CCSystem

SYSTEMARKITEKTUR

Gruppe 2:

#1 Stud.nr.: 09008	Navn: Flemming Thesbjerg (FT)
#2 Stud.nr.: 09753	Navn: Bjarne Møller Dideriksen (BD)
#3 Stud.nr.: 10228	Navn: Dennis Kristensen (DK)
#4 Stud.nr.: 10477	Navn: Henrik Vorsø Jachobsen (HJ)
#5 Stud.nr.: 10700	Navn: Michael Juhl (MJ)
#6 Stud.nr.: 10796	Navn: Anders Kielsholm (AK)
#7 Stud.nr.: 10802	Navn: Kasper Hedegård Munck (KM)
#8 Stud.nr.: 10857	Navn: Thomas Hoffmann Jespersen (TJ)
Dato:	***Godkendt af:***

1. Indholdsfortegnelse

1. Indholdsfortegnelse 2
2. Introduktion 3
2.1. Formål og omfang 3
2.2. Referencer 3
2.3. Definitioner og forkortelser 3
2.4. Dokumentstruktur 4
2.5. Dokumentets rolle i en iterativ udviklingsproces. 4
3. System oversigt 5
3.1. Dokumentets rolle i en iterativ udviklingsproces. 5
3.2. System introduktion 7
4. Systemets grænseflader 8
4.1. Grænseflader til person aktører 8
4.2. Grænseflader til hardware aktører 8
5. Use Case View 9
5.1. Oversigt over arktiktursignifikante Use Cases 9
6. Logisk View 10
6.1. Oversigt over arktiktursignifikante Use Cases 10
6.2. Arkitektursignifikante designpakker 11
6.3. Samlet klassediagram 21
6.4. Use Case realiseringer 21
7. Proces/task View 22
7.1. Oversigt over tråde 22
7.2. Trådgruppe – Driver Interface-pakke 23
7.3. Trådgruppe – Business Layer-pakke 24
7.4. Trådgruppe – GUI-pakke 25
8. Deployment View 25
9. Implementerings View 25
10. Generelle designbeslutninger 26
10.1. Arkitektur mål og begrænsninger 26
10.2. Arkitektur mønstre 26
10.3. Generelle brugergrænsefladeregler 26
10.4. Exception og fejlhåndtering 26
10.5. Implementeringsværktøj 27
11. Kvalitet 28
12. Figuroversigt 29

2. Introduktion

2.1. Formål og omfang

Dette dokument fastlægger den samlede arkitektur for en Cruise Controller kaldet CCSystem. Dokumentet er baseret på kravspecifikationen og indeholder designet for systemet uden at gå i detaljer med implementeringen.

Dokumentet gør det muligt for andre projektgrupper at fortsætte udviklingen eller opdatere systemet.

Dokumentationen kan læses uden kendskab til kravspecifikationen, og kan bidrage til at forstå designet af CCSystemet.

2.2. Referencer

- [1] Larman, C.: Applying UML and Patterns. An Introduction to Object-Oriented Analysis and Design and Iterative Development
- [2] Kravspecifikation for CCSystemet
- [3] Philippe Kruchten,"The 4+1 View of Architecture,"IEEE Software, 12(6) Nov. 1995
- [4] Implementeringsdokumentationen for CCSystemet kapitel 5.1.4 – Beskedsystemet.

2.3. Definitioner og forkortelser

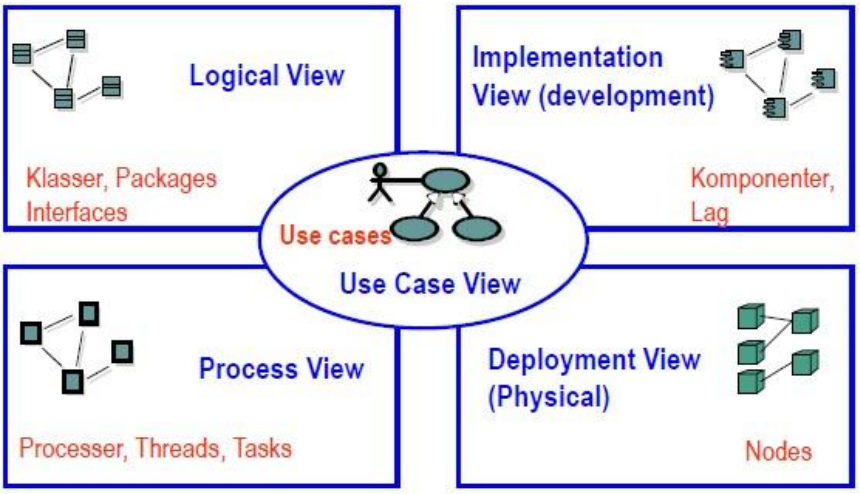
CCSystem: Produktets navn

GUI: Grafisk brugergrænseflade

2.4. Dokumentstruktur

Dokumentet er struktureret efter 4+1 modellen. Det betyder at den er bygget over fire forskellige måder at se en Use Case på.

Figur 1 - "4+1" View Model [3] herunder giver et overblik over 4+1 modellen:



Figur 1 - "4+1" View Model [3]

2.5. Dokumentets rolle i en iterativ udviklingsproces.

Udviklingen af systemet vil foregå gennem 4 iterationer, hvor der undervejs vil blive arbejdet iterativt på dokumentet. I iterationerne vil der blive arbejdet med følgende:

- Kravspecifikation, hvor der udarbejdes Use Cases.
- Design af Use Cases:
  - Sekvensdiagrammer
  - Klassediagram
- Implementering af Use Cases.
- Deltest af Use Cases.
- Samlet accepttest.
- Udarbejdelse af projektdokumentation og projektrapport.

3. System oversigt

Dette afsnit giver et overordnet billede af systemet og dets omgivelser.

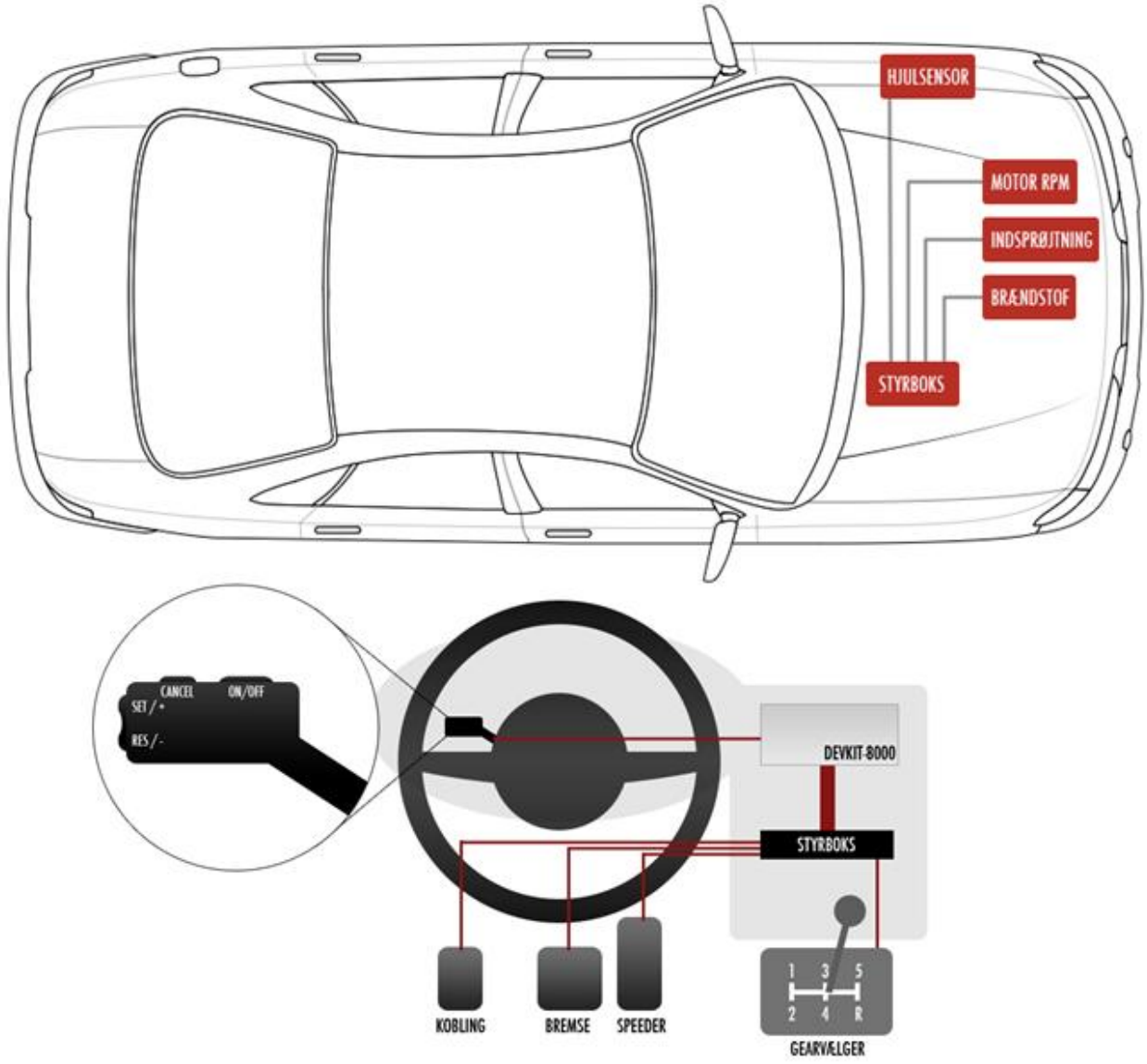
3.1. Dokumentets rolle i en iterativ udviklingsproces.

Systemet har to mål, hvor det primære mål er at fastholde en hastighed for et køretøj vha. en fartpilot. Det sekundære mål er at videregive specifikke værdier om bilen til brugeren, ved hjælp af en brugergrænseflade.

Dette projekt koncentrerer sig udelukkende om softwaredelen, selve hardwaredelen er leveret af ATOYOT A/S i form af en testsimulator.

3.1.1. Systemoversigt

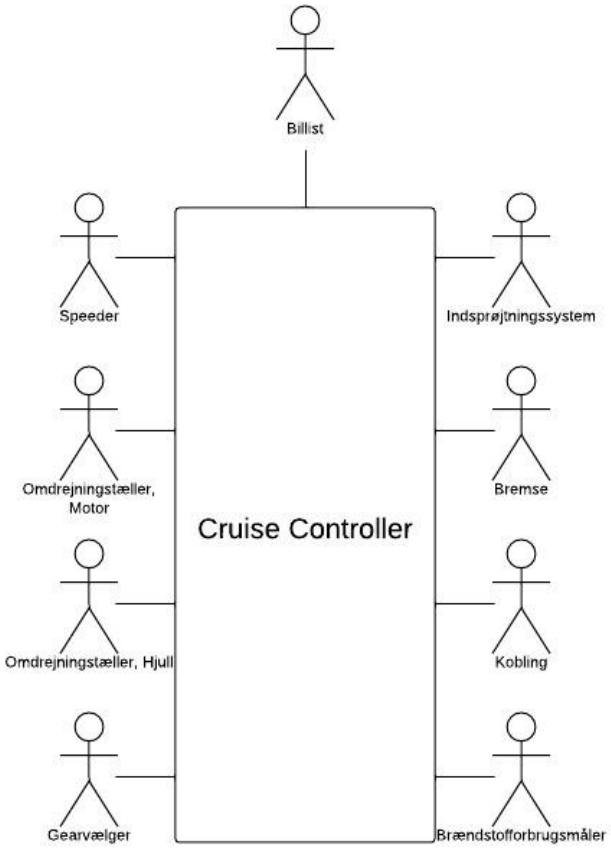
Fejl! Henvisningskilde ikke fundet. giver et billede af det overordnede system. Brugeren styrer systemet ved hjælp af en betjeningsarm og får diverse informationer via et display på DEVKIT-8000. Styreboksen modtager signaler fra sensorer på henholdsvis kobling, bremse, speeder, motoren, hjul og gearvælger.



Figur 2 - Systemoversigt

3.1.2. Aktør-kontekst diagram

Figur 3 - Aktør-kontekst diagram viser hvilke påvirkninger CCSystemet får fra omverdenen. Systemet har tre primære aktøre: Bilist som er brugeren af systemet, Bremse og Kobling som kan starte en funktionalitet i systemet. De sekundære aktører er: Speeder, Omdrejningstæller Motor, Omdrejningstæller Hjul, Gearvælger, Brændstofforbrugsmåler og Indsprøjtningssystem.



Figur 3 - Aktør-kontekst diagram

3.2. System introduktion

Det primære mål med systemet er at give billisten mulighed for at fastholde en given hastighed uden manuel påvirkning. Fartpiloten kan aktiveres, deaktiveres, genoptage tidligere hastighed og reguleres gennem en betjeningsarm. GUI'en skal samtidig give overblik over hastighed, motoromdrejning, triptæller, aktuelle gear og kilometertæller. CCSystemet gør det nemmere at køre økonomisk korrekt, da den altid vil bidrage med en "gear-anbefaling" ud fra motorens omdrejningstal.

4. Systemets grænseflader

4.1. Grænseflader til person aktører

CCSystemet har kun en personaktør, billist.
Grænsefladen består af en touch-skærm og en betjeningsarm. Betjeningsarmen giver billisten mulighed for at styre fartpiloten, og touch-skærmen viser diverse værdier for bilen, giver billisten mulighed for at nulstille triptælleren og kalibrere systemet. Se brugervejledning for yderlige information.

4.2. Grænseflader til hardware aktører

4.2.1. Gear

Der leveres følgende signaler fra GPIO2 (på Devkit 8000-Addon Board).

GPIO2 bit	**0**	**1**	**2**	**3**	**4**	**5**
Bakgear	1	0	0	0	0	0
1. Gear	0	1	0	0	0	0
2. Gear	0	0	1	0	0	0
3. Gear	0	0	0	1	0	0
4. Gear	0	0	0	0	1	0
5. Gear	0	0	0	0	0	1
Neutral	0	0	0	0	0	0

4.2.2. Speeder

Speedersignalet leverer et analogt signal (0-10V). Signalet er 0, når pedalen er sluppet og er ellers en spænding der er proportional med pedalens tryk. Leveres på AIN0.

4.2.3. Flowmåler

Flowmåleren angiver hvor meget brændstof motoren får tilført. Flowmåleren giver et analogt signal (0-10V), der proportional med brænstofflowet. Leveres på AIN1.

4.2.4. Bremse og Koblings-pedal

Bremse- og koblingspedal sensorerne leverer hver et digitalt signal(0-5V), der er logisk 1 når pedalen aktiveres af føreren, og logisk 0 når pedalen er sluppet. Koblingspedal: GPIO1, bit 0. Bremsepedal: GPIO1, bit 1.

4.2.5. Hjulomdrejningsmåler

Hjulsensoren giver 2 pulser pr. hjulomdrejning svarende til 40 Hz for maksimal hastighed.
Hjulomdrejningsmåleren leveres på GPIO1, bit 3.

4.2.6. Motoromdrejningsmåler

Motoromdrejningsmåleren leverer pulser fra 0-3 kHz proportional med omdrejningstallet.
Motoromdrejningsmåleren leverer på GPIO1, bit 4.

4.2.7. Indsprøjtningssystem

Den sidder parallelt med den normale speeder og giver en lineær bevægelse styret af aktuatorens input signal. Aktuatoren skal gives en analog spænding 0-10 V(max 8mA). Sendes til AOUT0.

5. Use Case View

5.1. Oversigt over arktiktursignifikante Use Cases

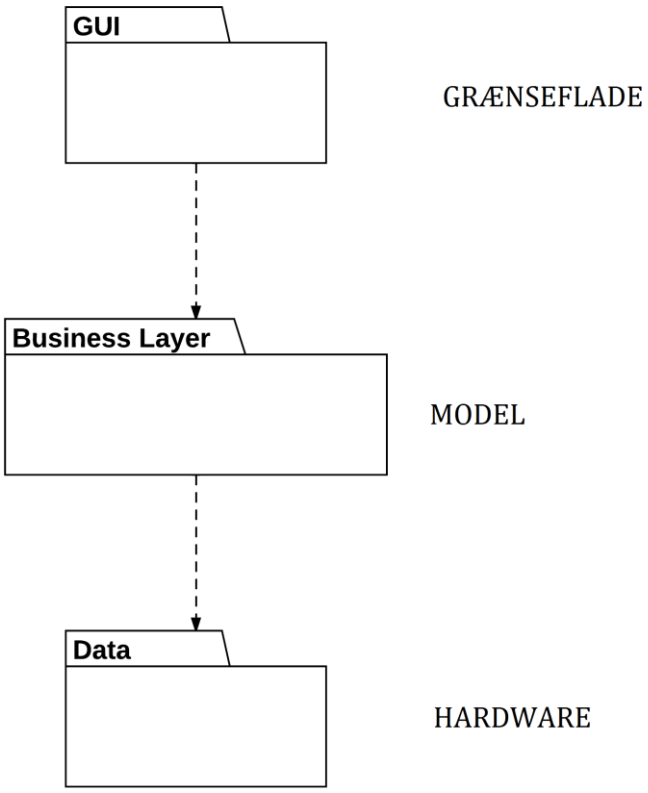
Se Kravspecifikation for CCSystemet. [2]

6. Logisk View

Dette kapitel beskriver systemets opdeling i delsystemer.

6.1. Oversigt over arktiktursignifikante Use Cases

CCSystemet er opdelt i 3 hovedpakker: GUI(Grænseflade), Business Layer(Model), Data(Hardware Access).



Figur 4 - Oversigt over pakker

GUI-pakken: Pakken sørger for den grafiske repræsentation af data og nuværende status for CCSystemet.

Business Layer-pakken: Pakken indeholder CCSystemets funktionalitet for fartpiloten.

Data-pakken: Pakkens hovedansvar er at hente data fra filer som indeholder værdier for diverse hardware enheder. Værdierne er gjort tilgængelige via device drivers. Data-pakken er også ansvarlig for at gøre værdierne tilgængelig for de øvrige lag.

De enkelte pakker er detaljeret beskrevet i næste kapitel.

6.2. Arkitektursignifikante designpakker

Dette afsnit beskriver pakkerne som indgår i systemet og giver en kort beskrivelser af deres klasser.

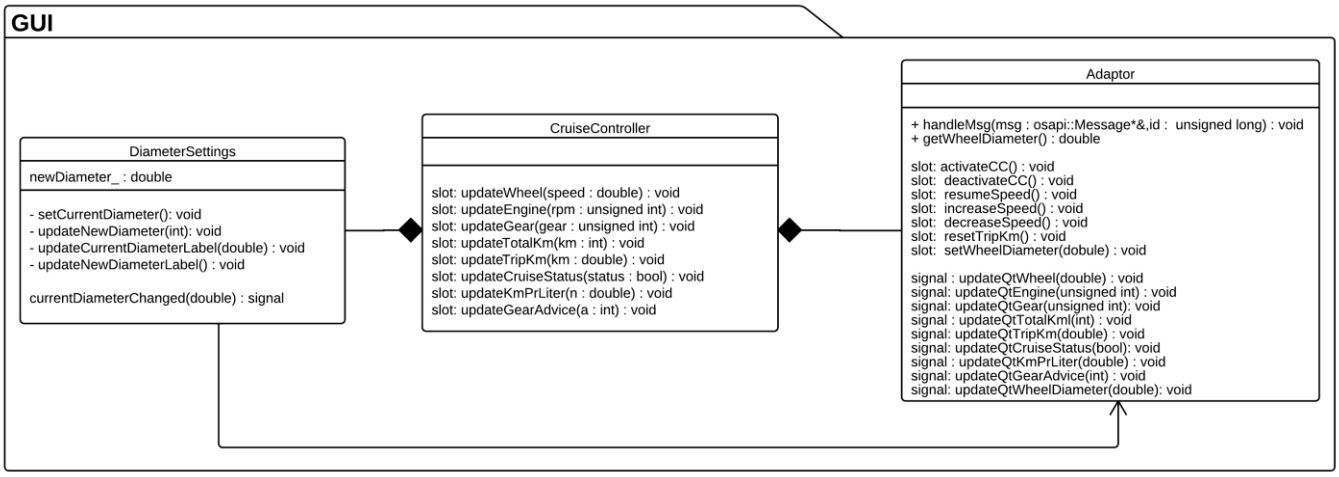
Beskrivelser af de enkelte klassers funktioner kan ses i koden (Se Bilag "03_Sourcekode")

6.2.1. Pakke 1: GUI

Ansvar:

GUI-pakken står for den grafiske repræsentation af data og behandling af input fra billisten. Pakken opdateres ved en ændring i Data-pakken.

Klassediagram:



Figur 5 - Klassediagram over GUI-pakken

DiameterSettings

Klassen giver mulighed for at ændre hjuldiameteren.

CruiseController

Klassen har ansvaret for at opdatere diverse værdier på brugergrænsefladen, samt at modtage input fra billisten.

Adaptor

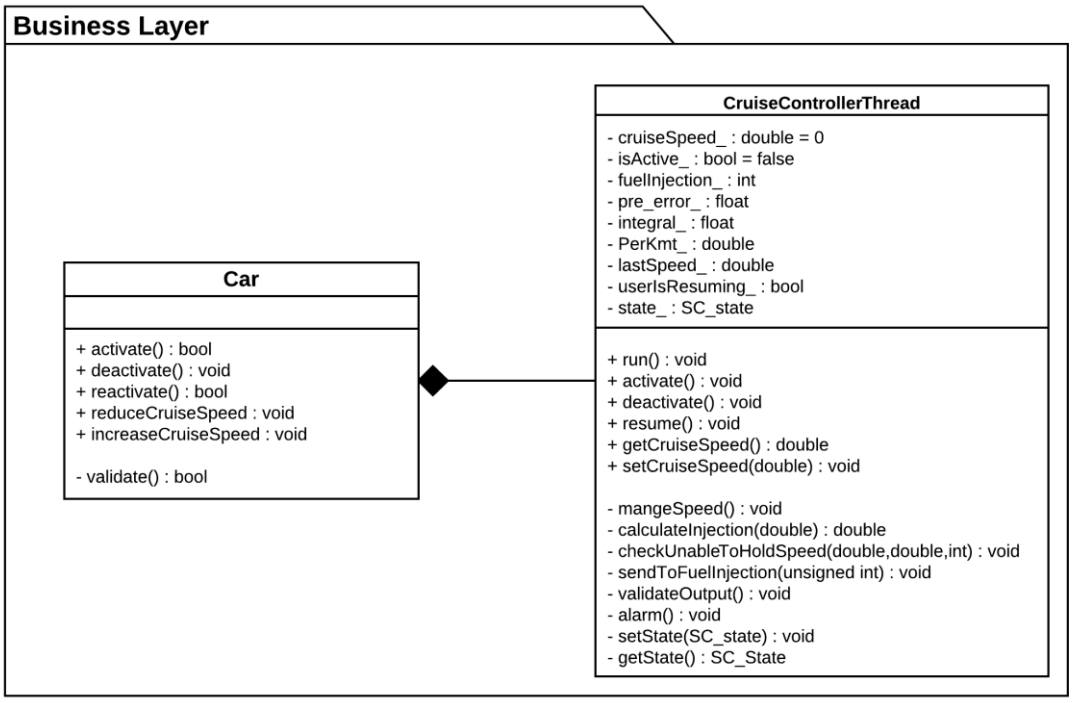
Klassen håndterer kommunikationen med Business Layer-pakken og Data Container-pakken. Den modtager besked om dataændringer fra et beskedsystem (beskrevet i implementeringsdokumentation kapitel 5.1.4), og kan gennem singleton objektet af klassen Car aktivere, deaktivere, resume, regulere hastighed, nulstille triptæller og kalibrere hjulets diameter.

6.2.2. Pakke 2: Business Layer

Ansvar:

Business Layer-pakken er hovedsageligt ansvarlig for styring af fartpiloten, og styrer hvordan Data-pakkens værdier ændrer fartpiloten.

Klassediagram:



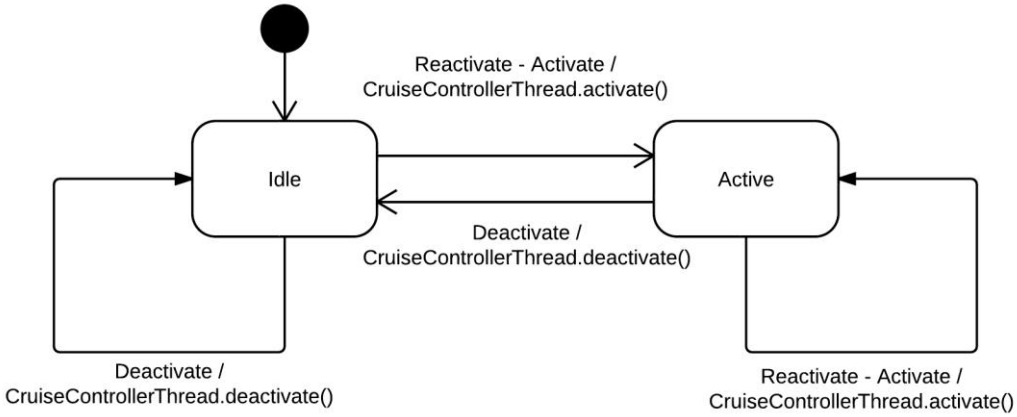
Figur 6 - Klassediagram over Business Layer-pakken

Car

Klassen står for at styre fartpilotens funktioner ud fra ændringer i Data-pakkens værdier fra diverse hardware enheder. Den skal blandt andet deaktivere fartpiloten hvis billisten kobler ud eller bremser.

CruiseControllerThread

Klassen står for fartpilotens funktionalitet og indeholder en tilstandsmaskine med to tilstande. Den kan enten stå og vente på at billisten aktiverer fartpiloten, eller den kan være aktiv hvor den regulerer på indsprøjtninssystemet, og derved holde hastigheden stabil. Klassen kan beskrives ud fra tilstandsdiagrammet på



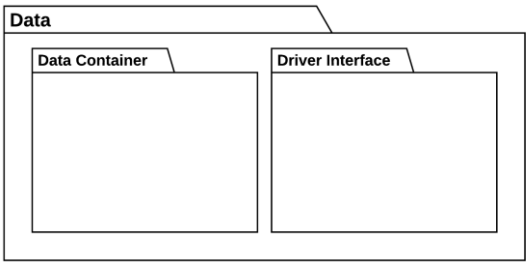
Figur 7 - Tilstandsdiagram for CruiseControllerThread

6.2.3. Pakke 3: Data

Ansvar:

Data-pakken består af at hente/skrive data fra hardware-enheder og give besked til Business Layer-pakken og GUI-pakken ved brug af et beskedsystem [4].

Klassediagram:



Figur 8 - Klassediagram over Data-pakken

Pakken består af to pakker: *Data Container* og *Driver Interface*. Disse pakker og deres tilhørende klasser er beskrevet herunder:

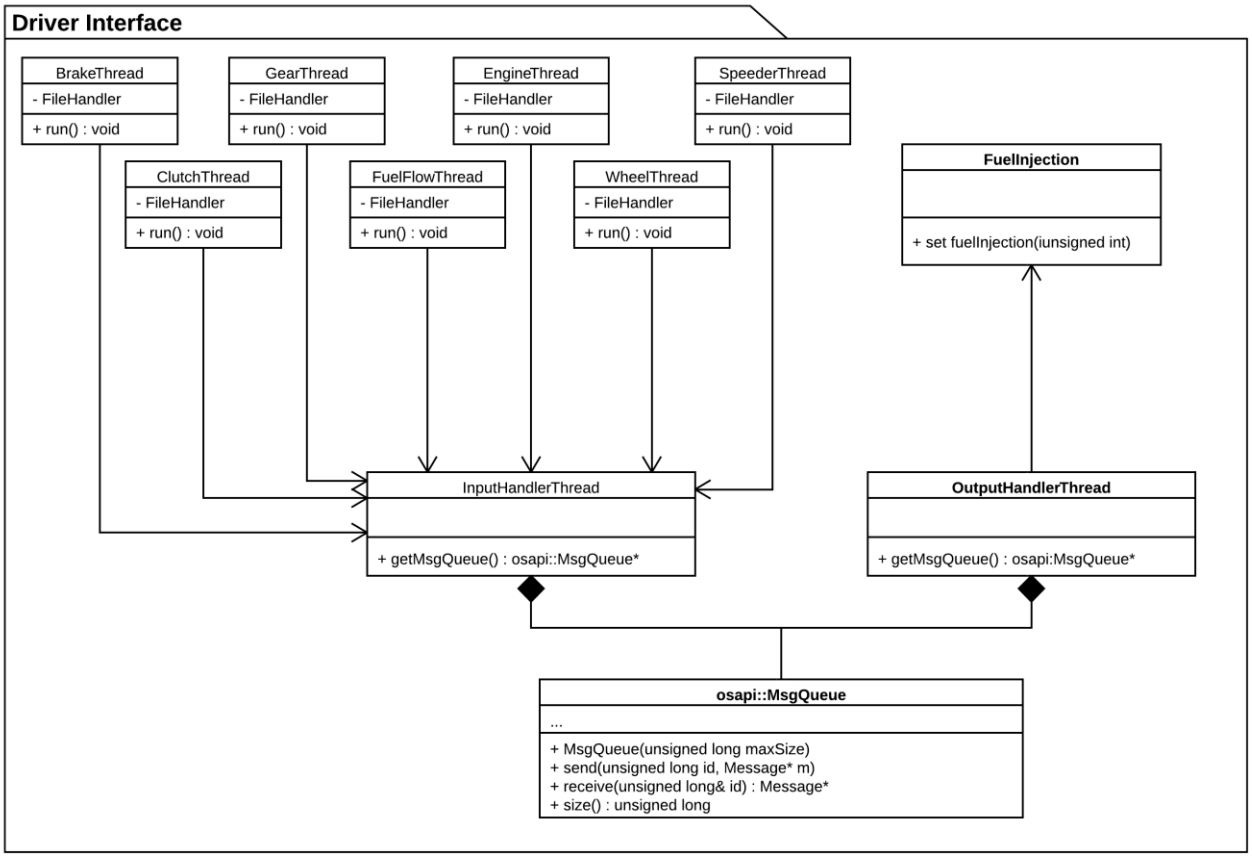
6.2.4. Pakke 4: Driver Interface

Ansvar:

Driver Interface sørger for tilgangen til hardwaren. Da man ikke kan tilgå hardware direkte fra user-space, benytter Driver Interface-pakken forskellige filer, som er oprettet af diverse Device Drivers, som står for at gøre hardwarens værdi tilgængelig for user-space.

Driver Interface-pakken består af diverse tråde som sender den nyeste hardware værdi til en MessageHandler, som sender værdierne videre til den korresponderende klasse i Data Container-pakken.

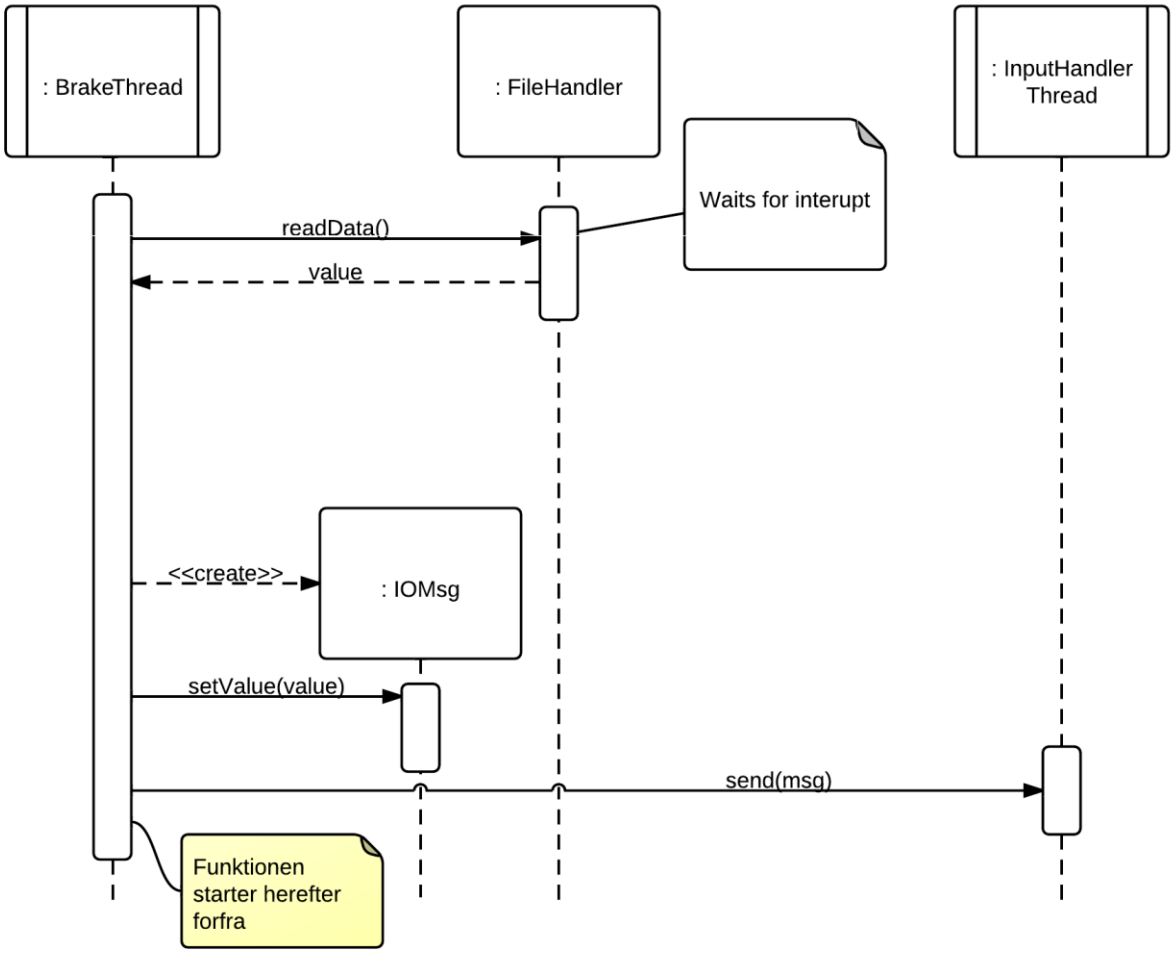
Klassediagram:



Figur 9 - Klassediagram over Driver Interface-pakken

BrakeThread, ClutchThread

Henter værdier og sender dem til InputHandlerThread ved en data ændring. Se Figur 10 - Sekvensdiagram over BrakeThread.



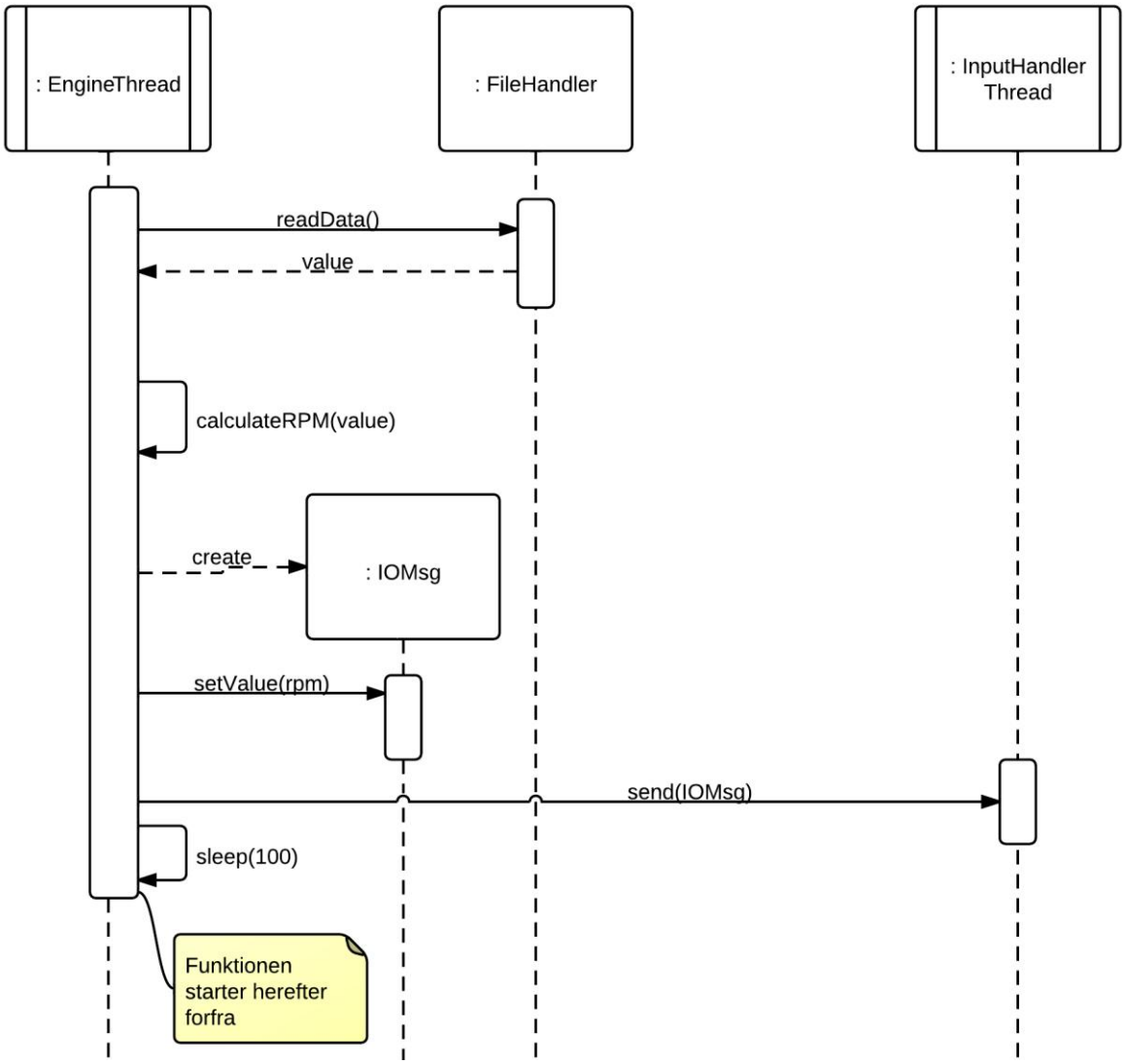
Figur 10 - Sekvensdiagram over BrakeThread

GearThread, FuelFlowThread, WheelThread, SpeederThread

Henter værdier og sender dem til InputHandlerThread cirka 10 gange i sekunder.

EngineThread

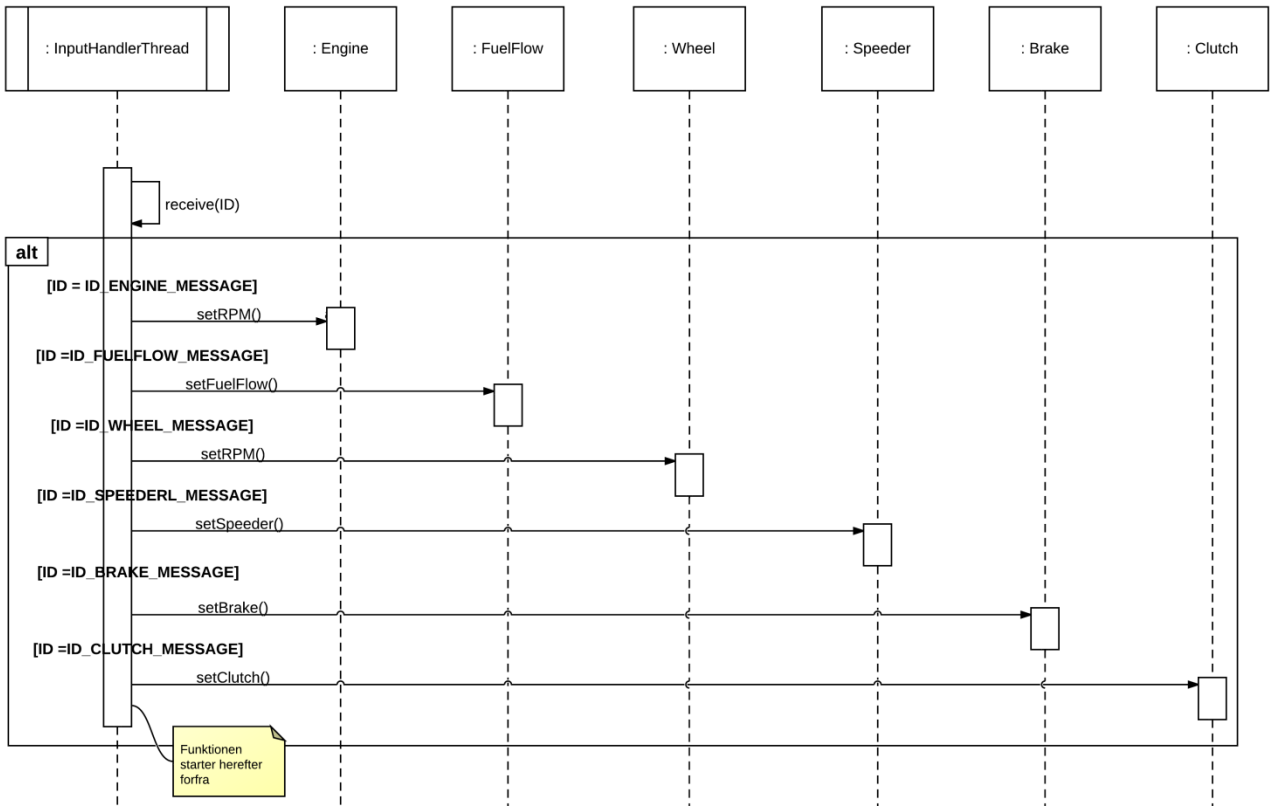
Henter værdi, og foretager en omregning og sendes til InputHandlerThread, men en forsinkelse på 100 ms efter hvert gennemløb. Se Figur 11 - Sekvensdiagram over EngineThread.



Figur 11 - Sekvensdiagram over EngineThread

InputHandlerThread

Modtager beskeder og ud fra beskedens ID sættes en klasses værdi i Data Container-pakken. Se Figur 12 - Sekvensdiagram over InputHandlerThread.



Figur 12 - Sekvensdiagram over InputHandlerThread

OutputHandlerThread

Modtager beskeder fra Business Layer-pakken og ud fra beskedens ID sættes FuelInjection.

FuelInjection

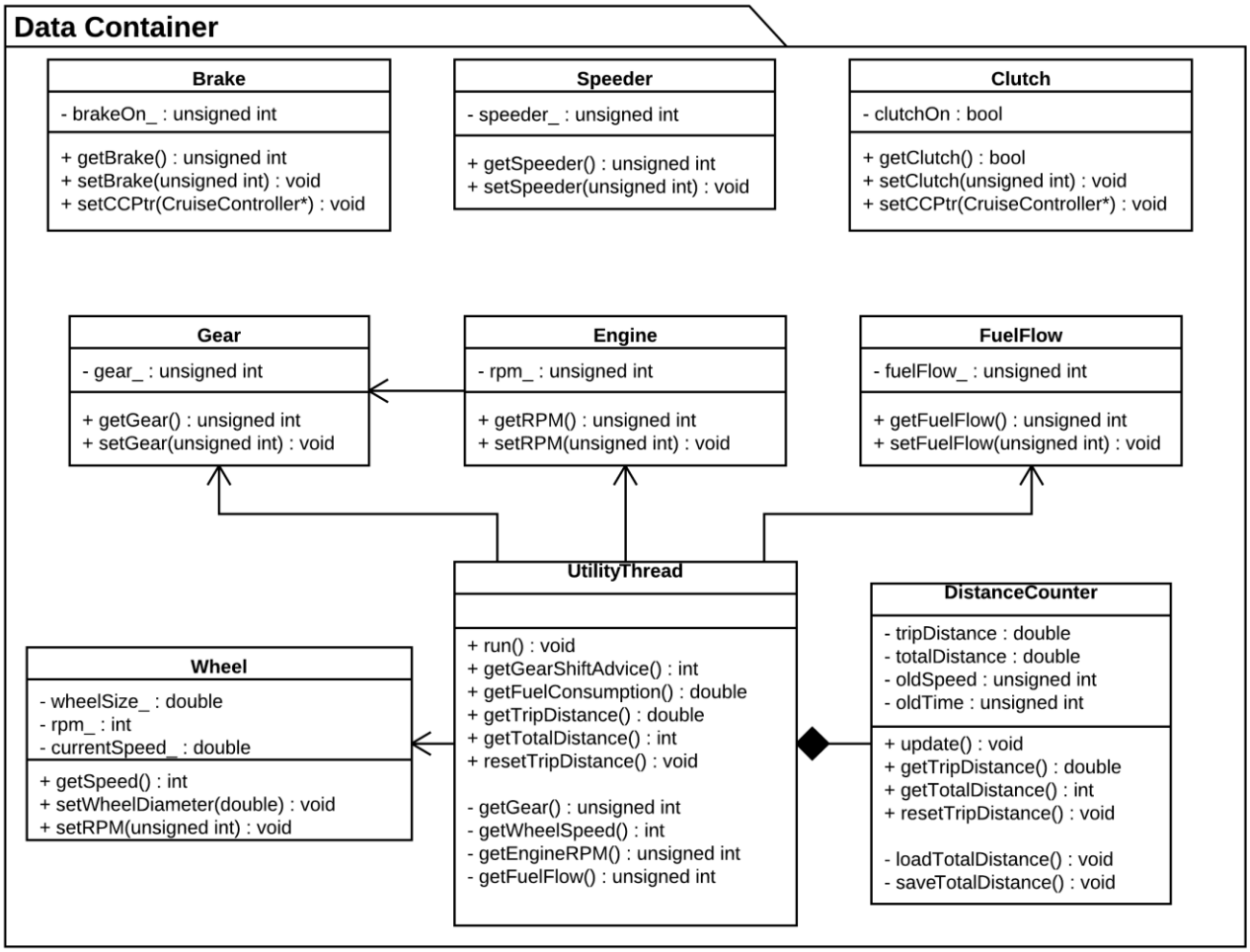
Giver muligheden for at skrive en værdi til Indsprøjtningssystemet og derved ændre værdien for indsprøjtningssystemet.

6.2.5. Pakke 5: Data Container

Ansvar:

Pakkens primære ansvar er at indeholde data, behandle data, give besked til Business Layer-pakken og GUI-pakken når ny data er tilgængeligt, hvilket foregår gennem et OSApi'ets beskedsystem (beskrevet i implementeringsdokumentation kapitel 5.1.4).

Klassediagram:



Figur 13 - Klassediagram over Data Container-pakken

Overordnet for Speeder, Engine, Brake, Gear, FuelFlow, Clutch, Wheel

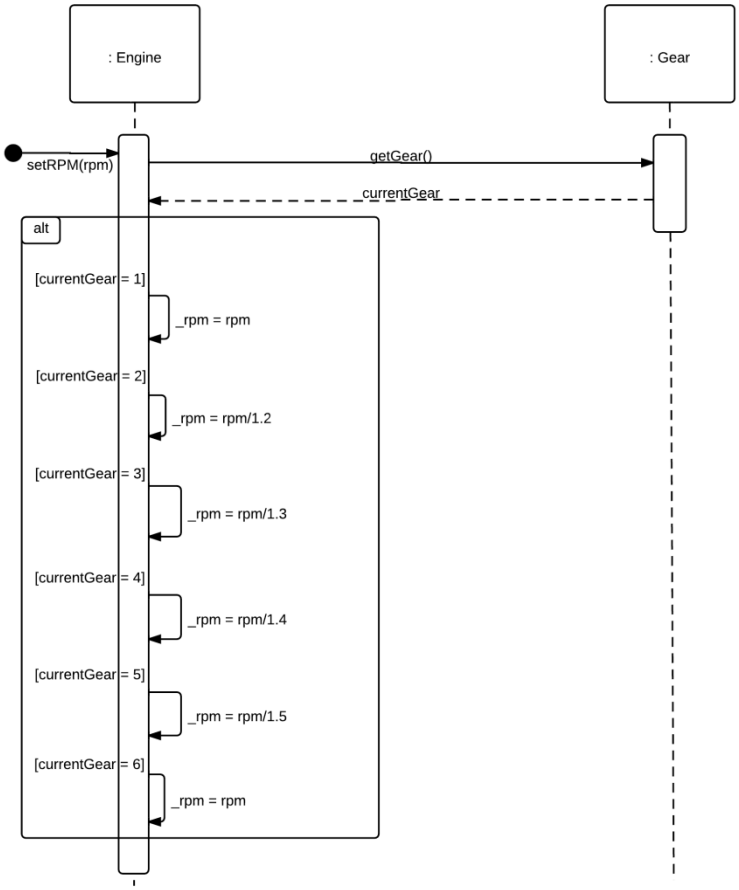
Klasserne modtager alle deres værdier fra Driver Interface-pakken. Se eksempel på Figur 12 - Sekvensdiagram over InputHandlerThread

Speeder

Indeholder speederpedalens niveau udtrykt som en værdi mellem 0 og 1000 promille.

Engine

Indeholder antal motoromdrejninger. Grundet hardwarens begrænsninger, beregner klassen motorens omdrejninger ud fra gearposition. Se Figur 14 - Sekvensdiagram over Engine::setRPM.



Figur 14 - Sekvensdiagram over Engine::setRPM

Brake

Indeholder værdi for bremsen. 0 betyder at bremsen ikke er aktiveret, 1 betyder den er aktiveret.

Wheel

Indeholder bilens hastighed, samt hjulets diameter. Den modtager RPM værdier fra Driver Interface pakken, og ud fra denne beregnes en nyere hastighed.

Gear

Klassen indeholder nuværende gearposition. Mulig værdi (0-6). Se Bilag ”04_DriverInterface”.

FuelFlow

Indeholder brændstof-flowet, udtrykt som en værdi mellem 0-1000 promille.

Clutch

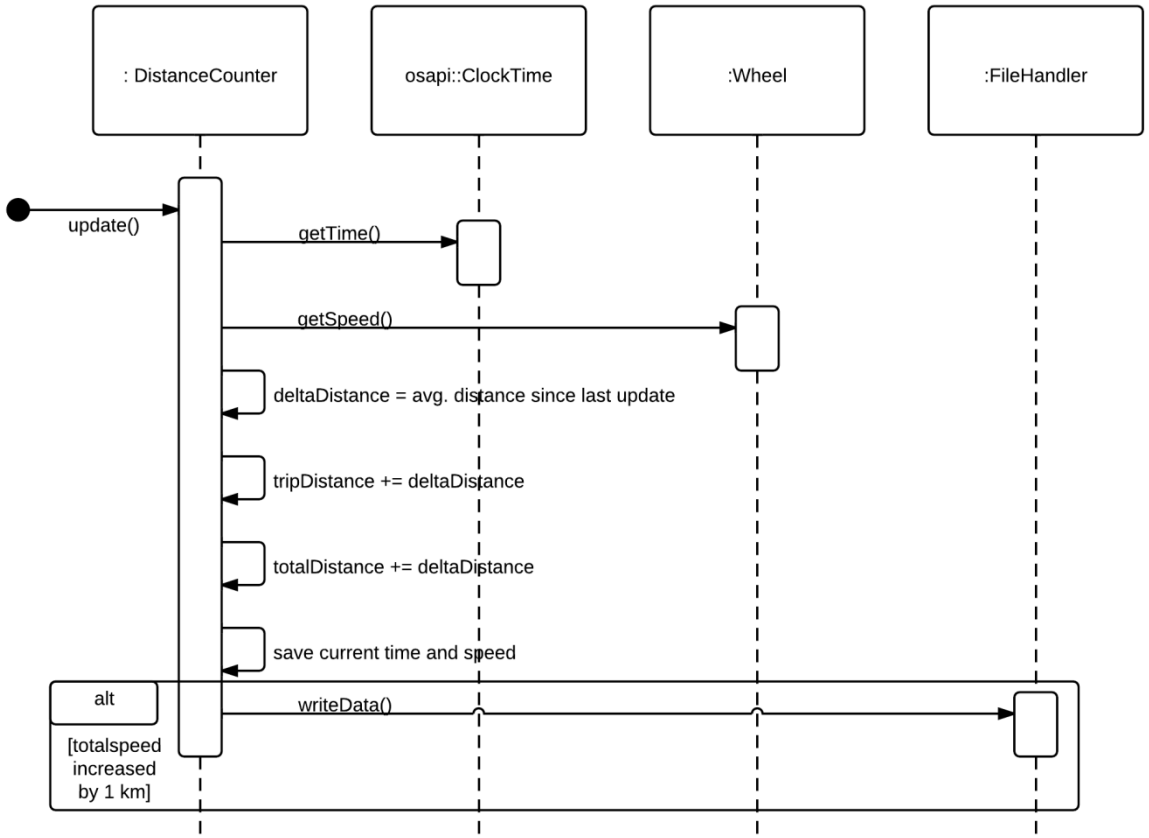
Indeholder værdi for kobling. 0 betyder at koblingen ikke er trykket ned, 1 betyder der er koblet ud.

UtilityThread

Opdaterer værdierne for kilometertæller, triptæller, brændstofforbrug samt hvorvidt bilisten skal skifte gear.

DistanceCounter

Beregner antal kilometer kørt. Se Figur 15 - Sekvensdiagram over DistanceCounter::update(), som opdaterer og indeholder antal kilometer og triptælleren.



Figur 15 - Sekvensdiagram over DistanceCounter::update()

6.3. Samlet klassediagram

Det samlede udetaljerede klassediagram kan ses i bilag 07_Klassediagram.

6.4. Use Case realiseringer

6.4.1. Aktiver Fartpilot Use Case

Fartpiloten aktiveres vha. af betjeningsarmen, som herefter videregiver brugerinput til Business Layer-pakken. Car-klassen aktiverer herefter CruiseControllerThread, som starter med at holde den nuværende hastighed ud fra hastigheden og speeder niveau fra Data Container-pakken, og tilfører mere/mindre brændstof ved brug af Driver Interface-pakken. Aktiver fartpilot er illustreret ved sekvensdiagrammet som findes i bilag " 08_Use Case Realiseringer – Sekvensdiagrammer".

6.4.2. Deaktiver Fartpilot Use Case

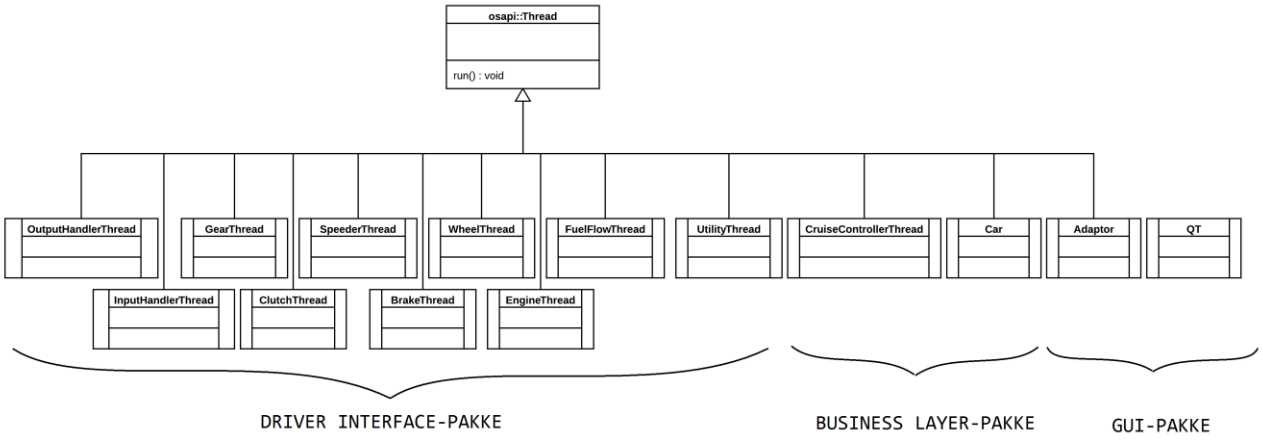
Fartpiloten deaktiveres vha. betjeningsarmen eller igennem Driver Interface-pakken. Deaktiver fartpilot er illustreret ved sekvensdiagrammet som findes i bilag " 08_Use Case Realiseringer – Sekvensdiagrammer".

6.4.3. Reaktiver Fartpilot Use Case

Reaktiver fartpilot undersøger hvorvidt der har været aktiveret før, hvis den ikke har det skal der ikke ske noget. Reaktiver fartpilot er illustreret ved sekvensdiagrammet som findes i bilag " 08_Use Case Realiseringer – Sekvensdiagrammer".

7. Proces/task View

7.1. Oversigt over tråde



Figur 16 - Oversigt over tråde

Figur 16 - Oversigt over tråde angiver de aktive klasser. De er implementeret ved at nedarve fra en abstrakt klasse fra vores OSApi'et, som hedder Thread. Det ses på det samlede klassediagram hvordan trådene indgår i den overordnede model. I GUI-pakken er dog en Qt-klasse som ikke nedarver fra OSApi'et, men derimod benytter sig af QT frameworkets interne trådsystem.

Trådene, som indgår i Driver Interface pakken, skal holde systemet opdateret med de nyeste værdier fra deres korresponderende hardware.

Business Layer-pakkens tråde skal håndtere fartpiloten, samt input fra GUI- og Data-pakken.

Vores GUI er opbygget ved brug af Qt, og denne benytter tråde til at undersøge hvorvidt der input fra bruger.

Alle tråde startes op når CCSystemet startes op, og nedlægges først når CCSystemet slukkes.

7.2. Trådgruppe – Driver Interface-pakke

Trådgruppe 1 dækker over alle tråde som indgår i Driver Interface pakken.

7.2.1. Kommunikation mellem trådene i Driver-Interface pakken.

Hardware trådene kommunikerer med InputHandlerThread som har en beskedkø fra OSApi'et.

7.2.2. BrakeThread, ClutchThread

Beskrivelse: Ansvarlig for at sende beskeder til InputHandlerThread, hvis værdien for kobling/bremse ændres.
Aktiv: Trådene aktiveres når bremse/koblingens værdi ændres.

7.2.3. EngineThread, WheelThread, SpeederThread, FuelFlowThread, GearThread

Beskrivelse: Hver tråd er ansvarlig for at opsamle data fra diverse hardware enheder.
Aktiv: Trådene aktiveres med 100 ms. forsinkelse efter hvert gennemløb.

7.2.4. InputHandlerThread

Beskrivelse: Ansvarlig for at modtage beskeder fra drivertrådene vha. dens MsgQueue og ud fra beskedens ID, sættes en værdi af en korresponderendes klasse i Data-pakken.
Aktiv: Tråden aktiveres for hver besked den modtager i dens MsgQueue.

7.2.5. OutputHandlerThread

Beskrivelse: Ansvarlig for at modtage beskeder fra Business Layer-pakken, og sætte værdien i klassen FuelInjection.
Aktiv: Tråden aktiveres for hver besked den modtager i dens MsgQueue.

7.2.6. UtilityThread

Beskrivelse: UtilityThread kalder løbende DistanceCounter's update-funktion således dens data er opdateret.
Aktiv: Tråden aktiveres med 10 ms. forsinkelse efter hvert gennemløb.

7.3. Trådgruppe – Business Layer-pakke

Trådgruppe 2 dækker over de to tråde som indgår i Business Layer-pakken.

7.3.1. Kommunikation mellem trådene i Business Layer-pakken

Car-tråden sætter diverse attributter i CruiseControllerThread, og disse beskyttes af mutexes fra OSApi'et.

7.3.2. Car

Beskrivelse: Behandler input fra GUI-pakken og venter på data ændringer på kobling og bremse.

Levetid: Oprettes når CCSystemet startes og nedlægges når CCSystemet slukkes.

Aktiv: Tråden aktiveres for hver besked den modtager i dens MsgQueue.

7.3.3. CruiseControllerThread

Beskrivelse: Tråden sørger for al funktionalitet af fartpiloten.

Levetid: Oprettes når CCSystemet startes og nedlægges når CCSystemet slukkes.

Aktiv: Tråden er kun aktiv når fartpiloten er aktiv, og i denne er den aktiveret med 20 ms. Forsinkelse efter hvert gennemløb.

7.4. Trådgruppe – GUI-pakke

Trådgruppe 3 dækker over alle tråde som indgår i GUI pakken.

7.4.1. Kommunikation mellem trådene i GUI-pakken.

Trådene i trådgruppe 3 kommunikerer med henholdsvis OSApi'ets besked system og QT frameworkets interne trådsystem.

7.4.2. Adaptor

Beskrivelse: Ansvarlig for at modtage beskeder fra OSApi'ets beskedsystem.
Levetid: Oprettes når CCSystemet startes og nedlægges når CCSystemet slukkes.
Aktiv: Trådene aktiveres når en besked modtages fra beskedsystemet og videresender beskeden i et format GUI frameworket kan forstå.

7.4.3. QT

Beskrivelse: Ansvarlig for at GUI systemet kører flydende og uafhængigt af det resterende system.
Levetid: Oprettes når CCSystemet startes og nedlægges når CCSystemet slukkes.
Aktiv: aktiveres når et GUI element skal gentegnes på skærmen.

8. Deployment View

Ikke relevant.

9. Implementerings View

Ikke relevant.

10. Generelle designbeslutninger

10.1. Arkitektur mål og begrænsninger

Målet med arkitekturen har været at lave en stabil softwarearkitektur.

Der har været krav fra kunden om at projektudviklingen skulle gennemgås iterativt med delleveringer.

10.2. Arkitektur mønstre

3 lags modellen har været grundstenen for CCSystemets arkitektur.

10.3. Generelle brugergrænsefladeregler

Det er et krav fra kunden, at brugergrænsefladen er så brugervenlig og intuitiv som muligt. Dette skyldes at der ikke stilles krav om stor teknisk forståelse til brugerne af CCSystemet.

10.4. Exception og fejlhåndtering

Systemet kaster kun exceptions ved fatale fejl for CCSystemet. Exceptions lukker systemet ned, da fejlen er så fatal at programmet ikke kan fortsætte med at påvirke den fortsatte kørsel, evt. ved at en fil ikke kan findes.

Systemet benytter logging til at gemme information omkring hvordan systemet enten kører eller har kørt.

Loggen skrives på forskellige niveauer, og niveauet beskriver alvorsgraden til loggingen.

Følgende niveauer er tilgængelige i loggen:

- Emergency: Benyttes når programmet crasher - sker når en exception kastes.
- Warning: Benyttes når nogen kritisk sker, men programmet kan fortsætte.
- Info: Benyttes for at informere om programmets kørsel. F.eks. et tilstandsskift.
- Debug: Benyttes til at debugge systemet.

10.5. Implementeringsværktøj

Følgende værktøjer er benyttet:

Til implementeringen:

- Microsoft Visual Studio 2010
- QT Designer 4

Til dokumentation:

- Google Docs
- Microsoft Word
- LucidChart (hjemmeside)

Til SVN:

- Tortoise (Windows)
- Versions (MAC)
- Rabbit (Linux)

11. Kvalitet

Dette afsnit giver et overblik over kvalitetskravene og beskriver hvad der er gjort for at overholde dem bedst muligt.

Pålidelighed: meget vigtigt
Tidskrævende operationer som fx læsning fra filer er implementeret i tråde, hvilket gør at hvis noget går galt, bliver der ikke taget mere end højst nødvendigt CPU kraft fra vigtigere operationer.

Vedligeholdelsesvenlighed: ikke vigtig
Systemet er vedligholdelsesvenlig, da systemet er strukturet ud fra 3 lags modellen som medfører lav kobling for systemet, og dele af systemet kan derved nemmere skiftes ud.

Brugervenlighed: meget vigtigt
GUI er blevet designet så det er så enkelt som muligt, så enhver kyndig aldersgruppe hurtigt kan sætte sig ind i systemet. Der er nem adgang til at kalibrere systemet, hvis man f.eks. vil skifte hjulstørrelse.

Integritet: vigtig
Data der bliver præsenteret for brugeren er den nyeste tilgængelige fra hardwaren.

Effektivitet: meget vigtigt
Det er vigtigt for systemet, at reaktionstiden er så lav som muligt, især i forhold til en deaktivering af fartpilot, lige meget om det sker gennem bremse, kobling eller betjeningsarmen. Grundet dette er der implementeret to tråde til at tjekke begge noder for en ændring, og herefter give signal videre hvis en ændring forekommer. På denne måde får applikationslaget hurtigt besked hvis bilisten kobler ud eller starter en bremsning.

12. Figuroversigt

Figur 1 - "4+1" View Model [3]

Figur 2 - Systemoversigt

Figur 3 - Aktør-kontekst diagram

Figur 4 - Oversigt over pakker

Figur 5 - Klassediagram over GUI-pakken

Figur 6 - Klassediagram over Business Layer-pakken

Figur 8 - Klassediagram over Data-pakken

Figur 9 - Klassediagram over Driver Interface-pakken

Figur 10 - Sekvensdiagram over BrakeThread

Figur 11 - Sekvensdiagram over EngineThread

Figur 12 - Sekvensdiagram over InputHandlerThread

Figur 13 - Klassediagram over Data Container-pakken

Figur 14 - Sekvensdiagram over Engine::setRPM

Figur 15 - Sekvensdiagram over DistanceCounter::update()

Figur 16 - Oversigt over tråde