

INGENIØRHØJSKOLEN ÅRHUS

ELEKTRO-INGENIØR LINIEN

SEMESTERPROJEKT E4PRJ4

RAPPORT

---

**Bias Reducing Operating System**

---

*Authors:*

---

Nicolai Glud  
11102

---

Mick Holmark  
11065

---

---

Johnny Kristensen  
10734

---

Rasmus Lund-Jensen  
11111

---

---

Jacob Roesen  
10095

---

---

Carl Jakobsen  
*Vejleder*





# Resume 1

---

Denne rapport for projekt på 4. semester Elektro på Ingeniørhøjskolen Aarhus, Aarhus Universitet præsenterer en gennemgang af gruppens projekt: "Bias Reducing Operating System". Formålet er selvstændigt at definere et projekt, der inddrager discipliner fra semesters fag. Der er i projektet lagt stor vægt på processen, men også i at opnå erfaring med dokumentation og test. Der er blevet udfærdiget grundig dokumentation gennem hele forløbet der dækker over kravspecifikation, systemarkitektur samt design og implementering. Der er under alle faser også defineret tests, der sikrer at implementeringen er sket rigtigt, og at alle blokke kan arbejde sammen som foreskrevet i dokumentationen. Gennem projektet er der brugt Scrum som værktøj til strukturering af arbejdet.

Projektformuleringen, der er udformet af gruppen, indebærer implementering af slagssidereregulering af bulkskibe. Dette kan med fordel anvendes på bulkskibe under lastning og løsning for at hjælpe hele processen herunder. Der er fokuseret på at udvikle et system af så høj kvalitet som muligt og for at hæve kvaliteten af projektet har gruppen benyttet hinandens individuelle faglige og projektrelaterede styrke.

Projektet er endt ud med et stort set implementeret system som beskrevet i grundsystemet. Der er gennemført flere iterationer, der gradvist har øget systemet og dokumentationens præcision. Gruppen har opnået bedre erfaring med teamwork, udviklingsforståelse, arbejdsstrukturering samt faglig perspektivering.



# **Abstract** 2

---

This report for 4<sup>th</sup> semesters electronic engineering project at Aarhus School of Engineering, Aarhus University presents an explanation of the groups project: "Bias Reducing Operating System". The purpose of this project is to define a process including disciplines from this semesters courses. There has been put a tremendous amount of work in the processes of making this project and acquire experience in documentation and testing. A thorough documentation of the whole processes, covering requirements specification, system architecture, design and implementation, has been made. Tests have been defined for every phase which guaranties proper implementation and integration as defined by documentation. The tool 'Scrum' has been used to structure the progress.

The group has written a project development document entitling the implementation of bias regulating a bulk carrier. The system is for use while loading and unloading bulk carriers to assist the crew. Focus has been put on developing a system of high quality. To ensure high quality the group has used each other's individual professional and project-related forces.

The project has resulted in a nearly fully implemented system as described in the documentation. Multiple iterations has increased the precision of the system and following documentation. The group has reached a higher level of experience in fields such as teamwork, understanding development, work planning and professional perspectivation.

# Indholdsfortegnelse

---

<b>Kapitel 1</b>	<b>Resume</b>	<b>3</b>
<b>Kapitel 2</b>	<b>Abstract</b>	<b>5</b>
<b>Kapitel 3</b>	<b>Forord</b>	<b>9</b>
<b>Kapitel 4</b>	<b>Indledning</b>	<b>11</b>
4.1	Læsevejledning . . . . .	11
<b>Kapitel 5</b>	<b>Projektformulering</b>	<b>13</b>
5.1	Opgaveformulering . . . . .	13
5.2	Projektformulering . . . . .	13
<b>Kapitel 6</b>	<b>Systembeskrivelse</b>	<b>15</b>
<b>Kapitel 7</b>	<b>Kravspecifikation</b>	<b>17</b>
7.0.1	Overordnede krav til systemet . . . . .	17
7.0.2	Funktionelle krav . . . . .	17
7.0.3	Ikke-funktionelle krav . . . . .	18
7.0.4	Krav til udviklingsprocess og teknologi . . . . .	18
7.0.5	Krav til grænseflader . . . . .	18
7.0.6	Kvalitetsfaktorer . . . . .	19
<b>Kapitel 8</b>	<b>Afgrænsning</b>	<b>21</b>
<b>Kapitel 9</b>	<b>Projektbeskrivelse</b>	<b>23</b>
9.1	Projektgennemførelse . . . . .	23
9.1.1	Rollefordelinger . . . . .	23
9.2	Metoder . . . . .	24
9.2.1	Scrum . . . . .	24
9.2.2	V-model . . . . .	25
9.3	Analyse . . . . .	25
9.3.1	Hvordan mäter vi hældning? . . . . .	25
9.3.2	Hvordan skal de forskellige modulerne forsynes? . . . . .	26
9.3.3	Raspberry Pi som host for Kontrolinterfacet . . . . .	27
9.3.4	Server . . . . .	28
9.3.5	Valg af database . . . . .	28
9.3.6	Webinterface . . . . .	28
9.3.7	Jura . . . . .	28
9.4	Systemarkitektur . . . . .	29
9.4.1	Stadier for konceptuelle klasser . . . . .	30
9.5	Design og Implementering . . . . .	33

---

9.5.1	Kontrolinterfacet . . . . .	33
9.5.2	Vandballasttankenhed . . . . .	36
9.5.3	Styringsmodulet . . . . .	40
9.5.4	Strømforsyning . . . . .	42
9.5.5	Database . . . . .	44
9.5.6	Serveren . . . . .	45
9.5.7	Webinterface . . . . .	47
9.5.8	MySQL . . . . .	49
9.6	Resultater . . . . .	49
9.6.1	Kontrolinterfacet . . . . .	49
9.6.2	Database og webinterface . . . . .	50
9.6.3	Styringsmodul . . . . .	50
9.6.4	Vandballasttankenhederne . . . . .	50
9.6.5	Strømforsyning . . . . .	51
9.6.6	Samlede resultat og vurdering af resultater . . . . .	51
9.7	Opnåede erfaringer . . . . .	52
9.7.1	Gruppen . . . . .	52
9.7.2	Agile udviklingsmetoder . . . . .	52
9.7.3	Udvikling af nye komponenter . . . . .	52
9.7.4	Test . . . . .	52
9.7.5	Værktøjer . . . . .	53
9.8	Udviklingsværktøjer . . . . .	53
<b>Kapitel 10</b>	<b>Konklusion</b>	<b>55</b>
<b>Kapitel 11</b>	<b>Forbedringer til systemet</b>	<b>57</b>
<b>Kapitel 12</b>	<b>Referencer</b>	<b>59</b>
12.1	Artefakter . . . . .	59
12.1.1	Kravspecifikation . . . . .	59
12.1.2	Accepttestspezifikation . . . . .	59
12.1.3	Systemarkitektur . . . . .	59
12.1.4	Integrationstestspezifikation . . . . .	59
12.1.5	Detaljeret design . . . . .	59
12.1.6	Enhedstestspezifikation . . . . .	59
12.2	Hjemmesider . . . . .	60
12.3	Liste over bilag på CD . . . . .	60
12.3.1	Logbøger . . . . .	60
12.3.2	Kode . . . . .	60
12.3.3	Dokumentation . . . . .	62
12.3.4	Datablade . . . . .	63
12.3.5	Billeder . . . . .	63
12.3.6	Jura . . . . .	63



# Forord 3

---

Denne rapport er udarbejdet af fem ingeniørstuderende ved Ingeniørhøjskolen i Aarhus, Aarhus Universitet. Rapporten er hovedproduktet i et obligatorisk projektforløb på 4. semester og gennemgår overordnet gruppens besvarelse og gennemførelse af projektet. Ud over rapporten er der blevet udarbejdet en række projektdokumentationsdokumenter og et fysisk produkt. For yderligere detaljer henvises der til projektdokumentationen.

Det har været gruppens formål med projektet at lære udviklingsværktøjene bedre at kende. Systemet er derfor anset som det emne, redskaberne anvendes på, mere end det egentlig mål med projektet.

Rapporten er skrevet med henblik på at læseren er af samme faglige niveau som gruppen, hvilket vil afspejle sig i det faglige sprog.



# Indledning 4

---

Projektets emne er "slagsideregulering af bulkskib", som er et selvvalgt emne. Rapporten beskriver udviklingsprocessen herunder projektforløbet, hvilke metoder der er anvendt og hvilke overvejelser der ligger til grund for de valgte løsninger. I forbindelse med at de valgte løsninger bliver beskrevet, vil der også blive fremlagt alternativer og begrundelser for, at de ikke blev valgt.

Der har fra ekstern side været stillet nogle krav til projektet og det system, der skulle udvikles. Arbejdsprocessen og nogle af komponenterne er således påvirket af disse krav. Kravene kan ses i afsnit 5.1.

Formålet med projektet er at anvende de teorier og metoder, som er blevet tilegnet gennem studiet, og ikke mindst tilegne sig ny viden på egen hånd. Først da er gennemførelsen af et komplet projektforløb fuldført.

Projektet har været inddelt i fem udviklingsfaser. Udviklingsfaserne er som følger:

- Kravspecifikation
- Analyse og arkitektur
- Detaljeret design
- Implementering
- Test

## 4.1 Læsevejledning

Rapportens opbygning er struktureret således at den giver den bedste gennemgang af hele projektforløbet. Rapporten er i hovedtræk delt op i to dele. De første afsnit beskriver det overordnede system og projekt. Tilblivelsen af systemet og projektet med tilhørende overvejelser beskrives i de efterfølgende afsnit. Denne adskillelse sker mellem afsnit 8 og 9.

Rapportens egentlige indhold begynder fra afsnit 5.1 – Opgaveformulering. Opgaveformuleringen indeholder minimumskrav til projektet og er givet til gruppen af vejlederen.

I projektformuleringen bliver efterfølgende der defineret præcist hvad dette projekt kommer til at dreje sig om, og hvordan gruppen har formuleret dette. Herefter følger en beskrivelse af det samlede, tænkte system.

I afsnit 7 – Kravspecifikationen, fremlægges kravene der er stillet til projektet af gruppen. Herefter beskrives projektafgrænsningen samt arbejdsmetoder og fremgangsmåde i afsnit 8 til 9.2

Afsnit 9.3 beskriver det analysearbejde projektet har gennemgået. I afsnittene fra 9.4 og 9.5 nedbrydes hele projektet fra øverste abstraktionsniveau og ned til implementeringen. Der er her gået i dybden med de vigtige aspekter i forhold til dette projekt. Disse afsnit

har samtidig også en naturlig overgang til hinanden ud fra systemarkitekturen. Hernæst samles der op på de opnåede resultater i afsnit 9.6. Efter resultaterne er præsenteret, fremlægges de erfaringer gruppen har opnået igennem hele projektforløbet, samt hvilke ting, der har fungeret godt. Afslutningsvis konkluderes der på hele projektet på godt og ondt i afsnit 10 for til sidst at sætte nogle ord på forbedringer til systemet i afsnit 11.

Alle afsnit er skrevet så de som udgangspunkt godt kan stå alene, hvorfor der igennem rapporten vil komme gentagelser, hvis man læser denne fortløbende. Det anbefales dog at læse rapporten fortløbende for at få den bedste, samlede forståelse for projektet og produktet.

# Projektformulering

5

## 5.1 Opgaveformulering

Opgaveformuleringen er givet til gruppen for projektet som retningslinier. Følgende er givet som krav i disse retningslinier.

- Systemet skal interagere med omverdenen vha. sensorer og aktuatorer.
- Der skal anvendes relevante faglige elementer fra semestrets kurser.
- Systemet skal omfatte pålidelig transmission af data mellem udvalgte enheder.
- Systemet skal kunne interagere med en bruger

## 5.2 Projektformulering

### Baggrund

Når man laster eller losser et bulkskib bruges der mange resourcer på at kontrollere at skibet ikke får slagseite. Der bruges lang tid på at planlægge hvor, hvornår og hvad der skal lastes. Hertil står skibsføreren på broen under hele lastningen/losningen og overvåger at det bliver gjort korrekt. Inde på terminalen bliver skibet også overvåget for at sikre, at der ikke sker fejl. Derudover er det ifølge lovgivningen<sup>1</sup> også muligt at imødekomme en slagsside, forsaget af en forestående last, ved at flytte rundt på ballast i ballasttanke. Gruppen vil derfor gerne arbejde med et system der automatisk kan afhjælpe problemer med slagsside og minimere tiden, der skal bruges på planlægning af en lastning/losning.

### Projektdefinition

Det er gruppens ønske at lave et system der korrigerer skibe fra at få slagseite i forbindelse med lastning og losning af gods. Overordnede systemkrav udarbejdet i projektformuleringen:

- Systemet skal logge data omkring skibsnavn og status til en database.
- Systemet skal monitorere om skibet er i vatter.
- Systemet skal modkompensere hvis skibet måles til at være ude af vatter.

<sup>1</sup>Se bilaget *Jura* for relevant lovgivning og henvisning til lovskrifter



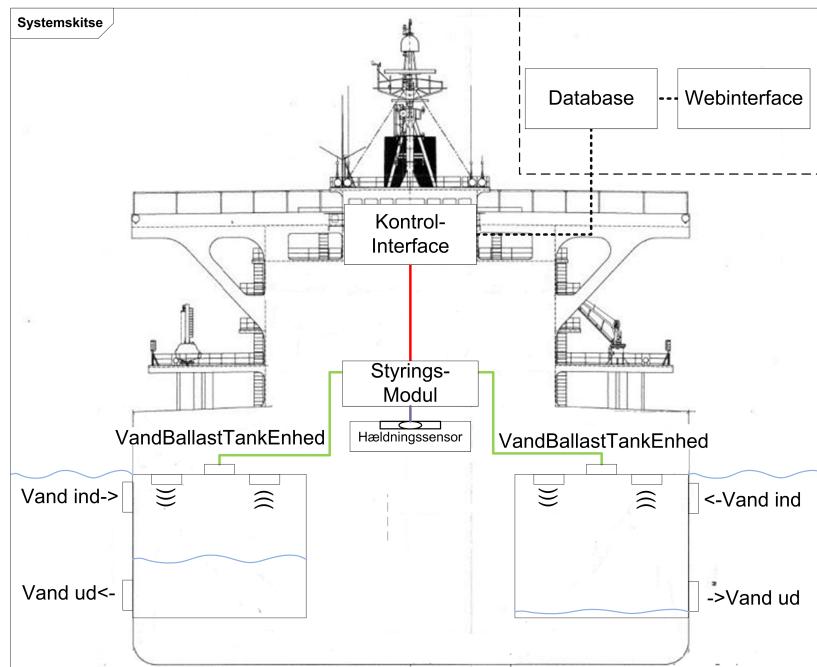
# Systembeskrivelse 6

BROS er et sikkerhedssystem til skibe. Systemet aktiveres ved lastning eller losning. Her er det systemets opgave at sørge for at skibet ikke får slagsside - heraf navnet: Bias Reducing Operating System (Slagsidereducerende Operativt System). I systemet er der indbygget en hældningssensor og to vandballasttanke - en i hver side af skibet. På baggrund af målinger fra hældningssensoren vil Styringsmodulet vurdere hvorledes indholdet af tankene skal justeres af Vandballasttankeenhederne således at der korrigeres for en slagseite af skibet.

Hele systemet styres fra Skibsførens kontor hvor Kontrolinterfacet - en grafisk brugergrænseflade - er installeret. Her kan skibets hældning, vandindholdet af tankene og statusmeldinger for systemet aflæses. Det er også her systemet aktiveres og deaktiveres.

Som udgangspunkt vil systemet automatisk opretholde en hældning på nul grader, men hvis der opstår et behov kan man her manuelt indstille skibet til en mindre slagseite. Dette kan gøres for at imødekomme en større, modsatrettet, slagseite påført af forestående ændringer i skibets last.

For at indsætte et ekstra sikkerhedselement vil systemet under hele processen løbende sende værdier for systemet til en ekstern database. Dermed kan også terminalrepræsentanten følge skibets status.



Figur 6.1. Systemskitse af BROS



# Kravspecifikation 7

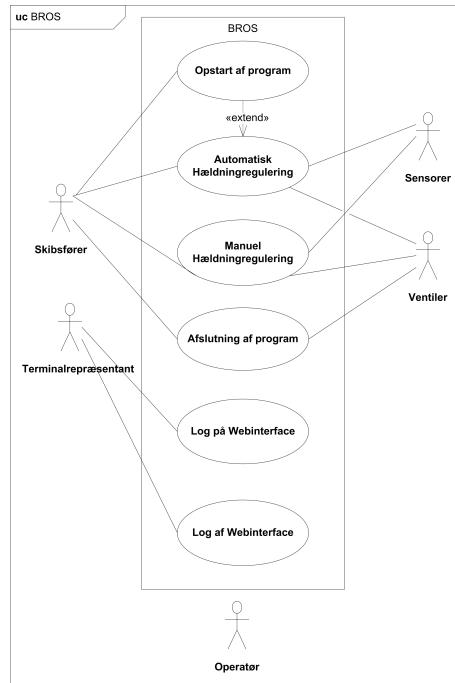
Kravspecifikationen er udarbejdet i begyndelsen af projektforløbet og omfatter use cases, ikke-funktionelle krav samt kvalitetsfaktorer. For den fulde kravspecifikation henvises der til dokumentationsdokumentet *Kravspecifikation*.

## 7.0.1 Overordnede krav til systemet

Systemet skal indeholde en til flere sensorer til måling af vandstand i tanke, en sensor eller flere sensorer til måling af skibshældning, en grafisk brugergrænseflade til skibsføreren samt et webinterface til terminalrepræsentanten.

## 7.0.2 Funktionelle krav

Systemets funktioner er beskrevet ved hjælp af use cases. Før use cases udfærdiges nedskrives systemets aktører. Der udarbejdes en beskrivelse af deres funktion samt ansvar i systemet. Hver use case repræsenterer en enkelt funktionalitet i systemet.



**Figur 7.1.** Use case diagram for BROS

Der er også beskrevet alternative hændelser, såfremt hændelsesforløbet ikke forløber som planlagt. Alle use cases har fulgt samme fremgangsprocedure. Se kravspecifikationsdoku-

mentet for alle systemets use cases. På figur 7.1 ses alle use cases for systemet.

Når systemet startes op første gang anvendes *Opstart af program* (use case 1). Når systemet er startet op kan brugeren vælge mellem *Automatisk hældningsregulering* (use case 2) eller *Manuel hældningsregulering* (use case 3). Hvis brugeren vil deaktivere systemet kan han afslutte programmet (use case 4). En terminalrepræsentant kan tilgå databaseinformationen via et Webinterface (use case 5 og 6).

### 7.0.3 Ikke-funktionelle krav

I dette afsnit beskrives krav til tolerancer og andre krav der ikke er use case specifikke. Nedenfor er noteret eksempler på ikke-funktionelle krav:

- Sensorer:
  - ◊ Hældningssensor:  
Hældningssensor måler skibets hældning i forhold til vandret i området -7.5 til 7.5 grader, med en nøjagtighed på  $\pm 2.5$  grader.
  - ◊ Afstandssensor i ballasttank  
Vandniveauet i ballasttanken skal måles fra 0-100% med en nøjagtighed på  $\pm 2.5\%$ -point.
- Alarmer udløses når:
  - ◊ Vandniveauet i ballasttanke bliver mere end 70%.
  - ◊ Hældning på skibet bliver større end  $\pm 5$  grader.
  - ◊ Mistet eller ingen forbindelse internt i systemet.
  - ◊ Mistet eller ingen forbindelse til ekstern database.

### 7.0.4 Krav til udviklingsprocess og teknologi

Projektforløbet skal følge V-modellen. Scrum skal bruges i projektstyringsprocessen til at give overblik over projektets arbejdsopgaver. Blandt gruppens medlemmer uddelegeres posterne Scrum-master og projektleder.

SysML anvendes til at beskrive systemet overordnet og i detaljer.

Til programmeringen af systemets indlejrede enheder anvendes C. Til Kontrolinterfacet og Serveren anvendes C++. Til databaselagring anvendes MySQL. Webinterfacet skal skrives i php og HTML 4.01.

### 7.0.5 Krav til grænseflader

Kommunikationen mellem Kontrolinterfacet og Styringsmodulet skal følge UART protokollen. Det kan kun tilkobles ét Kontrolinterface og ét Styringsmodul.

Til Styringsmodulet tilkobles én sensor samt to Vandballasttankenheder ved hjælp af I2C-protokollen.

På kontrolinterfacet skal alt funktionaliteten være indbygget i brugergrænsefladen. Det skal være muligt at skifte mellem automatisk hældningsregulering og manuel hældningsregulering.

### 7.0.6 Kvalitetsfaktorer

For at sikre kvaliteten af produktet er der opstillet en række kvalitetsfaktorer. Det er meget vigtigt at systemet er pålideligt, sikkert og effektivt. Pålideligheden er vigtig da systemet kan risikere at være fatalt for et skib, hvis der sker en fejl. Systemet skal være sikkert da det er et kritisk komponent. Endvidere skal systemet være effektivt da det ikke må sløve lastning- og losningsprocessen.

Krav til bruger- og vedligeholdsesvenlighed er middelvægtet, da brugerens anvendelse af systemet skal hjælpe til processen og ikke gøre den yderligere kompliceret. Systemet skal kunne vedligeholdes af en tilkaldt operatør.



# Afgrænsning 8

---

Afsnittet vil give en oversigt over afgrænsninger dette projekt er udarbejdet med.

## Afgrænsning fra ekstern kilde:

Systemet skal kunne interagere med omverdenen ved hjælp af sensorer og aktuatorer. Endvidere skal der også anvendes faglige elementer fra fjerde semesterets kurser. Transmission af data mellem enheder i projektet skal være pålidelig. Til slut skal systemet indeholde brugerinteraktion.

## Afgrænsninger sat af gruppen selv

Gruppen udarbejdede i starten af projektet ideer til en række funktionaliteter. Disse funktionaliteter blev inddelt i henholdsvis grundsystem og udvidelser, som vist i nedenfor.

- **Grundsystem:**

- ◊ Elektronisk måling af hældning
- ◊ Automatisk hældningsregulering
- ◊ Niveaumåling i ballasttanke
- ◊ Advarselssignaler

- **Udvidelser:**

- ◊ Måling af afstand til terminal-kaj
- ◊ Måling af dybdegang
- ◊ Manuel hældningsregulering
- ◊ Pålidelig kommunikation med ekstern enhed

Grundsystemet er det system gruppen har fastlagt sig på at implementere i projektforløbet. Funktionaliteterne her er valgt for at opfylde basiskrav fra opgaveformuleringen i afsnit 5.1.

Udvidelser er nedprioriterede funktionaliteter da de enten har lille relevans eller ikke er kritiske for at systemets grundformål kan opfyldes, se afsnit 5.2. De vil derfor kun blive prioriteret såfremt tiden er dertil.

Man vil senere hen ligeledes kunne udvide systemet med funktionaliteter på baggrund af feedback fra kunden og markedsundersøgelser.



# Projektbeskrivelse

9

## 9.1 Projektgennemførelse

Projektet er udført af en gruppe på fem personer. Gruppen er ny og sammensat af personer fra fire forskellige projektgrupper. Grundet gode erfaringer fra de forskellige grupper har gruppen valgt at arbejde med projektet ud fra Scrum-modellen. Gruppen har fastlagt rollerne i tabel 9.2.

Projektet er blevet brudt ned i mindre dele og det enkelte gruppemedlem har fået ansvaret for hver sin del. Der har så internt været sparring mellem de dele, der har kommunikation med hinanden. Nedenfor ses fordelingen af arbejdet:

Kontrolinterfacet:	Rasmus Lund-Jensen
Webserver og database:	Mick Holmark
Strømforsyning:	Jacob Roesen
Vandballasttankenhed:	Johnny Kristensen
Styringsmodulet:	Nicolai Glud

**Tabel 9.1.** Tabel over ansvarsfordeling

Der er blevet udarbejdet en tidsplan, der løbende er blevet revirderet. Projektets overordnede tidsplan for faserne og de eksterne milestones ligger som bilag.

### 9.1.1 Rollefordelinger

Projektleder:	Jacob Roesen
Projektkoordinatore:	Nicolai Glud
	Jacob Roesen
Scrummaster:	Johnny Kristensen

**Tabel 9.2.** Tabel over rollefordelinger

Vi har valgt at lave roller ud fra vores udviklingsmetode, Scrum, der er beskrevet senere. Projektlederen har haft til ansvar at strukture arbejds- og Scrummøder. Projektkoordinatørs ansvar har været at planlægge møder og bestille lokaler. Scrummasteren er anvarlig for udviklingsplatformen, Scrum, og sørge for at metoden anvendes som tiltænkt.

## 9.2 Metoder

En kort præsentation af de to mest dominerende arbejdsmetoder der er anvendt.

I dette projekt er der anvendt metoder indlært gennem et tidligere projekt. Værktøjerne er de værktøjer som gruppen føler sig trygge ved og som gruppen føler bidrager mest til processen.

### 9.2.1 Scrum

I gruppens implementering af Scrum startes der med at lave en produktbacklog, som er den kunden ser. Derefter planlægges det første sprint. Et sprint spænder over to uger. Når et sprint starter bliver opgaver overført fra backloggen til sprintet. Her bliver omfanget af opgaven vurderet og anført som en arbejdsvægtning. Herefter fordeles opgaverne mellem gruppens medlemmer. En gang om ugen holdes et Scrummøde. Mødets dagsorden har tre punkter: Status på opgaver, eventuelle problemer og til slut hvad der arbejdes videre med. Nye sprint startes hver anden uge. Hvis en opgave ikke er færdiggjort, vurderes det om opgaven skal videreføres i det efterfølgende sprint.

I projektet har der i alt været syv sprint. På figur 9.1 bruges sprint nummer to som eksempel.

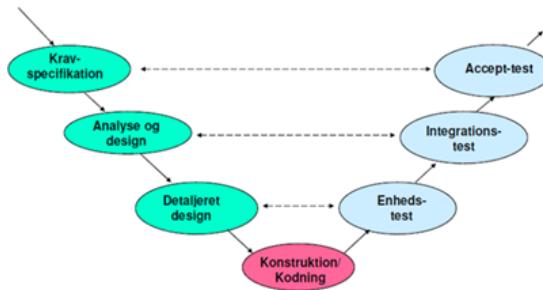
Sprint nr:	Scrummaster:	JK	Koordinator:	NG		
	Projektleder:	JR				
Overordnet opgave:	Opgaver:	Arbejdsvægtning:	Resource #1	Resource #2	Process (% done):	Kommentar:
Videreført Accepttest	Videreført: Jura Videreført: Snak m. Arne vedr. sensor	3 1 2	MH NG RLJ	RLJ JK NG	100% 100% 100%	NG, JK NG, JR, JK, MH
Systemarkitektur	Systemkomponenter - Enheder	2	JR	RLJ	80%	JR, JK
	Komponentvalg	3	ALLE		100%	RLJ: er afsnittet skrevet og læst af alle?
	Strukturel systemarkitektur Behavior systemarkitektur	4 5	MH JK	JK MH	60% 60%	MH: Tilrettelse af tegning, tror vi skal sætte os ned os diskutere, flere ting JK, NG
Teknologiundersøgelse	HW Systemarkitektur Grænseflader / interface Forside	0 3 1 5	JR NG JR RLJ	JK 80% JR ALLE	20% 80% 20% 100%	NG, JK
Integrationstest		3	NG		80%	NG, JK
	Total:	32		Procentvis færdig:	81.25 %	

**Figur 9.1.** Sprint 2

Sprintet er afsluttet. De opgaver der ikke er færdige blev i det her eksempel videreført til sprint nummer tre. Billedet illustrerer også hvordan planlægningen er opbygget. Forklaring af statusfarver er vist på figur 9.2.

**Figur 9.2.** Forklaring af sprint points

### 9.2.2 V-model

**Figur 9.3.** V modellen

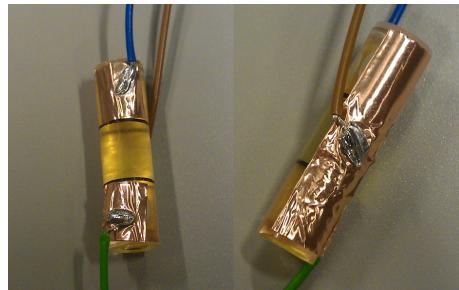
Vi har valgt at anvende V-modellen som udviklingsmodel. Dette muliggør iterative processer hvilket er optimalt for gruppens udviklingstil.

## 9.3 Analyse

### 9.3.1 Hvordan måler vi hældning?

Før vi kunne begynde på at lave en hældningssensor, blev vi nødt til at finde ud af hvilke muligheder, der eksisterer for hældningsmåling.

En af mulighederne var at anvende et pendul eller en libelle. Vi startede med at udvikle på en prototype af en libellesensor vist på figur 9.4. Den første hældningssensorprototype

**Figur 9.4.** Hældningssensor baseret på libelle

indeholder to højpasfiltrer, en frekvensgenerator og en differensforstærker. Vi kom frem til at libellesensoren har en kapacitet på omkring  $1 \times 10^{-15} [F]$ . Det gør det praktisk talt umuligt at anvende, da vores filter skulle designes til en cutoff frekvens  $> 3.0 [MHz]$ . Den høje

frekvens giver en stor selvinduktion i vores ledning. Et lavt signal fra hældningssensoren kombineret med stor selvinduktion, gjorde at vi måtte finde en anden løsning.

Næste prototype bestod af et potmeter og et pendul. Dog havde potmeteret en for stor friktionsmodstand, der gjorde det for upræcist i forhold til vores krav.

Vi har gennem et tredje semestersfag fundet ud af at PSoC'en indeholder et accelerometer<sup>1</sup>.

Efter en prototypeopbygning fandt vi ud af at accelerometeret opfyldte de krav, der er stillet i kravspecifikation.

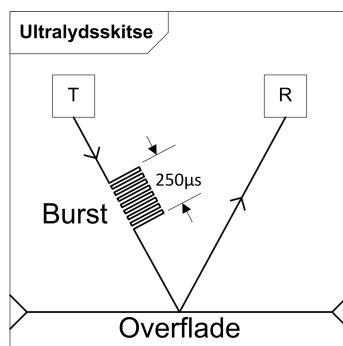
Softwaren er implementeret således, at der kan kalibreres for det offset, der er individuelt for hver PSoC.

### Hvordan mäter vi niveauet i vandballasttankene?

Da ingen i projektgruppen havde arbejdet med ultralyd før valgte vi for udfordringens skyld at forsøge med det. Man kunne have valgt at købe en færdiglavet enhed, men det blev vurderet at dette ikke gav ret meget faglig erfaring. Elektroniklaboratoriet lå inde med nogle rå tranducere og receivere, og gruppen ville derfor forsøge at udvikle en afstandsmåler baseret på ultralyd.

Der blev i starten lavet en del teknologiundersøgelse af ultralyd og tranducere og receivere blev testede for at se hvordan de aggerede. Vi fandt at der var en del destruktive reflektioner. En løsning var at fyde akustisk skum i toppen af tankene, men da det skum der blev anskaffet var elektrisk ledende kunne det ikke anvendes. Herudover blev der lavet forsøg med forskellige ultralydskredse fundet på Internettet, hvilket gav anledning til erfaring og viden.

Den valgte metode anvender bursts og kan ses på figur 9.5. Der er valgt et burst på  $250\mu s$ .



**Figur 9.5.** illustration af et burst til afstandsmåling

### 9.3.2 Hvordan skal de forskellige modulerne forsynes?

Da modulerne for eksempelvis kan være placeret ved styrbord, bagbord og midten af skibet, skal der overvejes hvordan de kan forsynes og derigennem overvejes hvilke muligheder der er for forsyningen. En tanke var om man kunne bruge et batteri eller om der skulle bruges en kabelt forsyning:

<sup>1</sup>Se bilag/datablade/KXSC7-2050

- Batteri:
  - Smart, da man undgår at trække en forsyningsledning.
  - Det kan være meget problematisk med et batteri, da der ikke vides hvor meget strøm der skal trækkes dvs. hvor stort det skal være både i kapacitet og fysisk størrelse.
- Kablet forsyning
  - Da der i forvejen er en kablet kommunikation, skal der alligevel kabel ud til modulet.

Ud fra de overstående overvejelser og tanker vil der arbejdes videre med kablet forsyning. Valget giver anledning til yderligere spørgsmål, for eksempel hvilken forsyningsspænding skal der være?

Da forsyningsspændingen på et skib (fx 480V) ligger langt over det der bruges i vores system, skal der bruges en transformator, der kan regulere spænding ned på 24V AC. De 24V AC kan føres ud til modularerne sammen med kommunikationen.

Ved modularerne kan der bygges en strømforsyning, der regulerer de 24V AC om til DC. Her kommer der to mulige strømforsyninger op:

- En SwitchModePowerSupply: *Effektiv, høj virkningsgrad.*
- En lineær strømforsyning: *Stabil, mellem virkningsgrad*

Da virkningsgraden ikke har afgørende betydning for systemet, samt erfaringen med SMPS ikke er stor, vil der tages udgangspunkt i en lineær strømforsyning.

Ved udviklingen af prototypen, kan der overvejes om der skal designes en eller flere strømforsyninger.

### 9.3.3 Raspberry Pi som host for Kontrolinterfacet

I den indledende fase var det gruppens ønske at kunne implementere Kontrolinterfacet på et Raspberry Pi-modul. Denne mulighed blev undersøgt, og det viste sig at det godt kunne lade sig gøre. Ved at anvende den nyeste beta-version af Qt kunne der skrives programmer til Raspberry Pi'en.

Versionerne imellem var der dog store forskelle på fundamentale områder af frameworkt; nogle af teknologiundersøgelserne måtte derfor foretages igen.

Det var også oprindeligt gruppens ønske at anvende I2C-protokollen mellem Kontrolinterfacet og Styringsmodulet. Denne protokol er også understøttet af Raspberry Pi, men kan kun implementeres med Python medmindre der selv udvikles headerfiler.

Dette var en større opgave end gruppen ønskede at prioritere den. Vi begyndte derfor at kigge efter andre protokoller.

Næste emne var rs232-protokollen, da den er kendt for gruppen. Protokollen viste sig at være god med Raspberry Pi. Efter nogle teknologiundersøgelser fandt gruppen også ud af hvordan rs232-kommunikationen kunne implementeres med den nye Qt-version.

Gruppen løb desværende igen i problemer, da der skulle kompiles til modulet. Her valgte gruppen at nedprioritere ønsket om at implementere Kontrolinterfacet på Raspberry Pi-modulet og dermed stoppede undersøgelserne.

### 9.3.4 Server

Opbygningen af serveren er gjort på baggrund af behovet for at kunne overføre data via et netværk. Til denne overførelse er TCP blevet valgt grundet erfaring med denne. Ved at benytte TCP kan der være flere end en der kobler til samtidig. TCP giver stabilitet og sikker dataoverførelse. Dette er prioriteret højt i prototypen, da systemet skal være sikkert under lastning og losning. Sikker dataoverførelse sikrer at hvis en pakke går tabt under overførelsen vil denne automatisk blive forsøgt sendt igen. Data vil således også komme frem i korrekt rækkefølge.

Fra starten var det meningen at Serveren skulle gemme data direkte til MySQL-databasen, men grundet problemer med at kompile MySQL-header og -driver blev dette efter noget tid droppet. I stedet blev backup-muligheden, der også er indskrevet i kravspecifikationen, fremtaget som eneste mulighed.

### 9.3.5 Valg af database

For at vælge database til systemet blev der kigge på MySQL, Microsoft Acces og en lagring til en tekstfil.

At lagre direkte i en tekst fil blev overvejet, da det er meget simpelt at lagre til tekstdfiler fra c++ og php kan håndtere at hente fra den. Det kan gå galt ved loading og lagring til filen samtidig med at formater kan blive et problem.

Der var i forvejen kendskab til MySQL og dette er et færdigudviklet system, der fungerer med webinterfaces og for programmeringssprog som C++ er der udviklet headere. Systemet er et system der fungere på mange platforme og løbende bliver udviklet på. På baggrund af dette blev dette system valgt.

### 9.3.6 Webinterface

Til den grafiske brugergrænseflade for Databasen blev et webinterface valgt for at flere på havneterminalen kunne tilgå data på samme tid. På baggrund af et middel kendskab til interpretere sproget php og dens integration med MySQL-databaser blev php valgt. Php giver desuden mulighed for at benytte ajax princippet og at bygge en hovedside der inkluderer de andre sider. I sidste ende vil dette valg give simplere vedligeholdelsen og mindske overførelsestiden til slutbrugeren.

Websitet er implementeret på en Apacheserver. Apacheserveren gør det muligt senere at højne sikkerheden på websitet.

### 9.3.7 Jura

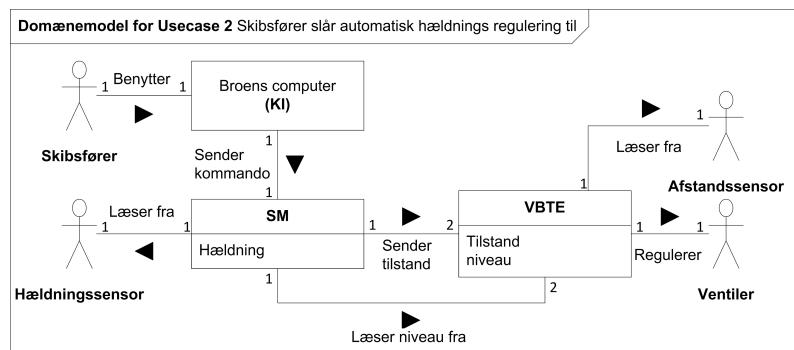
I startfasen blev de forskellige juridiske problemstillinger undersøgt. Dette gav grund til en del overvejelser omkring opbygningen af systemet BROS. Ansvarshavende personer samt krav til ballastvand er blevet overvejet. Et udpluk af de vigtigste begreber findes i *bilag for Jura*. For at få en uddybning af disse problemstillinger er Maersk A/S blevet kontaktet. Desværre har vi ikke fået noget svar. De juridske problemstillinger har blandt andet gjort at projektet blev indskrænket til kun at omhandle bulkskibe i nationalt farvand.

## 9.4 Systemarkitektur

I dette afsnit vil systemarkitekturen for projektet blive beskrevet. Systemarkitekturen tager udgangspunkt i dokumentationsdokumentet *Arkitektur*, og for nærmere uddybning henvises der til det dokument.

Systemarkitekturen er udarbejdet på grundlag af kravspecifikationen, og systemet som beskrevet i projektbeskrivelsen.

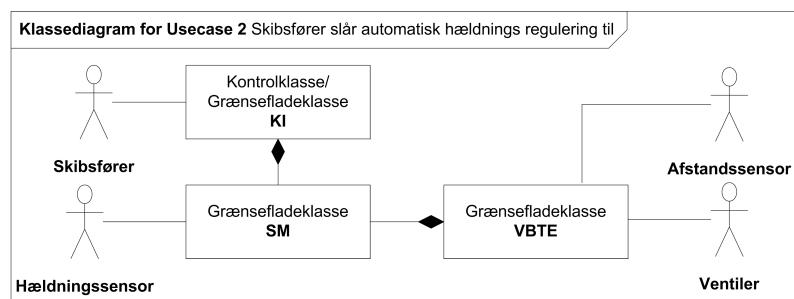
Systemarkitekturen starter med funktionaliteten beskrevet i use casene, og udvider så beskrivelsen til at omfatte de elementer af systemet, som brugeren ikke interagerer med. Dette afsnit vil give et eksempel på, hvordan en use case er blevet behandlet i systemarkitekturen. Use casen, der vil blive gennemgået, vil være use case 2: *Skibsfører slår automatisk hældningsregulering til*.



**Figur 9.6.** Domænemodel for use case 2

### Relationen mellem elementer og aktører

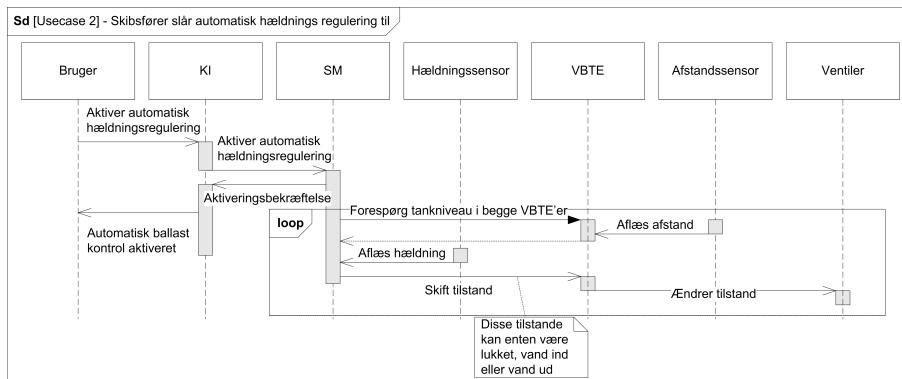
I domænemodellen på figur 9.6 beskrives relationen mellem systemets elementer og aktørerne. Disse enheder omdannes i klassediagrammet (figur 9.7) til konceptuelle klasser af en given type. Det ses at i use case 2 fungerer Kontrolinterfacet både som kontrolklasse og grænsefladeklasse. Kontrolinterfacet er kontrolklasse fordi det er initiativ- og beslutnings-tager til den efterfølgende udvikling i systemet. Både Kontrolinterfacet, Styringsmodulet og Vandballasttankenhederne fungerer som grænsefladeklasser, fordi alle tre klasser er i berøring med en aktør.



**Figur 9.7.** Klassediagram for use case 2

## Kommunikation og timing

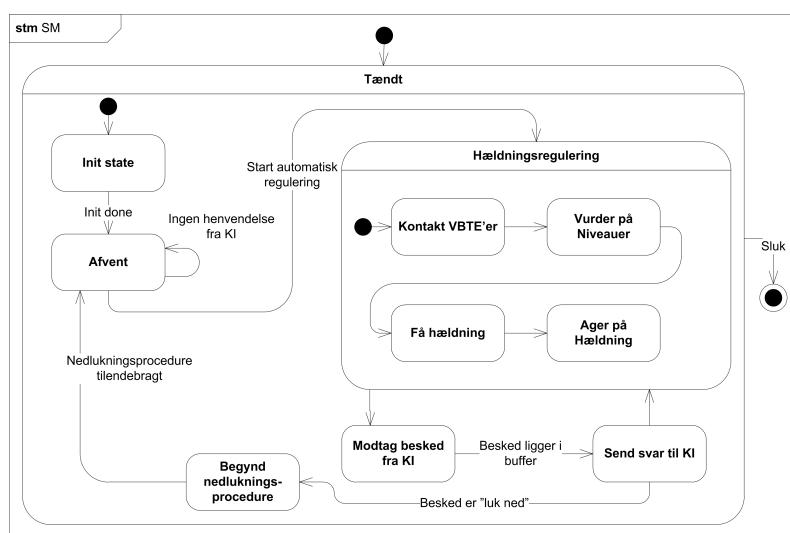
I sekvensdiagrammet (figur 9.8) beskrives kommunikationen og timingen imellem klasserne. Det ses at brugeren igangsætter processen, herefter videresender Kontrolinterfacet kommandoen og Styringsmodulet bekræfter modtagelsen. Styringsmodulet begynder så at regulere på vandniveauet i tankene ved hjælp af Vandballasttankenhederne. Reguleringen sker på grundlag af de målinger, der modtages fra hældningssensoren. Vandniveauet vil blive ved med at blive reguleret med det formål at opnå og derpå opretholde en hældning på nul grader.



**Figur 9.8.** Sekvensdiagram for Use Case 2

### 9.4.1 Stadier for konceptuelle klasser

Når alle use cases er blevet bearbejdet som use case 2 er blevet i figur 9.6, 9.7 og 9.8 har gruppen dannet state machines for de konceptuelle klasser i systemet. Her visualiseres hvilke stadier, klasserne har brug for at gennemgå i løbet af klassens levetid for at opfylde kravene sat i kravspecifikationen. Som et eksempel på en sådan state machine, vises state machinen for Styringsmodulet på figur 9.9.



**Figur 9.9.** State machine for Styringsmodulet

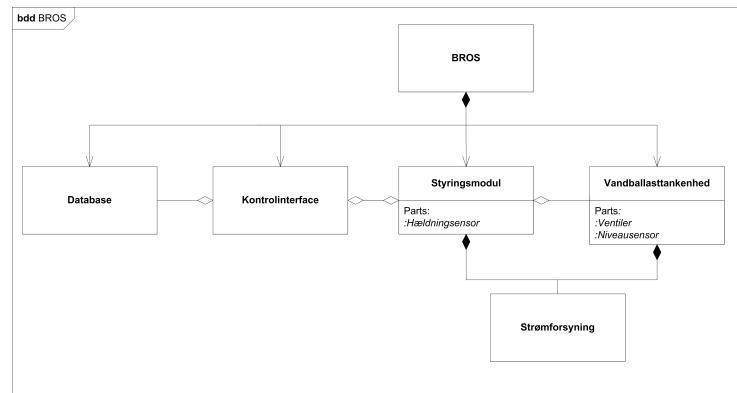
Flowet i state machinen bygger på det beskrevne i sekvensdiagrammerne. Starten på use case 2 er et brugerinput til Kontrolinterfacet. Dette medfører en besked fra Kontrolinterfacet til Styringsmodulet. Styringsmodulet kan være i to forskellige stadier når den modtager beskeden:

Hvis programmet er nyopstartet, vil beskeden få Styringsmodulet ud af *Afvent*-stadiet og til at aktivere automatisk hældningsregulering. Hvis programmet befinner sig i *Hældningsregulering*-stadiet med reguleringstypen *Manuel*, vil beskeden fra Kontrolinterfacet få Styringsmodulet over i stadiet *Modtaget besked fra KI*. Styringsmodulet besvarer beskeden med en aktiveringsbekræftelse, og returnerer derpå til *Hældningsregulering*-stadiet - nu med reguleringstypen *Automatisk*.

Dette var for use case 2. Styringsmodulets opførelse for alle use cases kan forudsiges på samme måde ud fra figur 9.9.

### Fra konceptuelle klasser til system

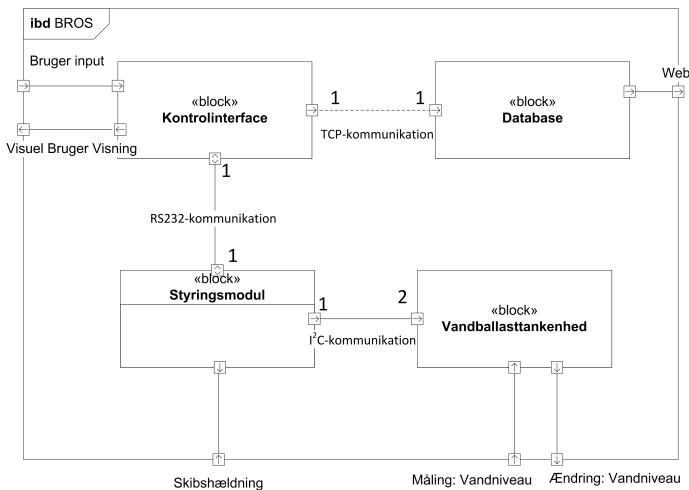
Næste trin er, at omdanne de konceptuelle klasser til et regulært system. Nogle konceptuelle klasser kan grupperes, andre står for sig selv. Den fysiske opbygning af systemet, bliver så vist i et blokdefinitionsdiagram. Blokdefinitionsdiagrammet for systemet kan ses i figur 9.10. Her ses det, at systemet ender ud med fire hovedblokke: Kontrolinterfacet, Styringsmodulet, Vandballasttankenhed og Database. De konceptuelle klasser, der ikke er endt som hovedblokke, er i stedet for blevet til underblokke af enten styringsmodulblokken eller vandballasttankenhedblokken.



**Figur 9.10.** Blokdefinitionsdiagram for systemet

### Kommunikationsveje i systemet

Herefter er opgaven to-delt. Kommunikationsvejene skal kortlægges både internt mellem blokkene og for inputs og outputs af systemet. Begge dele gøres i et internt blokdiagram. Det interne blokdiagram for systemet kan ses i figur 9.11. Her angives det også hvor mange gange de enkelte blokke skal optræde i systemet.



**Figur 9.11.** Internt blokdiagram for systemet

Som det kan ses i figur anvendes der en række protokoller til kommunikation mellem systemets interne blokke. Disse protokoller skal defineres inden design og implementering af systemets blokke kan begynde.

Som eksempel vil der blive givet kommunikationsprotokollen mellem Vandballasttankenherne og Styringsmodulet. Protokollen er beskrevet i tabel 9.4 med indstillingerne angivet i tabel 9.3.

#### Forbindelsesindstillinger

Data Rate (kbps)	100
VBTE1-Adresse	2
VBTE2-Adresse	3
Bytes sendt	1

**Tabel 9.3.** Forbindelsesinstillinger I2C

#### Protokoloversigt

Navn	Værdi	Beskrivelse
VBTE1NIVEAU	5	Sendes når SM ønsker at aflæse vandstandsniveauet
TOPVENTIL	6	Sendes når VBTE skal åbne for topventilen (og dermed lukke for bundventilen)
BUNDVENTIL	7	Sendes når VBTE skal åbne for bundventilen (og dermed lukke for topventilen).
LUKVENTIL	8	Sendes når VBTE skal lukke for begge ventiler.

**Tabel 9.4.** Protokoloversigt for I2C

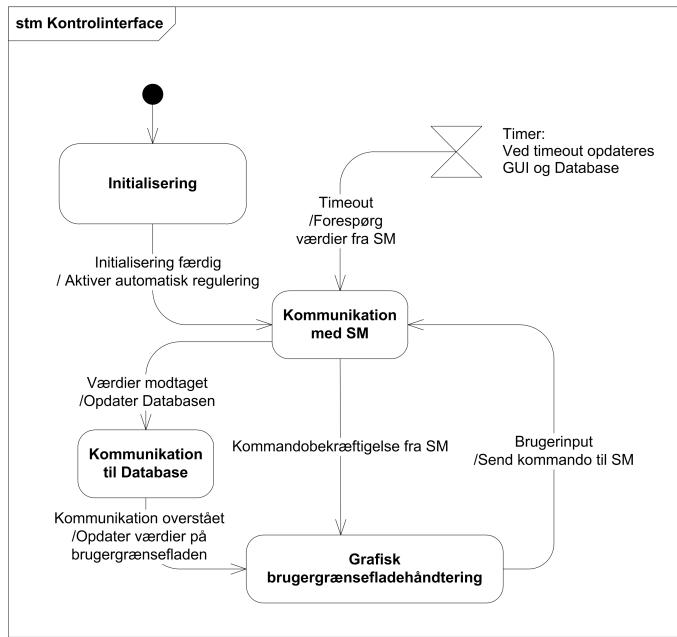
Systemet er nu opdelt i fysiske blokke. For hver blok er der angivet en state machine og kommunikationsprotokollen imellem blokkene er beskrevet. Designfasen kan påbegynde.

## 9.5 Design og Implementering

### 9.5.1 Kontrolinterfacet

I dette afsnit beskrives designet og implementeringen af Kontrolinterfacet som det er lavet på baggrund af kravspecifikationen og systemarkitekturen.

Opbygningen af Kontrolinterfacets design tager udgangspunkt i den state machine, der er udarbejdet i systemarkitekturen, se figur 9.12.



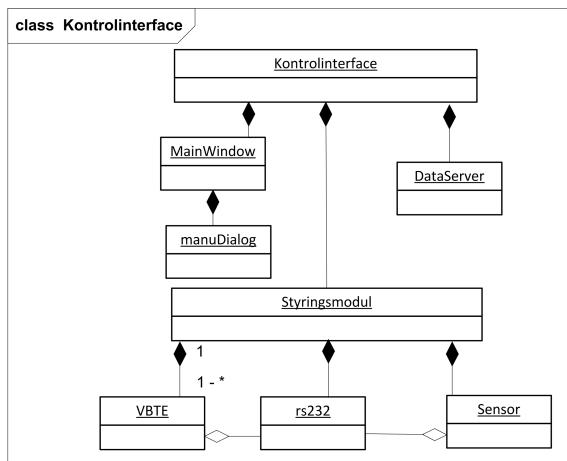
**Figur 9.12.** Den simplificerede state machine for Kontrolinterfacet fra Systemarkitekturen

State machinen er god til at få en forståelse for programmet, men for at komme videre må den uddybes væsentligt. Det første trin i designfasen har derfor været at udarbejde en mere detaljeret state machine. Resultatet kan ses i Kontrolinterfacets afsnit i dokumentationsdokumentet *Detaljeret Softwaredesign*.

Det næste skridt er at udarbejde et klassediagram for programmet. Klassediagrammet er ligesom state machinen først lavet i en simplificeret version og dernæst i en detaljeret. Det første kan ses på figur 9.13 med en beskrivelse af hver klasse i tabel 9.5.

Klassediagrammet på figur 9.13 afspejler et program designet med forbillede i det samlede system. Dette er gjort for at trække på de løsninger, der er fundet i forbindelse med det fulde system.

Det ses hvordan hovedklassen er Kontrolinterfacet. Alt andet oprettes enten direkte af Kontrolinterface-klassen eller af en klasse oprettet af samme. Kontrolinterfacet har kun kontakt til brugergrænsefladen (MainWindow), Styringsmodulet og DataServer. Kommunikationen mellem Kontrolinterface-klassen og Styringsmodul spejler den kommunikation, der er imellem de fysiske blokke Kontrolinterfacet og Styringsmodulet i det samlede system.



**Figur 9.13.** Det simple klassediagram for Kontrolinterfacet. Beskrivelse af hver klasse i tabel 9.5

### Kontrolinterfacets klasser

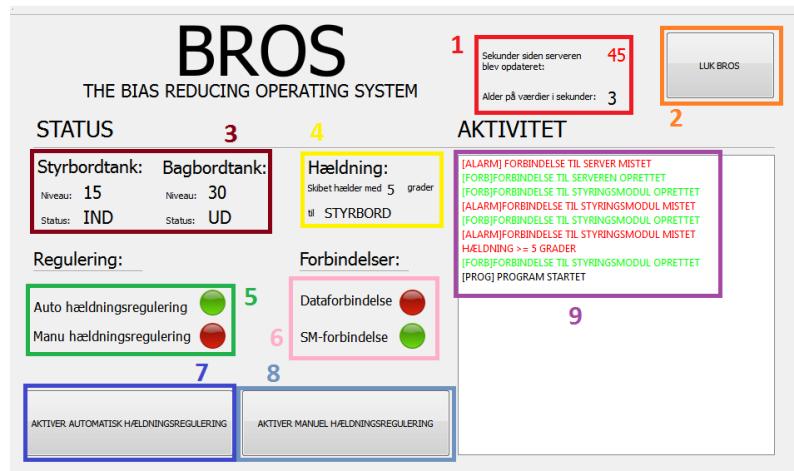
Kontrolinterface:	Programmets hovedklasse. Eksisterer for at rydde op i main-funktionen.
DataServer:	Står for alt TCP-kommunikationen med databasen. Oprettes af KI-klassen
Styringsmodul:	Oprettes af KI og opretter VBTE-, Sensor- og RS232klasserne.
Sensor:	Oprettes af SM og er ansvarlig for hældningsværdien.
VBTE:	Der eksisterer et VBTE-objekt for hvert fysisk VBTE-modul. Det er objektets ansvar at holde styr på værdierne for sit VBTE-modul.
RS232:	Objektet oprettes af SM-klassen og VBTE- og Sensorobjekterne har en delt association til den. Objektet har ansvaret for kommunikationen med det fysiske SM-modul. Objektet formidler sig på en protokol forstået den kommunikationsansvarlige kode på SM-modulet. Protokollen kan ses i dokumentet for det detaljerede softwaredesign.
MainWindow:	Oprettes af KI-klassen. Kontrollerer og overvåger den grafiske brugergrænseflade.
manudialog:	Oprettes af MainWindow og styrer den dialog, der fremkommer når man ønsker en manuel hældningsregulering.

**Tabel 9.5.** Kontrolinterfacets klasser

Det gælder for VBTE-, Styringsmodul- og Sensor-klasserne at når der efterspørges en af de værdier, klassen har ansvaret for, så benyttes RS232-objektet til at fremskaffe disse værdier ved hjælp af den serielle kommunikationsprotokol.

### Grafisk brugergrænseflade

I denne sektion vil der kort blive gennemgået det vigtigste vindue i den grafiske brugergrænseflade; hovedvinduet. En gennemgang af de resterende vinduer vil kunne findes i *Softwaredesign Appendix A: Kontrolinterface*. På 9.14 er hovedvinduet vist og vinduets elementer indrammet. Herefter vil hovedvinduets elementer blive beskrevet.



**Figur 9.14.** På figuren ses Kontrolinterfacets programs hovedvindue

### 1: Forsinkelse i sekunder

Det øverste tal fortæller tiden i sekunder siden serveren sidst er blevet opdateret succesfuldt. Nedenunder udskrives tiden i sekunder siden værdierne i boks tre og fire er blevet opdateret. Hvis der er gået over 10 sekunder bliver tallet rødt.

### 2: Nedlukningsknap

Anvendes til at lukke programmet. Programmet åbner en dialog som kan ses i *Designdokumentet, Appendix A*.

### 3: Vandballasttankene

Her kan status for vandballasttankene aflæses. Niveauet fortæller hvor fyldt tanken er, angivet i procent. Hvis niveauet er over 70% skrives tallet med rødt. Status angiver vandets flow i tanken: IND/UD/LUKKET.

### 4: Hældningssensor

Værdien for hældningen af skibet angives i antal grader og til hvilken side skibet hælder.

### 5: Regulatingsstatus

Her angives hvorvidt automatisk eller manuel hældningsregulering er aktiveret. Der vil altid kun være én af disse aktiveret. Derfor vil der altid være en rød og en grøn indikator tændt. I dette eksempel er den automatisk hældningsregulering aktiveret.

### 6: Forbindelser

Indikerer hvorvidt der er forbindelse til Styringsmodulet og Server. Dataforbindelsens indikator er rød hvis det ikke lykkedes at oprette forbindelse til serveren ved sidste forsøg. SM-forbindelse er rød hvis det ikke lykkedes at få de ventede svar fra Styringsmodulet. I denne situation er der forbindelse til Styringsmodulet, men ikke Serveren.

### 7: Automatisk reguleringsknap

Ved tryk på denne knap vil man komme til en dialog, hvor man bedes bekræfte øsnket om at gå væk fra manuel hældningsregulering. Hvis automatisk regulering allerede er aktiveret vil man få dette af vide istedet. Dialogerne kan ses i *Designdokumentet, Appendix A*

## 8: Manuel reguleringsknap

Frembringer en dialog hvori hældningen vælges fra 0.5 grader til 2.5 grader. Herefter vælges siden skibet skal hælde til. Dialogen kan ses i *Designdokumentet, Appendix A*.

## 9: Aktivitetslog

Her udskrives vigtige hændelser i programmet med farvekoder. I dette eksempel kan det ses hvordan alarmer skrives med rødt og oprettede forbindelser skrives med grønt.

### 9.5.2 Vandballasttankenhed

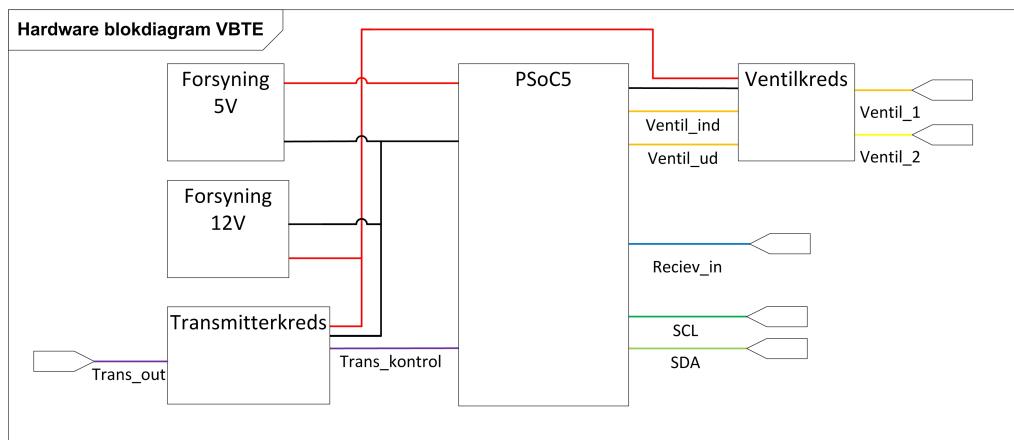
I dette afsnit beskrives design og implementering af Vandballasttankenhederne for både software og hardware.

#### Hardware

Til VBTE'en er der blevet designet hardware til at styre de to ventiler samt den keramiske ultralydstrmitter og receiver. Hardware er blevet udfærdiget i to dele. Den ene er programmeret hardware på PSoC'en, den anden er hardware uden for PSoC'en. Designprocessen for hardware har gennemgået tre faser:

1. Overordnet design
2. Nedbrydning af blokke
3. Opbygning af design

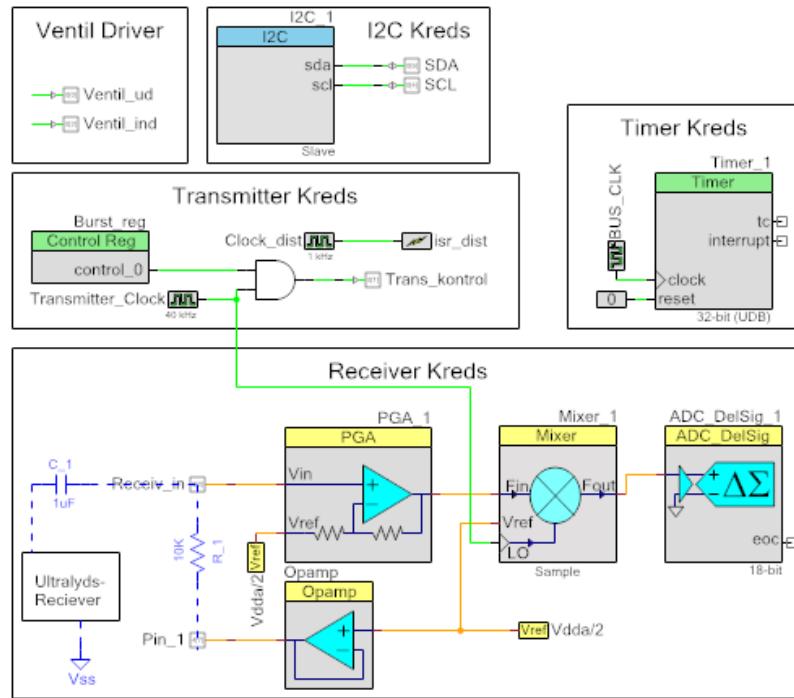
Gennem disse faser er designet blevet udfærdiget. Fremgangsmåden er anvendt for at overskueliggøre systemet og lette arbejdet ved at dele systemet op i mindre dele. På figur 9.15 ses det overordnede design af VBTE'en. Der vil i rapporten tages udgangspunkt i ventilkredsen samt receiverdriveren på PSoC'en.



**Figur 9.15.** Illustrering af overordnet design af hardware på VBTE.

#### Hardware på PSoC'en

Hardware opbygget på PSoC'en kunne uden problemer være blevet designet uden for PSoC'en, men PSoC-miljøet gør det meget nemt at arbejde med mange forskellige elementer. Der vil i rapporten lægges vægt på transmitterkredsen samt receiverkredsen. På figur 9.16 ses hardwaredesignet på PSoC'en.



Figur 9.16. Hardware på PSoC'en.

### Transmitterkreds

Transmitterkredsen står for at sende burst samt at tme hvert burst. Dette opnåes med to clocks, en AND gate, en output pin samt et kontrolregister. Transmitterclocken er indstillet til 40kHz da ultralydstransmitteren, ifølge databladet<sup>2</sup>, opererede ved  $40\text{ kHz} \pm 1\text{ kHz}$ . Clocken er blevet målt på oscilloskop til  $40,3\text{ kHz}$ . Burst kontrolregisteret er anvendt til at AND'e clocken ud på trans\_kontrol pinden.

Clock\_dist interruptrutinen sørger for at tælle en variabel op, der anvendes i main til at kalde burst funktionen. Dette gøres for at lave et nonblocking delay så andet kan køres på PSoC'en mens et burst er sendt afsted og der afventes detektion.

### Receiverkreds

Receiverkredsen modtager signalet fra ultralydsreceiveren og omsætter det til en detektion. Dette sker efter følgende opskrift:

Løfte signalet → Forstærkning → Mixning.

Signalet bliver løftet op til  $\frac{V_{dd}}{2}$  fordi PSoC'en ikke kan arbejde med værdier under Vssa (GND). Dette gøres ved hjælp af en kondensator, en modstand og en spændingsfølger med  $\frac{V_{dd}}{2}$  på det positive ben.

Efter signalet er løftet, bliver det forstærket op af en PGA. Den første forstærkning, fundet i teknologiundersøgelsen, viste sig at være for lav og forstærkningen blev justeret til 32. Der ligger her et potentieligt problem, da et meget klart signal vil kunne få systemet til at gå i mætning. Det ser dog ikke ud til at ske. Situationen virker også usandsynligt i vores opstilling da lyden bliver dæmpet af vandet.

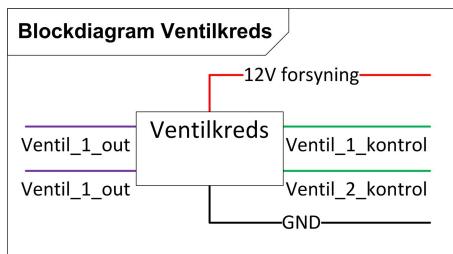
Herefter bliver signalet mixet med 40 kHz og filtreret. Efter mixeren giver det, stort set,

<sup>2</sup>Se bilag/datablade/400SRT 160

en DC og summen af de to signaler. Filteret er indbygget i Delta-Sigma ADC'en og har en knækfrekvens ved  $\frac{\text{Samplefrekvens}}{3}$ . For at være sikker på at det meste af signalet er kommet over regnes der en opladningstid på filteret til 1/4 ms. Efter undersøgelser af signalet ind på ADC'en blev en spænding på 0,3 V valgt til at repræsentere en detektion.

### Hardware uden for PSoC'en

Uden for PSoC'en er der lavet 2 hardwareblokke (Ventilkreds og transmitterkreds). Disse har til ansvar at omsætte et kontrolsignal til en større spænding over de respektive enheder. Herunder vil der blive fokuseret på ventilkredsen.



**Figur 9.17.** Hardwareblok for ventil

### Ventilkreds

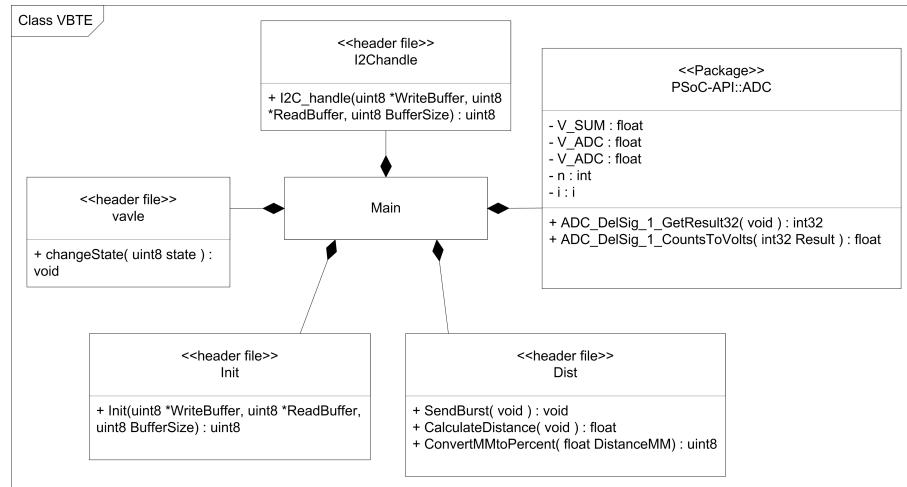
Ventilkredsen får to kontrolsignaler fra PSoC'en og disse skal styre de to ventiler. Ventilerne monteret på kredsen er fra Danfoss og er af modellerne EV210A-1.2 og EV210A-4.5. Disse ventiler drives ved 12V 0.4A. Der er valgt en BD139 transistor til at forstærke signalet op. Denne har en forstærkning mellem 40 og 160 (aflæst fra databladet<sup>3</sup>). IO's på PSoC'en kan maksimalt trække en strøm på 4 mA og det vil i værste tilfælde ikke være nok til at trække ventilerne.

Der er derfor lavet en darlingtonkobling for at mindske belastningen af PSoC'en og for at sikre, at der bliver lukket nok op for transistoren. Dette er gjort med en transistor af typen BC547. Transistorne er valgt ud fra pris og tilgængelighed. Der var først valgt en MOSFET IRLZ44n, men denne var både for dyr og kunne klare en unødvendig stor effekt. Det smarte ved MOSFET'en er dog at den er spændingsstyret i forhold til de to andre transistorer.

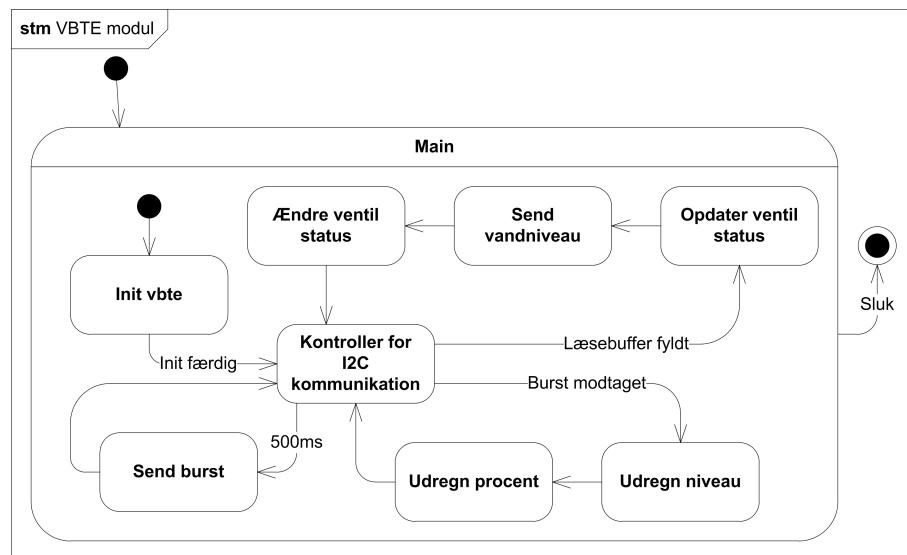
### Software

Softwareen til VBTE-modulet er blevet designet til at kunne opfylde kravene i kravspecifikationen samt arkitekturen i systemarkitekturen. Den er blevet designet til at passe til hardwaren så den kan kontrollere de enkelte elementer. I designfasen er der blevet udfærdiget et klassediagram samt en statemachine over funktionaliteten. På figur 9.18 ses klassediagrammet for VBTE-modulets software.

<sup>3</sup>Se bilag/datablade/BD139.pdf

**Figur 9.18.** Klassediagram over VBTE

Selvom softwareen er skrevet i C er der lavet klasser for alle h-filer for at gøre systemet overskueligt. For detaljerede metodebeskrivelser henvises til detaljeret software design. For at overskueligtgøre funktionaliteten af softwaren anvendes statemachinediagrammet. Dette viser de forskellige tilstande programmet til enhver tid kan befinde sig i.

**Figur 9.19.** State machine for software på VBTE'en

De forskellige betingelser der skal til for at skifte tilstand vil herefter blive beskrevet. Som nævnt i hardware anvendes et interrupt til at lave et nonblocking delay på 500ms. Dette håndteres på VBTE'en i main ved hjælp af en *if*-sætning der tjekker op på variablen fra interruptet.

*Burst received* bliver opnået når flaget hertil bliver sat. Dette flag bliver sat inde i ADC'ens interruptrutine når den måler en værdi der indikerer at et burst er modtaget.

*Read complete*-betingelsen bliver kontrolleret inde i I2C\_Handle. Heri aflæses data fra Styringsmodulet og omsættes til et state for ventilerne. Herefter fyldes afstanden for

tankene ind i readbufferen for I2C'en med henblik på at skulle afhentes af Styringsmodulet. Herefter ændres ventiltilstanden så den passer med den modtagede tilstand.  
For yderligere detaljer om software for VBTE henvises til detaljeret design dokumentation.

### 9.5.3 Styringsmodulet

I dette afsnit beskrives design og implementering af Styringsmodulet. Styringsmodulet består af både software og hardware.

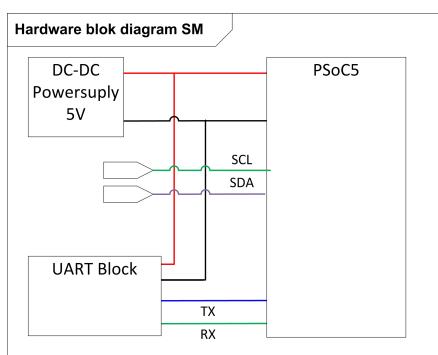
#### Hardware

Styringsmodulet hardware består af en konverteringskreds og en PSoC. På PSoC'en er monteret et Kionix KXSC7-2050 accelerometer. Konverteringskredsen anvendes til at sende UART signaler fra PSoC til Kontrolinterfacet. Accelerometerets x-akse anvendes til hældningsmålinger for hældningssensorblokken.

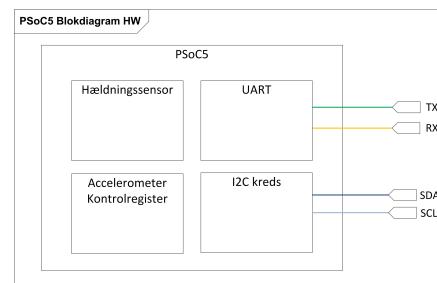
Designfasen til Styringsmodulet er delt op i 3 faser:

1. Overordnet design
2. Nedbrydning af blokke
3. Opbygning af design

Denne fremgangsmåde gør det muligt for en udefrakommende at følge med i processen og at kunne implementere modulet så det kan opfylde de krav, der er opstillet. Ligeledes gør fremgangsmåden det lettere at overskue flere løsninger til hvert problem. På figur 9.20 ses det overordnede design og på figur 9.21 ses PSoC-blokken i Styringsmodulet. Der bliver efterfølgende taget udgangspunkt i hældningssensoren.

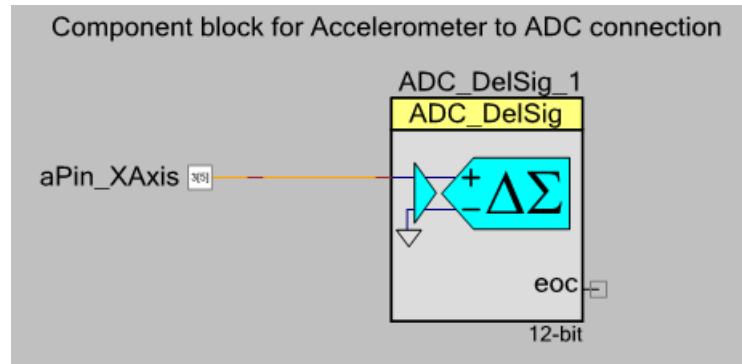


**Figur 9.20.** Hardware blok for SM



**Figur 9.21.** PSoC blok for SM

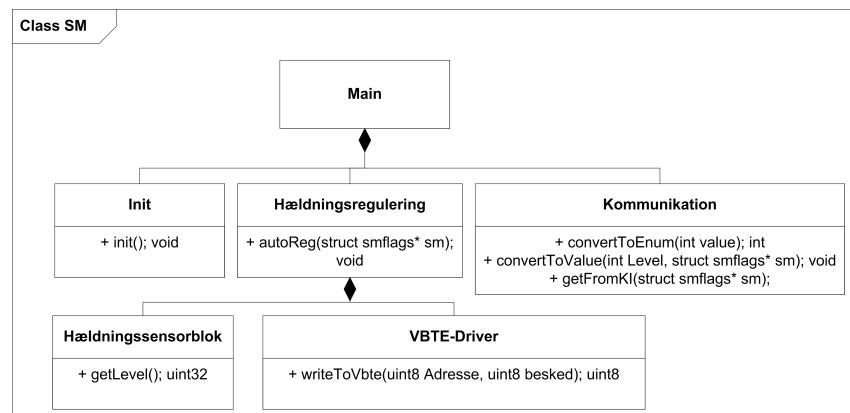
Hældningssensoren består af to komponenter: det førnævnte accelerometer samt en DelSig ADC internt i PSoC'en. Valget af accelerometer kommer af at have lavet en række prototyper, der ikke opfyldte kravene, hvilket accelerometeret i PSoC'en gjorde. Valget af ADC faldt på en DelSig da den er meget støjimmun grundet det indbyggede lavpasfilter og den høje oplosning. Valgte komponenter er illustret på figur 9.22. ADC konverterer det analoge signal fra, en pin forbundet til, accelerometer til en digital værdi der så senere bliver anvendt i softwaren. For ADC og accelerometerets opsætning se da *Detaljeret hardwaredesign* i bilag.



**Figur 9.22.** Hældningssensorens implementering

## Software

Styringsmodulets software behandler den konverterede data fra ADC'en samt kommunikation med Vandballasttankenhederne og Kontrolinterfacet. Sofwarens udvikling har fulgt de samme faser som hardwaren, hvilket har ført til letforståelig og læselig kode. Softwaren er illustreret på figur 9.23. Der vil efterfølgende blive taget udgangspunkt i et state machine samt funktionen `getFromKI`.



**Figur 9.23.** Klassediagram for SM

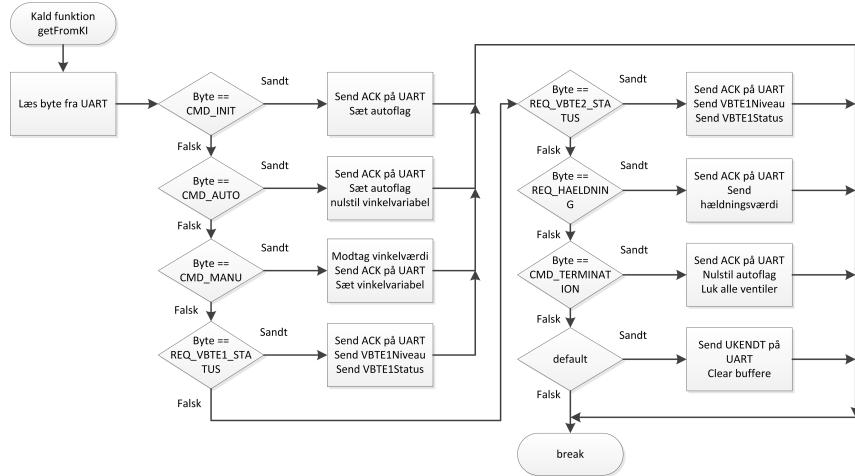
## Styringsmodulets funktionalitet

Funktionalitet af Styringsmodulet kan bedst beskrives med et state diagram. På figur 9.9 ses state machine diagrammet for Styringsmodulet. Diagrammet indeholder *Hældningsregulering*-stadiet, der beskriver den automatiske og manuelle regulering i systemet.

Den automatiske regulering styres med et flag. Den manuelle styres med en værdi, der bliver sat. Flaget bliver sat når Styringsmodulet opnår forbindelse med Kontrolinterfacet i *Afvent*-stadiet. Flaget bliver nulstillet hvis Kontrolinterfacet sender besked til Styringsmodulet om at det skal termineres, hvorefter Styringsmodulet går tilbage i *Afvent*-stadiet. Behandlingen af beskeder fra Kontrolinterfacet i *Modtag besked fra KI*-stadiet og *Send svar til KI*-stadiet ses i følgende afsnit `getFromKI`.

## getFromKI

Funktionen er bedst beskrevet med et flowdiagram:



**Figur 9.24.** Flowdiagram for funktionen getFromKI

Funktionen har til ansvar at kommunikere med Kontrolinterfacet efter de krav opsat i Kravspecifikation. Hver ACK er relateret til den besked, der er modtaget, så Kontrolinterfacet ved hvilken besked, Styringsmodulet har modtaget. Dette sikre pålidelig overførsel da Kontrolinterfacet har mulighed for at validere på overførslen. getFromKI bliver kaldt cirkla hver andet sekund for at se om Kontrolinterfacet har skrevet noget til bufferen. Hvis der ikke er noget i bufferen returnerer funktionen med det samme.

### 9.5.4 Strømforsyning

I dette afsnit beskrives design og implementering af strømforsyningen. Strømforsyningen implementeres til Styringsmodulet og Vandballastankenhederne.

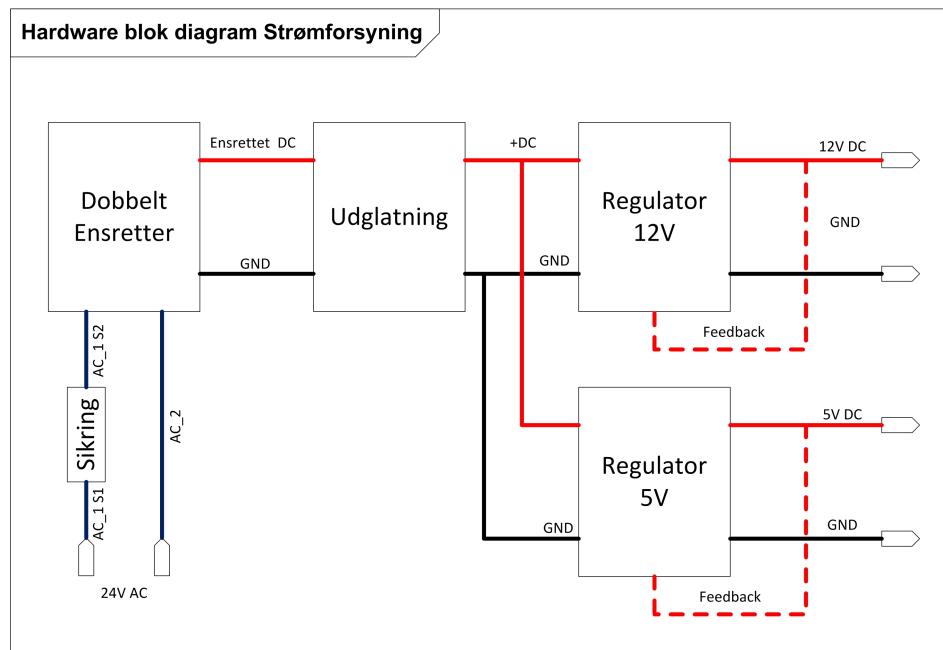
#### Hardware

Strømforsyning er designet til at levere to forsyningsspændinger: **12VDC 1A** og **5VDC 0.5A**. Dette gøres fra en 24V AC forsyningskilde.

For at gøre designet af strømforsyning mere overskueligt er designfasen delt op i tre faser:

1. Overordnet design
2. Nedbrydning af blokke
3. Opbygning af design

Igennem de tre faser er designet af strømforsyning blevet udarbejdet. Dette er gjort ved at have startet med et overordnet blokdiagram, hvorefter hver enkel blok er blevet beskrevet og designet. Figur 9.25 viser det overordnede blokdiagram for strømforsyningen.



**Figur 9.25.** Illustrering af overordnet blokdiagram af strømforsyningen.

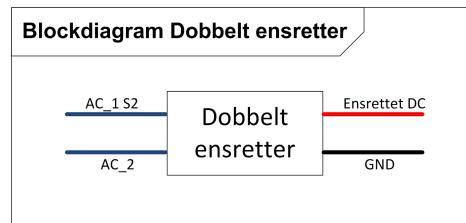
### Beskrivelse af vitale hardwareblokke

I dette afsnit vil der være en beskrivelse af de udvalgte blokke.

#### Dobbelt-ensretter og udglatning

Da strømforsyningen får leveret strøm fra en AC-spændingskilde skal AC-signalet ensrettes til DC. Dette gøres med dobbelt-ensretterblokken og udglatningsblokken.

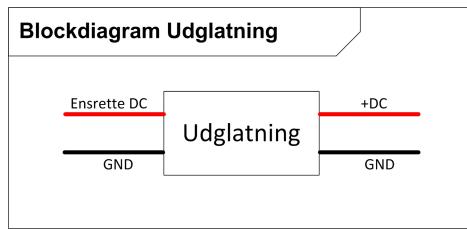
#### Dobbelt-ensretter



**Figur 9.26.** Blok for dobbelt-ensretter

Dobbelt-ensretteren er lavet som en diodebro bestående af fire dioder, hvor to af dioderne leder samtidigt. Dette medfører at der vil være positive halvperioder på udgangen *ensrettet DC* med en spidsværdi på cirka indgangsspændingen \*  $\sqrt{2}$

## Udglatning

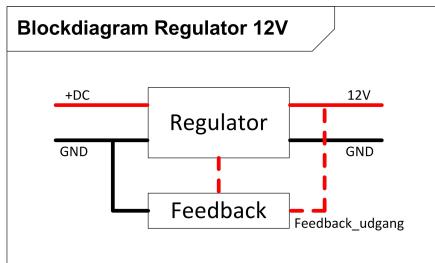


**Figur 9.27.** Blok for dobbelt ensretter

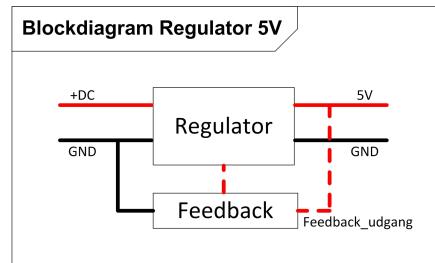
Udglatningen er en elektrolytkondensator som oplades og aflades i forhold til de positive halvperioder. Da kondensatoren kun når at aflade lidt for hver nedadgående periode, vil der være en jævn og stabil DC.

## Regulator 12V og Regulator 5V

Da de enkle moduler bruger 12V og/eller 5V skal DC-spændingen fra udglatningen reguleres ned til et pasende niveau. Dette gøres i blokkene regulator 12V og regulator 5V.



**Figur 9.28.** Blok for 12V regulator



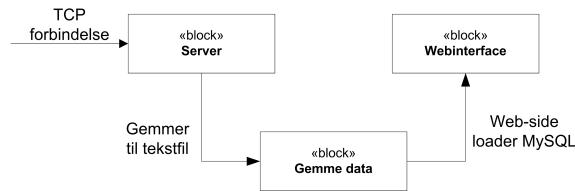
**Figur 9.29.** Blok for 5V regulator

For at regulatoren kan regulere spændingen ned til 12V og 5V, bruges et feedback fra udgangen. Feedbacket er indstillet til 12V ved hjælp af to modstande.

### 9.5.5 Database

Denne section beskriver design og implementering af databasedelen. Fasen er lavet på baggrund af kravspecifikationen og systemarkitekturen.

Databasen, som vi har defineret den, består af en MySQL-database, Server og Webinterface.

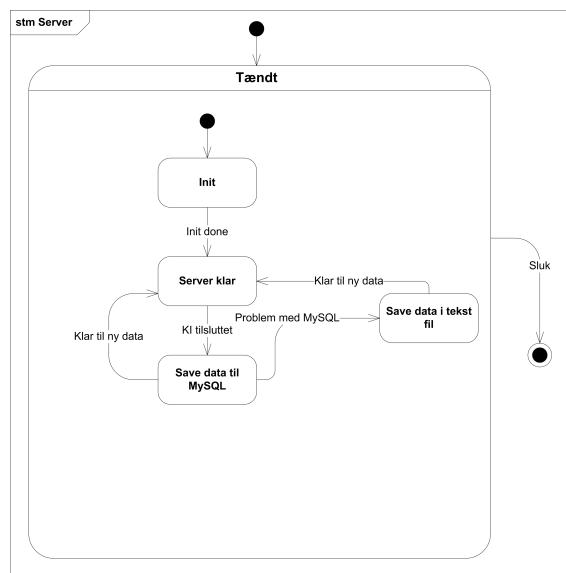


**Figur 9.30.** Illustrerer overordnet hvordan databasen fungerer

### 9.5.6 Serveren

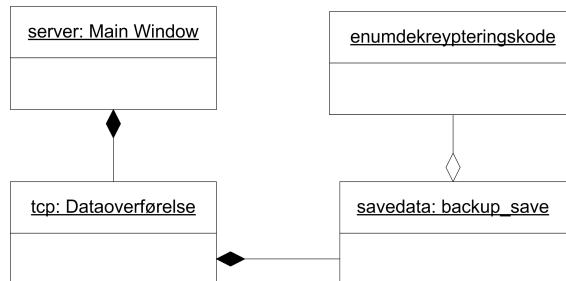
Denne section beskriver design og implementering af serveren.

Opbygningen af serveren er lavet så hvert element i serveren er implementeret som en klasse. Dette er illustreret på figur 9.32. For at få forståelse for flowet i programet er der udarbejdet en state machine. State machinen kan ses på figur 9.31.



**Figur 9.31.** Illustrerer overordnet hvordan databasen fungerer

Det gælder at når Kontrolinterfacet forsøger at kontakte databasen, tilsluttes denne via en TPC-socket. Når tilslutningen er gjort kan der overføres data fra Kontrolinterfacet til databasen. Data bliver gemt i en tekst fil, der indlæses af Webinterfacet.



**Figur 9.32.** Overordnet illustration af serverens funktionalitet

## Serverens klasser

Server	Er hovedvinduet på serveren. Information til brugeren skrives her og knapper er implementeret.
TCP forbindelse (tcp)	Åbner en port som Kontrolinterfacet har mulighed for at forbinde til databasen på.
Gemme data (sa- vedata)	Gemmer data til en tekstfil.
Dekryptering (enumDeKrypte- ringskode)	Dekrypterer <i>level</i> så værdien bliver i grader

**Tabel 9.6.** Serverens klasser

## Hovedvindue for serveren

Dette giver et kort billede på hvordan serverens hovedvindue ser ud. Den fulde model kan ses i *Detaljeret Softwaredesign, Appendix B: Database*. Serverens hovedvindue giver mulighed



**Figur 9.33.** Billede af serverens hovedvindue

for at brugeren kan se at serveren kører og hvilke kommandoer der er blevet udført. Fra serverens hovedmodul har brugeren mulighed for at vælge at åbne den web-baserede database. Der er en mulighed for at lukke serveren. Denne er ment som en mulighed for at lukke serveren i tilfælde af fejl eller restart. Ellers er det meningen at serveren skal køre konstant. Serverens grafiske visning er ikke til brug for terminaloperatøren, men ment for at man kan se at serveren køre i stedet for at få udskrifter i terminalen.

### 1: Info

Brugeren får her udskrevet beskeder om serveren. Brugeren kan se om Kontrolinterfacet har tilsluttet sig og om der er blevet overført data. Desuden vil fejlmeddelelser blive udskrevet her. Dialogboksen er lavet sådan at den nyeste besked altid står øverst, se figur 9.33.

### 2: Database

Anvendes til at lade brugeren tilgå den web-baserede database. Efter login og valg af skib kommer brugeren til databasen for det valgte skib. Se 9.35.

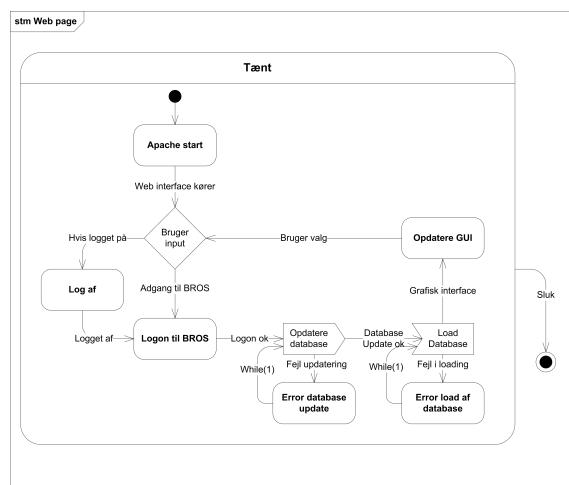
## 2: Luk serveren

Anvendes til at lukke programmet. Bruges til nedlukning

### 9.5.7 Webinterface

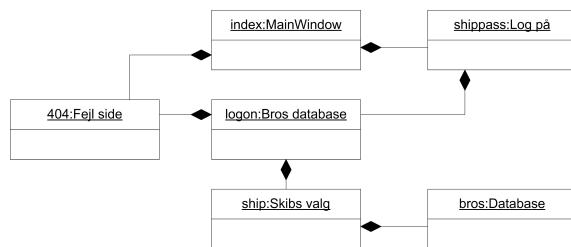
Denne section beskriver design og implementering af web-siden.

Opbygningen af Webinterface er gjort i php hvor dette gav mening. Desuden er html og styles (css) blevet benyttet til at sikre en identisk stil på hele web-siden og for simpelt at kunne vedligeholde siden. For at sikre mindst data transmission imellem computer og server er princippet i ajax blevet benyttet. Til at fremvise et grafisk program flow benyttes der en state machine figur 9.34



**Figur 9.34.** Illustrere overordnet hvordan databasen fungere

Når havneterminalen ønsker at se data om skibe, der er opkoblet på BROS kan denne tilgå disse via den web-baserede database. Når serveren kører kan brugeren tilgå databasen via knappen "Database" eller ip-adressen. Dette giver mulighed for at tilgå siden via Internettet hvorved behovet for adgangskode ved login bliver nødvendig for at sikre data imod uautoriseret brug. Efter login på siden, kan brugeren vælge, hvilket skib, der skal benyttes. Når skibet er valgt kommer brugeren ind på den valgte database. Her kan brugeren se ID på skibet vandtilstand i ballasttanke, hældning og tid siden sidste opdatering (for nærmere se *Detaljeret Softwaredesign, Appendix B*). Webinterfacet checker hvert 5. sekund om der er kommet ny data. Ved ny data vil et script sørge for at dette bliver gemt i MySQL-databasen. Webinterfacet tilgår den angivne tabel i den angivne database. Til at håndtere wibinterfacet benyttes en Apache-server som er installeret på serveren. Apache-serveren kan benyttes med de fleste operative systemer.



**Figur 9.35.** Illustrere overordnet hvordan web-siden fungere

### Webinterface beskrivelse

MainWindow (index)	Index loader shippas ind som er en form som håndtere adgangskoder. Dette er siden brugeren kommer til som det første. Brugeren skal taste sin adgangskode for at få adgang til databasen
Databasen (logon)	Denne side bliver kaldt efter at brugeren er logget på. Her vælger brugeren hvilket skib denne ønsker at få vist data for.

**Tabel 9.7.** Web sidens opbygning

### Skibs data i database

Et billede af hvordan data fremvises for brugeren. For de andre sider på websitet kan disse ses i *Detaljeret Softwaredesign, Appendix B*.

ID på skib	Styrbord vandstand	Bagbord vandstand	Hældning i grader	Skib tilsluttet
TERMINATED				
Martha	2%	2%	1	5s
Martha	5%	2%	1	15s
Martha	2%	4%	2	5s
Martha	2%	2%	3	5s
Martha	2%	2%	2	5s
Martha	2%	2%	3	10s

**Figur 9.36.** Billede af databasen for skibet Martha

### 1: Skib er sidst opdateret

Brugeren kan se hvilket skib denne er inde på og hvornår databasen sidst er opdateret.

**2: ID på skib**

Skibets ID bliver vidst. Hvis at KI lukker tcp forbindelsen vil skibts ID skifte til "TERMINATED".

**3: Styrbord vandtanksniveau**

Der vises hvor meget vand der er i styrbordets ballasttank.

**4: Bagbord vandtanksniveau**

Der vises hvor meget vand der er i bagbordets ballasttank.

**5: Hældning i grader**

Der vises hvor mange grader sibet hælder i forhold til styrbord.

**6: Skib tilsluttet**

Der vises hvor lang tid i sekunder, der er gået fra seneste overførelse før til denne.

### 9.5.8 MySQL

Som database benyttes MySQL. Denne er en multithreading database som giver mulighed for at flere kan tilgå denne samtidigt. Det er et færdigt produkt, der kan implementeres på de fleste styresystemer. Systemet giver mulighed for at oprette interne databaser med egne taballer til lagring af data (se *Detaljeret Softwaredesign, Appendix B*) for flere detaljer). MySQL giver desuden mulighed for at man ved hjælp af php direkte på et website kan oprette databaser og tabeller. Dette betyder at når et nyt skib, som ikke har været koblet til systemet før, tilkobler serveren, kan serveren oprette en fil med dennes data. Når et website med dette implementeret loader filen kan denne se på om dette skib allerede findes. Hvis det ikke gør, kan den oprette en intern database med tabeller til lagring af data og derefter gemme den nye data. Dette benyttes ikke i denne prototype, men systemet er klargjort til det med få ændringer.

## 9.6 Resultater

I dette afsnit samles der op på resultaterne gruppen har opnået gennem projektforløbet. Her vil der blive beskrevet resultater for de enkelte moduler gruppen har implementeret. Til sidst bliver der knyttet en kommentar til det samlede resultat. Der vil i afsnittet blive henvist til testresultater dokumenteret i dokumentationsdokumenterne, henholdsvis *Enhedstest*, *Integrationstest* og *Accepttest*.

### 9.6.1 Kontrolinterfacet

Kontrolinterfacet er fuldt implementeret og har opfyldt alle enhedstests for modulet. TCP- og UART-kommunikationen er udviklet i Qt med API'er specifikke for Qt. Der er dog flere steder hvor der godt kunne ske forbedringer (se afsnit 11). Slutresultatet er et tilfredsstilende resultat med tanke på at arbejdsbyrden er løftet som en enkeltmandsopgave.

### 9.6.2 Database og webinterface

Database og Webinterfacet er fuldt implementeret. I database-modulet er serveren fuldt kørende og har en stabil forbindelse med Kontrolinterfacet via TCP. Serveren er i stand til at gemme alle data til en tekstfil men at gemme direkte til MySQL databasen blev aldrig implementeret, da der opstod problemer med MySQL driver og header.

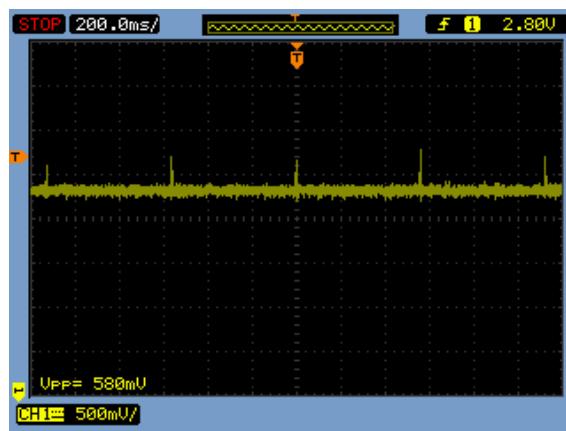
Webinterfacet er fuldt implementeret og ved opstart fungerer indtastningen af brugeradgangskoden korrekt med undtagelse ved forkert indtastet adgangskode. Her bliver brugeren ikke bedt om at taste nyt password, men får i stedet en blank side. Brugeren har dog ikke adgang til data.

### 9.6.3 Styringsmodul

Styringsmodulet er fuldt implementeret, men ikke finjusteret. Forbindelsen til Kontrolinterfacet er fuldt funktionel. Forbindelsen til VBTE virker 95% af tiden. Automatisk hældningsregulering er implementeret, men da niveaumålinger ikke modtages fra Vandballasttankenhederne er det ikke muligt at tømme tankene inden der fyldes i den anden tank. Prototypen af modulet er monteret med 7 LED's der bruges til fejlfinding. Prototypen er også lagt ud på print med indeholdende levelkonvertering og I2C pull-up modstande.

### 9.6.4 Vandballasttankenhederne

VBTE-modulet nåede aldrig at komme til at fungere optimalt. Efter at have implementeret ultralydsafstandsensoren og undersøgt/tweaket på parametrene viste det sig at det ikke fungerede ret godt over længere afstande (i hvert fald i dette design). Dette kan skyldes at der blev anvendt for lav en spænding over transduceren eller at der skulle filtreres på det modtagede signal så det kunne forstærkes mere op. Under kontrollerede forhold i laboratoriet lykkedes det at få flotte detektioner på ultralydsafstandssensoren som kan ses på *figur 9.37*.



**Figur 9.37.** Detektioner af ultralydsafstandssensoren med faste intervaller.

Derudover virkede I2C-delen, dog med fejl på data i enkelte transmissioner. Det vides ikke om det er et normalt problem ved I2C eller om det havde noget at gøre med vores system. Til sidst er der udlagt print til VBTE'en, en 2x16 LCD skærm til test samt dipswitches og knap til at skifte I2C-adresse mens systemet kører.

### 9.6.5 Strømforsyning

Strømforsyningen er fuldt implementeret. I testen er der blevet brugt effektmodstande som lod, en laboratoriestrømforsyning og et voltmeter. Enhedstesten er blevet godkendt. Vitale dele på strømforsyningen bliver dog noget varme ved fuld load på udgangen, men da det med stor sandsynlighed sjældent vil ske, ses det ikke som et problem. Strømforsyningen er implementeret med Styringsmodulet og Vandballasttankenhederne. Derudover er der udlagt et print med 12V- og 5V-forsyning.

### 9.6.6 Samlede resultat og vurdering af resultater

Til de samlede resultater har gruppen udarbejdet en tabel der viser hvordan resultatet af testforløbet fra enhedstest til accepttest. For detaljer om den enkelte test henvises til *Enhedstest*, *Integrationstest* og *Accepttest*. Testresultaterne er opdelt i fire katagorier som angivet i tabellen:

Godkendt	Delvist godkendt	Ikke godkendt	Ikke testet
✓	✗	÷	◊

**Tabel 9.8.** Testgodkendelseskategorier

Test	Test case ID						
	1	2	3	4	5	6	7
<b>Enhedstest software</b>							
Kontrolinterface	✓	✓	✓	✓	✓		
Database	✓	✓	✓	✓	✓		
Styringsmodulet	✓	✓	✓	✓	✓	✓	✓
VBTE	✓	✓	✓	✓	✓		
<b>Enhedstest hardware</b>							
Styringsmodul	✓	✓					
VBTE	✓	✓	✗				
Strømforsyning	✓	✓	✓				
<b>Integrationstest</b>							
Accepttest	✓	✗	✗	✓	✓	✓	

**Tabel 9.9.** Samlet tabel med alle resultater

Som tabellen illustrerer er der blevet implementeret en masse funktionaliteter, som er blevet godkendt gennem enhedstest. Dog var der problemer med afstandsmåling på VBTE og denne testcase kunne derfor kun delvist godkendes.

Integrationen af systemets enheder er blevet godkendt.

Grundet problemer med afstandsmåling på VBTE kan Accepttest case 2 og 3 kun delvist godkendes.

## 9.7 Opnåede erfaringer

### 9.7.1 Gruppen

Det har været rigtig interessant at skulle samarbejde fem personer fra fire forskellige grupper. Fordi gruppen er ny har vi skulle finde en fælles måde at gøre tingene på. De gamle metoder er blevet sammenholdt og forskellene har givet anledning til spørgsmål; spørgsmål der ikke tidligere er blevet stillet. Diskussionerne det har medført har været mange, lange, og til tider frustrerende, men ens for dem er at de har været utrolig lærerige. De diskussioner har medført at gruppens forståelse af udviklingsprocessen, samt hvad der er vigtigt når man strukturer et produkt, har rykket sig væsentlig mere end det har gjort sig gældende på de foregående semestre - og det uanset hvilken gruppe man tidligere har været i.

Gruppen er blevet udfordret på kommunikationen internt. Alle har skullet vænne sig til hvordan de nye gruppemedlemmer arbejder, hvad deres forcer er og ikke mindst hvad deres svagheder er.

I udviklingsforløbet er systemets moduler blevet delt op i ansvarsområder. Gruppen har erfaret at dette er en force, da det sikrer et tilhørelsesforhold, og dermed også et ansvar, for at hvert enkelt moduls opgaver er blevet udført.

### 9.7.2 Agile udviklingsmetoder

Gruppen har gennem dette projektforløb fået en bedre forståelse for iterationer og hvordan dokumentationen holdes opdateret løbende. Endvidere har gruppen erfaret vigtigheden i omstilling og fordelen ved at arbejde agilt.

Gruppen har arbejdet med faserne i projekt og har opnået en bedre forståelse for vigtigheden af faser og iterationer. Især kravspecifikation og systemarkitektur er igennem iterationer blevet forbedret, hvilket efterfølgende har gjort implementering og tests lettere at håndtere og gennemføre.

### 9.7.3 Udvikling af nye komponenter

Gennem udvikling af projektet har gruppen erfaret, at udviklingen af nye moduler kan være så tidskrævende, at det ofte kan svare sig at købe færdiglavede komponenter og inkorporere dem i systemet for at sikre en overordnet funktion. Udviklingstiden for et modul vurderes ud fra teknologiundersøgelsen, og da gruppen ikke har prøvet at lave et komponent helt fra bunden er dette en ny erfaring. Derudover kan der spares betydelig tid og købte produkter kan bidrage til et pålideligt system da disse komponenter er langt mere gennemtestedede end hvad vi overhovedet kan nå.

### 9.7.4 Test

Fra tidligere projekter har gruppen tænkt tests som noget der bare skulle laves. Dette har medført et ambivalent forhold til test. I dette projekt har gruppen erfaret at test er med til at fremhæve mulige forbedringer af den implementering, der er udført. Testresultater medtages i den iterative designprocedure, med henblik på at forbedre det testede modul. Denne erfaring har ført til mere tilfredstillende prototyper igennem projektforløbet.

### 9.7.5 Værktøjer

SVN har været anvendt igennem projektet med stor succes. Det har gjort arbejdet nemmere når der skulle skrives dokumentation og rapport. Derudover har gruppen for første gang anvendt L<sup>A</sup>T<sub>E</sub>X med stor tilfredshed.

## 9.8 Udviklingsværktøjer

I projektet er der anvendt værktøjer både i forbindelse med hardware- og softwaredesign samt dokumentation og rapport.

- Multisim 11.0
- PSoC Creator 2.1
- Qt-creator med Qt 4.8.1 og 5-beta
- Microsoft Visio
- Notepad++
- Mathcad
- Google Docs
- Google Code
- Texmaker
- TortoiseSVN 1.7.10
- Dropbox
- TextWrangler



# Konklusion 10

---

Gruppen ser i slutningen af semesteret tilbage på et godt projektforløb.

Forløbet har givet en væsentlig forbedring af forståelsen for udviklingsprocessen. Både forståelsen af faserne og deres sampspil er blevet udbygget. På den front må projektet siges at have været meget succesfuldt.

Gruppen har i accepttesten bevist, at den har formået at implementere langt størstedelen af den ønskede funktionalitet. At hele accepttesten ikke har kunnet godkendes skyldes problemer med afstandsmålingen, der ellers opererede godt under kontrollerede forhold. Derfor må implementeringen siges at være delvist succesfuld.

Gruppen havde i starten en udfordring i at være sammensat af fem personer fra fire forskellige grupper. Denne udfordring blev vendt til en fordel, da gruppen begyndte at udnytte, at den anskuede udfordringer med fire forskellige erfaringssæt. Dette har i høj grad bidraget til læringsprocessen, og samarbejdet må dermed konkluderes at have været succesfyldt.



# Forbedringer til systemet 11

---

I dette afsnit samles op på det endelige system, og der diskuteres forbedringer til systemet. Der diskutes ikke udvidelser, men reelle forbedringer til det dokumenterede system. Afsnittet er delt op i fire punkter: *Sikkerhed*, *Optimering af overførsel af data*, *Effektivitet* og *Prototype*.

- Sikkerhed

Før systemet skulle implementeres mere er der meget sikkerhed der skal tages til overvejelse. Visse scenarier vil få katastrofale konsekvenser hvis systemet kom ud på et stort skib. Forsvandt forbindelsen mellem SM og VBTE mens en VBTE er ved at fylde i en tank vil dennes tilstand ikke blive ændret og VBTE'en vil blive ved med at fylde vand ind til der ikke kunne være mere og skibet ville kæntre. Forhåbentligt ville SM og den anden VBTE, hvis der stadig var forbindelse der, kunne oprette holde vatter for skibet ved at fylde denne tank også. Men forsvandt SM fuldstændigt kunne det gå helt galt.

Her kunne indføres timere på hver VBTE til at lukke for ventilerne hvis der ikke blev opdateret tilstand i en hvis tidsperiode. En slags watchdog.

- Optimering af overførsel af data

Sådan som systemet lige nu håndterer data bliver der udskrevet til brugeren, at forbindelsen er mistet til Styringsmodulet, hvis der bliver svaret med forkert data til Kontrolinterfacet. Dette kan optimeres ved at forsøge flere gange, hvis det rigtige svar ikke kommer tilbage. Derudover er der ikke nogen validering af data mellem Styringsmodulet og Vandballasttankenhederne. Den eneste sikring indført her er at ventilerne bliver lukket hvis der bliver modtaget data uden for protokollen. Man kunne indføre at VBTE svarer tilbage med hvad den har sat sig til og herefter sender niveauet tilbage.

- Effektivitet

Systemet tømmer/fylder kun en tank af gangen som implementeringen er sket. Dette kunne optimeres på flere måder. Den ene er mere intelligent software på Styringsmodulet. Den anden, og mere interessante, kunne være helt at undlade at implementere Styringsmodulet og have en hældningsmåler på hver Vandballasttankenhed. Vandballasttankenheden vil derfor reagere selvstændigt på hældningen resulterende i at der kan blive både tømt og fyldt på samme tid i hver sin side. Problemet med dette bliver så kontakt til Kontrolinterfacet og at der skal laves væsentlige ændringer i hele struktureringen af systemet.

Til effektivitet kan også nævnes at manuel hældningsregulering også manuelt skal sættes tilbage igen. Dette kunne håndteres ved hjælp af Styringsmodulet. Hvis det var nødvendigt med manuel hældning ville man også kunne antage at systemet ville ramme tilbage i vatter i takt med den store lastning. Dermed burde Styringsmodulet automatisk aktivere automatisk hældningsregulering når hældningen igen er vatter. Desuden kunne et check på om Databasen har modtaget data fra Kontrolinterfacet sikre at terminalrepræsentanten har mulighed for at opdage en mulig fejl.

- Prototype

Der er i projektet anvendt ventiler til at styre in/out-flow af vand til tankene. Dette er anvendt, da det var dem vi kunne få af Danfoss. Dette vil ikke fungere på et skib og det gør også reguleringen af vores prototyper langsommere. Det vil derfor være meget bedre at anvende pumper til systemet.

# Referencer 12

---

## 12.1 Artefakter

### 12.1.1 Kravspecifikation

Kravspecifikationsdokumentet er udarbejdet i begyndelsen af projektet og omfatter beskrivelse af use cases, ikke-funktionelle krav samt kvalitetsfaktorer. Den fuldstændige kravspecifikation kan ses i bilaget *Kravspecifikation.pdf*

### 12.1.2 Accepttestspecifikation

Accepttestspecifikationsdokumentet beskriver de tests, der skal laves for at undersøge om de ønskede krav er opfyldt. Den fuldstændige accepttestspecifikation kan ses i bilaget *Accepttest.pdf*.

### 12.1.3 Systemarkitektur

Systemarkitekturdokumentet beskriver systemets HW/SW opbygning og grænseflader. Den fuldstændige systemarkitektur kan ses i bilaget *Systemarkitektur.pdf*.

### 12.1.4 Integrationstestspezifikation

Integrationstestspezifikationen beskriver de tests, der skal laves for at undersøge hvorledes de forskellige komponenter kan kommunikere internt. Den fuldstændige Integrationstest kan ses i bilag *Integrationstest.pdf*.

### 12.1.5 Detaljeret design

Det detaljerede designdokument beskriver hvordan HW/SW er designet og hvordan systemets komponenter fungerer. Det fuldstændige Detaljeret designdokument kan ses i bilag *Detaljeret Hardwaredesign.pdf* og *Detaljeret Softwaredesign.pdf*.

### 12.1.6 Enhedstestspezifikation

Enhedstestspezifikationen beskriver de tests der skal laves for at undersøge om de forskellige stubbe af systemet fungerer efter hensigten. Den fuldstændige enhedstestspezifikation kan ses i bilag *Enhedstest.pdf*.

## 12.2 Hjemmesider

<http://kurser.ih.a.dk/eit/eit-lab/lagerliste.htm>  
<https://www.retsinformation.dk/Forms/R0710.aspx?id=26449>

## 12.3 Liste over bilag på CD

Komponentliste.pdf  
Scrum P4G3.xlsx

### 12.3.1 Logbøger

LogdJohnny.pdf  
Lundslogbog.pdf  
Micklogbog.pdf  
NicolaiGludlogbog.pdf  
RoesenLogbog.pdf

### 12.3.2 Kode

#### Kontrolinterfacet

BROS.pro.pdf  
dataserver.cpp.pdf  
dataserver.h.pdf  
kontrolinterface.cpp.pdf  
kontrolinterface.h.pdf  
main.cpp.pdf  
mainwindow.cpp.pdf  
mainwindow.h.pdf  
manudialog.cpp.pdf  
manudialog.h.pdf  
protokolenum.h.pdf  
rs232.cpp.pdf  
rs232.h.pdf  
sensor.cpp.pdf  
sensor.h.pdf  
status.h.pdf  
styringsmodul.h.pdf  
vbte.h.pdf Kontrolinterface.rar (Qt Creator 2.5.83 Projekt i et rar arkiv)

**Styringsmodulet**

Haeldningsregulering.c.pdf  
Haeldningsregulering.h.pdf  
Haeldningssensorblok.c.pdf  
Haeldningssensorblok.h.pdf  
Init.c.pdf  
Init.h.pdf  
Kommunikation.c.pdf  
Kommunikation.h.pdf  
komnavn.h.pdf  
main.c.pdf  
protokolenum.h.pdf  
smflags.h.pdf  
VBTE-Driver.c.pdf  
VBTE-Driver.h.pdf  
sm.rar (PSoC Creator 2.1 Projekt i et rar arkiv)

**Vandballasttankenhed**

ADC\_DelSig\_1\_INT.c  
Change\_I2C\_ADD.c  
dist.c  
dist.h  
I2Chandle.c  
I2Chandle.h  
init.c  
init.h  
isr\_dist.c  
main.c  
valve.c  
valve.h  
VBTE.rar (PSoC Creator 2.1 Projekt i et rar arkiv)

**Server**

enumdekrypteringskode.cpp  
enumdekrypteringskode.h  
main.cpp  
protokulenumber.h  
savedata.h  
savedata.cpp  
server.h  
Server.pro  
server.ui  
tcp.cpp

tcp.h  
update.h  
Server.zip

### **Webinterface**

404.php  
bros.php  
godkend.php  
index.php  
logon.php  
mysql.php  
pass\_form.php  
ship.php  
shippas.php  
style.css  
bg\_body.jpg  
header.jpg  
mnu\_bottomshadow.gif  
mnu\_topshadow.gif  
Webinterface.zip

### **12.3.3 Dokumentation**

Accepttest.pdf  
Arkitektur.pdf  
Detaljeret.hardware\_design.pdf  
Detaljeret.software\_design.pdf  
Enhedstest.pdf  
Integrationstest.pdf  
Kravspecifikation.pdf

### 12.3.4 Datablade

Datablad:	Type:
CY8C55	PSoC5
KXSC7-2050	Accelerometer
ST3232cn	Level konverter
BD139	Transistor
BC547	Transistor
400SRT 160	Ultralydstransmitter og receiver
Ventil	Ventil til ventilspolen
Ventilspole	Ventilspole der driver ventilen
LM317	Spændingsregulator
tbl04	Diodebro

### 12.3.5 Billeder

BROS - Systembillede.jpg

### 12.3.6 Jura

Gruppen har inden start lavet en grundig undersøgelse af de juridiske problemstillinger som opstår i forbindelse med et projekt som BROS. De vigtigste dele er blevet vedlagt i bilag Jura.