

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

19. december 2012



# Resume 1

---

Udkast fra Johnny: Denne rapport for projekt på 4. semester E på IHA præsenterer en gennemgang af gruppens projekt: "Bias Reducing Operating System". Formålet er selvstændigt at definere et projekt der inddrager discipliner fra semestrets fag. Hertil var der stillet krav til enkelte elementer der som minimum skulle inddrages i projektet. Der er i projektet lagt stor vægt på processen, men også i at opnå stor 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 slagsside regulering af bulkskibe. Dette kan med fordel implementeres på bulkskibe under lastning og losning for at hjælpe hele processen herunder. Der er fokuseret på 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 foreskrevet i grundsystemet (se *tabel 8.1*). Der er gennemført flere iterationer der gradvist har øget systemet og dokumentationens præcision. Gruppen har opnået god erfaring med teamwork, arbejdsstrukturering samt faglig perspektivering.



# **Abstract** 2

---

# 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	Krav til kvalitet . . . . .	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 . . . . .	23
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 . . . . .	27
9.3.5	Valg af database . . . . .	27
9.3.6	Webinterface . . . . .	28
9.3.7	Jura . . . . .	28
9.4	Systemarkitektur . . . . .	30
9.4.1	Stadier for konceptuelle klasser . . . . .	32
9.5	Design og Implementering . . . . .	35

9.5.1	Kontrolinterfacet . . . . .	35
9.5.2	VBTE . . . . .	39
9.5.3	SM . . . . .	44
9.5.4	Strømforsyning . . . . .	47
9.5.5	Database . . . . .	50
9.5.6	Serveren . . . . .	50
9.5.7	Webinterface . . . . .	52
9.5.8	MySQL . . . . .	54
9.6	Resultater . . . . .	55
9.6.1	KI . . . . .	55
9.6.2	Database og webinterface . . . . .	55
9.6.3	SM . . . . .	56
9.6.4	VBTE . . . . .	56
9.6.5	Strømforsyning . . . . .	56
9.6.6	Samlede resultat og vurdering af resultater . . . . .	56
9.7	Opnåede erfaringer . . . . .	57
9.7.1	Gruppen . . . . .	57
9.7.2	Agile udviklingsmetoder . . . . .	58
9.7.3	Udvikling af nye Komponenter . . . . .	58
9.7.4	Test . . . . .	58
9.7.5	værktøjer . . . . .	58
9.8	Udviklingsværktøjer . . . . .	58
<b>Kapitel 10</b>	<b>Konklusion</b>	<b>61</b>
10.0.1	Bulletpoints til Konklusion . . . . .	61
<b>Kapitel 11</b>	<b>Forbedringer til systemet</b>	<b>63</b>
<b>Kapitel 12</b>	<b>Referencer</b>	<b>65</b>
12.1	Artefakter . . . . .	65
12.1.1	Kravspecifikation . . . . .	65
12.1.2	Accepttestspezifikation . . . . .	65
12.1.3	Systemarkitektur . . . . .	65
12.1.4	Integrationstestspezifikation . . . . .	65
12.1.5	Detaljeret design . . . . .	65
12.1.6	Enhedstestspezifikation . . . . .	65
12.2	Hjemmesider . . . . .	66
12.3	Liste over bilag på CD . . . . .	66
12.3.1	Kode . . . . .	66
12.3.2	Dokumentation . . . . .	66
12.3.3	Datablade . . . . .	67
12.3.4	Billeder . . . . .	67



# **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. Ud over rapporten er der blevet udarbejdet 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

---

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 <sup>1</sup>.

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 fem udviklingsfaser. Udviklingsfaserne er som følger:

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

## 4.1 Læsevejledning

<sup>2</sup> 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 9 og 10<sup>3</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<sup>4</sup>. Her gennemgås opgaveformuleringen. Opgaveformuleringen indeholder minimumskrav til projektet og er givet til gruppen af vejlederen. 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

---

<sup>1</sup>Fixme Note: reference til problemformuleringen i projektformuleringen

<sup>2</sup>Fixme Note: Referencer er hardcoded og potentieligt forkerte

<sup>3</sup>Fixme Note: lav dynamiske referencer

<sup>4</sup>Fixme Note: hardcoded reference. lav ordentlig

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 gennemgået. I afsnittene fra 10.2-10.5 nedbrydes hele projektet fra øverste abstraktionsniveau 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>5</sup>

---

<sup>5</sup>FiXme Note: Lave refrencerne til de enkelte afsnit.

# Projektformulering

5

## 5.1 Opgaveformulering

Opgaveformuleringen er givet af vejledere til projektet som retningslinier for projektet. 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 ofte mange resourcer på at kontrollere at skibet ikke får slagseite. Skibets ansvarshavende officer står på broen under hele lastningen/losningen og holder øje med at det 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 stor last placeret i modsatte side. For at sikre at havnen har et overblik over alle skibe i havnen, sender systemet statusbeskeder til en database på havnekontoret. Kontoret kan derfor tage kontakt til skibet, der har en alarm.

### Projektdefinition

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

- Systemet skal føre en log-fil hvori data omkring skibsnavn og status sendes til en database
- Systemet skal monitorere om skibet er i vatter
- Systemet skal agere i tilfælde af skibet er ude af vatter

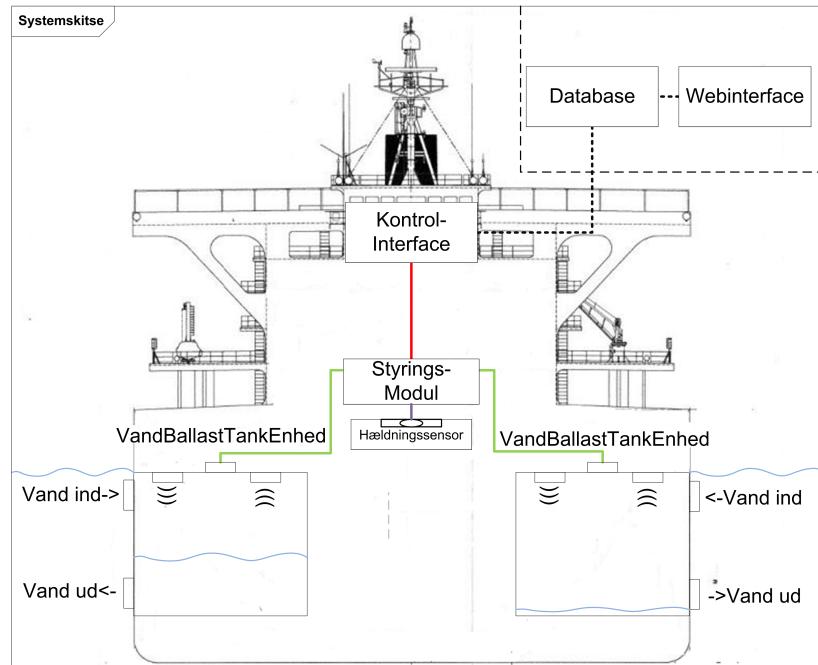


# Systembeskrivelse 6

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 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 slagseite. Dette kan gøres for at imødekomme en større slagseite 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 6.1. Systemskitse af BROS



# Kravspecifikation 7

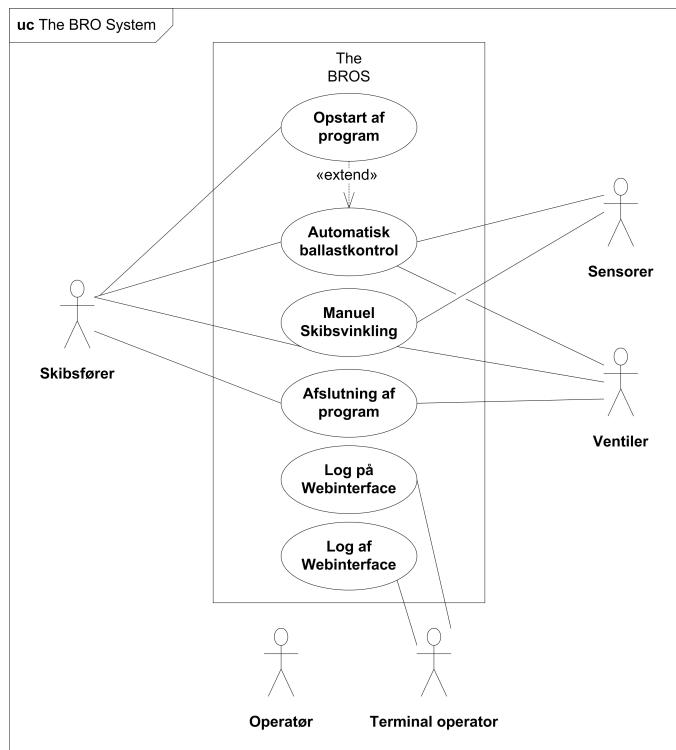
Kravspecifikationen er udarbejdet i begyndelsen af projektforløbet og omfatter use cases, ikke funktionelle krav samt kvalitets faktorer. For den fulde kravspecifikation, henvises der til dokumentations dokumentet.

## 7.0.1 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.

## 7.0.2 Funktionelle krav

Systemets funktioner er beskrevet vha. Use Cases. Før use cases udfærdiges nedskrives systemets akører. Der laves hertil en beskrivelse af deres funktion samt ansvar i systemet. I hver Use case er der en beskrivelse af en enkelt funktionalitet systemet skal have.



Figur 7.1. Use Case Diagram for BROS

Der er også beskrevet alternative hændelser, såfremt hældsesforløbet ikke forløber som planlagt. Alle use cases har fulgt samme fremgangsprocedure. Se kravspecifikationsdokumentet for alle systemets Use Cases. På *Figur 7.1* ses Use Cases for systemet.

Når systemet startes 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).

### 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 nogle 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 0.5 grader.
  - ◊ Ballasttank  
Vandniveauet i ballasttanken skal måles fra 0-100% med en nøjagtighed på ±2.5%-point.
- Alarmer:
  - ◊ Vandniveauet i ballasttanke bliver mere end 70%.
  - ◊ Hældning på skibet bliver større end ±5 grader.
  - ◊ Mistet eller ingen forbindelse internt i systemet.
  - ◊ Mistet eller ingen forbindelse til ekstern database.

### 7.0.4 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 Vandballasttankenenhederne, anvendes C. Til kontrolinterfaces og databasen anvendes C++. Til Databaselagring anvendes mySQL. Webinterfacet skal skrives i php og HTML 5.

### 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 via I2C protokollen.

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

---

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

### 7.0.6 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.



# Afgrænsning 8

---

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

## Afgrænsning givet:

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.

## 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 8.1. 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 Advarselssignaler
<b>Udvidelser:</b>	Måling af afstand til terminal-kaj Måling af dybdegang Manuel styring af ballast niveau Pålidelig kommunikation med ekstern enhed

**Tabel 8.1.** Grundsystem og Udvidelser til BROS

---

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

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



# 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 mange erfaringer fra de forskellige grupper har gruppen valgt at arbejde med projektet ud fra scrum princippet. Gruppen har lagt roller fast 9.1 og arbejdsmetoder. Projektet er blevet brudt ned i mindre dele og gruppens enkle medlemmer har fået ansvar for hver deres del og sparet med den i gruppen som blokken kommunikere med.

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 revirderet 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.

### 9.1.1 Rollefordelinger

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

**Tabel 9.1.** 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.

## 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 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. Nye sprint startes hver anden uge og 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 9.1* vises det 2. sprint.

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

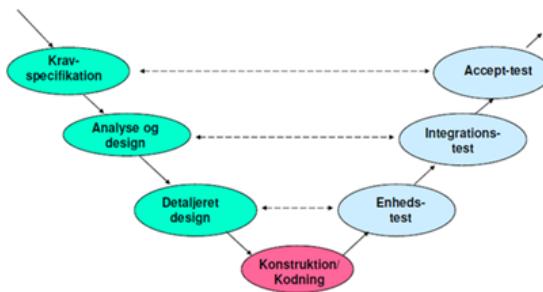
*Figur 9.1.* Sprint 2

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 9.2*



*Figur 9.2.* Forklaring af sprint points

### 9.2.2 V-model



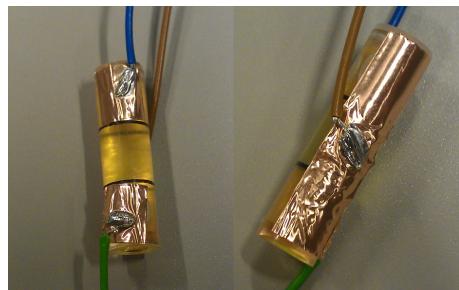
**Figur 9.3.** V modellen

Vi har valgt at anvende V modellen som udviklingsmodel. Dette muliggør iterative processer hvilket er optimalt for vores 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 var hældningsumåling. 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*. Hældningssensorprototype 1 indeholder 2 højpas filterer samt en frekvensgenerator og



**Figur 9.4.** Hældningssensor baseret på libelle

en differensforstærker. Vi kom frem til at libellesensoren har en kapacitet på omkring  $1 * 10^{-15}[F]$ . Det gør det praktisk umuligt at anvende da vores komponenter i det filter der skulle designes til den færdige sensor, da cutoff frekvens kommer til at være  $> 3.0[MHz]$ . 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 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 det komponentet opfyldte de krav stille i

<sup>1</sup>KXSC7-2050 se bilag

kravspecifikation. Dog er prototypen implementeret med en '#define' da databladet er vagt og output varierer mellem PSoCs.

### Hvordan mäter vi niveauet i vandballasttankene?

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 få træducere og receivere ligende 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 træducere og receivere for at se hvordan de aggerede. Vi fandt at der var en del destruktive reflektioner og vi overvejede at fyldes akustisk skum i toppen af tankene, men da det skum vi kunne finde var elektrisk ledende kunne det ikke anvendes. Herudover blev der lavet forsøg med forskellige ultralydkredse fundet på nettet, hvilket gav anledning til erfaring og viden. Jeg kom frem til at et burst skulle sendes for at måle afstanden og en puls på  $250\mu s$  blev valgt, hvilket svarer til 10 perioder. Receiverkredsen blev udviklet med viden fra MSE kurset fra sidste semester omkring mixere og operationsforstærkere.

#### 9.3.2 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 kabel kommunikation, skal der alligevel kabel ud til modulet, derfor kan det være lige så smart at have en kabel forsyning.

- Kablet forsyning

Ved at vælge at bruge en kabelt 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 sandsynlighed 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.

### 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 at skrive Qt-programmer til Raspberry Pi'en hvis man anvendte den nyeste beta-version af Qt.

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

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. Desværre løb gruppen 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 dataoverførelse har valget ligget imellem UDB eller TCP. UDB giver mulighed for flere tilslutninger men har ikke mulighed for pålidelig data overførelse. Ved at benytte TCP, kan der ikke være flere end en der kan koble til samtidig men derimod giver denne stabilitet og ordnet levering noget der blev prioriteret højst i prototypen da der kun var et skib(kontrolinterface). Dette vil sige at hvis en pakke går tabt under overførelsen vil denne automatisk blive forsøgt sendt igen og så er vi sikre på at data kommer frem i samme orden som de blev afsendt. der blev valgt at benytte TCP som gav den udfordring at ved afsendelse fra KI kom alle data over i en streng selv om vi forsøgte at sende dem over i 5 forskellige strenge og satte delays. Dette skyldes at TCP ser det som en streng der skal overføres og ikke som 5. For at løse dette satte vi nul terminering på ved afsendelse. Ved modtagelse blev dataet lagt ind i en variable som blev splittet ved nul terminaringen og efterfølgende gemt. Fra start var dte meningen at Serveren skulle gemme data direkte til MySQL databasen men grundet problemer med at kompile MySQL header og driver med blev dette efter noget tid droppet og i stedet blev backup muligheden der også er indskravet i kravspecifikationen ophøjet til 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.

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 fungere under Windows og da systemet skulle være alsidigt var dette ikke den bedste løsning. At lagre direkte i en tekst fil blev overvejet da det er

meget simpelt at lagre til tekstdokumenter 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 fungere på mange platforme og løbende bliver udviklet på. På baggrund af dette blev dette system valgt.

### 9.3.6 Webinterface

Til den grafiske brugere grænseflade for Databasen blev et Webinterface valgt på baggrund af muligheden for at flere på havneterminalen kunne tilgå dataerne på samme tid. På baggrund af et på forhånd middel kendskab til PHP som er et interpreteret sprog og dens integrationen med MySQL database blev PHP valgt som sprog til kodning af Webinterface. PHP giver desuden mulighed for at benytte AJAX principippet og at bygge en hoved side der inkluderer de andre sider. Dette giver mulighed for at let veligholdes samt minimal dataoverførsel fra webserveren som mindsker loading time.

### 9.3.7 Jura

I dette afsnit vil vi beskrive de vigtigste juridiske problem stillinger som vi har arbejdet ud fra.

Teknisk forskrift om lastning og losning af bulkskibe.

Bilag til foreskriften

Nummer:	Beskrivelse:
7	Bulkskibes og deres besætnings sikkerhed kan forbedres ved at mindske risikoen for, at de lastes eller losses uhensigtsmæssig ved terminaler for fast bulklast. Det kan gøres ved at etablere harmoniserede procedurer for samarbejde og kommunikation mellem skib og terminal og ved at stille egnethedskrav til skibe og terminaler.
9	Bulkskibe, der anløber terminaler med henblik på lastning eller losning af fast bulklast, bør være egnede til dette formål. Terminalerne bør ligeledes være egnede til at modtage og laste eller losse anløbne bulkskibe. Til disse formål er der fastsat egnethedsriterier i BLU-koden.
10	Terminalerne bør med det formål at forbedre samarbejde og kommunikation med skibsførerne om lastning og losning af fast bulklast udpege en terminalrepræsentant, der har ansvaret for lastning og losning i terminalen, og stille informationshæfter med terminalens og havnens krav til rådighed for skibsførerne, jf. bestemmelserne i BLU-koden.
12	For at sikre, at lastning og losning nøje bliver forberedt, aftalt og udført på en sådan måde, at skibets eller besætningens sikkerhed ikke kan bringes i fare, bør skibsførerens og terminalrepræsentantens ansvar fastlægges. Til dette formål findes der bestemmelser i den internationale konvention af 1974 om sikkerhed for menneskeliv på søen (SOLAS-konventionen af 1974), IMO-resolution A.862(20) og BLU-koden. Til samme formål kan der med udgangspunkt i disse internationale instrumenter ligeledes fastsættes procedurer for, hvordan lastning og losning forberedes, aftales og udføres.

## Artikel 2 Anvendelsesområde

Direktivet finder anvendelse på:

Nummer:	Beskrivelse:
1	alle bulkskibe, uanset flag, som anløber en terminal med henblik på lastning eller losning af fast bulklast, og
2	alle terminaler i medlemsstaterne, som anløbes af bulkskibe, der er omfattet af dette direktiv.

BILAG IV skibsførerens pligter før og under lastning og losning (som omhandlet i artikel 7, nr. 1, litra d))

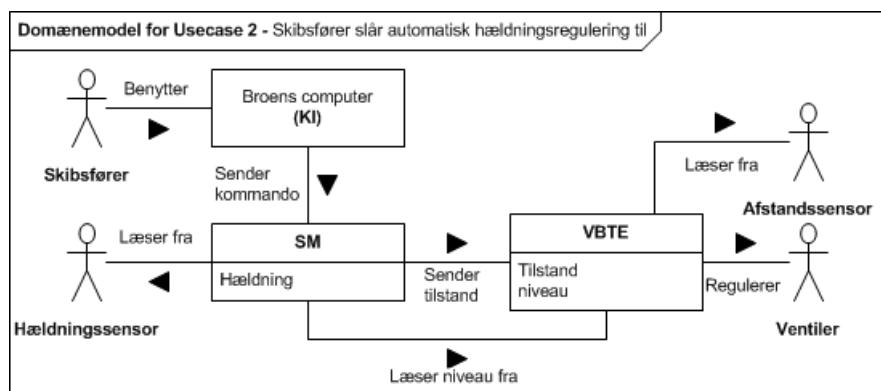
Før og under lastning og losning skal skibets fører sørge for følgende:

Nummer:	Beskrivelse:
1	Skibets ansvarshavende officer skal lede lastning og losning af ladningen og udømning og indtag af ballastvand.
2	Fordelingen af ladning og ballastvand skal overvåges under hele laste- eller losseprocessen for at sikre, at skibets konstruktion ikke overbelastes.
3	Skibet skal holdes på ret køl; kræves der af driftsmæssige årsager en vis slagside, skal den holdes så lille som mulig.
7	Terminalrepræsentanten skal gøres opmærksom på behovet for afpasning af deballastning eller ballastning og laste- og losserater for skibet og på enhver afvigelse fra deballastnings- eller ballastningsplanen og andre forhold, der kan have betydning for lastning eller losning af ladningen.

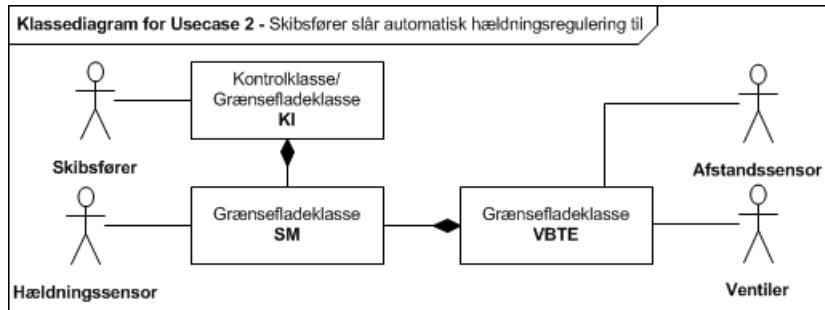
## 9.4 Systemarkitektur

I dette afsnit vil systemarkitekturen for projektet blive beskrevet. Systemarkitekturen tager udgangspunkt i dokumentationsdokumentet "Systemarkitektur", og for nærmere uddybning henvises der til det dokument. Systemarkitekturen er udarbejdet på grundlag af kravspecifikationen og systemet som beskrevet i projektbeskrivelsen.

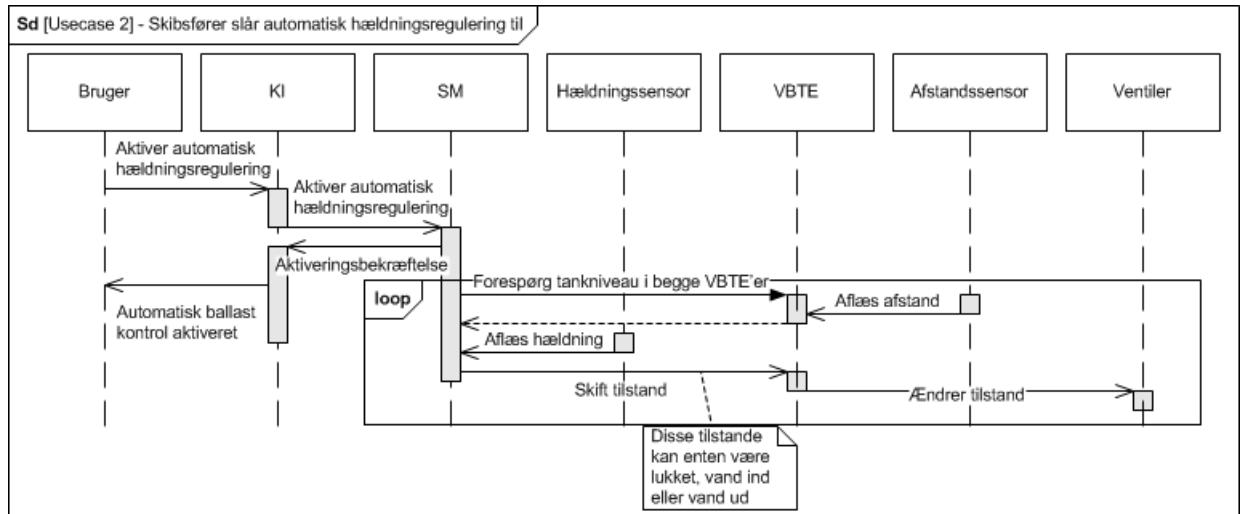
Systemarkitekturen starter med funktionaliteten beskrevet i Use Casene, og udvider så beskrivelsen til, at omfatte de elementer af systemet som brugeren ikke interagerer med. Dette afsnit vil give et eksempel på, hvordan en Use Case er blevet behandlet i systemarkitekturen. Use Casen der vil blive brugt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



*Figur 9.5.* Domænemodel for Use Case 2



Figur 9.6. Klassediagram for Use Case 2



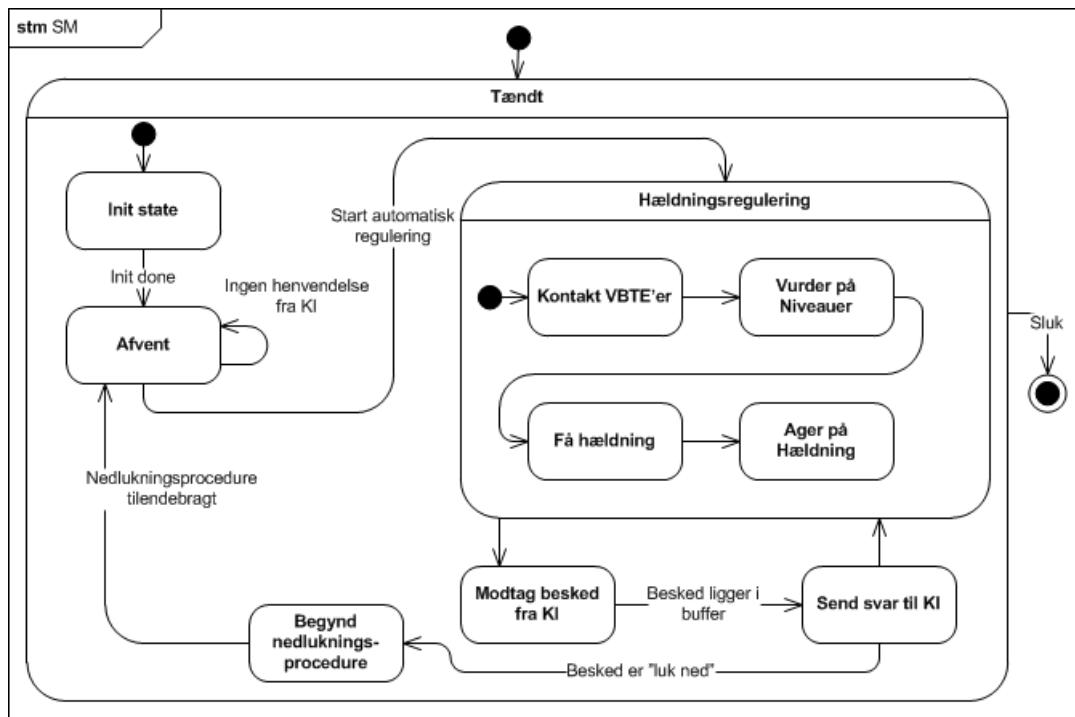
Figur 9.7. Sekvensdiagram for Use Case 2

### Relationen mellem elementer og aktører

I domænemodellen (figur 9.5) beskrives relationen mellem systemets elementer og aktørerne. Disse enheder omdannes i klassediagrammet (figur 9.6) 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.

### Kommunikation og timing

I sekvensdiagrammet (figur 9.7) 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 hældningssensorer. Vandniveauet vil blive ved med at blive reguleret, med et formål at opnå og derpå opretholde en hældning på nul grader.



**Figur 9.8.** State machine for Styringsmodulet

#### 9.4.1 Stadier for konceptuelle klasser

Når alle Use Cases er blevet bearbejdet som Use Case 2 er blevet i figur 9.5, 9.6 og 9.7 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 9.8.

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:

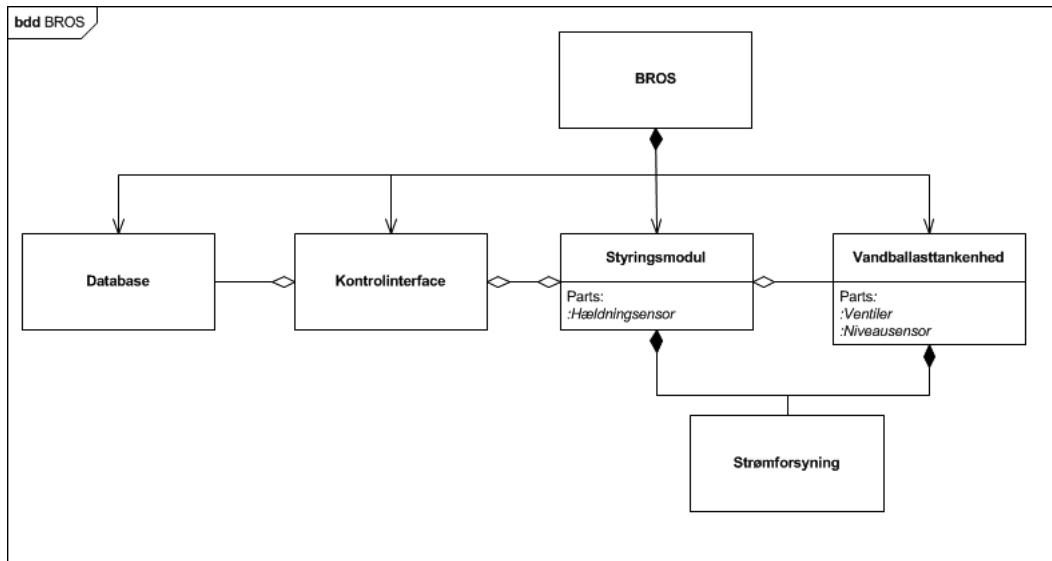
Hvis programmet er nyopstartet, vil beskeden få styringsmodulet ud af "Afvent-stadiet" og til at aktivere automatisk hældningsregulering. Hvis programmet befinner sig i "Hældningsregulerings-stadiet" med reguleringstypen *manuel*, vil beskeden fra KI få styringsmodulet over i stadiet "Modtaget besked 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.

#### 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 9.9. Her ses det, at systemet ender ud med fire hovedblokke: Kontrolinterfacet,

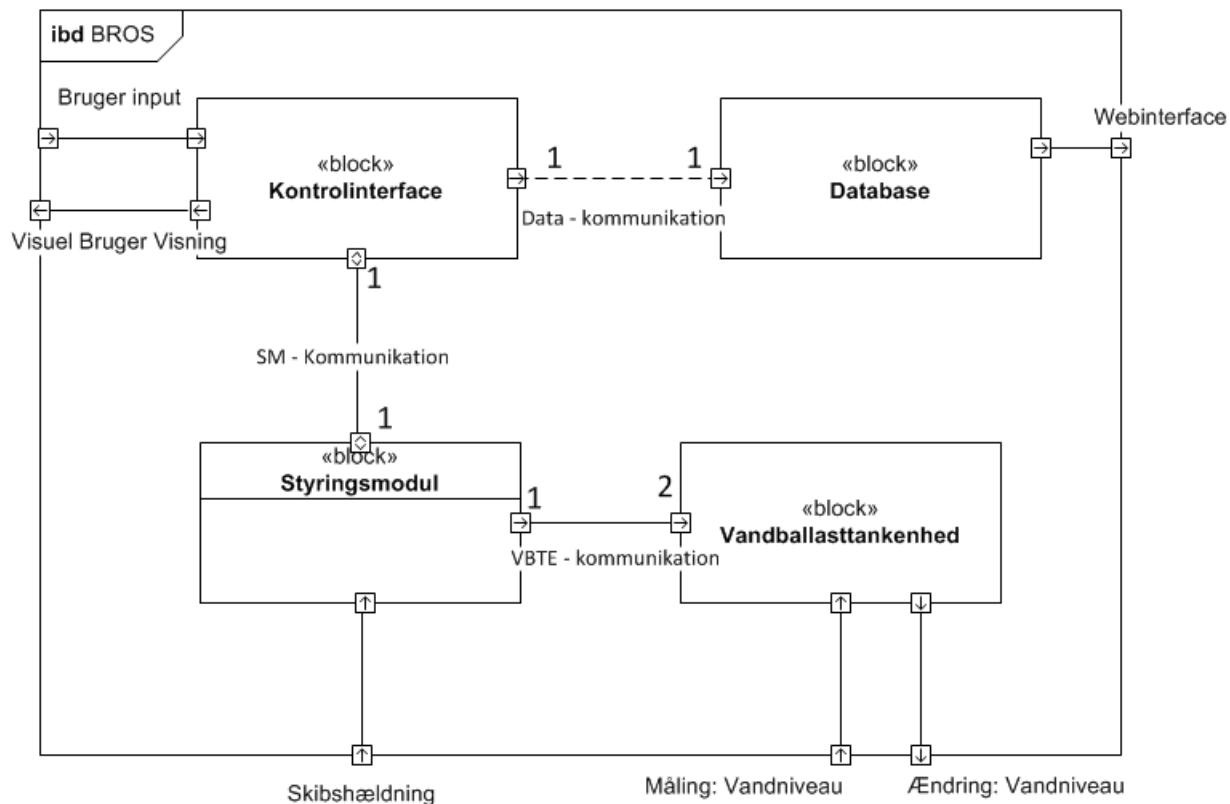
Styringsmodul, Vandballasttankenhed og Database. De konceptuelle klasser der ikke er endt som hovedblokke, er i stedet for blevet til underblokke af enten styringsmodulblokken eller vandballasttankenhedblokken.



**Figur 9.9.** 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 9.10. Her angives det også hvor mange gange de enkelte blokke skal optræde i systemet.



**Figur 9.10.** 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 VBTE og Styringsmodulet som opskrevet i tabel 9.4.1 med de indstillinger angivet 9.4.1.

#### Forbindelsesindstillinger

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

#### Protokoloversigt

Navn	Værdi	Beskrivelse
VBTNIVEAU	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.

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

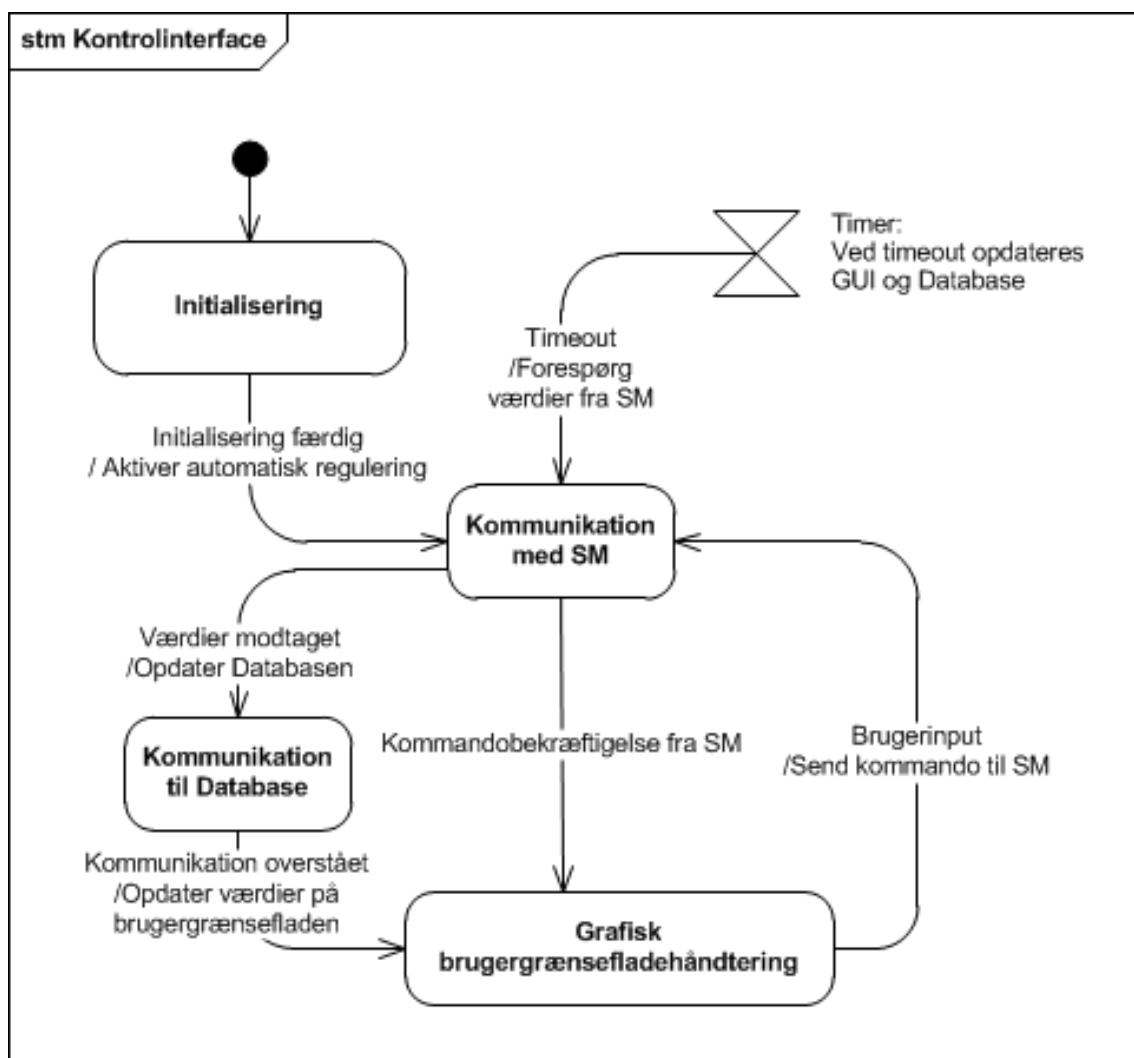
Det endelige produkt skal have strøm fra en strømforsyning<sup>2</sup>

## 9.5 Design og Implementering

### 9.5.1 Kontrolinterfacet

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

Opbygningen af Kontrolinterfacets design tager udgangspunkt i den state machine der er udarbejdet i systemarkitekturen, se figur 9.5.1



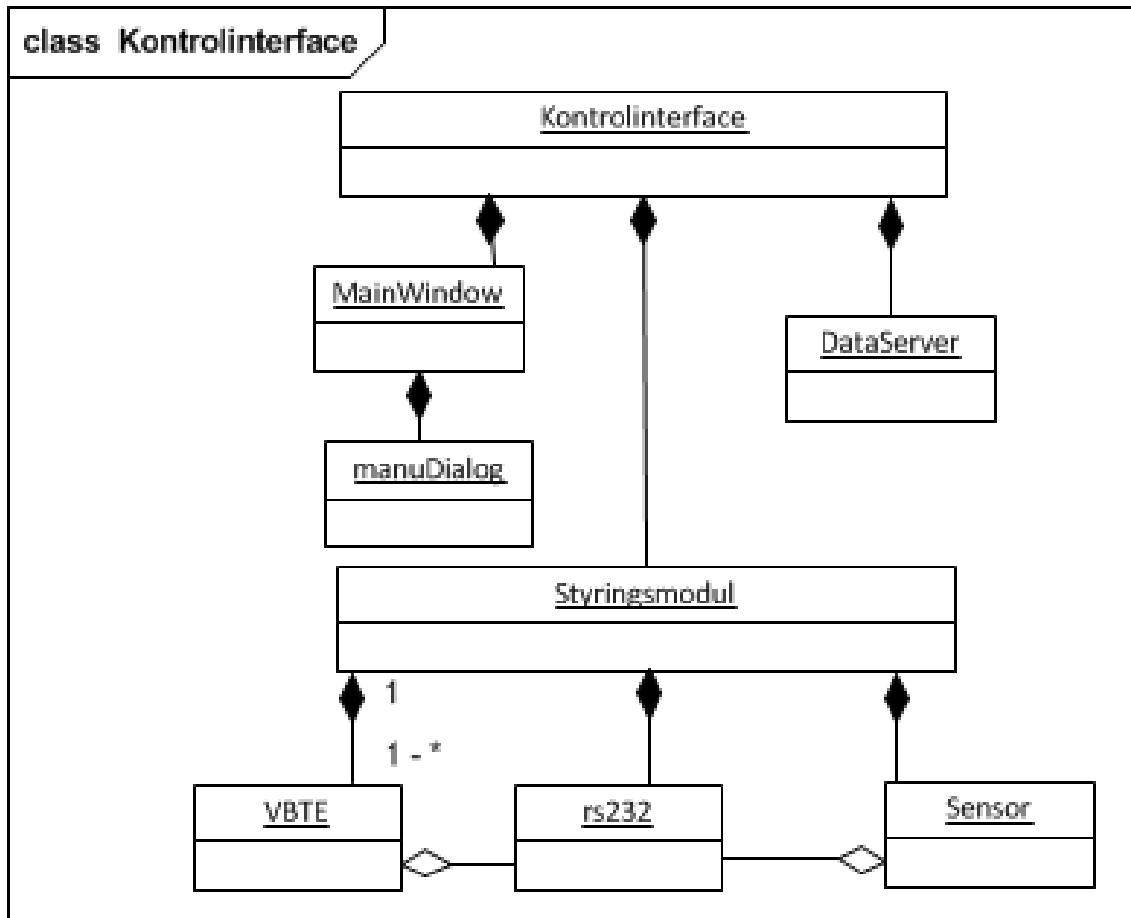
**Figur 9.11.** På figuren ses den simplificerede state machine for Kontrolinterfacet. En mere detaljeret kan ses i desindokumentationen for Kontrolinterfacet

State machinen på figur 9.5.1 er god til at få en forståelse for programmet, men for at

<sup>2</sup>Fixme Note:

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 Kontrolinterface-afsnittet i dokumentationsdokumentet for Detaljeret Software Design.

Det videre skridt er at udarbejde et klassediagram for programmet. Klassediagrammet er ligesom state machines først lavet i en simplificeret version og dernæst i en detaljeret. Det første kan ses på figur 9.5.1 med en beskrivelse af hver klasse i tabel 9.2.



**Figur 9.12.** På figuren ses det simple klassediagram for Kontrolinterfacet. Se en beskrivelse af hver klasse i tabel 9.2

## 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 9.2.** Kontrolinterfacets klasser

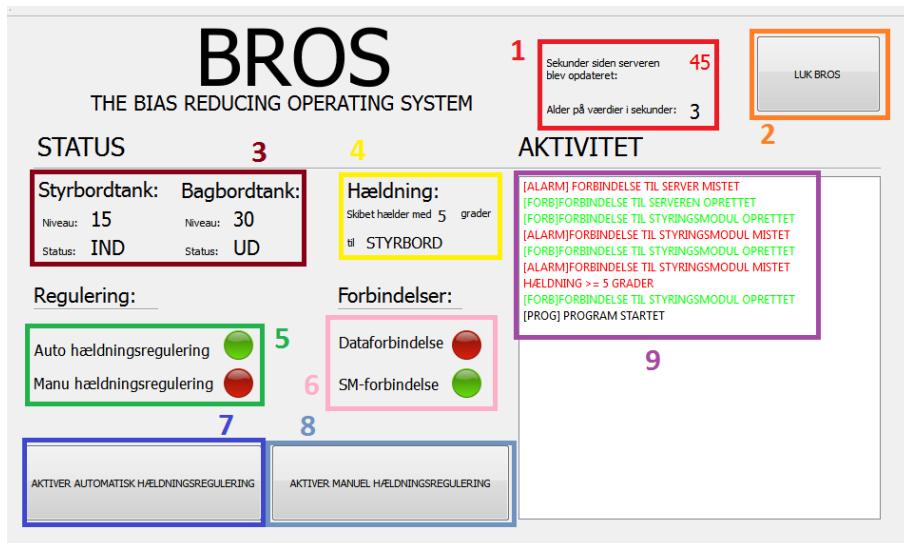
Klassediagrammet på figur 9.5.1 afspejler et program designet med forbillede i det samlede system. Dette er gjort for at trække på de løsninger der er fundet i forbindelse med det fulde system.

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 Kontrolinterface og Styringsmodul i det samlede system.

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.

## 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 9.5.1 er hovedvinduet vist. Her er vinduets elementer indrammet. En beskrivelse af hvert element vil kunne findes i tabel 9.5.1.



**Figur 9.13.** På figuren ses hovedvinduet for Kontrolinterface-programmet

## Hovedvinduets elementer

<b>1: Forsinkelse i sekunder</b>	Det øverste tal fortæller tiden i sekunder siden serveren sidst er blevet opdateret succesfuldt. Nedenunder udkrives 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 kan ses i Designdokumentet, Appendix A.
<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 en dialog hvor man beder bekræfte at man ønsker at gå væk fra manuel hældningsregulering. Hvis automatisk regulering allerede er aktiveret vil man få dette af vide i en dialog. Dialogerne kan ses i Designdokumentet, Appendix A.
<b>8: Manuel reguleringsknap</b>	Bringer dig til en dialog hvor hældningen vælges fra 0.5 grader til 2.5 grader. Herefter vælges siden skibet skal hælde til. Dialogen kan ses i Designdokumentet, Appendix A.
<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.

**9.5.2 VBTE**

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

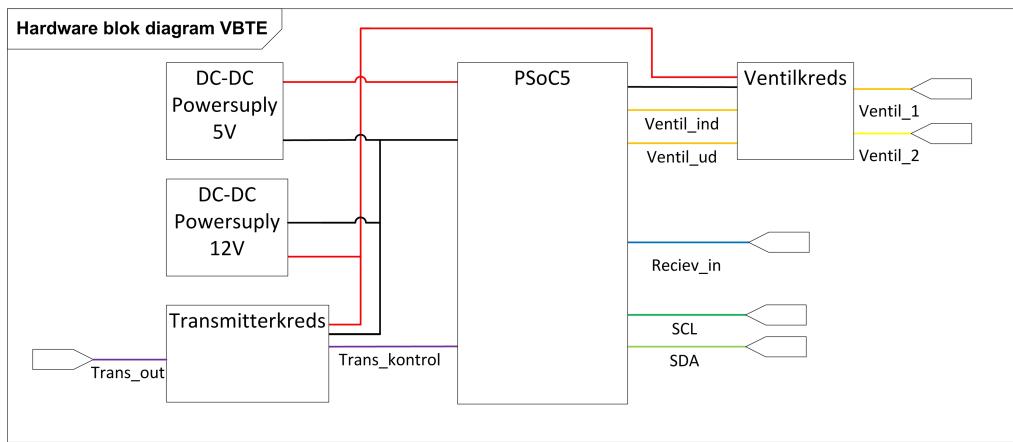
**Hardware**

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:

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

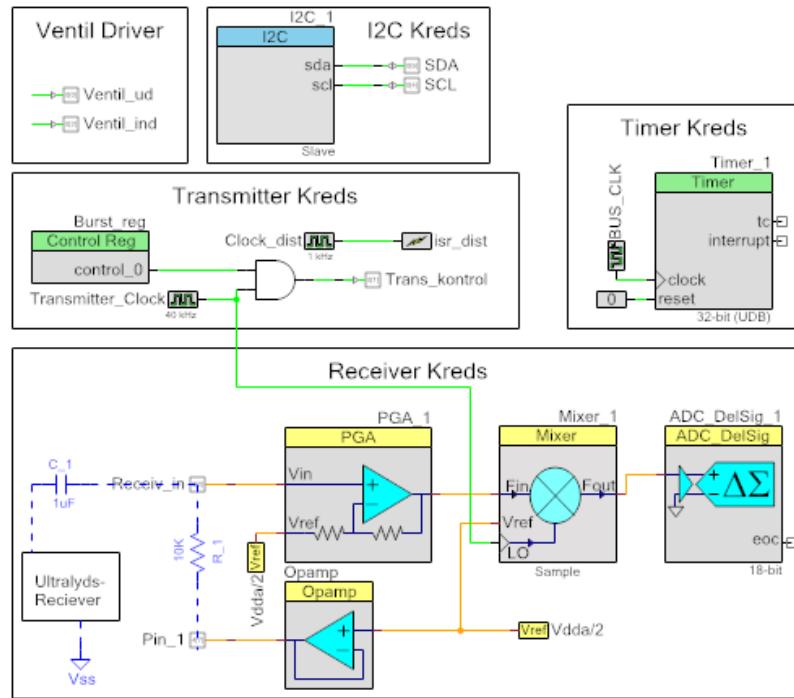
Gennem disse faser er designet blevet udfærdiget. Fremgangsmåden er anvendt for at overskueligtgøre systemet og lette arbejdet ved at dele systemet op i små dele. På *figur 9.14* ses det overordnede design af VBTE'en. Der vil i rapporten tages udgangspunkt i ventilkredsen samt receiverdriveren på PSoC'en.



**Figur 9.14.** 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 9.15 ses hardware designet på PSoC'en.



Figur 9.15. 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 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 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.

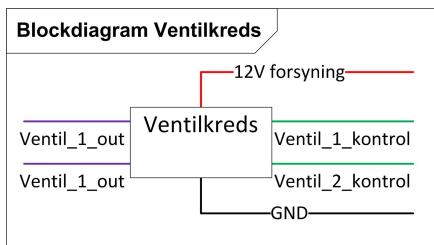
Efter signalet er løftet, bliver det forstærket op af en PGA. Den første forstærkning, fundet i teknologiundersøgelsen, viste sig at være for lav og forstærkningen blev justeret til 32. Problemet heri ligger at et meget klart signal vil kunne få systemet til at gå i mætning. Men dette ser ikke ud til at ske.

Herved bliver signalet mixet med 40kHz og filtreret. Efter mixeren giver det, stort set, en DC og summen af de to signaler. Filteret er indbygget 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

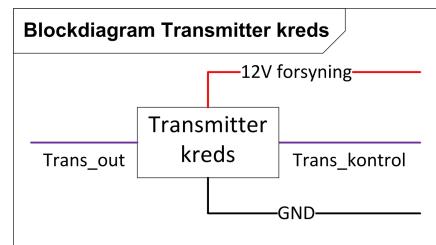
over regnes der en opladningstid på filteret til 1/4 ms. Efter undersøgelser af signalet ind på ADC'en blev en spænding på 0,3 V valgt til at representere en detektion.

### 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 9.16.** Hardwareblok for ventil



**Figur 9.17.** 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>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 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.

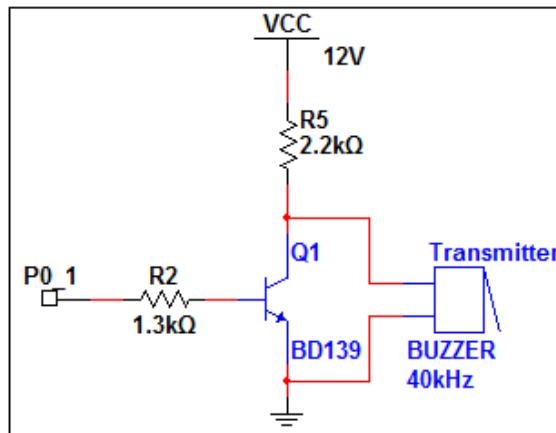
### Transmitterkreds

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

Via. kontrollinsegnal 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 omtrentlig 10 kΩ. Derfor kan PSoC'en uden problemer selv trække denne kreds.

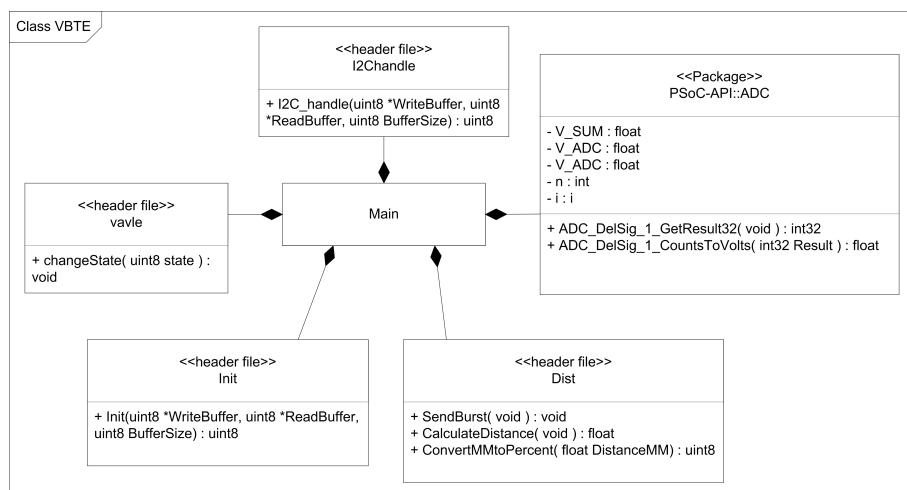
<sup>3</sup>Se bilag/BD139.pdf

**Figur 9.18.** Transmitterkreds

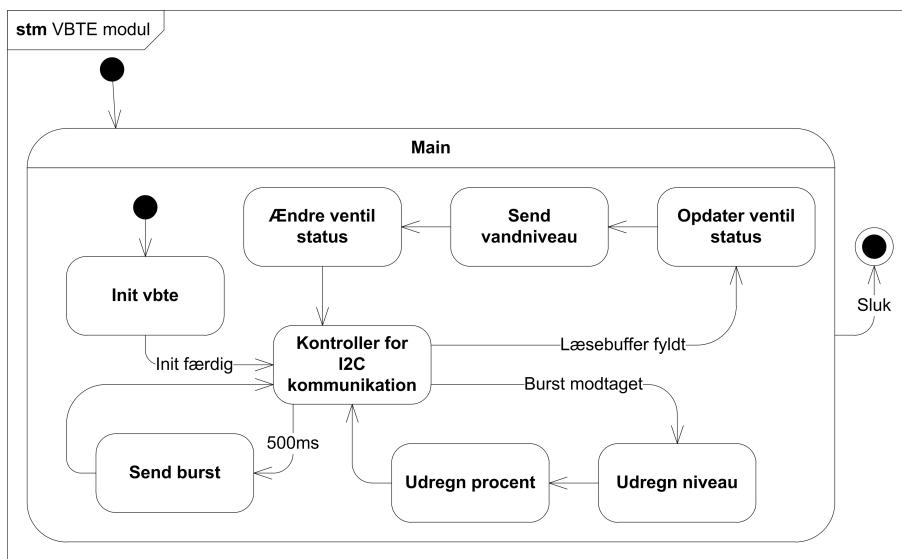
For yderligere detaljer vedrørende hardware på VBTE henvises til *Deltajeret hardware design*.

### 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 9.5.2 ses klassediagrammet for VBTE modulets software.

**Figur 9.19.** 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 detaljert software design. For at overskueliggøre funktionaliteten af softwaren anvendes statemachinediagrammet. Dette viser de forskellige tilstande programmet til enhver tid kan befinde sig i.



**Figur 9.20.** 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'ens interruptroutine 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 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.

### 9.5.3 SM

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

#### Hardware

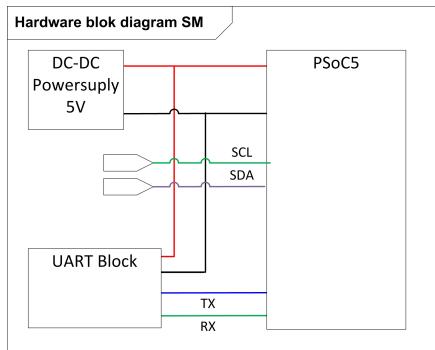
SM modulets hardware består af en konverteringskreds og en PSoC. På PSoC'en er monteret et Kionix KXSC7-2050 accelerometer. Konverteringskredsen anvendes til at sende UART signaler fra PSoC til en KI modulet. Accelerometerets x-akse anvendes til hældningsmålinger for hældningssensorblokken.

Designfasen til SM er delt op i 3 faser:

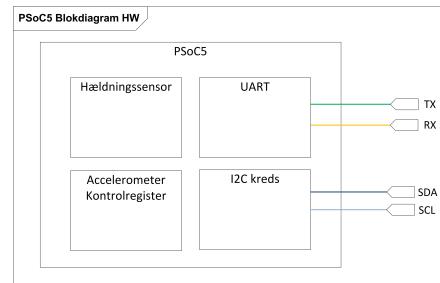
1. Overordnet design
2. Nedbrydning af blokke
3. Opbygning af design

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 9.21* ses det Overordnede design og på *Figur 9.22* ses PSoC blokken i SM. Der bliver efterfølgende taget udgangspunkt i Hældningssensoren.

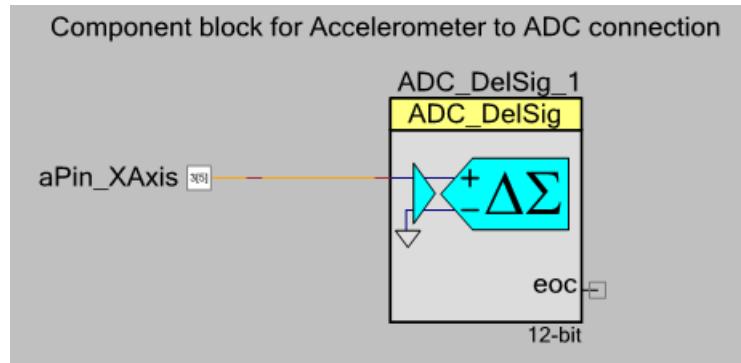


**Figur 9.21.** Hardware blok for SM



**Figur 9.22.** PSoC blok for sm

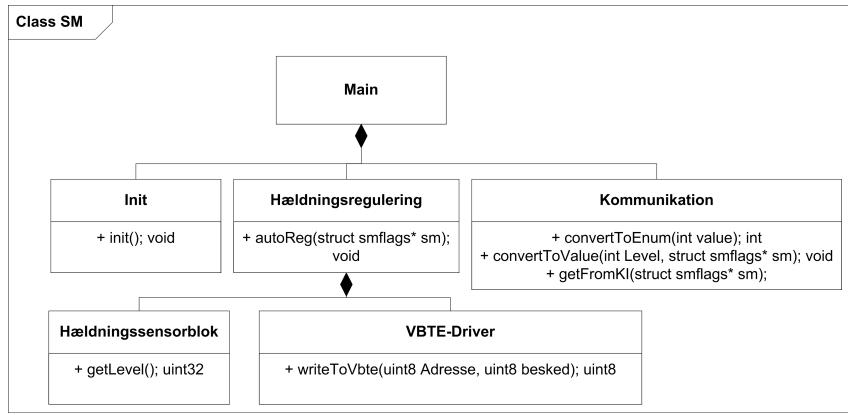
Hældningssensoren består af 2 komponenter: det førnævnte accelerometer samt en DelSig ADC internt i PSoC'en. Valget af accelerometer kommer af at have lavet en række prototyper der ikke mødte vores krav, hvilket accelerometeret i PSoC'en gjorde. Valget af ADC faldt på en DelSig da, den er meget støj immun grundet det indbyggede lavpas filter og har en høj oplosning. Valgte komponenter er illustret på *figur 9.23*. 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 12.1.5 *Detaljeret hardware design* i Bilag.



**Figur 9.23.** Hældningssensorens implementering

## Software

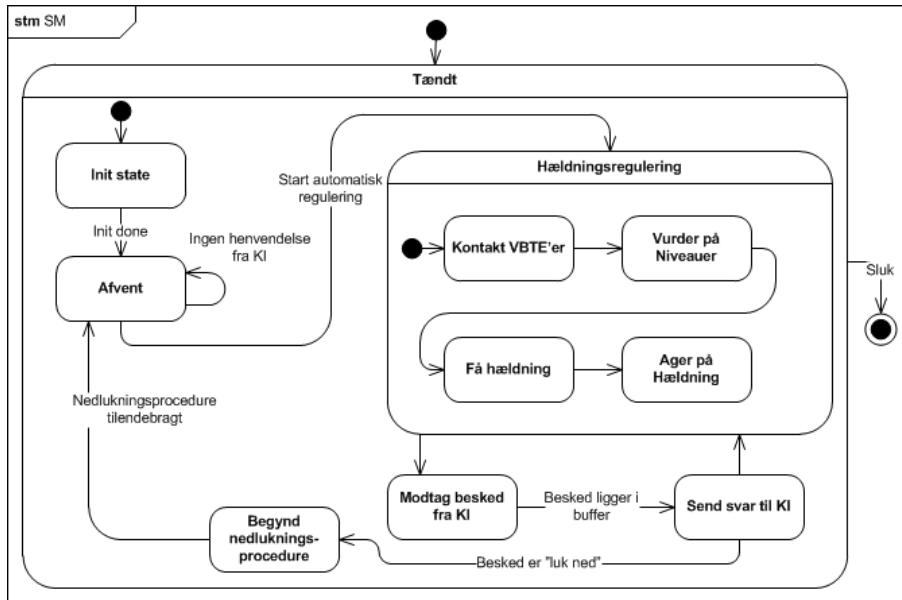
SM software behandler den konverterede data fra ADC'en samt kommunikation med VBTE og KI modulernerne. Sofwarens udvikling har fulgt de samme faser som hardwaren, hvilket har ført til letforståelig og læselig kode. Softwaren er illustreret på *figur 9.24*. Der vil efterfølgende blive taget udgangspunkt i et statemachine samt funktionen getFromKI.



Figur 9.24. Klassediagram for SM

### SM funktionalitet

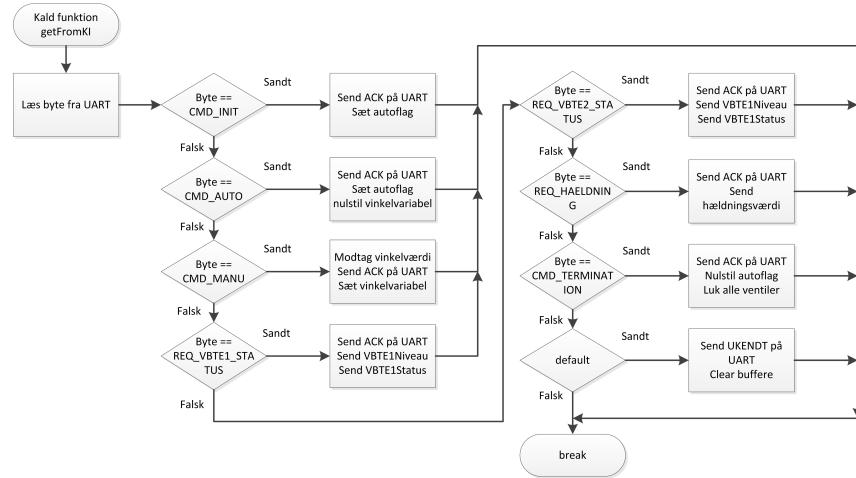
Funktionalitet af SM kan bedst beskrives med et state diagram. På figur 9.25 ses state machine diagrammet for SM modulet. Diagrammet indeholder hældningsregulering state der beskriver den automatiske og manuelle regulering i systemet. Den automatiske regulering styres med et flag. Den manuelle styres 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 9.25. State machine for SM

### getFromKI

Funktionen er bedst beskrevet med et flowdiagram:



**Figur 9.26.** 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 sikre 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.

#### 9.5.4 Strømforsyning

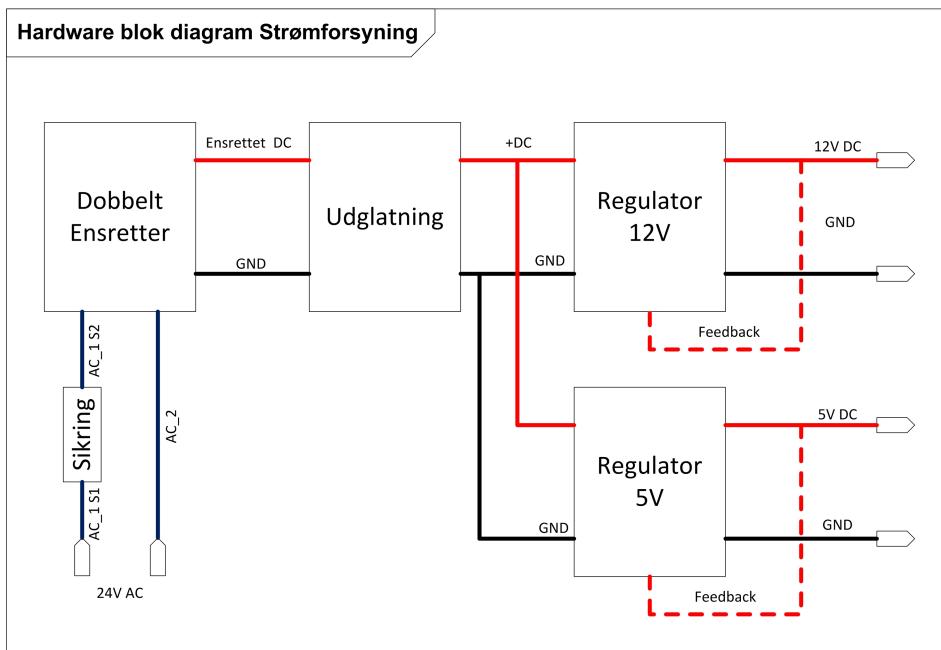
I dette afsnit beskrives design og implementering af strømforsyning. Strømforsyningen implementeres sammen med SM og VBTE.

##### Hardware

Strømforsyning er designet til at levele to forsyningsspændinger: **12VDC 1A og 5VDC 0.5A** dette gøres fra en 24V AC forsyningsskilde. For at gøre designet af strømforsyning mere overskuelig er designfasen delt op i 3 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, hvor efter hver enkel blok er blevet beskrevet og designet. *figur 9.27* viser det overordnet blokdiagram af strømforsyningen.



**Figur 9.27.** Illustrering af overordnet blokdiagram af strømforsyningen.

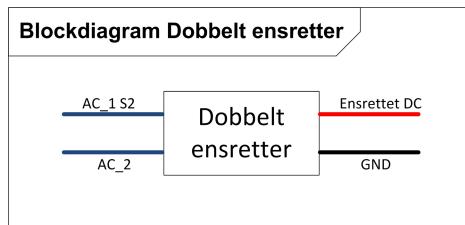
### Beskrivelse af vitale hardware blokke

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

#### Dobbelt ensretter og udglatning

Da strømforsyningen for leveret strøm fra en AC spændingskilde, skal AC signalet ensrettes til DC, dette gøres med dobbelt ensretteren og udglatningsblokken.

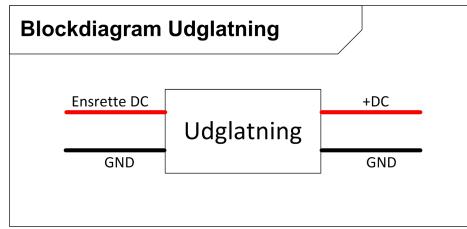
#### Dobbelt ensretter



**Figur 9.28.** Blok for dobbeltensretter

Dobbeltensretteren er lavet som en diodebro bestående af 4 dioder, hvor to af dioderne samtidig, hvilket betyder at der vil være positive halvperioder på udgangen "ensrettet DC" med en spidsværdi på ca. indgangsspænding \*  $\sqrt{2}$

## Udglatning



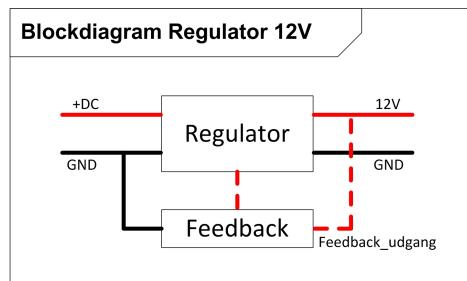
**Figur 9.29.** Blok for dobbelt ensretter

Udglatningen er en elektrolyt kondensator som oplades og aflades i forhold til de positive halvperioder, da kondensator 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 spænding fra Udglatningen reguleres ned til det pasende niveau, dette gøres i blokkene regulator 12V og regulator 5V.

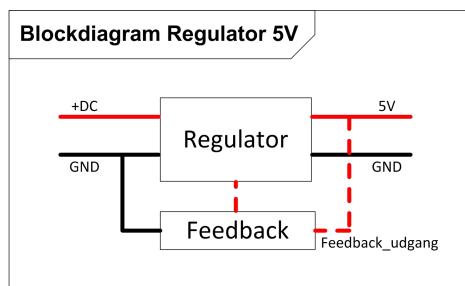
### 12V regulator



**Figur 9.30.** Blok for 12V regulator

For at regulatoren kan regulere spændingen ned til 12V, skal der bruges et feedback fra udgangen "12V", feedbacket er indstillet til 12V ved hjælp af to modstande.

## 5V regulator



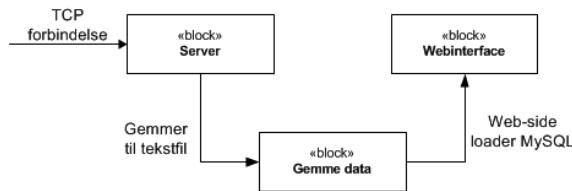
*Figur 9.31.* Blok for 5V regulator

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

### 9.5.5 Database

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

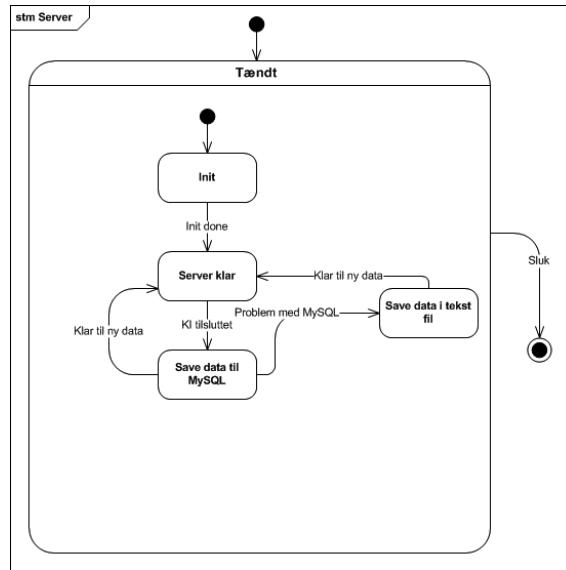
Databasen er blevet opdelt i 2 dele som håndtere denne side. Databasen er opdelt i server og Webinterface.



*Figur 9.32.* Illustrere overordnet hvordan databasen fungere

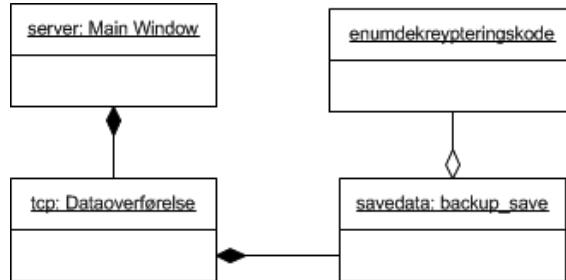
### 9.5.6 Serveren

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 9.34*. For forståelse af det program flow der sker på serveren er der udarbejdet en statemachine 9.33 der hviser dette flow



**Figur 9.33.** Illustrere overordnet hvordan databasen fungere

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 9.34.** 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 9.3.** 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 9.35.** 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 i stedet for at få udskrifter i terminalen.

### Hovedvinduets elementer

#### 1: Info

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 9.35

#### 2: Database

Anvendes til at lade brugeren tilgå den web baserede database som ses i ??

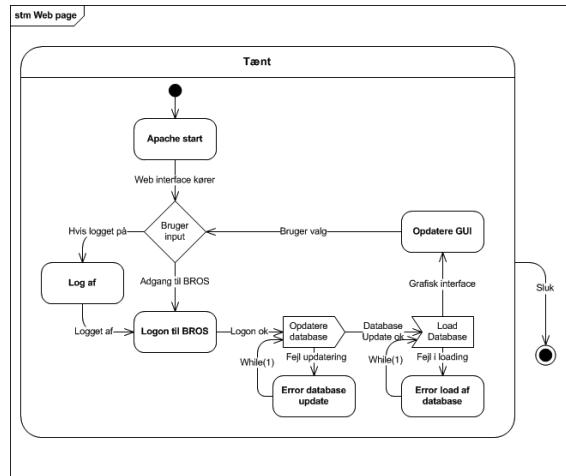
#### 2: Luk serveren

Anvendes til at lukke programmet. Bruges til nedlukning

### 9.5.7 Webinterface

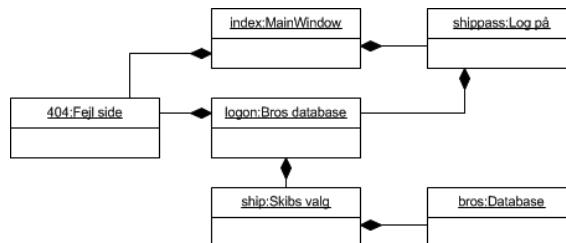
Denne section beskriver design og implementering af web siden.

Opbygningen af Webinterface er gjort i php hvor dette gav mening. Desuden er html og styles (css) blevet benyttet til at sikre ens stil på hele web siden og for simpelt at kunne veligeholdes. 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 statemaschine 9.36



**Figur 9.36.** 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 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>4</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. Til at håndtere wibinterfacet benyttes en apache server som er installeret på serveren. Apache serveren kan benyttes med de fleste operativ systemer.



**Figur 9.37.** Illustrere overordnet hvordan web-siden fungere

<sup>4</sup>Fixme Note: link til appendix

## 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 (login)	Denne side bliver kaldt efter at brugeren er logget på. Her vælger brugeren hvilket skib denne ønsker at få vidst data for.

**Tabel 9.4.** 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 appendix

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 9.38.** 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 IP 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.

## 9.5.8 MySQL

Som database benyttes MySQL. Denne er en multitreading database som giver mulighed for at flere kan tilgå denne samtid. Det er et færdigt produkt der kan implementeres på de

fleste styresystemer. Systemet giver mulighed for at oprette interne databaser med egne taballer til algring af data (se appendix 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 som normalt oprette en fil med dennes data. Når et website med dette implementeret loader filen kan denne se på om dette skib allerede findes, gør det ikke 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 hhv. "Enhedstest", "Integrationstest" og "Accepttest".

### 9.6.1 KI

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. På trods af at der er flere steder hvor der godt kunne ske forbedringer (se afsnit <sup>5</sup>, så er slutresultatet, med tanke på arbejdsbyrden set i forhold til at opgaven har været en enkeltmandsopgave, et tilfredsstillende resultat.

### 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 KI via TCP. Serveren er i stand til at modtage data fra KI. Der benyttes en dekryptering til at omsætte hældningen fra SM til grader og gemme denne som en tekstfil. Lagring direkte til MySQL blev aldrig implementeret grundet problemer med MySQL driver. Selve MySQL databasen er fuldt kørende og fungere fejlfrit. Webinterfacet er fuldt implementeret og ved opstart af webinterfacet fungere bruger adgangskoden næsten korrekt. Der opstår et problem ved fejl password hvor brugeren får en hvid side i stedet for igen at blive bedt om nyt password men får dog ikke adgang til data. En mulig løsning er fundet men grundet tidspres ikke implementeret. Ved korrekt adgangskode får brugeren mulighed for at vælge skib og få dennes data frem. Ved inndlæsning af data siden bliver tekstuften som serveren genererer hentet. Data bliver taget ud fra den og gemt i MySQL databasen og filen slettet. Alle data bliver hentet fra MySQL databasen med de nyeste øverst. Siden checker hvert 5 sekund om der er ny data. Hvis der ikke er ny data foretages der intet.

---

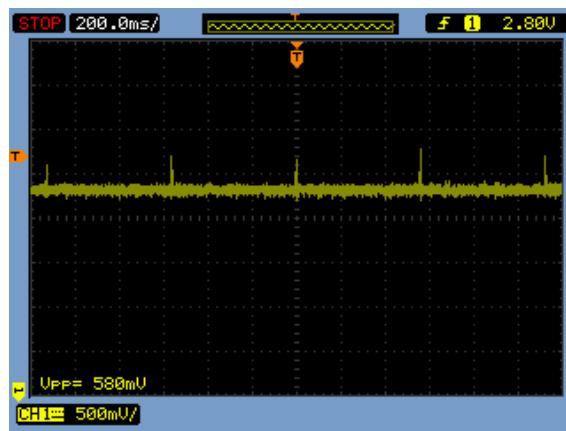
<sup>5</sup>Fixme Note: indsæt reference til forbedringer

### 9.6.3 SM

SM modulet er fuldt implementeret, men ikke finjusteret. Forbindelsen til KI er fuldt funktionel. Forbindelsen til VBTE virker 95% af tiden. Automatisk hældningsregulering er implementeret, men da niveaumålinger ikke modtages fra VBTE modulet er det ikke muligt at tømme tankene inden vi fylder i den anden tank. Prototypen er 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 VBTE

VBTE modulet nåede aldrig at komme til at fungere optimalt. Efter at have implementeret ultralydsafstandsensoren fandt jeg ud af at det ikke fungerede ret godt over særligt lange afstande (i hvert fald i mit design). Men under kontrolerede forhold i laboratoriet lykkedes det at få flotte detektioner på ultralydsafstandssensoren som kan ses på *figur 9.6.4*.



*Figur 9.39.* Detektioner af ultralydsafstandssensoren med faste intervaller.

Derudover virkede I2C delen, dog med fejl på data i nogle 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

Strømforsyning er fuldt implementeret, i testen er der blevet brugt effekt modstande som lod, en labitoriestrømforsyning 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 sjælende vil ske, ses det ikke som et problem. Strømforsyningen er implementeret med SM og VBTE. 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 test's er gået helt nede fra enhedstest til accepttest. For detaljer om de 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 9.5.** Testgodkendelseskategorier

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

**Tabel 9.6.** 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.

Integration af systemets enheder er blevet godkendt.

Grundet problemer med afstandsmåling på VBTE kan Accepttest case 2 og 3 blive mere end delvist godkendt.

## 9.7 Opnåede erfaringer

### 9.7.1 Gruppen

Det har været rigtig interessant at skulle samarbejde fem personer fra fire forskellige sammenstømrede 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 hver enkelt moduls opgaver blev udført.

### 9.7.2 Agile udviklingsmetoder

Gruppen har gennem dette projekt forløb fået en bedre forståelse for iterationer og hvordan dokumentation holdes opdateret efter behov. Endvidere har gruppen erfaret vigtigheden i omstilling og muligheden for arbejde iterativt.

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 gennemtestede 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 eksperimentet med at anvende latex til at skrive rapport og dokumentation.

## 9.8 Udviklingsværktøjer

I projektet er der værktøjer både i forbindelse med hardware og software design samt dokumentation og rapport.

- Multisim 11.0
- PSoC Creator 2.1

- Qt creator 4.8.1 og 5-beta
- Microsoft Visio
- Notepad++
- Mathcad
- Google Docs
- Google Code
- Texmaker
- TortoiseSVN 1.7.10
- Dropbox
- TextWrangler



# Konklusion 10

---

## 10.0.1 Bulletpoints til Konklusion

- Delvist godkendt acceptest
- Vellykket dokumentation
- Manglende implementering (som funktion af iterationer og opdatering af arkitektur)
- vellykket gruppearbejde

<sup>1</sup> Det er desværre ikke lykkedes for gruppen at implementere hele systemet, som det var ønsket. Grundsystemets krav, opgivet i kapitel 8, 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 løsningerne i systemet - det gælder især for hældningssensoren og vandballasttankenes 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 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 tre punkter "Sikkerhed", "Optimering af overførsel af data" og "Effektivitet".

- 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 SM hvis der bliver svaret med forkert data til KI. Dette kan optimeres ved at forsøge flere gange hvis det rigtige svar ikke kommer tilbage.

Derudover er der ikke nogen tjek af data mellem SM og VBTE. 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 SM. Den anden, og mere interessante, kunne være at helt udelade SM og have en hældningsmåler på hver VBTE. VBTE vil derfor reagere selvstændigt på hældningen resulterende i at der både bliver tömt og fyldt på samme tid i hver sin side. Problemet med dette bliver så kontakt til KI og der skal laves væsentlige ændringer i hele struktureringen af systemet.

Til effektivitet kan også nævnes at manuel vinkling også manuelt skal sættes tilbage igen. Dette kunne håndteres ved hjælp af SM. 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 SM aktivere automatik igen når den når vatter. Desuden kunne et check på om Databasen har modtaget data fra KI sikre at Havneterminalen opdager en mulig fejl.

# 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 bilag. (Kravspecifikation.pdf)

### 12.1.2 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).

### 12.1.3 Systemarkitektur

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

### 12.1.4 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).

### 12.1.5 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).

### 12.1.6 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).

## 12.2 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  
https://www.retsinformation.dk/Forms/R0710.aspx?id=26449  
http://www.apache.org/  
http://www.mysql.com/

## 12.3 Liste over bilag på CD

Komponentliste.pdf  
SCRUM.xls  
Logbog.pdf

### 12.3.1 Kode

#### KI

hest

#### SM

hestning

#### VBTE

honning

#### Databasen

### 12.3.2 Dokumentation

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

### 12.3.3 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

### 12.3.4 Billeder

hvis vi har billeder af vores produkt!