Udgave

2

3. SEMESTERPROJEKT

Gruppe 1

CCSystem

Systemarkitektur

Benjamin Sørensen, 02284
Tomas Stæhr Hansen, 03539
Nikki Ashton, 01087
Jonas Livbjerg, 02797
Jeppe Hasager, 01048

CCSYSTEM

Systemarkitektur

Dalgas Avenue 2 • 8000 Aarhus C
IKT

Versionshistorie

Ver.	Dato	Initialer	Beskrivelse
1.00	04.03.2004	BS	Første udgave af systemarkitektur
2.00	06.05.2004	BS/TS	Anden udgave af systemarkitektur Omfattende genskrivning af dokumentet Emner fra Analysen er indarbejdet

Godkendelsesformular

Forfatter(e):	Benjamin Sørensen(BS), Tomas Stæhr(TS), Nikki Ashton(NA), Jonas Livbjerg(JL), Jeppe Hasager(JH)
Godkendes af:	Willy Friis Juul
Projektnummer:	3. semesterprojekt
Antal sider:	39
Kunde:	Atoyot A/S

Ved underskrivelse af dette dokument anses systemarkitekturen som dokumenteret.

Dato og underskrifter:

Århus d.

Benjamin Sørensen

Indholdsfortegnelse

Kapitel 1

1 INTRODUKTION

1.1 Formål og omfang

Dette dokument fastlægger den overordnede arkitektur for CCSystem implementationen. Dokumentet omhandler arkitekturen for hele systemet baseret på det foreliggende analysedokument. Det indeholder designet for systemet uden at gå i detaljer med den egentlige implementering.

Dokumentet indeholder de tre design niveauer specificeret i ROPES udviklingsmodellen: Arkitektur design, mekanistisk design og detaljeret design.

Dokumentationen bidrager til, at andre projektgrupper kan overtage udvikling og opdateringen af CCSystem.

Dokumentationen kan læses uden kendskab til kravspecifikationen.

1.2 Referencer

[GoF 1994] : Design Patterns, Addison-Wesley 1994

[Fowler 2004] : UML Distilled Third Edition, Addison-Wesley 2003

[Kravspecifikation] : Kravspecifikation for CCSystem

1.3 Definitioner og forkortelser

CCSystem: Produktets navn

MVC: Model-View-Controller

UI: Brugergrænseflade

1.4 Dokumentstruktur og læsevejledning

Dokumentationen er opbygget omkring 4+1 view modellen. For at opretholde konsistens med Cruise International Dokumentationsteknik har vi valgt at medtage alle punkter fra 4+1 view modellen. De punkter der er uinteressante i dette projekt, er markeret som ikke relevante.

Derudover er dokumentet udvidet til også at omfatte mekanistisk design og detaljeret design.

1.5 Dokumentets rolle i en iterativ udviklingsproces

CCSystem er udviklet gennem to iterative processer. Dette dokument er udarbejdet efter analysedokumentationen. Under design og implementeringen er der foretaget en række ændringer af beskrivelsen, og dokumentet er reviewet internt efter hver iteration.

Dokumentet indeholder til ethvert tidspunkt den senest implementerede løsning i den iterative proces.

Versionsstyringen nummereres fortløbende. For hvert review stiger versionen med 1.0, og for ændringer under design eller implementering stiger versionen med 0.1

Kapitel 2

2 SYSTEM OVERSIGT

Dette afsnit giver et overordnet billede af systemet, samt de omgivelser systemet skal fungere i.

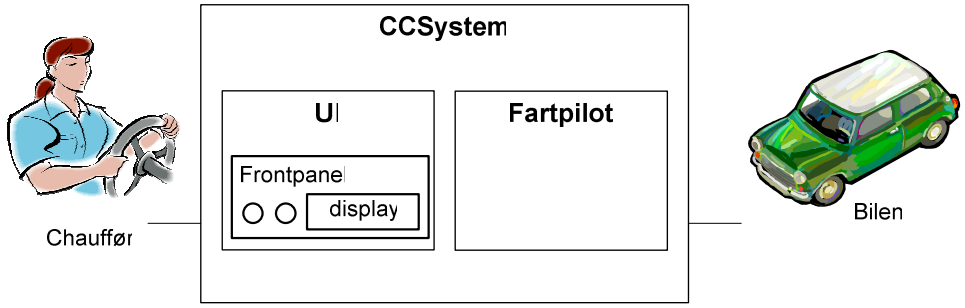
2.1 System kontekst

Systemets navn er CCSystem. Systemet opfylder to formål. Det primære mål er at opretholde en valgt hastighed for en bil. Det sekundære mål er at videregive information om bilen til brugeren.

Det primære mål varetages af en fartpilot og det sekundære af en UI del. I dette projekt skal der kun udvikles fartpiloten og den tilhørende UI-del. Selve hardwaren skal ikke udvikles. Til simulation af hardwaren har Atoyot A/S leveret en testsimulator.

2.1.1 Systemoversigt

CCSystemet kan skitseres som i figur 2-1



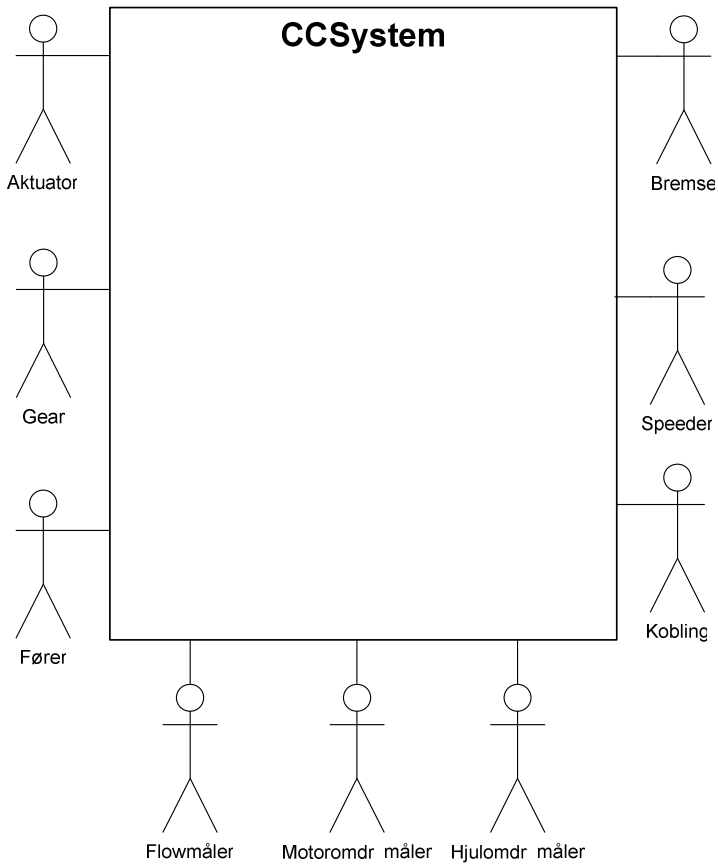
Figur 2-1: Systemoversigt

Figur 2-1 beskriver CCSystemets grænseflader til omverdenen. Chaufføren kan påvirke CCSystemet vha. knapper og får information gennem et display. Dette udgør UI-delen.

Fartpiloten påvirker bilen gennem elektriske signaler. CCSystemet kan overvåge bremse, speeder, kobling og gear og reagere derefter.

2.1.2 Aktør-kontekst diagram

Vi kan beskrive systemets påvirkning på og fra omverdenen gennem et aktør-kontekst diagram.



Figur 2-2: Aktør-kontekst diagram

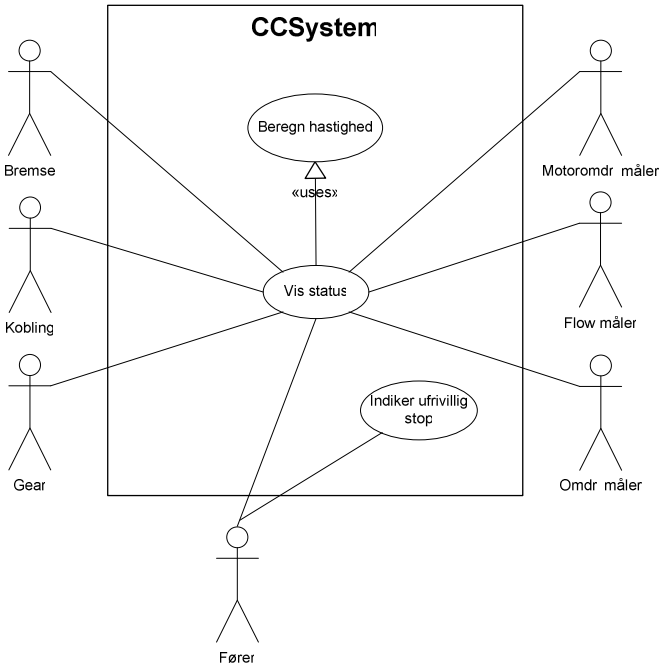
På figur 2-2 ses de aktører der er i kontakt med systemet. Bremse, aktuator, flowmåler, motoromdr. måler, hjulomdr. måler, gear, speeder og kobling er virtuelle aktører, hvorimod Føreren er en virkelig person.

2.2 Systemets funktioner

Systemets funktioner, de funktionelle krav, er fundet og beskrevet vha. use case teknikken. De følgende diagrammer viser systemets funktioner udtrykt som use cases. Formålet med disse diagrammer er at give et overblik over funktionaliteten i det system, der skal udvikles. Hver use case er beskrevet i kapitel 3 i [Kravspecifikation].

2.2.1 Use Case diagrammer

Systemet skal gengive alle relevante oplysninger fra bilen og informationer fra CCSystem til Fører. Denne funktionalitet er samlet i use casen "Vis status" som vist på figur 2-3.

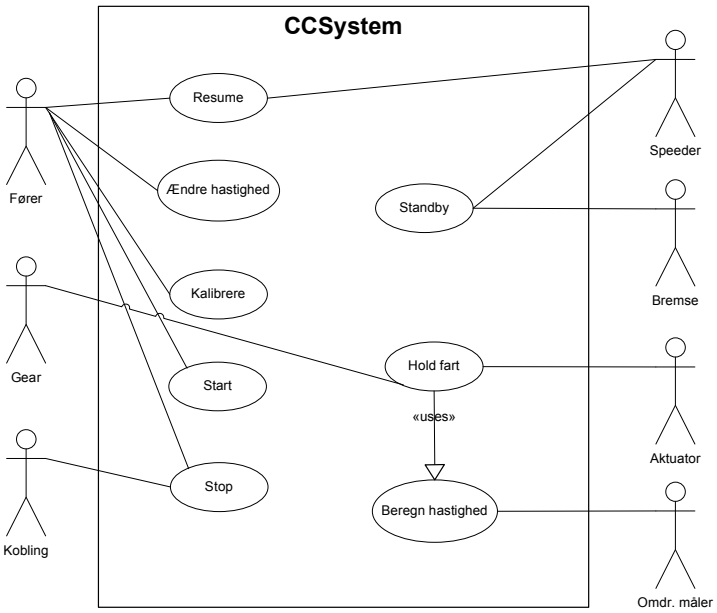


Figur 2-3: Display use cases

Use casen "Vis status" er en systeminitieret use case, der sørger for information til frontpanelet. Den aktiveres når motoren startes og deaktiveres når motoren slukkes.

"Indiker ufrivillig stop" er desuden medtaget på figuren, da den også kommunikerer et budskab til Fører.

Figur 2-4 viser sammenhængen mellem de øvrige use cases og aktører. Det er i disse use cases at selve fartpilot funktionaliteten ligger.



Figur 2-4: Fartpilot use cases

Aktøren Fører har mulighed for at aktivere fartpiloten ved at starte use casen "Start ". Denne use case aktiverer systemets hastighedsregulering, som er styret af "Hold hastighed". Fører kan deaktivere fartpiloten igen vha. "Stop" use casen.

Derudover kan Fører regulere den hastighed fartpiloten skal holde igennem "Ændre hastighed" use casen.

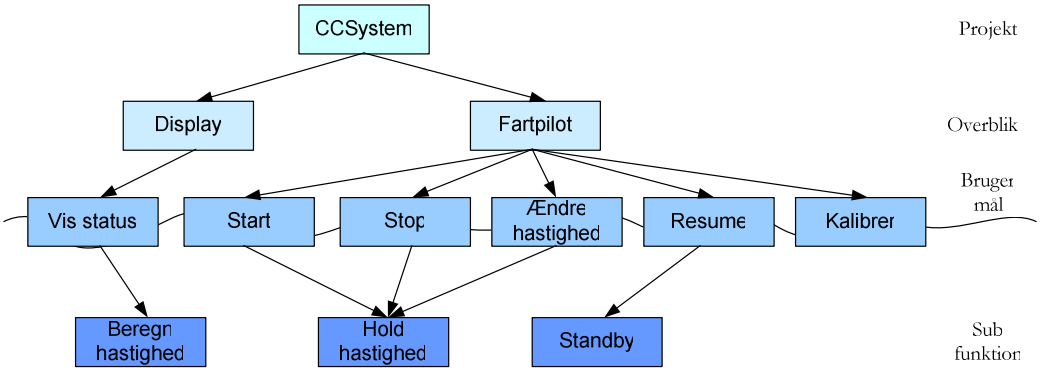
Fører har også mulighed for at kalibrere CCSystemet ved at starte use casen "Kalibrere".

Når fartpiloten er aktiveret kan der ske en række påvirkninger:

1. Hvis der bremses vil use casen "Standby" blive aktiveret. CCSystemet vil stoppe hastighedsreguleringen indtil der trykkes på resumeknappen.
2. Hvis der trykkes på koblingen vil use casen "Stop" starte. Dette medfører at hastighedsreguleringen stoppes.
3. Hvis der trykkes på speederen afbrydes hastighedsreguleringen og genoptages først, når speederen slippes. Dette gøres vha. henholdsvis "Standby" og "Resume" use cases.
4. Når fartpiloten er i standby er der mulighed for at sætte en ny hastighed ved tryk på "Cruise On/Off".

Hvis den valgte hastighed ikke kan holdes pga. forkert gear vil use casen "Indiker ufrivillig deaktivering" starte. CCSystemet vil indikere overfor Føreren, at hastighedsreguleringen stoppes.

De forskellige use cases kan grupperes hierarkisk. Figur 2-5 viser denne gruppering.



Figur 2-5: Use case hierarki

Øverst har vi projektet CCSystem som er delt op i en fartpilot og et display. Under disse to abstraktioner har vi de egentlige use cases. Først de use cases der varetager direkte brugermål, og derefter de use cases der hjælper med at opfylde målene. Brugermål svarer til sealevel niveauet i [Fowler 2004].

Kapitel 3

3 SYSTEMETS GRÆNSEFLADER

Dette kapitel beskriver grænsefladerne for de eksterne aktører.

3.1 Grænseflader til person aktører

CCSystem har kun én personaktør, Fører. Den fysiske grænseflade til Fører, er en monitor og en tastaturboks. Både monitorudskrift og tastatur input er beskrevet i "Brugervejledningen".

3.2 Grænseflader til eksterne system aktører

Ikke relevant.

3.3 Grænseflader til hardware aktører

CCSystem kommunikerer med følgende hardwareaktører:

3.3.1 Aktuator

Aktuator er fastlagt. Aktuatoren sidder parallelt med den normale speeder og giver en lineær bevægelse styret af aktuatorens input signal. Aktuatoren skal fødes med en analog spænding 0-10 V (max 8 mA). PV-2019 kort, analog output: Aout 0.

3.3.2 Gear

Der leveres følgende digitale signaler PA0-PA5 fra gearsensoren (0-5V). IO686 kort, Port A, bit PA0-PA5. (1= 5V, 0= 0V)

	PA0	PA1	PA2	PA3	Pa4	PA5
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
Frigear	0	0	0	0	0	0

3.3.3 Speeder

Speederpedalsensoren leverer et analogt signal (0-10 V). Signalet er 0 når pedalen er sluppet ellers en spænding der er proportional med pedaltrykket. PV-2019 kort, analog input AINP1.

3.3.4 Bremsepedal og Koblingspedal

Bremsepedal og koblingspedal sensorerne leverer hver et digitalt signal (0-5 V), der er logisk 1 når pedalen aktiveres af føreren og logisk 0 når pedalen er sluppet. Koblingspedal: IO686 kort, Port A, bit PA6. Bremsepedal: IO686 kort, Port A, bit PA7.

3.3.5 Hjulomdr. måler

Hjulsensoren sidder på de ikke drivende hjul og giver 2 pulser pr. hjulomdrejning svarende til 65 Hz for maksimal hastighed. IO686 kort, Port C, input bit PC3. (kan via jumper på IO686 kobles til interrupt IRQ5, ledningen forbindes fra jumper stik til Port C, bit PC3.).

3.3.6 Motoromdrj. Måler.

Omdrejningssensoren sidder på motorens knastaksel og leverer pulser fra 0-3KHz, der er proportional med omdrejningstallet. IO686 kort, Timer 2, CLK2 input.

3.3.7 Flowmåler

Flowmåleren sidder i forbindelse med benzintilførslen og anvendes ifm. økonomifunktionerne. Flowmåleren er ikke pt. endelig fastlagt men giver et analogt output (0-10V), der er proportionalt med benzinflowet. PV-2019 kort, analog input AINP0.

3.4 Grænseflader til eksterne software aktører

Ikke relevant.

Kapitel 4

4 USE CASE VIEW

Dette kapital beskriver udvalgte use cases og scenarier, der har betydning for systemarkitekturen.

4.1 Oversigt over arkitektursignifikante Use Cases

Systemarkitekturen tager udgangspunkt i fire use cases: Start, Hold hastighed, Beregn hastighed og Vis status. Den grundlæggende struktur er en MVC[1], der er udvidet med 3 Controllere. De fire use cases er beskrevet efterfølgende.

4.2 Start Use Case

4.2.1 Mål

Aktiverer fastholdelse af hastighed.

4.2.2 Scenarier

Normalforløb:
1. Fører aktiverer fartpiloten.

2. Tjek at bilen står i et af de to højeste gear.
[*Forkert gear*]

3. Fartpiloten skifter tilstand til aktiv.

Undtagelser:
Forkert gear
Use casen afsluttes.

[1] MVC: Model-View-Controller arkitekturpattern, se [GoF 1994]

4.3 Hold hastighed Use Case

4.3.1 Mål

Fastholde hastighed.

4.3.2 Scenarier

Normalforløb:
1. Tjek for aktiv tilstand.

[*Ikke aktiv*]

2. Tjek at bilen står i et af de to højeste gear.
[*Forkert gear*]

3. Udfør Beregn hastighed use casen.

4. Aktuatoren justeres efter hastighed.

5. Starter forfra i normalforløb.
[*Motor slukkes*]

Undtagelser:
Ikke aktiv

Ingen regulering af hastighed.

Forkert gear
Use casen afsluttes.

Motor slukkes
Afbrydelse af systemet medfører at use casen afsluttes.

4.4 Beregn hastighed Use Case

4.4.1 Mål

Beregning af den aktuelle hastighed.

4.4.2 Scenarier

Normalforløb:
1. Signalet fra hjulomdrejningsmåleren aflæses.

2. Hastighed beregnes.

4.5 Vis status Use Case

4.5.1 Mål

Visning af hastighed, brændstofforbrug, kilometertæller, triptæller, motoromdrejninger og gearposition på frontpanelet.

4.5.2 Scenarier

Normalforløb:
1. Aflæser de gemte værdier af triptæller og kilometertæller.
2. Aflæser de aktuelle værdier af brændstofforbrug, motoromdrejninger, gear og hastighed.
3. Opdaterer displayet.

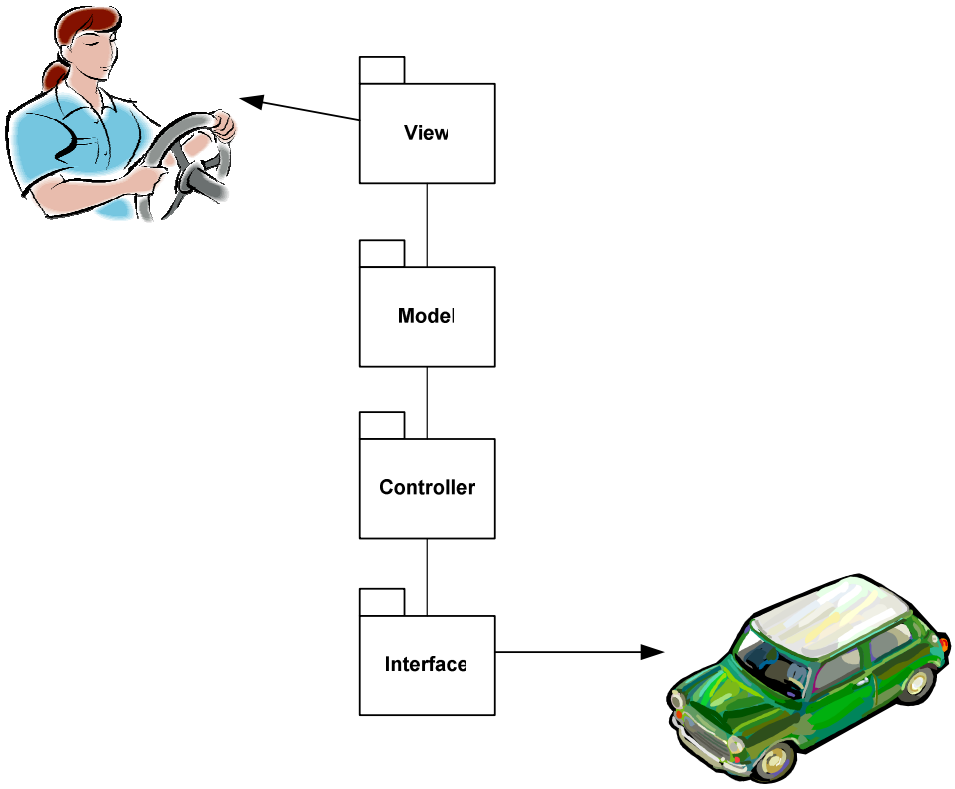
Kapitel 5

5 LOGISK VIEW

Dette kapitel beskriver systemets opdeling i delsystemer (pakker).

5.1 Oversigt

Systemet er opdelt i en 4-lags struktur som vist på figur 5.1.



Figur 5-1: Logisk opdeling i 4 lag. Hvert lag udgør en pakke.

På øverste lag har vi View pakken, der sørger for den grafiske repræsentation af Model pakken, som ligger på 2. lag. Controllerpakken udgør 3. lag. Controllerpakken indeholder den logiske styring af CCSystems funktioner. På 4. lag har vi Interfacepakken, der består af forskellige klasserepræsentationer af objekter i domænet – bilen. Disse klasser er samlet i et fælles interface.

De enkelte pakker er beskrevet i detaljer i næste kapitel.

Kapitel 6

6 ARKITEKTURSIGNIFIKANTE DESIGNPAKKER

Her angives de pakker, der har betydning for udformningen af arkitekturen.

6.1 Beskriver de enkelte pakker.

6.2 Viser det samlede klassediagram.

6.3 Gennemgår use case realiseringer og sekvensdiagrammer over disse.

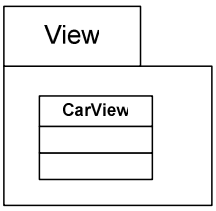
6.1 Pakkebeskrivelser

6.1.1 Pakke 1: View

Ansvar:

Viewpakken står for den grafiske datarepræsentation. Viewpakken indeholder én klasse, CarView, der implementerer displayet[2]. Pakken bliver opdateret af modelpakken, hver gang der sker en ændring i denne.

Klassediagram:



Figur 6-1: View klassediagram

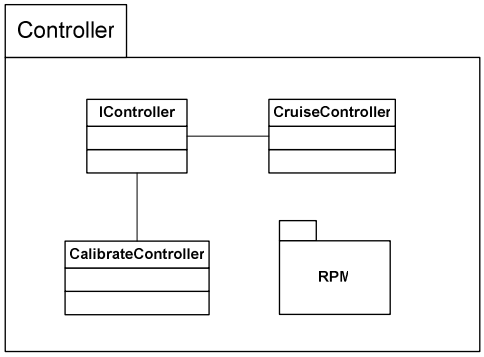
6.1.2 Pakke 2: Controller

Ansvar:

Controllerpakken er ansvarlig for styringslogikken i fartpiloten og behandling af input fra brugeren. Interaktion fra brugeren medfører en opdatering af Modelpakken og derved også Viewpakken.

[2] Se figur 4.1 i kravspecifikationen.

Klassediagram:



Figur 6-2: Controller klassediagram

Pakken består af følgende klasser:

- IController
- CruiseController
- CalibrateController

Samt pakken:

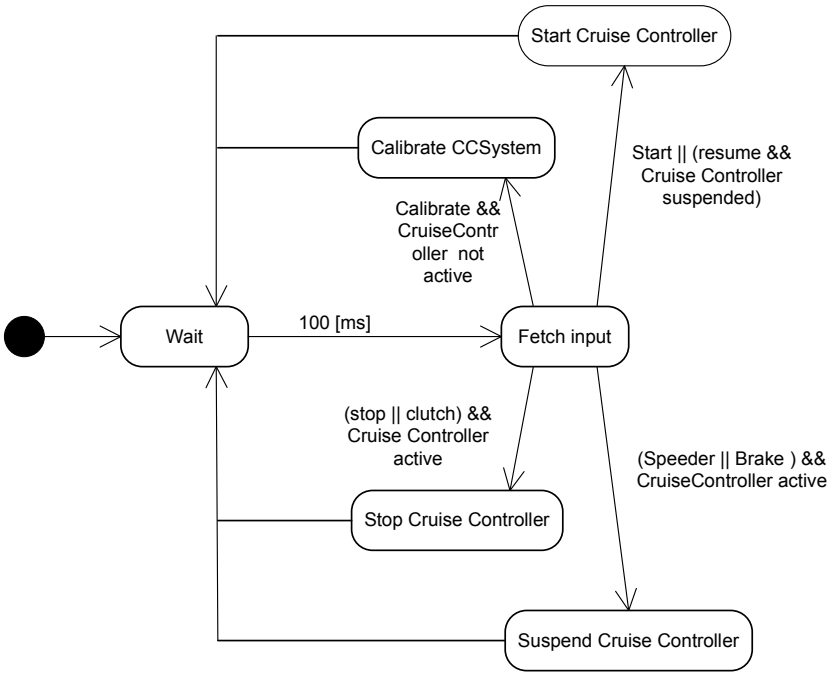
- RPM (Er beskrevet i afsnit 6.1.5)

IController

IControlleren sørger for, at input fra Fører bliver modtaget og behandlet korrekt. Når brugeren trykker på en knap eller en pedal, opdaterer IControlleren Modelpakken. Derudover sørger IController for at udføre den handling, der er forbundet med inputtet.

Tilstandsdiagram:

IControlleren kan beskrives ud fra tilstandsmaskinen i figur 6.3



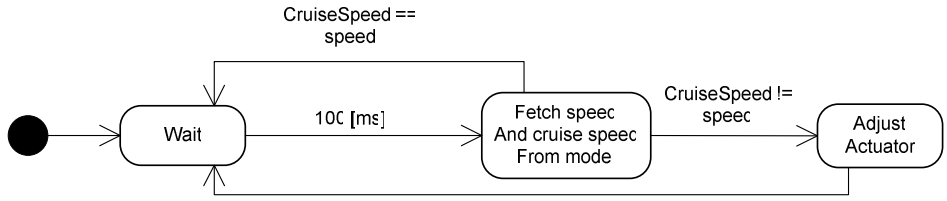
Figur 6-3: IController tilstandsdiagram

CruiseController

CruiseController sørger for styring af fartpiloten ved at regulere aktuatoren.

Tilstandsdiagram:

CruiseController kan beskrives ud fra tilstandsmaskinen i figur 6.4



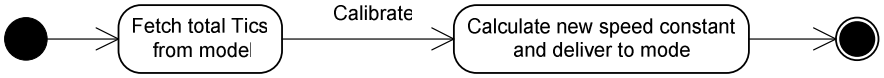
Figur 6-4: CruiseController tilstandsdiagram

CalibrateController

CalibrateController sørger for styring af kalibreringen af CCSystem. Kalibreringen foregår ved at tælle antal hjulomdrejninger pr. kilometer, og beregne en konstant, der bruges til beregning af hastighed.

Tilstandsdiagram:

CalibrateController kan beskrives med en simpel tilstandsmaskine.



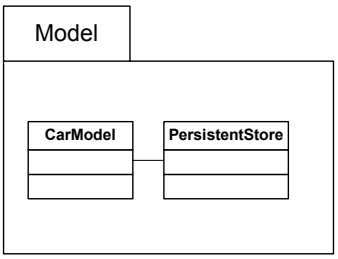
Figur 6-5: CalibrateController tilstandsdiagram

6.1.3 Pakke 3: Model

Ansvar:

Modelpakken står for selve databehandlingen. Modellen udgør en abstrakt modellering af det fysiske problemdomæne. For CCSystem er domænet en bil. Pakken har også ansvar for persistens af data.

Klassediagram:



Figur 6-6: Model klassediagram

Pakken består af følgende klasser:

- CarModel
- PersistentStore

CarModel

CarModel står for håndtering af systemets data. Controllerpakken sørger for at holde modellens data opdateret, og modellen opdaterer Viewpakken når nye data ankommer.

PersistentStore

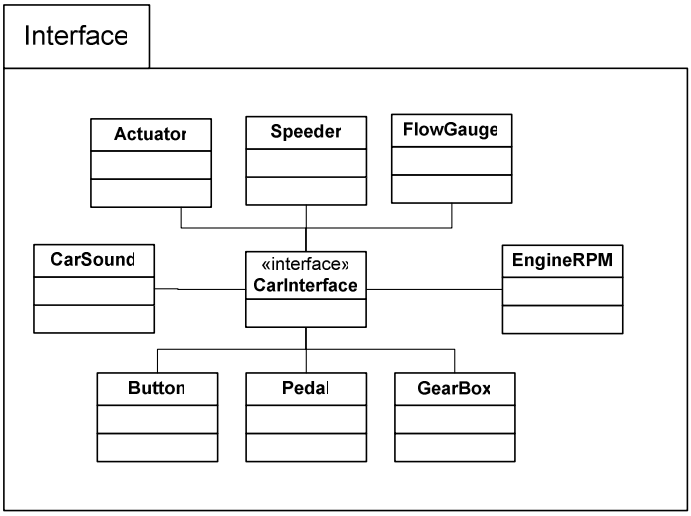
PersistentStore klassen er ansvarlig for at indlæse gemt data ved start af systemet og løbende gemme opdaterede oplysninger på systemets disk. Der gemmes oplysninger om værdier for triptæller, kilometertæller og kalibreringsdata.

6.1.4 Pakke 4: Interface

Ansvar:

Interfacepakken sørger for tilgangen til hardwaren. Interfacepakken specificerer en række funktioner, som Controllerpakken kan bruge for adgang til hardwaren. Alt hardwarespecifikt kode befinder sig i Interfacepakken. Dette medfører en høj genbrugelighed og udskiftelighed.

Klassediagram:



Figur 6-7: Interface klassediagram

Interfacepakkens klasser er hvert tilknyttet et stykke hardware. Adgangen til disse klasser er samlet i en facade for at lette tilgangen.

Pakken består af følgende klasser:

- Actuator
- Speeder
- FlowGauge
- Button
- Pedal
- GearBox
- EngineRPM
- CarSound

som er samlet i en fælles header kaldet CarInterface.

Actuator

Det er muligt at sætte en værdi mellem 0-4095 på aktuatoren.

Speeder

Det er muligt at teste om speederen er aktiv, og hente speederpresset udtrykt som en værdi mellem 0 og 4095.

FlowGauge

Det er muligt at hente benzinflowet udtrykt som en værdi mellem 0-4095.

Button

Det er muligt at lave knapper af disse typer: Start, Resume, Down, Up, Calibrate og Reset. Det er muligt at teste om der er trykket på en knap eller ej. Det er mulig at lave to ekstra knapper, city og highway – disse har dog ingen dokumenteret effekt.

Pedal

Det er muligt at lave to typer pedaler: Bremse og Kobling. Der kan testes om pedal er trykket ned eller ej.

GearBox

Det er muligt at læse gearpositionen. Der sker følgende mapning af gear:

-1 : Bakgear

0 : Fri gear

1: 1. gear

2. 2. gear

3. 3. gear

4. 4. gear

5 : 5. gear

Det er muligt at sætte simulatoren i to eller flere gear på samme tid. I så fald vil GearBox returnere det laveste gear.

EngineRPM

Det er muligt at læse antal motoromdrejninger siden sidste læsning.

CarSound

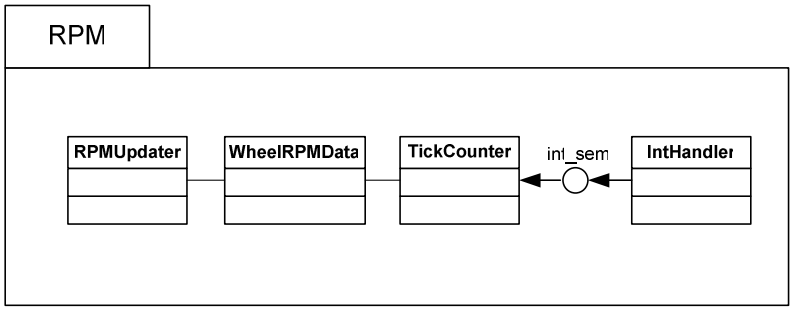
Det er muligt at lave et bip.

6.1.5 Pakke 5: RPM

Ansvar:

RPM-pakken sørger for opdatering af hjulomdrejninger og motoromdrejninger.

Klassediagram:



Figur 6-8: RPM klassediagram

Pakken består af følgende klasser:

- RPMUpdater
- WheelRPMData
- TickCounter

samt en IntHandler til håndtering af interrupts.

IntHandler, TickCounter & WheelRPMData

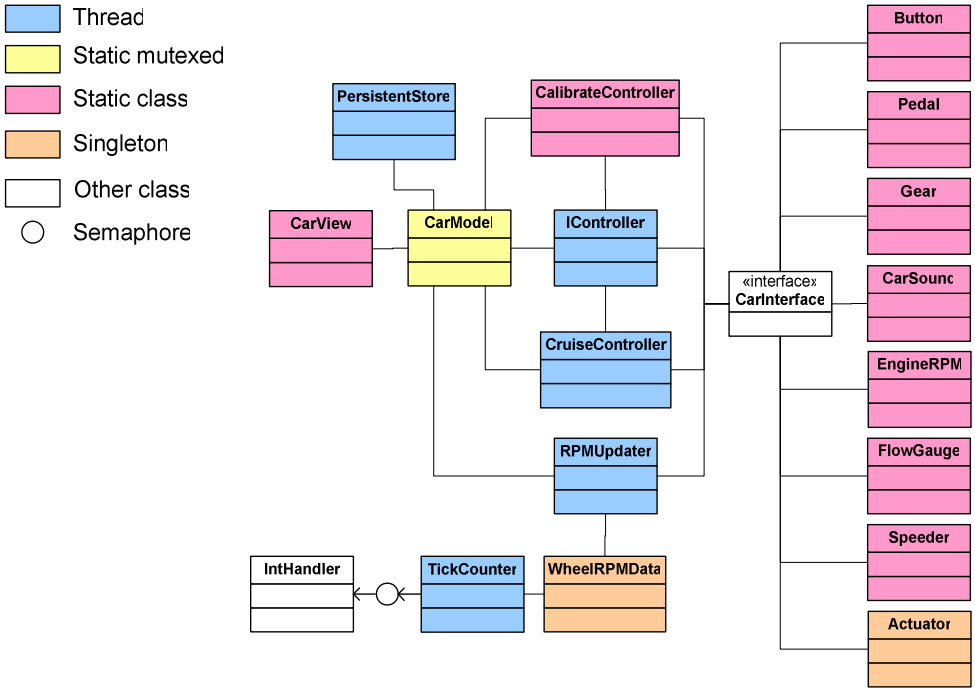
Disse tre klasser sørger tilsammen for at fange interrupts, genereret af hjulomdrejningssensoren, og gemme dem i WheelRPMData klassen.

RPMUpdater

RPMUpdater er ansvarlig for at opdatere CarModel med RPM data. Dvs. data fra WheelRPMData og EngineRPM. RPMUpdater indeholder en instans af EngineRPM, der er med i Interfacepakken.

6.2 Samlet klassediagram

Sammenhængen mellem klasserne kan ses i dette klassediagram



Figur 6-9: Samlet klassdiagram

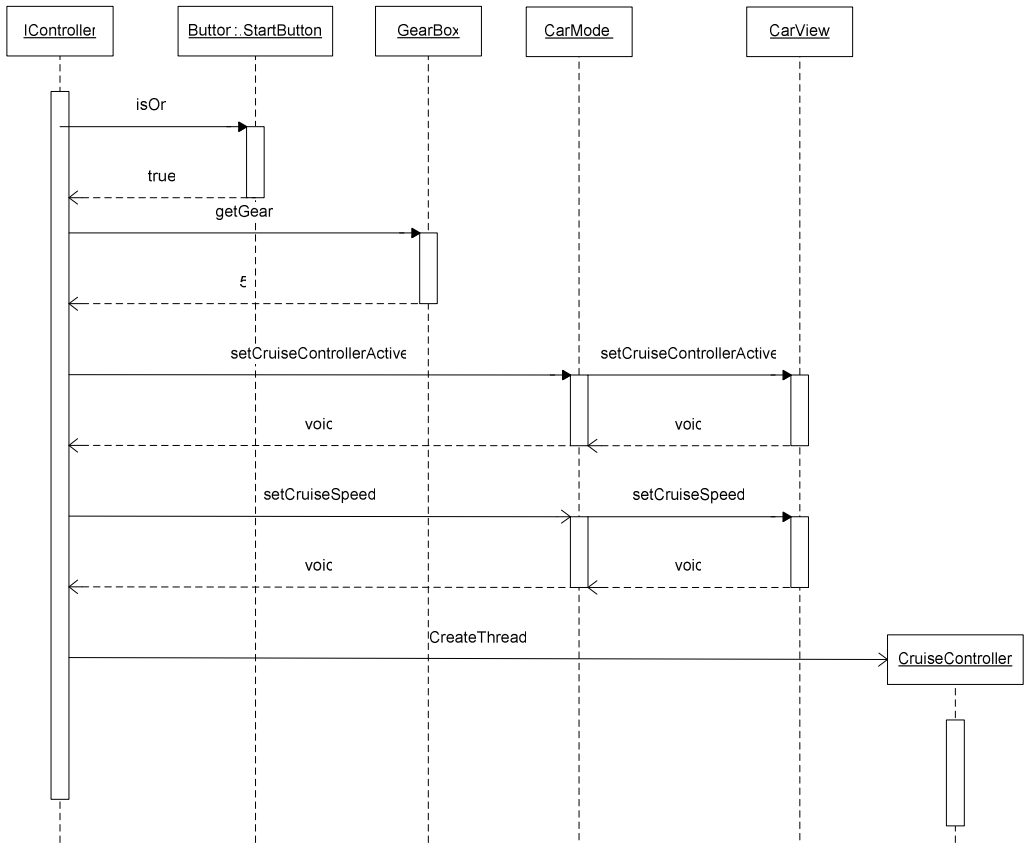
Figur 6.9 viser den statiske sammenhæng i CCSystem. Næste kapitel omhandler den dynamiske sammenhæng, og beskriver CCSystem gennem use case realiseringer. Det fuldstændige diagram, med alle funktioner er vedlagt i bilag 1.

6.3 Use case realiseringer

6.3.1 Start Use Case

Realiseringen af start, stop, resume, standby og ændre hastighed ligger i IControlleren. IControlleren tjekker knapper og pedaler ved at bruge CarInterface'et og opdaterer efterfølgende CarModel. Hvis fartpiloten skal startes, så startes CruiseController i en separat tråd. Skal fartpiloten stoppes eller gå i standby, suspenderes tråden. Skal systemet kalibreres, så startes CalibrateController'en – dog ikke i en separat tråd.

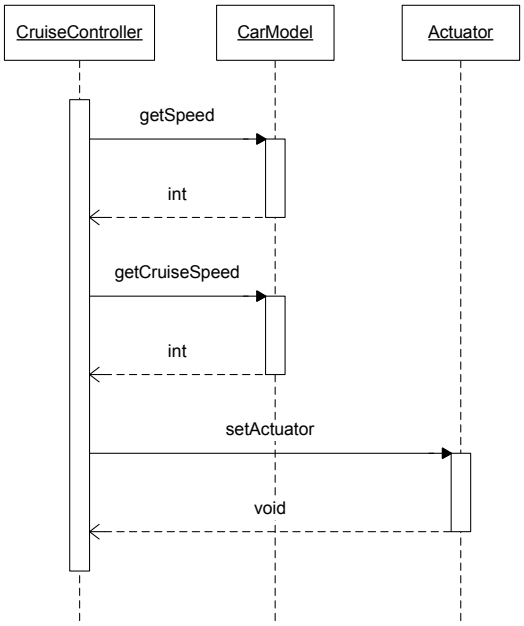
Interaktionen for start use casen kan visualiseres med sekvensdiagrammet i figur 6.10



Figur 6-10: Sekvensdiagram over Start use casen

6.3.2 Hold hastighed Use Case

Kontrolstrukturen for Hold hastighed use casen ligger i CruiseController. CruiseController henter den aktuelle hastighed og den hastighed der ønskes holdt fra CarModel. Hvis de ikke er ens, så reguleres den spænding, der sendes ud til aktuatoren. Dette sker ved at give Actuator en ny værdi.



Figur 6-11: Sekvensdiagram over Hold hastighed use casen

Figur 6.11 viser interaktionen for CruiseController.

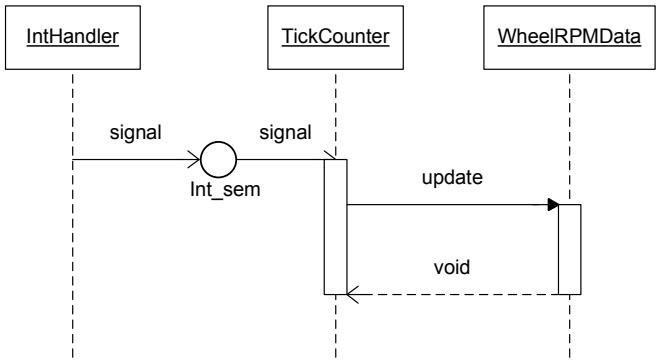
6.3.3 Beregn hastighed Use Case

Realiseringen af Beregn hastighed use casen ligger i RPM pakken, dvs. i følgende klasser:

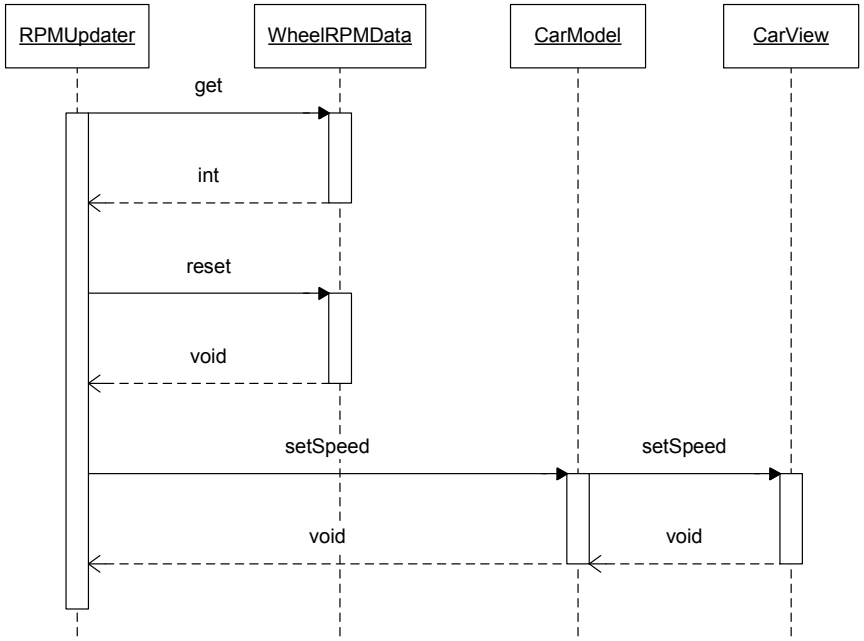
RPMUpdater, WheelRPMData, TickCounter og IntHandler.

For hver halve hjulomdrejning genereres et interrupt på IRQ 5. Servicerutinen i IntHandler sender et signal til int_sem semaphoren. Derved sørger TickCounter for at tælle antal halve hjulomdrejninger op. Disse gemmes i WheelRPMData. RPMUpdater beregner hastigheden hvert 250ms.

Udførelsen af use casen kan visualiseres gennem disse to sekvensdiagrammer.



Figur 6-12: Sekvensdiagram over interruphandler



Figur 6-13: Sekvensdiagram over Beregn hastighed use casen

6.3.4 Vis status Use Case

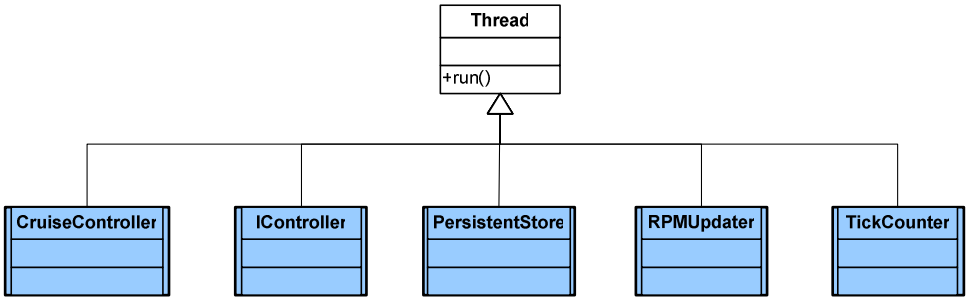
CarView udgør realiseringen af displayet. Dette display opdateres af CarModel, dvs. data 'pushes' til view'et, når nye data ankommer til CarModel. Der foregår ikke noget specielt i denne realisering, men use casen ligger til grund for valget af systemarkitektur. Det anses for unødvendigt med sekvensdiagrammer til beskrivelse af samarbejdet mellem CarModel og CarView.

Kapitel 7

7 PROCES/TASK VIEW

Dette kapitel beskriver systemets opdeling i tråde og hvorledes disse kommunikerer og synkroniserer.

7.1 Oversigt over tråde



Figur 7-1: Taskdiagram

Figur 7.1 angiver de aktive klasser. De er implementeret ved at nedarve fra en abstrakt klasse: Thread. Bilag 1 indeholder det fuldstændige klassediagram, der viser hvordan trådene indgår i den statiske model.

For at holde modellen og userinterfacet opdateret med de korrekte data, implementeres hver Controller som en tråd. CalibrateControlleren er ikke implementeret i en tråd, da det ikke er ønskværdigt at systemet skal reagere på brugerinput mens der kalibreres.
RPMUpdater og TickCounter implementeres som tråde for at optimere nøjagtigheden af de løbende beregninger.
PersistentStore implementeres som en tråd for at adskille skrivning til disken fra resten af systemet og derved undgå evt. forsinkelser i den forbindelse.
Da flere tråde kommunikerer med Modelpakken (se afsnit 5.1) er det nødvendig at sørge for korrekt ressourcedeling.

7.2 Implementering

Der anvendes de semaforer RTKernel stiller til rådighed.

7.3 Kommunikation og synkronisering

Trådene kommunikerer igennem og med Modelpakken. Ressource delingen kontrolleres vha. mutex semaforer i Modelpakken.

7.4 Trådgruppe 1

Trådgruppe 1 dækker over Controllertrådene.
Prioritet 1 angiver højeste prioritet.

7.4.1 Kommunikation i gruppe 1

IController er ansvarlig for at oprette CruiseController-tråden og bestemmer dermed hvornår den skal startes og sættes ud af drift.

7.4.2 Tråd 1 CruiseController

Beskrivelse: Kontrolstruktur for Hold fart use casen.

Levetid: Oprettes når fartpiloten bliver aktiveret og nedlægges når fartpiloten deaktiveres.

Aktiv: Tråden aktiveres med 100 ms forsinkelse efter hvert gennemløb.

Prioritet: 2

7.4.3 Tråd 2 IController

Beskrivelse: Kontrolstruktur for behandling og opsamling af input fra Fører og bil.

Levetid: Oprettes når CCSystemet starter og nedlægges når CCSystem stopper. (CCSystem starter/stopper når bilen tændes/slukkes).

Aktiv: Tråden aktiveres med 100 ms forsinkelse efter hvert gennemløb.

Prioritet: 1

7.5 Trådgruppe 2

Trådgruppe 2 dækker over RPMUpdater og TickCounter.

7.5.1 Kommunikation i gruppe 2

Ikke relevant, da der ikke er direkte kommunikation mellem trådene i gruppe 2.

7.5.2 Tråd 1 RPMUpdater

Beskrivelse: RPMUpdater er ansvarlig for at opdatere CarModel med RPM data.

Levetid: Oprettes når CCSystemet starter og nedlægges når CCSystem stopper. (CCSystem starter/stopper når bilen tændes/slukkes).

Aktiv: Tråden aktiveres med 100 ms forsinkelse efter hvert gennemløb.

Prioritet: 2

7.5.3 Tråd 2 TickCounter

Beskrivelse: TickCounter sørger for at opsamle data fra hjulomdrejningsmåleren vha. en interruptsemafor.

Levetid: Oprettes når CCSystemet starter og nedlægges når CCSystem stopper. (CCSystem starter/stopper når bilen tændes/slukkes).

Aktiv: Tråden aktiveres en gang for hver interrupt på IRQ5.

Prioritet: 2

7.6 Trådgruppe 3

Trådgruppe 3 dækker over PersistentStore.

7.6.1 Kommunikation i gruppe 3

Ikke relevant, da der ikke er kommunikation mellem trådene i gruppe 3.

7.6.2 Tråd 1 PersistentStore

Beskrivelse: PersistentStore er ansvarlig for at indlæse systemoplysninger fra disk ved start og løbende gemme nødvendige oplysninger.

Levetid: Oprettes når CCSystemet starter og nedlægges når CCSystem stopper. (CCSystem starter/stopper når bilen tændes/slukkes).

Aktiv: Tråden aktiveres med 100 ms forsinkelse efter hvert gennemløb.

Prioritet: 3

Kapitel 8

8 DEPLOYMENT VIEW

Ikke relevant.

8.1 Oversigt over systemkonfigureringer

8.2 Systemkonfigureringer

8.2.1 Konfigurering 1.

8.2.2 Konfigurering 2.

8.3 Node-beskrivelser

8.3.1 Node 1. beskrivelse

8.3.2 Node 2. beskrivelse

Kapitel 9

9 IMPLEMENTERINGS VIEW

Ikke relevant

9.1 Oversigt

9.2 Komponentbeskrivelser

9.2.1 Komponent 1 View

9.2.2 Komponent 2

Kapitel 10

10 DATA VIEW

Da sammenhængen mellem implementering og datamodel er triviel er dette kapitel ikke relevant.

10.1 Data model

10.2 Implementering af persistens

Kapitel 11

11 GENERELLE DESIGNBESLUTNINGER

Dette afsnit fastholder de generelle designbeslutninger der tages under arkitekturdesignet eller som er givet som ultimative krav.

11.1 Arkitektur mål og begrænsninger

Hovedformålet er, at fastlægge en stabil arkitektur. CCSystem skal overholde tidskrav, og kører derfor på et realtime operativsystem.

11.2 Arkitektur mønstre

Der anvendes et overordnet MVC mønster. Dette mønster er grundstenen i arkitekturen. Det generelle mønster er udvidet med tre ekstra controllers.

WheelRPMData, Actuator, PPI og PV2019 er implementeret som singletons. Dette sikrer, at der kun instansers ét af hvert objekt. Dette er vigtigt, da flere objekter forsøger at oprette disse objekter og hardwaren kan kun initieres en gang.

For at lette overblikket over adgangen til hardwareklasserne, er indgangen dertil lagt i et samlet interface. Dette letter udskifteligheden af de enkelte klasser. Det samlede interface er et eksempel på en facade.

Singleton og facade mønstrene er beskrevet i [GoF 1994] bogen. MVC berøres også i [GoF 1994].

11.3 Generelle brugergrænsefladeregler

Det er et krav fra kundens side, at brugergrænsefladen skal være så brugervenlig og intuitiv som muligt. Baggrunden for dette er, at der ikke stilles krav om teknisk forståelse til brugerne af CCSystem.

11.4 Exception og fejlhåndtering

Da CCSystem er et indlejret system, må der ikke ske terminering under kørsel. Fejlhåndteringen og recovery skal ske på runtime. Det er ikke muligt at genstarte programmet manuelt.

11.5 Implementeringssprog og værktøjer

CCSystem er udviklet i C++. Der er anvendt Microsoft Visual Studio .NET som editor og compiler. Da CCSystem skal kompileres til RTKernel, anvendes OnTime, der er integreret med Visual Studio.

Til dokumentering af koden anvendes Javadoc standarden, HTML dokumentation er autogeneret med Doxygen.

11.6 Implementeringsbiblioteker

RTKernel 4.07

Kapitel
12

12 STØRRELSE OG YDELSE

Ikke relevant

Kapitel 13

13 KVALITET

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

Pålidelighed: særdeles vigtigt.

Tidskrævende operationer som fx skrivning til disk er implementeret i tråde, så hvis noget går galt, bliver der ikke taget CPU kraft fra vigtigere operationer.

Vedligeholdelsesvenlighed: ikke særlig vigtigt

Dette er der ikke taget højde for

Udvidelsesvenlighed: vigtigt

Designet er opbygget i pakker, så der uden de store problemer kan tilføjes ekstra funktionalitet senere.

Brugervenlighed: særdeles vigtigt

Der er designet med så få knapper som muligt, for ikke at fjerne brugerens fokus fra vejen. Alle nyttige oplysninger vises tydeligt på displayet. Der er nem adgang til at kalibrere systemet, hvis man f.eks. skifter dækstørrelse.

Genbrugbarhed: ikke særlig vigtigt

Systemet vil kunne bruges i alle slags biler blot ved at udskifte interface klasserne.

Integritet: vigtigt

En tråd sørger for at triptæller og kilometertæller, samt kalibreringsdata skrives til disken hvert 100 ms.

Effektivitet: særdeles vigtigt

Det er naturligvis meget vigtigt at der reageres hurtigt på brugerinput. Til dette har vi IControlleren der er implementeret som en selvstændig tråd.

Kapitel 14

14 OVERSÆTTELSE

I det følgende beskrives hvordan kildeteksten til programmet oversættes og linkes til et eksekverbart program, og derefter installeres.

Maskinen der bruges til at oversætte og linke på vil blive omtalt som PC'en. Maskinen programmet skal installeres og afvikles på vil blive omtalt som SBC'en.

14.1 Oversættelses-hardware

PC'en som programmet oversættes og linkes på, skal være en 32 bit maskine med en processor i x86 familien.

14.2 Oversættelses-software

Styresystemet på PC'en skal være enten Windows 9x, Windows NT, Windows 2000 eller Windows XP. Softwaren der bruges på PC'en til at oversætte og linke skal være Visual Studio .Net 2003 version 7.1 med RTKernel plugin installeret.

14.3 Oversættelse og linkning

Følg nedenstående trin på PC'en for at oversætte og linke:

1. Start en kommando prompt i Windows.
2. Indtast kommandoen ”rtk407” og tryk enter.
3. Indtast kommandoen ”devenv” og tryk enter.
4. Åben CCSystem projektet.
5. I Visual Studio trykkes der i menuen på Build -> Build Solution.

Hvis man ønsker at debugge programmet, skal PC’en og SBC’en være forbundet med et serielt kabel, og Visual Studio skal startes på følgende måde:

1. Start en kommando prompt i Windows.
2. Indtast kommandoen ”rtk407” og tryk enter.
3. Indtast kommandoen ”dbgshell devenv.exe” og tryk enter.
4. Åben CCSystem projektet.

14.4 Installation

Det oversatte og linkede program ligger i projekt folderen under debug eller release, alt efter hvordan programmet blev oversat og linket.

Følg nedenstående trin for at installere programmet

1. Indsæt en tom og formateret diskette i drev a: på PC'en

2. Kopier indholdet af Build-mappen fra OnTime installationsmappen over i projektfolderen.
3. Åbn en kommandoprompt.
4. Indtast kommandoen ”rtloc –Dgr ccsystem disk.cfg” og tryk enter.
5. Indtast kommandoen ”bootdisk ccsystem a:” og tryk enter.
6. Indsæt disketten i SBC'en.
7. Start SBC’en og programmet starter automatisk.

Kapitel 15

15 KØRSEL

CCSystemet startes når bilsimulatoren opstartes.

15.1 Kørsels-hardware

CCSystemet udvikles til den udleverede bilsimulator med tilhørende SBC686 med tilhørende I/O kort: IO686 og PV2019.

15.2 Kørsels-software

CCSystemets software udvikles på Cruise-Internationals indlejrede udviklingsplatform SBC686 med tilhørende I/O kort og med anvendelse af Ontimes realtidssoftware.

15.3 Start, genstart og stop

CCSystemet starter og stopper automatisk når bilen startes og slukkes. Fartpiloten er altid deaktiveret når bilen startes og aktiveres på Cruise On/Off knappen. Data vedrørende kilometertæller og kalibrering gemmes og er ikke berørt af, at bilen slukkes. Det er ikke muligt at nulstille kilometertælleren.

15.4 Fejludskrifter

Ikke relevant

Kapitel 16

16 FIGUROVERSIGT

Kapitel 16

17 BILAG

Bilag 1: Fuldstændigt klassediagram.