122:         at2.rotate(Math.toRadians(yawRef), 10, 30);
00123:         Shape desShape = new Path2D.Float(desired, at2);
00124:
00125:         // Tegne skipene i koordinatsystemet
00126:
00127:         g2d.setStroke(new BasicStroke(3));
00128:
00129:         g2d.setColor(Color.RED);
00130:         g2d.draw(force);
00131:         g2d.fill(force);
00132:         g2d.draw(upArrow);
```

```
00133:         g2d.fill(upArrow);
00134:         g2d.draw(rightArrow);
00135:         g2d.fill(rightArrow);
00136:         g2d.fill(actShape);
00137:         g2d.draw(actShape);
00138:
00139:         g2d.setColor(Color.BLUE);
00140:
00141:         g2d.draw(desShape);
00142:
00143:         g2d.dispose();
00144:     }
00145:
00146:     /*
00147:      data = Yaw referanse, Reell yaw, avstand fra referanse nord,
00148:     avstand fra referanse Ã¸st
00149:      */
00150:     public void dataUpdated(float[] data) {
00151:         this.data = data;
00152:         SwingUtilities.invokeLater(new Runnable() {
00153:             @Override
00154:             public void run() {
00155:                 repaint();
00156:             }
00157:         });
00158:
00159:     }
00160:
00161: }
```

```
00001: /*
00002:  * To change this license header, choose License Headers in Project Properties.
00003:  * To change this template file, choose Tools | Templates
00004:  * and open the template in the editor.
00005:  */
00006: package application;
00007:
00008:
00009:
00010: /**
00011:  *
00012:  * @author vegard Klasse for Ã¥ mellomlagre data sendt fra USV
00013:  */
00014: public class DataStorage {
00015:
00016:     private float[] data;
00017:     private double[] latLon;
00018:     private boolean dataUpdated = true;
00019:     private boolean egnosEnabled=false;
00020:     private boolean latLonUpdated = true;
00021:
00022:     public DataStorage() {
00023:         data = new float[23];
00024:         latLon = new double[2];
00025:     }
00026:
00027:     // Sett data mottatt fra USV
00028:     public synchronized void setArray(float[] data, double[] latLon) {
00029:         dataUpdated = true;
00030:         latLonUpdated = true;
00031:         this.data = data;
00032:         this.latLon = latLon;
00033:     }
```

```
00034:
00035:     // Hent all data bortsett fra lengde- og breddegrad
00036:     public synchronized float[] getDataArray() {
00037:         dataUpdated = false;
00038:         return data;
00039:     }
00040:
00041:     // Hent lengde- og breddegrad
00042:     public synchronized double[] getLatLonFromUSV() {
00043:         latLonUpdated = false;
00044:         return latLon;
00045:     }
00046:
00047:     // Sett egnos-status
00048:     public synchronized void setEgnosEnabled(boolean enabled) {
00049:         egnosEnabled = enabled;
00050:     }
00051:
00052:     // Hent egnos-status
00053:     public synchronized boolean getEgnosEnabled() {
00054:         return egnosEnabled;
00055:     }
00056:
00057:     // Er data oppdatert?
00058:     public synchronized boolean updated() {
00059:         return (dataUpdated && latLonUpdated);
00060:     }
00061: }
```

```
00001: package application;
00002:
00003: import java.awt.BorderLayout;
00004: import java.math.RoundingMode;
00005: import java.text.DecimalFormat;
00006: import java.util.logging.Level;
00007: import java.util.logging.Logger;
00008: import java.util.regex.Pattern;
00009: import java.util.Timer;
00010: import javax.swing.BorderFactory;
00011: import javax.swing.plaf.basic.BasicArrowButton;
00012: import org.jfree.chart.ChartPanel;
00013:
00014: /**
00015:  *
00016:  * @author Albert
00017:  *
00018:  */
00019: public class GUI extends javax.swing.JFrame implements Runnable {
00020:
00021:     private final Client r;
00022:     // Antall desimaler som vises
00023:     DecimalFormat df1 = new DecimalFormat("#.######");
00024:     DecimalFormat df2 = new DecimalFormat("#.###");
00025:     private CoordinateSystem frame;
00026:     private DataStorage storage;
00027:     private float headingRef;
00028:     private TrendPlot swayPlot;
00029:     private TrendPlot surgePlot;
00030:     private TrendPlot yawPlot;
00031:     private int guiCommand;
00032:
00033:     /**
```

```
00034:      * Creates new form GUI
00035:      */
00036:     public GUI(Client r, DataStorage storage) {
00037:         this.r = r;
00038:         this.storage = storage;
00039:         guiCommand = 0;
00040:         initComponents();
00041:         df1.setRoundingMode(RoundingMode.CEILING);
00042:         frame = (CoordinateSystem) coordinateSystemFrame;
00043:         // Trend plot for avviksverdier
00044:         swayPlot = new TrendPlot("Sway error", "Sway");
00045:         surgePlot = new TrendPlot("Surge error", "Surge");
00046:         yawPlot = new TrendPlot("Yaw error", "Yaw");
00047:         showPlot();
00048:         // Intitialiser vinduet uten reelle verdier
00049:         frame.dataUpdated(new float[]{0f, 0f, 0f, 0f});
00050:         this.setVisible(true);
00051:         headingRef = 0f;
00052:     }
00053:
00054:     // Oppdater navigasjonsdata
00055:     private void updateNavDataFields() {
00056:         float[] data = storage.getDataArray();
00057:         double[] latLon = storage.getLatLonFromUSV();
00058:         boolean egnosEnabled = storage.getEgnosEnabled();
00059:
00060:         latitudeLabel.setText("Latitude USV: "
00061:                 + df1.format((double) latLon[0]));
00062:         longitudeLabel.setText("Longitude USV: "
00063:                 + df1.format((double) latLon[1]));
00064:
00065:         latRefLabel.setText("Latitude Reference: "
00066:                 + df1.format((double) data[8]));
```

```
00067:         longRefLabel.setText("Longitude Reference: "
00068:                 + df1.format((double) data[9]));
00069:
00070:         surgeLabel.setText("Deviation North: "
00071:                 + df2.format((double) data[0]));
00072:         swayLabel.setText("Deviation East: "
00073:                 + df2.format((double) data[1]));
00074:         yawLabel.setText("Heading: "
00075:                 + df2.format((double) data[2]));
00076:
00077:         speedLabel.setText("Speed: "
00078:                 + df2.format((double) data[3]));
00079:         rotSpeedLabel.setText("Rotational Speed: "
00080:                 + df2.format((double) data[22]));
00081:         windSpeedLabel.setText("Wind Speed: "
00082:                 + df2.format((double) data[5]));
00083:         windDirLabel.setText("Wind Direction: "
00084:                 + df2.format((double) data[6]));
00085:         connectedLabel.setText("Connected to USV: "
00086:                 + r.isConnected());
00087:         dGPSlabel.setText("EGNOS enabled: " + egnosEnabled);
00088:
00089:         // Sett status tekst, basert pÃ¥ modus
00090:         switch(guiCommand) {
00091:             case 0:
00092:                 if(r.isConnected()) statusLabel.setText("Status: Idle");
00093:                 else statusLabel.setText("Status: Disconnected");
00094:                 break;
00095:             case 1:
00096:                 statusLabel.setText("Status: Dynamic Positioning");
00097:                 break;
00098:             case 2:
00099:                 statusLabel.setText("Status: Remote Control");
```

```
00100:                 break;
00101:             case 3:
00102:                 if(!r.isConnected()) statusLabel.setText("Status: Connecting");
00103:                 else statusLabel.setText("Status: Idle");
00104:                 break;
00105:             case 4:
00106:                 statusLabel.setText("Status: Disconnected");
00107:         }
00108:
00109:         swayPlot.updatePlot(data[1]);
00110:         surgePlot.updatePlot(data[0]);
00111:         yawPlot.updatePlot(headingRef - data[2]);
00112:
00113:         // Oppdater koordinatsystemet
00114:         // (data = heading ref, actual heading, surge, sway, X, Y
00115:         frame.dataUpdated(new float[]
00116:         {headingRef, data[2], data[0], data[1], data[19], data[20]});
00117:     }
00118:
00119:     // Oppdater tekstfelt
00120:     private void updateTextFields() {
00121:         float[] data = storage.getDataArray();
00122:
00123:         proportionalSurgeTextField.setText("Proportional Gain: "
00124:                 + df2.format((double) data[10]).replace(",", "."));
00125:         integralSurgeTextField.setText("Integral Gain: "
00126:                 + df2.format((double) data[11]).replace(",", "."));
00127:         derivativeSurgeTextField.setText("Derivative Gain: "
00128:                 + df2.format((double) data[12]).replace(",", "."));
00129:
00130:         proportionalSwayTextField.setText("Proportional Gain: "
00131:                 + df2.format((double) data[13]).replace(",", "."));
00132:         integralSwayTextField.setText("Integral Gain: "
```

```
00133:                 + df2.format((double) data[14]).replace(",", "."));
00134:         derivativeSwayTextField.setText("Derivative Gain: "
00135:                 + df2.format((double) data[15]).replace(",", "."));
00136:
00137:         proportionalYawTextField.setText("Proportional Gain: "
00138:                 + df2.format((double) data[16]).replace(",", "."));
00139:         integralYawTextField.setText("Integral Gain: "
00140:                 + df2.format((double) data[17]).replace(",", "."));
00141:         derivativeYawTextField.setText("Derivative Gain: "
00142:                 + df2.format((double) data[18]).replace(",", "."));
00143:
00144:         headingReferenceTextField.setText("Heading Ref (deg): "
00145:                 + df2.format((double) headingRef));
00146:     }
00147:
00148:     /**
00149:      * This method is called from within the constructor to initialize the form.
00150:      * WARNING: Do NOT modify this code. The content of this method is always
00151:      * regenerated by the Form Editor.
00152:      */
00153:     @SuppressWarnings("unchecked")
00154:     // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents
00155:     private void initComponents() {
00156:
00157:         jPanel2 = new javax.swing.JPanel();
00158:         northButton = new BasicArrowButton(BasicArrowButton.NORTH);
00159:         southButton = new BasicArrowButton(BasicArrowButton.EAST);
00160:         eastButton = new BasicArrowButton(BasicArrowButton.WEST);
00161:         eastButton1 = new BasicArrowButton(BasicArrowButton.SOUTH);
00162:         jPanel1 = new javax.swing.JPanel();
00163:         jPanel3 = new javax.swing.JPanel();
00164:         navDataPanel = new javax.swing.JPanel();
00165:         latitudeLabel = new javax.swing.JLabel();
```

```
00166:         latRefLabel = new javax.swing.JLabel();
00167:         windSpeedLabel = new javax.swing.JLabel();
00168:         longitudeLabel = new javax.swing.JLabel();
00169:         longRefLabel = new javax.swing.JLabel();
00170:         windDirLabel = new javax.swing.JLabel();
00171:         surgeLabel = new javax.swing.JLabel();
00172:         swayLabel = new javax.swing.JLabel();
00173:         yawLabel = new javax.swing.JLabel();
00174:         speedLabel = new javax.swing.JLabel();
00175:         rotSpeedLabel = new javax.swing.JLabel();
00176:         connectedLabel = new javax.swing.JLabel();
00177:         navDataLabel = new javax.swing.JLabel();
00178:         jPanel4 = new javax.swing.JPanel();
00179:         jLabel1 = new javax.swing.JLabel();
00180:         proportionalSurgeTextField = new javax.swing.JTextField();
00181:         integralSurgeTextField = new javax.swing.JTextField();
00182:         derivativeSurgeTextField = new javax.swing.JTextField();
00183:         jLabel2 = new javax.swing.JLabel();
00184:         jLabel3 = new javax.swing.JLabel();
00185:         jLabel4 = new javax.swing.JLabel();
00186:         proportionalSwayTextField = new javax.swing.JTextField();
00187:         integralSwayTextField = new javax.swing.JTextField();
00188:         derivativeSwayTextField = new javax.swing.JTextField();
00189:         proportionalYawTextField = new javax.swing.JTextField();
00190:         integralYawTextField = new javax.swing.JTextField();
00191:         derivativeYawTextField = new javax.swing.JTextField();
00192:         jPanel7 = new javax.swing.JPanel();
00193:         trendPanelSurge = new javax.swing.JPanel(new BorderLayout(1,1));
00194:         trendPanelSway = new javax.swing.JPanel(new BorderLayout(1,1));
00195:         trendPanelYaw = new javax.swing.JPanel(new BorderLayout(1,1));
00196:         coordinateSystemFrame = new CoordinateSystem(500,450);
00197:         setpointLabel2 = new javax.swing.JPanel();
00198:         refNorthButton = new BasicArrowButton(BasicArrowButton.NORTH);
```

```
00199:         refSouthButton = new BasicArrowButton(BasicArrowButton.SOUTH);
00200:         refEastButton = new BasicArrowButton(BasicArrowButton.EAST);
00201:         refWestButton = new BasicArrowButton(BasicArrowButton.WEST);
00202:         setpointLabel1 = new javax.swing.JLabel();
00203:         jLabel7 = new javax.swing.JLabel();
00204:         jPanel5 = new javax.swing.JPanel();
00205:         dGPSlabel = new javax.swing.JLabel();
00206:         statusLabel = new javax.swing.JLabel();
00207:         connectButton = new javax.swing.JButton();
00208:         remoteButton = new javax.swing.JButton();
00209:         dynPosButton = new javax.swing.JButton();
00210:         idleButton = new javax.swing.JButton();
00211:         headingReferenceTextField = new javax.swing.JTextField();
00212:         jLabel5 = new javax.swing.JLabel();
00213:
00214:         jPanel2.setBorder(javax.swing.BorderFactory.createEtchedBorder());
00215:
00216:         northButton.setText("jButton1");
00217:
00218:         southButton.setText("jButton1");
00219:
00220:         eastButton.setText("jButton1");
00221:
00222:         eastButton1.setText("jButton1");
00223:
00224:         javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);
00225:         jPanel2.setLayout(jPanel2Layout);
00226:         jPanel2Layout.setHorizontalGroup(
00227:             jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00228:             .addGroup(jPanel2Layout.createSequentialGroup()
00229:                 .addGap(34, 34, 34)
00230:                 .addComponent(eastButton, javax.swing.GroupLayout.PREFERRED_SIZE, 25, javax.swing.GroupLayout.PREFERRED_SIZE)
00231:                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
```

```
00232:                 .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00233:                     .addComponent(northButton, javax.swing.GroupLayout.PREFERRED_SIZE, 25, javax.swing.GroupLayout.PREFERRED_SIZE)
00234:                     .addComponent(eastButton1, javax.swing.GroupLayout.PREFERRED_SIZE, 25, javax.swing.GroupLayout.PREFERRED_SIZE))
00235:                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
00236:                 .addComponent(southButton, javax.swing.GroupLayout.PREFERRED_SIZE, 25, javax.swing.GroupLayout.PREFERRED_SIZE)
00237:                 .addContainerGap(38, Short.MAX_VALUE))
00238:         );
00239:         jPanel2Layout.setVerticalGroup(
00240:             jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00241:             .addGroup(jPanel2Layout.createSequentialGroup()
00242:                 .addGap(23, 23, 23)
00243:                 .addComponent(eastButton))
00244:             .addGroup(jPanel2Layout.createSequentialGroup()
00245:                 .addComponent(northButton)
00246:                 .addGap(18, 18, 18)
00247:                 .addComponent(eastButton1))
00248:             .addGroup(jPanel2Layout.createSequentialGroup()
00249:                 .addGap(21, 21, 21)
00250:                 .addComponent(southButton))
00251:         );
00252: 
00253:         setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
00254: 
00255:         jPanel3.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(0, 0, 0)));
00256: 
00257:         navDataPanel.setBorder(javax.swing.BorderFactory.createEtchedBorder());
00258: 
00259:         latitudeLabel.setText("Latitude USV:");
00260: 
00261:         latRefLabel.setText("Latitude Reference:");
00262: 
00263:         windSpeedLabel.setText("Wind Speed:");
00264: 
```

```
00265:         longitudeLabel.setText("Longitude USV:");
00266:
00267:         longRefLabel.setText("Longitude Reference:");
00268:
00269:         windDirLabel.setText("Wind Direction:");
00270:
00271:         surgeLabel.setText("Deviation North:");
00272:
00273:         swayLabel.setText("Deviation East");
00274:
00275:         yawLabel.setText("Heading:");
00276:
00277:         speedLabel.setText("Speed:");
00278:
00279:         rotSpeedLabel.setText("Rotational speed:");
00280:
00281:         connectedLabel.setText("Connected to USV:");
00282:
00283:         javax.swing.GroupLayout navDataPanelLayout = new javax.swing.GroupLayout(navDataPanel);
00284:         navDataPanel.setLayout(navDataPanelLayout);
00285:         navDataPanelLayout.setHorizontalGroup(
00286:             navDataPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00287:             .addGroup(navDataPanelLayout.createSequentialGroup()
00288:                 .addContainerGap()
00289:                 .addGroup(navDataPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
00290:                     .addComponent(latitudeLabel, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
00291:                     .addComponent(longitudeLabel, javax.swing.GroupLayout.DEFAULT_SIZE, 146, Short.MAX_VALUE)
00292:                     .addComponent(swayLabel, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
00293:                     .addComponent(surgeLabel, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
00294:                 .addGap(18, 18, 18)
00295:                 .addGroup(navDataPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
00296:                     .addComponent(latRefLabel, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
00297:                     .addComponent(longRefLabel, javax.swing.GroupLayout.DEFAULT_SIZE, 164, Short.MAX_VALUE)
```

```
00298:                     .addComponent(yawLabel, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
00299:                     .addComponent(speedLabel, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
00300:                 .addGap(18, 18, 18)
00301:                 .addGroup(navDataPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00302:                     .addComponent(rotSpeedLabel, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
00303:                     .addGroup(navDataPanelLayout.createSequentialGroup()
00304:                         .addGroup(navDataPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
00305:                             .addComponent(windSpeedLabel, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
00306:                             .addComponent(windDirLabel, javax.swing.GroupLayout.DEFAULT_SIZE, 140, Short.MAX_VALUE))
00307:                         .addGap(0, 0, Short.MAX_VALUE))
00308:                     .addComponent(connectedLabel, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
00309:                 .addGap(21, 21, 21))
00310:         );
00311:         navDataPanelLayout.setVerticalGroup(
00312:             navDataPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00313:             .addGroup(navDataPanelLayout.createSequentialGroup()
00314:                 .addContainerGap()
00315:                 .addGroup(navDataPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
00316:                     .addComponent(latitudeLabel)
00317:                     .addComponent(latRefLabel)
00318:                     .addComponent(windSpeedLabel))
00319:                 .addGap(18, 18, 18)
00320:                 .addGroup(navDataPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
00321:                     .addComponent(longitudeLabel)
00322:                     .addComponent(longRefLabel)
00323:                     .addComponent(windDirLabel))
00324:                 .addGap(18, 18, 18)
00325:                 .addGroup(navDataPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
00326:                     .addComponent(surgeLabel)
00327:                     .addComponent(yawLabel)
00328:                     .addComponent(rotSpeedLabel))
00329:                 .addGap(18, 18, 18)
00330:                 .addGroup(navDataPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
```

```
00331:                     .addComponent(swayLabel)
00332:                     .addComponent(speedLabel)
00333:                     .addComponent(connectedLabel))
00334:                 .addContainerGap(24, Short.MAX_VALUE))
00335:         );
00336:
00337:         navDataLabel.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
00338:         navDataLabel.setText("Navigational Data");
00339:
00340:         javax.swing.GroupLayout jPanel3Layout = new javax.swing.GroupLayout(jPanel3);
00341:         jPanel3.setLayout(jPanel3Layout);
00342:         jPanel3Layout.setHorizontalGroup(
00343:             jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00344:             .addComponent(navDataLabel, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
00345:             .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel3Layout.createSequentialGroup()
00346:                 .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
00347:                 .addComponent(navDataPanel, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
00348:                 .addContainerGap())
00349:         );
00350:         jPanel3Layout.setVerticalGroup(
00351:             jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00352:             .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel3Layout.createSequentialGroup()
00353:                 .addContainerGap()
00354:                 .addComponent(navDataLabel)
00355:                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 9, Short.MAX_VALUE)
00356:                 .addComponent(navDataPanel, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
00357:                 .addContainerGap())
00358:         );
00359:
00360:         jPanel4.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(0, 0, 0)));
```

```
00361:
00362:         jLabel1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
00363:         jLabel1.setText("PID Controller Gains");
00364:
00365:         proportionalSurgeTextField.setText("Proportional Gain:");
00366:         proportionalSurgeTextField.addFocusListener(new java.awt.event.FocusAdapter() {
00367:             public void focusLost(java.awt.event.FocusEvent evt) {
00368:                 proportionalSurgeTextFieldFocusLost(evt);
00369:             }
00370:         });
00371:         proportionalSurgeTextField.addActionListener(new java.awt.event.ActionListener() {
00372:             public void actionPerformed(java.awt.event.ActionEvent evt) {
00373:                 proportionalSurgeTextFieldActionPerformed(evt);
00374:             }
00375:         });
00376:
00377:         integralSurgeTextField.setText("Integral Gain:");
00378:         integralSurgeTextField.addFocusListener(new java.awt.event.FocusAdapter() {
00379:             public void focusLost(java.awt.event.FocusEvent evt) {
00380:                 integralSurgeTextFieldFocusLost(evt);
00381:             }
00382:         });
00383:         integralSurgeTextField.addActionListener(new java.awt.event.ActionListener() {
00384:             public void actionPerformed(java.awt.event.ActionEvent evt) {
00385:                 integralSurgeTextFieldActionPerformed(evt);
00386:             }
00387:         });
00388:
00389:         derivativeSurgeTextField.setText("Derivative Gain:");
00390:         derivativeSurgeTextField.addFocusListener(new java.awt.event.FocusAdapter() {
00391:             public void focusLost(java.awt.event.FocusEvent evt) {
00392:                 derivativeSurgeTextFieldFocusLost(evt);
00393:             }
```

```
00394:         });
00395:         derivativeSurgeTextField.addActionListener(new java.awt.event.ActionListener() {
00396:             public void actionPerformed(java.awt.event.ActionEvent evt) {
00397:                 derivativeSurgeTextFieldActionPerformed(evt);
00398:             }
00399:         });
00400: 
00401:         jLabel2.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
00402:         jLabel2.setText("Surge");
00403: 
00404:         jLabel3.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
00405:         jLabel3.setText("Sway");
00406: 
00407:         jLabel4.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
00408:         jLabel4.setText("Yaw");
00409: 
00410:         proportionalSwayTextField.setText("Proportional Gain:");
00411:         proportionalSwayTextField.addFocusListener(new java.awt.event.FocusAdapter() {
00412:             public void focusLost(java.awt.event.FocusEvent evt) {
00413:                 proportionalSwayTextFieldFocusLost(evt);
00414:             }
00415:         });
00416:         proportionalSwayTextField.addActionListener(new java.awt.event.ActionListener() {
00417:             public void actionPerformed(java.awt.event.ActionEvent evt) {
00418:                 proportionalSwayTextFieldActionPerformed(evt);
00419:             }
00420:         });
00421: 
00422:         integralSwayTextField.setText("Integral Gain:");
00423:         integralSwayTextField.addFocusListener(new java.awt.event.FocusAdapter() {
00424:             public void focusLost(java.awt.event.FocusEvent evt) {
00425:                 integralSwayTextFieldFocusLost(evt);
00426:             }
```

```
00427:         });
00428:         integralSwayTextField.addActionListener(new java.awt.event.ActionListener() {
00429:             public void actionPerformed(java.awt.event.ActionEvent evt) {
00430:                 integralSwayTextFieldActionPerformed(evt);
00431:             }
00432:         });
00433: 
00434:         derivativeSwayTextField.setText("Derivative Gain:");
00435:         derivativeSwayTextField.addFocusListener(new java.awt.event.FocusAdapter() {
00436:             public void focusLost(java.awt.event.FocusEvent evt) {
00437:                 derivativeSwayTextFieldFocusLost(evt);
00438:             }
00439:         });
00440:         derivativeSwayTextField.addActionListener(new java.awt.event.ActionListener() {
00441:             public void actionPerformed(java.awt.event.ActionEvent evt) {
00442:                 derivativeSwayTextFieldActionPerformed(evt);
00443:             }
00444:         });
00445: 
00446:         proportionalYawTextField.setText("Proportional Gain:");
00447:         proportionalYawTextField.addFocusListener(new java.awt.event.FocusAdapter() {
00448:             public void focusLost(java.awt.event.FocusEvent evt) {
00449:                 proportionalYawTextFieldFocusLost(evt);
00450:             }
00451:         });
00452:         proportionalYawTextField.addActionListener(new java.awt.event.ActionListener() {
00453:             public void actionPerformed(java.awt.event.ActionEvent evt) {
00454:                 proportionalYawTextFieldActionPerformed(evt);
00455:             }
00456:         });
00457: 
00458:         integralYawTextField.setText("Integral Gain:");
00459:         integralYawTextField.addFocusListener(new java.awt.event.FocusAdapter() {
```

```
00460:             public void focusLost(java.awt.event.FocusEvent evt) {
00461:                 integralYawTextFieldFocusLost(evt);
00462:             }
00463:         });
00464:         integralYawTextField.addActionListener(new java.awt.event.ActionListener() {
00465:             public void actionPerformed(java.awt.event.ActionEvent evt) {
00466:                 integralYawTextFieldActionPerformed(evt);
00467:             }
00468:         });
00469: 
00470:         derivativeYawTextField.setText("Derivative Gain:");
00471:         derivativeYawTextField.addFocusListener(new java.awt.event.FocusAdapter() {
00472:             public void focusLost(java.awt.event.FocusEvent evt) {
00473:                 derivativeYawTextFieldFocusLost(evt);
00474:             }
00475:         });
00476:         derivativeYawTextField.addActionListener(new java.awt.event.ActionListener() {
00477:             public void actionPerformed(java.awt.event.ActionEvent evt) {
00478:                 derivativeYawTextFieldActionPerformed(evt);
00479:             }
00480:         });
00481: 
00482:         javax.swing.GroupLayout jPanel4Layout = new javax.swing.GroupLayout(jPanel4);
00483:         jPanel4.setLayout(jPanel4Layout);
00484:         jPanel4Layout.setHorizontalGroup(
00485:             jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00486:             .addGroup(jPanel4Layout.createSequentialGroup()
00487:                 .addContainerGap()
00488:                 .addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00489:                     .addComponent(jLabel1, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
00490:                     .addGroup(jPanel4Layout.createSequentialGroup()
00491:                         .addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
00492:                             .addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED_SIZE, 130, javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
00493:                             .addComponent(integralSurgeTextField)
00494:                             .addComponent(derivativeSurgeTextField)
00495:                             .addComponent(proportionalSurgeTextField, javax.swing.GroupLayout.DEFAULT_SIZE, 140, Short.MAX_VALUE))
00496:                         .addGap(47, 47, 47)
00497:                         .addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
00498:                             .addComponent(derivativeSwayTextField, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT_SIZE, 138,
Short.MAX_VALUE)
00499:                             .addComponent(integralSwayTextField, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT_SIZE, 138,
Short.MAX_VALUE)
00500:                             .addComponent(proportionalSwayTextField, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT_SIZE, 138,
Short.MAX_VALUE)
00501:                             .addComponent(jLabel3, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
00502:                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
00503:                         .addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)
00504:                             .addComponent(derivativeYawTextField)
00505:                             .addComponent(integralYawTextField)
00506:                             .addComponent(proportionalYawTextField, javax.swing.GroupLayout.Alignment.LEADING, javax.swing.GroupLayout.DEFAULT_SIZE, 139,
Short.MAX_VALUE)
00507:                             .addComponent(jLabel4, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
00508:                         .addGap(17, 17, 17)))
00509:                 .addContainerGap())
00510:         );
00511:         jPanel4Layout.setVerticalGroup(
00512:             jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00513:             .addGroup(jPanel4Layout.createSequentialGroup()
00514:                 .addComponent(jLabel1)
00515:                 .addGap(33, 33, 33)
00516:                 .addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
00517:                     .addComponent(jLabel2)
00518:                     .addComponent(jLabel3)
00519:                     .addComponent(jLabel4))
00520:                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
```

```
00521:                 .addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
00522:                     .addComponent(proportionalSurgeTextField, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
00523:                     .addComponent(proportionalSwayTextField, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
00524:                     .addComponent(proportionalYawTextField, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
00525:                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
00526:                 .addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
00527:                     .addComponent(integralSurgeTextField, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
00528:                     .addComponent(integralSwayTextField, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
00529:                     .addComponent(integralYawTextField, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
00530:                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
00531:                 .addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
00532:                     .addComponent(derivativeSurgeTextField, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
00533:                     .addComponent(derivativeSwayTextField, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
00534:                     .addComponent(derivativeYawTextField, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
00535:                 .addContainerGap(22, Short.MAX_VALUE))
00536:         );
00537:
00538:         trendPanelSurge.setMaximumSize(new java.awt.Dimension(600, 120));
00539:         trendPanelSurge.setMinimumSize(new java.awt.Dimension(0, 120));
00540:
00541:         javax.swing.GroupLayout trendPanelSurgeLayout = new javax.swing.GroupLayout(trendPanelSurge);
00542:         trendPanelSurge.setLayout(trendPanelSurgeLayout);
00543:         trendPanelSurgeLayout.setHorizontalGroup(
00544:             trendPanelSurgeLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
```

```
00545:             .addGap(0, 0, Short.MAX_VALUE)
00546:         );
00547:         trendPanelSurgeLayout.setVerticalGroup(
00548:             trendPanelSurgeLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00549:             .addGap(0, 120, Short.MAX_VALUE)
00550:         );
00551:
00552:         trendPanelSway.setMaximumSize(new java.awt.Dimension(32767, 120));
00553:         trendPanelSway.setMinimumSize(new java.awt.Dimension(0, 120));
00554:
00555:         javax.swing.GroupLayout trendPanelSwayLayout = new javax.swing.GroupLayout(trendPanelSway);
00556:         trendPanelSway.setLayout(trendPanelSwayLayout);
00557:         trendPanelSwayLayout.setHorizontalGroup(
00558:             trendPanelSwayLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00559:             .addGap(0, 0, Short.MAX_VALUE)
00560:         );
00561:         trendPanelSwayLayout.setVerticalGroup(
00562:             trendPanelSwayLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00563:             .addGap(0, 120, Short.MAX_VALUE)
00564:         );
00565:
00566:         trendPanelYaw.setMaximumSize(new java.awt.Dimension(32767, 120));
00567:         trendPanelYaw.setMinimumSize(new java.awt.Dimension(0, 120));
00568:
00569:         javax.swing.GroupLayout trendPanelYawLayout = new javax.swing.GroupLayout(trendPanelYaw);
00570:         trendPanelYaw.setLayout(trendPanelYawLayout);
00571:         trendPanelYawLayout.setHorizontalGroup(
00572:             trendPanelYawLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00573:             .addGap(0, 0, Short.MAX_VALUE)
00574:         );
00575:         trendPanelYawLayout.setVerticalGroup(
00576:             trendPanelYawLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00577:             .addGap(0, 120, Short.MAX_VALUE)
```

```
00578:         );
00579:
00580:         javax.swing.GroupLayout jPanel7Layout = new javax.swing.GroupLayout(jPanel7);
00581:         jPanel7.setLayout(jPanel7Layout);
00582:         jPanel7Layout.setHorizontalGroup(
00583:             jPanel7Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00584:             .addComponent(trendPanelSurge, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
00585:             .addComponent(trendPanelSway, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
00586:             .addComponent(trendPanelYaw, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
00587:         );
00588:         jPanel7Layout.setVerticalGroup(
00589:             jPanel7Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00590:             .addGroup(jPanel7Layout.createSequentialGroup()
00591:                 .addComponent(trendPanelSurge, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
00592:                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
00593:                 .addComponent(trendPanelSway, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
00594:                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
00595:                 .addComponent(trendPanelYaw, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
00596:                 .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
00597:         );
00598:
00599:         javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
00600:         jPanel1.setLayout(jPanel1Layout);
00601:         jPanel1Layout.setHorizontalGroup(
00602:             jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00603:             .addGroup(jPanel1Layout.createSequentialGroup()
00604:                 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00605:                     .addComponent(jPanel3, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
00606:                     .addComponent(jPanel4, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT_SIZE,
```

```
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
00607:                     .addComponent(jPanel7, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
00608:                 .addContainerGap())
00609:         );
00610:         jPanel1Layout.setVerticalGroup(
00611:             jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00612:             .addGroup(jPanel1Layout.createSequentialGroup()
00613:                 .addComponent(jPanel3, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
00614:                 .addGap(18, 18, 18)
00615:                 .addComponent(jPanel4, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
00616:                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
00617:                 .addComponent(jPanel7, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
00618:         );
00619:
00620:         coordinateSystemFrame.setMaximumSize(new java.awt.Dimension(502, 502));
00621:         coordinateSystemFrame.setMinimumSize(new java.awt.Dimension(502, 502));
00622:         coordinateSystemFrame.setPreferredSize(new java.awt.Dimension(502, 502));
00623:
00624:         javax.swing.GroupLayout coordinateSystemFrameLayout = new javax.swing.GroupLayout(coordinateSystemFrame);
00625:         coordinateSystemFrame.setLayout(coordinateSystemFrameLayout);
00626:         coordinateSystemFrameLayout.setHorizontalGroup(
00627:             coordinateSystemFrameLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00628:             .addGap(0, 0, Short.MAX_VALUE)
00629:         );
00630:         coordinateSystemFrameLayout.setVerticalGroup(
00631:             coordinateSystemFrameLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00632:             .addGap(0, 502, Short.MAX_VALUE)
00633:         );
00634:
00635:         setpointLabel2.setBorder(javax.swing.BorderFactory.createEtchedBorder());
00636:
00637:         refNorthButton.setText("jButton1");
```

```
00638:         refNorthButton.addActionListener(new java.awt.event.ActionListener() {
00639:             public void actionPerformed(java.awt.event.ActionEvent evt) {
00640:                 refNorthButtonActionPerformed(evt);
00641:             }
00642:         });
00643: 
00644:         refSouthButton.setText("jButton2");
00645:         refSouthButton.addActionListener(new java.awt.event.ActionListener() {
00646:             public void actionPerformed(java.awt.event.ActionEvent evt) {
00647:                 refSouthButtonActionPerformed(evt);
00648:             }
00649:         });
00650: 
00651:         refEastButton.setText("jButton3");
00652:         refEastButton.addActionListener(new java.awt.event.ActionListener() {
00653:             public void actionPerformed(java.awt.event.ActionEvent evt) {
00654:                 refEastButtonActionPerformed(evt);
00655:             }
00656:         });
00657: 
00658:         refWestButton.setText("jButton4");
00659:         refWestButton.addActionListener(new java.awt.event.ActionListener() {
00660:             public void actionPerformed(java.awt.event.ActionEvent evt) {
00661:                 refWestButtonActionPerformed(evt);
00662:             }
00663:         });
00664: 
00665:         setpointLabel1.setText("Use arrows to move reference setpoint");
00666: 
00667:         jLabel7.setText("in 0.5 meter increments");
00668: 
00669:         javax.swing.GroupLayout setpointLabel2Layout = new javax.swing.GroupLayout(setpointLabel2);
00670:         setpointLabel2.setLayout(setpointLabel2Layout);
```

```
00671:         setpointLabel2Layout.setHorizontalGroup(
00672:             setpointLabel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00673:             .addGroup(setpointLabel2Layout.createSequentialGroup()
00674:                 .addContainerGap()
00675:                 .addGroup(setpointLabel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00676:                     .addComponent(setpointLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 235, javax.swing.GroupLayout.PREFERRED_SIZE)
00677:                     .addComponent(jLabel7))
00678:                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
00679:                 .addComponent(refWestButton)
00680:                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
00681:                 .addGroup(setpointLabel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00682:                     .addComponent(refSouthButton)
00683:                     .addComponent(refNorthButton))
00684:                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
00685:                 .addComponent(refEastButton))
00686:         );
00687:         setpointLabel2Layout.setVerticalGroup(
00688:             setpointLabel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00689:             .addGroup(setpointLabel2Layout.createSequentialGroup()
00690:                 .addGap(6, 6, 6)
00691:                 .addComponent(refNorthButton)
00692:                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
00693:                 .addComponent(refSouthButton)
00694:                 .addGap(0, 11, Short.MAX_VALUE))
00695:             .addGroup(setpointLabel2Layout.createSequentialGroup()
00696:                 .addGroup(setpointLabel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00697:                     .addGroup(setpointLabel2Layout.createSequentialGroup()
00698:                         .addGap(23, 23, 23)
00699:                         .addGroup(setpointLabel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
00700:                             .addComponent(refEastButton)
00701:                             .addComponent(refWestButton)))
00702:                     .addGroup(setpointLabel2Layout.createSequentialGroup()
00703:                         .addContainerGap()
```

```
00704:                         .addComponent(setpointLabel1)
00705:                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
00706:                         .addComponent(jLabel7)))
00707:                 .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
00708:         );
00709:
00710:         jPanel5.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(0, 0, 0)));
00711:
00712:         dGPSlabel.setText("EGNOS enabled:");
00713:
00714:         statusLabel.setText("Status:");
00715:
00716:         connectButton.setText("Connect");
00717:         connectButton.addActionListener(new java.awt.event.ActionListener() {
00718:             public void actionPerformed(java.awt.event.ActionEvent evt) {
00719:                 connectButtonActionPerformed(evt);
00720:             }
00721:         });
00722:
00723:         remoteButton.setText("Remote Control");
00724:         remoteButton.addActionListener(new java.awt.event.ActionListener() {
00725:             public void actionPerformed(java.awt.event.ActionEvent evt) {
00726:                 remoteButtonActionPerformed(evt);
00727:             }
00728:         });
00729:
00730:         dynPosButton.setText("Dynamic Positioning");
00731:         dynPosButton.addActionListener(new java.awt.event.ActionListener() {
00732:             public void actionPerformed(java.awt.event.ActionEvent evt) {
00733:                 dynPosButtonActionPerformed(evt);
00734:             }
00735:         });
00736:
```

```
00737:         idleButton.setText("Idle");
00738:         idleButton.addActionListener(new java.awt.event.ActionListener() {
00739:             public void actionPerformed(java.awt.event.ActionEvent evt) {
00740:                 idleButtonActionPerformed(evt);
00741:             }
00742:         });
00743: 
00744:         headingReferenceTextField.setText("Heading Ref (deg):");
00745:         headingReferenceTextField.addFocusListener(new java.awt.event.FocusAdapter() {
00746:             public void focusLost(java.awt.event.FocusEvent evt) {
00747:                 headingReferenceTextFieldFocusLost(evt);
00748:             }
00749:         });
00750:         headingReferenceTextField.addActionListener(new java.awt.event.ActionListener() {
00751:             public void actionPerformed(java.awt.event.ActionEvent evt) {
00752:                 headingReferenceTextFieldActionPerformed(evt);
00753:             }
00754:         });
00755: 
00756:         jLabel5.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
00757:         jLabel5.setText("Set heading before DP");
00758: 
00759:         javax.swing.GroupLayout jPanel5Layout = new javax.swing.GroupLayout(jPanel5);
00760:         jPanel5.setLayout(jPanel5Layout);
00761:         jPanel5Layout.setHorizontalGroup(
00762:             jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00763:             .addGroup(jPanel5Layout.createSequentialGroup()
00764:                 .addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00765:                     .addGroup(jPanel5Layout.createSequentialGroup()
00766:                         .addGap(214, 214, 214)
00767:                         .addComponent(connectButton)
00768:                         .addGap(0, 0, Short.MAX_VALUE))
00769:                     .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel5Layout.createSequentialGroup()
```

```
00770:                         .addContainerGap()
00771:                         .addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
00772:                             .addGroup(jPanel5Layout.createSequentialGroup()
00773:                                 .addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00774:                                     .addComponent(dGPSlabel, javax.swing.GroupLayout.PREFERRED_SIZE, 150, javax.swing.GroupLayout.PREFERRED_SIZE)
00775:                                     .addComponent(statusLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 150, javax.swing.GroupLayout.PREFERRED_SIZE))
00776:                                 .addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00777:                                     .addGroup(jPanel5Layout.createSequentialGroup()
00778:                                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
00779:                                         .addComponent(jLabel5)
00780:                                         .addGap(12, 12, 12))
00781:                                     .addGroup(jPanel5Layout.createSequentialGroup()
00782:                                         .addGap(27, 27, 27)
00783:                                         .addComponent(headingReferenceTextField))))
00784:                             .addGroup(jPanel5Layout.createSequentialGroup()
00785:                                 .addComponent(idleButton, javax.swing.GroupLayout.PREFERRED_SIZE, 129, javax.swing.GroupLayout.PREFERRED_SIZE)
00786:                                 .addGap(48, 48, 48)
00787:                                 .addComponent(dynPosButton, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)))
00788:                         .addGap(44, 44, 44)
00789:                         .addComponent(remoteButton, javax.swing.GroupLayout.PREFERRED_SIZE, 129, javax.swing.GroupLayout.PREFERRED_SIZE)))
00790:                 .addContainerGap())
00791:         );
00792:         jPanel5Layout.setVerticalGroup(
00793:             jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00794:             .addGroup(jPanel5Layout.createSequentialGroup()
00795:                 .addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00796:                     .addGroup(jPanel5Layout.createSequentialGroup()
00797:                         .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
00798:                         .addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
00799:                             .addComponent(statusLabel)
00800:                             .addComponent(jLabel5)))
00801:                     .addGroup(jPanel5Layout.createSequentialGroup()
```

```
00802:                          .addGap(20, 20, 20)
00803:                          .addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
00804:                              .addComponent(headingReferenceTextField, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
00805:                              .addComponent(dGPSlabel))
00806:                          .addGap(0, 0, Short.MAX_VALUE)))
00807:                  .addGap(18, 18, 18)
00808:                  .addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
00809:                      .addComponent(remoteButton)
00810:                      .addComponent(dynPosButton)
00811:                      .addComponent(idleButton))
00812:                  .addGap(28, 28, 28)
00813:                  .addComponent(connectButton))
00814:          );
00815:
00816:          javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
00817:          getContentPane().setLayout(layout);
00818:          layout.setHorizontalGroup(
00819:              layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00820:              .addGroup(layout.createSequentialGroup()
00821:                  .addContainerGap()
00822:                  .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
00823:                  .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
00824:                  .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
00825:                      .addComponent(setpointLabel2, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
00826:                      .addComponent(jPanel5, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
00827:                      .addComponent(coordinateSystemFrame, javax.swing.GroupLayout.DEFAULT_SIZE, 504, Short.MAX_VALUE))
00828:                  .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
00829:          );
00830:          layout.setVerticalGroup(
00831:              layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00832:              .addGroup(layout.createSequentialGroup()
00833:                  .addContainerGap()
```

```
00834:                 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00835:                     .addGroup(layout.createSequentialGroup()
00836:                         .addComponent(setpointLabel2, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
00837:                         .addGap(18, 18, 18)
00838:                         .addComponent(coordinateSystemFrame, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
00839:                         .addGap(26, 26, 26)
00840:                         .addComponent(jPanel5, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
00841:                         .addGap(21, 21, 21))
00842:                     .addGroup(layout.createSequentialGroup()
00843:                         .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
00844:                         .addContainerGap())))
00845:         );
00846:
00847:         pack();
00848:     }// </editor-fold>
00849: /*
00850:     ActionListenere for knappene
00851:     Aksjoner avhenger av hvilken knapp som trykkes
00852:     */
00853:     private void dynPosButtonActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_dynPosButtonActionPerformed
00854:
00855:         r.setGUIcommand(1);
00856:         guiCommand = 1;
00857:     }//GEN-LAST:event_dynPosButtonActionPerformed
00858:
00859:     private void idleButtonActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_idleButtonActionPerformed
00860:         r.setGUIcommand(0);
00861:         guiCommand = 0;
00862:     }//GEN-LAST:event_idleButtonActionPerformed
00863:
00864:     private void remoteButtonActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_remoteButtonActionPerformed
```

```
00865:         r.setGUIcommand(2);
00866:         guiCommand = 2;
00867:     }//GEN-LAST:event_remoteButtonActionPerformed
00868: 
00869:     private void connectButtonActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_connectButtonActionPerformed
00870:         if (!r.isConnected()) {
00871:             r.setGUIcommand(3);
00872:             guiCommand = 3;
00873:             connectButton.setText("Disconnect");
00874:         } else {
00875:             try {
00876:                 r.setGUIcommand(4);
00877:                 guiCommand = 4;
00878:                 connectButton.setText("Connect");
00879:                 Thread.sleep(1000);
00880:                 // r.setGUIcommand(0);
00881:             } catch (InterruptedException ex) {
00882:                 Logger.getLogger(GUI.class.getName())
00883:                         .log(Level.SEVERE, null, ex);
00884:             }
00885:         }
00886:     }//GEN-LAST:event_connectButtonActionPerformed
00887: 
00888:     private void proportionalSurgeTextFieldActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_proportionalSurgeTextFieldActionPerformed
00889:         r.gainChanged(1, getValueFromTextInput(proportionalSurgeTextField.getText()));
00890:         this.requestFocus();
00891:     }//GEN-LAST:event_proportionalSurgeTextFieldActionPerformed
00892: 
00893:     private void integralSurgeTextFieldActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_integralSurgeTextFieldActionPerformed
00894:         r.gainChanged(2, getValueFromTextInput(integralSurgeTextField.getText()));
00895:         this.requestFocus();
00896:     }//GEN-LAST:event_integralSurgeTextFieldActionPerformed
00897: 
```

```
00898:     private void derivativeSurgeTextFieldActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_derivativeSurgeTextFieldActionPerformed
00899:         r.gainChanged(3, getValueFromTextInput(derivativeSurgeTextField.getText()));
00900:         this.requestFocus();
00901:     }//GEN-LAST:event_derivativeSurgeTextFieldActionPerformed
00902: 
00903:     private void proportionalSwayTextFieldActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_proportionalSwayTextFieldActionPerformed
00904:         r.gainChanged(4, getValueFromTextInput(proportionalSwayTextField.getText()));
00905:         this.requestFocus();
00906:     }//GEN-LAST:event_proportionalSwayTextFieldActionPerformed
00907: 
00908:     private void integralSwayTextFieldActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_integralSwayTextFieldActionPerformed
00909:         r.gainChanged(5, getValueFromTextInput(integralSwayTextField.getText()));
00910:         this.requestFocus();
00911:     }//GEN-LAST:event_integralSwayTextFieldActionPerformed
00912: 
00913:     private void derivativeSwayTextFieldActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_derivativeSwayTextFieldActionPerformed
00914:         r.gainChanged(6, getValueFromTextInput(derivativeSwayTextField.getText()));
00915:         this.requestFocus();
00916:     }//GEN-LAST:event_derivativeSwayTextFieldActionPerformed
00917: 
00918:     private void proportionalYawTextFieldActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_proportionalYawTextFieldActionPerformed
00919:         r.gainChanged(7, getValueFromTextInput(proportionalYawTextField.getText()));
00920:         this.requestFocus();
00921:     }//GEN-LAST:event_proportionalYawTextFieldActionPerformed
00922: 
00923:     private void integralYawTextFieldActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_integralYawTextFieldActionPerformed
00924:         r.gainChanged(8, getValueFromTextInput(integralYawTextField.getText()));
00925:         this.requestFocus();
00926:     }//GEN-LAST:event_integralYawTextFieldActionPerformed
00927: 
00928:     private void derivativeYawTextFieldActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_derivativeYawTextFieldActionPerformed
00929:         r.gainChanged(9, getValueFromTextInput(derivativeYawTextField.getText()));
00930:         this.requestFocus();
```

```
00931:     }//GEN-LAST:event_derivativeYawTextFieldActionPerformed
00932:
00933:     private void proportionalSurgeTextFieldFocusLost(java.awt.event.FocusEvent evt) {//GEN-FIRST:event_proportionalSurgeTextFieldFocusLost
00934:
00935:     }//GEN-LAST:event_proportionalSurgeTextFieldFocusLost
00936:
00937:     private void integralSurgeTextFieldFocusLost(java.awt.event.FocusEvent evt) {//GEN-FIRST:event_integralSurgeTextFieldFocusLost
00938:
00939:     }//GEN-LAST:event_integralSurgeTextFieldFocusLost
00940:
00941:     private void derivativeSurgeTextFieldFocusLost(java.awt.event.FocusEvent evt) {//GEN-FIRST:event_derivativeSurgeTextFieldFocusLost
00942:
00943:     }//GEN-LAST:event_derivativeSurgeTextFieldFocusLost
00944:
00945:     private void proportionalSwayTextFieldFocusLost(java.awt.event.FocusEvent evt) {//GEN-FIRST:event_proportionalSwayTextFieldFocusLost
00946:
00947:     }//GEN-LAST:event_proportionalSwayTextFieldFocusLost
00948:
00949:     private void integralSwayTextFieldFocusLost(java.awt.event.FocusEvent evt) {//GEN-FIRST:event_integralSwayTextFieldFocusLost
00950:
00951:     }//GEN-LAST:event_integralSwayTextFieldFocusLost
00952:
00953:     private void derivativeSwayTextFieldFocusLost(java.awt.event.FocusEvent evt) {//GEN-FIRST:event_derivativeSwayTextFieldFocusLost
00954:
00955:     }//GEN-LAST:event_derivativeSwayTextFieldFocusLost
00956:
00957:     private void proportionalYawTextFieldFocusLost(java.awt.event.FocusEvent evt) {//GEN-FIRST:event_proportionalYawTextFieldFocusLost
00958:
00959:     }//GEN-LAST:event_proportionalYawTextFieldFocusLost
00960:
00961:     private void integralYawTextFieldFocusLost(java.awt.event.FocusEvent evt) {//GEN-FIRST:event_integralYawTextFieldFocusLost
00962:
00963:     }//GEN-LAST:event_integralYawTextFieldFocusLost
```

```
00964:
00965:     private void derivativeYawTextFieldFocusLost(java.awt.event.FocusEvent evt) {//GEN-FIRST:event_derivativeYawTextFieldFocusLost
00966:
00967:     }//GEN-LAST:event_derivativeYawTextFieldFocusLost
00968:
00969:     private void headingReferenceTextFieldActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_headingReferenceTextFieldActionPerformed
00970:         headingRef = getValueFromTextInput(headingReferenceTextField.getText());
00971:         r.setHeadingReference(headingRef);
00972:         updateTextFields();
00973:     }//GEN-LAST:event_headingReferenceTextFieldActionPerformed
00974:
00975:     private void headingReferenceTextFieldFocusLost(java.awt.event.FocusEvent evt) {//GEN-FIRST:event_headingReferenceTextFieldFocusLost
00976:         updateTextFields();
00977:     }//GEN-LAST:event_headingReferenceTextFieldFocusLost
00978:
00979:     private void refNorthButtonActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_refNorthButtonActionPerformed
00980:         r.incrementNorth(1);
00981:         updateNavDataFields();
00982:     }//GEN-LAST:event_refNorthButtonActionPerformed
00983:
00984:     private void refEastButtonActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_refEastButtonActionPerformed
00985:         r.incrementEast(1);
00986:         updateNavDataFields();
00987:     }//GEN-LAST:event_refEastButtonActionPerformed
00988:
00989:     private void refSouthButtonActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_refSouthButtonActionPerformed
00990:         r.incrementNorth(-1);
00991:         updateNavDataFields();
00992:     }//GEN-LAST:event_refSouthButtonActionPerformed
00993:
00994:     private void refWestButtonActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_refWestButtonActionPerformed
00995:         r.incrementEast(-1);
00996:         updateNavDataFields();
```

```
00997:     }//GEN-LAST:event_refWestButtonActionPerformed
00998: /*
00999:     Metode for Ã¥ konvertere string til desimaltall hentet fra java api
01000:     */
01001:     private float getValueFromTextInput(String text) {
01002:         final String Digits = "(\\p{Digit}+)";
01003:         final String HexDigits = "(\\p{XDigit}+)";
01004:         // an exponent is 'e' or 'E' followed by an optionally
01005:         // signed decimal integer.
01006:         final String Exp = "[eE][+-]?" + Digits;
01007:         final String fpRegex
01008:                 = ("[\\x00-\\x20]*"
01009:                 + // Optional leading "whitespace"
01010:                 "[+-]?("
01011:                 + // Optional sign character
01012:                 "NaN|"
01013:                 + // "NaN" string
01014:                 "Infinity|"
01015:                 + // "Infinity" string
01016:                 // A decimal floating-point string representing a finite positive
01017:                 // number without a leading sign has at most five basic pieces:
01018:                 // Digits . Digits ExponentPart FloatTypeSuffix
01019:                 //
01020:                 // Since this method allows integer-only strings as input
01021:                 // in addition to strings of floating-point literals, the
01022:                 // two sub-patterns below are simplifications of the grammar
01023:                 // productions from section 3.10.2 of
01024:                 // The Javaâ„¢ Language Specification.
01025:                 // Digits ._opt Digits_opt ExponentPart_opt FloatTypeSuffix_opt
01026:                 "(((" + Digits + "(\\.)?(" + Digits + "?)(" + Exp + ")?)|"
01027:                 + // . Digits ExponentPart_opt FloatTypeSuffix_opt
01028:                 "(\\.(" + Digits + ")(" + Exp + ")?)|"
01029:                 + // Hexadecimal strings
```

```
01030:                 "(("
01031:                 + // 0[xX] HexDigits ._opt BinaryExponent FloatTypeSuffix_opt
01032:                 "(0[xX]" + HexDigits + "(\\.)?)|"
01033:                 + // 0[xX] HexDigits_opt . HexDigits BinaryExponent FloatTypeSuffix_opt
01034:                 "(0[xX]" + HexDigits + "?(\\.)" + HexDigits + ")"
01035:                 + ")[pP][+-]?" + Digits + "))"
01036:                 + "[fFdD]?))"
01037:                 + "[\\x00-\\x20]*");// Optional trailing "whitespace"
01038: 
01039:         if (Pattern.matches(fpRegex, text)) {
01040:             return Float.valueOf(text); // Will not throw NumberFormatException
01041:         } else {
01042:             String[] s = text.split(":");
01043:             return Float.valueOf(s[s.length - 1].trim());
01044:         }
01045: 
01046:     }
01047: 
01048:     /**
01049:      * @param args the command line arguments
01050:      */
01051:     public static void main(String args[]) {
01052:         /* Set the Nimbus look and feel */
01053:         //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
01054:         /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
01055:          * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
01056:          */
01057:         try {
01058:             for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
01059:                 if ("Nimbus".equals(info.getName())) {
01060:                     javax.swing.UIManager.setLookAndFeel(info.getClassName());
01061:                     break;
01062:                 }
```

```
01063:             }
01064:         } catch (ClassNotFoundException ex) {
01065:             java.util.logging.Logger.getLogger(GUI.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
01066:         } catch (InstantiationException ex) {
01067:             java.util.logging.Logger.getLogger(GUI.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
01068:         } catch (IllegalAccessException ex) {
01069:             java.util.logging.Logger.getLogger(GUI.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
01070:         } catch (javax.swing.UnsupportedLookAndFeelException ex) {
01071:             java.util.logging.Logger.getLogger(GUI.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
01072:         }
01073:         // initialiser objekter
01074:         DataStorage st = new DataStorage();
01075:         //</editnavDataLabel>
01076:         JoystickReader jReader = new JoystickReader();
01077:         Timer t = new Timer();
01078:         // Start joystickleseren
01079:         t.scheduleAtFixedRate(jReader, 1000, 100);
01080:         // start klienten
01081:         Client r = new Client(jReader, st);
01082:         r.start();
01083:         GUI gui = new GUI(r, st);
01084:         // Start gui
01085:         Thread t1 = new Thread(gui);
01086:         t1.start();
01087: 
01088:     }
01089: 
01090:     // Variables declaration - do not modify//GEN-BEGIN:variables
01091:     private javax.swing.JButton connectButton;
01092:     private javax.swing.JLabel connectedLabel;
01093:     private javax.swing.JPanel coordinateSystemFrame;
01094:     private javax.swing.JLabel dGPSlabel;
01095:     private javax.swing.JTextField derivativeSurgeTextField;
```

```
01096:     private javax.swing.JTextField derivativeSwayTextField;
01097:     private javax.swing.JTextField derivativeYawTextField;
01098:     private javax.swing.JButton dynPosButton;
01099:     private javax.swing.JButton eastButton;
01100:     private javax.swing.JButton eastButton1;
01101:     private javax.swing.JTextField headingReferenceTextField;
01102:     private javax.swing.JButton idleButton;
01103:     private javax.swing.JTextField integralSurgeTextField;
01104:     private javax.swing.JTextField integralSwayTextField;
01105:     private javax.swing.JTextField integralYawTextField;
01106:     private javax.swing.JLabel jLabel1;
01107:     private javax.swing.JLabel jLabel2;
01108:     private javax.swing.JLabel jLabel3;
01109:     private javax.swing.JLabel jLabel4;
01110:     private javax.swing.JLabel jLabel5;
01111:     private javax.swing.JLabel jLabel7;
01112:     private javax.swing.JPanel jPanel1;
01113:     private javax.swing.JPanel jPanel2;
01114:     private javax.swing.JPanel jPanel3;
01115:     private javax.swing.JPanel jPanel4;
01116:     private javax.swing.JPanel jPanel5;
01117:     private javax.swing.JPanel jPanel7;
01118:     private javax.swing.JLabel latRefLabel;
01119:     private javax.swing.JLabel latitudeLabel;
01120:     private javax.swing.JLabel longRefLabel;
01121:     private javax.swing.JLabel longitudeLabel;
01122:     private javax.swing.JLabel navDataLabel;
01123:     private javax.swing.JPanel navDataPanel;
01124:     private javax.swing.JButton northButton;
01125:     private javax.swing.JTextField proportionalSurgeTextField;
01126:     private javax.swing.JTextField proportionalSwayTextField;
01127:     private javax.swing.JTextField proportionalYawTextField;
01128:     private javax.swing.JButton refEastButton;
```

```
01129:     private javax.swing.JButton refNorthButton;
01130:     private javax.swing.JButton refSouthButton;
01131:     private javax.swing.JButton refWestButton;
01132:     private javax.swing.JButton remoteButton;
01133:     private javax.swing.JLabel rotSpeedLabel;
01134:     private javax.swing.JLabel setpointLabel1;
01135:     private javax.swing.JPanel setpointLabel2;
01136:     private javax.swing.JButton southButton;
01137:     private javax.swing.JLabel speedLabel;
01138:     private javax.swing.JLabel statusLabel;
01139:     private javax.swing.JLabel surgeLabel;
01140:     private javax.swing.JLabel swayLabel;
01141:     private javax.swing.JPanel trendPanelSurge;
01142:     private javax.swing.JPanel trendPanelSway;
01143:     private javax.swing.JPanel trendPanelYaw;
01144:     private javax.swing.JLabel windDirLabel;
01145:     private javax.swing.JLabel windSpeedLabel;
01146:     private javax.swing.JLabel yawLabel;
01147:     // End of variables declaration//GEN-END:variables
01148: 
01149:     @Override
01150:     public void run() {
01151:         while (true) {
01152:             updateNavDataFields();
01153:             // Hvis GUI har fokus
01154:             if (this.isFocusOwner()) {
01155:                 // Hvis ny data
01156:                 if (storage.updated()) {
01157:                     // oppdater tekstfelt
01158:                     updateTextFields();
01159:                 }
01160:             }
01161:         }
```

```
01162:     }
01163: /*
01164:     Sett opp og vis trendplot i gui
01165:     */
01166:     private void showPlot() {
01167:         ChartPanel p1 = swayPlot.getChartPanel();
01168:         ChartPanel p2 = surgePlot.getChartPanel();
01169:         ChartPanel p3 = yawPlot.getChartPanel();
01170:
01171:         trendPanelSway.setBorder(javax.swing.BorderFactory.
01172:                 createLineBorder(new java.awt.Color(0, 0, 0)));
01173:         trendPanelSurge.setBorder(javax.swing.BorderFactory.
01174:                 createLineBorder(new java.awt.Color(0, 0, 0)));
01175:         trendPanelYaw.setBorder(javax.swing.BorderFactory.
01176:                 createLineBorder(new java.awt.Color(0, 0, 0)));
01177:
01178:         p1.setDomainZoomable(true);
01179:         p2.setDomainZoomable(true);
01180:         p3.setDomainZoomable(true);
01181:         p1.setSize(trendPanelSurge.getWidth()-1, trendPanelSurge.getHeight());
01182:         p2.setSize(trendPanelYaw.getWidth()-1, trendPanelYaw.getHeight());
01183:         p3.setSize(trendPanelSway.getWidth()-1, trendPanelSway.getHeight());
01184:
01185:         trendPanelYaw.add(p3, BorderLayout.CENTER);
01186:         trendPanelSurge.add(p2, BorderLayout.CENTER);
01187:         trendPanelSway.add(p1, BorderLayout.CENTER);
01188:
01189:
01190:
01191:     }
01192: }
```

```
00001: /*
00002:  * To change this license header, choose License Headers in Project Properties.
00003:  * To change this template file, choose Tools | Templates
00004:  * and open the template in the editor.
00005:  */
00006: package application;
00007:
00008: import java.awt.geom.Path2D;
00009:
00010: /**
00011:  *
00012:  * @author vegard Tegner en enkel todimensjonal
00013:  * strekfigur
00014:  */
00015: public class GUIShape extends Path2D.Float {
00016:
00017:     /**
00018:      *
00019:      */
00020:     public enum ShapeType {
00021:
00022:         RIGHT_ARROW, UP_ARROW, BOAT;
00023:     }
00024:
00025:     public GUIShape(ShapeType type) {
00026:         switch (type) {
00027:             case BOAT:
00028:                 moveTo(10, 0);
00029:                 lineTo(20, 15);
00030:                 lineTo(20, 60);
00031:                 lineTo(0, 60);
00032:                 lineTo(0, 15);
00033:                 lineTo(10, 0);
```

```
00034:                 break;
00035:             case RIGHT_ARROW:
00036:                 moveTo(498,250);
00037:                 lineTo(488,255);
00038:                 lineTo(488,245);
00039:                 lineTo(498,250);
00040:             case UP_ARROW:
00041:                 moveTo(250,4);
00042:                 lineTo(255,14);
00043:                 lineTo(245,14);
00044:                 lineTo(250,4);
00045:         }
00046:
00047:     }
00048:
00049: }
```

```
00001: /*
00002:  * To change this license header, choose License Headers in Project Properties.
00003:  * To change this template file, choose Tools | Templates
00004:  * and open the template in the editor.
00005:  */
00006: package application;
00007:
00008: import java.util.TimerTask;
00009: import net.java.games.input.Component;
00010: import net.java.games.input.Component.Identifier;
00011: import net.java.games.input.Controller;
00012: import net.java.games.input.ControllerEnvironment;
00013: //import net.wimpi.modbus.msg.ReadMultipleRegistersResponse;
00014:
00015: /**
00016:  *
00017:  * @author vegard
00018:  * Klasse for Ã¥ lese data fra en joystick.
00019:  * Benytter Jinput for Ã¥ finne og lese joystick
00020:  */
00021: public class JoystickReader extends TimerTask {
00022:
00023:     private final double[] axisValues;
00024:     private Controller controller;
00025:
00026:     public JoystickReader() {
00027:         axisValues = new double[3];
00028:
00029:         searchForController();
00030:         if (controller == null) {
00031:             //System.exit(1);
00032:         }
00033:
```

```
00034:     }
00035:
00036:     // Sjekk om joystick er tilkoblet
00037:     private void searchForController() {
00038:         Controller[] controllers = ControllerEnvironment.
00039:                 getDefaultEnvironment().getControllers();
00040:         for (Controller cont : controllers) {
00041:             if (cont.getType() == Controller.Type.STICK) {
00042:                 controller = cont;
00043:                 break;
00044:             }
00045:         }
00046:     }
00047:
00048:     @Override
00049:     public void run() {
00050:
00051:         // Poll data fra controlleren.
00052:         controller.poll();
00053:         // Returnerer et array av Component-objekter, som inneholder
00054:         // data fra joystick
00055:         Component[] components = controller.getComponents();
00056:         // GÃ¥ gjennom hver komponent, for Ã¥ hente ut Ã¸nsket data
00057:         for (int i = 0; i < components.length; i++) {
00058:             Component comp = components[i];
00059:             Identifier componentIdentifier = comp.getIdentifier();
00060:
00061:             // Leser kun analogverdier.
00062:             // Hvis ikke analog, gÃ¥ til neste komponent
00063:             if (comp.isAnalog()) {
00064:                 float axisValue = comp.getPollData();
00065:
00066:                 // X akse
```

```
00067:                 if (componentIdentifier == Component.Identifier.Axis.X) {
00068:                     setXaxisValue(axisValue);
00069:                     continue; // gÃ¥ til neste komponent
00070:                 }
00071:                 // Y akse
00072:                 if (componentIdentifier == Component.Identifier.Axis.Y) {
00073:                     setYaxisValue(axisValue);
00074:                     continue; // GÃ¥ til neste komponent
00075:                 }
00076:                 // Vridning
00077:                 if (componentIdentifier == Component.Identifier.Axis.RZ) {
00078:                     setZaxisValue(axisValue);
00079:                 }
00080:
00081: //                printValues();
00082:             }
00083:         }
00084:     }
00085:
00086:     /**
00087:      *
00088:      * Mapper akseverdiene fra [-1 , 1] til antatt maksimalverdier i respektive
00089:      * retninger
00090:      */
00091:     private synchronized void setXaxisValue(double axisValue) {
00092:         if (axisValue < -0.05f) {
00093:             axisValues[1] = axisValue * 58;
00094:         } else if (axisValue > 0.05f) {
00095:             axisValues[1] = axisValue * 68;
00096:         } else {
00097:             axisValues[1] = 0.0f;
00098:         }
00099:     }
```

```
00100:
00101:      /*
00102:      *
00103:      * Mapper akseverdiene fra [-1 , 1] til antatt maksimalverdier i respektive
00104:      * retninger
00105:      */
00106:     private synchronized void setYaxisValue(double axisValue) {
00107:         if (axisValue < -0.05f) {
00108:             axisValues[0] = -axisValue * 68;
00109:         } else if (axisValue > 0.05f) {
00110:             axisValues[0] = -axisValue * 58;
00111:         }
00112:         else axisValues[0] = 0.0f;
00113:     }
00114:
00115:     // Returnerer et array med verdier fra hver akse
00116:     public synchronized double[] getAxisValues() {
00117:         return axisValues;
00118:     }
00119:
00120:     // Mapper akseverdier fra z-aksen til antatt maksimale verdier for moment
00121:     private synchronized void setZaxisValue(double axisValue) {
00122:         float len1 = 0.8f;
00123:         float len2 = 0.87f;
00124:         float lenB = 0.2f;
00125:         if (axisValue < -0.05f) {
00126:             axisValues[2] = (-34.0f * len1 - 29.0f * len2 - 2 * 34.0f
00127:                     - (34.0f + 29.0f) * lenB) * (-axisValue);
00128:         } else if (axisValue > 0.05f) {
00129:             axisValues[2] = (34.0f * len1 + 29.0f * len2 + 2 * 34.0f
00130:                     + (34.0f + 29.0f) * lenB) * axisValue;
00131:         }
00132:         else axisValues[2] = 0;
```

```
00133:     }
00134:
00135:     private void printValues() {
00136:         double[] values = getAxisValues();
00137:         System.out.println("X axis: " + values[0] + " Y axis: " + values[1] + " Yaw: " + values[2]);
00138:     }
00139: }
```

```
00001: /*
00002:  * To change this license header, choose License Headers in Project Properties.
00003:  * To change this template file, choose Tools | Templates
00004:  * and open the template in the editor.
00005:  */
00006: package application;
00007:
00008: import org.jfree.chart.ChartFactory;
00009: import org.jfree.chart.ChartPanel;
00010: import org.jfree.chart.JFreeChart;
00011: import org.jfree.chart.axis.ValueAxis;
00012: import org.jfree.chart.plot.XYPlot;
00013: import org.jfree.data.time.Millisecond;
00014: import org.jfree.data.time.TimeSeries;
00015: import org.jfree.data.time.TimeSeriesCollection;
00016:
00017: /**
00018:  *
00019:  * @author vegard
00020:  * Klasse som lager et trendplott basert pÃ¥ sendte verdier
00021:  */
00022: public class TrendPlot {
00023:
00024:     private ChartPanel label;
00025:     private TimeSeries timeSeries;
00026:     private String plotName;
00027:
00028:
00029:     /**
00030:      * KonstruktÃ¸ren kaller initialize()
00031:      *
00032:      * @param plotName
00033:      *
```

```
00034:      */
00035:     public TrendPlot(String data, String plotName) {
00036:
00037:         this.plotName = plotName;
00038:         timeSeries = new TimeSeries(data, Millisecond.class);
00039:
00040:         initialize();
00041:     }
00042:
00043:     /**
00044:      * initialiser plottet
00045:      */
00046:     private void initialize() {
00047:         TimeSeriesCollection dataset = new TimeSeriesCollection(timeSeries);
00048:         JFreeChart chart = ChartFactory.createTimeSeriesChart(plotName,
00049:                 "Time",
00050:                 "Error Value",
00051:                 dataset,
00052:                 true,
00053:                 true,
00054:                 false);
00055:
00056:         final XYPlot plot = chart.getXYPlot();
00057:         ValueAxis axis = plot.getDomainAxis();
00058:         axis.setAutoRange(true);
00059:         axis.setFixedAutoRange(20000.0);
00060:         label = new ChartPanel(chart);
00061:     }
00062:
00063:     /**
00064:      * Oppdaterer plottet
00065:      * @param errorValue
00066:      */
```

```
00067: 
00068:     public void updatePlot(double errorValue) {
00069:         //metode i timeseries som oppdaterer plottet
00070:         timeSeries.addOrUpdate(new Millisecond(), errorValue);
00071: 
00072:     }
00073: 
00074:     /**
00075:      * Getter som brukes av GUI for Ã¥ hente plottet
00076:      *
00077:      * @return
00078:      */
00079:     public ChartPanel getChartPanel() {
00080:         return label;
00081:     }
00082: 
00083: }
00084: 
```

Vedlegg 10
Kildekode serverapplikasjon

```
00001: package USVProsjekt;
00002:
00003: import USVProsjekt.NMEAparser.GPSPosition;
00004: import java.io.File;
00005: import java.io.FileWriter;
00006: import java.io.IOException;
00007: import java.io.PrintWriter;
00008: import java.net.ServerSocket;
00009: import java.util.Date;
00010: import java.util.Timer;
00011: import org.apache.commons.io.FileUtils;
00012:
00013: /**
00014:  * Hovedklasse for applikasjon
00015:  *
00016:  * @author Albert
00017:  */
00018: public class Application extends Thread {
00019:
00020:     private SerialConnection serialGPS;
00021:     private SerialConnection serialIMU;
00022:     private SerialConnection serialWind;
00023:     private SerialConnection serialThrust;
00024:
00025:     private GPSreader gps;
00026:     private IMUreader imu;
00027:     private WindReader windReader;
00028:     private ThrustWriter thrustWriter;
00029:     private GPSPositionStorageBox gpsPositionStorage;
00030:     private GPSPosition gpsPosition;
00031:
00032:     private double latitudeBody;
00033:     private double longitudeBody;
```

```
00034:     private double latitudeReference;
00035:     private double longitudeReference;
00036:
00037:     private double xNorth;
00038:     private double yEast;
00039:     private float yaw, heading, headingReference;
00040:
00041:     private double speed;
00042:     private double direction;
00043:
00044:     private double windSpeed;
00045:     private double windDirection;
00046:     private double temperature;
00047:
00048:     private int guiCommand;
00049:     private Server server;
00050:
00051:     private boolean dpStarted;
00052:     private DynamicPositioning dynamicPositioning;
00053:     private RemoteOperation remoteOperation;
00054:     private double[] northEastPosition;
00055:
00056:     private Timer timer;
00057:     private int gainChanged;
00058:     private float newControllerGain;
00059:     private int northInc;
00060:     private int eastInc;
00061:     private float incrementAmountX;
00062:     private float incrementAmountY;
00063:     private double[] remoteCommand;
00064:     private int egnos;
00065:     private NorthEastPositionStorageBox northEastPositionStorage;
00066:     private float yawSpeed;
```

```
00067: 
00068:     public Application(Server server) {
00069: 
00070:         xNorth = 0.0f;
00071:         yEast = 0.0f;
00072:         yaw = 0.0f;
00073:         heading = 0.0f;
00074:         speed = 0.0f;
00075:         direction = 0.0f;
00076:         windSpeed = 0.0f;
00077:         windDirection = 0.0f;
00078:         temperature = 0.0f;
00079:         latitudeBody = 0.0f;
00080:         longitudeBody = 0.0f;
00081:         latitudeReference = 0.0f;
00082:         longitudeReference = 0.0f;
00083: 
00084:         guiCommand = 0;
00085: 
00086:         headingReference = 0;
00087:         newControllerGain = 0;
00088:         gainChanged = 0;
00089:         northInc = 0;
00090:         eastInc = 0;
00091: 
00092:         remoteCommand = new double[3];
00093: 
00094:         this.server = server;
00095:     }
00096: 
00097:     @Override
00098:     public void run() {
00099:         readPreviousTuningsFromFile();
```

```
00100:         while (guiCommand != 4) {
00101:             //*********************************************
00102:             //Lagrer verdier fra klient
00103:             guiCommand = server.getGuiCommand();
00104:             headingReference = server.getHeadingReference();
00105:             if (server.isGainChanged()) {
00106:                 gainChanged = server.getGainChanged();
00107:                 newControllerGain = server.getControllerGain();
00108:             } else {
00109:                 gainChanged = 0;
00110:             }
00111:             northInc = server.getNorthIncDecRequest();
00112:             eastInc = server.getEastIncDecRequest();
00113:             remoteCommand = server.getRemoteCommand();
00114: 
00115:             //*********************************************
00116:             //UtfÃ¸rer sekundere oppgaver basert pÃ¥ klientÃ¸nske
00117:             //Stopper timere og resetter flagget
00118:             if (guiCommand != 1 && dpStarted) {
00119:                 timer.cancel();
00120:                 gps.setReferencePositionOff();
00121:                 dpStarted = false;
00122:                 incrementAmountX = 0;
00123:                 incrementAmountY = 0;
00124:                 northEastPositionStorage.setPosition(new double[]{0, 0});
00125:                 float[][] a = dynamicPositioning.getAllControllerTunings();
00126:                 storeControllerTunings(a);
00127:                 dynamicPositioning = new DynamicPositioning(thrustWriter,
00128:                         northEastPositionStorage, imu);
00129:                 dynamicPositioning.setPreviousGains(a);
00130:                 System.out.println("timer cancelled and flag reset");
00131:             }
00132:             //Endrer forsterkningskonstanter dersom endring er oppdaget
```

```
00133:             if (gainChanged != 0) {
00134:                 dynamicPositioning.setNewControllerGain(gainChanged,
00135:                         newControllerGain);
00136:                 float[][] a = dynamicPositioning.getAllControllerTunings();
00137:                 storeControllerTunings(a);
00138:             }
00139:             //endrer referansen dersom en eller flere av parameterene != 0
00140:             if (northInc != 0 || eastInc != 0) {
00141:                 setIncrementAmount(northInc, eastInc);
00142:             }
00143: 
00144:             //*********************************************
00145:             //UtfÃ¸rer primÃ¦re oppgaver basert pÃ¥ klientÃ¸nske
00146:             switch (guiCommand) {
00147:                 default:
00148:                     idle();
00149:                     break;
00150:                 case 1:
00151:                     dynamicPositioning();
00152:                     break;
00153:                 case 2:
00154:                     remoteOperation();
00155:                     break;
00156:             }
00157:             server.setDataFields(getDataLine());
00158: 
00159:         }
00160: 
00161:         stopThreads();
00162:         System.out.println("Run()-method in Application Class finished");
00163:     }
00164: 
00165:     /**
```

```
00166:      * BÃ¥ten ligger pÃ¥ "tomgang"
00167:      */
00168:     private void idle() {
00169:         dynamicPositioning.stopWriter();
00170:         thrustWriter.setThrustForAll(new double[]{0d, 0d, 0d, 0d});
00171:         thrustWriter.writeThrust();
00172:         updateBasicFields();
00173:     }
00174:
00175:     /**
00176:      * Aktiverer DP modus
00177:      */
00178:     private void dynamicPositioning() {
00179:         updateAllFields();
00180:         gps.lockReferencePosition();
00181:         dynamicPositioning.setReferenceHeading(headingReference);
00182:         if (!dpStarted) {
00183:             dynamicPositioning.startWriter();
00184:             dynamicPositioning.resetControllerErrors();//nullstill feil
00185:             int startTime = 0;
00186:             int periodTime = 200;
00187:             timer = new Timer();
00188:             timer.scheduleAtFixedRate(dynamicPositioning,
00189:                     startTime, periodTime); //start DP pÃ¥ konstant intervall
00190:             dpStarted = true;
00191:         }
00192:     }
00193:
00194:     /**
00195:      * Fjernstyringsmodus
00196:      */
00197:     private void remoteOperation() {
00198:         dynamicPositioning.stopWriter();
```

```
00199:         updateBasicFields();
00200:         remoteOperation.remoteOperate(remoteCommand);
00201:     }
00202:
00203:     /**
00204:      * Oppdaterer variabler for alminnelige felt
00205:      */
00206:     private void updateBasicFields() {
00207:         if (gpsPositionStorage.isNewPosition()) {
00208:             gpsPosition = gpsPositionStorage.getPosition();
00209:         }
00210:         latitudeBody = gpsPosition.lat;
00211:         longitudeBody = gpsPosition.lon;
00212:         windSpeed = windReader.getWindSpeed();
00213:         windDirection = windReader.getWindDirection();
00214:         temperature = windReader.getTemperature();
00215:         heading = imu.getHeading();
00216:         yawSpeed = imu.getYawSpeedValue();
00217:         speed = gpsPosition.velocity;
00218:         direction = gpsPosition.dir;
00219:         egnos = gpsPosition.quality;
00220:         latitudeReference = 0;
00221:         longitudeReference = 0;
00222:         xNorth = 0;
00223:         yEast = 0;
00224:         yaw = 0;
00225:     }
00226:
00227:     /**
00228:      * Oppdaterer alle felt
00229:      */
00230:     private void updateAllFields() {
00231:         updateBasicFields();
```

```
00232:         if (northEastPositionStorage.isNewPosition()) {
00233:             northEastPosition = northEastPositionStorage.getPosition();
00234:         }
00235:         latitudeReference = gps.getLatRef();
00236:         longitudeReference = gps.getLonRef();
00237:         xNorth = northEastPosition[0] + incrementAmountX;
00238:         yEast = northEastPosition[1] + incrementAmountY;
00239:         yaw = imu.getYawValue();
00240:
00241:     }
00242:
00243:     /**
00244:      * Initialiserer applikasjonen
00245:      */
00246:     public void initializeApplication() {
00247:         boolean windows = System.getProperty("os.name").contains("Windows");
00248:         String comPortGPS;
00249:         String comPortIMU;
00250:         String comPortWind;
00251:         String comPortThrust;
00252:         //communication parameters
00253:         if (windows) {
00254:             comPortGPS = "COM4";
00255:             comPortIMU = "COM5";
00256:             comPortWind = "COM6";
00257:             comPortThrust = "COM7";
00258:         } else {
00259:             comPortGPS = "ttyACM0";
00260:             comPortIMU = "ttyACM1";
00261:             comPortWind = "ttyACM3";
00262:             comPortThrust = "ttyACM2";
00263:         }
00264:         int baudRateGPS = 115200;
```

```
00265:
00266:          int baudRateIMU = 57600;
00267:
00268:          int baudRateWind = 57600;
00269:
00270:          int baudRateThrust = 115200;
00271:
00272:          //One serial connection for each sensor/port
00273:          serialGPS = new SerialConnection(comPortGPS,
00274:                  baudRateGPS);
00275:
00276:          serialIMU = new SerialConnection(comPortIMU,
00277:                  baudRateIMU);
00278:
00279:          serialWind = new SerialConnection(comPortWind,
00280:                  baudRateWind);
00281:
00282:          serialThrust = new SerialConnection(comPortThrust,
00283:                  baudRateThrust);
00284:
00285:          northEastPositionStorage = new NorthEastPositionStorageBox();
00286:          // Create and start threads
00287:          gpsPositionStorage = new GPSPositionStorageBox();
00288:          gps = new GPSreader(serialGPS, Identifier.GPS,
00289:                  northEastPositionStorage);
00290:          // Set gps position storage box, and initialize with values
00291:          gps.setStorageBox(gpsPositionStorage);
00292:          gps.connectToSerialPortAndDisplayGPSInfo();
00293:          gps.setName("GPS Reader Thread");
00294:          gps.start();
00295:
00296:          northEastPosition = northEastPositionStorage.getPosition();
00297:
```

```
00298:         gpsPosition = gpsPositionStorage.getPosition();
00299:
00300:         imu = new IMUreader(serialIMU, Identifier.IMU);
00301:         imu.connectToSerialPortAndDisplayIMUInfo();
00302:         imu.setName("IMU Reader Thread");
00303:         imu.start();
00304:
00305:         windReader = new WindReader(serialWind, Identifier.WIND);
00306:         windReader.connectToSerialPortAndDisplayWindInfo();
00307:         windReader.setName("Wind Reader Thread");
00308:         windReader.start();
00309:
00310:         thrustWriter = new ThrustWriter(serialThrust, Identifier.THRUSTERS);
00311:         dynamicPositioning = new DynamicPositioning(thrustWriter,
00312:                 northEastPositionStorage, imu);
00313:         remoteOperation = new RemoteOperation(thrustWriter);
00314:     }
00315:
00316:     public static void main(String[] args) throws Exception {
00317:         ServerSocket ssocket = new ServerSocket(2345);
00318:         //While true sÃ¸rger for at vi kan koble til og koble fra som Ã¸nsket
00319:         while (true) {
00320:             Server server = new Server(ssocket);
00321:             //denne metoden blokker til forbindelse
00322:             //er opprettet
00323:             server.acceptConnection();
00324:             Application app = new Application(server);
00325:             app.initializeApplication();
00326:             server.start();
00327:             app.start();
00328:         }
00329:     }
00330:
```

```
00331:     /**
00332:      * Stringen med data som skal sendes til klienten
00333:      *
00334:      * @return
00335:      */
00336:     private String getDataLine() {
00337:         float[][] a = dynamicPositioning.getAllControllerTunings();
00338:         float[] vector = dynamicPositioning.getPIDOutputVector();
00339:         return "Latitude: " + latitudeBody + " Longitude: "
00340:                 + longitudeBody + " xNorth: " + xNorth + " Sway: " + yEast
00341:                 + " Heading: " + heading + " Speed: " + speed + " Direction: "
00342:                 + direction + " WindSpeed: " + windSpeed
00343:                 + " WindDirection: " + windDirection
00344:                 + " Temperature: " + temperature + " LatRef: "
00345:                 + latitudeReference + " LonRef: " + longitudeReference + " "
00346:                 + a[0][0] + " " + a[0][1] + " " + a[0][2] + " "
00347:                 + a[1][0] + " " + a[1][1] + " " + a[1][2] + " "
00348:                 + a[2][0] + " " + a[2][1] + " " + a[2][2] + " "
00349:                 + egnos + " " + vector[0] + " " + vector[1] + " " + vector[2]
00350:                 + " " + yawSpeed;
00351:     }
00352:
00353:     /**
00354:      * Lar alle trÃ¥dene gÃ¥ ut av run()
00355:      */
00356:     private void stopThreads() {
00357:         gps.stopThread();
00358:         imu.stopThread();
00359:         windReader.stopThread();
00360:         thrustWriter.closeSerialConn();
00361:         server.stopThread();
00362:     }
00363:
```

```
00364:     /**
00365:      * Ã˜ker inkrement i referanse
00366:      *
00367:      * @param northInc
00368:      * @param eastInc
00369:      */
00370:     private void setIncrementAmount(int northInc, int eastInc) {
00371:         incrementAmountX -= northInc / 2.0f;
00372:         incrementAmountY -= eastInc / 2.0f;
00373:         dynamicPositioning.setIncrementAmount(northInc, eastInc);
00374:     }
00375:
00376:     /**
00377:      * Lagrer alle forsterkningskonstanter slik de kan brukes neste gang appen
00378:      * starter
00379:      *
00380:      * @param a
00381:      */
00382:     private void storeControllerTunings(float[][] a) {
00383:         File log = new File("PIDControllerTunings.txt");
00384:         try {
00385:             System.out.println("PID-tunings files created.");
00386:             log.createNewFile();
00387:             PrintWriter out = new PrintWriter(new FileWriter(log, false));
00388:
00389:             out.println(a[0][0] + " " + a[0][1] + " " + a[0][2] + " "
00390:                     + a[1][0] + " " + a[1][1] + " " + a[1][2] + " "
00391:                     + a[2][0] + " " + a[2][1] + " " + a[2][2] + " ");
00392:             out.println("Tunings stored at " + new Date().toString() + " by "
00393:                     + System.getProperty("user.name"));
00394:             out.close();
00395:         } catch (IOException e) {
00396:             System.out.println("COULD NOT LOG!!");
```

```
00397:         }
00398:     }
00399:
00400:     private void readPreviousTuningsFromFile() {
00401:         try {
00402:             String s = FileUtils.readFileToString(
00403:                     new File(System.getProperty("user.dir")
00404:                             + "//PIDControllerTunings.txt"));
00405:
00406:             String d[] = s.split(" ");
00407:             float a[][] = new float[][]{{Float.parseFloat(d[0]),
00408:                 Float.parseFloat(d[1]), Float.parseFloat(d[2])},
00409:             {Float.parseFloat(d[3]), Float.parseFloat(d[4]),
00410:                 Float.parseFloat(d[5])},
00411:             {Float.parseFloat(d[6]), Float.parseFloat(d[7]),
00412:                 Float.parseFloat(d[8])}};
00413:             dynamicPositioning.setPreviousGains(a);
00414:         } catch (IOException ex) {
00415:             System.out.println("Exception readfile)");
00416:         }
00417:
00418:     }
00419:
00420: }
```

```
00001: package USVProsjekt;
00002:
00003: import java.io.File;
00004: import java.io.FileWriter;
00005: import java.io.IOException;
00006: import java.io.PrintWriter;
00007: import java.util.Date;
00008: import java.util.TimerTask;
00009:
00010: /**
00011:  * Oppretter alle PID regulatorene, kjÃ¸rer pÃ¥ bestemt tid og kjÃ¸rer thrusterene
00012:  *
00013:  * @author Albert
00014:  */
00015: public class DynamicPositioning extends TimerTask {
00016:
00017:     private PIDController xNorthPID;
00018:     private PIDController yEastPID;
00019:     private PIDController headingPID;
00020:
00021:     private float outputX;
00022:     private float outputY;
00023:     private float outputN;
00024:
00025:     private float xNorthInput;
00026:     private float yEastInput;
00027:     private float headingInput;
00028:
00029:     private float incrementAmountX;
00030:     private float incrementAmountY;
00031:
00032:     private float headingReference;
00033:     private double[] position;
```

```
00034:
00035:     private ThrustAllocator thrustAllocator;
00036:
00037:     private RotationMatrix Rz;
00038:     private double[] forceOutputNewton;
00039:
00040:     private ThrustWriter thrustWriter;
00041:     private float xNorthReference;
00042:     private float yEastReference;
00043:     private PrintWriter nedWriter;
00044:     private NorthEastPositionStorageBox northEast;
00045:     private IMUreader imu;
00046:
00047:     public DynamicPositioning(ThrustWriter thrustWriter,
00048:             NorthEastPositionStorageBox northEast, IMUreader imu) {
00049:         xNorthPID = new PIDController();
00050:         yEastPID = new PIDController();
00051:         headingPID = new PIDController();
00052:         this.northEast = northEast;
00053:         this.imu = imu;
00054:         outputX = 0.0f;
00055:         outputY = 0.0f;
00056:         outputN = 0.0f;
00057:         position = new double[2];
00058:         headingReference = 0.0f;
00059:         thrustAllocator = new ThrustAllocator();
00060:         forceOutputNewton = new double[4];
00061:         this.thrustWriter = thrustWriter;
00062:     }
00063:
00064:     public void setPreviousGains(float[][] a) {
00065:         xNorthPID.setTunings(a[0][0], a[0][1], a[0][2]);
00066:         yEastPID.setTunings(a[1][0], a[1][1], a[1][2]);
```

```
00067:         headingPID.setTunings(a[2][0], a[2][1], a[2][2]);
00068:     }
00069:
00070:     public void setReferenceNorth(float xNorth) {
00071:         xNorthReference = xNorth;
00072:     }
00073:
00074:     public void setReferenceEast(float yEast) {
00075:         yEastReference = yEast;
00076:     }
00077:
00078:     public void setReferenceHeading(float heading) {
00079:         headingReference = heading;
00080:     }
00081:
00082:     @Override
00083:     public void run() {
00084:         try {//XYN from SNAME notation
00085:             if (northEast.isNewPosition()) {
00086:                 position = northEast.getPosition();
00087:             }
00088:             //Henter nord-, Ã¸st- og peilingsverdiene
00089:             xNorthInput = (float) (position[0] + incrementAmountX);
00090:             yEastInput = (float) (position[1] + incrementAmountY);
00091:             headingInput = imu.getHeading();
00092:
00093:             //Henter output fra hver PID regulator
00094:             float X = xNorthPID.computeOutput(xNorthInput, xNorthReference,
00095:                     false);
00096:             float Y = yEastPID.computeOutput(yEastInput, yEastReference,
00097:                     false);
00098:             float N = headingPID.computeOutput(imu.getHeading(),
00099:                     headingReference, true);
```

```
00100:             //setter kreftene X og Y, og momentet N.
00101:             setPIDOutputVector(X, Y, N);//synchronized
00102:             nedWriter.println((xNorthReference - xNorthInput) + " "
00103:                     + (yEastReference - yEastInput) + " "
00104:                     + (headingReference - headingInput));
00105:
00106:             Rz = new RotationMatrix(headingInput);
00107:             //Rz'*Tau. Mutltipliserer vektoren fra PIDene med den transponerte
00108:             //rotasjons matrisen Rz(psi)
00109:             double[] XYNtransformed = Rz.multiplyRzwithV(outputX, outputY,
00110:                     outputN);
00111:             //returnerer kreftene til hver thruster
00112:             forceOutputNewton = thrustAllocator.calculateOutput(XYNtransformed);
00113:             thrustWriter.setThrustForAll(forceOutputNewton);
00114:             thrustWriter.writeThrust();
00115:         } catch (Exception ex) {
00116:             System.out.println("exception dp");
00117:         }
00118:     }
00119:
00120:     /**
00121:      * Setter PID kreftene og momentet.
00122:      *
00123:      * @param X
00124:      * @param Y
00125:      * @param N
00126:      */
00127:     public synchronized void setPIDOutputVector(float X, float Y, float N) {
00128:         outputX = X;
00129:         outputY = Y;
00130:         outputN = N;
00131:     }
00132:
```

```
00133:     /**
00134:      * Henter PID vektoren
00135:      *
00136:      * @return
00137:      */
00138:     public synchronized float[] getPIDOutputVector() {
00139:         return new float[]{outputX, outputY, outputN};
00140:     }
00141:
00142:     /**
00143:      * Returnerer alle konstantene i reguleringen som en multidimensjonal vektor
00144:      *
00145:      * @return
00146:      */
00147:     public float[][] getAllControllerTunings() {
00148:         return new float[][]{
00149:             xNorthPID.getTunings(),
00150:             yEastPID.getTunings(),
00151:             headingPID.getTunings()
00152:         };
00153:     }
00154:
00155:     /**
00156:      * Stopper filskriving av data
00157:      */
00158:     public void stopWriter() {
00159:         if (nedWriter != null) {
00160:             nedWriter.close();
00161:         }
00162:     }
00163:
00164:     /**
00165:      * starter skriving av data til NED_Data.txt
```

```
00166:      */
00167:     public void startWriter() {
00168:         File log = new File("NED_Data.txt");
00169:
00170:         try {
00171:             log.createNewFile();
00172:             nedWriter = new PrintWriter(new FileWriter(log, true));
00173:             nedWriter.println(new Date().toString());
00174:         } catch (IOException ex) {
00175:         }
00176:     }
00177:
00178:     /**
00179:      * Setter nye forsterkningskonstanter
00180:      *
00181:      * @param gainChanged
00182:      * @param newControllerGain
00183:      */
00184:     public void setNewControllerGain(int gainChanged, float newControllerGain) {
00185:         System.out.println("gainChanged: " + gainChanged);
00186:         if (gainChanged < 4) {
00187:             xNorthPID.setGain(gainChanged, newControllerGain);
00188:         } else if (gainChanged > 3 && gainChanged < 7) {
00189:             yEastPID.setGain(gainChanged, newControllerGain);
00190:         } else {
00191:             headingPID.setGain(gainChanged, newControllerGain);
00192:         }
00193:
00194:     }
00195:
00196:     /**
00197:      * nullstiller avvikene i alle regulatorene
00198:      */
```

```
00199:     public void resetControllerErrors() {
00200:         xNorthPID.resetErrors();
00201:         yEastPID.resetErrors();
00202:         headingPID.resetErrors();
00203:     }
00204: /**
00205:  * Endrer referansepunktet. Flytter refpunktet northInc/2 meter
00206:  * @param northInc
00207:  * @param eastInc
00208:  */
00209:     public synchronized void setIncrementAmount(int northInc, int eastInc) {
00210:         incrementAmountX -= northInc / 2.0f;
00211:         incrementAmountY -= eastInc / 2.0f;
00212:     }
00213:
00214: }
```

```
00001: package USVProsjekt;
00002:
00003: import USVProsjekt.NMEAparser.GPSPosition;
00004:
00005: /**
00006:  * Lagrer geodetisk posisjon, delt objekt
00007:  *
00008:  * @author vegard
00009:  */
00010: public class GPSPositionStorageBox {
00011:
00012:     private GPSPosition position;
00013:     private boolean newPosition;
00014:
00015:     public GPSPositionStorageBox() {
00016:         newPosition = true;
00017:     }
00018:
00019:     /**
00020:      * getter for posisjonsobjektet
00021:      *
00022:      * @return
00023:      */
00024:     public synchronized GPSPosition getPosition() {
00025:         newPosition = false;
00026:         return position;
00027:     }
00028:
00029:     /**
00030:      * setter for posisjonsobjektet
00031:      *
00032:      * @param position
00033:      */
```

```
00034:     public synchronized void setPosition(GPSPosition position) {
00035:         this.position = position;
00036:         newPosition = true;
00037:     }
00038:
00039:     /**
00040:      * flag for Ã¥ sjekke om ny posisjon er satt
00041:      *
00042:      * @return
00043:      */
00044:     public synchronized boolean isNewPosition() {
00045:         return newPosition;
00046:     }
00047:
00048: }
```

```
00001: package USVProsjekt;
00002:
00003: import java.io.File;
00004: import java.io.FileNotFoundException;
00005: import java.io.FileWriter;
00006: import java.io.IOException;
00007: import java.io.PrintWriter;
00008: import java.io.UnsupportedEncodingException;
00009:
00010: /**
00011:  * Klasse for Ã¥ lese NMEA setninger fra GPS.
00012:  *
00013:  * @author Bachelor USV
00014:  */
00015: public class GPSreader extends Thread {
00016:
00017:     private NMEAparser nmea;
00018:     private NEDtransform gpsProc;
00019:
00020:     private double latBody;
00021:     private double lonBody;
00022:     private double latReference;
00023:     private double lonReference;
00024:
00025:     private Identifier ID;
00026:
00027:     private final SerialConnection serialConnection;
00028:     private int initPeriod;
00029:     private boolean dynamicPositioning;
00030:     private boolean stop;
00031:
00032:     private GPSPositionStorageBox gpsStorage;
00033:     private NorthEastPositionStorageBox northEastStorage;
```

```
00034:
00035:     private PrintWriter nmeaWriter;
00036:
00037:     private boolean writerStarted;
00038:
00039:     public GPSreader(SerialConnection serialConnection, Identifier ID,
00040:             NorthEastPositionStorageBox northEastStorage) {
00041:         this.serialConnection = serialConnection;
00042:         this.northEastStorage = northEastStorage;
00043:         nmea = new NMEAparser();
00044:         initPeriod = 0;
00045:         gpsProc = new NEDtransform();
00046:
00047:         latBody = 0.0f;
00048:         lonBody = 0.0f;
00049:
00050:         this.ID = ID;
00051:         stop = false;
00052:     }
00053:
00054:     @Override
00055:     public void run() {
00056:         String line;
00057:         String[] lineData;
00058:         while (serialConnection.isConnected() && !stop) {
00059:             setReference();
00060:             if (!writerStarted) {
00061:                 try {
00062:                     File log = new File("NMEAData.txt");
00063:                     nmeaWriter = new PrintWriter(new FileWriter(log, true));
00064:                 } catch (FileNotFoundException |
00065:                         UnsupportedEncodingException ex) {
00066:                 } catch (IOException ex) {
```

```
00067:                     System.out.println("IO");
00068:                 }
00069:                 writerStarted = true;
00070:             }
00071:             line = serialConnection.getSerialLine();
00072:             lineData = line.split("\r\n");
00073:             if (lineData[0].startsWith("$")
00074:                     && lineData[1].startsWith("$") && checkEmpty(lineData[0])) {
00075:                 String NMEA1 = lineData[0];
00076:                 String NMEA2 = lineData[1];
00077:                 nmea.parse(NMEA1);
00078:                 nmea.parse(NMEA2);
00079:                 nmeaWriter.println(NMEA1);
00080:                 nmeaWriter.println(NMEA2);
00081:                 nmeaWriter.println(" ");
00082:                 latBody = (nmea.position.lat * (Math.PI) / 180.0);
00083:                 lonBody = (nmea.position.lon * (Math.PI) / 180.0);
00084:                 // Store gps coordinates
00085:                 gpsStorage.setPosition(nmea.position);
00086:             }
00087:
00088:             double[] xyNorthEast = gpsProc.getFlatEarthCoordinates(latBody,
00089:                     lonBody, latReference, lonReference);
00090:             // lagre N-E position relative to reference
00091:             northEastStorage.setPosition(xyNorthEast);
00092:
00093:             System.out.println("-------------------------------------------");
00094:             System.out.println("X position: " + xyNorthEast[0]);
00095:             System.out.println("Y position: " + xyNorthEast[1]);
00096:             System.out.println("-------------------------------------------");
00097:
00098:         }
00099:         System.out.println("Connection lost/closed on Thread: "
```

```
00100:                 + this.getName());
00101:         serialConnection.close();
00102:         if (nmeaWriter != null) {
00103:             nmeaWriter.close();
00104:         }
00105:     }
00106:
00107:     /**
00108:      * Kobler til seriellport og lytter etter data
00109:      */
00110:     public void connectToSerialPortAndDisplayGPSInfo() {
00111:         serialConnection.connectAndListen(ID);
00112:
00113:     }
00114:
00115:     private void setReference() {
00116:         while (!dynamicPositioning && !stop) {
00117:             if (writerStarted) {
00118:                 nmeaWriter.close();
00119:                 writerStarted = false;
00120:             }
00121:             //init period for Ã¥ forhindre Ã¥ parse korrupte data
00122:             while (initPeriod < 10 && serialConnection.isConnected()) {
00123:                 serialConnection.getSerialLine();
00124:                 initPeriod++;
00125:             }
00126:             String line = serialConnection.getSerialLine();
00127:             String[] lineData = line.split("\r\n");
00128:             if (lineData[0].startsWith("$") && lineData[1].startsWith("$")
00129:                     && checkEmpty(lineData[0])) {
00130:                 String NMEA1 = lineData[0];
00131:                 String NMEA2 = lineData[1];
00132:                 nmea.parse(NMEA1);
```

```
00133:                 nmea.parse(NMEA2);
00134:                 latReference = (nmea.position.lat * Math.PI / 180.0);
00135:                 lonReference = (nmea.position.lon * Math.PI / 180.0);
00136:                 gpsStorage.setPosition(nmea.position);
00137:             }
00138:
00139:         }
00140:     }
00141:
00142:     /**
00143:      * sjekker om NMEA setningen har tomme felt
00144:      *
00145:      * @param lineData
00146:      * @return
00147:      */
00148:     private boolean checkEmpty(String lineData) {
00149:         String[] checkData = lineData.split(",");
00150:         for (int i = 0; i < 6; i++) {
00151:             if (checkData[i].isEmpty()) {
00152:                 return false;
00153:             }
00154:         }
00155:         return true;
00156:     }
00157:
00158:     /**
00159:      * lÃ¥ser referanseposisjonen
00160:      */
00161:     public void lockReferencePosition() {
00162:         dynamicPositioning = true;
00163:     }
00164:
00165:     /**
```

```
00166:      * returener breddegrad i decimalgrader
00167:      *
00168:      * @return
00169:      */
00170:     public double getLatRef() {
00171:         return (latReference * (180.0 / Math.PI));
00172:     }
00173:
00174:     /**
00175:      * setter GPSposistion objektet
00176:      *
00177:      * @param storage
00178:      */
00179:     public void setStorageBox(GPSPositionStorageBox storage) {
00180:         this.gpsStorage = storage;
00181:         gpsStorage.setPosition(nmea.position);
00182:     }
00183:
00184:     /**
00185:      * getter for lengdegrad i decimalgrader
00186:      *
00187:      * @return
00188:      */
00189:     public double getLonRef() {
00190:         return (lonReference * (180.0 / Math.PI));
00191:     }
00192:
00193:     /**
00194:      * stopper trÃ¥den / gÃ¥r ut av run()
00195:      */
00196:     void stopThread() {
00197:         stop = true;
00198:     }
```

```
00199:
00200:     /**
00201:      * slÃ¥r av lÃ¥sen pÃ¥ referanseoppdatering
00202:      */
00203:     void setReferencePositionOff() {
00204:         dynamicPositioning = false;
00205:     }
00206:
00207: }
```

```
00001: package USVProsjekt;
00002:
00003: /**
00004:  *
00005:  * @author Albert
00006:  */
00007: public enum Identifier {
00008:     GPS,IMU,THRUSTERS,WIND,SURGE,SWAY,HEADING;
00009:
00010: }
```

```
00001: package USVProsjekt;
00002: 
00003: /**
00004:  *
00005:  * @author Albert
00006:  */
00007: public class IMUreader extends Thread {
00008: 
00009:     //IMU/Compass variables
00010:     private float yaw;
00011:     private float pitch;
00012:     private float roll;
00013:     private float yawSpeed;
00014: 
00015:     private Identifier ID;
00016:     private final SerialConnection serialConnection;
00017:     private int initPeriod;
00018:     private boolean stop;
00019: 
00020:     public IMUreader(SerialConnection serialConnection, Identifier ID) {
00021:         this.serialConnection = serialConnection;
00022:         this.ID = ID;
00023:         yaw = 0.0f;
00024:         pitch = 0.0f;
00025:         roll = 0.0f;
00026:         yawSpeed = 0.0f;
00027:         stop = false;
00028: 
00029:     }
00030: 
00031:     @Override
00032:     public void run() {
00033:         String line;
```

```
00034:         float[] magnData;
00035:
00036:         while (initPeriod < 5 && serialConnection.isConnected()) {
00037:             line = serialConnection.getSerialLine();
00038:             initPeriod++;
00039:         }
00040:         while (serialConnection.isConnected() && !stop) {
00041:             line = serialConnection.getSerialLine();
00042:             setYawValue(parseReceivedIMUline(line));
00043:         }
00044:         System.out.println("Connection lost/closed on Thread: "
00045:                 + this.getName());
00046:         serialConnection.close();
00047:     }
00048:
00049:     /**
00050:      * Endrer yaw fra -180 til 180 til 0 -360
00051:      *
00052:      * @param yaw
00053:      * @return
00054:      */
00055:     private float getHeadingFromYawValue(float yaw) {
00056:         if (yaw >= 90 && yaw <= 180) {
00057:             return yaw - 90;
00058:         }
00059:         if (yaw >= -180 && yaw <= 90) {
00060:             return yaw + 270;
00061:         } else {
00062:             return yaw;
00063:         }
00064:     }
00065:
00066:     //yaw:     (-179.9) - 0    //yaw:     0 - 180
```

```
00067:     //heading:  0 - 179.9     //heading: 180 - 359.9
00068:     public void connectToSerialPortAndDisplayIMUInfo() {
00069:         serialConnection.connectAndListen(ID);
00070:     }
00071:
00072:     /**
00073:      * henter data fra mottatt setning
00074:      *
00075:      * @param line
00076:      * @return
00077:      */
00078:     private float parseReceivedIMUline(String line) {
00079:         float yw;
00080:         if (line.startsWith("#")) {
00081:             String[] lineData = line.split(",");
00082:             if (lineData.length >= 3) {
00083:                 String xString = lineData[0].substring(5);
00084:                 yw = Float.parseFloat(xString);
00085:                 // Derivering for Ã¥ finne rotasjonshastighet
00086:                 setYawSpeedValue((yw - yaw) / 0.1f);
00087:                 return yw;
00088:             }
00089:         }
00090:         setYawSpeedValue(0.0f);
00091:         return yaw;
00092:     }
00093:
00094:     /**
00095:      * returnere yaw verdi
00096:      *
00097:      * @return
00098:      */
00099:     public synchronized float getYawValue() {
```

```
00100:         return yaw;
00101:     }
00102:
00103:     /**
00104:      * returnerer yaw hastighet
00105:      *
00106:      * @return
00107:      */
00108:     public synchronized float getYawSpeedValue() {
00109:         return yawSpeed;
00110:     }
00111:
00112:     /**
00113:      * lagrer yaw hastighets verdi
00114:      */
00115:     private synchronized void setYawSpeedValue(float yawSpeed) {
00116:         this.yawSpeed = yawSpeed;
00117:     }
00118:
00119:     public synchronized void setYawValue(float yaw) {
00120:         this.yaw = yaw;
00121:     }
00122:
00123:     public synchronized float getHeading() {
00124:         return getHeadingFromYawValue(yaw);
00125:     }
00126:
00127:     public float getPitchAngle() {
00128:         return pitch;
00129:     }
00130:
00131:     public float getRollAngle() {
00132:         return roll;
```

```
00133:     }
00134:
00135:     void stopThread() {
00136:         stop = true;
00137:     }
00138:
00139: }
```

```
00001: package USVProsjekt;
00002:
00003: /**
00004:  * Transformerer geodetiske koordinater for nÃ¥- og referanseverdi til
00005:  * koordinater i NED koordinatsystemet
00006:  * @author Albert
00007:  */
00008: public class NEDtransform extends Thread {
00009:
00010:     //World Geodetic System 1984 konstanter
00011:     //Lengste radius av jordens ellipsoide
00012:     private final double semimajorAxis = 6378137.0;
00013:     //korteste radius av jorden ellipsoide
00014:     private final double semiminorAxis = 6356752.0;
00015:     private final double flattening =
00016:             (semimajorAxis - semiminorAxis) / semimajorAxis;
00017:     private final double f;
00018:     private final double R;
00019:
00020:     public NEDtransform() {
00021:         f = flattening;
00022:         R=semimajorAxis;
00023:     }
00024:
00025:     /**
00026:      * Flat Earth Coordinates, optimal for small changes in lat/lon used.
00027:      *
00028:      * @param latBody
00029:      * @param lonBody
00030:      * @param latRef
00031:      * @param lonRef
00032:      * @return
00033:      */
```

```
00034:     public double[] getFlatEarthCoordinates(double latBody, double lonBody,
00035:             double latRef, double lonRef) {
00036:         double dMy = latBody - latRef;
00037:         double dL = lonBody - lonRef;
00038:
00039:         double rN = (R / (Math.sqrt(1 - (2 * f - f * f) *
00040:                 Math.pow(Math.sin(latRef), 2))));
00041:         double rM = rN * ((1 - (2 * f - f * f)) / (1 - (2 * f - f * f) *
00042:                 Math.pow(Math.sin(latRef), 2)));
00043:         double dN = (dMy / Math.atan(1 / rM));
00044:         double dE = (dL / Math.atan(1 / (rN * Math.cos(latRef))));
00045:         return new double[]{dN,dE};
00046:     }
00047:
00048:
00049:     //Denne gjelder for jord som IKKE er "flat" ikke brukt pga mer kompleks
00050:     //flat jord er godt for DP
00051:     /**
00052:      * Works for all distances.
00053:      *
00054:      * @param latitudeBody
00055:      * @param longitudeBody
00056:      * @param latitudeReference
00057:      * @param longitudeReference
00058:      * @return
00059:      */
00060: //    public double[] getBodyCInNEDByGeodeticBodyPosAndRef(double latitudeBody,
00061: //            double longitudeBody, double latitudeReference,
00062: //            double longitudeReference) {
00063: //        double NRef = getN(latitudeReference, longitudeReference);
00064: //        double[] xyzRef = llh2ECEF(latitudeReference, longitudeReference,
00065: //                0, NRef);
00066: //        double NBody = getN(latitudeReference, longitudeReference);
```

```
00067: //        double[] xyzBody = llh2ECEF(latitudeBody, longitudeBody, 0, NBody);
00068: //        double dx = xyzBody[0] - xyzRef[0];
00069: //        double dy = xyzBody[1] - xyzRef[1];
00070: //        double dz = xyzBody[2] - xyzRef[2];
00071: //
00072: //        double cosPhi = Math.cos(latitudeReference);
00073: //        double sinPhi = Math.sin(latitudeReference);
00074: //        double cosLambda = Math.cos(longitudeReference);
00075: //        double sinLambda = Math.sin(longitudeReference);
00076: //
00077: //        double t = cosLambda * dx + sinLambda * dy;
00078: //
00079: //        double dxEast = -sinLambda * dx + cosLambda * dy;
00080: //        double dzUp = cosPhi * t + sinPhi * dz;
00081: //        double dyNorth = -sinPhi * t + cosPhi * dz;
00082: //
00083: //        double xNorth = dyNorth;
00084: //        double yEast = dxEast;
00085: //        double zDown = -dzUp;
00086: //        return new double[]{xNorth, yEast, zDown};
00087: //    }
00088: //
00089: //    private double getN(double lat, double lon) {
00090: //        double a = semimajorAxis;
00091: //        double b = semiminorAxis;
00092: //        double acos = a * Math.cos(lat);
00093: //        double bsin = b * Math.cos(lon);
00094: //        double aa = a * a;
00095: //        return aa / Math.sqrt(acos * acos + bsin * bsin);
00096: //    }
00097:     /**
00098:      * Converts latitude, longitude and height to ECEF coordinate system
00099:      *
```

```
00100:      * @param latitude
00101:      * @param longitude
00102:      * @param height
00103:      * @return var-index: x-0, y-1, z-2
00104:      */
00105: //    private double[] llh2ECEF(double lat,
00106: //            double lon, double height, double N) {
00107: //        double x = (N + height) * Math.cos(lat) * Math.cos(lon);
00108: //        double y = (N + height) * Math.cos(lat) * Math.sin(lon);
00109: //        double bOvera = semiminorAxis / semimajorAxis;
00110: //        double z = (N * bOvera * bOvera + height) * Math.sin(lat);
00111: //        double[] xyz = new double[]{x, y, z};
00112: //        return xyz;
00113: //    }
00114: }
```

```
00001: package USVProsjekt;
00002:
00003: import java.util.HashMap;
00004: import java.util.Map;
00005:
00006:
00007: public class NMEAparser {
00008:
00009:
00010:     interface SentenceParser {
00011:         public boolean parse(String [] tokens, GPSPosition position);
00012:     }
00013:
00014:     // utils
00015:     static double Latitude2Decimal(String lat, String NS) {
00016:         double med = Double.parseDouble(lat.substring(2))/60.0;
00017:         med +=  Double.parseDouble(lat.substring(0, 2));
00018:         if(NS.startsWith("S")) {
00019:             med = -med;
00020:         }
00021:         return med;
00022:     }
00023:
00024:     static double Longitude2Decimal(String lon, String WE) {
00025:         double med = Double.parseDouble(lon.substring(3))/60.0;
00026:         med +=  Double.parseDouble(lon.substring(0, 3));
00027:         if(WE.startsWith("W")) {
00028:             med = -med;
00029:         }
00030:         return med;
00031:     }
00032:
00033:     // parsers
```

```
00034:     class GPGGA implements SentenceParser {
00035:         public boolean parse(String [] tokens, GPSPosition position) {
00036:             position.time = Double.parseDouble(tokens[1]);
00037:             position.lat = Latitude2Decimal(tokens[2], tokens[3]);
00038:             position.lon = Longitude2Decimal(tokens[4], tokens[5]);
00039:             position.quality = Integer.parseInt(tokens[6]);
00040:             position.satNum = Integer.parseInt(tokens[7]);
00041:             position.altitude = Double.parseDouble(tokens[9]);
00042:             return true;
00043:         }
00044:     }
00045:
00046:     class GPGGL implements SentenceParser {
00047:         public boolean parse(String [] tokens, GPSPosition position) {
00048:             position.lat = Latitude2Decimal(tokens[1], tokens[2]);
00049:             position.lon = Longitude2Decimal(tokens[3], tokens[4]);
00050:             position.time = Double.parseDouble(tokens[5]);
00051:             return true;
00052:         }
00053:     }
00054:
00055:     class GPRMC implements SentenceParser {
00056:         public boolean parse(String [] tokens, GPSPosition position) {
00057:             position.time = Double.parseDouble(tokens[1]);
00058:             position.lat = Latitude2Decimal(tokens[3], tokens[4]);
00059:             position.lon = Longitude2Decimal(tokens[5], tokens[6]);
00060:             position.velocity = (
00061:                                 0.51444*Double.parseDouble(tokens[7]));
00062:             position.dir = Double.parseDouble(tokens[8]);
00063:             return true;
00064:         }
00065:     }
00066:
```

```
00067:     class GPVTG implements SentenceParser {
00068:         public boolean parse(String [] tokens, GPSPosition position) {
00069:             position.dir = Double.parseDouble(tokens[3]);
00070:             return true;
00071:         }
00072:     }
00073:
00074:     class GPRMZ implements SentenceParser {
00075:         public boolean parse(String [] tokens, GPSPosition position) {
00076:             position.altitude = Double.parseDouble(tokens[1]);
00077:             return true;
00078:         }
00079:     }
00080:
00081:         class PGTOP implements SentenceParser {
00082:         public boolean parse(String [] tokens, GPSPosition position) {
00083:                         char number = tokens[2].charAt(0);
00084:             position.antenna = Integer.parseInt(""+number);
00085:             return true;
00086:         }
00087:     }
00088:
00089:     public class GPSPosition {
00090:         public double time = 0.0f;
00091:         public double lat = 0.0f;
00092:         public double lon = 0.0f;
00093:         public boolean fixed = false;
00094:         public int quality = 0;
00095:         public double dir = 0.0f;
00096:         public double altitude = 0.0f;
00097:         public double velocity = 0.0f;
00098:                 public int antenna = 0;
00099:                 public int satNum = 0;
```

```
00100:
00101:         public void updatefix() {
00102:             fixed = quality > 0;
00103:         }
00104:
00105:                 @Override
00106:         public String toString() {
00107:             return String.format("POSITION: lat: %d, lon: "
00108:                                  + "%d, time: %d, Q: %d, dir: %d, alt(m):"
00109:                                  + " %d, vel(m/s): %d, ant: %d, satNum: "
00110:                                  + "%d", lat, lon, time, quality, dir,
00111:                                  altitude, velocity, antenna,satNum);
00112:         }
00113:     }
00114:
00115:     GPSPosition position = new GPSPosition();
00116:
00117:     private static final Map<String, SentenceParser> sentenceParsers
00118:                 = new HashMap<String, SentenceParser>();
00119:     /**
00120:          * Konverterer NMEA setninger til brukende variabler
00121:          */
00122:     public NMEAparser() {
00123:         sentenceParsers.put("GPGGA", new GPGGA());
00124:         sentenceParsers.put("GPGGL", new GPGGL());
00125:         sentenceParsers.put("GPRMC", new GPRMC());
00126:         sentenceParsers.put("GPRMZ", new GPRMZ());
00127:         sentenceParsers.put("PGTOP", new PGTOP());
00128:         //only really good GPS devices have this sentence but ...
00129:         sentenceParsers.put("GPVTG", new GPVTG());
00130:     }
00131:
00132:     public synchronized GPSPosition parse(String line) {
```

```
00133:         if(line.startsWith("$")) {
00134:             String nmea = line.substring(1);
00135:             String[] tokens = nmea.split(",");
00136:             String type = tokens[0];
00137:             //sjekker hvilken setning som er mottatt og parser
00138:             if(sentenceParsers.containsKey(type)) {
00139:                             sentenceParsers.get(type).parse(tokens, position);
00140:             }
00141:             position.updatefix();
00142:         }
00143:
00144:         return position;
00145:     }
00146: }
```

```
00001: package USVProsjekt;
00002: 
00003: /**
00004:  *Lagringsboks for Nord og Ã˜st posisjon, delt mellom objekter
00005:  * @author vegard
00006:  */
00007: public class NorthEastPositionStorageBox {
00008:     private double[] position;
00009:     private boolean newPosition;
00010: 
00011:     public NorthEastPositionStorageBox() {
00012:         position = new double[2];
00013:         newPosition = true;
00014:     }
00015:     /**
00016:      * setter posisjonen
00017:      * @param position
00018:      */
00019:     public synchronized void setPosition(double[] position) {
00020:         this.position = position;
00021:         newPosition = true;
00022:     }
00023:     /**
00024:      * henter posisjoner
00025:      * @return
00026:      */
00027:     public synchronized double[] getPosition() {
00028:         newPosition = false;
00029:         return position;
00030:     }
00031:     /**
00032:      * flag for Ã¥ sjekke om posisjon er blitt satt
00033:      * @return
```

```
00034:      */
00035:     public synchronized boolean isNewPosition() {
00036:         return newPosition;
00037:     }
00038: }
```

```
00001: package USVProsjekt;
00002:
00003: /**
00004:  *
00005:  * @author Albert
00006:  */
00007: public class PIDController {
00008: //
00009:     //IO Variables
00010:
00011:     private float outputVariable;
00012:
00013:     //Gains
00014:     private float Kp;
00015:     private float Ki;
00016:     private float Kd;
00017:
00018:     //Class variables
00019:     private float integralTerm;
00020:     private float lastError;
00021:     private final float cycleTimeInSeconds;
00022:
00023:     private float maxOutput;
00024:     private float minOutput;
00025:
00026:     public PIDController() {
00027:         outputVariable = 0.0f;
00028:         integralTerm = 0.0f;
00029:         lastError = 0.0f;
00030:
00031:         maxOutput = 2 * 34.0f;//Maks output jaging og tverrskipss
00032:         minOutput = 2 * -29.0f;//Minimum output jaging og tverrskips
00033:
```

```
00034:         Kp = 1.0f;
00035:         Ki = 0.0f;
00036:         Kd = 0.0f;
00037:         cycleTimeInSeconds = 0.2f;
00038:     }
00039:
00040:     /**
00041:      * Beregner output for regulatoren
00042:      *
00043:      * @param newInput
00044:      * @param referenceVariable
00045:      * @param continuous
00046:      * @return
00047:      */
00048:     public float computeOutput(float newInput, float referenceVariable,
00049:             boolean continuous) {
00050:
00051:         float error = referenceVariable - newInput;
00052:
00053:         // Dersom kontinuerlig kan wrap around
00054:         if (continuous) {
00055:             maxOutput = 121.0f; //torque(yaw)
00056:             minOutput = -121.0f; //torque(yaw)
00057:             if (Math.abs(error) > 180) {
00058:                 if (error > 0) {
00059:                     error = error - 360.0f;
00060:                 } else {
00061:                     error = error + 360.0f;
00062:                 }
00063:             }
00064:         }
00065:         // Integrator anti-windup og integrator ledd
00066:         if ((integralTerm + error * cycleTimeInSeconds * Ki) < maxOutput
```

```
00067:                 && (integralTerm + error * cycleTimeInSeconds * Ki)
00068:                 > minOutput) {
00069:             integralTerm += Ki * error * cycleTimeInSeconds;
00070:         }
00071:         float dError = (error - lastError) / cycleTimeInSeconds;
00072:         //Beregn PID Output
00073:         outputVariable = Kp * error + integralTerm + Kd * dError;
00074:         limitOutputVariable();
00075:         lastError = error;
00076:         return outputVariable;
00077:
00078:     }
00079:
00080:     /**
00081:      * resetter feil
00082:      */
00083:     public void resetErrors() {
00084:         integralTerm = 0;
00085:         lastError = 0;
00086:     }
00087:
00088:     /**
00089:      * setter forsterkningskonstant for denne regulatoren
00090:      *
00091:      * @param Kp
00092:      * @param Ki
00093:      * @param Kd
00094:      */
00095:     public void setTunings(float Kp, float Ki, float Kd) {
00096:         this.Kp = Kp;
00097:         this.Ki = Ki;
00098:         this.Kd = Kd;
00099:     }
```

```
00100: 
00101:     /**
00102:      * returener forsterkningskonstantene for denne regulatoren
00103:      *
00104:      * @return
00105:      */
00106:     public float[] getTunings() {
00107:         return new float[]{Kp, Ki, Kd};
00108:     }
00109: 
00110:     /**
00111:      * setter forsterkningskonstantene for vdenne regulatoren
00112:      *
00113:      * @param gainChanged
00114:      * @param newControllerGain
00115:      */
00116:     void setGain(int gainChanged, float newControllerGain) {
00117:         switch (gainChanged) {
00118:             case 1:
00119:             case 4:
00120:             case 7:
00121:                 Kp = newControllerGain;
00122:                 break;
00123:             case 2:
00124:             case 5:
00125:             case 8:
00126:                 Ki = newControllerGain;
00127:                 break;
00128:             case 3:
00129:             case 6:
00130:             case 9:
00131:                 Kd = newControllerGain;
00132:                 break;
```

```
00133:         }
00134:     }
00135:
00136:     /**
00137:      * begrenser output variablene
00138:      */
00139:     private void limitOutputVariable() {
00140:         if (outputVariable > maxOutput) {
00141:             outputVariable = maxOutput;
00142:         } else if (outputVariable < minOutput) {
00143:             outputVariable = minOutput;
00144:         }
00145:     }
00146:
00147: }
```

```
00001: package USVProsjekt;
00002:
00003: /**
00004:  * Klasse som kontrollerer fjernstyring
00005:  *
00006:  * @author Albert
00007:  */
00008: public class RemoteOperation {
00009:
00010:     private ThrustAllocator thrustAllocator;
00011:     private ThrustWriter thrustWrite;
00012:
00013:     public RemoteOperation(ThrustWriter thrustWriter) {
00014:         thrustAllocator = new ThrustAllocator();
00015:         this.thrustWrite = thrustWriter;
00016:     }
00017:
00018:     /**
00019:      * Metode for styring
00020:      *
00021:      * @param remoteCommand
00022:      */
00023:     public void remoteOperate(double[] remoteCommand) {
00024:         try {
00025:             thrustWrite.setThrustForAll(
00026:                     thrustAllocator.calculateOutput(remoteCommand));
00027:             thrustWrite.writeThrust();
00028:         } catch (Exception ex) {
00029:             System.out.println("exception ro");
00030:         }
00031:     }
00032: }
```

```
00001: package USVProsjekt;
00002:
00003: import org.apache.commons.math3.linear.ArrayRealVector;
00004: import org.apache.commons.math3.linear.BlockRealMatrix;
00005: import org.apache.commons.math3.linear.RealVector;
00006:
00007: /**
00008:  * Klasse for Ã¥ opprette rotasjonsmatrise mellom BODY og NED
00009:  *
00010:  * @author Albert
00011:  */
00012: public class RotationMatrix {
00013:
00014:     private BlockRealMatrix Rz;
00015:     private double[][] raw;
00016:     ArrayRealVector vector;
00017:
00018:     public RotationMatrix(float headingDegrees) {
00019:         float headingRadians = headingDegrees * (float) Math.PI / 180.0f;
00020:         raw = new double[][]{
00021:             {c(headingRadians), -s(headingRadians), 0},
00022:             {s(headingRadians), c(headingRadians), 0},
00023:             {0, 0, 1}};
00024:         Rz = new BlockRealMatrix(raw);
00025:     }
00026:
00027:     /**
00028:      * transponerer rotasjonsmatrisen og multipliserer den med input vektor
00029:      *
00030:      * @param u
00031:      * @param v
00032:      * @param w
00033:      * @return
```

```
00034:      */
00035:     public double[] multiplyRzwithV(float u, float v, float w) {
00036:         vector = new ArrayRealVector(new double[]{u, v, w});
00037:         //Rz'*returnVector 3x3  * 3x1 = 3x1
00038:         RealVector returnVector = Rz.transpose().operate(vector);
00039:         return returnVector.toArray();
00040:     }
00041:
00042:     /**
00043:      * cosinus funskjon
00044:      *
00045:      * @param radians
00046:      * @return
00047:      */
00048:     private float c(float radians) {
00049:         return (float) Math.cos(radians);
00050:     }
00051:
00052:     /**
00053:      * sinusfunksjon
00054:      *
00055:      * @param radians
00056:      * @return
00057:      */
00058:     private float s(float radians) {
00059:         return (float) Math.sin(radians);
00060:     }
00061: }
```

```
00001: package USVProsjekt;
00002:
00003: import com.fazecast.jSerialComm.SerialPort;
00004: import com.fazecast.jSerialComm.SerialPortDataListener;
00005: import com.fazecast.jSerialComm.SerialPortEvent;
00006: import java.io.BufferedReader;
00007: import java.io.IOException;
00008: import java.io.InputStreamReader;
00009: import java.io.PrintWriter;
00010:
00011: /**
00012:  * Klasse for Ã¥ opprette seriell forbindelse
00013:  *
00014:  * @author Albert
00015:  */
00016: public class SerialConnection {
00017:
00018:     private int baudRate;
00019:     private SerialPort comPort;
00020:     private String serialString;
00021:     private boolean available;
00022:
00023:     public SerialConnection(String comPortString, int baudRate) {
00024:         this.comPort = SerialPort.getCommPort(comPortString);
00025:         this.baudRate = baudRate;
00026:         serialString = "";
00027:         available = false;
00028:     }
00029:
00030:     /**
00031:      * lister opp tilgjengelige porter
00032:      */
00033:     public void listAvailableSerialPorts() {
```

```
00034:         SerialPort ports[] = SerialPort.getCommPorts();
00035:         for (SerialPort port : ports) {
00036:             System.out.println(port.getSystemPortName());
00037:         }
00038:
00039:     }
00040:
00041:     /**
00042:      * Kobler til med en ID (thruster,GPS...etc)
00043:      *
00044:      * @param ID
00045:      */
00046:     public void connect(Identifier ID) {
00047:         if (comPort.openPort()) {
00048:             comPort.setBaudRate(baudRate);
00049:             comPort.setComPortTimeouts(
00050:                     SerialPort.TIMEOUT_READ_SEMI_BLOCKING, 100, 0);
00051:             System.out.println("Successfully connected to " + ID
00052:                     + " via " + comPort.getSystemPortName() + " with Baud Rate:"
00053:                     + " " + comPort.getBaudRate());
00054:         } else {
00055:             System.out.println("Not able to connect to " + ID + "..");
00056:
00057:         }
00058:     }
00059:
00060:     /**
00061:      * metode som skriver pwm verdier over seriellporten
00062:      *
00063:      * @param thrustMicros1
00064:      * @param thrustMicros2
00065:      * @param thrustMicros3
00066:      * @param thrustMicros4
```

```
00067:      */
00068:     public void writeThrustMicros(int thrustMicros1, int thrustMicros2,
00069:             int thrustMicros3, int thrustMicros4) {
00070:         //skriver til arduino
00071:         String writeString;
00072:         writeString = "" + thrustMicros1 + ":" + thrustMicros2 + ":"
00073:                 + thrustMicros3 + ":" + thrustMicros4 + ":";
00074:
00075:         PrintWriter output = new PrintWriter(comPort.getOutputStream());
00076:         output.write(writeString);
00077:         output.flush();
00078:         //Leser tilbake dataene fra arduino
00079:         try {
00080:             BufferedReader br = new BufferedReader(
00081:                     new InputStreamReader(comPort.getInputStream(), "UTF-8"));
00082:             System.out.println(br.readLine());
00083:         } catch (IOException e) {
00084:             System.out.println("ioex writeThrustMicros() in SerialConnection");
00085:         }
00086:     }
00087:
00088:     /**
00089:      * kobler til og lytter pÃ¥ data pÃ¥ seriellportene
00090:      *
00091:      * @param ID
00092:      */
00093:     public void connectAndListen(Identifier ID) {
00094:         connect(ID);
00095:         comPort.addDataListener(new SerialPortDataListener() {
00096:             @Override
00097:             public int getListeningEvents() {
00098:                 return SerialPort.LISTENING_EVENT_DATA_AVAILABLE;
00099:             }
```

```
00100:
00101:             @Override
00102:             public void serialEvent(SerialPortEvent event) {
00103:                 if (event.getEventType()
00104:                         != SerialPort.LISTENING_EVENT_DATA_AVAILABLE) {
00105:                     return;
00106:                 }
00107:                 byte[] newData = null;
00108:                 try {
00109:                     switch (ID) {
00110:                         case GPS:
00111:                             Thread.sleep(90);
00112:                             newData = new byte[comPort.bytesAvailable()];
00113:                             break;
00114:                         case WIND:
00115:                             Thread.sleep(120);
00116:                             newData = new byte[comPort.bytesAvailable()];
00117:                             break;
00118:                         case IMU:
00119:                             //IMU
00120:                             Thread.sleep(30);
00121:                             newData = new byte[comPort.bytesAvailable()];
00122:                             break;
00123:                     }
00124:                 } catch (InterruptedException ex) {
00125:                 }
00126:                 comPort.readBytes(newData, newData.length);
00127:                 setSerialLine(new String(newData));
00128:             }
00129:         }
00130:         );
00131:     }
00132:
```

```
00133:     /**
00134:      * setter seriell linjen som er mottatt
00135:      *
00136:      * @param serialLine
00137:      */
00138:     public synchronized void setSerialLine(String serialLine) {
00139:         while (available) {
00140:             try {
00141:                 wait();
00142:             } catch (InterruptedException ex) {
00143:             }
00144:         }
00145:
00146:         available = true;
00147:         notifyAll();
00148:         serialString = serialLine;
00149:     }
00150:
00151:     /**
00152:      * heter den mottatte seriell linjen fra denne klassen
00153:      *
00154:      * @return
00155:      */
00156:     public synchronized String getSerialLine() {
00157:         while (!available) {
00158:             try {
00159:                 wait();
00160:             } catch (InterruptedException ex) {
00161:
00162:             }
00163:         }
00164:         available = false;
00165:         notifyAll();
```

```
00166:         return serialString;
00167:     }
00168: /**
00169:  * sjekker om porten er Ã¥pnet
00170:  * @return
00171:  */
00172:     public boolean isConnected() {
00173:         return comPort.isOpen();
00174:     }
00175: /**
00176:  * lukker porten
00177:  */
00178:     public void close() {
00179:         comPort.closePort();
00180:     }
00181: }
```

```
00001: package USVProsjekt;
00002:
00003: import java.io.BufferedReader;
00004: import java.io.IOException;
00005: import java.io.InputStreamReader;
00006: import java.io.PrintStream;
00007: import java.net.ServerSocket;
00008: import java.net.Socket;
00009:
00010: /**
00011:  *
00012:  * @author vegard
00013:  */
00014: public class Server extends Thread {
00015:
00016:     private Socket csocket;
00017:     private PrintStream printStream;
00018:     private ServerSocket ssocket;
00019:     private int guiCommand;
00020:     private String data;
00021:     private float headingReference;
00022:     private double[] remoteCommand;
00023:     private float controllerGain;
00024:     private int id;
00025:     private int northIncRequest;
00026:     private int eastIncRequest;
00027:     private boolean gainChanged;
00028:     private boolean available;
00029:     private boolean stop;
00030:
00031:     public Server(ServerSocket ssocket) {
00032:
00033:         remoteCommand = new double[3];
```

```
00034:         gainChanged = false;
00035:         stop = false;
00036:         this.ssocket = ssocket;
00037:     }
00038:
00039:     public void acceptConnection() throws IOException {
00040:         System.out.println("SERVER LISTENING");
00041:         csocket = ssocket.accept();
00042:         System.out.println("SERVER ACCEPTED");
00043:     }
00044:
00045:     public synchronized int getGuiCommand() {
00046:         return guiCommand;
00047:     }
00048:
00049:     public synchronized boolean isGainChanged() {
00050:         return gainChanged;
00051:     }
00052:
00053:     private synchronized void setGuiCommand(int guiCommand) {
00054:         this.guiCommand = guiCommand;
00055:     }
00056:
00057:     public synchronized float getHeadingReference() {
00058:         return headingReference;
00059:     }
00060:
00061:     private synchronized void setHeadingReference(float headingReference) {
00062:         this.headingReference = headingReference;
00063:     }
00064:
00065:     public synchronized float getControllerGain() {
00066:         gainChanged = false;
```

```
00067:         return controllerGain;
00068:     }
00069:
00070:     public synchronized void setControllerGain(int id, float controllerGain) {
00071:         if (id != 0) {
00072:             gainChanged = true;
00073:             this.id = id;
00074:             this.controllerGain = controllerGain;
00075:         }
00076:     }
00077:
00078:     public synchronized int getGainChanged() {
00079:         return id;
00080:     }
00081:
00082:     public synchronized int getNorthIncDecRequest() {
00083:         return northIncRequest;
00084:     }
00085:
00086:     public synchronized int getEastIncDecRequest() {
00087:         return eastIncRequest;
00088:     }
00089:
00090:     public synchronized void setNorthIncDecRequest(int request) {
00091:         northIncRequest = request;
00092:     }
00093:
00094:     public synchronized void setEastIncDecRequest(int request) {
00095:         eastIncRequest = request;
00096:     }
00097:
00098:     private void setRemoteCommand(String[] lineData) {
00099:         for (int i = 0; i < 3; i++) {
```

```
00100:             remoteCommand[i] = Double.parseDouble(lineData[i + 3]);
00101:         }
00102:     }
00103:
00104:     public synchronized double[] getRemoteCommand() {
00105:
00106:         return remoteCommand;
00107:     }
00108:
00109:
00110:     @Override
00111:     public void run() {
00112:         try {
00113:             printStream = new PrintStream(csocket.getOutputStream(), true);
00114:             while (csocket.isConnected() && !stop) {
00115:                 BufferedReader r = new BufferedReader(
00116:                         new InputStreamReader(csocket.getInputStream()));
00117:                 String line = r.readLine();//linjen for mottatt data
00118:                 String[] lineData;
00119:                 // System.out.println(line);
00120:                 if (line !=null && !line.isEmpty()) {
00121:                     //System.out.println("Server received data");
00122:                    lineData = line.split(" ");
00123:                     setGuiCommand(Integer.parseInt(lineData[0]));
00124:                     if (guiCommand == 2) {
00125:                         //setter denne linjen dersom det er fjernstyring
00126:                         setRemoteCommand(lineData);
00127:                     } else {
00128:                         //setter variablene i synkroniserte metoder
00129:                         setHeadingReference(Float.parseFloat(lineData[2]));
00130:                         setControllerGain(Integer.parseInt(lineData[1]),
00131:                                 Float.parseFloat(lineData[3]));
00132:                         setNorthIncDecRequest(Integer.parseInt(lineData[4]));
```

```
00133:                         setEastIncDecRequest(Integer.parseInt(lineData[5]));
00134:                     }
00135: 
00136:                 }
00137:                 //returnerer data til klient
00138:                 printStream.println(getDataFields());
00139:             }
00140:             printStream.close();
00141:             csocket.close();
00142:             System.out.println("Server thread run() exit:");
00143:         } catch (IOException ex) {
00144:             System.out.println("IO ex server");
00145: 
00146:         }
00147:     }
00148: /**
00149:  * setter linjen som skal sendes
00150:  * @param data
00151:  */
00152:     public synchronized void setDataFields(String data) {
00153:         //hindret for mye inkrementering ved ett klikk i GUI
00154:         while (available) {
00155:             try {
00156:                 wait();
00157:             } catch (InterruptedException ex) {
00158:             }
00159:         }
00160:         notifyAll();
00161:         available = true;
00162:         this.data = data;
00163:     }
00164: /**
00165:  * henter linjen med data som skal sendes
```

```
00166:  * @return
00167:  */
00168:     private synchronized String getDataFields() {
00169:         while (!available) {
00170:             try {
00171:                 wait();
00172:             } catch (InterruptedException ex) {
00173:             }
00174:         }
00175:         notifyAll();
00176:         available = false;
00177:         return data;
00178:     }
00179: 
00180:     public void stopThread() {
00181:         stop = true;
00182:     }
00183: 
00184:     public boolean isClosed() {
00185:         return csocket.isClosed();
00186:     }
00187: }
```

```
00001: /*
00002:  * To change this license header, choose License Headers in Project Properties.
00003:  * To change this template file, choose Tools | Templates
00004:  * and open the template in the editor.
00005:  */
00006: package USVProsjekt;
00007:
00008: import com.joptimizer.functions.ConvexMultivariateRealFunction;
00009: import com.joptimizer.functions.LinearMultivariateRealFunction;
00010: import com.joptimizer.functions.PDQuadraticMultivariateRealFunction;
00011: import com.joptimizer.optimizers.JOptimizer;
00012: import com.joptimizer.optimizers.OptimizationRequest;
00013: import com.joptimizer.optimizers.OptimizationResponse;
00014: import org.apache.commons.math3.linear.ArrayRealVector;
00015: import org.apache.commons.math3.linear.BlockRealMatrix;
00016: import org.apache.commons.math3.linear.RealVector;
00017:
00018: /**
00019:  *
00020:  * @author vegard
00021:  *
00022:  */
00023: public class ThrustAllocator {
00024:
00025:     private double[][] Phi;
00026:     private double[][] A1;
00027:     private BlockRealMatrix A2;
00028:     private BlockRealMatrix C2;
00029:     private ArrayRealVector p;
00030:     private PDQuadraticMultivariateRealFunction objectiveFunction;
00031:     ConvexMultivariateRealFunction[] inequalities;
00032:     private JOptimizer opt;
00033:     // NOTE: Lx1 er negativ, Lx2 er positiv, Ly1 er negativ, Ly2 er positiv
```

```
00034:     public ThrustAllocator(double Lx1, double Lx2, double Ly1, double Ly2) {
00035:         setUp(Lx1, Lx2, Ly1, Ly2);
00036:     }
00037:     /**
00038:      * KonstruktÃ¸r for Pioner 8 mini
00039:      */
00040:     public ThrustAllocator(){
00041:         setUp(-0.8, 0.87, -0.2, 0.2);
00042:     }
00043:
00044:     /*
00045:     Sett opp matriser og vektorer
00046:     */
00047:     private void setUp(double Lx1, double Lx2, double Ly1, double Ly2) {
00048:         Phi = new double[][]{{1., 0., 0., 0., 0., 0., 0.}, {0., 1., 0., 0., 0., 0., 0.,},
00049:         {0., 0., 1., 0., 0., 0., 0.}, {0., 0., 0., 1., 0., 0., 0.}, {0., 0., 0., 0., 10., 0., 0.},
00050:         {0., 0., 0., 0., 0., 10., 0.}, {0., 0., 0., 0., 0., 0., 10.}};
00051:
00052:         A1 = new double[][]{{1., 1., 0., 0., -1., 0., 0.}, {0., 0., -1., -1., 0., -1., 0.},
00053:         {-Ly1, -Ly2, -Lx1, -Lx2, 0., 0., -1.}};
00054:
00055:         A2 = new BlockRealMatrix(new double[][]{{-1., 0., 0., 0., 0., 0., 0.}, {0., -1., 0., 0., 0., 0., 0.},
00056:         {0., 0., -1., 0., 0., 0., 0.}, {0., 0., 0., -1., 0., 0., 0.}, {1., 0., 0., 0., 0., 0., 0.},
00057:         {0., 1., 0., 0., 0., 0., 0.}, {0., 0., 1., 0., 0., 0., 0.}, {0., 0., 0., 1., 0., 0., 0.}});
00058:
00059:         C2 = new BlockRealMatrix(new double[][]{{0., 0., 0., -1., 0., 0., 0., 0., 0., 0.}, {0., 0., 0., 0., -1., 0., 0., 0., 0., 0.},
00060:         {0., 0., 0., 0., 0., -1., 0., 0., 0., 0.}, {0., 0., 0., 0., 0., 0., -1., 0., 0., 0.},
00061:         {0., 0., 0., 0., 0., 0., 0., 1., 0., 0.}, {0., 0., 0., 0., 0., 0., 0., 0., 1., 0.},
00062:         {0., 0., 0., 0., 0., 0., 0., 0., 0., 1.}, {0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1.}});
00063:
00064:         p = new ArrayRealVector(new double[]{0., 0., 0., -29., -29., -29., -29., 34., 34., 34., 34.});
00065:
00066:         objectiveFunction = new PDQuadraticMultivariateRealFunction(Phi, null, 0);
```

```
00067:         inequalities = new ConvexMultivariateRealFunction[8];
00068:         opt = new JOptimizer();
00069:     }
00070:
00071:     /*
00072:     Beregn output basert pÃ¥ en Ã¸nsket kraftvektor
00073:     */
00074:     public double[] calculateOutput(double[] tau) throws Exception {
00075: //        long time1 = System.currentTimeMillis();
00076:         // Sett opp p-vektoren med Ã¸nskede verdier for tau (kraftvektor)
00077:         p.setEntry(0, tau[0]);
00078:         p.setEntry(1, tau[1]);
00079:         p.setEntry(2, tau[2]);
00080:
00081:
00082:         // Oppsett for ulikheten A2*z <= C2*p
00083:         RealVector v2 = C2.operate(p);
00084:
00085:
00086:         for (int i = 0; i < 8; i++) {
00087:             // Hver ulikhet settes opp som et LinearMultivariateRealFunction
00088:             inequalities[i] = new LinearMultivariateRealFunction(A2.getRow(i), -v2.toArray()[i]);
00089:         }
00090:         // Optimaliseringsproblemet
00091:         OptimizationRequest or = new OptimizationRequest();
00092:         // Sett objektivfunksjonen
00093:         or.setF0(objectiveFunction);
00094:
00095:         // Sett ulikheten
00096:         or.setFi(inequalities);
00097:
00098:         // Sett likheten
00099:         or.setA(A1);
```

```
00100:         or.setB(p.getSubVector(0, 3).toArray());
00101:
00102:         // Sett toleransen pÃ¥ resultatet. Lavere tall = stÃ¸rre nÃ¸yaktighet
00103:         or.setTolerance(1.E-1);
00104:
00105:         // Optimalisering
00106:
00107:         opt.setOptimizationRequest(or);
00108:         int returnCode = opt.optimize();
00109:
00110:         if (returnCode == OptimizationResponse.FAILED) {
00111:             System.out.println("Optimization FAIL");
00112:         }
00113: //        long time2 = System.currentTimeMillis()-time1;
00114:         //System.out.println("Tidsbruk = " + time2);
00115:         return opt.getOptimizationResponse().getSolution();
00116:
00117:     }
00118: }
```

```
00001: package USVProsjekt;
00002: 
00003: /**
00004:  *
00005:  * @author root
00006:  */
00007: public class ThrustWriter {
00008: 
00009:     private int pulseWidth1;
00010:     private int pulseWidth2;
00011:     private int pulseWidth3;
00012:     private int pulseWidth4;
00013: 
00014:     private SerialConnection serialConnection;
00015:     private Identifier ID;
00016: 
00017:     /**
00018:      * Klasse for Ã¥ konvertere newton til pwm og skrive verdier
00019:      *
00020:      * @param serialConnection
00021:      * @param ID
00022:      */
00023:     public ThrustWriter(SerialConnection serialConnection, Identifier ID) {
00024:         pulseWidth1 = 1500;
00025:         pulseWidth2 = 1500;
00026:         pulseWidth3 = 1500;
00027:         pulseWidth4 = 1500;
00028: 
00029:         this.serialConnection = serialConnection;
00030: 
00031:         this.ID = ID;
00032:         this.serialConnection.connect(this.ID);
00033:     }
```

```
00034: 
00035:     /**
00036:      * skriver verdier via SerialConnection
00037:      */
00038:     public void writeThrust() {
00039:         if (serialConnection.isConnected()) {
00040:             serialConnection.writeThrustMicros(pulseWidth1, pulseWidth2, pulseWidth3, pulseWidth4);
00041:         }
00042:     }
00043: 
00044:     /**
00045:      * setter pwm variablene
00046:      *
00047:      * @param newton
00048:      */
00049:     public void setThrustForAll(double[] newton) {
00050:         pulseWidth1 = newtonToPulseWidth(newton[0]);
00051:         pulseWidth2 = newtonToPulseWidth(newton[1]);
00052:         pulseWidth3 = newtonToPulseWidth(newton[2]);
00053:         pulseWidth4 = newtonToPulseWidth(newton[3]);
00054:     }
00055: 
00056:     /**
00057:      * konverterer newton til pwm
00058:      *
00059:      * @param xNewton
00060:      * @return
00061:      */
00062:     public int newtonToPulseWidth(double xNewton) {
00063:         double x = xNewton / 9.81; //From Newton to kgF because of regression
00064: 
00065:         int pulseWidth = 1500;
00066: 
```

```
00067:         if (xNewton > 0.0f) {
00068:             double pulseWidthFloat = 6.8044878418 * x * x * x - 45.9428509628 * x * x + 183.3806624031 * x + 1528.9249065579;
00069:             pulseWidth = (int) pulseWidthFloat;
00070:         } else if (xNewton <= 0.0f) {
00071:
00072:             double pulseWidthFloat = 8.1266850947 * x * x * x + 53.0777818795 * x * x + 206.7802157499 * x + 1466.368281074;
00073:             pulseWidth = (int) pulseWidthFloat;
00074:         } else {
00075:             pulseWidth = 1500;
00076:         }
00077:
00078:         if (pulseWidth < 1100) {
00079:             pulseWidth = 1100;
00080:         }
00081:         if (pulseWidth > 1900) {
00082:             pulseWidth = 1900;
00083:         }
00084:         return pulseWidth;
00085:     }
00086:
00087:     /**
00088:      * lukker seriell forbindelsen
00089:      */
00090:     void closeSerialConn() {
00091:         serialConnection.close();
00092:         System.out.println("ThrustWriter: Connection closed");
00093:     }
00094:
00095: }
```

```
00001: package USVProsjekt;
00002:
00003: /**
00004:  * Klasse for Ã¥ lese wind, temperatur og trykk sensorer
00005:  *
00006:  * @author root
00007:  */
00008: public class WindReader extends Thread {
00009:
00010:     private float windSpeed;
00011:     private float windDirection;
00012:     private float temperature;
00013:
00014:     private SerialConnection serialConnection;
00015:     private Identifier ID;
00016:     private int initPeriod;
00017:     private boolean stop;
00018:     private float airPressurehPa;
00019:
00020:     public WindReader(SerialConnection serialConnection, Identifier ID) {
00021:         windSpeed = 0.0f;
00022:         windDirection = 0.0f;
00023:         temperature = 0.0f;
00024:         this.serialConnection = serialConnection;
00025:         this.ID = ID;
00026:         stop = false;
00027:     }
00028:
00029:     @Override
00030:     public void run() {
00031:         String line;
00032:         //unngÃ¥r korrupte seriell setninger
00033:         while (initPeriod < 5 && serialConnection.isConnected()) {
```

```
00034:             line = serialConnection.getSerialLine();
00035:             initPeriod++;
00036:         }
00037:         while (serialConnection.isConnected() && !stop) {
00038:             line = serialConnection.getSerialLine();
00039:             parseWindSerialSentence(line);
00040:         }
00041:         serialConnection.close();
00042:         System.out.println("Connection lost/closed on Thread:"
00043:                 + " " + this.getName());
00044:     }
00045: 
00046:     /**
00047:      * kobler til sensor/mikrokontroller og lagrer data
00048:      */
00049:     public void connectToSerialPortAndDisplayWindInfo() {
00050:         serialConnection.connectAndListen(ID);
00051:     }
00052: 
00053:     /**
00054:      * henter data fra string setningen
00055:      *
00056:      * @param line
00057:      */
00058:     private void parseWindSerialSentence(String line) {
00059:         if (line.startsWith("&")) {
00060:             String[] lineData = line.split(" ");
00061:             windSpeed = Float.parseFloat(lineData[2]);
00062:             windDirection = Float.parseFloat(lineData[5]);
00063:             temperature = Float.parseFloat(lineData[7]);
00064:             airPressurehPa = Float.parseFloat(lineData[9]);
00065:         }
00066:     }
```

```
00067: 
00068:     /**
00069:      * getter for vind fart
00070:      *
00071:      * @return
00072:      */
00073:     public float getWindSpeed() {
00074:         return windSpeed;
00075:     }
00076: 
00077:     /**
00078:      * getter for vindretning
00079:      *
00080:      * @return
00081:      */
00082:     public float getWindDirection() {
00083:         return windDirection;
00084:     }
00085: 
00086:     /**
00087:      * getter for temperatur
00088:      *
00089:      * @return
00090:      */
00091:     public float getTemperature() {
00092:         return temperature;
00093:     }
00094: 
00095:     /**
00096:      * stopper trÃ¥den/gÃ¥r ut av run()
00097:      */
00098:     void stopThread() {
00099:         stop = true;
```

```
00100:     }
00101:
00102: }
```

Vedlegg 11
Kildekode Arduino

```
#include <Adafruit_GPS.h>
#include <SoftwareSerial.h>
SoftwareSerial mySerial(3, 2); //initialize software
serial port
Adafruit_GPS GPS(&mySerial); //create GPS object

String NMEA1;//Variable for first NMEA sentence
String NMEA2;
String response;

char c; //to read character coming from the GPS
String antennaString;
char antennaStatus;

void setup() {
  Serial.begin(115200); //Turn on serial monitor,
googlea earth real time uses 38400
  GPS.begin(9600); //Turn on GPS
  //GPS.sendCommand(PGCMD_NOANTENNA); //turn off
antenna update nuisance data
   GPS.sendCommand(PGCMD_NOANTENNA);
  GPS.sendCommand(PMTK_SET_NMEA_UPDATE_5HZ); //set
update rate to 5Hz
  GPS.sendCommand("$PMTK386,0*23"); //Nav speed
threshold is zero
   GPS.sendCommand(PMTK_SET_NMEA_OUTPUT_RMCGGA);
//Request RMC and GGA Sentences only
   GPS.sendCommand(PMTK_API_SET_FIX_CTL_5HZ);


  delay(1000);

}
```

```
void loop() {
  readGPS();
}

void readGPS() {
  while (!GPS.newNMEAreceived()) { //loop until you
have a good NMEA sentence
    c = GPS.read();
  }
  GPS.parse(GPS.lastNMEA()); //parse last good NMEA
sentece
  NMEA1 = GPS.lastNMEA();
  while (!GPS.newNMEAreceived()) { //loop until you
have a good NMEA sentence
    c = GPS.read();
  }
  GPS.parse(GPS.lastNMEA()); //parse last good NMEA
sentece
  NMEA2 = GPS.lastNMEA();

   Serial.println(NMEA1.substring(1));
   Serial.println(NMEA2.substring(1));
  Serial.println("");
}
```

```
/*****************************************************************
**
* Razor AHRS Firmware v1.4.2
* 9 Degree of Measurement Attitude and Heading
Reference System
* for Sparkfun "9DOF Razor IMU" (SEN-10125 and
SEN-10736)
* and "9DOF Sensor Stick" (SEN-10183, 10321 and
SEN-10724)
*
* Released under GNU GPL (General Public License) v3.0
* Copyright (C) 2013 Peter Bartz [http://ptrbrtz.net]
* Copyright (C) 2011-2012 Quality & Usability Lab,
Deutsche Telekom Laboratories, TU Berlin
*
* Infos, updates, bug reports, contributions and
feedback:
*      https://github.com/ptrbrtz/razor-9dof-ahrs
*
*
* History:
*    * Original code (http://code.google.
com/p/sf9domahrs/) by Doug Weibel and Jose Julio,
*      based on ArduIMU v1.5 by Jordi Munoz and William
Premerlani, Jose Julio and Doug Weibel. Thank you!
*
*    * Updated code (http://groups.google.
com/group/sf_9dof_ahrs_update) by David Malik
(david.zsolt.malik@gmail.com)
*      for new Sparkfun 9DOF Razor hardware (SEN-10125)
*
*    * Updated and extended by Peter Bartz
```

```
(peter-bartz@gmx.de):
*     * v1.3.0
*       * Cleaned up, streamlined and restructured
most of the code to make it more comprehensible.
*       * Added sensor calibration (improves precision
and responsiveness a lot!).
*       * Added binary yaw/pitch/roll output.
*       * Added basic serial command interface to set
output modes/calibrate sensors/synch stream/etc.
*       * Added support to synch automatically when
using Rovering Networks Bluetooth modules (and
compatible).
*       * Wrote new easier to use test program (using
Processing).
*       * Added support for new version of "9DOF Razor
IMU": SEN-10736.
*       --> The output of this code is not compatible
with the older versions!
*       --> A Processing sketch to test the tracker is
available.
*     * v1.3.1
*       * Initializing rotation matrix based on
start-up sensor readings -> orientation OK right away.
*       * Adjusted gyro low-pass filter and output
rate settings.
*     * v1.3.2
*       * Adapted code to work with new Arduino 1.0
(and older versions still).
*     * v1.3.3
*       * Improved synching.
*     * v1.4.0
*       * Added support for SparkFun "9DOF Sensor
Stick" (versions SEN-10183, SEN-10321 and SEN-10724).
```

```
*     * v1.4.1
*       * Added output modes to read raw and/or
calibrated sensor data in text or binary format.
*       * Added static magnetometer soft iron
distortion  compensation
*     * v1.4.2
*       * (No core firmware changes)
*
* TODOs:
*   * Allow optional use of EEPROM for storing and
reading calibration values.
*   * Use self-test and temperature-compensation
features of the sensors.
************************************************
************************************************
*/

/*
  "9DOF Razor IMU" hardware versions: SEN-10125 and
SEN-10736
  ATMega328@3.3V, 8MHz
  ADXL345  : Accelerometer
  HMC5843  : Magnetometer on SEN-10125
  HMC5883L : Magnetometer on SEN-10736
  ITG-3200 : Gyro
  Arduino IDE : Select board "Arduino Pro or Pro Mini
(3.3v, 8Mhz) w/ATmega328"
*/

/*
  "9DOF Sensor Stick" hardware versions: SEN-10183,
SEN-10321 and SEN-10724
  ADXL345  : Accelerometer
```

```
  HMC5843  : Magnetometer on SEN-10183 and SEN-10321
  HMC5883L : Magnetometer on SEN-10724
  ITG-3200 : Gyro
*/

/*
  Axis definition (differs from definition printed on
the board!):
    X axis pointing forward (towards the short edge
with the connector holes)
    Y axis pointing to the right
    and Z axis pointing down.

  Positive yaw   : clockwise
  Positive roll  : right wing down
  Positive pitch : nose up

  Transformation order: first yaw then pitch then roll
*/

/*
  Serial commands that the firmware understands:

  "#o<params>" - Set OUTPUT mode and parameters. The
available options are:

      // Streaming output
      "#o0" - DISABLE continuous streaming output.
Also see #f below.
      "#o1" - ENABLE continuous streaming output.

      // Angles output
      "#ob" - Output angles in BINARY format
```

(yaw/pitch/roll as binary float, so one output frame is 3x4 = 12 bytes long).

"#ot" - Output angles in TEXT format (Output frames have form like "#YPR=-142.28,-5.38,33.52", followed by carriage return and line feed [\r\n]).

// Sensor calibration

"#oc" - Go to CALIBRATION output mode.

"#on" - When in calibration mode, go on to calibrate NEXT sensor.

// Sensor data output

"#osct" - Output CALIBRATED SENSOR data of all 9 axes in TEXT format.
One frame consist of three lines - one for each sensor: acc, mag, gyr.

"#osrt" - Output RAW SENSOR data of all 9 axes in TEXT format.
One frame consist of three lines - one for each sensor: acc, mag, gyr.

"#osbt" - Output BOTH raw and calibrated SENSOR data of all 9 axes in TEXT format.
One frame consist of six lines - like #osrt and #osct combined (first RAW, then CALIBRATED).
NOTE: This is a lot of number-to-text conversion work for the little 8MHz chip on the Razor boards.
In fact it's too much and an output frame rate of 50Hz can not be maintained. #osbb.

"#oscb" - Output CALIBRATED SENSOR data of all 9 axes in BINARY format.
One frame consist of three 3x3 float

values = 36 bytes. Order is: acc x/y/z, mag x/y/z, gyr x/y/z.

"#osrb" - Output RAW SENSOR data of all 9 axes in BINARY format.

One frame consist of three 3x3 float values = 36 bytes. Order is: acc x/y/z, mag x/y/z, gyr x/y/z.

"#osbb" - Output BOTH raw and calibrated SENSOR data of all 9 axes in BINARY format.

One frame consist of 2x36 = 72 bytes - like #osrb and #oscb combined (first RAW, then CALIBRATED).

// Error message output

"#oe0" - Disable ERROR message output.

"#oe1" - Enable ERROR message output.

"#f" - Request one output frame - useful when continuous output is disabled and updates are required in larger intervals only. Though #f only requests one reply, replies are still bound to the internal 20ms (50Hz) time raster. So worst case delay that #f can add is 19.99ms

"#s<xy>" - Request synch token - useful to find out where the frame boundaries are in a continuous binary stream or to see if tracker is present and answering. The tracker will send "#SYNCH<xy>\r\n" in response (so it's possible to read using a readLine() function).

x and y are two mandatory but arbitrary bytes

```
that can be used to find out which request
          the answer belongs to.


  ("#C" and "#D" - Reserved for communication with
optional Bluetooth module.)

  Newline characters are not required. So you could
send "#ob#o1#s", which
  would set binary output mode, enable continuous
streaming output and request
  a synch token all at once.

  The status LED will be on if streaming output is
enabled and off otherwise.

  Byte order of binary output is little-endian: least
significant byte comes first.
*/



/*****************************************************
***********/
/*********** USER SETUP AREA! Set your options here!
*************/
/*****************************************************
***********/

// HARDWARE OPTIONS
/*****************************************************
***********/
// Select your hardware here by uncommenting one line!
```

```
//#define HW__VERSION_CODE 10125 // SparkFun "9DOF
Razor IMU" version "SEN-10125" (HMC5843 magnetometer)
#define HW__VERSION_CODE 10736 // SparkFun "9DOF Razor
IMU" version "SEN-10736" (HMC5883L magnetometer)
//#define HW__VERSION_CODE 10183 // SparkFun "9DOF
Sensor Stick" version "SEN-10183" (HMC5843
magnetometer)
//#define HW__VERSION_CODE 10321 // SparkFun "9DOF
Sensor Stick" version "SEN-10321" (HMC5843
magnetometer)
//#define HW__VERSION_CODE 10724 // SparkFun "9DOF
Sensor Stick" version "SEN-10724" (HMC5883L
magnetometer)


// OUTPUT OPTIONS
/*************************************************
**********/
// Set your serial port baud rate used to send out
data here!
#define OUTPUT__BAUD_RATE 57600

// Sensor data output interval in milliseconds
// This may not work, if faster than 20ms (=50Hz)
// Code is tuned for 20ms, so better leave it like tha
#define OUTPUT__DATA_INTERVAL 100  // in milliseconds

// Output mode definitions (do not change)
#define OUTPUT__MODE_CALIBRATE_SENSORS 0 // Outputs
sensor min/max values as text for manual calibration
#define OUTPUT__MODE_ANGLES 1 // Outputs
yaw/pitch/roll in degrees
#define OUTPUT__MODE_SENSORS_CALIB 2 // Outputs
```

```
calibrated sensor values for all 9 axes
#define OUTPUT__MODE_SENSORS_RAW 3 // Outputs raw
(uncalibrated) sensor values for all 9 axes
#define OUTPUT__MODE_SENSORS_BOTH 4 // Outputs
calibrated AND raw sensor values for all 9 axes
// Output format definitions (do not change)
#define OUTPUT__FORMAT_TEXT 0 // Outputs data as text
#define OUTPUT__FORMAT_BINARY 1 // Outputs data as
binary float

// Select your startup output mode and format here!
int output_mode = OUTPUT__MODE_ANGLES;
int output_format = OUTPUT__FORMAT_TEXT;

// Select if serial continuous streaming output is
enabled per default on startup.
#define OUTPUT__STARTUP_STREAM_ON true  // true or
false

// If set true, an error message will be output if we
fail to read sensor data.
// Message format: "!ERR: reading <sensor>", followed
by "\r\n".
boolean output_errors = false;  // true or false

// Bluetooth
// You can set this to true, if you have a Rovering
Networks Bluetooth Module attached.
// The connect/disconnect message prefix of the module
has to be set to "#".
// (Refer to manual, it can be set like this: SO,#)
// When using this, streaming output will only be
enabled as long as we're connected. That way
```

```
// receiver and sender are synchronzed easily just by
connecting/disconnecting.
// It is not necessary to set this! It just makes life
easier when writing code for
// the receiving side. The Processing test sketch also
works without setting this.
// NOTE: When using this, OUTPUT__STARTUP_STREAM_ON
has no effect!
#define OUTPUT__HAS_RN_BLUETOOTH false  // true or
false


// SENSOR CALIBRATION
/*****************************************************
**********/
// How to calibrate? Read the tutorial at
http://dev.qu.tu-berlin.de/projects/sf-razor-9dof-ahrs
// Put MIN/MAX and OFFSET readings for your board here
// Accelerometer
// "accel x,y,z (min/max) = X_MIN/X_MAX  Y_MIN/Y_MAX
Z_MIN/Z_MAX"
#define ACCEL_X_MIN ((float) -270)
#define ACCEL_X_MAX ((float) 262)
#define ACCEL_Y_MIN ((float) -263)
#define ACCEL_Y_MAX ((float) 271)
#define ACCEL_Z_MIN ((float) -229)
#define ACCEL_Z_MAX ((float) 233)

// Magnetometer (standard calibration mode)
// "magn x,y,z (min/max) = X_MIN/X_MAX  Y_MIN/Y_MAX
Z_MIN/Z_MAX"
#define MAGN_X_MIN ((float) -400)
#define MAGN_X_MAX ((float) 694)
```

```
#define MAGN_Y_MIN ((float) -580)
#define MAGN_Y_MAX ((float) 486)
#define MAGN_Z_MIN ((float) -479)
#define MAGN_Z_MAX ((float) 487)

// Magnetometer (extended calibration mode)
// Uncommend to use extended magnetometer calibration
(compensates hard & soft iron errors)
//#define CALIBRATION__MAGN_USE_EXTENDED true
//const float magn_ellipsoid_center[3] = {0, 0, 0};
//const float magn_ellipsoid_transform[3][3] = {{0, 0,
0}, {0, 0, 0}, {0, 0, 0}};

// Gyroscope
// "gyro x,y,z (current/average) = .../OFFSET_X  ...
/OFFSET_Y  .../OFFSET_Z
#define GYRO_AVERAGE_OFFSET_X ((float) 0.0)
#define GYRO_AVERAGE_OFFSET_Y ((float) 0.0)
#define GYRO_AVERAGE_OFFSET_Z ((float) 0.0)

/*
// Calibration example:
// "accel x,y,z (min/max) = -277.00/264.00
-256.00/278.00  -299.00/235.00"
#define ACCEL_X_MIN ((float) -277)
#define ACCEL_X_MAX ((float) 264)
#define ACCEL_Y_MIN ((float) -256)
#define ACCEL_Y_MAX ((float) 278)
#define ACCEL_Z_MIN ((float) -299)
#define ACCEL_Z_MAX ((float) 235)
// "magn x,y,z (min/max) = -511.00/581.00  -516.00/568
00  -489.00/486.00"
//#define MAGN_X_MIN ((float) -511)
```

```
//#define MAGN_X_MAX ((float) 581)
//#define MAGN_Y_MIN ((float) -516)
//#define MAGN_Y_MAX ((float) 568)
//#define MAGN_Z_MIN ((float) -489)
//#define MAGN_Z_MAX ((float) 486)
// Extended magn
#define CALIBRATION__MAGN_USE_EXTENDED true
const float magn_ellipsoid_center[3] = {91.5, -13.5,
-48.1};
const float magn_ellipsoid_transform[3][3] = {{0.902,
-0.00354, 0.000636}, {-0.00354, 0.9, -0.00599}, {0.
000636, -0.00599, 1}};
// Extended magn (with Sennheiser HD 485 headphones)
//#define CALIBRATION__MAGN_USE_EXTENDED true
//const float magn_ellipsoid_center[3] = {72.3360, 23.
0954, 53.6261};
//const float magn_ellipsoid_transform[3][3] = {{0.
879685, 0.000540833, -0.0106054}, {0.000540833, 0.
891086, -0.0130338}, {-0.0106054, -0.0130338, 0.
997494}};
//"gyro x,y,z (current/average) = -40.00/-42.05
98.00/96.20  -18.00/-18.36"
#define GYRO_AVERAGE_OFFSET_X ((float) -42.05)
#define GYRO_AVERAGE_OFFSET_Y ((float) 96.20)
#define GYRO_AVERAGE_OFFSET_Z ((float) -18.36)
*/


// DEBUG OPTIONS
/**********************************************
**********/
// When set to true, gyro drift correction will not be
applied
```

```
#define DEBUG__NO_DRIFT_CORRECTION false
// Print elapsed time after each I/O loop
#define DEBUG__PRINT_LOOP_TIME false


/*****************************************************
**********/
/******************* END OF USER SETUP AREA!
*********************/
/*****************************************************
**********/









// Check if hardware version code is defined
#ifndef HW__VERSION_CODE
  // Generate compile error
  #error YOU HAVE TO SELECT THE HARDWARE YOU ARE
USING! See "HARDWARE OPTIONS" in "USER SETUP AREA" at
top of Razor_AHRS.ino!
#endif

#include <Wire.h>

// Sensor calibration scale and offset values
#define ACCEL_X_OFFSET ((ACCEL_X_MIN + ACCEL_X_MAX) /
```

```
2.0f)
#define ACCEL_Y_OFFSET ((ACCEL_Y_MIN + ACCEL_Y_MAX) /
2.0f)
#define ACCEL_Z_OFFSET ((ACCEL_Z_MIN + ACCEL_Z_MAX) /
2.0f)
#define ACCEL_X_SCALE (GRAVITY / (ACCEL_X_MAX -
ACCEL_X_OFFSET))
#define ACCEL_Y_SCALE (GRAVITY / (ACCEL_Y_MAX -
ACCEL_Y_OFFSET))
#define ACCEL_Z_SCALE (GRAVITY / (ACCEL_Z_MAX -
ACCEL_Z_OFFSET))

#define MAGN_X_OFFSET ((MAGN_X_MIN + MAGN_X_MAX) / 2.
0f)
#define MAGN_Y_OFFSET ((MAGN_Y_MIN + MAGN_Y_MAX) / 2.
0f)
#define MAGN_Z_OFFSET ((MAGN_Z_MIN + MAGN_Z_MAX) / 2.
0f)
#define MAGN_X_SCALE (100.0f / (MAGN_X_MAX -
MAGN_X_OFFSET))
#define MAGN_Y_SCALE (100.0f / (MAGN_Y_MAX -
MAGN_Y_OFFSET))
#define MAGN_Z_SCALE (100.0f / (MAGN_Z_MAX -
MAGN_Z_OFFSET))



// Gain for gyroscope (ITG-3200)
#define GYRO_GAIN 0.06957 // Same gain on all axes
#define GYRO_SCALED_RAD(x) (x * TO_RAD(GYRO_GAIN)) //
Calculate the scaled gyro readings in radians per
second

// DCM parameters
```

```
#define Kp_ROLLPITCH 0.02f
#define Ki_ROLLPITCH 0.00002f
#define Kp_YAW 1.2f
#define Ki_YAW 0.00002f

// Stuff
#define STATUS_LED_PIN 13  // Pin number of status LED
#define GRAVITY 256.0f // "1G reference" used for DCM
filter and accelerometer calibration
#define TO_RAD(x) (x * 0.01745329252)  // *pi/180
#define TO_DEG(x) (x * 57.2957795131)  // *180/pi

// Sensor variables
float accel[3];  // Actually stores the NEGATED
acceleration (equals gravity, if board not moving).
float accel_min[3];
float accel_max[3];

float magnetom[3];
float magnetom_min[3];
float magnetom_max[3];
float magnetom_tmp[3];

float gyro[3];
float gyro_average[3];
int gyro_num_samples = 0;

// DCM variables
float MAG_Heading;
float Accel_Vector[3]= {0, 0, 0}; // Store the
acceleration in a vector
float Gyro_Vector[3]= {0, 0, 0}; // Store the gyros
turn rate in a vector
```

```
float Omega_Vector[3]= {0, 0, 0}; // Corrected
Gyro_Vector data
float Omega_P[3]= {0, 0, 0}; // Omega Proportional
correction
float Omega_I[3]= {0, 0, 0}; // Omega Integrator
float Omega[3]= {0, 0, 0};
float errorRollPitch[3] = {0, 0, 0};
float errorYaw[3] = {0, 0, 0};
float DCM_Matrix[3][3] = {{1, 0, 0}, {0, 1, 0}, {0, 0,
1}};
float Update_Matrix[3][3] = {{0, 1, 2}, {3, 4, 5}, {6,
7, 8}};
float Temporary_Matrix[3][3] = {{0, 0, 0}, {0, 0, 0},
{0, 0, 0}};

// Euler angles
float yaw;
float pitch;
float roll;

// DCM timing in the main loop
unsigned long timestamp;
unsigned long timestamp_old;
float G_Dt; // Integration time for DCM algorithm

// More output-state variables
boolean  output_stream_on;
boolean  output_single_on;
int curr_calibration_sensor = 0;
boolean reset_calibration_session_flag = true;
int num_accel_errors = 0;
int num_magn_errors = 0;
int num_gyro_errors = 0;
```

```
void read_sensors() {
  Read_Gyro(); // Read gyroscope
  Read_Accel(); // Read accelerometer
  Read_Magn(); // Read magnetometer
}

// Read every sensor and record a time stamp
// Init DCM with unfiltered orientation
// TODO re-init global vars?
void reset_sensor_fusion() {
  float temp1[3];
  float temp2[3];
  float xAxis[] = {1.0f, 0.0f, 0.0f};

  read_sensors();
  timestamp = millis();

  // GET PITCH
  // Using y-z-plane-component/x-component of gravity
vector
  pitch = -atan2(accel[0], sqrt(accel[1] * accel[1] +
accel[2] * accel[2]));

  // GET ROLL
  // Compensate pitch of gravity vector
  Vector_Cross_Product(temp1, accel, xAxis);
  Vector_Cross_Product(temp2, xAxis, temp1);
  // Normally using x-z-plane-component/y-component of
compensated gravity vector
  // roll = atan2(temp2[1], sqrt(temp2[0] * temp2[0] +
temp2[2] * temp2[2]));
  // Since we compensated for pitch,
```

```
x-z-plane-component equals z-component:
  roll = atan2(temp2[1], temp2[2]);

  // GET YAW
  Compass_Heading();
  yaw = MAG_Heading;

  // Init rotation matrix
  init_rotation_matrix(DCM_Matrix, yaw, pitch, roll);
}

// Apply calibration to raw sensor readings
void compensate_sensor_errors() {
    // Compensate accelerometer error
    accel[0] = (accel[0] - ACCEL_X_OFFSET) *
ACCEL_X_SCALE;
    accel[1] = (accel[1] - ACCEL_Y_OFFSET) *
ACCEL_Y_SCALE;
    accel[2] = (accel[2] - ACCEL_Z_OFFSET) *
ACCEL_Z_SCALE;

    // Compensate magnetometer error
#if CALIBRATION__MAGN_USE_EXTENDED == true
    for (int i = 0; i < 3; i++)
      magnetom_tmp[i] = magnetom[i] -
magn_ellipsoid_center[i];
     Matrix_Vector_Multiply(magn_ellipsoid_transform,
magnetom_tmp, magnetom);
#else
    magnetom[0] = (magnetom[0] - MAGN_X_OFFSET) *
MAGN_X_SCALE;
    magnetom[1] = (magnetom[1] - MAGN_Y_OFFSET) *
MAGN_Y_SCALE;
```

```
    magnetom[2] = (magnetom[2] - MAGN_Z_OFFSET) *
MAGN_Z_SCALE;
#endif

    // Compensate gyroscope error
    gyro[0] -= GYRO_AVERAGE_OFFSET_X;
    gyro[1] -= GYRO_AVERAGE_OFFSET_Y;
    gyro[2] -= GYRO_AVERAGE_OFFSET_Z;
}

// Reset calibration session if
reset_calibration_session_flag is set
void  check_reset_calibration_session()
{
  // Raw sensor values have to be read already, but no
error compensation applied

  // Reset this calibration session?
  if (!reset_calibration_session_flag) return;

  // Reset acc and mag calibration variables
  for (int i = 0; i < 3; i++) {
    accel_min[i] = accel_max[i] = accel[i];
    magnetom_min[i] = magnetom_max[i] = magnetom[i];
  }

  // Reset gyro calibration variables
  gyro_num_samples = 0;  // Reset gyro calibration
averaging
  gyro_average[0] = gyro_average[1] = gyro_average[2]
= 0.0f;

  reset_calibration_session_flag = false;
```

```
}

void  turn_output_stream_on()
{
  output_stream_on = true;
  digitalWrite(STATUS_LED_PIN, HIGH);
}

void  turn_output_stream_off()
{
  output_stream_on = false;
  digitalWrite(STATUS_LED_PIN, LOW);
}

// Blocks until another byte is available on serial
port
char  readChar()
{
  while (Serial.available() < 1) { } // Block
  return Serial.read();
}

void setup()
{
  // Init serial output
   Serial.begin(OUTPUT__BAUD_RATE);

  // Init status LED
  pinMode (STATUS_LED_PIN, OUTPUT);
  digitalWrite(STATUS_LED_PIN, LOW);

  // Init sensors
  delay(50);  // Give sensors enough time to start
```

```
  I2C_Init();
  Accel_Init();
  Magn_Init();
  Gyro_Init();

  // Read sensors, init DCM algorithm
  delay(20);  // Give sensors enough time to collect
data
   reset_sensor_fusion();

  // Init output
#if (OUTPUT__HAS_RN_BLUETOOTH == true) ||
(OUTPUT__STARTUP_STREAM_ON == false)
   turn_output_stream_off();
#else
   turn_output_stream_on();
#endif
}

// Main loop
void loop()
{
  // Read incoming control messages
  if (Serial.available() >= 2)
  {
    if (Serial.read() == '#') // Start of new control
message
    {
      int command = Serial.read(); // Commands
      if (command == 'f') // request one output _f_ram
        output_single_on = true;
      else if (command == 's') // _s_ynch request
      {
```

```
        // Read ID
        byte id[2];
        id[0] = readChar();
        id[1] = readChar();

        // Reply with synch message
        Serial.print("#SYNCH");
        Serial.write(id, 2);
        Serial.println();
      }
      else if (command == 'o') // Set _o_utput mode
      {
        char output_param = readChar();
        if (output_param == 'n')  // Calibrate _n_ext
sensor
        {
          curr_calibration_sensor =
(curr_calibration_sensor + 1) % 3;
          reset_calibration_session_flag = true;
        }
        else if (output_param == 't') // Output angles
as _t_ext
        {
          output_mode = OUTPUT__MODE_ANGLES;
          output_format = OUTPUT__FORMAT_TEXT;
        }
        else if (output_param == 'b') // Output angles
in _b_inary format
        {
          output_mode = OUTPUT__MODE_ANGLES;
          output_format = OUTPUT__FORMAT_BINARY;
        }
        else if (output_param == 'c') // Go to
```

```
_c_alibration mode
        {
          output_mode = OUTPUT__MODE_CALIBRATE_SENSORS
          reset_calibration_session_flag = true;
        }
        else if (output_param == 's') // Output
_s_ensor values
        {
          char values_param = readChar();
          char format_param = readChar();
          if (values_param == 'r')  // Output _r_aw
sensor values
            output_mode = OUTPUT__MODE_SENSORS_RAW;
          else if (values_param == 'c')  // Output
_c_alibrated sensor values
            output_mode = OUTPUT__MODE_SENSORS_CALIB;
          else if (values_param == 'b')  // Output
_b_oth sensor values (raw and calibrated)
            output_mode = OUTPUT__MODE_SENSORS_BOTH;

          if (format_param == 't') // Output values as
_t_ext
            output_format = OUTPUT__FORMAT_TEXT;
          else if (format_param == 'b') // Output
values in _b_inary format
            output_format = OUTPUT__FORMAT_BINARY;
        }
        else if (output_param == '0') // Disable
continuous streaming output
        {
          turn_output_stream_off();
          reset_calibration_session_flag = true;
        }
```

```
        else if (output_param == '1') // Enable
continuous streaming output
        {
          reset_calibration_session_flag = true;
          turn_output_stream_on();
        }
        else if (output_param == 'e') // _e_rror
output settings
        {
          char error_param = readChar();
          if (error_param == '0') output_errors =
false;
          else if (error_param == '1') output_errors =
true;
          else if (error_param == 'c') // get error
count
          {
            Serial.print("#AMG-ERR:");
            Serial.print(num_accel_errors); Serial.
print(",");
            Serial.print(num_magn_errors); Serial.
print(",");
            Serial.println(num_gyro_errors);
          }
        }
      }
#if OUTPUT__HAS_RN_BLUETOOTH == true
      // Read messages from bluetooth module
      // For this to work, the connect/disconnect
message prefix of the module has to be set to "#".
      else if (command == 'C') // Bluetooth "#CONNECT"
message (does the same as "#o1")
        turn_output_stream_on();
```

```
      else if (command == 'D') // Bluetooth
"#DISCONNECT" message (does the same as "#o0")
        turn_output_stream_off();
#endif // OUTPUT__HAS_RN_BLUETOOTH == true
    }
    else
    { } // Skip character
  }

  // Time to read the sensors again?
  if((millis() - timestamp) >= OUTPUT__DATA_INTERVAL)
  {
    timestamp_old = timestamp;
    timestamp = millis();
    if (timestamp > timestamp_old)
      G_Dt = (float) (timestamp - timestamp_old) /
1000.0f; // Real time of loop run. We use this on the
DCM algorithm (gyro integration time)
    else G_Dt = 0;

    // Update sensor readings
    read_sensors();

    if (output_mode ==
OUTPUT__MODE_CALIBRATE_SENSORS)  // We're in
calibration mode
    {
      check_reset_calibration_session();  // Check if
this session needs a reset
      if (output_stream_on || output_single_on)
output_calibration(curr_calibration_sensor);
    }
    else if (output_mode == OUTPUT__MODE_ANGLES)   //
```

```
Output angles
    {
      // Apply sensor calibration
      compensate_sensor_errors();

      // Run DCM algorithm
      Compass_Heading(); // Calculate magnetic heading
      Matrix_update();
      Normalize();
      Drift_correction();
      Euler_angles();

      if (output_stream_on || output_single_on)
output_angles();
    }
    else  // Output sensor values
    {
      if (output_stream_on || output_single_on)
output_sensors();
    }

    output_single_on = false;

#if DEBUG__PRINT_LOOP_TIME == true
    Serial.print("loop time (ms) = ");
    Serial.println(millis() - timestamp);
#endif
  }
#if DEBUG__PRINT_LOOP_TIME == true
  else
  {
    Serial.println("waiting...");
  }
```

```
#endif
}

/* This file is part of the Razor AHRS Firmware */

// DCM algorithm

/**************************************************/
void Normalize(void)
{
  float error=0;
  float temporary[3][3];
  float renorm=0;

  error= -Vector_Dot_Product(&DCM_Matrix[0][0],
&DCM_Matrix[1][0])*.5; //eq.19

  Vector_Scale(&temporary[0][0], &DCM_Matrix[1][0],
error); //eq.19
  Vector_Scale(&temporary[1][0], &DCM_Matrix[0][0],
error); //eq.19

  Vector_Add(&temporary[0][0], &temporary[0][0],
&DCM_Matrix[0][0]);//eq.19
  Vector_Add(&temporary[1][0], &temporary[1][0],
&DCM_Matrix[1][0]);//eq.19


Vector_Cross_Product(&temporary[2][0],&temporary[0][0],
&temporary[1][0]); // c= a x b //eq.20

  renorm= .5 *(3 - Vector_Dot_Product(&temporary[0][0]
&temporary[0][0])); //eq.21
```

```
  Vector_Scale(&DCM_Matrix[0][0], &temporary[0][0],
renorm);

  renorm= .5 *(3 - Vector_Dot_Product(&temporary[1][0]
&temporary[1][0])); //eq.21
  Vector_Scale(&DCM_Matrix[1][0], &temporary[1][0],
renorm);

  renorm= .5 *(3 - Vector_Dot_Product(&temporary[2][0]
&temporary[2][0])); //eq.21
  Vector_Scale(&DCM_Matrix[2][0], &temporary[2][0],
renorm);
}

/**************************************************/
void Drift_correction(void)
{
  float mag_heading_x;
  float mag_heading_y;
  float errorCourse;
  //Compensation the Roll, Pitch and Yaw drift.
  static float Scaled_Omega_P[3];
  static float Scaled_Omega_I[3];
  float Accel_magnitude;
  float Accel_weight;


  //*****Roll and Pitch***************

  // Calculate the magnitude of the accelerometer
vector
  Accel_magnitude =
sqrt(Accel_Vector[0]*Accel_Vector[0] +
```

```
Accel_Vector[1]*Accel_Vector[1] +
Accel_Vector[2]*Accel_Vector[2]);
  Accel_magnitude = Accel_magnitude / GRAVITY; //
Scale to gravity.
  // Dynamic weighting of accelerometer info
(reliability filter)
  // Weight for accelerometer info (<0.5G = 0.0, 1G =
1.0 , >1.5G = 0.0)
  Accel_weight = constrain(1 - 2*abs(1 -
Accel_magnitude),0,1);  //



Vector_Cross_Product(&errorRollPitch[0],&Accel_Vector[0
],&DCM_Matrix[2][0]); //adjust the ground of reference
   Vector_Scale(&Omega_P[0],&errorRollPitch[0],
Kp_ROLLPITCH*Accel_weight);

   Vector_Scale(&Scaled_Omega_I[0],&errorRollPitch[0],
Ki_ROLLPITCH*Accel_weight);
   Vector_Add(Omega_I,Omega_I,Scaled_Omega_I);

   //*****YAW***************
  // We make the gyro YAW drift correction based on
compass magnetic heading

  mag_heading_x = cos(MAG_Heading);
  mag_heading_y = sin(MAG_Heading);
   errorCourse=(DCM_Matrix[0][0]*mag_heading_y) -
(DCM_Matrix[1][0]*mag_heading_x);  //Calculating YAW
error
   Vector_Scale(errorYaw,&DCM_Matrix[2][0],
errorCourse); //Applys the yaw correction to the XYZ
rotation of the aircraft, depeding the position.
```

```
   Vector_Scale(&Scaled_Omega_P[0],&errorYaw[0],Kp_YAW)
//.01proportional of YAW.
   Vector_Add(Omega_P,Omega_P,Scaled_Omega_P);//Adding
Proportional.

   Vector_Scale(&Scaled_Omega_I[0],&errorYaw[0],Ki_YAW)
//.00001Integrator
   Vector_Add(Omega_I,Omega_I,Scaled_Omega_I);//adding
integrator to the Omega_I
}

void Matrix_update(void)
{
  Gyro_Vector[0]=GYRO_SCALED_RAD(gyro[0]); //gyro x
roll
  Gyro_Vector[1]=GYRO_SCALED_RAD(gyro[1]); //gyro y
pitch
  Gyro_Vector[2]=GYRO_SCALED_RAD(gyro[2]); //gyro z ya

  Accel_Vector[0]=accel[0];
  Accel_Vector[1]=accel[1];
  Accel_Vector[2]=accel[2];

  Vector_Add(&Omega[0], &Gyro_Vector[0],
&Omega_I[0]);  //adding proportional term
  Vector_Add(&Omega_Vector[0], &Omega[0],
&Omega_P[0]); //adding Integrator term

#if DEBUG__NO_DRIFT_CORRECTION == true // Do not use
drift correction
  Update_Matrix[0][0]=0;
   Update_Matrix[0][1]=-G_Dt*Gyro_Vector[2];//-z
```

```
   Update_Matrix[0][2]=G_Dt*Gyro_Vector[1];//y
   Update_Matrix[1][0]=G_Dt*Gyro_Vector[2];//z
  Update_Matrix[1][1]=0;
   Update_Matrix[1][2]=-G_Dt*Gyro_Vector[0];
   Update_Matrix[2][0]=-G_Dt*Gyro_Vector[1];
   Update_Matrix[2][1]=G_Dt*Gyro_Vector[0];
  Update_Matrix[2][2]=0;
#else // Use drift correction
  Update_Matrix[0][0]=0;
   Update_Matrix[0][1]=-G_Dt*Omega_Vector[2];//-z
   Update_Matrix[0][2]=G_Dt*Omega_Vector[1];//y
   Update_Matrix[1][0]=G_Dt*Omega_Vector[2];//z
  Update_Matrix[1][1]=0;
   Update_Matrix[1][2]=-G_Dt*Omega_Vector[0];//-x
   Update_Matrix[2][0]=-G_Dt*Omega_Vector[1];//-y
   Update_Matrix[2][1]=G_Dt*Omega_Vector[0];//x
  Update_Matrix[2][2]=0;
#endif

   Matrix_Multiply(DCM_Matrix,Update_Matrix,
Temporary_Matrix); //a*b=c

  for(int x=0; x<3; x++) //Matrix Addition (update)
  {
    for(int y=0; y<3; y++)
    {
       DCM_Matrix[x][y]+=Temporary_Matrix[x][y];
    }
  }
}

void  Euler_angles(void)
{
```

```
  pitch = -asin(DCM_Matrix[2][0]);
  roll = atan2(DCM_Matrix[2][1],DCM_Matrix[2][2]);
  yaw = atan2(DCM_Matrix[1][0],DCM_Matrix[0][0]);
}

/* This file is part of the Razor AHRS Firmware */

void Compass_Heading()
{
  float mag_x;
  float mag_y;
  float cos_roll;
  float sin_roll;
  float cos_pitch;
  float sin_pitch;

  cos_roll = cos(roll);
  sin_roll = sin(roll);
  cos_pitch = cos(pitch);
  sin_pitch = sin(pitch);

  // Tilt compensated magnetic field X
  mag_x = magnetom[0] * cos_pitch + magnetom[1] *
sin_roll * sin_pitch + magnetom[2] * cos_roll *
sin_pitch;
  // Tilt compensated magnetic field Y
  mag_y = magnetom[1] * cos_roll - magnetom[2] *
sin_roll;
  // Magnetic Heading
  MAG_Heading = atan2(-mag_y, mag_x);
}

/* This file is part of the Razor AHRS Firmware */
```

```
// Computes the dot product of two vectors
float Vector_Dot_Product(const float v1[3], const
float v2[3])
{
  float result = 0;

  for(int c = 0; c < 3; c++)
  {
    result += v1[c] * v2[c];
  }

  return result;
}

// Computes the cross product of two vectors
// out has to different from v1 and v2 (no in-place)!
void Vector_Cross_Product(float out[3], const float
v1[3], const float v2[3])
{
  out[0] = (v1[1] * v2[2]) - (v1[2] * v2[1]);
  out[1] = (v1[2] * v2[0]) - (v1[0] * v2[2]);
  out[2] = (v1[0] * v2[1]) - (v1[1] * v2[0]);
}

// Multiply the vector by a scalar
void Vector_Scale(float out[3], const float v[3],
float scale)
{
  for(int c = 0; c < 3; c++)
  {
    out[c] = v[c] * scale;
  }
```

```
}

// Adds two vectors
void Vector_Add(float out[3], const float v1[3], const
float v2[3])
{
  for(int c = 0; c < 3; c++)
  {
    out[c] = v1[c] + v2[c];
  }
}

// Multiply two 3x3 matrices: out = a * b
// out has to different from a and b (no in-place)!
void Matrix_Multiply(const float a[3][3], const float
b[3][3], float out[3][3])
{
  for(int x = 0; x < 3; x++)  // rows
  {
    for(int y = 0; y < 3; y++)  // columns
    {
      out[x][y] = a[x][0] * b[0][y] + a[x][1] *
b[1][y] + a[x][2] * b[2][y];
    }
  }
}

// Multiply 3x3 matrix with vector: out = a * b
// out has to different from b (no in-place)!
void Matrix_Vector_Multiply(const float a[3][3], const
float b[3], float out[3])
{
  for(int x = 0; x < 3; x++)
```

```
  {
    out[x] = a[x][0] * b[0] + a[x][1] * b[1] + a[x][2]
* b[2];
  }
}

// Init rotation matrix using euler angles
void init_rotation_matrix(float m[3][3], float yaw,
float pitch, float roll)
{
  float c1 = cos(roll);
  float s1 = sin(roll);
  float c2 = cos(pitch);
  float s2 = sin(pitch);
  float c3 = cos(yaw);
  float s3 = sin(yaw);

  // Euler angles, right-handed, intrinsic, XYZ
convention
  // (which means: rotate around body axes Z, Y', X'')
  m[0][0] = c2 * c3;
  m[0][1] = c3 * s1 * s2 - c1 * s3;
  m[0][2] = s1 * s3 + c1 * c3 * s2;

  m[1][0] = c2 * s3;
  m[1][1] = c1 * c3 + s1 * s2 * s3;
  m[1][2] = c1 * s2 * s3 - c3 * s1;

  m[2][0] = -s2;
  m[2][1] = c2 * s1;
  m[2][2] = c1 * c2;
}
```

```
/* This file is part of the Razor AHRS Firmware */

// Output angles: yaw, pitch, roll
void output_angles()
{
  if (output_format == OUTPUT__FORMAT_BINARY)
  {
    float ypr[3];
    ypr[0] = TO_DEG(yaw);
    ypr[1] = TO_DEG(pitch);
    ypr[2] = TO_DEG(roll);
    Serial.write((byte*) ypr, 12);  // No new-line
  }
  else if (output_format == OUTPUT__FORMAT_TEXT)
  {
    Serial.print("#YPR=");
    Serial.print(TO_DEG(yaw)); Serial.print(",");
    Serial.print(TO_DEG(pitch)); Serial.print(",");
    Serial.print(TO_DEG(roll)); Serial.println();
  }
}

void output_calibration(int calibration_sensor)
{
  if (calibration_sensor == 0)  // Accelerometer
  {
    // Output MIN/MAX values
    Serial.print("accel x,y,z (min/max) = ");
    for (int i = 0; i < 3; i++) {
      if (accel[i] < accel_min[i]) accel_min[i] =
accel[i];
      if (accel[i] > accel_max[i]) accel_max[i] =
accel[i];
```

```
      Serial.print(accel_min[i]);
      Serial.print("/");
      Serial.print(accel_max[i]);
      if (i < 2) Serial.print("  ");
      else Serial.println();
    }
  }
  else if (calibration_sensor == 1)  // Magnetometer
  {
    // Output MIN/MAX values
    Serial.print("magn x,y,z (min/max) = ");
    for (int i = 0; i < 3; i++) {
      if (magnetom[i] < magnetom_min[i])
magnetom_min[i] = magnetom[i];
      if (magnetom[i] > magnetom_max[i])
magnetom_max[i] = magnetom[i];
      Serial.print(magnetom_min[i]);
      Serial.print("/");
      Serial.print(magnetom_max[i]);
      if (i < 2) Serial.print("  ");
      else Serial.println();
    }
  }
  else if (calibration_sensor == 2)  // Gyroscope
  {
    // Average gyro values
    for (int i = 0; i < 3; i++)
      gyro_average[i] += gyro[i];
    gyro_num_samples++;

    // Output current and averaged gyroscope values
    Serial.print("gyro x,y,z (current/average) = ");
    for (int i = 0; i < 3; i++) {
```

```
      Serial.print(gyro[i]);
      Serial.print("/");
      Serial.print(gyro_average[i] / (float)
gyro_num_samples);
      if (i < 2) Serial.print("  ");
      else Serial.println();
    }
  }
}

void output_sensors_text(char raw_or_calibrated)
{
  Serial.print("#A-"); Serial.
print(raw_or_calibrated); Serial.print('=');
  Serial.print(accel[0]); Serial.print(",");
  Serial.print(accel[1]); Serial.print(",");
  Serial.print(accel[2]); Serial.println();

  Serial.print("#M-"); Serial.
print(raw_or_calibrated); Serial.print('=');
  Serial.print(magnetom[0]); Serial.print(",");
  Serial.print(magnetom[1]); Serial.print(",");
  Serial.print(magnetom[2]); Serial.println();

  Serial.print("#G-"); Serial.
print(raw_or_calibrated); Serial.print('=');
  Serial.print(gyro[0]); Serial.print(",");
  Serial.print(gyro[1]); Serial.print(",");
  Serial.print(gyro[2]); Serial.println();
}

void output_sensors_binary()
{
```

```
  Serial.write((byte*) accel, 12);
  Serial.write((byte*) magnetom, 12);
  Serial.write((byte*) gyro, 12);
}

void output_sensors()
{
  if (output_mode == OUTPUT__MODE_SENSORS_RAW)
  {
    if (output_format == OUTPUT__FORMAT_BINARY)
      output_sensors_binary();
    else if (output_format == OUTPUT__FORMAT_TEXT)
      output_sensors_text('R');
  }
  else if (output_mode == OUTPUT__MODE_SENSORS_CALIB)
  {
    // Apply sensor calibration
    compensate_sensor_errors();

    if (output_format == OUTPUT__FORMAT_BINARY)
      output_sensors_binary();
    else if (output_format == OUTPUT__FORMAT_TEXT)
      output_sensors_text('C');
  }
  else if (output_mode == OUTPUT__MODE_SENSORS_BOTH)
  {
    if (output_format == OUTPUT__FORMAT_BINARY)
    {
      output_sensors_binary();
      compensate_sensor_errors();
      output_sensors_binary();
    }
    else if (output_format == OUTPUT__FORMAT_TEXT)
```

```
    {
      output_sensors_text('R');
      compensate_sensor_errors();
      output_sensors_text('C');
    }
  }
}

/* This file is part of the Razor AHRS Firmware */

// I2C code to read the sensors

// Sensor I2C addresses
#define ACCEL_ADDRESS ((int16_t) 0x53) // 0x53 = 0xA6
/ 2
#define MAGN_ADDRESS  ((int16_t) 0x1E) // 0x1E = 0x3C
/ 2
#define GYRO_ADDRESS  ((int16_t) 0x68) // 0x68 = 0xD0
/ 2

// Arduino backward compatibility macros
#if ARDUINO >= 100
  #define WIRE_SEND(b) Wire.write((byte) b)
  #define WIRE_RECEIVE() Wire.read()
#else
  #define WIRE_SEND(b) Wire.send(b)
  #define WIRE_RECEIVE() Wire.receive()
#endif


void I2C_Init()
{
  Wire.begin();
```

```
}

void Accel_Init()
{
   Wire.beginTransmission(ACCEL_ADDRESS);
  WIRE_SEND(0x2D);  // Power register
  WIRE_SEND(0x08);  // Measurement mode
   Wire.endTransmission();
  delay(5);
   Wire.beginTransmission(ACCEL_ADDRESS);
  WIRE_SEND(0x31);  // Data format register
  WIRE_SEND(0x08);  // Set to full resolution
   Wire.endTransmission();
  delay(5);

  // Because our main loop runs at 50Hz we adjust the
output data rate to 50Hz (25Hz bandwidth)
   Wire.beginTransmission(ACCEL_ADDRESS);
  WIRE_SEND(0x2C);  // Rate
  WIRE_SEND(0x09);  // Set to 50Hz, normal operation
   Wire.endTransmission();
  delay(5);
}

// Reads x, y and z accelerometer registers
void Read_Accel()
{
  int i = 0;
  uint8_t buff[6];

   Wire.beginTransmission(ACCEL_ADDRESS);
  WIRE_SEND(0x32);  // Send address to read from
   Wire.endTransmission();
```

```
   Wire.beginTransmission(ACCEL_ADDRESS);
  Wire.requestFrom(ACCEL_ADDRESS, 6);  // Request 6
bytes
  while(Wire.available())  // ((Wire.
available())&&(i<6))
  {
    buff[i] = WIRE_RECEIVE();  // Read one byte
    i++;
  }
  Wire.endTransmission();

  if (i == 6)  // All bytes received?
  {
    // No multiply by -1 for coordinate system
transformation here, because of double negation:
    // We want the gravity vector, which is negated
acceleration vector.
    accel[0] = (int16_t)((((uint16_t) buff[3]) << 8) |
buff[2]);  // X axis (internal sensor y axis)
    accel[1] = (int16_t)((((uint16_t) buff[1]) << 8) |
buff[0]);  // Y axis (internal sensor x axis)
    accel[2] = (int16_t)((((uint16_t) buff[5]) << 8) |
buff[4]);  // Z axis (internal sensor z axis)
  }
  else
  {
    num_accel_errors++;
    if (output_errors) Serial.println("!ERR: reading
accelerometer");
  }
}
```

```
void Magn_Init()
{
   Wire.beginTransmission(MAGN_ADDRESS);
  WIRE_SEND(0x02);
  WIRE_SEND(0x00);  // Set continuous mode (default
10Hz)
   Wire.endTransmission();
  delay(5);

   Wire.beginTransmission(MAGN_ADDRESS);
  WIRE_SEND(0x00);
  WIRE_SEND(0b00011000);  // Set 50Hz
   Wire.endTransmission();
  delay(5);
}

void Read_Magn()
{
  int i = 0;
  uint8_t buff[6];

   Wire.beginTransmission(MAGN_ADDRESS);
  WIRE_SEND(0x03);  // Send address to read from
   Wire.endTransmission();

   Wire.beginTransmission(MAGN_ADDRESS);
  Wire.requestFrom(MAGN_ADDRESS, 6);  // Request 6
bytes
  while(Wire.available())   // ((Wire.
available())&&(i<6))
  {
    buff[i] = WIRE_RECEIVE();  // Read one byte
    i++;
```

```
  }
  Wire.endTransmission();

  if (i == 6)  // All bytes received?
  {
// 9DOF Razor IMU SEN-10125 using HMC5843 magnetometer
#if HW__VERSION_CODE == 10125
    // MSB byte first, then LSB; X, Y, Z
    magnetom[0] = -1 * (int16_t)(((((uint16_t)
buff[2]) << 8) | buff[3]));  // X axis (internal
sensor -y axis)
    magnetom[1] = -1 * (int16_t)(((((uint16_t)
buff[0]) << 8) | buff[1]));  // Y axis (internal
sensor -x axis)
    magnetom[2] = -1 * (int16_t)(((((uint16_t)
buff[4]) << 8) | buff[5]));  // Z axis (internal
sensor -z axis)
// 9DOF Razor IMU SEN-10736 using HMC5883L magnetomete
#elif HW__VERSION_CODE == 10736
    // MSB byte first, then LSB; Y and Z reversed: X,
Z, Y
    magnetom[0] = -1 * (int16_t)(((((uint16_t)
buff[4]) << 8) | buff[5]));  // X axis (internal
sensor -y axis)
    magnetom[1] = -1 * (int16_t)(((((uint16_t)
buff[0]) << 8) | buff[1]));  // Y axis (internal
sensor -x axis)
    magnetom[2] = -1 * (int16_t)(((((uint16_t)
buff[2]) << 8) | buff[3]));  // Z axis (internal
sensor -z axis)
// 9DOF Sensor Stick SEN-10183 and SEN-10321 using
HMC5843 magnetometer
#elif (HW__VERSION_CODE == 10183) || (HW__VERSION_CODE
```

```
== 10321)
    // MSB byte first, then LSB; X, Y, Z
    magnetom[0] = (((uint16_t) buff[0]) << 8) |
buff[1];          // X axis (internal sensor x axis)
    magnetom[1] = -1 * (int16_t)(((((uint16_t)
buff[2]) << 8) | buff[3]));  // Y axis (internal
sensor -y axis)
    magnetom[2] = -1 * (int16_t)(((((uint16_t)
buff[4]) << 8) | buff[5]));  // Z axis (internal
sensor -z axis)
// 9DOF Sensor Stick SEN-10724 using HMC5883L
magnetometer
#elif HW__VERSION_CODE == 10724
    // MSB byte first, then LSB; Y and Z reversed: X,
Z, Y
    magnetom[0] = (int16_t)((((uint16_t) buff[0]) <<
8) | buff[1]);          // X axis (internal sensor x
axis)
    magnetom[1] = -1 * (int16_t)(((((uint16_t)
buff[4]) << 8) | buff[5]));  // Y axis (internal
sensor -y axis)
    magnetom[2] = -1 * (int16_t)(((((uint16_t)
buff[2]) << 8) | buff[3]));  // Z axis (internal
sensor -z axis)
#endif
  }
  else
  {
    num_magn_errors++;
    if (output_errors) Serial.println("!ERR: reading
magnetometer");
  }
}
```

```
void Gyro_Init()
{
  // Power up reset defaults
   Wire.beginTransmission(GYRO_ADDRESS);
  WIRE_SEND(0x3E);
  WIRE_SEND(0x80);
  Wire.endTransmission();
  delay(5);

  // Select full-scale range of the gyro sensors
  // Set LP filter bandwidth to 42Hz
   Wire.beginTransmission(GYRO_ADDRESS);
  WIRE_SEND(0x16);
  WIRE_SEND(0x1B);  // DLPF_CFG = 3, FS_SEL = 3
  Wire.endTransmission();
  delay(5);

  // Set sample rato to 50Hz
   Wire.beginTransmission(GYRO_ADDRESS);
  WIRE_SEND(0x15);
  WIRE_SEND(0x0A);  //  SMPLRT_DIV = 10 (50Hz)
  Wire.endTransmission();
  delay(5);

  // Set clock to PLL with z gyro reference
   Wire.beginTransmission(GYRO_ADDRESS);
  WIRE_SEND(0x3E);
  WIRE_SEND(0x00);
  Wire.endTransmission();
  delay(5);
}
```

```
// Reads x, y and z gyroscope registers
void Read_Gyro()
{
  int i = 0;
  uint8_t buff[6];

   Wire.beginTransmission(GYRO_ADDRESS);
  WIRE_SEND(0x1D);  // Sends address to read from
   Wire.endTransmission();

   Wire.beginTransmission(GYRO_ADDRESS);
  Wire.requestFrom(GYRO_ADDRESS, 6);  // Request 6
bytes
  while(Wire.available())  // ((Wire.
available())&&(i<6))
  {
    buff[i] = WIRE_RECEIVE();  // Read one byte
    i++;
  }
  Wire.endTransmission();

  if (i == 6)  // All bytes received?
  {
    gyro[0] = -1 * (int16_t)((((uint16_t) buff[2]) <<
8) | buff[3]));    // X axis (internal sensor -y axis)
    gyro[1] = -1 * (int16_t)((((uint16_t) buff[0]) <<
8) | buff[1]));    // Y axis (internal sensor -x axis)
    gyro[2] = -1 * (int16_t)((((uint16_t) buff[4]) <<
8) | buff[5]));    // Z axis (internal sensor -z axis)
  }
  else
  {
    num_gyro_errors++;
```

```
    if (output_errors) Serial.println("!ERR: reading
gyroscope");
  }
}
```

```
/*
  Software serial multple serial test

 Receives from the hardware serial, sends to software
serial.
 Receives from software serial, sends to hardware
serial.

 The circuit:
 * RX is digital pin 10 (connect to TX of other device
 * TX is digital pin 11 (connect to RX of other device

 Note:
 Not all pins on the Mega and Mega 2560 support change
interrupts,
 so only the following can be used for RX:
 10, 11, 12, 13, 50, 51, 52, 53, 62, 63, 64, 65, 66,
67, 68, 69

 Not all pins on the Leonardo support change
interrupts,
 so only the following can be used for RX:
 8, 9, 10, 11, 14 (MISO), 15 (SCK), 16 (MOSI).

 created back in the mists of time
 modified 25 May 2012
 by Tom Igoe
 based on Mikal Hart's example

 This example code is in the public domain.

 */
```

```
#include <SoftwareSerial.h>

SoftwareSerial mySerial(10, 11); // RX, TX

void setup() {
  // Open serial communications and wait for port to
open:
  Serial.begin(57600);
  while (!Serial) {
    Serial.println("here");
    ; // wait for serial port to connect. Needed for
native USB port only
  }

  // set the data rate for the SoftwareSerial port
  mySerial.begin(57600);

}

void loop() { // run over and over
  if (mySerial.available()) {
    Serial.write(mySerial.read());
  }

  if (Serial.available()) {
    mySerial.write(Serial.read());
  }

}
```

```
#include <Servo.h>

int servoPins[] = {3, 5, 6, 9};
int outputPWM[4];
int stopSignal = 1500;
String thrusterCommands;

String line0;
String line1;
String line2;
String line3;
char c;

Servo servo[4];

void setup() {
  Serial.begin(115200);
  delay(30000);//unngår serielldata ved oppstart
  readBuffer();
  for (int i = 0; i < 4; i++) {
    outputPWM[i] = stopSignal; //stop/idle signal
    servo[i].attach(servoPins[i]);
    servo[i].writeMicroseconds(1500);
  }
  //give ESC time to start
  delay(2000);

}
void readBuffer(){
  while(Serial.available() >0){
    c = Serial.read();
  }
}
```

```
void loop() {
  if (Serial.available() > 0) {
    //read values from serial port
    //eksempel string "1600:1433:1700:1900:" (kolon
til slutt viktig)
    line0 = Serial.readStringUntil(':');
    line1 = Serial.readStringUntil(':');
    line2 = Serial.readStringUntil(':');
    line3 = Serial.readStringUntil(':');

    Serial.println(line0 + " " + line1 + " " + line2 +
" " + line3);

    outputPWM[0] = line0.toInt();
    outputPWM[1] = line1.toInt();
    outputPWM[2] = line2.toInt();
    outputPWM[3] = line3.toInt();
    //write values to thrusters ESC
    for (int i = 0; i < 4; i++) {
       servo[i].writeMicroseconds(outputPWM[i]);
    }
  }

}
```

```
#include <stdint.h>
#include "SparkFunBME280.h"
//Library allows either I2C or SPI, so
include both.
#include "Wire.h"
#include "SPI.h"

//Global sensor object
BME280 mySensor;

// diameter of anemometer
float diameter = 0.6985; //meters from
center pin to middle of cup
float metersPerSecond; //meters per
second
float speedPerClick = 0.3333; // meter
per second per half_revolution

int half_revolutions1 = 0;
int half_revolutions2 = 0;
int half_revolutions3 = 0;
unsigned long lastmillis1 = 0;
unsigned long lastmillis2 = 0;
unsigned long lastmillis3 = 0;
unsigned long lastmicros = 0;

int vaneInput = 0;
```

```
int inc = 1;



void setup() {
  Serial.begin(57600);

  //***Driver
settings*********************************/
/
  //commInterface can be I2C_MODE or
SPI_MODE
  //specify chipSelectPin using arduino
pin names
  //specify I2C address.  Can be
0x77(default) or 0x76

  //For I2C, enable the following and
disable the SPI section
  mySensor.settings.commInterface =
I2C_MODE;
  mySensor.settings.I2CAddress = 0x77;

  //***Operation
settings*****************************//
```

```
  //renMode can be:
  //  0, Sleep mode
  //  1 or 2, Forced mode
  //  3, Normal mode
  mySensor.settings.runMode = 3;
//Normal mode

  //tStandby can be:
  //  0, 0.5ms
  //  1, 62.5ms
  //  2, 125ms
  //  3, 250ms
  //  4, 500ms
  //  5, 1000ms
  //  6, 10ms
  //  7, 20ms
  mySensor.settings.tStandby = 0;

  //filter can be off or number of FIR
coefficients to use:
  //  0, filter off
  //  1, coefficients = 2
  //  2, coefficients = 4
  //  3, coefficients = 8
  //  4, coefficients = 16
  mySensor.settings.filter = 0;
```

```
  //tempOverSample can be:
  //  0, skipped
  //  1 through 5, oversampling *1, *2,
*4, *8, *16 respectively
  mySensor.settings.tempOverSample = 1;

  //pressOverSample can be:
  //  0, skipped
  //  1 through 5, oversampling *1, *2,
*4, *8, *16 respectively
  mySensor.settings.pressOverSample = 1;

  //humidOverSample can be:
  //  0, skipped
  //  1 through 5, oversampling *1, *2,
*4, *8, *16 respectively
  mySensor.settings.humidOverSample = 1;

  Serial.println(mySensor.begin(), HEX);

  pinMode(A0, INPUT);

attachInterrupt(digitalPinToInterrupt(3),
 rpm_fan, FALLING);
}

void loop() {
```

```
  vaneInput = analogRead(A0);

  if ((millis() - lastmillis1 >= 1000)
&& inc == 1) {
    metersPerSecond = speedPerClick *
half_revolutions1 / 3;
    half_revolutions1 = 0;
    lastmillis2 = millis();
    inc++;

printWindTempHumidity(metersPerSecond);
  }

  if ((millis() - lastmillis2 >= 1000)
&& inc == 2) {
    metersPerSecond = speedPerClick *
half_revolutions2 / 3;
    half_revolutions2 = 0;
    lastmillis3 = millis();
    inc++;

printWindTempHumidity(metersPerSecond);
  }

  if ((millis() - lastmillis3 >= 1000)
&& inc == 3) {
    metersPerSecond = speedPerClick *
```

```
half_revolutions3 / 3;
     half_revolutions3 = 0;
     lastmillis1 = millis();
     inc = 1;

printWindTempHumidity(metersPerSecond);
  }
}

// this code will be executed every time
the interrupt 0 (pin2) gets low.
void rpm_fan() {
  if (micros() - lastmicros >= 100) {
     lastmicros = micros();
     half_revolutions1++;
     half_revolutions2++;
     half_revolutions3++;
  }
}

float  lookupDegreesFromReading(int
input) {
  if (input >= 125 && input <= 135)
return 0;
  if (input >= 306 && input <= 318)
return 22.5;
  if (input >= 225 && input <= 235)
```

```
return 45;
  if (input >= 605 && input <= 615)
return 67.5;
  if (input >= 550 && input <= 560)
return 90;
  if (input >= 935 && input <= 940)
return 112.5;
  if (input >= 922 && input <= 934)
return 135;
  if (input >= 950 && input <= 960)
return 157.5;
  if (input >= 828 && input <= 840)
return 180;
  if (input >= 888 && input <= 900)
return 202.5;
  if (input >= 724 && input <= 736)
return 225;
  if (input >= 765 && input <= 780)
return 247.5;
  if (input >= 380 && input <= 393)
return 270;
  if (input >= 410 && input <= 425)
return 292.5;
  if (input >= 67 && input <= 80) return
315;
  if (input >= 184 && input <= 196)
return 337.5;
```

```
  return -1; //error
}

void printWindTempHumidity(float
metersPerSecond) {
  Serial.print("&Wind speed: ");
   Serial.print(metersPerSecond);
  Serial.print(" Wind direction(deg): ");
  Serial.
print(lookupDegreesFromReading(vaneInput)
);
  Serial.print(" Temperature(C): ");
  Serial.print(mySensor.readTempC(), 2);
  Serial.print(" Pressure(kPa): ");
  Serial.println((mySensor.
readFloatPressure())/100, 1);
}
```