

INGENIØRHØJSKOLEN ÅRHUS

ELEKTRO-INGENIØR LINIEN

SEMESTERPROJEKT E4PRJ4

---

# Bias Reducing Operating System

---

*Skrevet af:*

Nicolai GLUD

*Studienummer:* 11102

Johnny KRISTENSEN

*Studienummer:* 10734

Rasmus LUND-JENSEN

*Studienummer:* 11111

Mick HOLMARK

*Studienummer:* 11065

Jacob ROESEN

*Studienummer:* 10095

*Vejleder:*

Carl Jakobsen



AARHUS  
UNIVERSITET  
INGENIØRHØJSKOLEN

17. december 2012



# Resume 1

---

Utom



# **Abstract** 2

---

Not-empty

# 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>Opgaveformulering</b>	<b>13</b>
<b>Kapitel 6</b>	<b>Projektformulering</b>	<b>15</b>
<b>Kapitel 7</b>	<b>Systembeskrivelse</b>	<b>17</b>
<b>Kapitel 8</b>	<b>Kravspecifikation</b>	<b>19</b>
8.0.1	Overordnede krav til systemet . . . . .	19
8.0.2	Funktioner . . . . .	19
8.0.3	Krav til udviklingsprocess og teknologi . . . . .	19
8.0.4	Krav til grænseflader . . . . .	19
8.0.5	Krav til kvalitet . . . . .	20
<b>Kapitel 9</b>	<b>Afgrænsning</b>	<b>21</b>
<b>Kapitel 10</b>	<b>Projektbeskrivelse</b>	<b>23</b>
10.1	Projektgennemførelse . . . . .	23
10.1.1	Rollefordelinger . . . . .	23
10.2	Metoder . . . . .	23
10.2.1	SCRUM . . . . .	24
10.2.2	V-model . . . . .	25
10.3	Analyse . . . . .	25
10.3.1	Hvordan måler vi hældning? . . . . .	25
10.3.2	Hvordan skal de forskellige modulerne forsynes? . . . . .	25
10.3.3	På hvilken enhed ønsker vi at afvikle programmet KI . . . . .	25
10.3.4	Hvordan anvendes UART og TCP i Qt-frameworket . . . . .	26
10.3.5	Jura . . . . .	26
10.3.6	Valg af database . . . . .	26
10.4	Systemarkitektur . . . . .	26
10.4.1	Udkast fra Lund . . . . .	26
10.4.2	Tidl. text . . . . .	29
10.4.3	Systemkomponenter . . . . .	29
10.5	Design og Implementering . . . . .	30

10.5.1 Kontrolinterface . . . . .	30
10.5.2 VBTE . . . . .	33
10.5.3 SM . . . . .	38
10.5.4 Database . . . . .	41
10.5.5 Serveren . . . . .	41
10.5.6 Webinterface . . . . .	43
10.6 Resultater . . . . .	44
10.6.1 KI . . . . .	45
10.6.2 Database og webinterface . . . . .	45
10.6.3 SM . . . . .	45
10.6.4 VBTE . . . . .	45
10.6.5 powersupply . . . . .	45
10.6.6 Samlede resultat og vurdering af resultater . . . . .	45
10.7 Opnåede erfaringer . . . . .	45
10.7.1 Generelt . . . . .	45
10.7.2 Rettidig test . . . . .	49
10.7.3 Udvikling af hældningssensor . . . . .	49
10.7.4 Metoder . . . . .	49
10.7.5 Udvikling af afstandssensor . . . . .	49
10.7.6 Udvikling af Databasen . . . . .	50
10.7.7 Bulletpoints til skrivning af OpXP . . . . .	50
<b>Kapitel 11 Konklusion</b>	<b>51</b>
<b>Kapitel 12 Forbedringer til systemet</b>	<b>53</b>
<b>Kapitel 13 Referencer</b>	<b>55</b>
13.1 Artefakter . . . . .	55
13.1.1 Kravspecifikation . . . . .	55
13.1.2 Accepttestspezifikation . . . . .	55
13.1.3 Systemarkitektur . . . . .	55
13.1.4 Integrationstestspezifikation . . . . .	55
13.1.5 Detaljeret design . . . . .	55
13.1.6 Enhedstestspezifikation . . . . .	55
13.2 Hjemmesider . . . . .	56
13.3 Liste over bilag på CD . . . . .	56
13.3.1 Kode . . . . .	56
13.3.2 Dokumentation . . . . .	56
13.3.3 Datablade . . . . .	57
13.3.4 Billeder . . . . .	57



# Forord 3

---

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

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

---

Denne rapport omhandler 4. semesters projekt. Projektets emne er "slagsideregulering af bulkskib", som er et selvvalgt emne. Rapporten beskriver processen af projektet herunder hvordan forløbet har været, 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 været stillet enkelte krav til projektet og det system der skulle udvikles. Nogle af disse krav afspejler derfor også hvordan arbejdsprocessen har været og ikke mindst nogle af de komponenter som igår i det færdige system.

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, for at fuldføre gennemførelsen af et komplet projektforløb.

Projektet har været inddelt i et antal 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 2 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 9 og 10<sup>1</sup>. Alle afsnit er skrevet så de som udgangspunkt godt kanstå alene, hvorfor der igennem rapporten vil komme gentagelser hvis man læser denne fortløbende. Rapportens egentlige indhold begynder fra afsnit 6 – opgaveformulering. Her gennemgås opgaveformuleringen givet fra vejledere til gruppen, som indeholder minimumskrav til projektet. I projektformuleringen bliver 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 8 – krav, fremlægges kravene der fra gruppen er stillet til projektet. Herefter beskrives projektafgrænsningen samt arbejdsmetoder og fremgangsmåde i afsnit 8 og 10.1. Afsnit 10.2 beskriver det analyse arbejde projektet har

---

<sup>1</sup>Fixme Note: lav dynamiske referencer

gennemgået. I afsnittene fra 10.2-10.5 nedbrydes hele projektet fra øverste abstraktion og ned til implementering. 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 10.6. Efter resultaterne er præsenteret, fremlægges de erfaringer gruppen har opnået igennem hele projektforløbet, samt hvilke ting der har fungeret godt. I afsnit 10.8 snakkes der ganske kort om de anvendte udviklingsværktøjer. Afslutningsvist konkluderes der på hele projektet på godt og ondt i afsnit 11. Det anbefales dog at læse rapporten fortløbende for at få den bedste samlede forståelse for projektet og produktet.<sup>2</sup>

---

<sup>2</sup>**FiXme Note:** Lave refrencerne til de enkelte afsnit.

# Opgaveformulering 5

---

Gruppen skal fremstille et system der overholder nogle krav. Kravene er at systemet skal kunne interagere med omverdenen vha. 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.



# Projektformulering 6

---

Gruppen har valgt at arbejde med lastning og losning af et bulkskib. Vi vil gerne lave et system der aflaster skibets ansvarshavende officer, som står på broen under hele lastningen/losningen. Hans job er at holde øje med at hele operationen bliver gjort ordentligt. Denne opgave vil vi gerne gøre nemmere. Med et system der automatisk sørger for at skibet altid er i vatter skal kaptajnen kunne interagere med systemet hvis systemet kommer med en alarm. Kaptajnen kan manuelt vælge at flytte ballast til den ene side, hvis han ved at der komme en række tunge containere, der skal stå i modsatte side. For at sikre at alle skibe i havnen er i vatter, sender systemet statusbeskeder til en database på havnekontoret. Kontoret kan derfor sende bemanding eller tage kontakt til skibet, der har en alarm.



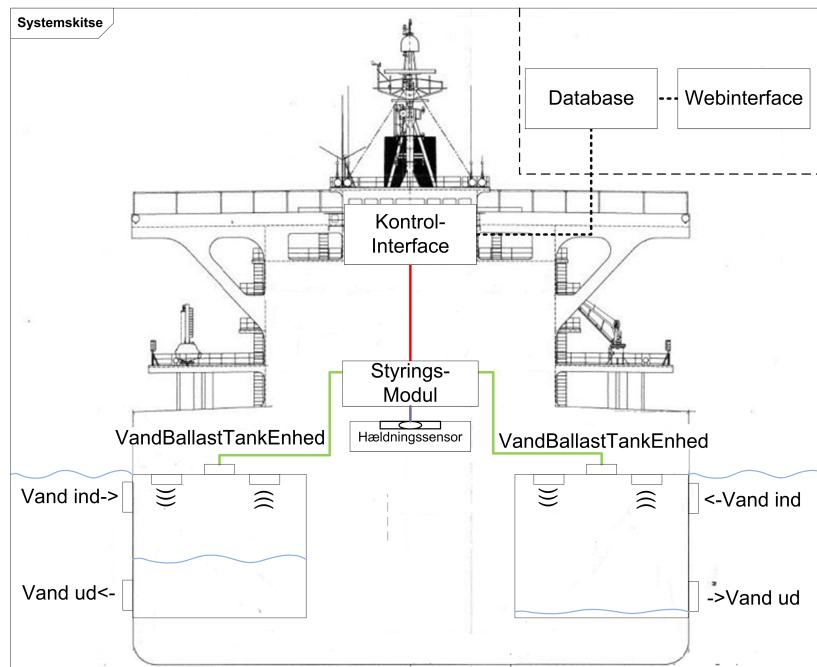
# Systembeskrivelse

7

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 slagside af skibet.

Hele systemet styres fra Skibsførens kontor hvor Kontrolinterfacet - en grafisk brugergrænseflade - er installeret. Her kan der aflæses skibets hældning, vandindholdet af tankene og statusmeldinger for systemet. Det er også her systemet aktiveres og deaktiveres. Som udgangspunkt vil systemet automatisk opretholde en hældning på nul grader, men hvis man ønsker det kan man her manuelt give skibet en mindre slagside. Dette kan gøres for at imødekomme en større slagside til modsatte side 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 en repræsentant fra terminalen følge skibets status.



Figur 7.1. Systemskitse af BROS



# Kravspecifikation

8

---

Kravspecifikationen er udarbejdet i begyndelsen af projektforløbet og omfatter use cases, ikke funktionelle krav samt kvalitets faktorer. For den fulde kravspecifikation, henvises der til dokumentations dokumentet.

## 8.0.1 Overordnede krav til systemet

Systemet skal indeholde én til flere sensorer samt en til flere aktuatorer. Endvidere skal systemet indeholde et kontrolinterfacet og et styringsmodul. Kontrolinterfacet fungerer som brugergrænseflade og har kontakt til en ekstern database.

## 8.0.2 Funktioner

Systemets funktioner er beskrevet vha. Use Cases. I hver Use Case er der en beskrivelse af en enkelt funktionalitet systemet skal have. Der er også beskrevet alternative hændelser, såfremt hældsesforløbet ikke forløber som planlagt. Se kravspecifikationsdokumentet for alle systemets Use Cases. På *Figur 8.1* ses Use Cases for systemet.

Når systemet sættes op første gange anvendes Opstart af Program (Use Case 1). Når systemet er startet op kan brugeren vælge mellem Automatisk ballastkontrol (Use Case 2) eller Manuel Skibsvinkling (Use Case 3). Hvis brugeren vil deaktivere systemet kan han afslutte programmet (Use Case 4). En Terminaloperator kan tilgå databaseinformationen via et Webinterface (Use Case 5 og 6).

## 8.0.3 Krav til udvinklingsprocess og teknologi

Projektforløbet skal følge V-modellen. SCRUM skal bruges i projektstyringsprocessen til at give overblik over projektets arbejdsopgaver. SCRUMmaster og projektleder uddeles til gruppemedlemmer.

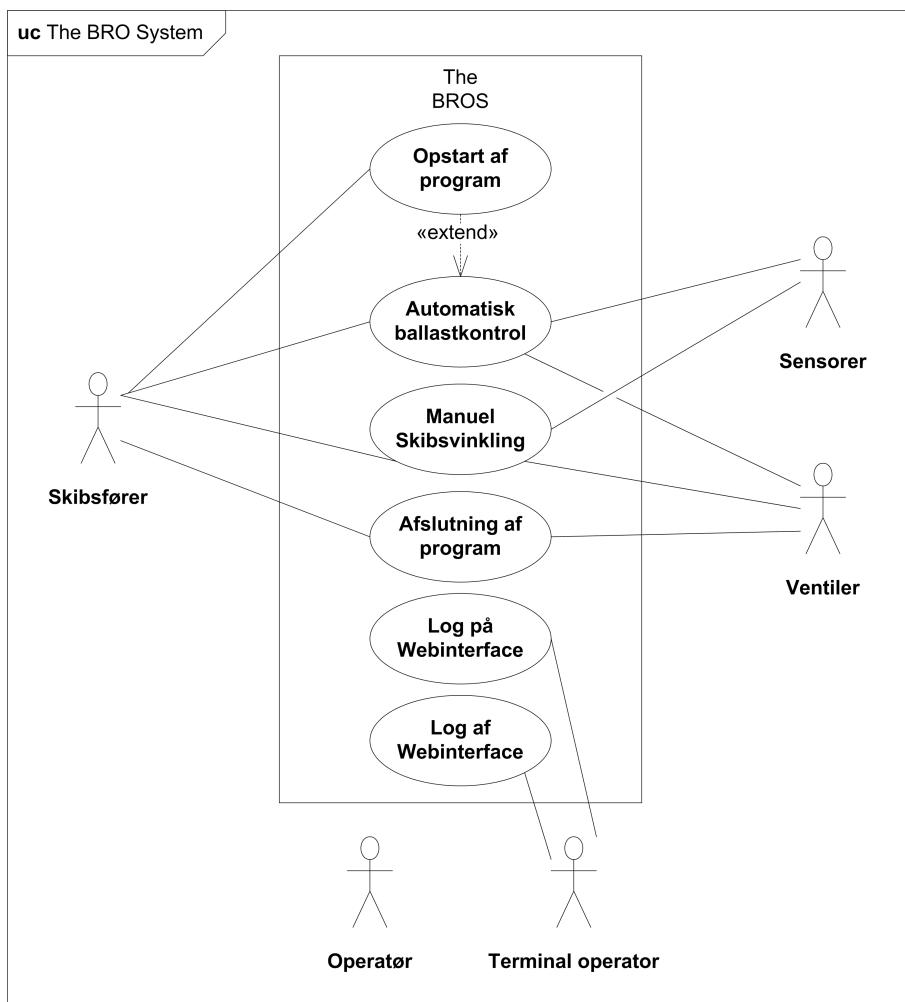
Til dokumentation skal der bruges SysML diagrammer til at beskrive systemet overordnet og i detaljer.

Programmeringen til systemets embedded enheder, Styringsmodulet og Vandballasttankenhederne, anvendes C. Til kontrolinterfaces og databasen anvendes C++. Til Databaselagring anvendes mySQL. Webinterface er skrevet i php og HTML 5.

## 8.0.4 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 via I2C protokollen.



**Figur 8.1.** Use Case Diagram for BROS

På kontrolinterfacet skal alt funktionaliteten være indbygget i brugergrænsefladen. Det skal være muligt at skifte mellem automatisk ballastkontrol og manuel skibsvinkling.<sup>1</sup>

### 8.0.5 Krav til kvalitet

For at sikre kvaliteten af vores produkt har vi valgt at opstille en række kvalitetsfaktorer. Det er meget vigtigt for vores system at det er pålideligt, sikkert og Effektivt. Pålideligheden skyldes at systemet kan risikere at være fatalt for et skib, hvis der sker en fejl. Systemet skal være sikkert da det er en kritisk komponent. Endvidere skal systemet være effektivt da det ikke må sløve lastning og losningsprocessen.

Krav til Brugervenlighed 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.

<sup>1</sup>FiXme Note: Lund skal lige beskrive resten af GUI'en

# Afgrænsning 9

---

*En oversigt over de afgrænsninger dette projekt er udarbejdet med.*

Afgrænsninger sat på forhånd:

- Systemet skal interagere med omverdenen vha. sensorer og aktuatorer.
- Systemet skal omfatte pålidelig datatransmission.
- Systemet skal have 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 tabel 9.1. Grundsystemet er det system gruppen har fastlagt sig på at implementere i projektforløbet. Funktionaliteterne her er valgt for at opfylde basiskrav fra opgaveformuleringen (se<sup>1</sup>). Udvidelser er nedprioriterede funktionaliteter da de enten har lille relevans eller ikke er kritiske for at systemets grundformål kan opfyldes (egen projektformulering, se<sup>2</sup>). 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.

<b>Grundsystem:</b>	Elektronisk måling af hældning Automatisk regulering af niveau i ballasttanke Niveaumåling i ballasttanke Mulighed for brugerinteraktion Advarselsnugger
<b>Udvidelser:</b>	Måling af afstand til terminal-kaj Måling af dybdegang Manuel styring af ballast niveau Pålidelig kommunikation med ekstern enhed

**Tabel 9.1.** Grundsystem og Udvidelser til BROS

---

<sup>1</sup>Fixme Note: udfyld reference til opgaveformulering

<sup>2</sup>Fixme Note: reference til projektformulering



# Projektbeskrivelse

10

## 10.1 Projektgennemførelse

Projektet er udført af en gruppe på fem personer. Gruppen er ny og sammensat af personer fra fire forskellige projektgrupper. Dette har medbragt både gode som dårlige ting. Det er godt at opdage nye måder at anskue et projektforløb og nye vinkler på udfordringer i processen. Det har dog desværre kostet en del energi at skulle tilpasse sig hinanden. En proces hvor alle har måtte gå på kompromis på et eller flere punkter.

Længden af faserne har vist sig at være skæve da processen ikke har været lige så gnidningsfri som først håbet. Som følge heraf er tidsplanen blevet revirideret flere gange og opgaver fra ét SCRUM har måttet videreføres i det næste.

Projektets overordnede tidsplan for faserne og de eksterne milestones ligger som bilag.

### 10.1.1 Rollefordelinger

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

*Tabel 10.1.* Tabel over rollefordelinger

Vi har valgt at lave roller ud fra vores udviklingsmetode, SCRUM, der er beskrevet senere. Projektlederen har haft som ansvar at strukture arbejds- og scrummøder. Projektkoordinators ansvar har ligget i at planlægge møder og bestille lokaler. Scrummasteren er anvarlig for udviklingsplatformen, SCRUM, og sørge for at metoden anvendes mest optimalt.

## 10.2 Metoder

En kort præsentation af de to mest dominerende arbejds metoder 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.

### 10.2.1 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 2 uger. Når et sprint starter bliver opgaver overført fra backloggen til sprintet. Når nye opgaver bliver sat på sprintet bliver opgaven vurderet for hvor stort et omfang den har. Derefter diskuteres der hvilke opgaver de forskellige dele af gruppen skal lave. En gang om ugen laver man et SCRUM-Meeting. Her bliver fulgt op på opgaver lavet i løbet af ugen samt tilføjelse af nye opgaver. Alle de opgaver der ikke er færdige, når sprintet er slut, overføres til næste sprint.

I projektet er der i alt 7 sprint. På *Figur 10.1* vises det 2. sprint.

Sprint nr:	Scrummaster:	JK	Koordinator:	NG		
2	Projektleader:	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 ALLE	JK JR 20% 80%	20% 20% 100%	NG, JK
Integrationstest		3	NG		80%	NG, JK
	Total:	32		Procentvis færdig:	81.25 %	

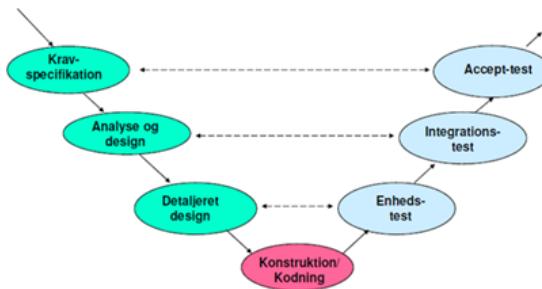
*Figur 10.1.* Sprint 2

Sprintet er afsluttet og man kan se hvordan nogle opgaver er færdige og hvordan resten skal overføres til næste sprint. Billedet illustrerer også hvordan planlægningen er opbygget. Forklaring af statusfarver er vist på *Figur 10.2*

0%	Ikke startet
20%	Påbegyndt
40%	Godt i gang
60%	Umiddelbart færdigt. Mangler 2. oppinion
80%	Mangler gennemlæsning af alle
100%	Gennemlæst af alle

*Figur 10.2.* Forklaring af sprint points

### 10.2.2 V-model



**Figur 10.3.** V modellen

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

## 10.3 Analyse

### 10.3.1 Hvordan måler vi hældning?

Før vi kunne begynde på at lave en hældningssensor blev vi nødt til at finde ud af hvilke muligheder der var hældningsumåling. En af mulighederne var at anvende et pendul eller en libelle. Disse løsninger blev undersøgt og en redegørelse findes i afsnit 10.7.

Valget faldt på at anvende et accelerometer. Det var en fordel af at anvende det accelerometer, der er monteret på PSoC'en.

### Hvordan måler vi niveauet i vandballasttankene?

### 10.3.2 Hvordan skal de forskellige modulerne forsynes?

Da modulerne sidder rundt om i skibbet, skal der overvejes hvordan de kan forsynes. Dette kan ske på flere forskellige måder:

- Batteri
  - Smart da man ungarer at trække en ledning med forsyning.
  - Dette kan være meget problematisk med et batteri, da der ikke vides hvor meget strøm der skal trækkes dvs. stort det skal være.
  - Da der i forvejen er en kablet kommunikation, skal der alligevel kabel ud til modulet, derfor kan lige så smart at have en kablet forsyning.

- Kablet forsyning

Ved at vælge at bruge en kablet forsyning dukker der andre spørgsmål op.

- F.eks. hvilken forsyningsspænding skal der være..?
- 24V, 230V, 12V, 5V eller noget helt andet..

Da der ikke er 100% kendskab til hvilke forsyning spænding der er på et skib, men nok er 230V AC skal der med stor bruges en strømforsyning til modulerne, da de bruger en lavere spænding.

Ved at have en transformator kan spænding evt. komme ned på 24V AC. De 24V AC kan føres ud til modulerne samme med kommunikationen.

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

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

Da virkningsgraden ikke har stor bestydning for produktet, 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.

### **10.3.3 På hvilken enhed ønsker vi at afvikle programmet KI**

RPi eller ej? Qt4 vs Qt5.<sup>1</sup>

### **10.3.4 Hvordan anvendes UART og TCP i Qt-frameworket**

Qt-spesifik eller C++? Boost? <sup>2</sup>

### **10.3.5 Jura**

Hvilke lovgivninger findes der på området? Er der et behov for dette? Hvad er de rigtige termer? <sup>3</sup>

### **10.3.6 Valg af database**

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

Microsoft Acces er et database system der medfølger i Microsoft Office pakken. Microsoft Access benytter sit eget format baseret på Access Jet Database Engine. En ulempe ved Microsoft Acces er at den kun fungerer under Windows og da systemet skulle altsidigt være dette ikke den bedste løsning. At lagre direkte i en tekst fil blev overvejet da det er meget simpelt at lagre til tekstmapper 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 færdig udviklet system til at fungere med webinterfaces og for programmerings sprog som C++ er der udviklet header. Systemet er et system der fungerer på mange platforme og løbende bliver udviklet på. På baggrund af dette blev dette system valgt.

## **10.4 Systemarkitektur**

### **10.4.1 Udkast fra Lund**

I dette afsnit vil systemarkitekturen for projektet blive beskrevet. Systemarkitekturen tager udgangspunkt i dokumentationsdokumentet "Systemarkitektur" og for nærmere

---

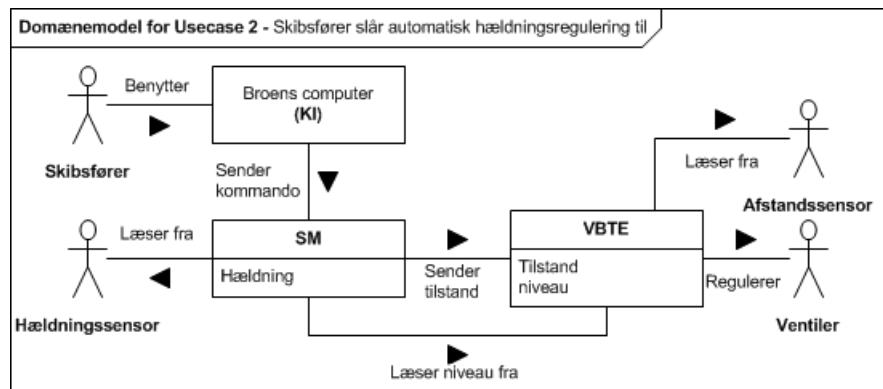
<sup>1</sup>Fixme Note: Skriv afsnit om RPi og Qt

<sup>2</sup>Fixme Note: Skriv afsnit om Qt-frameworket

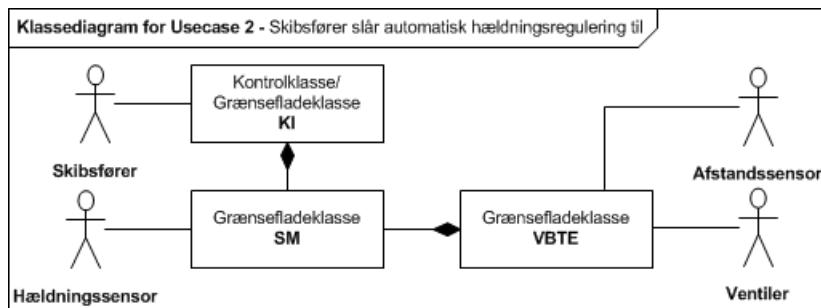
<sup>3</sup>Fixme Note: skriv afsnit om jura

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 brugt som eksempel vil være Use Case 2: Skibsfører slår automatisk hældningsregulering til.



*Figur 10.4.* Domænemodel for Use Case 2

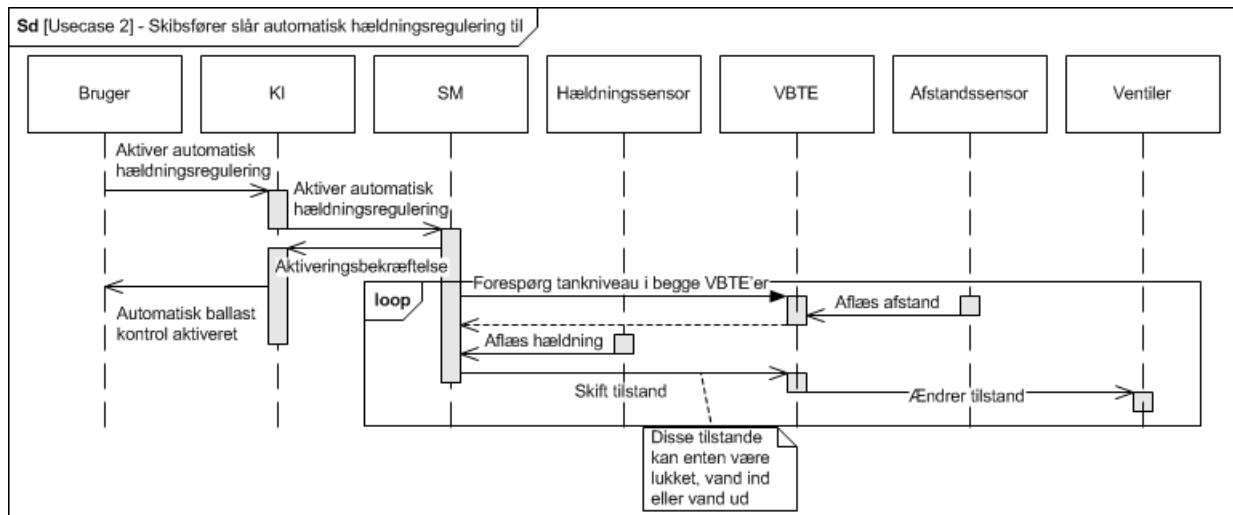
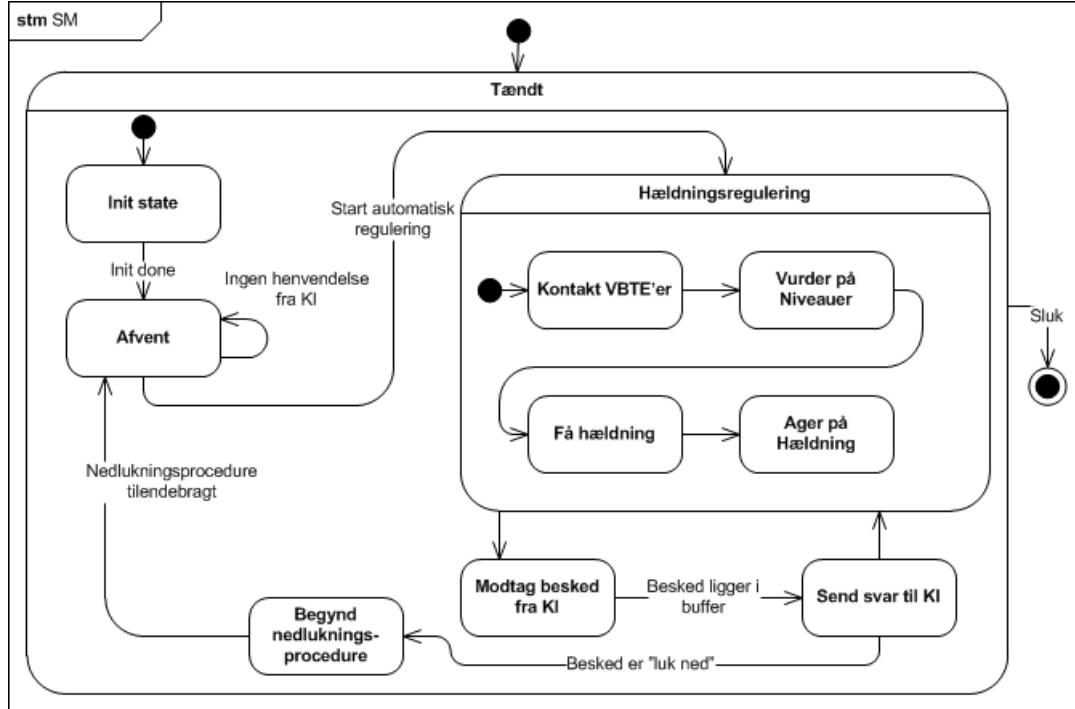


*Figur 10.5.* Klassediagram for Use Case 2

I domænemodelen (figur 10.4) beskrives relationen mellem systemets elementer og aktørerne. Disse enheder omdannes i klassediagrammet (figur 10.5) til konceptuelle klasser af en given type. Det ses at i Use Case 2 agerer KI som både en kontrolklasse og en grænsefladeklasse. Den agerer som en kontrolklasse fordi den er initiativtager og beslutningstager til den efterfølgende udvikling i systemet. Både KI, SM og VBTE agerer som en grænsefladeklasse fordi alle tre klasser er i berøring med en aktør.

I sekvensdiagrammet (figur 10.6) beskrives kommunikationen og timingen imellem klasserne. Det ses at brugeren igangsætter processen, KI videresender kommandoen, SM bekræftiger modtagelsen af kommandoen hvorefter den begynder at regulere på vandniveauet i tankene ved hjælp af VBTE'erne. Reguleringen sker på grundlag af de målinger den modtager fra hældningssensorerne. Vandniveauet vil blive ved med at blive reguleret med et formål at opnå og derpå opretholde en hældning på nul grader.

Når alle Use Cases er blevet bearbejdet som Use Case 2 er blevet i figur 10.4, 10.5 og 10.6 har gruppen dannet state machines for de konceptuelle klasser i systemet. Her visualiseres

*Figur 10.6.* Sekvensdiagram for Use Case 2*Figur 10.7.* State machine for Styringsmodulet

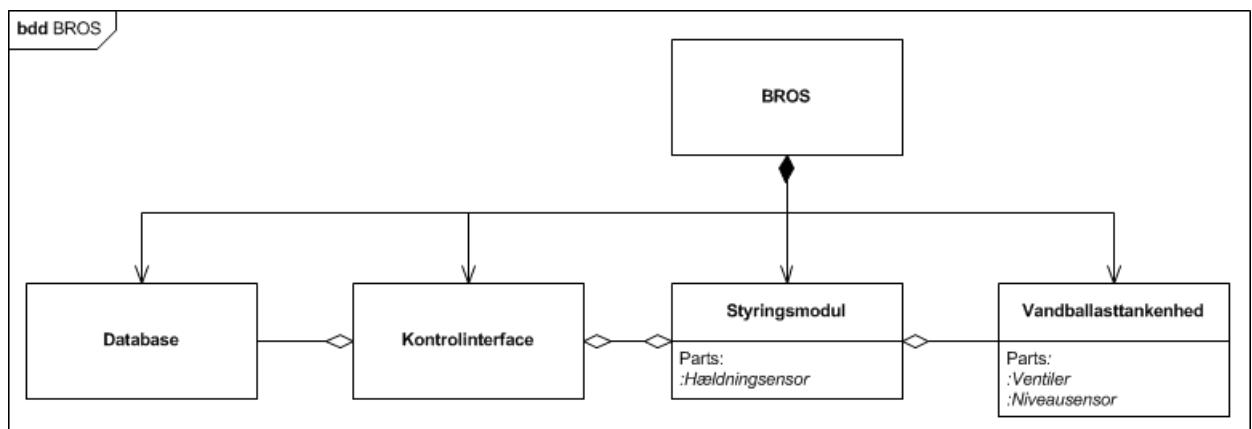
hvilke stadier klasserne har brug for at gennemgå i løbet af klassens levetid for at opfylde kravene sat i kravspecifikationen. Som et eksempel på en sådan state machine vises state machinen for Styringsmodulet på figur 10.7.

Flowet i state machinen bygger oven på det beskrevne i sekvensdiagrammerne. Starten på Use Case 2 er et brugerinput på KI'en. Dette medfører en besked fra KI til SM. Programmet 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 KI få styringsmodulet over i stadiet "Modtaget besked fra KI". SM besvarer beskeden med en aktiveringsbekræftigelse og programmet returnerer derpå til "Hældningsregulering-stadiet - nu med reguleringstypen *automatisk*.

Dette var for Use Case 2. Der i state machinen beskrevet styringsmodulets opførelse i alle Use Case tilfælde.

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 10.8. Her ses det at systemet ender ud med fire hovedblokke: Kontrolinterfacet, Styringsmodul, Vandballasttankenhed og Database. De konceptuelle klasser der ikke er endt som hovedblokke er i stedet for blevet til underblokke af enten styringsmodulblokken eller vandballasttankenhedblokken.

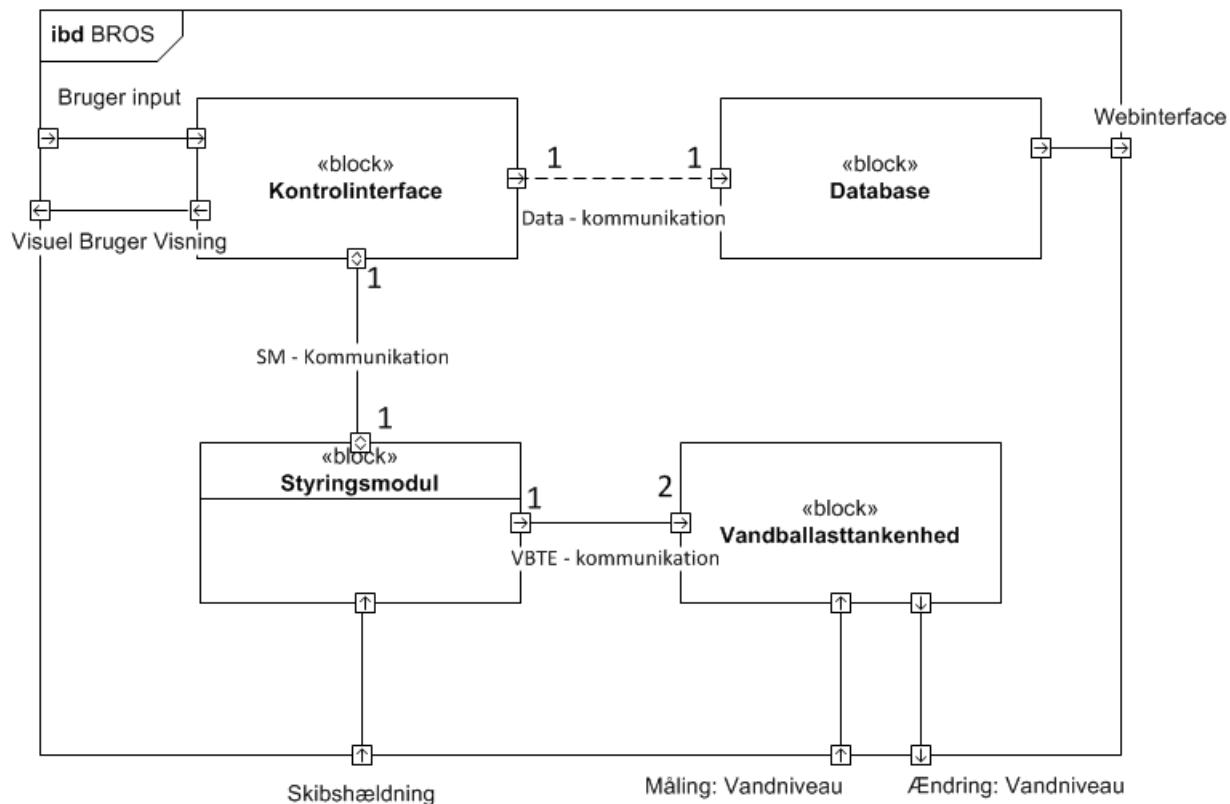


**Figur 10.8.** Blokdefinitionsdiagram for systemet

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 10.9. Her angives det også hvor mange gange de enkelte blokke skal optræde i systemet.

<sup>4</sup> Systemet er nu opdelt i fysiske blokke. For hver blok er der angivet et state machine. Kommunikationsprotokollen imellem blokkene er beskrevet. Designfasen kan påbegynde.

<sup>4</sup>Fixme Note: Indsæt protokoller



**Figur 10.9.** Internt blokdiagram for systemet

#### 10.4.2 Tidl. text

Dette afsnit beskriver systemarkitekturen for projektet "BROS" som formuleret i projektbeskrivelsen og specificeret i kravspecifikationen. Afsnittet indeholder beskrivelse af systemkomponenter, systemarkitektur, SW-komponenter, HW-komponenter og interfaces, i den givende rækkefølge.

#### 10.4.3 Systemkomponenter

Ud fra kravspecifikationen er der udvalgt disse beslutninger om komponenter til systemet og deres placering. Der refereres derfor til kravspecifikationens, ikke-funktionelle krav og krav generelt.

Brugeren integrerer med systemet igennem KI. KI er styringsmodulet for hele systemet. På KI har brugeren mulighed for at til- og frakoble systemet, justere den ønskede hældning på skibet. KI giver mulighed for at brugeren kan aflæse handlinger foretaget i systemet samtidig med at denne modtager advarsler i tilfælde at hældningen bliver for stor eller vandbalast tanke bliver overfyldte.

KI kommunikere til SM modulet igennem en uart. For at denne kommunikation kan foregå er der lavet en protocol for denne kommunikation. Kommunikationsformen er ved I<sub>2</sub>C.

SM står for at måle skibets hældning og sende denne til KI. KI kan så informere dette til brugeren. Når SM har målt hældningen på skibet giver denne besked til VBTE1 og VBTE2 om at åbne og lukke for ventilerne til tankene. VBTE1 og VBTE2 styres fra en

PSoC. VBTE1 og VBTE2 er placeret på være deres tank. For at kunne kontrollere hvor meget vand der er i tankende dette gøres ved hjælp af to ultra lydssensore som hele tiden mäter og vidergiver denne information til SM som så sender dette videre til KI der kan advare om vandstanden i tankene i tilfælde af at systemet af sat på manuel styring. KI sender data om skibet til databasen som lagre disse data i en mySQL database som så kan tilgås via et web interface.

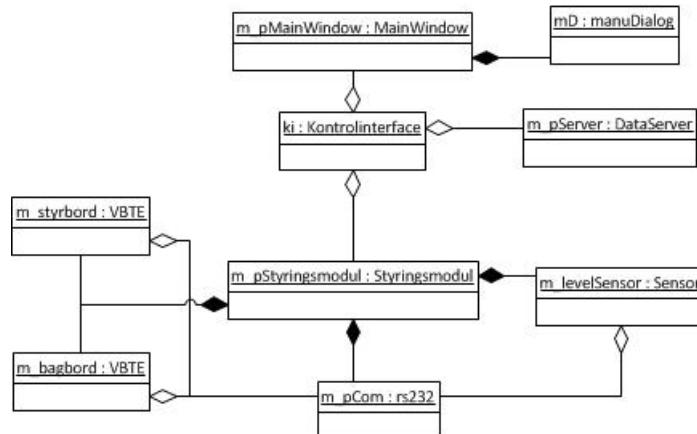
## 10.5 Design og Implementering

### 10.5.1 Kontrolinterfacet

I dette afsnit beskrives design og implementering af Kontrolinterfacet som lavet på baggrund af kravspecifikationen og systemarkitekturen.

Designet af Kontrolinterfacet afspejler meget den generelle opbygning af systemet. Således er hvert element af systemet implementeret som en klasse. Derudover er der nogle hjælpeklasser. En oversigt over klasserne og deres ansvar kan ses i tabel 10.2

Det gælder for VBTE-, SM- 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.



**Figur 10.10.** Oversigt over, og sammenhæng mellem, objekter i kontrolinterface-programmet

## Kontrolinterfacets klasser

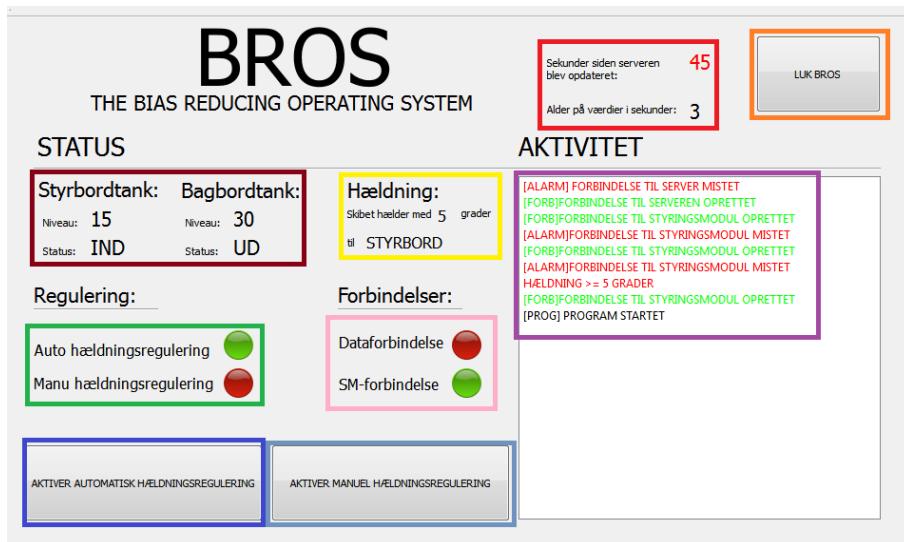
Kontrolinterface:	Programmets hovedklasse. Eksisterer for at rydde op i main-funktionen.
DataServer:	Står for alt TCP-kommunikationen med databasen. Oprettes af KI-klassen
Styringsmodul:	Oprettes af KI og opretter VBTE-, Sensor- og RS232klasserne.
Sensor:	Oprettes af SM og er ansvarlig for hældningsværdien.
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 10.2.** Kontrolinterfacets klasser

5

## 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 figur 10.5.1 er hovedvinduet vist. Her er vinduets elementer indrammet. En beskrivelse af hvert element vil kunne findes i tabel 10.5.1.



**Figur 10.11.** På figuren ses hovedvinduet for Kontrolinterface-programmet

<sup>5</sup>FiXme Note: tabellens bredde skal fixes.

## Hovedvinduets elementer

---

<b>1: Forsinkelse i sekunder</b>	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.
<b>2: Nedlukningsknap</b>	Anvendes til at lukke programmet. Programmet åbner dialogen som ses i ??
<b>3: Vandballasttankene</b>	Her kan status for vandballasttankene aflæses. Niveauet er 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.
<b>4: Hældningssensor</b>	Værdien for hældningen af skibet angives i antal grader og i hvilken side skibet hælder.
<b>5: Reguleringsstatus</b>	Her angives hvorvidt automatisk eller manuel hældningsregulering er aktiveret. Der vil altid kun være en og kun en af disse aktiveret. Derfor vil der altid være en rød og en grøn indikator tændt. I dette eksempel er den automatisk hældningsregulering aktiveret.
<b>6: Forbindelser</b>	Indikerer hvorvidt der er forbindelse til Styringsmodulet og serveren. Dataforbindelse 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.
<b>7: Automatisk reguleringsknap</b>	Ved tryk på denne knap vil man komme til dialogen på ?? såfremt automatisk styring ikke er aktiveret. Hvis den er aktiveret og man trykker på knappen vil dialogen på figur ?? fremkomme.
<b>8: Manuel reguleringsknap</b>	Bringer dig til dialogen på figur ??
<b>9: Aktivitetslog</b>	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.

6

### 10.5.2 VBTE

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

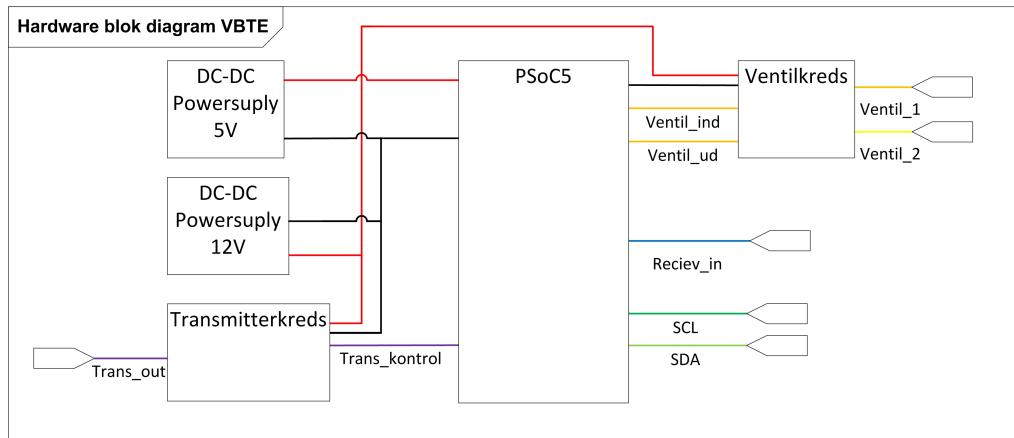
#### Hardware

Til VBTE'en er der blevet designet hardware til at styre de to ventiler samt den keramiske ultralydstransmitter 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. Hardware designprocessen til VBTE'en gik igennem 3 faser:

<sup>6</sup>Fixme Note: Tabellen "Hovedvinduets elementer" skal fixes både med bredde og streger.

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

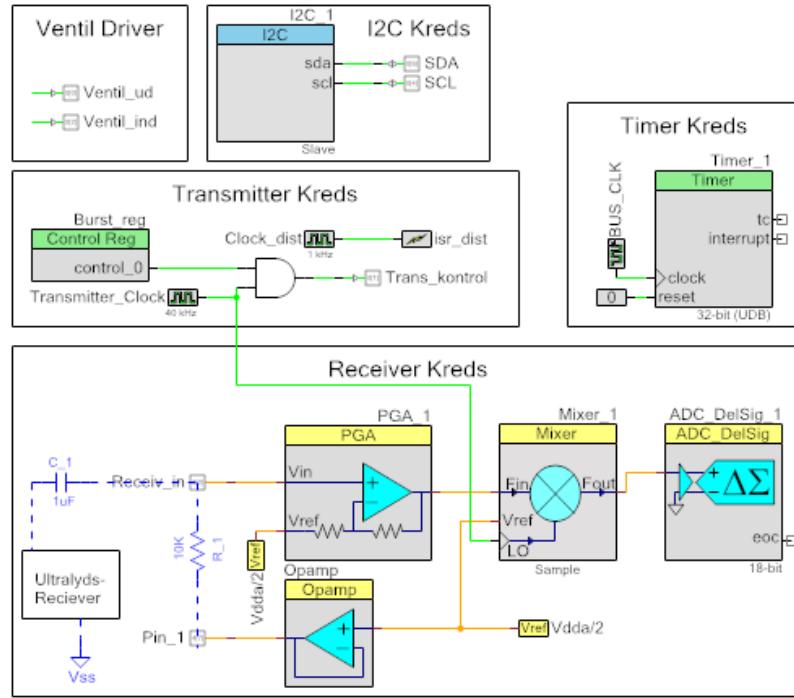
Gennem disse faser er designet blevet udfærdiget. Fremgangsmåden er anvendt for at overskueligtgøre systemet og lette arbejdet ved at dele systemet op i små dele. På *figur 10.12* ses det overordnede design af VBTE'en. Der vil i rapporten tages udgangspunkt i ventilkredsen samt receiverdriveren på PSoC'en.



**Figur 10.12.** 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 ligges vægt på transmitterkredsen samt receiverkredsen. På figur 10.12 ses hardware designet på PSoC'en.



Figur 10.13. Hardware på PSoC'en.

### Transmitterkreds

Transmitterkredsen står for at sende burst samt at tidsindstille hvert burst. Dette opnåes med to clocks, en AND gate, en output pin samt et kontrol register. Transmitterclocken er indstillet til 40kHz da ultralydstransmitteren, ifølge databladet, opererer 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 en 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 på 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.

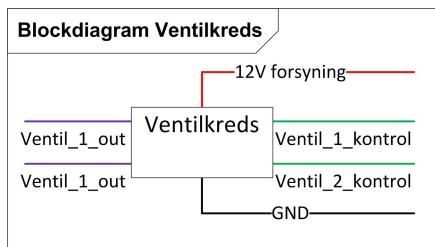
Efter signalet er løftet bliver det forstærket op af en PGA. Efter undersøgelser, hvor der blev sendt bursts, er der valgt en forstærkning på 16, da der ca. modtages et signal på 200 mVpp.

Herefter bliver signalet mixet med 40 kHz og filtreret. Efter mixeren giver det, stort set, en DC og summen af de to signaler. Filteret er inbygget i Delta-Sigma ADC'en og har en knækfrekvens ved  $\frac{Samplefrekvens}{3}$ . For at være sikker på at det meste af signalet er kommet

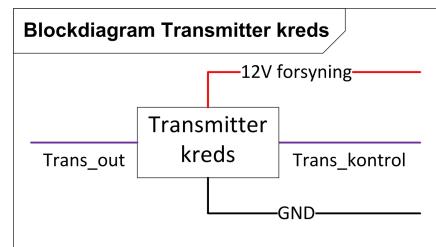
over regnes der en opladningstid på filteret til 1/4 ms. Efter undersøgelser af signalet ind på ADC'en blev en spænding på 0,3 V.

### Hardware uden for PSoC'en

Uden for PSoC'en er der lavet 2 hardwareblokke. Disse har til ansvar at omsætte et kontrolsignal til en større spænding over de respektive enheder.



**Figur 10.14.** Hardwareblok for ventil



**Figur 10.15.** Hardwareblok for transmitteren

### 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>7</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 darlington kobling 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.

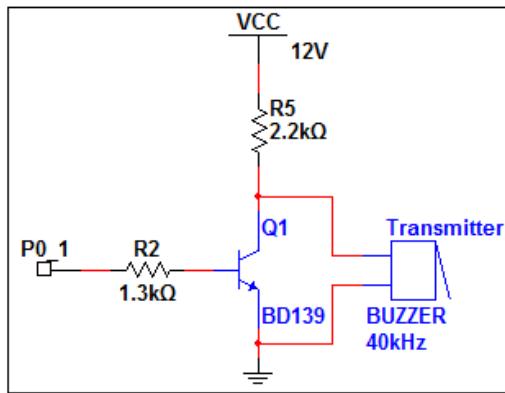
### Transmitterkreds

Transmitterkredsen anvender ultralydstranduceren som en højttaler. Denne er koblet over transistoren (Her anvendes også en BD139).

<sup>7</sup>Se bilag/BD139.pdf

Via. kontolsignalet fra PSoC'en, der bliver sendt ud ved 40 kHz, bliver der sat en spænding over transduceren.

Forsyningen hertil trækkes fra samme forsyning som ventilkredsen der bliver leveret fra powersupply'en. Ud fra databladet er der aflæst en ohmsk modstand på omtrent 10 k $\Omega$ . Derfor kan PSoC'en uden problemer selv trække denne kreds.

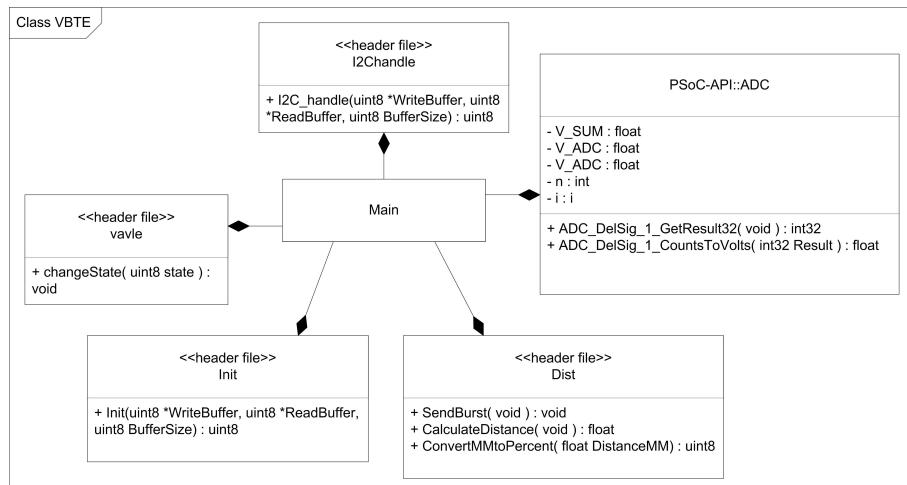


**Figur 10.16.** Transmitterkreds

For yderligere detaljer vedrørende hardware på VBTE henvises til *Deltajeret hardware design*.

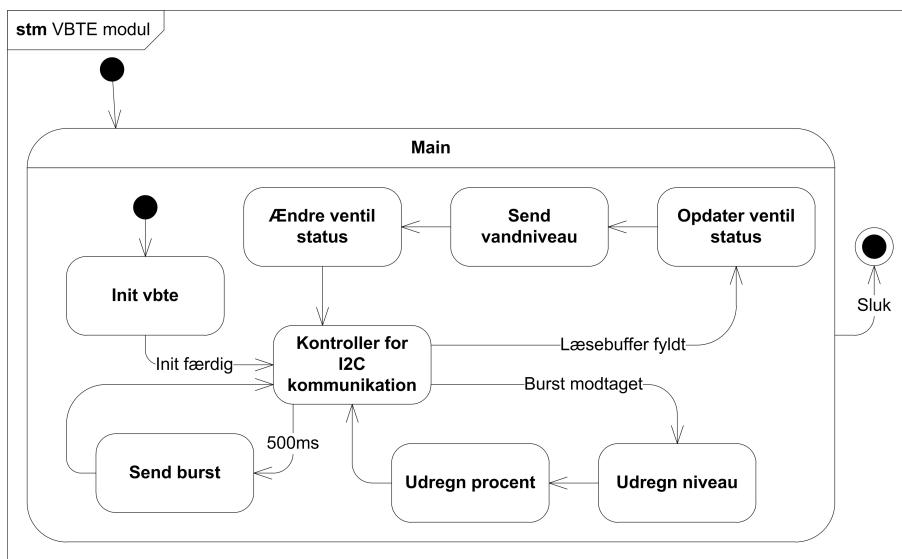
## Software

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 10.5.2 ses klassediagrammet for VBTE modulets software.



**Figur 10.17.** 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 detaljeredt software design. Selvom softwareen er skrevet i C er der lavet klasser for alle h-filer for at gøre systemet overskueligt. For detaljerede metodebeskrivelser henvises til detaljeredt software design. For at overskueligtgøre funktionaliteten af softwaren anvendes statemachinediagrammet. Dette viser de forskellige tilstande programmet til enhver til kan befinde sig i.



**Figur 10.18.** Statemachine for VBTE software

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 receiveder bliver opnået når flaget hertil bliver sat. Dette flag bliver sat inde i ADC'en interruptrutine når den mäter en værdi der indikerer at et burst er modtaget.

Read complete betingelsen bliver kontrolleret inde i I2C\_handle. Heri aflæses data fra SM og omsættes til et state for ventilerne. Herefter fyldes afstanden for tankene ind i readbufferen for I2C'en med henblik på at den sendes retur. Herefter ændres ventil tilstanden så den passer med den modtagede tilstand.

For yderligere detaljer om software for VBTE henvises til detaljeret design dokumentation.

### 10.5.3 SM

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

#### Hardware

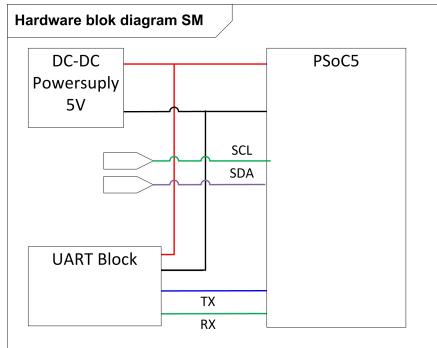
SM modulets hardware består af en konverteringskreds og en PSoC. På PSoC'en er monteret et Kionix KXSC7-2050 accelerometer. Konverteringskredsen anvendes til at sende UART signaler fra PSoC til en KI modulet. Accelerometerets x-akse anvendes til hældningsmålinger for hældningssensorblokken.

Designfasen til SM er delt op i 3 faser:

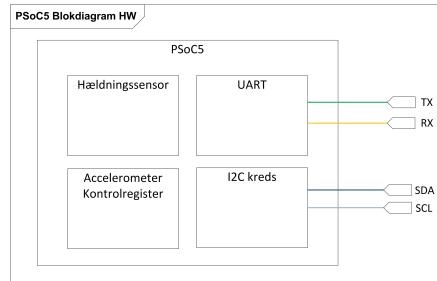
1. Overordnet design
2. Nedbrydning af blokke
3. Opbygning af design

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 10.19* ses det Overordnede design og på *Figur 10.20* ses PSoC blokken i SM. Der bliver efterfølgende taget udgangspunkt i Hældningssensoren.

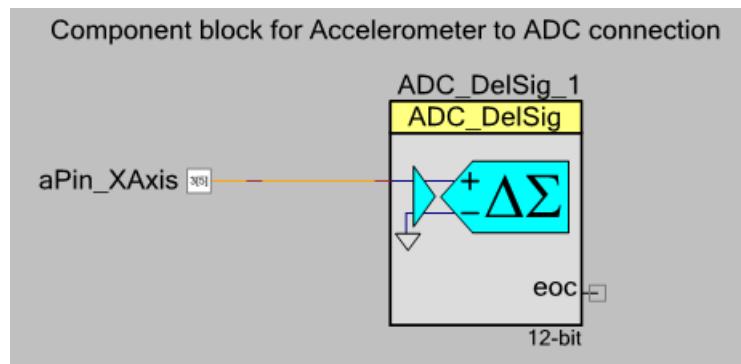


*Figur 10.19.* Hardware blok for SM



*Figur 10.20.* PSoC blok for sm

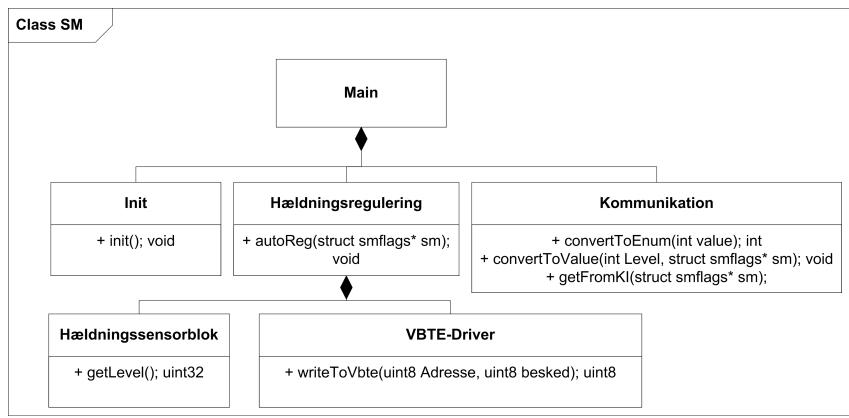
Hældningssensoren består af 2 komponenter: det førnævnte accelerometer samt en DelSig ADC internt i PSoC'en. Valget af accelerometer kommer af at have lavet en række prototyper der ikke mødte vores krav, hvilket accelerometeret i PSoC'en gjorde. Valget af ADC faldt på en DelSig da, den er meget støj immun grundet det indbyggede lavpas filter og har en høj opløsning. Valgte komponenter er illustret på *figur 10.21*. 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 afsnit 13.1.5 *Detaljeret hardware design* i Bilag.



*Figur 10.21.* Hældningssensorens implementering

## Software

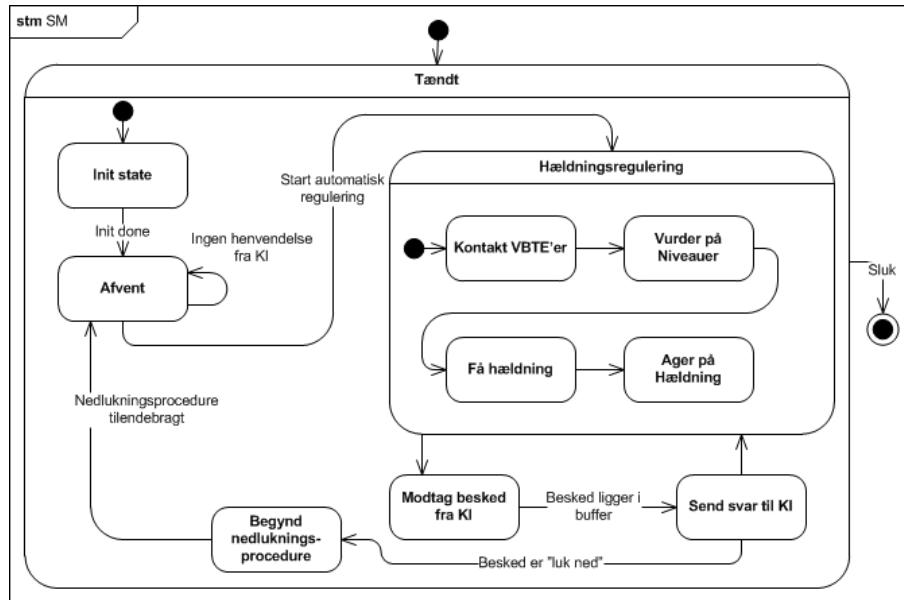
SM software behandler den konverterede data fra ADC'en samt kommunikation med VBTE og KI modulernerne. Sofwarens udvikling har fulgt de samme faser som hardwaren, hvilket har ført til letforståelig og læselig kode. Softwaren er illustreret på *figur 10.22*. Der vil efterfølgende blive taget udgangspunkt i et statemachine samt funktionen `getFromKI`.



**Figur 10.22.** Klassediagram for SM

## SM funktionalitet

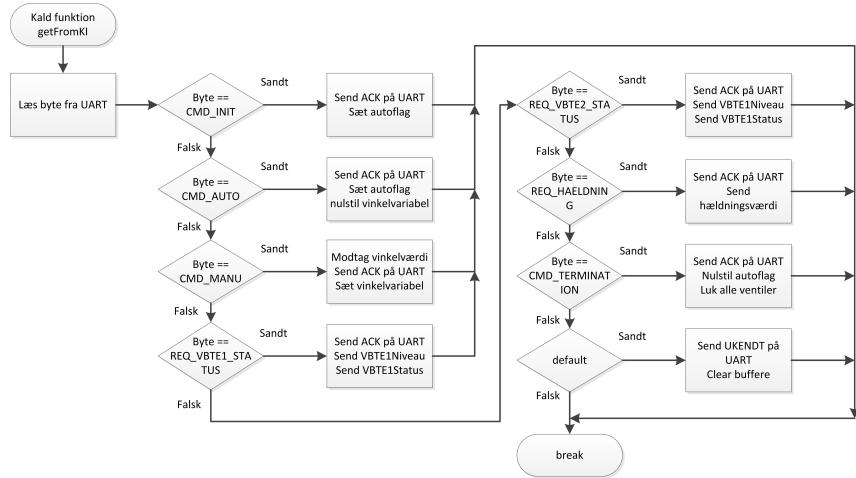
Funktionalitet af SM kan bedst beskrives med et state diagram. På *figur 10.23* ses state machine diagrammet for SM modulet. Diagrammet indeholder hældningsregulering state der beskriver den automatiske og manuelle regulering i systemet. Den automatiske regulering styres med et flag. Den manuelle styres med en værdi der bliver sat. Flaget bliver sat når SM opnår forbindelse med KI modulet i 'Afvent' statet. Flaget bliver nulstillet hvis KI sender besked til SM modulet om at det skal termineres, hvorefter SM går tilbage i 'Afvent' statet. Behandlingen af beskeder fra KI i 'Modtag besked fra KI' og 'Send svar til KI' states ses i følgende afsnit *getFromKI*.



**Figur 10.23.** State machine for SM

getFromKI

Funktionen er bedst beskrevet med et flowdiagram:

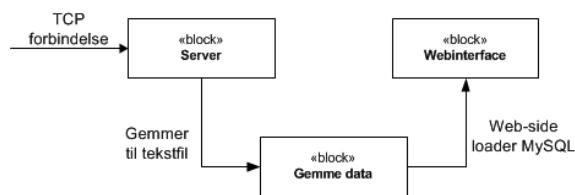
**Figur 10.24.** Flowdiagram for funktionen getFromKI

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

#### 10.5.4 Database

Denne section beskriver design og implementering af database delen som er lavet på baggrund af kravspecifikationen og systemarkitekturen.

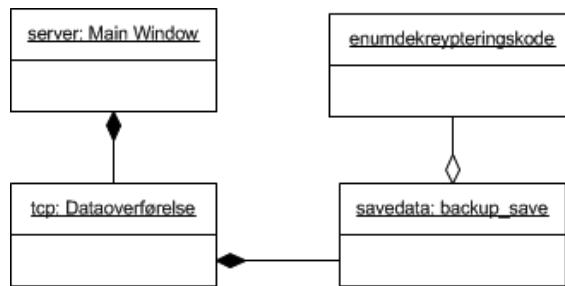
Databasen er blevet opdelt i 2 dele som håndtere denne side. Databasen er opdelt i server og Webinterface.

**Figur 10.25.** Illustrere overordnet hvordan databasen fungere

#### 10.5.5 Serveren

Denne section beskriver design og implementering af serveren. Opbygningen af serveren er gjort sådan at hver element i serveren er implementeret som en klasse, hvilket er illustreret på figur 10.26.

Det gælder at når KI forsøger at kontakte databasen, tilsluttes denne via tcp som benytter sig af socket. Når tilslutningen er gjort kan der overføres data fra KI til databasen. Data bliver gemt i en tekst fil, der indlæses af Webinterface.



**Figur 10.26.** Overordnet illustration af serveren funktionalitet

### Serverens klasser

Server	Er hovedvinduet på serveren. Information til brugeren udskrives herfra og knapper er implementeret.
TCP forbindelse (tcp)	Opretter en mulig tcp forbindelse hvor skibet kan tilslutte sig og overføre data.
Gemme data (savedata)	Gemmer data til en tekst fil.
De kryptering (enum-deKrypteringskode)	Dekryptere level fra sm så data bliver i grader

**Tabel 10.3.** Serverens klasser

### Hovedvindue for serveren

Dette giver et kort billede på hvordan serverens funktion ser ud. Den fulde model kan ses i appendix Serverens GUI giver mulighed for at brugeren kan se at serveren køre og hvilke



**Figur 10.27.** Billed af serverens hovedvindue

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 da det er 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.

### Hovedvinduets elementer

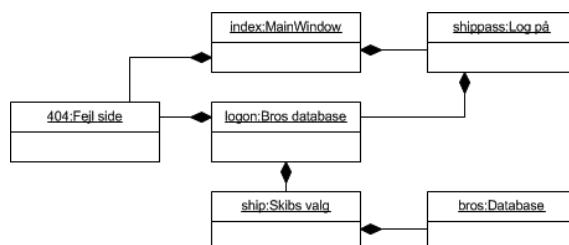
<b>1: Info</b>	Brugeren får her udskrevet beskeder om serveren. Brugeren kan se om KI har tilsluttet sig og om der er blevet overført data desuden vil fejl meddelelser blive udskrevet her. Dialog boksen er lavet sådan at den nyeste besked altid står øverst se 10.27
<b>2: Database</b>	Anvendes til at lade brugeren tilgå den web baserede database som ses i ??
<b>2: Luk serveren</b>	Anvendes til at lukke programmet. Bruges til nedlukning

### 10.5.6 Webinterface

Denne section beskriver design og implementering af web siden.

Opbygningen af Webinterface er gjort hovedsageligt i php hvor dette gav mening. Desuden er html og styles (css) blevet benyttet til at sikre ens stil på hele web siden og for simpelt at kunne velgehoides. For at sikre mindst data transmission imellem computer og server er princippet i ajax blevet benyttet.

Når havneterminalen ønsker at se data om skibe der er opkoblet på BROS kan denne tilgå disse via den web baserede database. Når serveren køre kan brugeren tilgå databasen via knappen "Database" ellers kan denne tilgås via ip adressen dette giver mulighed for at tilgå siden via internet hvorved at behovet for adgangskode ved log in er nødvendig for at sikre data imod uautoriseret brug. Efter log ind på siden kan brugeren vælge hvilket skib der skal benyttes. Når skib er valgt kommer brugeren ind på den valgte database. Her kan brugeren se ID på skibet vandtilstand i ballast tanke, hældning og tid siden sidste opdatering (for nærmere se appendix<sup>8</sup>). Webinterfacet checker hvert 5 secund om der er kommet ny data. Ved ny data vil et script sørge for at denne bliver gemt i MySQL databasen. Webinterfacet tilgår den angivne tabel i den angivne database.



*Figur 10.28.* 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 (log-on)	Denne side bliver kaldt efter at brugeren er logget på. Her vælger brugeren hvilket skib denne ønsker at få vidst data for.

*Tabel 10.4.* Web sidens opbygning

<sup>8</sup>Fixme Note: link til appendix

## Skibs data i database

Et billede af hvordan data fremvises for brugeren. For de andre sider på websitet kan disse ses i appendix

The screenshot shows a web-based interface for managing ship data. At the top, there's a header with 'December 2012' on the left and 'Log af' on the right. Below the header, the word 'BROS' is prominently displayed, followed by 'Webinterface'. A green button labeled 'Log af' is visible on the left. In the center, a table titled 'Skibs data for Martha' is shown. Above the table, a message says 'Database sidst opdateret 16/12/12 : 16:59:37'. The table has five columns: 'ID på skib', 'Styrbord vandstand', 'Bagbord vandstand', 'Hældning i grader', and 'Skib tilsluttet'. The first column contains the value 'TERMINATED'. The other columns show data for six entries, all labeled 'Martha'. The data is as follows:

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

At the bottom of the interface, there's a 'Log af' button and the text 'BROS © 2012'.

*Figur 10.29.* Billed databasen BROS for skibet Martha

Data fra skib i databasen

### 1: Skib og 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 styrbord ballasttank.

### 4: Bagbord vandtanksniveau

Der vises hvor meget vand der er i bagbord ballasttank.

### 5: Hældning i grader

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

### 6: Skib tilsluttet

Der vise hvor lang tid i secunder der er gået fra overførelsen før til denne.

## 10.6 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 samlerede resultat. Der vil i afsnittet blive henvist til testresultater dokumenteret i hhv. "Enhedstest", "Integrationstest" og "Accepttest".

**10.6.1 KI****10.6.2 Database og webinterface****10.6.3 SM****10.6.4 VBTE**

VBTE modulet er fuldt implementeret, men med problemer. Afstanden målt af ultralydssensoren er upålidelig og det er svært at måle over ret lange afstande. Derudover er det svært at vide hvordan og hvor meget af lyden der bliver reflekteret. Under kontrollerede forhold var afstande "nemme" at måle og der kom også derfor gode resultater ud af enhedstesten. Men skulle systemet sættes under andre forhold blev det straks meget sværere at få systemet til at fungere. Jeg vil dog gerne fremhæve resultatet at have fået opbygget en ultralydssensor udelukkende ud fra en transducer og receiver.

Derudover er der udlagt et print inklusiv testkredsløb der indeholder en 2x16 LCD skærm og kontakter til at skifte I2C adresse.

**10.6.5 powersupply****10.6.6 Samlede resultat og vurdering af resultater****10.7 Opnåede erfaringer****10.7.1 Generelt**

Gruppens fem medlemmer har været fordelt på fire forskellige projektgrupper i de tidlige projektforløb. Alle medlemmer er gået ligeværdigt ind i projektet med en positiv indstilling. Det har dog vist sig at være en større opgave at skabe en ny gruppe, end gruppen umiddelbart havde forestillet sig - og gruppen har ikke været sig denne opgave bevidst nok. Det har været en opgave for gruppen at forsøge at tage det bedste fra hver af de fire tidlige grupper og stykke sammen til en ny. På trods af at grupperne har gennemgået samme undervisning er det meget forskelligt hvordan de forskellige grupper har anskuet processen, rollefordelinger, metoder og faser. Der har også skullet afstemmes forventninger og gruppen har ligeledes haft brug for at se hinanden an fagligt. Det har således været sværere i projekts opstartsperiode at vurdere hvor stor en opgave gruppen har kunnet påtage sig. Travlheden i slutningen af implementeringsfasen kan muligvis være en konsekvens heraf.

**Faseforståelse**

I slutningen af projektforløbet blev forståelsen af systemarkitekturens formål væsentlig forbedret. Vi ved ikke hvor langt vi er kommet men det er vores helt klare opfattelse at vi er kommet et langt stykke videre i forståelsen af systemarkitekturen. Den forståelse vi har nu ligger på et markant højere niveau end den forståelse nogen af de fire grupper havde i de tidlige projektforløb. Det er ikke blot systemarkitekturen men hele samspillet imellem hovedfaserne. Formålet med kravspecifikationen set fra kundens synspunkt, overgangen derfra til systemarkitekturen ved hjælp af applikations og domænemodeller. Herfra virker springet til design og implementeringsfasen som en leg.

Den overgang var på ingen måde en leg for gruppen. Derimod blev faserne arkitektur og

design nærmest sprunget let og elegant over. Dette har bidraget til en implementeringsfase hvor modulerne godt nok er blevet implementeret - og hvor de virker efter hensigten når man ser på dem individuelt. Problemerne begynder dog at opstå når modulerne begynder at blive sat sammen. Her bliver det tydeligt at arkitekturen og designdelen ikke har været tilfredsstillende nok. Spørgsmål som "Hvordan gør dit modul, når dette sker?" kan ikke blive besvaret i dokumentationen, men må stilles til personen der har implementeret modulet. I slutfasen er use cases og systemarkitekturen blevet kigget igennem en ekstra gang og der er blevet kigget nærmere på applikationsmodeller og domænemodeller. Gruppen forsøgte også det tidligere i forløbet, men forståelsen var ikke nok på plads, og motivationen var ikke stor nok til at det blev forsøgt ihærdigt nok. Dette kan givetvis også skyldes den lidt manglende kommunikation i projektet hvor der blev arbejdet i fem forskellige lejre - se erfaringen "uddelegering af opgaver".

Det ligger dog givetvis også en pointe i at domæne- og applikationsmodeller er nemmere at opbygge for et system, som man har et indgående kendskab til fordi det er blevet implementeret. De frustrationer vi oprindeligt havde ved applikationsmodeller og domænemodeller udskød vi nok angiveligt til vi sad i implementeringsfasen. Og så igen når vi sad i testfasen og skulle sætte modulerne sammen. Erfaringen må være at jo tidligere i forløbet vi tager tyren ved hornene, jo "billigere" er det i arbejdstid - eller sagt på en anden måde: Selvom opgaven ser stor ud, så bliver den kun større af at vente med den.

### **Pointer og brainstorm**

Vi har implementeret for hurtigt

Vi er kommet til at hoppe fra teknologiundersøgelser til implementering - og dermed sprang vi implementerings og designfasen over.

Vi har ikke været motiveret for at arbejde med systemarkitekturen fordi vi ikke havde forståelsen for nødvendigheden af den. Vi spurgte os selv: "Hvad bidrager den med?". Vi vidste det ikke.

Vi har tidligere i de fire forskellige grupper kun svagt - eller helt undladt - at anvende domæne- og applikationsmodeller. Dette har skyldtes at processen har været tung og at der ikke har været en godbid i den anden ende - vi har ikke kunnet se frugten af arbejdet. Da vi var i systemarkitekturfasen var vi allerede dybt opslugt af teknologiundersøgelser - undersøgelser som begyndte mere at ligne implementeringer. Derfor var incitamentet for at begynde at strukturere systemet lavt. Gruppen var så opslugt af implementeringen at de forsøg der blev gjort for at få hul på bylden blev overhørt. Efterhånden stoppede forsøgene og systemarkitekturen blev for den største del af gruppen henlagt. Dette kan skyldes formegentlig at for størstedelen af gruppens medlemmer har systemarkitekturen tidligere været et obligatorisk arbejde uden egentlig bidrag til processen - resultatet har simpelthen ikke været tydeligt nok.

Systemarkitekturen og designfasen har været i højere grad været en hæmsko end et redskab i processen. Det har været en obligatorisk dokumentationsdel som lagt hen af vejen er blevet lavet i slutningen af - eller i bedste fald ad hoc med - implementeringsfasen. Vi har ikke fået øjenene op for hvad systemarkitekturen har at bidrage med. Vi har ikke set fordelene ved den. Arbejdet har ikke været time-efficient nok. Og så har implementeringen bare været så meget mere interessant.

## Faseforståelse 2

En stor del af læringen i forbindelse med et projekt ligger i processen. Det udviklede system er nærmere et middel end det er et mål for semesterprojektet. Målet hedder bedre faseforståelse. Forståelse for de faser en udviklingsproces indebærer. Hver fase har sit eget bidrag til processen. Igennem projektarbejdet stifter gruppen gentagne gange bekendtskab med faserne. Hele uddannelsen er en iterativ proces med ét mål: En god forståelse af udviklingsprocessen af et system.

For hver af gruppens medlemmer har der tidligere været en eller flere åbenbaringer i løbet af et semesterprojekt. Et tidspunkt hvor en fordelen ved at anvende en given metode går op for personen og bliver soleklar. Åbenbaringer giver en fantastisk følelse af at gennembryde en mur, at have gjort et fremskridt. Dette kan godt ske flere gange for samme metode. Dette behøver ikke at betyde at de tidligere åbenbaringer har været forkerte, men blot at man er kommet et niveau højere op i sin forståelse. Hver åbenbaring er således noget positivt.

For dette semesterprojekt har den helt store åbenbaring indtruffet sent - om end med meget stor kraft. Åbenbaringen er to-delt, men er faldet i samme tidsrum, da de har en tæt kobling.

Domæne- og applikationsmodeller i systemarkitekturen har tidligere været et mystisk fænomen for en stor del af gruppen. Det har ikke været tydeligt hvad formålet - og frugten - var med det arbejde der blev lagt i modellerne. De har været frustrerende at udarbejde og det har ikke været klart hvordan de dernæst skulle anvendes. Således er incitamentet for at lave dem faldet, og flere af de tidligere grupper har slet og ret ikke har dem med i deres arkitektur. Omend det kan have været en korrekt beslutning at udelade dem i tidligere projekter har gruppen sent i dette projekt set hvilket fantastisk redskab det kunne have været tidligere i processen. Da gruppen naturligt stødte på applikations- og domænemodeller i systemarkitekturfasen blev forsøget på at starte udarbejdelsen mødt med lavt engagement fra gruppen. Dette gjorde sig faktisk gældende for hele systemarkitekturfasen, omend i særlig høj grad for disse to modeltyper.

De to modeltyper blev så senere taget op igen af projektvejlederen. Der blev samtidig stilt nogle relevante spørgsmål til kravspecifikationen. Derpå blev kravspecifikationen og systemarkitekturen påny taget op ad gruppen. Her indtraf åbenbaringen en lørdag eftermiddag og kan sammenfattes til disse to sætninger:

Kravspecifikationen skal ses ud fra kundens perspektiv og dermed indeholde de informationer der for kunden er relevante. Domæne og applikationsmodeller er nøddeknekken til kravspecifikationen - et redskab der transformerer use cases til konceptuelle klasser der kan laves sekvensdiagrammer ud fra (Sekvensdiagrammer er top nice!)

Denne nyfundne forståelse medførte en revision af kravspecifikation og systemarkitekturen med en stor mængde rettelser til følge. Det blev hurtigt tydeligt hvordan dette redskab, hvis anvendt korrekt, kunne have gjort at gruppen havde undgået en del frustration, problemer og smårettelser i slutningen af udviklingsfasen - især i testfasen. Enhedstesten gik fint. Integrationstesten gik dog ikke lige så gnidningsfrit: I implementeringsfasen opstod behovet for at aftale hvordan specifikke situationer tackles ofte og en aftale blev derfor lavet mundtligt som følge af manglen på et fælles referencepunkt i en stringent og udførlig systemarkitektur. Problemerne det medførte blev først opdaget i integrationstesten hvor

adskillige småændringer var nødvendige for at modulerne kunne kommunikere tilfredsstilende. I testforløbet blev behovet for en række mindre ændringer synliggjort. Dette er formegentlig naturligt for et testforløbet, men det er gruppens klare formodning at en del af disse valg burde have været truffet tidligere i processen - eksempelvis i arkitekturen.

### Rammer for projektet

Omed det kan være rart at have frie tøjler er det gruppens erfaring at tøjlerne for dette projektforløb har været for frie. Enten har opgaven været for uinspirerende eller også har gruppen været for fantasiløs. Det var i hvert fald svært for gruppen at finde et projekt, der blev fundet både interessant nok samtidig med at faglige niveau var højt nok.

### Uddelegering af opgaver

Gruppen har fra start ville uddelegerede opgaver og ansvar for at lette arbejdsprocessen. Systemet blev derfor opdelt i moduler med en ansvarlig tilknyttet. Modulerne blev uddelt efter interesse. Dette har været en rigtig god løsning for flere af gruppens medlemmer tidligere, men i dette forløb har det vist sig også at være en hæmsko. Det fælles projektarbejde blev hurtigt fem personer, der sad med hvert deres lille område af projektet. Den interne kommunikation udeblev, og sparringen blev mangelfuld på grund af manglende indsigt i hvad de resterende gruppemedlemmer lavede.

### Programmering

Gruppen har i dette projekt bestået af 5 elektronik ingenørstuderende. Dette har skabt nogle problemer med hensyn til programmeringsdeler af projektet. Gruppen har dog været god til at søge hjælp til problematiske opgaver, hvilket har gjort at programmeringstunge moduler er blevet implementeret korrekt.

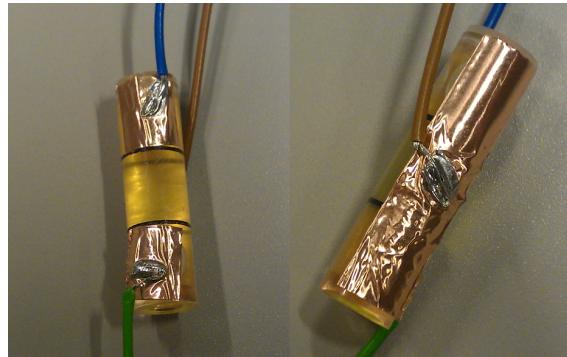
### Vigtigheden af en struktureret faseinddeling

For de fleste moduler i systemet startede implementeringen ret tidligt. Dette skyldtes for det første at det er denne fase gruppens medlemmer finder mest interessant, men ligeledes at det indledende arbejde gav blod på tanden til at afprøve de problemløsninger der langsomt begyndte at forme sig. Således gik en del af teknologiundersøgelsen ud på at afprøve nogle af disse problemløsninger, og før man har set sig om har man sprunget både systemarkitektur og designfasen over. Dette har for mere end et modul ført til en omskrivning af hele programmet op til flere gange. Disse omskrivninger har bl.a. været som følge af dårlig struktureret kode som følge af mange ændringer. Dette medførte hurtigt en besværliggørelse af læsningen af, og det videre arbejde med, koden. Tillige har det været svært at arbejde med koden i et tilfredsstillende tempo, da programmet ikke har været udformet og designet tilstrækkeligt inden implementeringsfasen. Erfaringen må være at et program hurtigt bliver større end hvad der kan overskues i hovedet af projektgruppens medlemmer på deres nuværende niveau. Vigtigheden af at faserne bliver benyttet er dermed større end først antaget.

### 10.7.2 Rettidig test

### 10.7.3 Udvikling af hældningssensor

Vi har startede med at udvikle på en prototype af en libellesensor vist på *Figur 10.30*. Vi



**Figur 10.30.** henning

kom frem til at den har en capacitet på omkring  $1 * 10^{-15}[F]$ . Det gør det praktisk umuligt at anvende da vores filter så har en alt for stor cutoff frekvens liggende over 3.0 MHz. Den høje frekvens giver en stor selvinduktion i vores ledning. Samtidig kan vi ikke lave sinus med denne frekvens med PSoC'en. Dette 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 upræcist i forhold til vores krav.

Vi har gennem et tredje semestersfag fundet ud af at PSoC'en indeholder et accelerometer. Vi valgte så at lave en prototype med det. Dette viste sig at være en god løsning.

### 10.7.4 Metoder

Gruppen var opsatte på at have læst review på projektets dokumentation, men de adspurgte grupper havde ingen interesse i at lave review udvekslinger. Dette har ført til at dokumentationen blev gennemlæst på et senere tidspunkt af gruppens vejleder. Grundet udskydelsen havde gruppen set sig nødsaget til at forlænge tidsplanen for at opdatere og forbedre foreløbigt færdig dokumentation. Dette førte til at design og, i sidste ende, implementering blev udskudt, hvilket medfører et stort pres imod slutningen af projektforløbet.

Gruppens opdeling i udviklingsmetoden har været meget flydende. Selv om der har været en projektleder og scrummaster har det ikke været nødvendigt, da gruppen har arbejdet godt sammen og diskuteret udviklingsmetoden til fuldest. Der har i forløbet været en udskiftning i koordinatorposten, hvorefter rumkoordinering har forløbet godt.

Gruppen har fået en bedre forståelse for eventuelt kommunikation med en kunde gennem flere iterationer af kravspecifikation.

### 10.7.5 Udvikling af afstandssensor

Ultralydsafstandsmålingen blev valgt for at prøve noget ingen havde kendskab til i projektet. Man kunne have valgt at købe en færdiglavet enhed men det blev vurderet at dette ikke gav ret meget faglig erfaring. I e-lab havde de nogle rå tranducere og receivere

liggende og der blev derfor valgt at forsøge med at udvikle en ultralydsafstandsmåler. Der blev i starten lavet en del teknologiundersøgelse inden for ultralyden og vi testede tranducere og receivere for at se hvordan de aggerede. Vi fandt at der var en del destruktive reflektioner og vi overvejede at fylde akustisk skum i toppen af tankene, men da det skum vi kunne finde var elektrisk ledende kunne vi ikke anvende det. Jeg kom frem til at et burst skulle sendes for at måle afstanden og en puls på  $250\mu\text{s}$  blev valgt, hvilket svarer til 10 perioder. Receiverkredsen blev udviklet med viden fra MSE kurset fra sidste semester omkring mixere og operationsforstærkere. Den største erfaring opnået gennem dette forløb må være at man bare, som hovedregel, bare selv skal købe noget der passer til formålet og implementere dette i projektet. Dette giver mere plads til at lave et større system, simplere. Derudover gav det god erfaring i mixerelektronik og ultralydsteknologi, som er meget besværligt at arbejde med.

#### 10.7.6 Udvikling af Databasen

#### 10.7.7 Bulletpoints til skrivning af OpXP

- Nyt samarbejde og nye gruppemedlemmer (Hvor godt vi har sammensat en ny gruppe)
- Hvor mange iterationer vi har været igennem (Forbedringer af start design). Ingen vigtigheden af at man altid kan gå tilbage og at dokumentation ikke er støbt i beton.
- Opnåede erfaring omkring faserne (Hvor vigtige de er og hvordan vi efterfølgende har haft det nemmere på faserne)
- Samarbejde med kunden. Inddrag kunden(Carl) i processen.
- Forståelse for test (Test er ikke til for bare at vise systemet virke men for at afsløre væsentlige mangler, hvilket har forbedret projektet (og produktet)).
- Uddeling af opgaver (Noget johnny sagde med at det var vigtigt der var en ansvarsperson bag hver modul).
- Udvikling (Sammenkogt erfaring af Johnny og Nicolai)

# Konklusion 11

---

<sup>1</sup> Det er desværre ikke lykkedes for gruppen at implementere hele systemet, som det var ønsket. Grundsystemets krav, opgivet i kapitel 9, er desværre ikke blevet opfyldt. Dette har dog ikke forhindret gruppen i at have en meget lærerig proces. Således er der blevet gjort mange overvejelser om løsninger i systemet - det gælder især for hældningssensoren og vandballastankenes niveausensorer. Der er også blevet implementeret tre forskellige kommunikationsprotokoller i projektet - kommunikationsprotokoller med hver deres overvejelser. Årsagen til den manglende fuldendte implementering skal findes i de problemer der kom sidst i implementeringsfasen. Der blev således brændt tre PSoC's af på to dage. Gruppen valgte på grund af tidspres og mangel på PSoC's at stoppe det videre testforløb. Det er dog gruppens klare formodning at problemet ligger i hvordan systemet er sammenkoblet. Da der røg to PSoCs på en gang var der tilknyttet tre PSoCs fra tre forskellige computere. Dette har resultateret i tre forskellige "stel", der har indført en form for ustabilitet i systemet. Problemerne kunne være opdaget tidligere i projektforløbet hvis gruppen havde påbegyndt testningen tidligere. Dette var dog ikke muligt da der har været ret mange last-minute tilføjelser til projektet - bl.a. tre forsyningsboards. Dette kan igen skyldes den mangelfulde strukturering af projektets faser.

---

<sup>1</sup>Fixme Note: Tanker



# **Forbedringer til systemet**

---

**12**



# Referencer 13

---

## 13.1 Artefakter

### 13.1.1 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 bilag. (Kravspecifikation.pdf)

### 13.1.2 Accepttestspezifikation

Accepttestspezifikationsdokumentet beskriver de tests der skal laves for at undersøge om de ønskede krav er opfyldt. Den fuldstændige accepttestspezifikation kan ses i bilag (Accepttest.pdf).

### 13.1.3 Systemarkitektur

Systemarkitektur dokumentet beskriver systemets HW/SW opbygning og grænseflader. Den fuldstændige systemarkitektur kan ses i bilag (Systemarkitektur.pdf).

### 13.1.4 Integrationstestspezifikation

Integrationstestspezifikation beskriver de test der skal laves for at undersøge hvorledes de forskellige komponenter kan kommunikere. Den fuldstændige Integrationstest kan ses i bilag (Integrationstest.pdf).

### 13.1.5 Detaljeret design

Det detaljerede design dokument beskriver hvordan HW/SW er designet og hvordan systemets komponenter fungerer. Det fuldstændige Detaljeret design dokument kan ses i bilag (Detaljeret Hardware design.pdf og Detaljeret Software design.pdf).

### 13.1.6 Enhedstestspezifikation

Enhedstestspezifikation beskriver de tests der skal laves for at undersøge om de forskellige stubbe af systemet fungere hensigtsmæssigt. Den fuldstændige enhedstestspezifikation kan ses i bilag (Enhedstest.pdf).

## 13.2 Hjemmesider

<http://www.docs.google.com>  
<http://office.microsoft.com/en-us/visio/>  
<http://www.maplesoft.com/>  
<http://www.ni.com/multisim/>  
<http://kurser.ih.a.dk/eit/eit-lab/lagerliste.htm>

## 13.3 Liste over bilag på CD

Komponentliste.pdf  
SCRUM.xls  
Logbog.pdf

### 13.3.1 Kode

#### KI

hest

#### SM

hestning

#### VBTE

honning

#### Server

### 13.3.2 Dokumentation

Accepttest.pdf  
Arkitektur.pdf  
Detaljeret\_hardware\_design.pdf  
Detaljeret\_software\_design.pdf  
Enhedstest.pdf  
Integrationstest.pdf  
Kravspecifikation.pdf

### 13.3.3 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

### 13.3.4 Billeder

hvis vi har billeder af vores produkt!