Designspecifikation
Remotely Operated Underwater Vehicle

Version 1.1

Författare: Patricia Sundin
Datum: 16 oktober 2012

Status

Granskad	Alla	04/10/2012
Godkänd	Isak Nielsen	05/10/2012

Kursnamn:	Reglerteknisk projektkurs, CDIO	E-mail:	tsrt10_rov@googlegroups.com
Projektgrupp:	ROV	Dokumentansvarig:	Patricia Sundin
Kurskod:	TSRT10	Författarens e-mail:	patsu498@student.liu.se
Projekt:	Underwater ROV	Dokumentnamn:	Designspecifikation

Projektidentitet

Grupp E-mail:	tsrt10_rov@googlegroups.com
Hemsida:	http://www.isy.liu.se/edu/projekt/reglerteknik/2012/rov/
Beställare:	Isak Nielsen, Avdelningen för Reglerteknik vid ISY, LiTH **Telefon:** +46(0)13-28 13 04, **E-mail:** isak.nielsen@liu.se
Kund:	Micael Derelöv, Saab Underwater Systems **Telefon:** +46(0)13-28 11 65 , **E-mail:** micael.derelov@liu.se
Kursansvarig:	Daniel Axehill, Avdelningen för Reglerteknik vid ISY, LiTH **Telefon:** +46(0)13-28 40 42, **E-mail:** daniel@isy.liu.se
Projektledare:	Emelie Nilsson
Handledare:	Jonas Linder, Avdelningen för Reglerteknik vid ISY, LiTH **Telefon:** +46(0)13-28 28 04 , **E-mail:** jonas.linder@liu.se

Gruppmedlemmar

Namn	Ansvar	Telefon	E-mail (@student.liu.se)
Emelie Nilsson (EN)	Projektledare	0704828489	emeni712
Alva Olsson (AO)	Kvalitetsansvarig	0768022695	alvol428
Christian Andersson Naesseth (CAN)	Designansvarig	0739802308	chrna575
Erik Bergman (EB)	Testansvarig	0735064402	eribe518
Joakim Zachrisson (JZ)	Utvecklingsansvarig mjukvara	0737772168	joaza772
Johan Andersson (JA)	Informationsansvarig	0768546491	johan607
Linus Envall (LE)	Utvecklingsansvarig hårdvara	0738052628	linen837
Patricia Sundin (PS)	Dokumentansvarig	0706944295	patsu498

Dokumenthistorik

Version	Datum	Utförda förändringar	Utförda av	Granskad
0.1	27/09/2012	Första utkastet	Alla	Alla
0.2	30/09/2012	Andra utkastet	Alla	Alla
0.3	02/10/2012	Tredje utkastet	Alla	Alla
0.4	04/10/2012	Fjärde utkastet	Alla	Alla
1.0	04/10/2012	Första versionen	Alla	Alla
1.1	09/10/2012	Version 1.1. Messages tillagt i Interface.	CAN, JZ	CAN

Innehåll

1 Inledning

Inom såväl civila som militära tillämpningar ökar intresset och behovet av autonoma farkoster som kan utföra uppdrag till sjöss, i luften och på land utan kontakt med en operatör. Exempel på uppgifter för en sådan farkost kan vara övervakning, räddningsuppdrag, kartering eller reparationsarbeten.

Syftet med detta projekt är att vidareutveckla en Remotely Operated (underwater) Vehicle (ROV) med ett robust reglersystem, samt införa nya sensorer för navigering för att komma närmare en helt autonom farkost. Projektet, som är en påbyggnad från tidigare års projekt där den fysiska konstruktionen av farkosten, modell, simuleringsverktyg samt viss reglering tagits fram, genomförs på Instutitionen för Systemteknik (ISY) på Linköpings Universitet (LiU) i samarbete med Saab Underwater Systems och Institutionen för Ekonomisk och Industriell utveckling (IEI).

I detta dokument ges en detaljerad beskrivning av ROV:n och dess delsystem. Det beskriver hur implementationen ska gå till för att uppfylla samtliga krav med prioritet 1 som finns listade i projektets kravspecifikation [1].

1.1 Definitioner

Arduino-kortet	Arduino Mega 2560, DEV-09949
AUV	Autonomous Underwater Vehicle
EKF	Extended Kalman Filter
GUI	Graphical User Interface
IEI	Instutitionen för Ekonomisk och Industriell utveckling
IMU	Inertial Measurement Unit
ISY	Instutitionen för Systemteknik
LiU	Linköpings Universitet
LQ	Linear Quadratic
MPC	Model Predictive Control
PC	Personal Computer
PWM	Pulse-Width Modulation
RAM	Random Access Memory
ROS	Robot Operating System
ROV	Remotely Operated (underwater) Vehicle
SONAR	Sound Navigation And Ranging
SSPC	Sea Scan PC
TCP/IP	Transmission Control Protocol / Internet Protocol
UDP	User Datagram Protocol
USB	Universal Serial Bus

2 Systemöversikt

Systemet ska vara uppbyggt så att framföring av ROV:n kan ske enligt följande:

- Manuell drift
- Stabiliserad drift
- Autonom drift

I manuell drift styrs ROV:ns motorer direkt via en Xbox-handkontroll kopplad till en extern PC. Reglersystemen är här inaktiva. I stabiliserad drift styrs ROV:n via handkontrollen och regulatorerna stabiliserar ROV:ns orientering. I autonom drift är stabiliseringen aktiv, och ROV:n styrs via kommandon, exempelvis att färdas till en viss position och orientering. Detta läge kan i framtiden komma att uppgraderas till att utföra mer komplexa uppdrag.

Hårdvarudelen av systemet ska bestå av en kontrollstation (extern PC) kopplad via en Ethernet-kabel till ROV:n, samt på själva ROV:n finns en intern PC och en sensor- och motorenhet. Figur 1 visar en bild på hur systemet är uppbyggt i hårdvara. All hårdvara beskrivs mer detaljerat i Kapitel 4.

Mjukvarudelen av systemet ska bestå av de fyra delsystemen *Reglersystem*, *Planering*, *Sensorfusion* och *Kommunikation*. Figur 2 visar en bild på hur mjukvaran är uppdelad i dess olika delsystem. Mjukvaran finns beskriven i mer detalj i Kapitel 3.

I Figur 1 och 2 visas även hur det huvudsakliga informationsflödet sker. Den röda streckprickade linjen i båda figurer illustrerar den hård- och mjukvara som finns på ROV:n. I Figur 2 illustreras mjukvarumoduler av helstreckade block samt hårdvara, inkluderat för att förtydliga informationsflödet, vilket representeras av de svarta streck-prickade linjerna. Nedan följer en mer detaljerad beskrivning av hårdvaruenheterna.

2.1 Intern PC

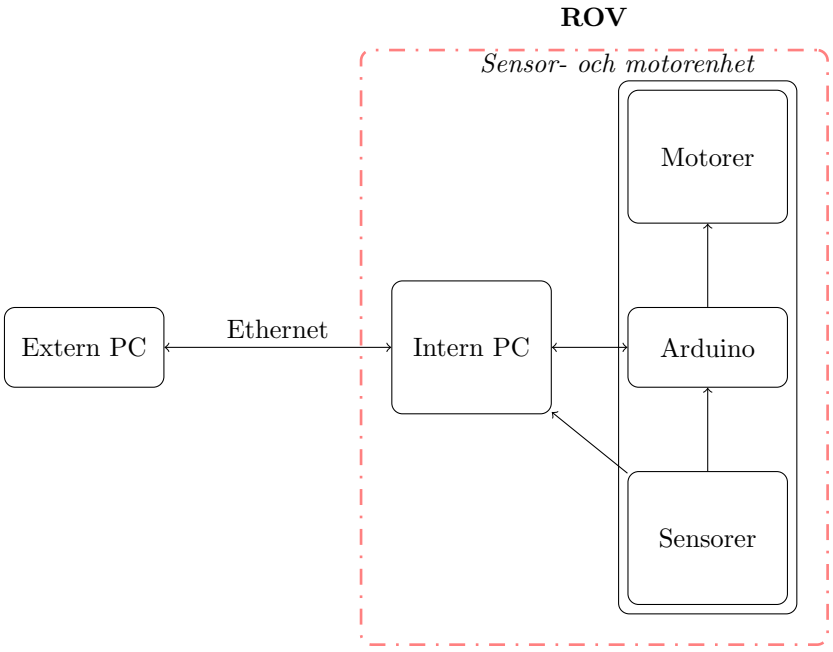
Denna enhet består av en dator som kör Robot Operating System (ROS). I denna enhet implementeras mjukvaran för de fyra delsystemen Reglersystem, Planering, Sensorfusion och Kommunikation. Dessa delsystem beskrivs i Kapitel 5, 6, 7 och 8. Hårdvaran beskrivs mer detaljerat i Kapitel 4. Samtlig kod som körs på den interna PC:n kommer att skrivas i C++.

2.1.1 ROS

I ROS delas systemet upp i *nodes*. Varje delsystem representeras av en node. Dessa nodes kommunicerar med varandra via *topics*. En node kan vara antingen *publisher* eller *subscriber* till en topic. Varje topic har en eller flera publishers respektive subscribers. En publisher är en node som skickar information genom att skriva ett *message* till en topic. Ett message är uppbyggt som en struct som kan innehålla flera olika datatyper, till exempel *integer*, *boolean* eller *floating point*. En node som behöver innehållet i en viss topic kallas för dess subscriber och kan hämta information genom att läsa från den.

2.2 Sensor- och motorenhet

Denna enhet ska kontrolleras av en Arduino Mega 2560, DEV-09949 (Arduino-kortet) som tar emot styrsignaler från regulatorn och omvandlar dessa till pulsbreddsmodulerade



Figur 1: Blockschema för hårdvaran i systemet där enheterna Extern PC, Intern PC samt Sensor- och motorenhet kan ses. Den röda streck-prickade linjen illustrerar den hård- och mjukvara som finns på ROV:n. Pilarna visar informationsflödet.

(*eng: Pulse-Width Modulation*) (PWM) signaler till motorerna. Signaler från sensorer skickas obehandlade vidare till sensorfusionssystemet. En Inertial Measurement Unit (IMU) är kopplad via USB direkt till den interna PC:n. Detta beskrivs mer detaljerat i Kapitel 4.

2.2.1 Motorer

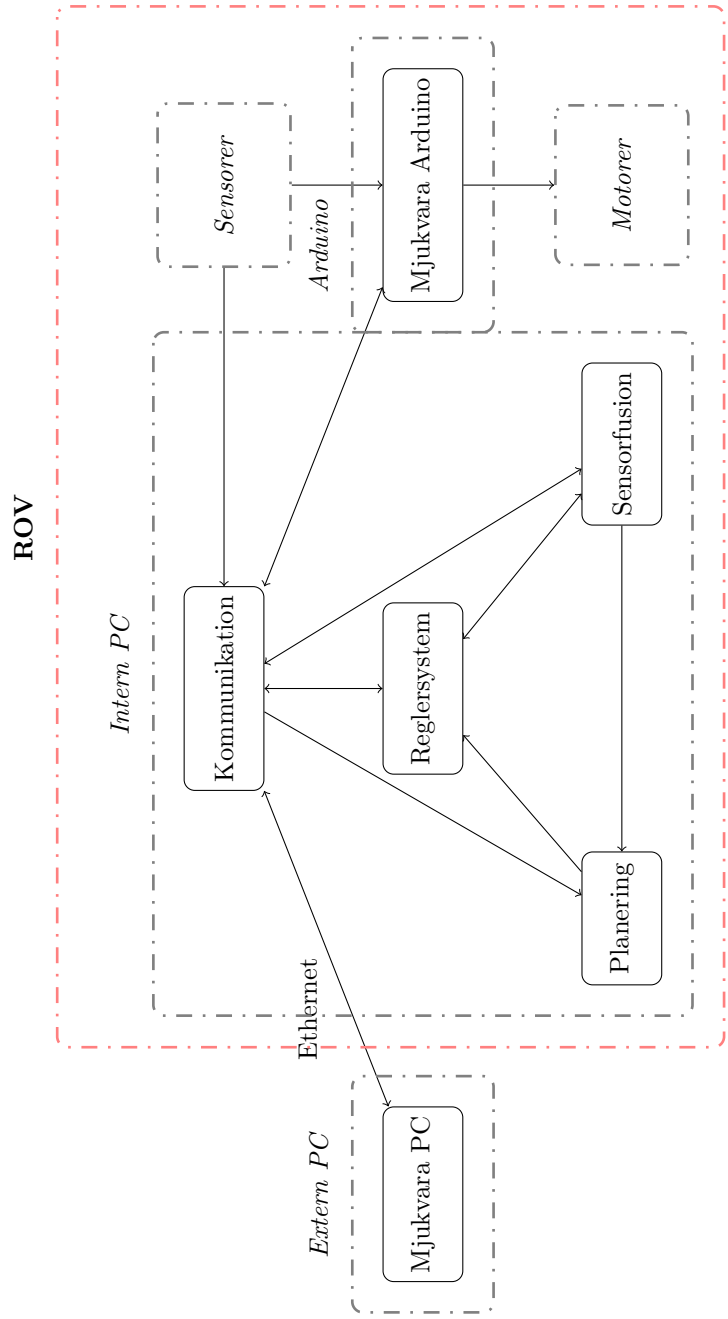
ROV:n har fem motorer som driver propellrar (se Figur 3 där motorerna är utmärkta i rött); en huvudmotor bak samt två mindre motorer vardera i y- och z-led. Detta medför att man kan styra fem av ROV:ns sex frihetsgrader separat. Dessa frihetsgrader innefattar x-, y- och z-position samt yaw- och pitchvinkel, men inte rollvinkel. Mer detaljerad information om motorernas mekanik finns att läsa i [3]. Hur samtliga koordinataxlar och vinklar är definierade illustreras i Figur 4.

2.2.2 Sensorer

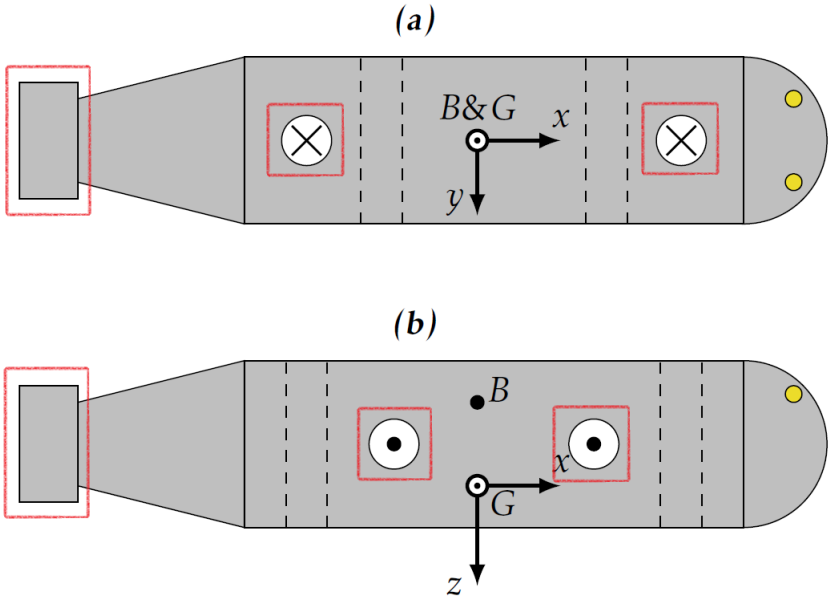
För navigation och tillståndsskattning skall information från en IMU, en trycksensor och SONAR-sensorer utnyttjas. IMU:n består av tre gyron, tre accelerometrar samt en magnetometer, som tillsammans med IMU:n och SONAR-sensorerna ska användas för att beräkna ROV:ns orientering och position. Trycksensorn ska användas för att estimera ROV:ns djup under vattenytan. Sensorerna beskrivs mer detaljerat i Kapitel 7.

2.3 Extern PC

Den externa PC:n ska vara en kontrollstation för ROV:n som kopplas med en Ethernetkabel till ROV:ns interna PC. Kommunikationen mellan den externa och den interna



Figur 2: Blockschema för mjukvaran i systemet. Intern PC innefattar de fyra delsystemen Kommunikation, Planering, Reglersystem och Sensorfusion. Mjukvara i den externa PC:n och Arduino-kortet visas också. Den röda streck-prickade linjen illustrerar den hård- och mjukvara som finns på ROV:n. Mjukvarumoduler illusteras av helstreckade block och hårdvara av de svarta streck-prickade linjerna. Pilarna visar informationsflödet.

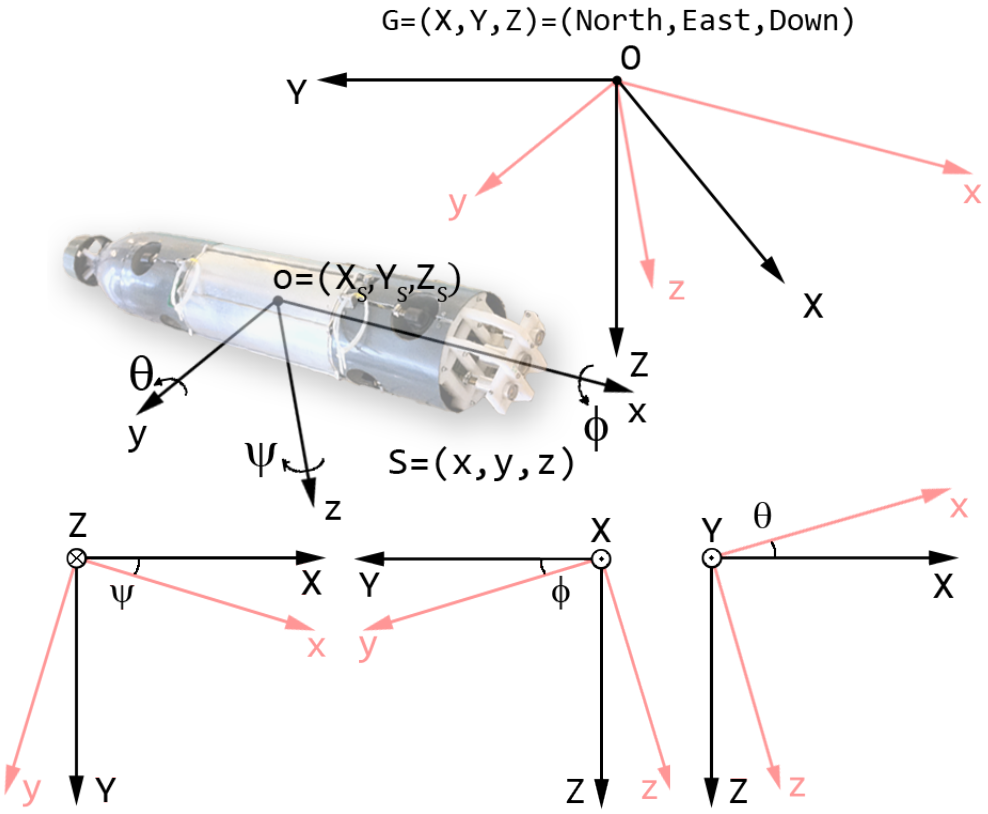


Figur 3: Schematisk bild av ROV:n och motorer sedd; (a) ovanifrån och (b) från sidan. Röda fyrkanter markerar motorernas position. G är tyngdpunkten, B markerar var flytkraften påverkar ROV:n.

PC:n sker via Transmission Control Protocol / Internet Protocol (TCP/IP). I PC:n ska det finnas ett grafiskt användargränssnitt där en användare har kontroll över ROV:n. Detta GUI finns beskrivet i Kapitel 9. Från den externa PC:n kan en användare även styra ROV:n med en Xbox-handkontroll, vilket vid manuell drift innebär att styra önskad rotationshastighet till motorerna, och vid stabiliserad drift kommer en användare att styra i vilken riktning ROV:n önskas färdas.

2.4 Koordinatsystem

Koordinatsystemen som används för att representera ROV:ns pose är ett ROV-fixt $S = (x, y, z)$ samt ett jordfixt $G = (X, Y, Z)$, där orienteringen definieras som vinklarna mellan det ROV-fixa och det jordfixa. Positionen är definierad som den punkt i G där ROV:ns tygndpunkt befinner sig. G är ett *North East Down* (NED)-system, vilket innebär att X är riktad mot den magnetiska nordpolen. För att underlätta kartrepresentationen kan ett till koordinatsystem $G' = (X', Y', Z)$ introduceras, som är roterat vinkeln ψ' i yaw-led relativt G, där ψ' är vinkeln mellan bassängen och nordlig riktning. Figur 4 illustrerar koordinatsystemen och ROV:ns orientering.



Figur 4: ROV:ns orientering relativt omgivningen. De röda axlarna föreställer ROV:ns koordinatsystem projicerat på omgivningens, för att illustrera hur orienteringen beskrivs genom de olika vinklarna.

3 Interface

I detta kapitel beskrivs interface mellan de olika delarna av ROV:n. I Avsnitt 3.1 beskrivs interface mellan de fyra delsystemen Reglersystem, Planering, Sensorfusion och Kommunikation. I Avsnitt 3.2 och 3.3 beskrivs interface mellan den interna PC:n och Arduino-kortet respektive den externa PC:n.

3.1 Intern PC

Delsystemen i den interna PC:n använder sig av topics och messages för att kommunicera mellan olika nodes. På den interna PC:n är varje delsystem en node. De nodes som finns, med tillhörande namn och kapitel, finns beskrivna i Tabell 1.

Tabell 1: Alla nodes

Namn	Motsvarande delsystem	Kapitel
control	Reglersystem	Kapitel 5
plan	Planering	Kapitel 6
sensor	Sensorfusion	Kapitel 7
com	Kommunikation	Kapitel 8

3.1.1 Topics

En topic är en typ av kommunikationsbuss i ROS som gör det möjligt för nodes att utbyta data. En topic har en eller flera publishers och en eller flera subscribers. Publishers och subscribers behöver ej vara medvetna om varandras existens. En topic har fördefinierade message-typer som anger strukturen på de messages som en publisher kan skicka. I Tabell 2 visas de topics som används, innehållet i relaterade messages samt publishers och subscribers.

Alla message består av varsin struct, som är uppbyggd enligt Tabell 4. Då vissa av dessa topics och messages är återanvända från tidigare projekt så följs ej namnkonventionen helt. En tabell som förklarar variabelnamn för kommunikation med externa enheter kan ses i Tabell 3. All topics har ett main message bestående av “msg_” + namnet på respektive topic minus “top_”.

3.2 Intern PC - Arduino

Kommunikation mellan den interna PC:n och Arduinon sker via seriell överföring med hjälp av ROS paketet *rosserial* [8].

3.3 Intern PC - Extern PC

Kommunikation mellan den externa PC:n och den interna PC:n sker via ROS:s transportlager *TCPROS* [7]. Detta transportlager använder sig av standard TCP/IP sockets för att transportera datameddelanden.

Tabell 2: Alla topics, innehållet i deras messages samt publishers och subscribers.

Topic	Messages	Publishers	Subscribers
top_Mode	Systemets status. Nödläge, manuell-, stabiliserad- eller autonom drift. LQ eller MPC.	com	control, plan, sensor
top_States	Systemets skattade tillstånd.	sensor	control, com, plan
top_ControlSig	Styrsignaler till motorer.	control	com, sensor
top_SensorData	Data från sensorer.	com	sensor
top_SetParam	Parametrar till LQ- och MPC-regulatorer samt parametrar för filtrering.	com	sensor, plan, control
top_Command	Styrkommandon till planeringen.	com	plan
top_Ref	Referenssignaler till regulatorerna.	plan	control
arduino	Batterinivåer, läckage- och tryckinformation från Arduino.	arduino	com
GUI/update	Motordata, batterinivåer och läckagesensor-info.	com	ros_gui
GUI/send	Intern PC redo att ta emot X-box signaler.	com	ros_gui
GUI/state	Starta/stoppa mainloopen.	ros_gui	com
imu/data	Data från IMU:n.	IMU	com
xbox	Data från X-box handkontroll.	ros_gui/Xbox	com

Tabell 3: Förklaring av vissa variabler i Tabell 4.

Variabel	Betydelse
bvx	Batterinivå för batteri nummer x $\in \{1, \ldots, 6\}$
leak	Data från läckagesensor
rx	Relä nummer x $\in \{1, 2, 3\}$ som styr ROV:ns strålkastare
uxy	Motordata $[-100, 100]$ från motor xy där x $\in$ {f, r} och y $\in$ {v, h}, f/r är front/rear och v/h är vertical/horisontal
utail	Motordata $[-100, 100]$ från bakre motor

Tabell 4: Beskrivning av alla messages.

Topic	Message	Innehåll
top_Mode	UInt8 (Status)	Ett heltal, Nödläge = 0, Manuell drift = 1, Stabiliserad drift = 2, Autonom drift = 3
	UInt8 (Controller)	Ett heltal, LQ = 0, MPC = 1
top_States	msg_Orientation	Två vektorer med tre flyttal, (ϕ, θ, ψ), $(\dot{\phi}, \dot{\theta}, \dot{\psi})$, stamped
	msg_Position	Två vektorer med tre flyttal, (x, y, z), $(\dot{x}, \dot{y}, \dot{z})$, stamped
top_ControlSig	msg_ControlSignals	En vektor med fem flyttal, $(u_{yf}, u_{yr}, u_{zf}, u_{zr}, u_T)$, stamped
top_SensorData	Imu	En kvaternion, 4 flyttal, (q_0, q_1, q_2, q_3), två vektorer med 3 flyttal $(\ddot{x}, \ddot{y}, \ddot{z})$, $(\dot{\phi}, \dot{\theta}, \dot{\psi})$, stamped
	msg_SONAR	Array av flyttal, stamped
	msg_Pressure	Ett heltal $(0 - 1023)$, stamped
	msg_Leak	Ett heltal (Int16), stamped
top_SetParam	msg_LQ	Matriser Q och R, stamped
	msg_MPC	Matriser av flyttal Q, R och P, tidshorisont (flyttal) T, stamped
	msg_EKF	Matriser Q (var(v_k)) och R (var(e_k)), stamped
top_Command	msg_Stab	Insignaler från Xbox, stamped
	msg_Auto	Två vektorer med tre flyttal, (x_1, y_1, z_1), $(\phi_1, \theta_1, \psi_1)$, stamped
top_Ref	msg_StabRef	Vektor med tre flyttal $(\dot{\theta}_{ref}, \dot{\psi}_{ref}, u_{Tref})$, stamped
	msg_AutoRef	Två vektorer med tre flyttal, (x_1, y_1, z_1), $(\phi_1, \theta_1, \psi_1)$, stamped
arduino	String	Sträng "bv1" + bv1 + "bv2" + bv2 + "bv3"+ bv3 + "bv4" + bv4 + "bv5" + bv5 + "bv6" + bv6 + "leakvolt" + leak + "pressure" + pressure + "end"
GUI/update	String	Sträng "r1 r2 r3 ufh ufv urh urv utail bv1 bv2 bv3 bv4 bv1 bv6 leak"
GUI/send	String	Sträng "send"
GUI/state	String	Sträng "go" eller "stop"
imu/data	sensor_msgs/Imu	3 st arrayer, var och en innehållande 3 st float64: orientering (grader), vinkelhastigheter (rad/s) och accelerationer (m/s^2) och kovarians för dessa storheter, stamped
xbox	String	Sträng "w1 v w2 v a1 v a2 v a3 v control v MPCon v relay v light v" där v är värdet för föregående variabel, som antingen är av typen double, bool eller int.

4 Hårdvara

Här beskrivs de hårdvarukomponenter som används i ROV:n och vilka komponenter som är mest intressanta för detta projekt, det vill säga de komponenter på vilka någon utveckling kommer att ske. En lista över de viktigaste komponenterna i ROV:n finns i Appendix A. Ett kopplingschema för den mesta hårdvaran finns att tillgå i [3]. De enda ändringar som kommer att ske är att SONAR-sensorer kopplas in; därför får detta kopplingsschema ses som aktuellt.

4.1 Intern PC

Den interna PC:n består av ett moderkort med inbyggd processor (Intel D525MW), Random Access Memory (RAM) (Kingston 4096MB (2x2GB) DDR3 PC3-8500 1066 MHz) och en hårddisk (OCZ 60 GB Vertex II E-series). På denna PC körs ROS och det är alltså här delsystemen Sensorfusion, Planering, Kommunikation och Reglering är implementerade.

4.2 Extern PC

Det kommer att finnas en extern PC i form av en laptop med operativsystemet Linux. Denna kan kopplas till ROV:ns interna PC via en Ethernet-kabel. I tidigare projekt har User Datagram Protocol (UDP) använts för dataöverföring, vilket inte fungerat önskvärt (problem med package loss uppstod). Därför har beslutet tagits att i detta projekt istället använda TCP/IP. Från denna externa PC kommer en användare kunna styra ROV:n, dels via ett grafiskt användargränssnitt, beskrivet i Kapitel 9 samt via en Xbox-handkontroll.

4.3 Sensor- och motorenhet

Denna enhet består av Arduino-kortet, som är anslutet till den interna PC:n via USB och har två uppgifter; att styra motorerna och att läsa in sensordata från en tryckgivare av modell PX2AN1XX100PAAAX från Honeywell, vilken är kopplad till en analog ingång på Arduino-kortet.

Kortet styr motorerna genom att läsa in styrsignaler från delsystemet reglersystem på den interna PC:n och omvandla dessa till PWM-signaler vilka skickas till motorernas styrkort.

Övriga sensorer innefattar en IMU och SONAR-sensorer, beskrivna i Kapitel 4.4. IMU:n är av modell MTi från Xsens och är kopplad via en USB-kabel till den interna PC:n. Vilka SONAR-sensorer som slutligen ska användas är ej bestämt i skrivande stund.

4.4 SONAR

Tre alternativ ska utvärderas och slutligen ska ett av dem väljas . De alternativ som finns att tillgå är följande:

- Delta T Imaging 837 från Imagenex
- Sea Scan PC (SSPC) Side Scan Sonar System från Marine Sonic Technology, Ltd
- Använda båda ovanstående

Det alternativ som framstår som bäst i skrivande stund är SSPC-sensorerna, då Delta T-sensorn troligtvis skulle göra ROV:n framtung. Detta skulle eventuellt påverka dynamiken

mer än vad som är att betrakta som försumbart. SONAR-sensorn från Imagenex ger som utsignal en bild, vilken används för att upptäcka föremål genom att jämföra bilder vid olika sampelögonblick och filtrera ut det som är konstant i tiden. SSPC-sensorerna fungerar i grova drag genom att ROV:n färdas över ett område och samlar in information och jämför med tidigare data.

Till SSPC-sensorerna hör ett kretskort, till vilket de är inkopplade. Eftersom sensorerna är av dubbelfrekvenstyp kopplas fyra signalkanaler in. Detta kort kommunicerar med den interna PC:n via RS232, och då behövs alltså en RS232/USB-adapter för att koppla in den på den interna PC:n. Den interna PC:n kan styra kortet genom att skicka kommandon med Host/Remote-protokollet från Marine Sonic. På kortet sitter en masslagringsenhet på vilken mätdata lagras i MSTIFF-filer. I dessa filer lagras navigeringsdata, exempelvis position. Denna information skickas också kontinuerligt till den interna PC:n.

Delta T-sensorn kommunicerar med den interna PC:n via TCP/IP med en Ethernetkabel. Denna kopplas längst fram på ROV:n vilket kan användas för att skatta position och hastighet i x-led.

5 Reglersystem

Detta kapitel beskriver vilka regleralgoritmer som ROV:n kommer att utnyttja och hur de fungerar samt reglersystemets interna kommunikation.

5.1 Översikt

ROV:n kommer att utnyttja två sorters regleralgoritmer för olika syften. Den kommer att ha en linjärkvadratisk (*eng: Linear Quadratic*) (LQ)-regulator för att bevisa att det är möjligt att kunna reglera ROV:n överhuvudtaget. Denna kommer senare användas som backup-regulator i den färdiga produkten. LQ-regulatorn är redan implementerad och simulerad men kan komma att behöva justeras för detta projekt. Den andra algoritmen är en MPC och det är den som den slutgiltiga produkten kommer att ha som huvudregulator för att uppfylla reglerkraven. Om reglerkraven inte går att uppfylla med en linjär MPC kommer en olinjär variant att användas. Detta på grund av att den vanligtvis beräkningstunga, olinjära MPC:n har inbyggt lösningsstöd i ACADO.

Båda reglersystemen kommer att använda sig av samma fysikaliska grundmodell på diskret tillståndsform,

$$x(k+1) = Fx(k) + Gu(k) \tag{1a}$$

$$y(k) = Cx(k) \tag{1b}$$

$$z(k) = Mx(k) \tag{1c}$$

där $x(k)$ är tillståndsvektorn, $u(k)$ styrsignalsvektorn, $y(k)$ mätsignalsvektorn och $z(k)$ reglerstorhetsvektorn. F,G,C och M är tillhörande matriser.

5.2 Reglersystemets kommunikation mellan delsystem

Reglersystemet kommer att ha kommunikation med delsystemen Planering, Kommunikation och Sensorfusion. Från kommunikationssystemet kommer den att få information om huruvida den ska använda LQ-reglering eller MPC. Reglersystemet kommer att få alla sina referensvärden från planeringen för att sedan ge utsignaler till ROV:ns motorer. Från delsystemet Sensorfusion kommer reglersystemet erhålla mätdata.

5.3 Linjärkvadratisk reglering

LQ-reglering bygger på att man vill reglera systemet med en avvägning mellan snabb reglering och små styrsignaler.

Detta kan översättas till följande ekvation

$$J_\infty(x(0)) = \min_{x,u} \sum_{k=0}^{\infty} (||x(k)||_{Q1}^2 + ||u(k)||_{Q2}^2), \tag{2}$$

där Q_1 och Q_2 är viktmatriser. Exempelvis gör ett stort Q_1 relativt Q_2 att regulatorn prioriterar att driva tillstånden till origo snabbt kontra små styrsignaler.

LQ-regleringen kommer även att ha tillgång till integralverkan för position, pitch- och yaw-vinklar för att motverka de stationära felen. För att implementera integralverkan kommer nya tillstånd att införas enligt (3).

$$x_I(k+1) = x_I(k) + r(k) - z(k) \tag{3}$$

Detta kommer att ge det utökade systemet följande utseende

$$\begin{bmatrix} x(k+1) \\ x_I(k+1) \end{bmatrix} = \begin{bmatrix} F & 0 \\ -M & I \end{bmatrix} \begin{bmatrix} x(k) \\ x_I(k) \end{bmatrix} + \begin{bmatrix} G \\ 0 \end{bmatrix} u(k) + \begin{bmatrix} 0 \\ I \end{bmatrix} r(k) \tag{4a}$$

$$z(k) = \begin{bmatrix} M & 0 \end{bmatrix} \begin{bmatrix} x(k) \\ x_I \end{bmatrix} \tag{4b}$$

som sedan optimeras enligt (2).

5.4 Modellbaserad prediktionsreglering

Ett problem med LQ-regleringen är att ROV:n har begränsade insignaler vilket LQ-regleringen inte tar hänsyn till. Att MPC kan hantera insignalsbegränsningar som bivillkor är en av MPC:ns största styrka.

MPC:n bygger likt LQ-reglering på att optimera över kvadratiska funktioner. Dock så har MPC:n en begränsad tidshorisont som måste sättas så pass lång så att den kan beskriva systemets dynamik.

Målfunktionen som beskriver optimeringsprobelemet för MPC:n beskrivs av

$$J_N(x(k)) = \sum_{j=0}^{N-1} ||x(k+j)||^2_{Q1} + ||u(k+j)||^2_{Q2} \tag{5}$$

Modellen ges enligt tidigare av (1). Rekursivt förfarande ger sedan de framtida tillstånden. I fallet med en tvåstegsprediktion fås

$$\begin{aligned} x(k+2) &= Fx(k+1) + Gu(k+1) \\ &= F^2x(k) + FGu(k) + Gu(k+1). \end{aligned} \tag{6}$$

För en lättarbetat struktur översätts det rekursiva skrivsättet till vektornotation vilket ger

$$U = \begin{bmatrix} u(k) \\ u(k+1) \\ \vdots \\ u(k+N-1) \end{bmatrix}, X = \begin{bmatrix} x(k) \\ x(k+1) \\ \vdots \\ x(k+N-1) \end{bmatrix} \tag{7}$$

$$\mathcal{F} = \begin{bmatrix} I \\ F \\ \vdots \\ F^{N-1} \end{bmatrix}, \mathcal{G} = \begin{bmatrix} 0 & 0 & 0 & ... & 0 \\ G & 0 & 0 & ... & 0 \\ FG & G & 0 & ... & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ F^{N-2}G & ... & FG & G & 0 \end{bmatrix} \tag{8}$$

där tillståndvektorn uttrycks av

$$X = \mathcal{F}x(k) + \mathcal{G}U \tag{9}$$

och viktmatriserna

$$\mathcal{Q}_1 = \begin{bmatrix} Q_1 & 0 & \dots & 0 \\ 0 & Q_1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & Q_1 \end{bmatrix}, \mathcal{Q}_2 = \begin{bmatrix} Q_2 & 0 & \dots & 0 \\ 0 & Q_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & Q_2 \end{bmatrix}. \tag{10}$$

ROV:n kommer dock att använda en MPC utökad med referensföljning och integralverkan. Den naturliga utvidgningen av prestandamåttet ges då av att istället för att straffa tillstånden, straffa skillnaden mellan reglerstorheten och referensvärdet, samt straffa ändringar i styrsignalen. Det prestandamåttet är

$$\sum_{j=0}^{N-1} ||z(k+j) - r(k+j)||^2_{Q1} + ||u(k+j) - u(k+j-1)||^2_{Q2}. \tag{11}$$

Analogt som för tillstånds- och styrsignalsvektorn skrivs även referenssignalen och reglerstorheten med vektornotation som

$$R = \begin{bmatrix} r(k) \\ r(k+1) \\ \vdots \\ r(k+N-1) \end{bmatrix}, \tag{12}$$

$$Z = \begin{bmatrix} Mx(k) \\ Mx(k+1) \\ \vdots \\ Mx(k+N-1) \end{bmatrix} = \begin{bmatrix} M & 0 & \dots & 0 \\ 0 & M & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & M \end{bmatrix} = \mathcal{M}X. \tag{13}$$

Vektorn med styrsignaler utökas till

$$\begin{bmatrix} u(k) - u(k-1) \\ u(k+1) - u(k) \\ \vdots \\ u(k+N-1) - u(k+N-2) \end{bmatrix} =$$

$$\begin{bmatrix} I & 0 & \dots & 0 \\ -I & I & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & -I & I \end{bmatrix} U - \begin{bmatrix} u(k-1) \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \Omega U - \delta. \tag{14}$$

Detta gör att prestandamåttet för en MPC med referensföljning och integralverkan kan uttryckas enligt (15).

$$(\mathcal{M}(\mathcal{F}x(k) + \mathcal{G}U) - R)^T \mathcal{Q}_1 (\mathcal{M}(\mathcal{F}x(k) + \mathcal{G}U) - R) + (\Omega U - \delta)^T \mathcal{Q}_2 (\Omega U - \delta) \tag{15}$$

Med denna notation kan sedan prestandamåttet identifieras som ett kvadratiskt optimeringsproblem enligt (16).

$$U^* = \underset{U}{\operatorname{argmin}} \frac{1}{2}(\mathcal{M}(\mathcal{F}x(k) + \mathcal{G}U) - R)^T \mathcal{Q}_1(\mathcal{M}(\mathcal{F}x(k) + \mathcal{G}U) - R) + (\Omega U - \delta)^T \mathcal{Q}_2(\Omega U - \delta) \tag{16}$$

$$\text{bivillkor } Au \leq b$$

Ur vektorn U^* tas sedan det första elementet $u^*(k)$ och används för pågående sampel.

Då detta optimeringsproblemet inte har en analytisk lösning löses det med hjälp av Matlabs ramverk för lösningar av kvadratiska optimeringsproblem. Sedan kommer denna Matlab-kod översättas till C-kod medelst ACADO.

För att översiktligt visa hur MPC:n i ROV:n fungerar finns följande algoritm:

while *true* **do**
  Mät $x(k)$
  Lös (16) för att räkna ut styrsignalsekvensen $u(\cdot)$.
  Använd första elementet $u(k)$ i styrsignalsekvensen under ett sampel.
  Tidsuppdatera $k := k + 1$.
end

6 Planering

Detta kapitel beskriver hur delsystemet planering ska kommunicera med andra delsystem samt hur en banplanering för att hitta kortaste vägen till en slutposition ska tas fram.

6.1 Kommunikation mellan delsystem

I första hand ska planeringen ta hand om styrkommandon som har skickats från den externa PC:n, via delsystem Kommunikation. Vid manuell drift och stabiliserad drift skickas dessa styrkommandon direkt vidare till delsystemet Reglersystem. Vid autonom drift ska planeringen kunna ta fram egna styrsignaler och skicka dessa vidare till delsystemet Reglersystem. Dessa olika lägen fås från den externa PC:n. Planeringen får även en karta över ROV:ns omgivning från delsystem Kommunikation. Denna karta behövs för att kunna ta fram en banplanering i det autonoma läget. Från delsystemet Sensorfusion fås information om skattade tillstånd.

6.2 Matematiska beräkningar för banplanering

Detta avsnitt förklarar hur banplanering ska beräknas.

6.2.1 Kortaste väg

För att hitta kortaste vägen till en position och för att nå en slutorientering, som är givet av den externa PC:n, ska ROV:n först roteras så att den är riktad mot slutpositionen, sedan ska den köra rakt mot slutpositionen. Väl vid slutpositionen ska ROV:n rotera så att den får rätt orientering. Planeringen behöver få en slutposition och slutorientering från den externa PC:n för att klara av denna uppgift samt en karta över omgivningen. Hur referenssignalerna ska tas fram för att nå slutpositionen och slutorienteringen beskrivs i detta delavsnitt.

ROV:n utgångsposition är (x_0, y_0, z_0) och utgångsorienteringen är (ψ_0, θ_0), då roll-vinkeln ej går att reglera finns inget utgångsvärde för denna vinkel. För att ROV:n ska komma till slutposition (x_1, y_1, z_1), ska den först roteras så att ROV:n kan köra rakt mot slutpositionen. ROV:n börjar med att ställa sig i rätt vinkel i yaw-led. Referensvinkeln i yaw-led ges av

$$\psi_{ref} = 90^\circ - \arctan \frac{x_1 - x_0}{y_1 - y_0} \tag{17}$$

Därefter beräknas referensvinkeln i pitch-led enligt

$$\theta_{ref} = \arctan \frac{z_1 - z_0}{y_1 - y_0}. \tag{18}$$

Då θ_{ref} och ψ_{ref} har beräknats, skickas dessa referenssignaler till delsystemet Reglersystem.

När ROV:n har erhållit önskad orientering, enligt de båda referenssignalerna ψ_{ref} och θ_{ref}, skickas en referenssignal som betyder att ska köra ROV:n rakt fram, samtidigt som orienteringen bibehålls. ROV:n stannar inte förrän dess position överensstämmer med slutpositionen med en felmarginal på maximalt 0.1 meter.

När slutpositionen har erhållits, roterar ROV:n för att nå slutorienteringen (ψ_1, θ_1). Dessa vinklar skickas som referenssignaler till reglersystemet, utan bearbetning av planeringen.

6.2.2 Snabbaste väg

Då all banplanering där en snabbaste väg till en position och orientering ska tas fram är av prioritet 2 i projektets kravspecifikation [1], så är en sådan algoritm ej beskriven i detta dokument.

6.2.3 Bruten kontakt

Om kontakten med den externa PC:n bryts ska planeringen ta fram en banplanering för att ROV:n ska komma upp till ytan. Denna planering innebär att referenspositionen är $z_1 = 0$ med bibehållna x- och y-positioner samt $\theta_{ref} = 0$. Referenssignaler skickas till reglersystemet så att ROV:n långsamt förflyttar sig till denna position.

6.3 Delsystemet i ROS

Detta delsystem kommer att implementeras i node *plan* i ROS. Mer om interface mellan olika nodes i Kapitel 3.

7 Sensorfusion

Detta kapitel beskriver teoretiskt lösningar och algoritmer som rör signalbehandling och skattningar av ROV:ns tillstånd.

7.1 Extended Kalman Filter

Eftersom en linjär modell av ROV:n troligtvis är otillräcklig kommer ett Extended Kalman Filter (EKF) att implementeras för att skatta dess tillstånd. Genom taylorutveckling av tillstånds- och sensormodellen och sedan implementation av Kalmanfiltret erhålls EKF. Beroende på om första eller andra ordningens taylorutvecklingar används erhålls antingen EKF1 eller EKF2. Enligt [2] användes EKF1 med tillfredställande resultat, så därför kommer EKF1 att användas i detta fall också. För att eventuellt kunna uppnå högre prestanda hos beräkningarna ska en *Square Root Implementation* av EKF1 undersökas. För vidare information om Kalmanfilter, EKF och Square Root Implementation, se Gustafsson 2010 [6].

7.1.1 Modell

Ett vanligt antagande är att anta att både både process- och mätbrus är additiva, och därför görs även detta antagande i detta fall. Tillstånds- och mätmodellen blir då

$$x_{k+1} = f(x_k, u_k) + v_k,\ Cov(v_k) = Q_k, \tag{19a}$$

$$y_k = h(x_k, u_k) + e_k,\ Cov(e_k) = R_k, \tag{19b}$$

7.1.2 Initiering

Innan första iterationen initieras tillståndsskattningen och kovariansen för felet i skattningen enligt

$$\hat{x}_{1|0} = E(x_0), \tag{20a}$$

$$P_{1|0} = Cov(x_0). \tag{20b}$$

7.1.3 Iterationer

Uppdatering av tillståndsskattning och kovarians för felet i skattningen givet en ny mätning ges enligt

while *true* **do**
Mätuppdatering:
$S_k = h'_x(\hat{x}_{k|k-1})P_{k|k-1}(h'_x(\hat{x}_{k|k-1}))^T + R_k$
$K_k = P_{k|k-1}(h'_x(\hat{x}_{k|k-1}))^T S_k^{-1}$
$\epsilon_k = y_k - h(\hat{x}_{k|k-1})$
Uppdatera skattning av tillståndsvektorn:
$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k\epsilon_k$
Uppdatera skattning av kovariansen för felet:
$P_{k|k} = P_{k|k-1} - P_{k|k-1}(h'_x(\hat{x}_{k|k-1}))^T S_k^{-1} h'_x(\hat{x}_{k|k-1})P_{k|k-1}$
Tidsuppdatera tillståndsvektorn:
$\hat{x}_{k+1|k} = f(\hat{x}_{k|k})$
Tidsuppdatera kovariansmatrisen:
$P_{k+1|k} = f'_x(\hat{x}_{k|k})P_{k|k}(f'_x(\hat{x}_{k|k}))^T + Q_k$
end

7.2 Sensorerna

Här modelleras hur mätningar från de olika sensorerna ser ut.

7.2.1 IMU

Med hjälp av en IMU kan ROV:ns orientering skattas. IMU:n är en enhet som är utrustad med accelerometrar, magnetometer och gyroskop som mäter resulterande acceleration och resulterande magnetfältstyrka i tre ortogonala riktningar samt vinkelhastigheter runt var och en av axlarna i dessa riktningar. Detta kommer göra det möjligt att mäta ROV:ns orientering i bassängen med stor noggrannhet. Dock kommer det med endast en IMU vara svårt att skatta hastighet och position då dessa skattningar kommer driva och felet kommer därmed öka snabbt.

IMU:n som ska användas i ROV:n har ett inbyggt Kalmanfilter som implementerats av tillverkaren, Xsens [2]. Därför finns valmöjligheten att använda antingen rå mätdata eller IMU:ns egna skattade orientering. Rå mätdata är tillgänglig både direkt från A/D-omvandlaren eller i fysikaliska storheter. Den skattade orienteringen finns tillgänglig som Eulervinklar, kvaternioner eller som en rotationsmatris. Det kan vara intressant att ha tillgång till både filtrerad och ofiltrerad data för att eventuellt kunna utvärdera skillnad i kvalitet hos skattningen om man använder redan filtrerad orientering kontra rådata i fusion av sensordata från samtliga sensorer.

Vid användning av det inbyggda Kalmanfiltret kommer en mätning från IMU:n att vara en vektor (ϕ, θ, ψ) där ϕ, θ och ϕ är definierade enligt Figur 4 och är mätta i grader. Om rådata istället tas ut kommer en mätning bestå av en vektor $(a_X, a_Y, a_Z, \dot{\phi}, \dot{\theta}, \dot{\psi}, B_X, B_Y, B_Z)$ där accelerationerna är angivna i m/s^2, vinkelhastigheterna rad/s och magnetiska fälten i *a.u.* (arbitrary units, normerade med avseende på jordens magnetiska fältstyrka). Även här är X, Y, Z, $\dot{\phi}$, $\dot{\theta}$ och $\dot{\psi}$ är definierade enligt Figur 4.

Alltså kommer en mätning från IMU:n att vara

$$y_k^{IMU} = (\phi_k\ \theta_k\ \psi_k)^T + e_k^{IMU},\ e_k^{IMU} \sim N(\mu_k, \sigma_k^2), \tag{21}$$

alternativt

$$y_k^{IMU} = (a_{Xk}\ a_{Yk}\ a_{Zk}\ \dot{\phi}_k\ \dot{\theta}_k\ \dot{\psi}_k\ B_{Xk}\ B_{Yk}\ B_{Zk})^T + e_k^{IMU},\ e_k^{IMU} \sim N(\mu_k, \sigma_k^2). \tag{22}$$

7.2.2 Trycksensor

Trycksensorn som ska integreras i systemet mäter, likt många andra trycksensorer, skillnaden mellan det aktuella trycket och ett referenstryck. I detta fall är dock referenstrycket 0 Pa (vakuum) [5], vilket innebär att trycket mäts absolut.

Utsignalen för trycksensorn är en skalning av dess matningsspänning och varierar mellan 10 % och 80 % av denna. 10 % svarar mot det minimala trycket (0 psi) som sensorn kan mäta och 80 % svarar mot det maximala trycket (100 $psi \approx 6,8\ atm$) som sensorn kan mäta. Enligt tillverkaren mäts trycket med en fel som maximalt är ± 2 % av det faktiska trycket. A/D-omvandling av utsignalen görs av Arduino-kortet som har 10 bitars upplösning där standardinställningar är att GND svarar mot 0 och $+5V$ svarar mot 1024, vilket innebär en upplösning på $4,88mV$.

Det hydrostatiska trycket som funktion av vattendjup kan beräknas

$$P_h = \frac{mg}{A} = \frac{\rho V g}{A} = \rho g h, \tag{23}$$

där A är den area som man tänker sig att kraften från vattenpelaren med höjd h appliceras på, $V = Ah$, g är tyngdaccelerationen på jordytan och ρ är densiteten för vattnet som kommer att bero på vattnets temperatur.

För att erhålla det totala trycket i vattnet adderas lufttrycket P_0 till det hydrostatiska trycket P_h

$$P_{total} = P_0 + P_h. \tag{24}$$

Utifrån detta borde sensorn i teorin klara av att mäta trycket upp till ungefär 59 meters djup beroende på lufttrycket och vattnets temperatur.

Enligt databladet [5] är utspänningen proportionellt mot trycket vilket ger en tryckmätning

$$y_k^p = KP_{tot} + c + e_k = K(\rho g h_k + P_0) + c + e_k^p,\ e_k^p \sim N(\mu_k, \sigma_k^2), \tag{25}$$

Eftersom bassängens koordinatsystem valts till ett NED-system med origo vid vattenytan är därför $h = Z$. Detta ger

$$y_k^p = K(\rho g Z_k + P_0) + c + e_k^p,\ e_k^p \sim N(\mu_k, \sigma_k^2), \tag{26}$$

där y_k^p är spänning och proportionalitetskonstanten K beskriver förhållandet mellan spänning och tryck. c är den offsetkonstant som krävs för att en linjär sensormodell ska kunna användas samtidigt som vakuum ska svara mot en nollskild spänning. Detta innebär alltså att $c = 0,1V_{cc}$, där V_{cc} är matningsspänningen till sensorn. Empiriska tester måste göras för att se att denna modell stämmer åtminstone i det arbetsområde som ROV:n ska verka i.

Med avseende på A/D-omvandlingens upplösning på $4,88\ mV$ kommer upplösningen för trycket att bli $\frac{4,88\ mV}{K}$. Om K verkligen är konstant kan K beräknas genom att lösas ut ur förhållandet som råder vid en mätning av maximaltrycket: $K = \frac{0,8V_{cc} - 0,1V_{cc}}{689476Pa} = 5,076 \cdot 10^{-6} V/Pa$. Detta ger att upplösningen i tryck som är $\frac{4,8\ mV}{K} = 961\ Pa$, vilket ger en upplösning i djup på $\frac{961\ Pa}{g\rho} = 0,098\ m = 9,8\ cm$.

Detta kan eventuellt bli ett problem då vi vill skatta djupet med ett fel på mindre än 10 cm. Därför kan hänsyn tas till att ROV:n bara kommer att operera på ett begränsat djup, exempelvis mellan 0 m och 20 m. Med detta antagande kommer spänningen att variera mellan $1,06\ V$ och $2,01\ V$ och detta medför en upplösning på $\frac{2,01-1,06}{1024}\ V = 0,93\ mV$ hos den A/D-omvandlade spänningen. Detta ger en upplösning av trycket på 183 Pa och för djupet $0,019\ m = 1,9\ cm$.

Ett problem som kan uppstå är att trycksensorn eventuellt kommer att påverkas av dynamiskt tryck vid högre hastigheter. Detta kommer att bero på vart sensor placeras på

ROV:n. I dagsläget är det ej känt vart sensorn kommer att vara placerad, så det kan bli aktuellt att ändra antingen modellen för sensorn eller placeringen av trycksensorn under projektets gång om det visar sig att dynamiskt tryck är någonting som måste tas hänsyn till.

7.2.3 SONAR: SSPC

Två stycken SSPC-enheter ska placeras på ROV:ns undersida sådant att dessa kan svepa över både bassängens botten och kanter. Data från mätningar lagras i MSTIFF-filer, vilket är ett filformat designat av tillverkaren Marine Sonic Technology Ltd. Dessa filer innehåller information om avstånd från SONAR-enheten till respektive punkt på bassängbotten som är representerad i filen. Utifrån denna placering av enheterna kan avståndet mellan ROV:n och bassängen i y- respektive z-led bestämmas. Fusion av denna information, kännedom om bassängens totala djup och information från tryckmätningar borde alltså ge ökad säkerhet i skattningen av ROV:ns Z-position.

För att kunna bestämma positionen helt behöver avståndet framåt bestämmas. Detta är eventuellt inte möjligt då sensorerna är placerade på sidorna. Om avståndet till bassängkanten är konstant, det vill säga om ROV:n färdas parallellt med denna så kommer positionen ej att kunna bestämmas. Dock om ROV:n färdas någorlunda snett relativt bassängen kommer förändringen av avståndet till bassängkanten att kunna användas för att bestämma position och även hastighet, eftersom ROV:n kommer att ha en karta över bassängen och veta sin orientering i denna. Om ROV:n är orienterad parallellt med bassängkanten och man vill detektera vad som finns framför den så finns alltid möjligheten att rotera en vinkel i yaw-led, exempelvis 90°, för att se omgivningen framåt och bakåt.

7.3 Sensorfusionen i ROS

Detta delsystem kommer att implementeras i node *sensor* i ROS. Mer om interface mellan olika nodes i Kapitel 3.

8 Kommunikation

Detta delsystem ska sköta kommunikationen mellan de externa komponenterna i systemet samt de interna delsystemen.

8.1 Kommunikation med externa enheter

Delsystem Kommunikation ska skicka framtagna styrsignaler samt ROV:ns skattade tillstånd, position och orientering till den externa PC:n. Den externa PC:n ska skicka information till delsystem Kommunikation om ROV:ns läge, det vill säga autonom-, stabilisera-, manuell- eller nöddrift. Delsystem Kommunikation har sedan som uppgift att distribuera denna information till de andra delsystemen.

Delsystem Kommunikation ansvarar även för att samla in rå sensordata från IMU:n, Arduino-kortet samt SONAR-sensorer.

8.2 Kommunikation med interna delsystem

Delsystem Kommunikation ska skicka och ta emot data från de olika delsystemen och den vidarebefordrar även viss data till den externa PC:n. Detta kan vara skattade tillstånd från sensorfusionen eller motorernas styrsignaler från regleringen.

Hur all kommunikation mellan kommunikationsenheten och de interna delsystemen ska ske beskrivs i Kapitel 3.

8.3 Delsystemet i ROS

Kommunikationen kommer att implementeras i node *com* i ROS.

9 Grafiskt användargränssnitt

Här beskrivs det grafiska användargränssnitt (*eng: Graphical User Interface*) (GUI) som utvecklas för den externa PC:n och skrivs i Matlab. GUI:t ska ha en enkel utformning och visa ett antal värden, en figur över en 2D-karta med ROV:ns position samt indikera djupet ROV:n befinner sig på. Det ska finnas funktionalitet för att ändra ett antal parametrar samt skicka kommandon till ROV:n. Det ska även finnas funktionalitet för att visa felmeddelanden, vilket ska ske om något ett antal fel inträffar.

Via detta användargränssnitt ska en användare kunna utföra följande:

- Växla mellan manuell, stabiliserad och autonom drift.
- Växla mellan LQ-regulator och MPC.
- Ändra reglerparametrar, vilket innefattar straffmatriser för mät- och processbrus, för LQ-regulatorn och MPC:n, samt prediktionshorisont för MPC:n.
- Ändra filterparametrar för sensorfusionsdelsystemet, vilket innefattar kovariansmatriser för process- och mätbrus.
- Skicka kommandon till ROV:n om att åka till en viss position med en viss orientering.
- Nödstopp.

Aktuell regulator och styrläge ska även kunna avläsas. Då ett kommando om färd skickats ska planerad trajektoria visas på kartan.

De numeriska värden som ska visas i GUI:t innefattar följande:

- ROV:ns skattade position och orientering med osäkerhetsparametrar, exempelvis varians eller konfidensintervall.
- Referenssignaler till aktuell regulator.
- Aktuella reglerparametrar.
- Aktuella filterparametrar.
- Aktuella styrsignaler till motorerna.
- Koordinater för origo samt det hörn som är längst från origo, alltså i motsatt hörn, vilka definierar kartan över en rätblocksformad bassäng.

Larm för följande fel eller problem ska kunna visas:

- Läckage.
- Låg batterinivå.
- Bruten kontakt mellan den externa PC:n och ROV:n.

A Komponentlista

Nedan följer en lista med de viktigaste komponenterna som används i ROV:n.

Komponent	Produktnamn	Artikelnr	Tillverkare
Batteriladdare med balansering	Hyperion EOS 1420i	HPEOS1420INET3	Hyperion
Batterier	DesirePower 4200mAh 3S 35C V8	DP42003S35C	DesirePower
Moderkort	Intel D525MW	BOXD525MW	Intel
RAM-minne	Kingston 4096MB DDR3 PC3-8500 1066 MHz	KTAMB1066K2/4G	Kingston
Hårddisk	OCZ 60 GB Vertex II E-series	OCZSSD2-2VTXE60G	OCZ
Handkontroll	Xbox gamepad wireless, PC	JR9-00010	Microsoft
Styrkort	Arduino Mega 2560, DEV-09949	ARD-M2560	Arduino
Kamera	LifeCam Cinema	H5D-00003	Microsoft
Framdrivningsmotor	RE 50 Ø50 mm, Graphite Brushes, 200 Watt	370354	Maxon Motor
Planetväxel till framdrivningsmotor	Planetary Gearhead GP 52 C Ø52 mm, 4 - 30 Nm, Ceramic Version	223081	Maxon Motor
Positioneringsmotor	RE 40 Ø40 mm, Graphite Brushes, 150 Watt	148867	Maxon Motor
Planetväxel till positioneringsmotor	Planetary Gearhead GP 42 C Ø42 mm, 3 - 15 Nm, Ceramic Version	203114	Maxon Motor
Tryckgivare	-	PX2AN1XX100PAAAX	Honeywell
IMU	MTi	-	Xsens

Referenser

[1] Sundin, P. *Kravspecifikation version 1.0*. TSRT10 Remotely Operated Underwater Vehicle, 2012-09-20.

[2] Bernhard, J, Johansson, P. *Advanced control of a remotely operated underwater vehicle*. Institutionen för systemteknik (ISY), Examensarbete, 2011.

[3] Eriksson, M. *Utvärdering och vidareutveckling av undervattensfarkost*. Institutionen för industriell och ekonomisk utveckling (IEI), Examensarbete, 2011.

[4] Xsens Technologies B.V.. *MTi and MTx User Manual and Technical Documentation*. Document MT0100P, Revision O, 15 Oct 2010.

[5] Honeywell. *PX2 Series Heave Duty Pressure Transducer*. 50069942 Rev. A - EN, February 2012

[6] F. Gustafsson. *Statistical sensor fusion*. Edition 1:1, Studentlitteratur AB, Lund. ISBN 978-91-44-054189-6

[7] Conley, K. (2011-12-07) *ROS/TCPROS*. Hämtat 25/09/2012, från http://www.ros.org/wiki/ROS/TCPROS

[8] Fergusson, M., Stambler, A. *rosserial*. Hämtat 27/09/2012, från http://www.ros.org/wiki/rosserial