Aarhus School of Engineering

Electronic Engineering

E4PRJ

Detaljeret Software Design

Author:
Nicolai Glud
Johnny Kristensen
Rasmus Lund-Jensen
Mick Holmark
Jacob Roesen

19. december 2012

Indholdsfortegnelse

Indledning 1

Dette dokument beskriver det detaljerede SW-design for BROS, som er fastlagt ud fra dokumenterne kravspecifikation og systemarkitektur.

1.1 Formål

Formålet med dokumentet er:

- At fastlægge systemets detaljerede softwarestruktur udfra kravene specificeret i kravsspecifikationen. Derudover beskrivelsen af softwarekomponenterne og deres grænseflader beskrevet i systemarkitektur-dokumentet.
- At fastlægge systemets softwareklasser og deres indbyrdes interaktioner.
- At beskrive de enkelte klassers vigtigste metoder.

1.2 Reference dokumentation

- Kravspecifikation for projektet.
- Systemarkitektur-dokument.

Kontrolinterface 2

Nedenfor følger design af software til Kontrolinterfacet. Dette er lavet på baggrund af kravspecifikation og systemarkitektur.

2.0.1 Modulets ansvar

Kontrolinterfacet er brugerens primære kontaktflade til systemet. Programmet indeholder en brugergrænseflade der opfylder kravene i Kravspecifikationen. Her kan der også ses en prototype på den grafiske brugergrænseflade. Kontrolinterfacet står for at modtage inputs fra brugeren. Disse inputs sendes som kommandoer til Styringsmodulet. Det er også herfra at Kontrolinterfacet modtager de værdier, som sidenhen vises på den grafiske brugergrænseflade. Kontrolinterfacet står også for kommunikationen til den eksterne database. Her sendes en række parametre om skibet og dets status.

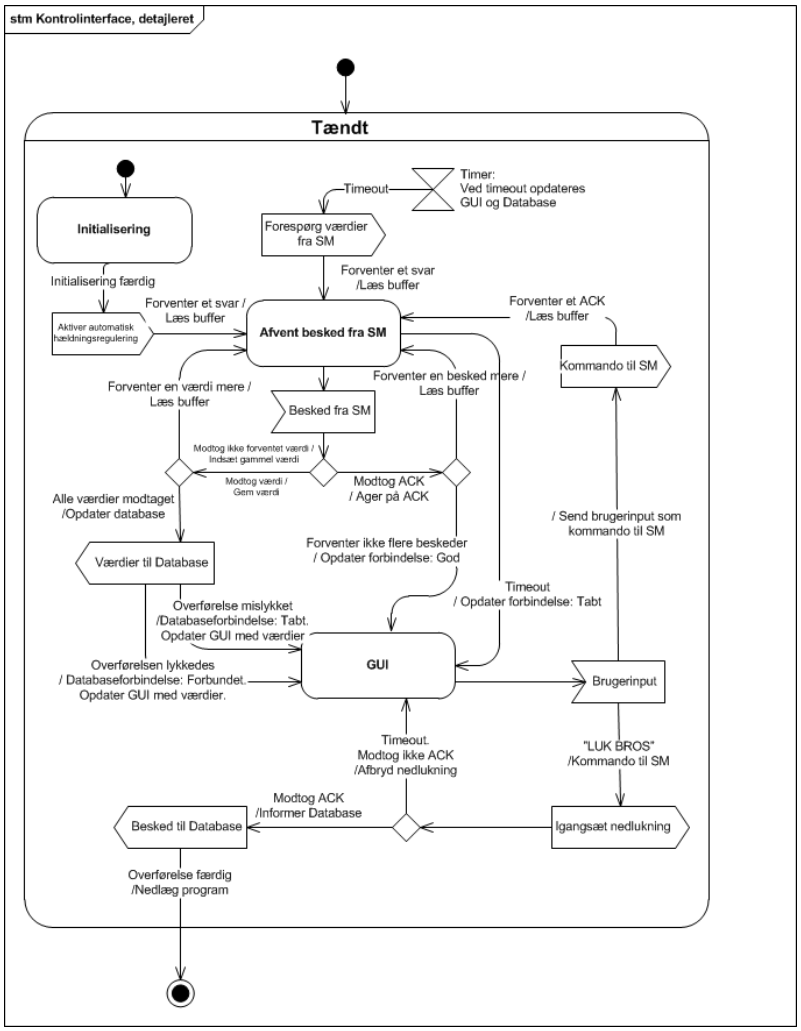
2.0.2 State machine diagram

Designet af Kontrolinterfacet tager udgangspunkt i arkitekturen fremstillet i artefaktet Systemarkitektur. Her blev der fremstillet et state machine for Kontrolinterfacet. State machinet var simplificeret og er i designfasen blevet udvidet. Resultatet ses i figur 2.0.2.
Figuren beskriver hvilke stadier programmet kan befinde sig i og sammenhængen imellem dem.
Størstedelen af kontrolinterfacets levetid foregår i et ventestadie hvor der afventes enten et brugerinput eller et timeout. Begge scenarier afføder et behov for kommunikation med styringsmodulet.
Hvis der har været et timeout skal der hentes værdier fra Styringsmodulet hvorefter Databasen opdateres.
Hvis der har været et brugerinput skal Kontrolinterfacet blot have en bekræftigelse fra Styringsmodulet på at kommandoen er modtaget.

2.0.3 Klassediagram

Således er det videre skridt at udarbejde et klassediagram for programmet. Her anvendes klassediagrammer. Klassediagrammet er ligesom state machines først lavet i en simplificeret version og dernæst i en detaljeret. Den første kan ses på figur 2.0.3.

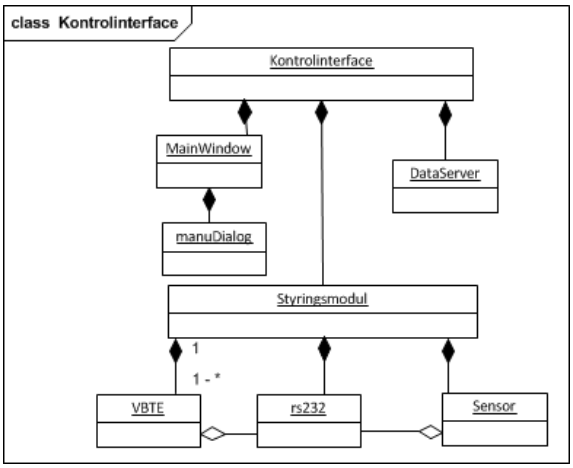
Klassediagrammet på figur 2.0.3 afspejler et system designet med forbillede i det samlede system. Dette er gjort for at trække på de løsninger der er fundet i forbindelse med det fulde system. Her ses det hvordan hovedklassen er Kontrolinterfacet. Alt andet oprettes enten direkte af Kontrolinterface-klassen eller af en klasse oprettet af samme. Kontrolinterfacet



Figur 2.1. På figuren ses det detaljerede state machine diagram for 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.

En beskrivelse af hver klasse kan findes i tabel 2.1.



Figur 2.2. På figuren ses det simple klassediagram for Kontrolinterfacet. Se en beskrivelse af hver klasse i tabel 2.1

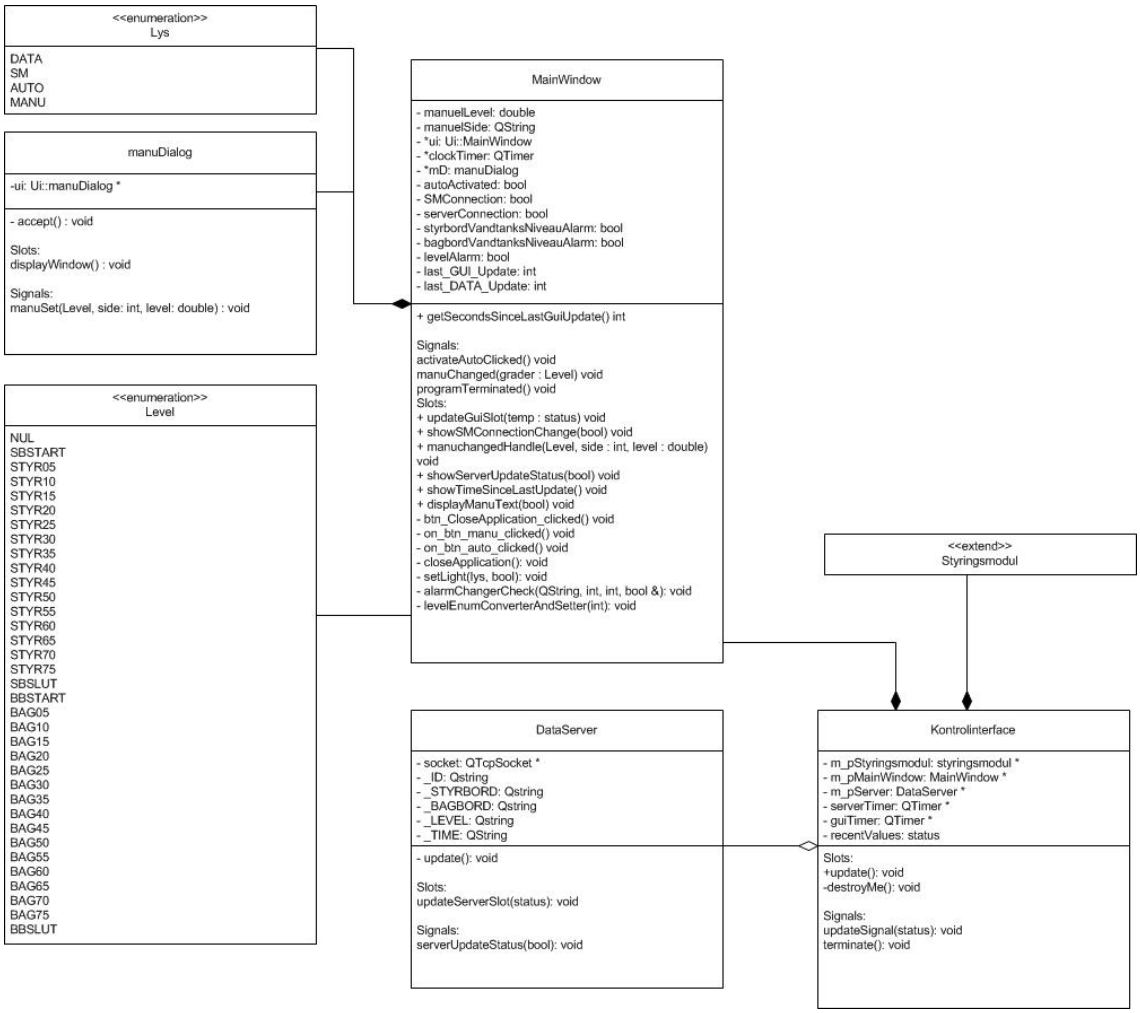
Kontrolinterfacets klasser

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 brugergrænsefalde.
manudialog:	Oprettes af MainWindow og styrer den dialog, der fremkommer når man ønsker en manuel hældningsregulering.

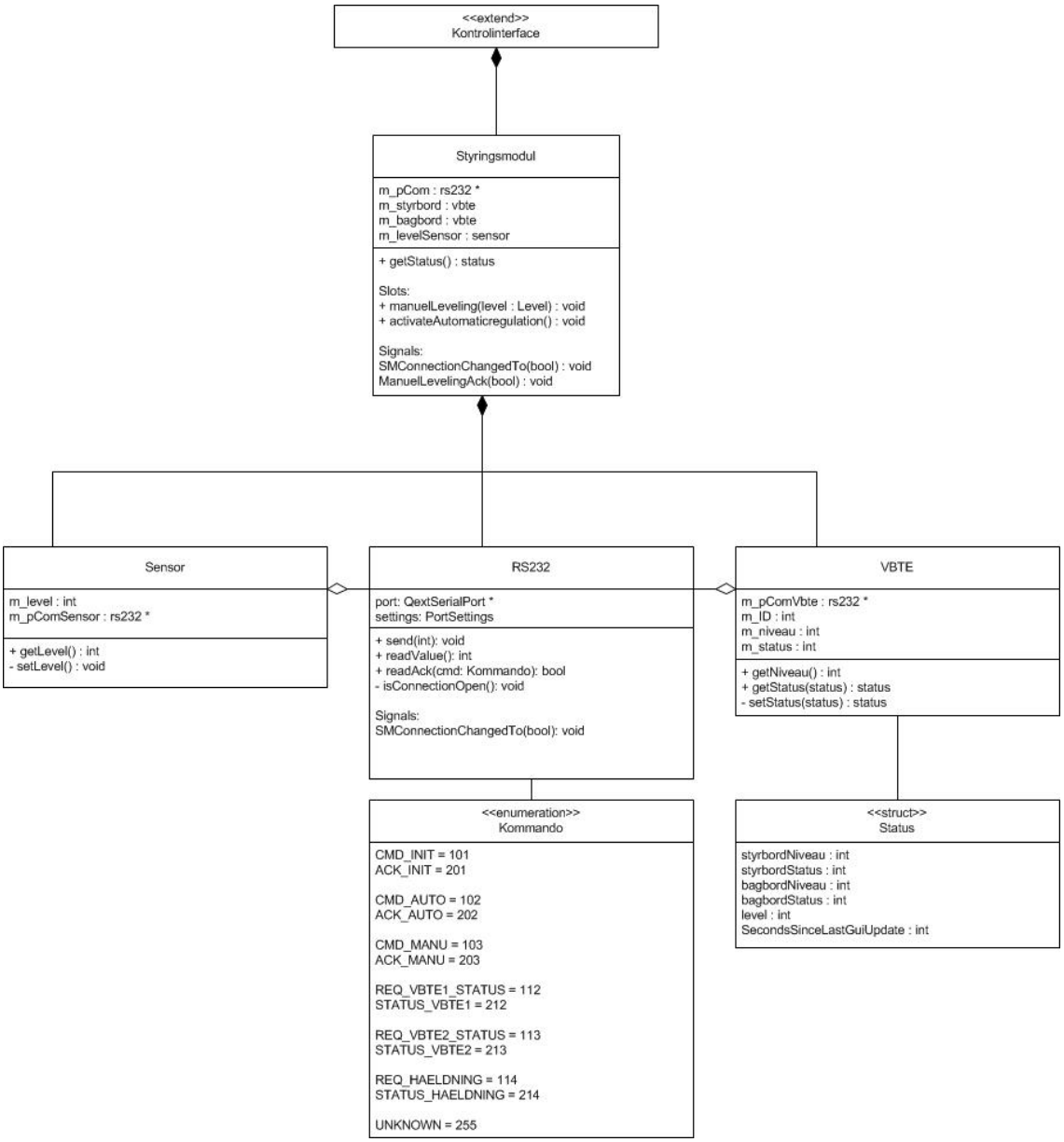
Tabel 2.1. Kontrolinterfacets klasser

Der er så udarbejdet et detaljeret klassediagram for klasserne i Kontrolinterfacet. Dette udarbejdes ud fra det detaljerede state machine diagram og de krav der er stillet i kravspecifikationen. Brugergrænsefladen udarbejdes ud fra de modeller der er aftalt med kunden og som ligger som bilag A i kravspecifikationen.
Det detaljerede klassediagram indeholder de samme klasser og relationer som det simplificerede. Derudover er der anført hvilke metoder, signaler, slots og attributter klasserne indeholder.
Signaler og slots er indført af Qt-frameworket og muliggør kommunikation mellem klasser

uden at koble klasserne unødvendig meget. En klasse abbonnerer så at sige på et signal. Udsendes signalet så vil funktionaliteten i en slot så blive kaldt. Flere klasser kan godt reagerer på samme signal. Således kan et signal godt påvirke flere elementer af programmet. Derudover er de anvendte enumeratorer og structs indført i diagrammet. Structs er indført i designet for at kunne pakke flere værdier ned og sende dem samtidigt. Enumerationer gør koden nemmere at læse og gør det nemt at implementere en protokol som fx protokol til Styringsmodulet.
Bemærk at klassediagrammet er delt op i to. Skæringsstedet er mellem Kontrolinterface-klassen og Styringsmodul-klassen og er markeret med «extend».
Klassediagrammet for Kontrolinterface-klassen og de tilhørende klasser kan ses på figur 2.3 og for Styringsmodul-delen henvises der til figur 2.4.



Figur 2.3. På figuren ses klassediagrammet for KI - Kontrolinterface-delen



Figur 2.4. På figuren ses klassediagrammet for KI - Styringsmodul-delen

2.1 Metode- og klassebeskrivelser

Når klassediagrammet er på plads skal metoderne i det beskrives. Det vil de blive gjort i dette afsnit.

2.1.1 MainWindow

Ansvar

Denne klasse indeholder de funktioner der er skrevet til Qt-formen MainWindow.ui hvori selve det grafiske er opbygget. Klassen indeholder de funktioner der anvendes i forbindelse med den grafiske brugergrænseflade. Det være sig når der kommer et input, eller der skal opdateres nogle værdier på skærmen.

Funktionsbeskrivelser

```
int getSecondsSinceLastGuiUpdate();
```

Beskrivelse:	En simpelt get-funktionen der returnerer værdien af klasseattributten *last_GUI_Update.*
Parametre:	Ingen
Returværdi:	`int secondsSinceLastGuiUpdate`

```
void updateGuiSlot(status temp);
```

Beskrivelse:	Bliver kaldt når GUI'en skal opdateres. Den modtager parameterent *temp* som er en struct af typen status. Ud fra denne struct hives de værdier ud, som skal vises på GUI'en. Værdierne vises ved de set-funktioner der er tilknyttet de anvendte widgets (og dermed en del af Qt-frameworket.) Når værdierne er opdateret vises det som en aktivitet i aktivitetsloggen
Parametre:	`status temp`
Returværdi:	Ingen

```
void showSMConnectionChange();
```

Beskrivelse:	Kaldes hvis SM-forbindelsen til styringsmodulet ændres fra forbundet til mistet forbindelse eller omvendt. Det udløser en aktivitet i aktivitetsloggen. Derudover skiftes lyset på gui'en. Parameteren state er den status som forbindelsen har ændret sig til.
Parametre:	`bool state`
Returværdi:	Ingen

```
void manuChangedHandle(Level samlet, int side, double level);
```

Beskrivelse:	Kaldes når brugeren har ændret i indstillingen til den manuelle hældning. Som parametre modtages hvilken side man ønsker at skibet skal hælde til (int side), hvor meget det skal hælde (double level) samt de to informationer samlet i en enum, Level samlet. Funktionen emitter signalet "manuChanged(Level temp). Det sætter også klasseattributerne manuelSide, manuelLevel samt autoActivated til deres rette værdier. Til sidst kaldes funktionen displayManuText(true)

Parametre:	`Level samlet`
	`int side`
	`double level`

Returværdi:	Ingen

`void showServerUpdateStatus(bool state);`

Beskrivelse:	Kaldes hver gang signalet DataServer::serverUpdateStatus() udsendes. Funktionen undersøger om forbindelsen har ændret sig ved at sammenligne med attributen serverConnection. Hvis forbindelsen har ændret sig udsendes dette som en aktivitet. Lyset ændres også således at det passer ved hjælp af setLight(DATA, serverConnection. Hvis vi modtager "true"vil last_DATA_Update opdateres til således til den nuværende værdi af sekunder siden epoch.
Parametre:	Ingen
Returværdi:	Ingen

`void showTimeSinceLastUpdate();`

Beskrivelse:	Kaldes hvert sekund. Funktionen opdaterer antallet af sekunder siden sidste overførelse af data til serveren eller til SM. Når tiden er længere end tiden mellem hver opdatering vil dette tal skifte til rødt.
Parametre:	Ingen
Returværdi:	Ingen

`void displayManuText(bool show);`

Beskrivelse:	Viser eller skjuler teksten med indstillingen af manuel hældning afhængig af parameteret show.
Parametre:	Ingen
Returværdi:	Ingen

`void activateAutoClicked();`

Beskrivelse:	udsendes når der er blevet trykket på knappen *activateAutoClicked*
Parametre:	Ingen
Returværdi:	Ingen

`void activateAutoClicked();`

Beskrivelse:	udsendes når der er blevet trykket på knappen *activateAutoClicked*
Parametre:	Ingen
Returværdi:	Ingen

`void manuChanged(Level grader);`

Beskrivelse:	Udsendes når der er blevet ændret en manuel indstilling.
Parametre:	`Level grader`
Returværdi:	Ingen

`void programTerminated();`

Beskrivelse:	Udsendes når programmet er blevet lukket ned.
Parametre:	Ingen
Returværdi:	Ingen

```
void btn_CloseApplication_clicked();
```

Beskrivelse:	Kaldes når luk-knappen på GUI'en er blevet trykket. Udsender signalet programTerminated() hvis brugeren bekræfter valget
Parametre:	Ingen
Returværdi:	Ingen

```
void on_btn_manu_clicked();
```

Beskrivelse:	Kaldes når der bliver trykket på knappen for manuel hældning. Viser dialogen "manuDialog".
Parametre:	Ingen
Returværdi:	Ingen

```
void on_btn_auto_clicked();
```

Beskrivelse:	Kaldes når der trykkes på *Automatisk Hældnings-knappen*. Aktiverer automatisk styring og deaktiverer den manuelle.
Parametre:	Ingen
Returværdi:	Ingen

```
void setLight(lys id, bool state);
```

Beskrivelse:	Sætter lyset i forhold til parametrene. lys er en enum der bestemmer hvilket element lyset skal ændres for. state er om lyset skal være tændt eller ej
Parametre:	`lys id` `bool state`
Returværdi:	Ingen

```
void alarmChangerCheck(QString sentence, int critical_point, int value, bool &e
```

Beskrivelse:	Tester om alarmerne har ændret sig. Sentence er starten af den sætning der skrives i aktivitetsloggen. critical_point er det kritiske punkt for det emne der arbejdes på. Value er den værdi den har. Earlier_state er hvilket stadie alarmen var i tidligere.

Parametre:	`QString sentence`
	`int critical_point`
	`int value`
	`bool &earlier_state`

Returværdi:	Ingen

```
void levelEnumConverterAndSetter(int level);
```

Beskrivelse:	Konverterer en integer baseret på Level-enumeratoren om til en side og en vinkel.
Parametre:	`int level`
Returværdi:	Ingen

2.1.2 Kontrolinterface

Ansvar

Dette er hovedklassen hvori selve programmet lever. Oprettelsen af et objekt af denne klasse er derfor også det eneste der sker i main.

Funktionsbeskrivelser

```
void update();
```

Beskrivelse:	Får en status-struct fra SM-klassen udfyldt med de nuværende værdier for systemet. Tilføjer antal sekunder siden sidste gui-update ved hjælp af getSecondsSinceLastGuiUpdate. Denne struct udsendes med signalet updateSignal(status recentValues)
Parametre:	Ingen
Returværdi:	Ingen

```
void updateSignal(status recentValues);
```

Beskrivelse:	Signalet der sendes når gui og server skal updateres. Indeholder en struct med alle relevante værdier.
Parametre:	`status recentValues`
Returværdi:	Ingen

```
void terminate();
```

Beskrivelse:	Udsendes når brugeren ønsker at terminere programmet.
Parametre:	Ingen
Returværdi:	Ingen

```
void destroyMe();
```

Beskrivelse:	Muliggør nedlæggelse af klassen med et funktionskald.
Parametre:	Ingen
Returværdi:	Ingen

2.1.3 DataServer

Ansvar

Klassen står for al kommunikation med serveren via en TCP-forbindelse. Forbindelse oprettes og nedlægges hver gang der tages kontakt. DataServer-objektet opdaterer serveren når den får ordre om det fra Kontrolinterface-klassen.

Funktionsbeskrivelser

```
void onDelete();
```

Beskrivelse:	Kaldes i destruktoren. Sender en besked til serveren om at programmet termineres .
Parametre:	Ingen
Returværdi:	Ingen

```
void updateServerSlot(status recentValues);
```

Beskrivelse:	Kaldes når signalet updateSignal(status recentValues) udsendes. Funktionen sørger så for at der via TCP-forbindelsen bliver udsendt de relevante værdier til databasen.
Parametre:	`status recentValues`
Returværdi:	Ingen

```
void serverUpdateStatus(bool status);
```

Beskrivelse:	Udsendes når databasen er blevet opdateret. Parameteren "status"indikerer hvorvidt overførelsen var succesfuld eller ej. Der ventes ikke noget svar fra databasen.
Parametre:	`bool status`
Returværdi:	Ingen

2.1.4 manuDialog

Ansvar

manuDialog-klassen står for håndtering af det vindue der åbnes ved tryk på *Manuel Hældningsregulering-knappen.*

Funktionsbeskrivelser

```
void manuSet(Level degrees, int side, double level);
```

Beskrivelse:	Udsendes i funktionen accept(). Der medsendes de data som brugeren har valgt på dialogen.

Parametre:	`Level degrees`
	`int side`
	`double level`

Returværdi:	Ingen

```
void accept();
```

Beskrivelse:	Kaldes når der trykkes på "OK-knappen på dialogen. Det indtastede omdannes til en værdi i forhold til enumeratoren "Level". Dialogen lukkes og signalet manuSet(...) udsendes

Parametre:	`Level degrees`
	`int side`
	`double level`

Returværdi:	Ingen

2.1.5 Styringsmodul

Ansvar

Klassen har samme rolle som det fysiske styringsmodul har i systemet. Det giver kontrolinterfacet adgang til sensor-værdier og vandstandsniveauer igennem sine underklasser, VBTE og Sensor.

Funktionsbeskrivelser

```
status getStatus();
```

Beskrivelse:	Indhenter værdierne for systemet fra VBTE'er og Hældningssensor. Disse værdier sættes ind i structen temp som så returneres.
Parametre:	Ingen
Returværdi:	`status recentValues`

```
void manuelLeveling(Level level);
```

Beskrivelse:	Sender kommando og vinkel til PSoC over RS232 vha. objektet m_pCom.
Parametre:	`Level level`
Returværdi:	Ingen

```
void activateAutomaticRegulation();
```

Beskrivelse:	Sætter hvorvidt automatisk regulering skal være aktiveret eller ej til styringsmodulet.
Parametre:	Ingen
Returværdi:	Ingen

```
void SMConnectionChangedTo(bool status);
```

Beskrivelse:	Udsendes hver gang der har været en overførelse. Status indikerer hvorvidt overførelsen var succesfuld eller ej.
Parametre:	`bool status`
Returværdi:	Ingen

```
void ManuelLevelingAck(bool status);
```

Beskrivelse:	Udsendes når der er blevet sendt kommandoen CMD_MANU til styrings-modulet. Bool status indikerer hvorvidt overførelsen var succesfuld ej.
Parametre:	`bool status`
Returværdi:	Ingen

2.1.6 VBTE

Ansvar

Håndterer værdierne for hver sin vandballasttankenhed. Kommunikerer til det fysiske styringsmodul ved hjælp af RS232-klassen.

Funktionsbeskrivelser

```
int getNiveau();
```

Beskrivelse:	Kalder setNiveau og returnerer værdien af niveau
Parametre:	Ingen
Returværdi:	int niveau()

```
status getStatus(status temp);
```

Beskrivelse:	Skaffer status vha. rs232-objektet og sætter niveau og status i structen temp som herefter returneres.
Parametre:	status temp()
Returværdi:	status temp()

2.1.7 Sensor

Ansvar

Håndterer værdierne for hældningssensoreren. Kommunikerer til det fysiske styringsmodul ved hjælp af RS232-klassen.

Funktionsbeskrivelser

```
int getLevel();
```

Beskrivelse:	Skaffer værdien af hældningen på skibet vha. rs232-objektet og returnerer det.
Parametre:	Ingen
Returværdi:	int level

2.1.8 RS232

Ansvar

Håndterer kommunikationen til det fysiske styringsmodul ved protokollen der ses i enumeratoren "Kommando".

Funktionsbeskrivelser

```
void send(int cmd);
```

Beskrivelse:	Sendefunktion. Sender den medsendte integer.
Parametre:	int cmd
Returværdi:	Ingen

```
int readValue();
```

Beskrivelse:	Modtagerfunktion. Returnerer den læste værdi.
Parametre:	Ingen
Returværdi:	int receivedValue

```
bool readAck(Kommando cmd);
```

Beskrivelse:	Kalder readValue() og sammenholder dennes returværdi med den værdi der er medsendt som parameter (cmd). Returnerer hvor vidt de to var identiske.
Parametre:	Kommando cmd
Returværdi:	bool status

```
void SMConnectionChangedTo(bool status);
```

Beskrivelse:	Når der har været en overførelse udsendes dette signal. Status indikerer hvorvidt overførelsen var succesfuld eller ej.
Parametre:	bool status
Returværdi:	Ingen

```
void SMConnectionChangedTo(bool status);
```

Beskrivelse:	Når der har været en overførelse udsendes dette signal. Status indikerer hvorvidt overførelsen var succesfuld eller ej.
Parametre:	bool status
Returværdi:	Ingen

```
bool isConnectionOpen();
```

Beskrivelse:	Tester om der er forbindelse til PSoC
Parametre:	Ingen
Returværdi:	bool connectionState

Databasen 3

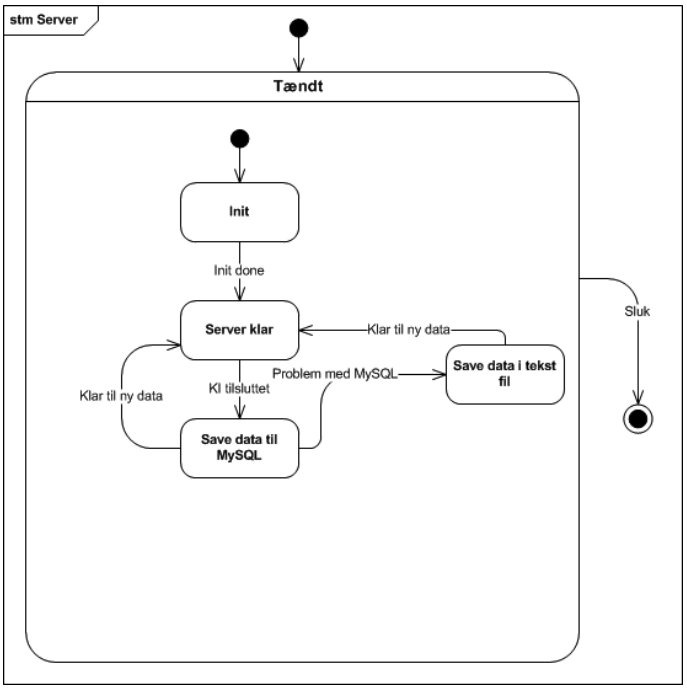
Nedenfor følger design af software til databasen og dens interface. Dette er lavet på baggrund af kravspecifikation og systemarkitektur.

3.0.9 Modulets Ansvar

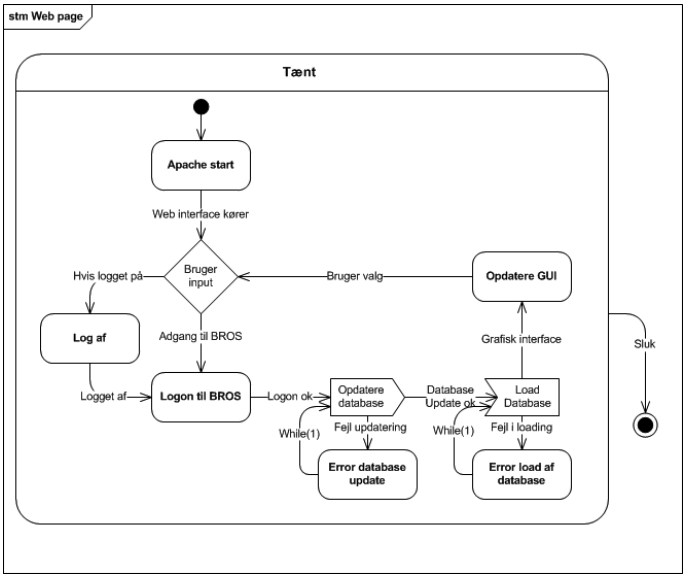
Databasen er der hvor havneterminalens personale kan aflæse data fra skibet. Disse data er sendt fra KI. Programmerne indeholder brugergrænseflader, der opfylder kravene beskrevet i kravspecifikationen. Her kan der også ses en prototype på brugergrænsefladen. Databasemodulet har tre dele; serveren, websiden og en MySQL-database.
Serveren står for kommunikationen immellem KI og Databasen. Serveren modtager data fra KI og lagrer disse i en tekstfil.
Webinterfacet giver brugeren mulighed for at se info om BROS samt at logge sig ind i BROS-databasen hvorfrax data om skibe, der er tilsluttet systemet, kan aflæses. Webinterfacets tre vigtigste funktioner er at gemme ny data til MySQL-databasen, slette den tekstfil som serveren lavede og vise data for brugeren. For at håndtere webinterfacet, der benytter sig af php (webprogrammering) kræves der en webserver, som er i stand til at håndtere dette. En af de mest udbredte er apacheserveren, som også benyttes for denne webside. **MySQL-databasen** er en database, som er installeret på computeren. Alle data, som er sendt fra KI, er lagret i MySQL-databasen.

3.0.10 State machine diagram

Designet af Databasen tager udgangspunkt i arkitekturen fremstillet i artefaktet Systemarkitektur. Her blev der fremstillet et state machine for serveren webinterfacet. State machinet var simplificeret og er i designfasen blevet udvidet. Resultatet ses i *figur 3.1* for serveren og *figur 3.2* webinterfacet. Figuren beskriver hvilke stadier programmet kan befinde sig i og sammenhængen imellem dem.



Figur 3.1. State machine diagram for serveren



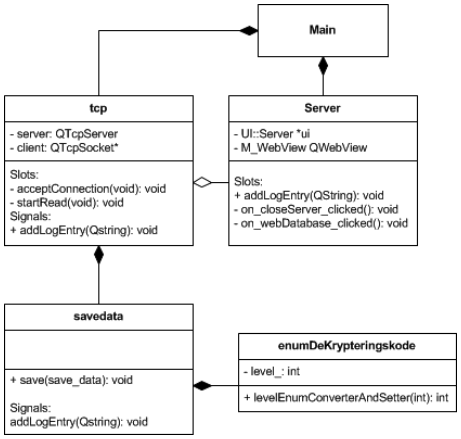
Figur 3.2. State machine diagram for serveren

3.0.11 Klassediagrammer

Nedenfor ses klassediagrammerne for databasen. Bemærk databasemodulet er lavet som en serverdel og en webdel

Serveredelen er opskrevet som et normalt klassediagram og websiden er opskrevet som et modificeret klassediagram. Dette skyldes at websiden er opbygget som en blanding imellem HTML og php. HTML kan man ikke lave et decideret klassediagram, da der ikke findes funktionskald i dette, men blot includes. Php har derimod funktionskald og

er lavet traditionelt. For at lette læsningen af diagrammet har alle blokke i webinterface-klassediagrammet skrevet i toppen om det er HTML eller php.

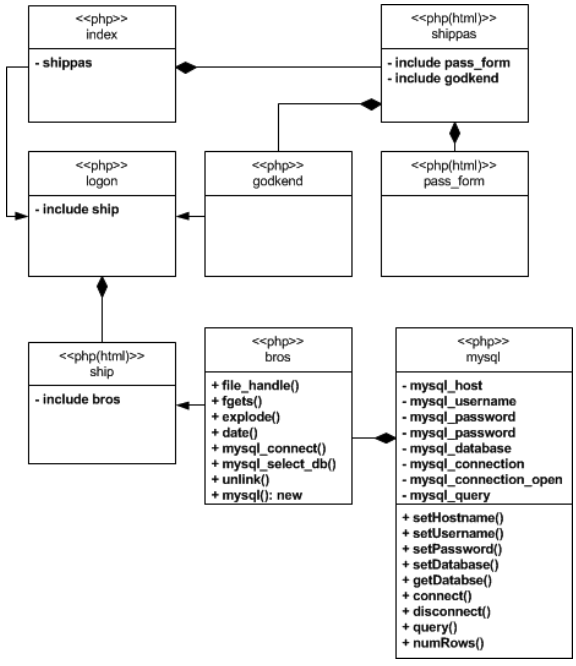


Figur 3.3. Klassedigram for databasens servere

Serverens klasser

Server:	Hovedklassen. Denne klasse står for udskrivning til GUI og meddele fejl.
tcp:	Står for modtagelse af data fra KI. Denne splitter den modtagede streng og sender den videre til savedata.
savedata:	Står for at gemme alle data til en tekstfil.
enumDeKryptering	Står for at dekryptere LEVEL til grader.

Tabel 3.1. Serveren klasser



Figur 3.4. Klassedigram for databasens webinterface

Webinterfacets klasser

index:	Denne er første side brugeren kommer til. Includer shippas.
shippas:	Denne includeres i index og håndtere adgangskoden fra brugeren ved hjælp af pass_form og godkend.
pass_form:	Indtastningsblokken for adgangskode til databasen.
godkend:	Checker om brugeren har tastet korrekt adgangskode, og sender brugeren videre til logon ved korrekt password.
logon:	Er hovedsiden i databasen. Denne includere ship og bros.
ship:	Denne håndtere de skibe der er tilsluttet BROS database.
bros:	Denne gemmer ny data til MySQL databasen. Sletter den filen data var lagret i. Henter og viser alle data i MySQL databasen.
mysql:	Tager kontakt til MySQL databasen. opretter forbindelse. Danner en row til håndtering af de data, der skal vises for brugeren.

Tabel 3.2. Webinterfacets klasser

3.0.12 Funktionsbeskrivelser

Server

Denne header står for at starte serveren og starte GUI for korte informationer til brugeren. Alle informationer om serverstart, connection og datamodtagelse vil blive udskrevet her.

`Void addLogEntry( QString )`

Beskrivelse	Står for at udskrive beskeder fra tcp-klassen til GUI
Parametre:	Ui::Server *ui;
Returværdi:	Ingen

`Void on_closeServer_clocked( )`

Beskrivelse:	Denne funktion håndterer luk-knappen. Ved trykknappen, vil brugeren blive bedt om at svare ja eller nej til at lukke serverconnectionen. En knap, der er implementeret som en hjælp, men ikke videre beskrevet
Parametre:	Ingen
Returværdi:	Ingen

`Void on_webDatabase_clicked( )`

Beskrivelse:	Denne funktion står for at håndtere den direkte adgang til den webbaserede database. Ved tryk vil brugeren få åbnet et nyt vindue med databaseadgang
Parametre:	QWebView* m_pWebView
Returværdi:	Ingen

update

Denne header indeholder de data, der skal sendes fra tcp-klassen til savedata

tcp

Denne header står for at forbindelsen fra KI kan oprettes. Denne opretter en socket og klargør serveren til modtagelse af data. Ved modtagelse af data sørger klassen for opdeling af strengen og flytte dem til variabler for videre behandling

`Void addLogEntry( QString )`	
Beskrivelse:	Står for at udskrive beskeder fra tcp-klassen til GUI
Parametre:	Ui::Server *ui;
Returværdi:	

`Void acceptConnection( void )`	
Beskrivelse:	Står for at acceptere forbindelse fra KI og for at tilslutte.
Parametre:	QTcpServer server
	QTcpSocket* client
Returværdi:	

`Void startRead( void )`	
Beskrivelse:	Læser data fra socket. Data, som bliver modtaget, som en streng, bliver splittet og lagt ud i de fem variabler ID, STYRBORD, BAGBORD, LEVEL og TIME
Parametre:	QTcpSocket* client
Returværdi:	

enumdekrypteringskode

Denne header står for at dekryptere LEVEL-niveau til grader fra sm.

`int levelEnumConverterAndSetter( int level )`	
Beskrivelse	Oversætter LEVEL til grader ud fra protokollen fra uart
Parametre:	level
Returværdi:	level

savedata

Denne header står for at håndtere lagring af data modtaget fra skibet. Den lagrer dataerne i ship.txt.

`int save( save_data )`	
Beskrivelse	Står for at gemme data til en tekstfil.
Parametre:	Ingen
Returværdi:	Ingen

Webinterface

Webinterface står for at fremvise skibsdata grafisk for terminalpersonalet. Desuden står den for at lagre data og loade data fra MySQL-databasen.

bros

`file_handle( )`

Beskrivelse:	Klargør tekstfil til læsning
Parametre:	Ingen
Returværdi:	Ingen

`fgets ( )`

Beskrivelse:	Tager fat i tekststrengen i tekstfilen
Parametre:	Ingen
Returværdi:	Ingen

`explode( )`

Beskrivelse:	Splitter linien op i flere variable
Parametre:	Ingen
Returværdi:	Ingen

`date( )`

Beskrivelse:	Opdaterer dato og tid når der gemmes til MySQL-databasen
Parametre:	Ingen
Returværdi:	Ingen

`mysql_connect( )`

Beskrivelse:	connecter til MySQL-databasen
Parametre:	Ingen
Returværdi:	Ingen

`mysql_select_db( )`

Beskrivelse:	Vælger hvilken database, der skal benyttes i MySQL
Parametre:	Ingen
Returværdi:	Ingen

`unlink( )`

Beskrivelse:	Sletter den tekstfil, som der lige er blevet læst fra
Parametre:	Ingen
Returværdi:	Ingen

`new mysql( )`

Beskrivelse:	Opretter ny databaseforbindelse
Parametre:	Ingen
Returværdi:	Ingen

mysql

`setHostName( )`

Beskrivelse:	Sætter servernavn/serverip
Parametre:	Ingen
Returværdi:	Ingen

`setUserName( )`

Beskrivelse:	Skriver brugernavn til databasen
Parametre:	Ingen
Returværdi:	Ingen

`setPassword( )`

Beskrivelse:	Skriver password til databasen
Parametre:	Ingen
Returværdi:	Ingen

`setDatabase( )`

Beskrivelse:	Fortæller MySQL hvilken database, der skal benyttes
Parametre:	Ingen
Returværdi:	Ingen

`getDatabase( )`

Beskrivelse:	Tager fat i databasen
Parametre:	Ingen
Returværdi:	Ingen

`connect( )`

Beskrivelse:	Står for at samle localhost, username, password, database og connecte til databasen
Parametre:	Ingen
Returværdi:	Ingen

`disconnect( )`

Beskrivelse:	Lukker databaseforbindelsen
Parametre:	Ingen
Returværdi:	Ingen

`query( )`

Beskrivelse:	Opretter databasekø og skriver data til skærm
Parametre:	Ingen
Returværdi:	Ingen

numRows( )	
Beskrivelse:	Checker hvor mange rækker der findes i databasen. Bruges desuden til at udskrive om databasen er tom.
Parametre:	Ingen
Returværdi:	Ingen

3.0.13 Tilpasning

Der er mulighed for at opsætte serverdelen sådan at denne lagrer direkte i MySQL-databasen hvis man har et ønske om at mindske ansvaret for websiden. For at gøre dette, kan man i tcp-klassens funktion startRead(void) erstatte funktionskaldet til klassen savedata() med funktionen save(tmp) og i stedet benytte klassen SQL med funktionskaldet saveSQL(tmp). funktionen saveSQL(tmp) har save(tmp) som, i tilfælde af at der opstår fejl med at gemme data til databasen, vil sikre at data bliver lagret i en backupfil, som så kan håndteres af websiden.

3.0.14 Apache

Apache HTTP-Server er en webserver fra Apache Software Foundation. Apache er en ofte benyttet webserver og er en open source-serverprogram. Serveren installeres på en computer og kan herefter benytte ved at benyttes den lokale ip-adresse 127.0.0.1 eller localhost. 'hvis computeren med Apache-serveren er placeret i et større netværk og skal benyttes fra andre computere, kan denne tilgås fra disse ved at indtaste computerens oprindelige ip-adresse på netværket. Dette giver også mulighed for at benytte serveren fra internettet ved at koble den op via DNS. Hvis dette gøres, skal man specielt være opmærksom på at lukke portnumrerne 20 og 21, da disse ofte er portene, som bliver hacket. Desuden bør man ligge adgangskode på serveren for uautoriseret brug. I BROS er Apache-serveren placeret i et internt netværk og derfor er det ikke nødvendigt at lukke portene eller benytte sig af adgangskodetilladelse for at arbejde til serveren. Generelt står apache-serverene for ca 60% af alle webserere i verden. fxnotekilde: IKN bog og wikipedia

3.0.15 mySQL

MySQL er en flertrådet SQL-databaseserver, som understøtter mange samtidige brugere. SQL(Structured Query Language) er det mest populære databasesprog i dag. MySQL'er et klient-/serverprogram, der består af en server (mysqld) og mange forskellige klientprogrammer.
MySQL er bygget op omkring forskellige databaser på en server, ofte har hver enkelt bruger en speciel adgang til en databse med en overordnet root-bruger, der har adgang til alle databaser.
MySQL er en relationel database, hvori man kan oprette flere tabeller til at håndtere flere ting. I BROSdatabasen benyttes der en database med navnet bros. I denne database kan der oprettes tabeller til de skibe, der skal kunne gemmes data for. C++ og php er i stand til at se på om en tabel er oprettet. Hvis den ikke er det, kan sprogene selv oprette disse(dette er ikke indkodet, da vi kun arbejder med et skib).
For at tilgå mySQL kan man benytte sig af terminalen [1] eller grafisk brugergrænseflade

[1] beskrevet i apendix

som f.eks. phpMyAdmin (benyttet under udviklingen). MySQL kan benytte flere forskellige datatyper, som vil blive beskrevet i apandix for MySQL.

SM 4

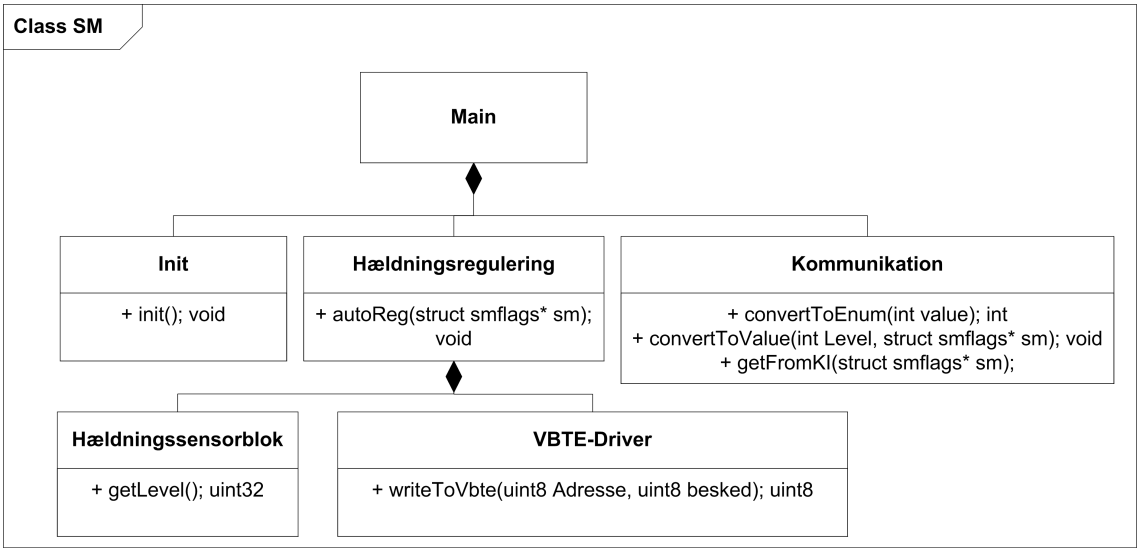
Dette afsnit beskriver designet af styringsmodulet, SM.

4.1 Klassens ansvar

Styringsmodulet har til ansvar at holde styr på hældningssensoren og værdierne fra VBTE. Den kommunikere med KI og VBTE med indbyggede API'er fra Cypress PSoC 5 biblioteker.

4.2 Klassediagram

Nedenfor ses klassediagrammet for SM. Bemærk at koden dog er i C men for overblikket er der lavet klassediagram.



Figur 4.1. På figuren ses klassediagrammet for SM

4.3 Variabler

Variabel	Beskrivelse
`autoflag`	Denne variable er et flag der holder styr på automatisk regulering. 0 = slukket. 1 = tændt.
`levelVal`	En variable med vores level værdi.
`VBTE1Niveau og VBTE2Niveau`	Holder styr på vandniveauet i ballasttanke i %.
`VBTE1Status og VBTE2Status`	Holder styr på ventilstatus for VBTE1 og 2. 0 = begge lukket. 1 = ind ventil åben. 2 = ud ventil åben.
`vinkelVal`	Indeholder værdien for manuel regulering.

Alle variabler er indkapslet i en struct navngivet "smflags".

4.4 Funktionsbeskrivelser

4.4.1 Init

Ansvar

Denne header har til ansvar at sørge for alle komponenter opretter og initieret. `void init( void);`

Beskrivelse:	Funktionen anvender API'et fra Cypress componenter og står for at initiere og starte vores PSoC hardware. Den sætter også et register tilhørende vores Accelerometer.
Parametre:	ingen
Returværdi:	ingen

4.4.2 Hældningssensorblok

Ansvar

Denne header har til ansvar at hente levelværdien ind fra vores accelerometer. `uint32 getLevel( void);`

Beskrivelse:	Funktionen anvender API'et fra Cypress componenter og venter på at vores ADC henter convertere det analoge signal. Funktionskald for ADC ses i PSoC databladet.
Parametre:	ingen
Returværdi:	`uint32 levelVal`

4.4.3 Hældningsregulering

Ansvar

Denne header har til ansvar at styre automatisk regulering. `void autoReg( struct smflags* sm);`

Beskrivelse:	autoReg anvender værdier fra VBTE moduler samt KI til at holde systemet i et bestemt level. Funktionen starter med at checke på automatisk og manuel styrings flagene. Derefter kalder den getLevel agere ud fra niveauet. Funktionen vil altid tømme fra en tank før den begynder at fylde en anden.
Parametre:	`struct smflags* sm`
Returværdi:	ingen

4.4.4 VBTE-Driver

Ansvar

Denne header har til ansvar at kommunikere med VBTE modulerne. `uint8 writeToVbte( uint8 Adresse, uint8 besked);`

Beskrivelse:	writeToVbte anvender I2C fra Cypress PSoC 5 API til at skrive til VBTE modulerne. Den tager adressen og beskeden man skal sende og sender til pågældende enhed. Derefter venter den på svar som den så returnere.
Parametre:	`uint8 Adresse` `uint8 besked`
Returværdi:	`uint8 VbteNiveau`

4.4.5 Kommunikation

Ansvar

Denne header har til ansvar at kommunikere med KI enheden.

```
int convertToEnum( int value);
```

Beskrivelse:	funktionen tager en level værdi ind for så at konvertere den til en Enum(integer) som den returnere.
Parametre:	`int value`
Returværdi:	`int Enum`

```
void convertToValue(int Level, struct smflags* sm);
```

Beskrivelse:	Funktionen tager en enum og en pointer som den så konvertere til en level værdi og sætter i sm structen.
Parametre:	`int Level,` `struct smflags* sm`
Returværdi:	ingen

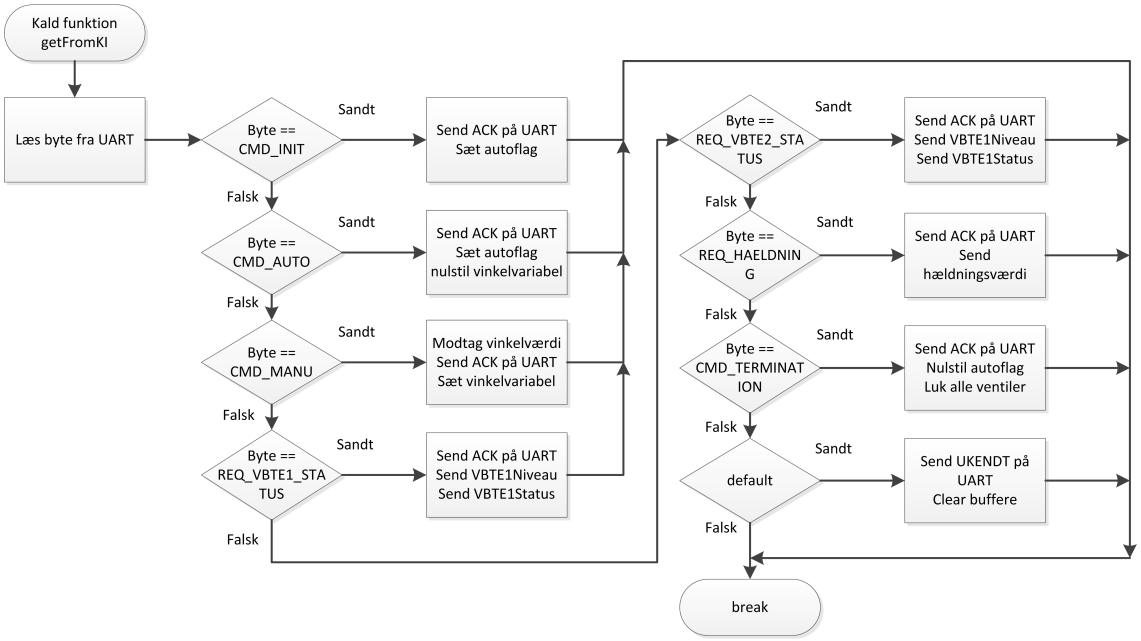
```
void getFromKI( struct smflags* sm);
```

Beskrivelse:	Funktionen anvender UART fra Cypress PSoC 5 API'en til at modtage en besked fra KI modulet som den så vurdere og agere på. Når den har modtaget noget sender den en ack tilbage til KI modulet. Derefter handler den og hvis det er nødvendigt sender data til KI.
Parametre:	`struct smflags* sm`
Returværdi:	ingen

4.5 Eventuelle Sekvensdiagrammer og state machines

4.5.1 getFromKI

Funktionalitet af getFromKI beskrives med et flowchart:



Figur 4.2. Flowchart for getFromKI

Hver ACK er specifik for den læste UART værdi. Dette gør at det muligt for KI modulet at vide hvad SM'en modtog.

VBTE 5

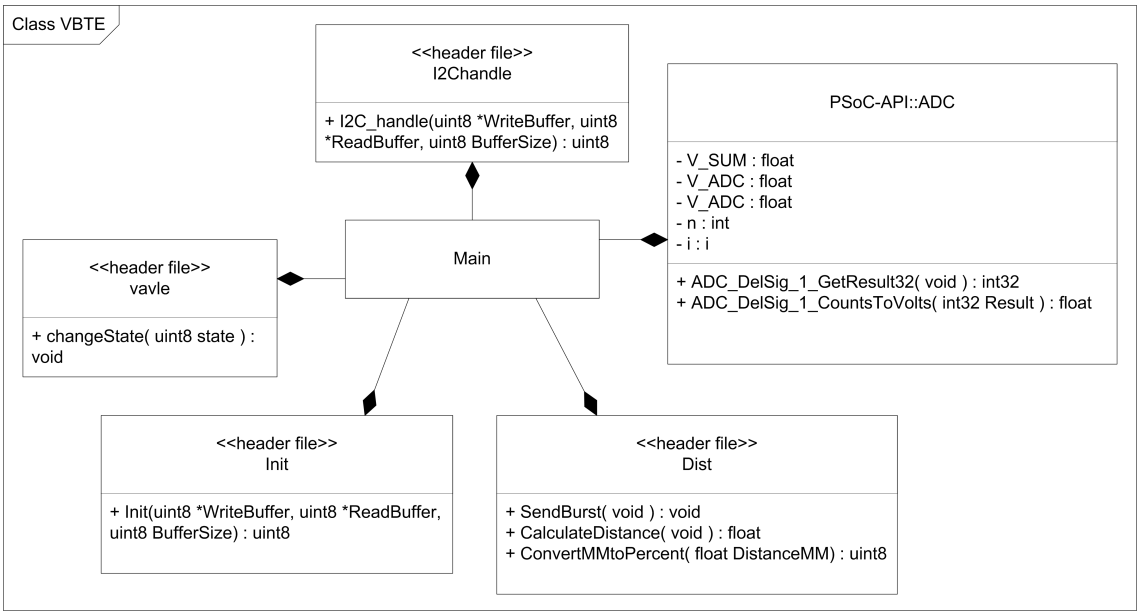
Nedenfor følger design af software til VBTE. Dette er lavet på baggrund af kravspecifikation og systemarkitektur. Bemærk der i dette design dokument blandt andet ikke er beskrevet mixer, pga osv. da deres eneste funktion er "Start();". Derudover er der en betydelig hardware del knyttet til dette modul, der refereres derfor til detaljeret hardware design for yderligere detaljer om VBTE modulet.

5.1 Modulets ansvar

Som beskrevet i systemarkitektur står VBTE'en for at måle vandniveauet i ballasttankene samt at lukke vand ind eller ud af ballasttankene. Hertil er der også en kommunikation med SM modulet indeholende instruktioner.

5.2 Klassediagram

Nedenfor ses klassediagrammet for VBTE. Bemærk at koden dog er i C men for overblikket er der lavet klassediagram.



Figur 5.1. På figuren ses klassediagrammet for VBTE **Billedet skal opdateres**

5.3 Globale variabler

Variabel	Beskrivelse
`BurstLengthVal`	Denne variabel er anvendt til at håndtere antallet af perioder burstet bliver sendt med.
`WaitBurstVar`	Bliver brugt til nonblocking delay til SendBurst funktionen.
`BurstTimerVal`	Holder på Timerens værdi når et burst er sendt.
`DistanceTimerVal`	Holder værdien på timeren når et burst er modtaget.
`CalcDistFlag`	Bliver sat når et burst er modtaget så en afstand kan blive beregnet.
`BurstFlag`	Bliver sat når et burst bliver sendt og hevet ned når et burst er modtaget. Dette sker for ikke at få flere detektioner på samme signal.

5.4 Metode- og klassebeskrivelser

5.4.1 Init

Ansvar

Denne header har til ansvar at initiere alle moduler og blokke på PSoC'en. Disse funktioner hentes fra PSoC'ens API.

Funktionsbeskrivelser

```
void Init( uint8* WriteBuffer, uint8* ReadBuffer, uint8 BufferSize )
```

Beskrivelse:	Funktionen anvender API'et fra alle PSoC blokke anvendt i designet og kalder deres start funktion. Derudover initierer den også read- og writebuffer til I2C modulet.

Parametre:	`uint8* WriteBuffer`
	`uint8* ReadBuffer`
	`uint8 BufferSize`

Returværdi:	Ingen

5.4.2 Valve

Ansvar

Denne header har til ansvar at styre ventilerne ud fra "state-variablen modtaget fra I2C_handle. Headeren benytter PSoC-API'et til kontrol registre..

Funktionsbeskrivelser

`void ChangeState( uint8 state )`	
Beskrivelse:	Funktionen modtager state som indeholder indformationer om ventilerne skal være lukkede eller hvilken ventil der skal være åben. Den benytter PSoC5 API'et for kontrol register.
Parametre:	`uint8 state`
Returværdi:	Ingen

5.4.3 Dist

Ansvar

Denne header har til ansvar at sende burst, beregne afstanden samt at omregne afstanden til procent.

Funktionsbeskrivelser

`void SendBurst( void )`	
Beskrivelse:	Denne metode aktiverer en 40kHz clock og tæller perioderne op til 10, lukker for burstet og ligger timerens værdi ind i den globale variabel BurstTimerVal. Herefter sættes BurstFlag'et.
Parametre:	Ingen
Returværdi:	Ingen

`float CalculateDistance( void )`	
Beskrivelse:	Denne metode anvender BurstTimerVal og DistanceTimerVal til at finde ud af hvor mange clocks der er gået fra burstet er blevet sendt til det igen er blevet registreret.
Parametre:	Ingen
Returværdi:	`float DistanceMM`

`uint8 ConvertMMtoPercent( float )`	
Beskrivelse:	Metoden modtager afstanden i millimeter og returnerer hvor mange % der er i tanken
Parametre:	`float DistanceMM`
Returværdi:	`uint8 DistancePercent`

5.4.4 I2Chandle

Ansvar

Denne header har til ansvar at håndtere I2C. Den kigger om der er kommet noget i writebufferen. Er der kommet noget i bufferen kigger den efter hvilket tilstand det er der skal ændres til og så smider den afstanden i procent i read bufferen.

Funktionsbeskrivelser

`uint8 I2C_handle( uint8* WriteBuffer, uint8* ReadBuffer, uint8 BufferSize )`

Beskrivelse:	Funktionen anvender API'et fra I2C blokken i PSoC miljøet. Med disse tjekker den om der er fyldt nyt i bufferen og aflæse dette. Herfer kalder den funktionen I2C_decode til at afkode beskeden fra SM. Herefter klargøres readbufferen til at sende vandniveau tilbage.

Parametre:	`uint8* WriteBuffer`
	`uint8* ReadBuffer`
	`uint8 BufferSize`

Returværdi:	`uint8 State`

5.4.5 PSoC-API::ADC

Ansvar

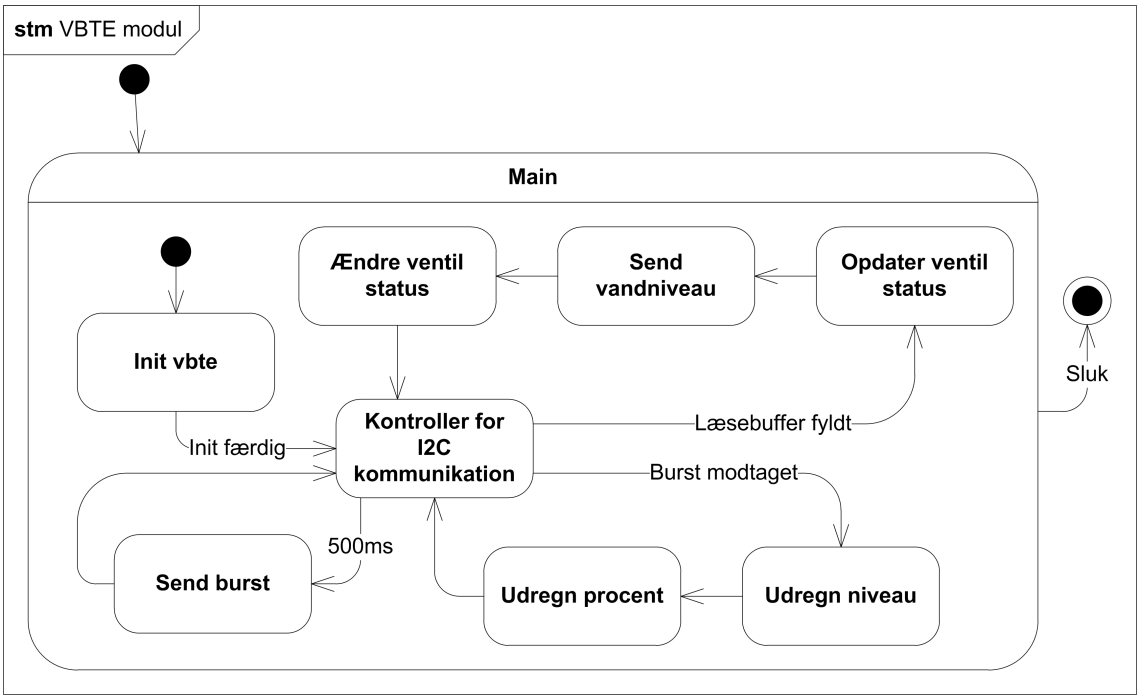
Denne header er kun beskrevet fordi der er implementeret en funktionalitet i dette API. Der er under ADC

Beskrivelse

Inde i ADC'ens interrupt header tilføjes funktionalitet så der, hver gang der bliver samplet, bliver valideret på om der er en detektion. Er der en detektion sættes flaget til udregning af afstand samt flaget for burst sættes til 0 igen. For at gøre dette anvendes funktionerne fra API'et for ADC'en.

5.4.6 State Machine

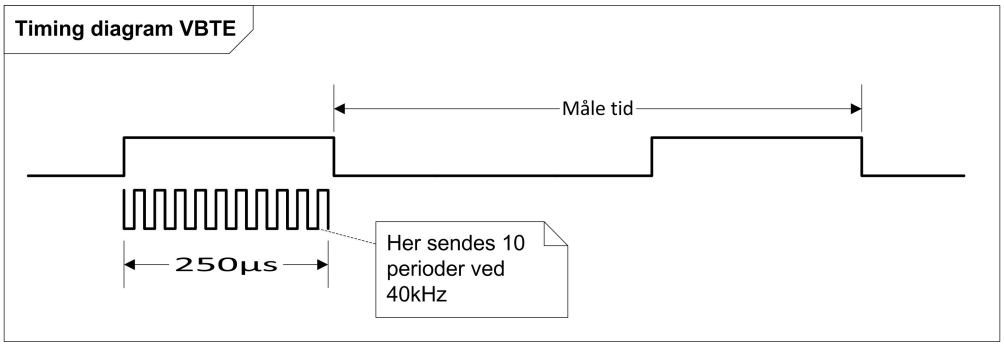
Nedenfor ses statemachine der beskriver det overordnede flow i VBTE programmet.



Figur 5.2. Statemachine for VBTE program

5.4.7 Timing Diagram

Nedenfor ses timing diagram for en ultralydspuls til afstandsmåling



Figur 5.3. Timing diagram for VBTE ultralydspuls

5.4.8 Interrupt rutiner

I dette afsnit beskrives interrupt rutinerne i VBTE programmet.

isr_dist

Isr_dist har til ansvar at tælle den globale variabel `WaitBurstVar` op. Den bliver triggeret af en clock på 1kHz. Hver gang variablen havner over 500 bliver `SendBurst();` funktionen kaldt og variablen bliver nulstillet.

isr_counter

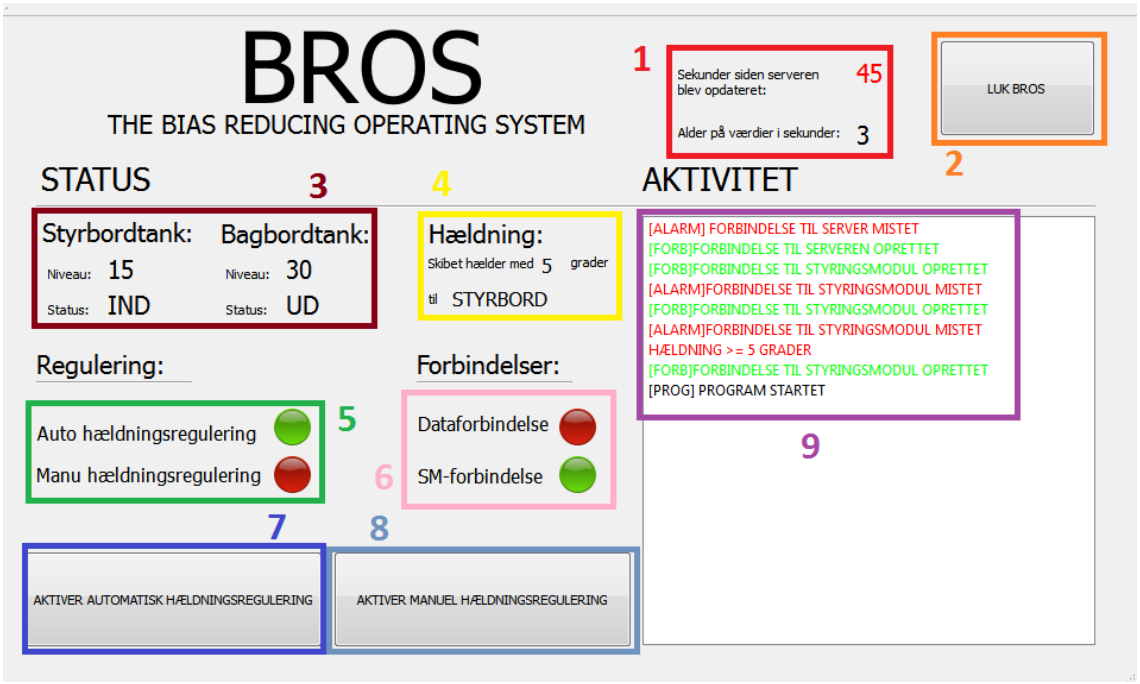
Isr_counter tæller variablen `BurstLengthVal` op. Denne anvendes til at styre at der kun bliver sendt 10 peroder i et burst.

Appendix A: Kontrolinterface

I dette appendix vil jeg gå nærmere ind på opbygningen af den grafiske brugergrænseflade på Kontrolinterfacet.

Hovedvindue

Det første vindue man ser ved programopstart er hovedvinduet, som vist på



Figur 5.4. På figuren ses hovedvinduet for Kontrolinterface-programmet

Hovedvinduets elementer

1: Forsinkelse i sekunder	Det øverste tal fortæller tiden i sekunder siden serveren sidst er blevet opdateret succesfuldt. Nedenunder udskrives tiden i sekunder siden værdierne i boks tre og fire er blevet opdateret.
2: Nedlukningsknap	Anvendes til at lukke programmet. Programmet åbner dialogen som ses i 5.4.8
3: Vandballasttankene	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.
4: Hældningssensor	Værdien for hældningen af skibet angives i antal grader og i hvilken side skibet hælder.
5: Reguleringsstatus	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.
6: Forbindelser	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.
7: Automatisk reguleringsknap	Ved tryk på denne knap vil man komme til dialogen på figur 5.4.8 såfremt automatisk styring ikke er aktiveret. Hvis den er aktiveret og man trykker på knappen vil dialogen på figur 5.4.8 fremkomme.
8: Manuel reguleringsknap	Bringer dig til dialogen på figur 5.4.8
9: Aktivitetslog	Her udskrives vigtige hændelser i programmet med farvekoder. I dette eksempel kan det ses hvordan alarmer skrives med rødt og oprettede forbindelser skrives med grønt.

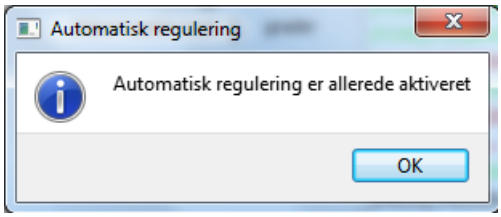
Manuel regulering af hældning

Ved tryk på knappen med teksten "AKTIVER MANUEL HÆLDNINGSREGULERING"(boks otte på figur 5.4.8) kommer dialogen på figur 5.4.8.

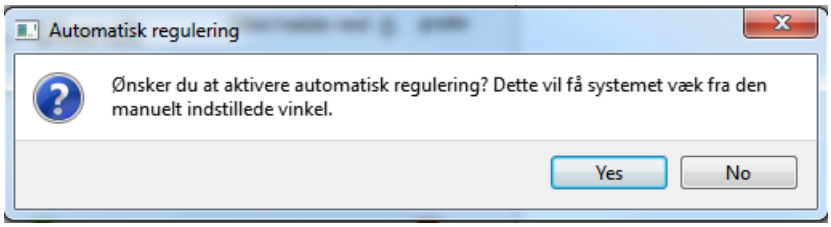
Figur 5.5. På figuren ses vinduet for manuel indstilling af vinkel

Automatisk regulering af hældning

Ved tryk på knappen med teksten "AKTIVER AUTOMATISK HÆLDNINGSREGULERING"(boks syv på figur 5.4.8) kommer en dialog frem. Såfremt automatisk hældnings allerede er aktiveret (som indikeret med en grøn cirkel øverst i boks fem på figur 5.4.8) fremkommer dialogen på figur 5.4.8 frem. Hvis automatisk hældning ikke er aktiveret fremkommer dialogen på figur 5.4.8 frem.



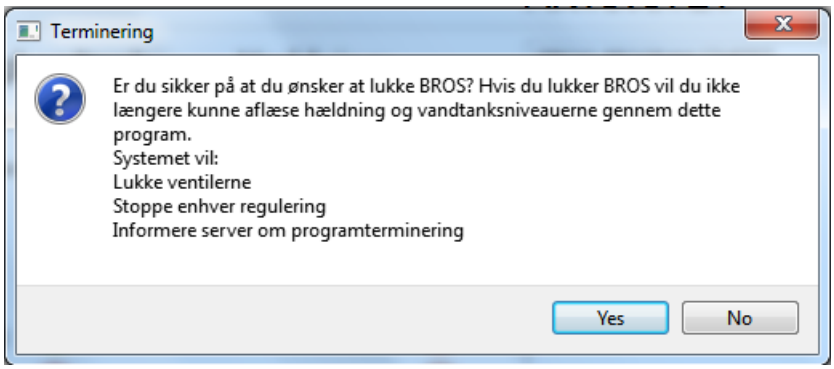
Figur 5.6. Ved tryk på AKTIVER AUTOMATISK HÆLDNINGSREGULERING når automatisk hældningsregulering allerede er aktiveret



Figur 5.7. Ved tryk på knappen i boks syv på figur 5.4.8 når automatisk hældningsregulering ikke er aktiveret

Termineringsdialog

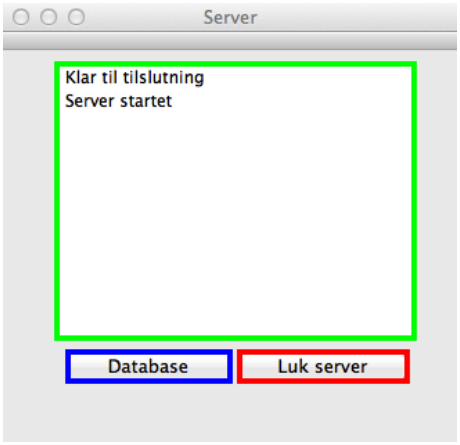
Denne advarsel fremkommer når man trykker på knappen i boks to på figur 5.4.8.



Figur 5.8. Advarsel før nedlukning af BROS

AppendixB: Database

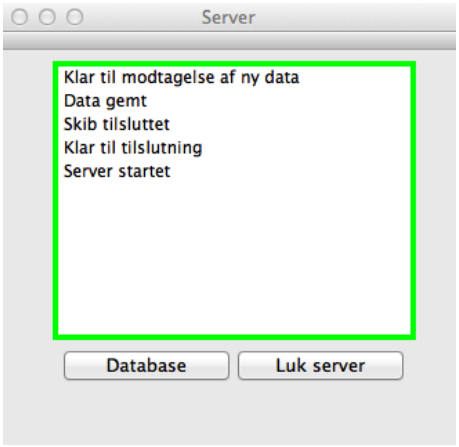
Hovedvindue for serveren



Figur 5.9. Billede af serverens hovedvindue

Serverens GUI giver mulighed for at brugeren kan se at serveren kører og hvilke kommandoer der er blevet udført. Fra serverens hovedmodul har brugeren mulighed for at vælge at åbne den webbaserede database og lukke severen.

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 fejlmeddelelser blive udskrevet her. Dialogboksen er lavet sådan at den nyeste besked altid står øverst. Se 5.10
2: Database	Anvendes til at lade brugeren tilgå den webbaserede database som ses i 5.12
2: Luk serveren	Anvendes til at lukke programmet. Denne er implementeret som et redskab for korrekt nedlukning, i stedet for at lukke på krydset. Tilmed kan man ved dette, give brugeren en advarsel, som kan sikre stabilt brug. Programmet åbner dialogen som ses i 5.11.



Figur 5.10. Billede af serverens hovedvindue

Luk Severen

Figur 5.11. Luk server

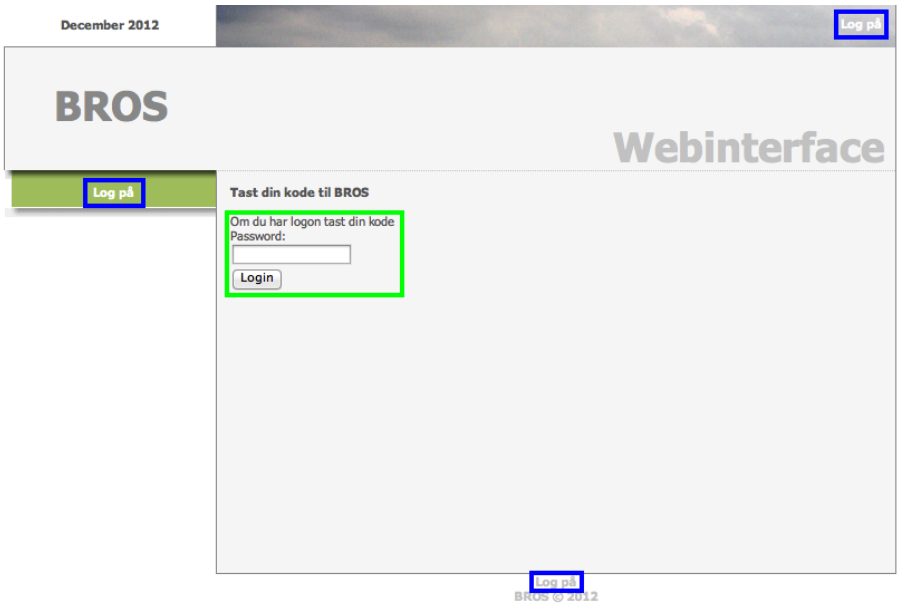
Ved tryk på "Luk server"vil figur 5.11 fremkomme og man kan trykke "Yes"for at lukke severen og "No"for at lade serveren fortsætte med at køre.

Lukke sekvens	
1: Info	Ved tryk på "No"vil brugeren returnere til 5.10 og fortsætte med at modtage data fra KI
2: Luk serveren	Ved tryk på "Yes"vil serveren blive lukket og brugeren kan ikke længere modtage data fra KI.

Webside

Log on

Den webbaserede adgang til databasen.

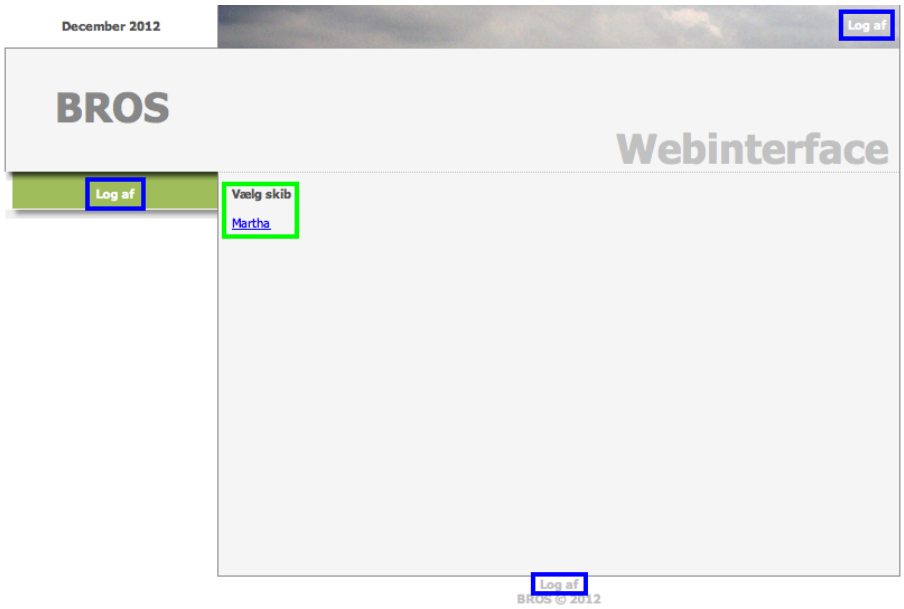


Figur 5.12. Billede af log in på den webbaserede database

Adgangkodekontrol for adgang til den webbaserede database	
1: Adgangskodekontrol	Når brugeren åbner databasen fra serveren eller fra en almindelig browser via internettet, vil denne komme til den webbaserede database. For at få adgang til databasen skal brugeren taste den adgangskode, som er blevet opgivet til denne.
1: Logpå/Log af	Når brugeren endnu ikke er logget på systemet, er disse tre knapper sat til Log på. Efter at brugeren er logget på vil disse knapper ændre sig til log af. Dette er brugerens bedste billede af hvor denne står på siden.

Skibsvalg

Efter log på kan brugeren vælge imellem de skibe, der er i dennes database. I dette tilfælde findes der kun et skib "Martha".

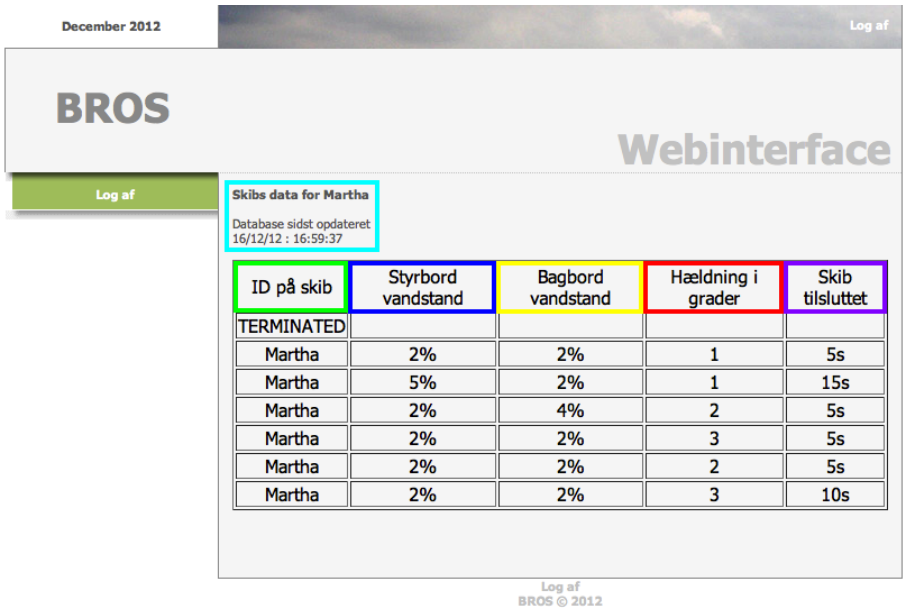


Figur 5.13. Billede efter log på, på databasen. Her kan brugeren vælge skib

Valg af skib, der skal tilgås	
1: Valg af skib	Efter at brugeren er logget på, som på 5.12, kan brugeren vælge imellem de skibe, som denne har adgang til fra sin databasedel. For havnekontoret ville dette normalt være alle skibe. I denne test version er der kun et skib, "Martha"
2: Log af	Efter at brugeren er logget ind, vil denne, på alle sider, have mulighed for at logge af. Dette sker ved at trykke på Log af. Efter at brugeren er logget af vil denne blive sendt tilbage til Log på vinduet.

Skibs data i database

Efter at brugeren har valgt skib, kan denne følge med i tilstanden på skibet. Denen side opdateres hver femte sekund hvis der er ny data.



Figur 5.14. Billede 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 database er opdateret.
2: ID på skib	Skibets ID bliver vist. Hvis KI lukker tcp-forbindelsen, vil skibets ID til "TERMINATED".
3: Styrbord vandtanksniveau	Der vises hvor meget vand, der er i styrbordballasttank.
4: Bagbord vandtanksniveau	Der vises hvor meget vand, der er i bagbordballasttank.
5: Hældning i grader	Der vises hvor mange grader skibet hælder, i forhold til styrbord.
6: Skib tilsluttet	Der vise hvor lang tid, i sekunder, der er gået fra overførelsen før til

MySQL

MySQL er en flertrådet SQL-database som understøtter mange samtidige brugere. MySQL er et open source program, som kan downloades på mysql.com og kan benyttes med mange forskellige operativsystemer som f.eks. Windows, Linux og MAC OS X Lion. Under projektet er mySQL blevet benyttet på Ubuntu(Linux version) og MAC OS X Lion. Tilgang til denne, er blevet gjort med den grafiske bruger grænseflade phpMyAdmin (webbaseret) og terminalen[1]

Datatyper

MySQL understøtter følgende datatyper[2]
INT -familien:
INT: Bruges udelukkende til heltal, som ikke indeholder mellemrum, linjeskift eller lignende.

[1] beskrevet under afsnittet: Kommandoer for adgang og brug af mySQL i terminal
[2] Kilde mysql.com

SmallINT: Fungere som INT, men bruges til små tal.
MediumINT: Fungere som INT, men bruges til mellemstore tal.
BigINT: Fungere som INT, men bruges til store tal.

Andre datatyper:
Varchar: Bruges til både tal, bogstaver og enkle tegn, en linje.
Char: Bruges udelukkende til bogstaver, en linje.
TinyText: Bruges til små, resumeer. Linjeskift er tilladt og alle former for tegn og bogstaver.
Text: Bruges til mellemlange tekster. Linjeskift er tilladt og alle former for tegn og bogstaver kan benyttes.
Longtext: Bruges til lange tekster. Linieskift er tilladt og alle former for tegn og bogstaver kan benyttes.
Decimal: Bruges udelukkende til decimaltal.
Date: Bruges udelukkende til at håndtere datoer. Datoformen skal være på dd-mm-år.

Kommandoer for adgang og brug af mySQL i terminal

For at kunne benytte mySQL med password og username skal dette opsættes. Dette er blevet gjort fra terminalen ved at åbne terminalen og skrive mysql. Dette vil logge en på første gang. For oprettelse af bruger, f.eks. root, gøres følgende:
mysql> use mysql;
mysql> update user set password=PASSWOD("NEWPASSWORD") where User='root';
mysql> flush privileges;
mysql> quit

root er nu sat med password. For at logge på med root benyttes: mysql -u root -p'password' herfra er følgende kommandoer muligt:

Kommandoer	Beskrivelse
show databases;	Viser alle databaser, der er tilgængelige for den bruger, der er logget på databasen(f.eks. her ship)
create database 'navn på database'	Opretter en database
use 'database';	For at vælge en bestemt database, f.eks.: use ship;
show tables;	Viser tabeller i databasen
create table 'navn på tabel'('row name' char(20), 'row lastname' char(20));	Opretter en tabel med et valgt navn('navn på tabel'). Tabellen skal vide hvor mange kolonner der skal oprettes. Her er en med name og en char længde på 20 og en lastname med en char på 20.
DESCRIBE 'tabel navn'	viser den oprettede tabel
select * from 'tabel'	Viser indholdet af en tabel
INSERT INTO 'tabel navn'('Indhold første row', 'indhold anden row');	Indsætter data i den oprettede tabel under de to kolonner
drop table 'tabel navn';	Sletter tabellen
drop database 'navn';	Sletter databasen
quit	Logger af mySQL

Tabel 5.1. Tabel over basale kommandoer i mySQL

Rettelser