

Projektdokumentation
Gruppe 1 - AU2
Den intelligente bil

4. Semesterprojekt E4PRJ4
Ingeniørhøjskolen, Aarhus Universitet
Vejleder: Arne Justesen

7. december 2015

Navn	Studienummer	Underskrift
Kristian Thomsen	201311478	
Philip Krogh-Pedersen	201311473	
Lasse Barner Sivertsen	201371048	
Henrik Bagger Jensen	201304157	
Kenn Hedegaard Eskildsen	201370904	
Karsten Schou Nielsen	201370045	
Jesper Pedersen	201370530	

Indhold

1 Projektformulering

Version

Dato	Version	Initialer	Ændring
29. September	1	Alle	Første udkast.
26. Oktober	2	PKP, KT og JEP	Mindre rettelser efter review
29. Oktober	3	PKP, KE	mindre rettelser efter vejledermøde
12. November	4	KSN og HBJ	Rettelser til usecases og PC protokol
13. November	5	KSN	Rettelser til PC protokol

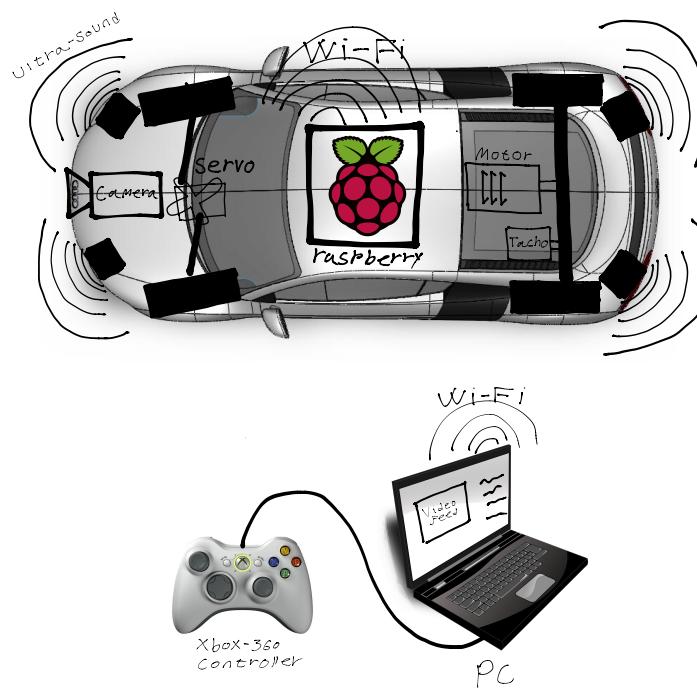
1.1 Problemformulering

Ifølge Niklas Alexander Chimirri, forsker inden for områder som barndom, psykologi og teknologi ved Roskilde Universitet, er leg en vigtig del af børns opvækst. Det er essentielt for deres fremtid da det gør børnene sociale, robuste, kreative og ikke mindst nysgerrige. Med til at skabe rammerne for børns leg er legetøj, og i dagens Danmark er det vigtigt at børn har mulighed for at anvende den teknologi der er til rådighed. Dette bekræftes i en artikel der er udgivet på Roskilde Universitets hjemmeside i november 2014. Han konkluderer at der er for stor forskel imellem den virkelighed børnene møder i, og uden for børnehaven ift. den teknologi der i dag er til rådighed.

1.2 Projektbeskrivelse

Projektet skal bidrage til eller i det mindste sætte fokus på, at det er vigtigt at børn har muligheden for at lege... og gerne med moderne teknologi. Derfor omhandler projektet design og implementering af en fjernstyret bil. Det skal ikke være en almindelig fjernstyret bil - den skal være intelligent og den får navnet "AU2". En skitse af bilen er vist på figur ??.

Den intelligente del består af sensorer samt en kommunikationsenheder, som gør det muligt at styre bilen over et trådløst netværk. Brugeren har hermed mulighed for at navigere bilen ved at betragte en computerskærm, der viser et live-stream med video fra et kamera monteret på bilen. Et billede inden for synsfeltet kan den selvfølgelig også styres ved at se direkte på den. For at undvige forhindringer på kørebanen, implementeres et anti-kollisionssystem bestående af afstandssensorer på bilen, placeret sådan at de kan detektere om bilen nærmer sig en forhindring. Således kan bilen selv standse eller undvige, hvis den nærmer sig en forhindring hastigt. Anti-kollisionssystemet har til formål at forhindre en evt. kollision og derved beskadigelse af bilen eller dens omgivelser.



Figur 1: Rigt billede af systemet i sin helhed

1.3 Ordforklaring

System

Det totale system indeholder bil, software på PC og kommunikation mellem Bil og PC.

HID (Human Interface Device)

Et interface som en bruger anvender til at interagere med en computer fx. tastetur og mus. I dette projekt anvendes desuden en Xbox-360 controller, med følgende funktionalitet:

- Right Trigger (RT)
- Left Trigger (LT)
- Flere knapper her.

Hovedvindue

Hovedvinduet i software på PC indeholder videostream, status på bilen samt muligheder for at konfigurere og kalibrere systemet.

Bil

Med bil menes den hardware der fysisk er placeret på bilen, dette være sig bla. bilens controllerenhed, her et Raspberry Pi 2 board, afstandssensorer, tachometer samt accelerometer.

Pi (Raspberry Pi 2 B)

En Raspberry Pi er en single board computer i kreditkortstørrelse. Den anvendes i dette system som en controller til at styre bilen med.

Wi-Fi netværk

Trådløst netværk af standarden "IEEE 802.11", som Bil og PC kommunikerer over. Dette netværk sættes op lokalt til brug udelukkende for kommunikationen imellem Bil og PC.

AKS (Anti-kollisionssystem)

Et system på bilen bestående af fire afstandssensorer, samt signalbehandling- og reguleringssoftware som er i stand til at forhindre en kollision ved at overtage styring fra Bruger i tilfælde af forestående kollision. Der differentieres mellem "Undvig forhindring" og "Tænd/Sluk AKS".

- Tænd/Sluk bruges i forbindelse med at koble AKS til eller fra, således at bilen ikke vil undgå en kollision hvis AKS er slukket, men vil undgå en kollision hvis AKS er tændt.
- Undvig forhindring bruges i forbindelse med en forestående kollision. Her overtager AKS styring af bilen indtil forhindringen er undveget.

Afstandssensorer

Afstandssensorerne er de 4 ultralydssensorer der er påmonteret bilen. Disse kan herefter benævnes som følgende:

- Front Left (FL)
- Front Right (FR)
- Rear Left (RL)
- Rear Right (RR)

UC (Use Case)

En use case er en standard for et brugsmønster til at afdække funktionalitet for et system.

G(tyngdeacceleration)

G er tyngdeaccelerationen og har enheden m/s^2 . $1G = 9.81m/s^2$

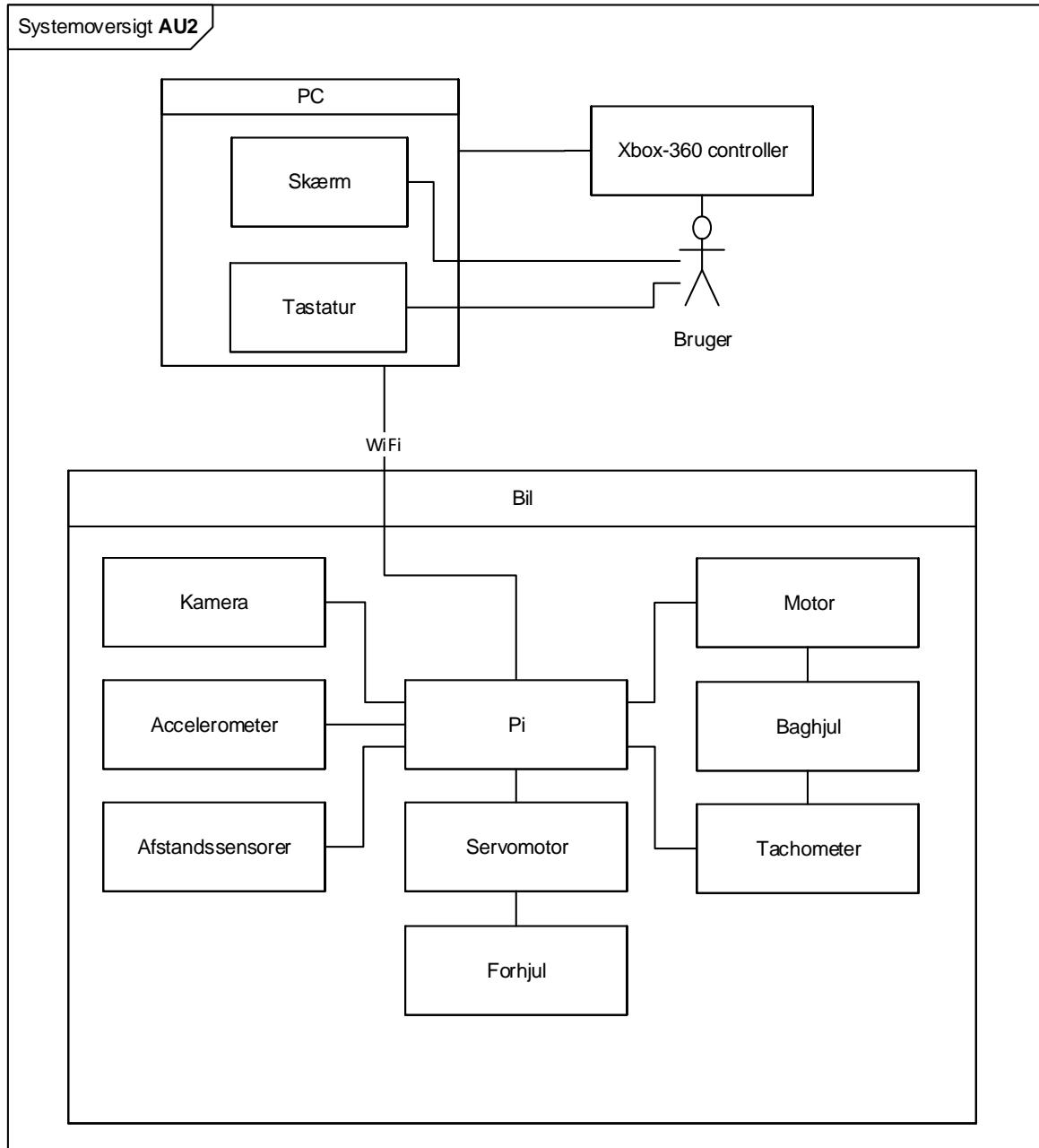
2 Kravspecifikation

Version

Dato	Version	Initialer	Ændring
29. september	1	Alle	Første udkast.
26. oktober	2	PKP, KT og JEP	Mindre rettelser efter review.
9. november	3	PKP	Rettelser til UC3.
12. november	4	KSN og HBJ	Rettelser til usecases og accepttest

2.1 Systemoversigt

På figur ?? ses den overordnede systemoversigt med kommunikationsveje og mekaniske forbindelser. Diagrammet skal give læseren et hurtigt overblik over det samlede system. I afsnittet beskrives blokke og kommunikationsveje mere detaljeret. Under figur ?? er blokkene kort beskrevet.



Figur 2: Overordnet systemoversigt

Pi

Systemets kerne er et Raspberry Pi 2 board. Pi'en står for at processere data fra afstandssensorene, og håndtere streaming af video. Derudover afvikles regulering til motor, samt styring af servo også fra Pi'en.

Servomotor

Servomotor har til opgave at omsætte signal fra Pi'en til mekanisk styring af bilens forhjul.

Afstandssensor

Bilens 2 fremadrettet og 2 bagudrettet afstandssensorer har til formål at indsamle data om eventuelle forhindringer i bilen kørebane.

Accelerometer

Der er påmonteret et accelerometer der anvendes til regulering af hastighed.

Kamera

Bilens kamera streamer video til PC'ens skærm så Bruger har mulighed for at navigere på baggrund af visuel feedback

PC

PC afvikler den software hvorigennem bilen kontrolleres, konfigureres og kalibreres. Det er ligeledes via computeren at Bruger får visuel feedback fra bilens kamera.

Xbox-360 Controller

Til at kontrollere bilen, benyttes en Xbox-360 controller. vha. en række trykknapper og styrepinde kan bilens hastighed, såvel som retning bestemmes.

Motor

Motoren omsætter data, herunder regulering fra Pi'en til mekanisk styring af bilens hastighed.

Tachometer

Motorens omdrejningshastighed kan via tachometeret aflæses og herefter benyttes til databehandling og regulering.

I figur ?? vises en skitse af hovedmenuen i softwaren på PC.



Figur 3: Skitse af hovedmenu

2.2 Aktør-kontekstdiagram

På figur ?? ses aktørkontekstdiagram over systemet.



Figur 4: Aktør kontekst diagram for AU2.

2.3 Aktørbeskrivelser

Som figur ?? viser, er der 2 aktører til systemet. *Bruger* og *Forhindring*.

Bruger - Primær Aktør

Brugeren vil typisk være et barn med alder over 8 år, men kan også være en voksen med interesse for fjernstyrede biler.

Bruger kan:

- Starte og stoppe systemet
- Styre bilen over et Wi-Fi netværk.
- Konfigurer og kalibrere system.

Forhindring - Sekundær Aktør

Forhindring er objekter i det miljø bilen kører i, og som dermed er risiko for at bilen kan kolidere med.

2.4 Funktionelle krav

Ambitionen for dette projekt er som absolut minimum at realisere nedenstående punkter under ”*skal*”. Det forventes desuden at punkterne under ”*bør*” realiseres, men de har lavere prioritet. Punkterne under ”*kan*” forventes ikke realiseret, og punkterne under ”*vil ikke...*” realiseres med sikkerhed ikke. Sidstnævnte punkter kan ses som udviklingsmuligheder i forhold til senere versioner af systemet.

Systemet...

1. ... *Skal* kunne køre frem og tilbage.
2. ... *Skal* kunne dreje.
3. ... *Skal* kunne regulere hastigheden på bilen.
4. ... *Skal* give Bruger mulighed for at begrænse maksimumshastighed.
5. ... *Skal* give Bruger mulighed for manuel styring via Xbox-360 controller af hastighed og retning.
6. ... *Skal* via Wi-Fi netværk kunne kommunikere mellem bil og PC.
7. ... *Skal* kunne identificere forhindringer foran og bag bilen.
8. ... *Skal* indeholde et anti-kollisionssystem baseret på afstandssensorer.
9. ... *Skal* via. anti-kollisionssystem kunne undvige og/eller stoppe før kollision.
10. ... *Skal* indeholde et kamera til at streame video.
11. ... *Bør* give Bruger mulighed for at aktivere/deaktivere anti-kollisionssystemet på bilen.
12. ... *Bør* have bremselflys, som aktiveres når bilen bremser.

2.5 Ikke-funktionelle krav

1. Bilens maksimumshastighed uden begrænsning er $10km/t \pm 1km/t$
2. Bilens bremselængde ved maksimumshastighed uden begrænsning må ikke overstige 1 m.
3. Bilen skal kunne accelerere fra $0km/t$ til maksimumshastighed uden begrænsning på højest 6 s.
4. Forsinkelse fra brugerinput til at bilen reagerer må ikke overstige 50ms.
5. Afstandssensorerne skal kunne identificere en forhindring i form af et kvadrat med en sidelængde på $S = \sqrt{K \times L}$, hvor $K = 0.015m$ og L er afstanden og det gælder at $0.20m < L < 6.00m$. Kvadratet skal være vinkelret på bilen, således at fladen på kvadratet vender direkte mod bilen. På afstande over 6m er det ikke et krav at systemet kan detektere forhindringen.
6. Mister bilen forbindelsen med PC i mere end 50ms, standser bilen automatisk.
7. Kameraet skal minimum have en opdateringshastighed på 15 billeder i sekundet.
8. Systemet skal vise video-stream med en oplosning på 640×480 pixels i hovedvinduet.
9. PC skal som minimum sende kommandoer til bilen 60 gange i sekundet.
10. HID skal bestå af en Xbox-360 controller, tastatur og mus.

2.6 Use Cases

På figur ?? ses use case diagram over de funktionelle krav.



Figur 5: Use case diagram for AU2.

2.6.1 Use Case beskrivelser - Initiering og Formål

UC1: Aktiver system

Initieres af: Bruger

Denne UC giver Bruger mulighed for at aktivere systemet. Bruger åbner software på PC, og sætter bilens "ON/OFF"-knap til "ON" for at tilslutte batteriet. Herefter konfigureres bilen, UC2 + UC3 initieres og PC'en viser hovedvinduet.

UC2: Stream Video

Initieres af: UC1: Aktiver system

Denne UC initierer videotostream fra kameraet, og forbindelsen over Wi-Fi netværket oprettes.

UC3: Overvåg sensorer

Initieres af: UC1: Aktiver system

Denne UC initierer overvågning af bilens sensorer, herunder, de 4 afstandssensorer, tachometer, samt accelerometer. Use casen kører kontinuerligt og henter løbende data fra sensorerne.

UC4: Undvig forhindring

Initieres af: UC3: Overvåg sensorer

Denne UC har til formål at lade AKS overtage styring af bilen under kørsel hvis en forhindring detekteres enten foran eller bagved bilen. Når forhindringen er undveget overgives styringen igen til Bruger.

UC5: Kør bil frem/tilbage

Initieres af: Bruger

Denne UC har til formål at give Bruger mulighed for at ændre hastighed på bilen via de trykfølsomme "LT" og "RT"-knapper på Xbox-360 controlleren. Bruger trykker på "LT" og bilen kører fremad, eller Bruger trykker på "RT" og bilen bakker.

UC6: Drej bil til højre/venstre

Initieres af: Bruger

Denne UC har til formål at lade Bruger ændre bilens retning. Bruger benytter venstre styrepind på Xbox-360 controlleren. Føres styrepinden til venstre, drejer bilens forhjul til venstre. Føres styrepinden til højre, drejer bilens forhjul til højre. Det har ingen betydning hvis styrepinden samtidig føres lidt opad eller nedad.

UC7: Brems bil

Initieres af: Bruger

Denne UC har til formål at lade Bruger sænke bilens hastighed. Bruger trykker "X" på Xbox-360 controlleren, jo længere tid knappen holdes nede jo mere sænkes bilens hastighed. Deaccelerationen er konstant.

UC8: Konfigurer IP-adresse

Initieres af: Bruger

Denne UC har til formål at lade Bruger konfigurere PC'ens IP-adresse således at der kan opnås forbindelse til bilen.

UC9: Tænd/sluk AKS

Initieres af: Bruger

Denne UC har til formål at give Bruger mulighed for at vælge om AKS skal være tændt eller slukket. Bruger kan via "Hovedvindue" på PC'en vælge status for AKS.

UC10: Indstil makshastighed

Initieres af: Bruger

Denne UC har til formål at give Bruger mulighed for at indstille en maksimumshastighed på bilen. Hastigheden indstilles via PC'ens "Hovedvindue".

UC11: Kalibrer styretøj

Initieres af: Bruger

Denne UC har til formål at give Bruger mulighed for at kalibrere bilens styretøj, så den kører ligedt når styrepinden ikke påvirkes. Bruger indtaster via menuen "Kalibrer styretøj" en værdi der angiver center for styretøjet.

UC12: Afbryd system

Initieres af: Bruger

Denne UC har til formål at lade Bruger afbryde hele systemet. Bruger afslutter software på PC, og sætte bilen "ON/OFF"-knap til "OFF" for at afbryde forbindelse til batteriet.

2.6.2 Fully Dressed Use Cases

Use Case 1: Aktiver system

Navn:	UC1: Aktiver system
Mål:	At aktivere system
Initiering:	Bruger
Aktører:	Bruger
Reference:	UC2: Stream video, UC3: Overvåg sensor, UC8: Konfigurer system
Antal samtidige forekomster:	Én
Forudsætning:	Netværksforbindelse er opsat og fungerende
Resultat:	Bil er initieret, PC viser Hovedvindue, UC2 og UC3 er initieret
Hovedscenarie:	<ol style="list-style-type: none"> 1. Bilens "ON/OFF"-switch sættes til "ON". 2. Bruger starter software på PC. 3. Hovedvindue fremkommer på skærmen. 4. Bruger trykker på "Opret forbindelse" 5. PC opretter forbindelse til bilen. <ul style="list-style-type: none"> • [Ext 5.a : Forbindelse kan ikke oprettes] 6. UC2: Stream video initieres af System. <ul style="list-style-type: none"> • [Ext 6.a : Initiering af UC2 fejler] 7. UC3: Overvåg sensorer initieres af System. <ul style="list-style-type: none"> • [Ext 7.a : Initiering af UC3 fejler] 8. PC prompter "Forbindelse oprettet" 9. UC1 afsluttes
Udvidelser:	<p>[Ext 5.a : Forbindelse kan ikke oprettes]</p> <ol style="list-style-type: none"> 1. System prompter "Forbindelse kan ikke oprettes". 2. UC8: Konfigurer IP. 3. Systemet fortsætter fra punkt 3 i hovedscenariet. <p>[Ext 6.a : Initiering af UC2 fejler]</p> <ol style="list-style-type: none"> 1. System prompter "Videostream kan ikke oprettes". 2. UC1 fortsætter fra punkt 3. <p>[Ext 7.a : Initiering af UC3 fejler]</p> <ol style="list-style-type: none"> 1. System prompter "Initiering af sensorer fejlet". 2. UC1 fortsætter fra punkt 3.

Tabel 1: UC1: Aktiver system

Use Case 2: Stream Video

Navn:	UC2: Stream video
Mål:	At starte videostreamen
Initiering:	UC1: Aktiver system
Aktører:	Ingen
Reference:	UC1
Antal samtidige forekomster:	Én
Forudsætning:	UC1 frem til punkt 6 er fuldført
Resultat:	Videostream er initieret og kørende
Hovedscenarie:	<ol style="list-style-type: none"> 1. Bilen initierer kameraet. <ul style="list-style-type: none"> • [Ext 1.a: Initiering af kamera fejler] 2. Bilen streamer video fra kamera til PC via Wi-Fi netværket.
Udvidelser:	<p>[Ext 1.a : Initiering af kamera fejler]</p> <ol style="list-style-type: none"> 1. System prompter PC med "Kamera-initiering fejlet". 2. UC2 afsluttes.

Tabel 2: UC2: Stream video

Use Case 3: Overvåg sensor

Navn:	UC3: Overvåg sensorer
Mål:	At overvåge sensorer
Initiering:	UC1: Aktiver system
Aktører:	Ingen
Reference:	UC1, UC4: Undvig forhindring, UC9: Tænd/sluk AKS
Antal samtidige forekomster:	Én
Forudsætning:	UC1 frem til punkt 7 er fuldført
Resultat:	Sensorer overvåges løbende
Hovedscenarie:	<ol style="list-style-type: none"> 1. Bilen initierer tachometer <ul style="list-style-type: none"> • [Ext 1.a: Initiering af tachometer fejler] 2. Bilen initierer accelerometer <ul style="list-style-type: none"> • [Ext 2.a: Initiering af accelerometer fejler] 3. Bilen initierer afstandssensorer. <ul style="list-style-type: none"> • [Ext 3.a: Initiering af afstandssensorer fejler] 4. Bilen overvåger sensorer. 5. UC4: Undvig forhindring initieres af System <ul style="list-style-type: none"> • [Ext 5.a: AKS er slukket via UC9: Tænd/sluk AKS] 6. Bilen modtager data fra PC. 7. Bilen sender sensor data til PC.
Udvidelser:	<p>[Ext 1.a : Initiering af tachometer fejler]</p> <ol style="list-style-type: none"> 1. Systemet prompter PC med "tachometer-initiering fejlet". 2. UC2 afsluttes. <p>[Ext 2.a : Initiering af accelerometer fejler]</p> <ol style="list-style-type: none"> 1. Systemet prompter PC med "accelerometer-initiering fejlet". 2. UC2 afsluttes. <p>[Ext 3.a : Initiering af afstandssensorer fejler]</p> <ol style="list-style-type: none"> 1. Systemet prompter PC med "afstandssensor-initiering fejlet". 2. UC2 afsluttes. <p>[Ext 5.a : AKS er slukket via UC9: Tænd/sluk AKS]</p> <ol style="list-style-type: none"> 1. Use casen fortsætter fra punkt 6.

Tabel 3: UC3: Overvåg sensorer

Use Case 4: Undvig forhindring

Navn:	UC4: Undvig forhindring
Mål:	At bilen undviger en evt. kollision med en forhindring.
Initering:	UC3: Overvåg sensor
Aktører:	Forhindring
Reference:	UC3, UC6: Drej bil til højre/venstre, UC7: Brems bil
Antal samtidige forekomster:	Én
Forudsætning:	UC1 er gennemført, UC3 er gennemført, bilen er på vej mod en forhindring.
Resultat:	UC5, UC6 og/eller UC7 gennemføres og UC3 fortsætter.
Hovedscenarie:	<ol style="list-style-type: none"> 1. Bilen analyserer indsamlet data fra afstandssensorer, kører den fremad analyseres de forreste sensorer ditto bagud. 2. AKS overtager styring fra Bruger midlertidigt. 3. <ul style="list-style-type: none"> • [ALT a: UC6: Drej bil til højre/venstre aktiveres, hvis en enkelt sensor registrerer en forhindring] • [ALT b: UC7: Brems bil aktiveres hvis begge sensorer registrerer en forhindring] 4. Bilen giver igen styring tilbage til brugeren. 5. UC4 afsluttes.
Udvidelser:	

Tabel 4: UC4: Undvig forhindring

Use Case 5: Kør bil frem/tilbage

Navn:	UC5: Kør bil frem/tilbage
Mål:	At få bilen til at køre frem eller tilbage.
Initiering:	Bruger
Reference:	Ingen
Antal samtidige forekomster:	Én
Forudsætning:	UC1: Aktiver system er fuldført og systemet er operationelt.
Resultat:	Bilens hastighed er ændret.
Hovedscenarie:	<ol style="list-style-type: none"> 1. Bruger ændrer position af RT på Xbox-360 controlleren. <ul style="list-style-type: none"> • [Ext 1.a: Bruger ændrer position af LT.] 2. Controllerens input streames til bilen. 3. Bilen ændrer fremadgående hastighed i henhold til brugerens input. Et hårdere tryk resulterer i en højere hastighed og et lettere tryk resulterer i en lavere hastighed. 4. UC5 afsluttes.
Udvidelser:	<p>[Ext 1.a : Bruger ændrer position af LT.]</p> <ol style="list-style-type: none"> 1. Controllerens input streames til bilen. 2. Bilen ændrer bagudgående hastighed i henhold til brugerens input. Et hårdere tryk resulterer i en højere hastighed og et lettere tryk resulterer i en lavere hastighed. 3. Systemet fortsætter fra punkt 4 i hovedscenariet.

Tabel 5: UC5: Kør bil frem/tilbage

Use Case 6: Drej bil til højre/venstre

Navn:	UC6: Drej til højre/venstre
Mål:	At få bilen til at dreje mod højre eller venstre
Initiering:	Bruger
Aktører:	Bruger
Reference:	UC3
Antal samtidige forekomster:	Én
Forudsætning:	UC1: Aktiver system er fuldført og systemet er operationelt
Resultat:	Retningen på bilens forhjul er ændret
Hovedscenarie:	<ol style="list-style-type: none"> 1. Bruger ændrer position på den venstre styrepind på xbox-360 controlleren. <ul style="list-style-type: none"> • [Ext 1.a: AKS bliver anvendt.] 2. Controllerens input streames til bilen. 3. Bilen behandler input fra Bruger, hvis styrepinden føres til venstre drejes forhjulene til venstre, hvis styrepinden føres til højre drejes forhjulene ligeledes til højre. 4. UC6 afsluttes.
Udvidelser:	<p>[Ext 1.a : AKS bliver anvendt.]</p> <ol style="list-style-type: none"> 1. Bilen analyserer input fra UC3. 2. Bilen drejer til højre, hvis sensor FL registrerer en forhindrer, ditto venstre og FR. 3. Bilen undviger forhindringen. 4. Systemet fortsætter fra punkt 3 i hovedscenariet.

Tabel 6: UC6: Drej til højre/venstre

Use Case 7: Brems bil

Navn:	UC7: Brems bil
Mål:	At få bilen til at bremse
Initiering:	Bruger
Aktører:	Bruger
Reference:	UC3
Antal samtidige forekomster:	Én
Forudsætning:	UC1: Aktiver system er fuldført og systemet er operationelt
Resultat:	hastigheden på bilen er sænket
Hovedscenarie:	<ol style="list-style-type: none"> 1. Bruger trykker på "X" knappen på Xbox-360 controlleren. <ul style="list-style-type: none"> • [Ext 1.a: AKS er anvendt.] 2. Controllerens input streames til bilen. 3. Bilen tjekker input, hvis bremsekmando modtages sænker bilen hastigheden. 4. UC7 afsluttes.
Udvidelser:	<p>[Ext 1.a : AKS er anvendt]</p> <ol style="list-style-type: none"> 1. Systemet initierer UC4 2. Systemet fortsætter fra punkt 3 i hovedscenariet

Tabel 7: UC7: Brems bil

Use Case 8: Konfigurer IP-adresse

Navn:	UC8: Konfigurer IP-adresse
Mål:	At konfigurere bilens IP-adresse til PC'en
Initering:	Bruger
Aktører:	Bruger
Reference:	Ingen
Antal samtidige forekomster:	Én
Forudsætning:	UC1: Aktiver system er udført til punkt 3, bilen og PC er på samme netværk, systemet viser "Hovedvindue" samt at systemet er operationelt
Resultat:	IP adressen på bilen er indstillet
Hovedscenarie:	<ol style="list-style-type: none"> 1. Bruger trykker på "Konfigurer IP". 2. Konfigurationssmenuen for IP-adressen vises, og der er mulighed for at indtaste en IP-adresse. 3. Bruger indtaster bilens IP-adresse. 4. Bruger trykker "Gem" og system viser "Hovedvindue". 5. Bruger trykker på "Opret forbindelse". 6. Hovedvindue viser "Forbindelse oprettet". <ul style="list-style-type: none"> • [Ext 6.a Hovedvindue viser "Forbindelse ikke oprettet"]. 7. UC8 afsluttet.
Udvidelser:	[Ext 6.a Hovedvindue viser "Forbindelse ikke oprettet"] <ol style="list-style-type: none"> 1. Bruger gentager fra punkt 2 i hovedscenarie.

Tabel 8: UC8: Konfigurer IP-adresse

Use Case 9: Tænd/sluk AKS

Navn:	UC9: Tænd/sluk AKS
Mål:	At tænde eller slukke for AKS på bilen
Initiering:	Bruger
Aktører:	Bruger
Reference:	UC11: Kalibrer styretøj
Antal samtidige forekomster:	Én
Forudsætning:	UC1: Aktiver system er udført, bilen og PC er på samme netværk, at systemet viser "Hovedvindue" samt at systemet er operationelt
Resultat:	AKS status ændret
Hovedscenarie:	<ol style="list-style-type: none"> 1. Bruger trykker på "AKS-On". <ul style="list-style-type: none"> • [Ext 1.a: Knappens tekst er "AKS-Off"] 2. Systemet opdater AKS-status 3. System ændre knappen til "AKS-Off"
Udvidelser:	<p>[Ext 1.a: Knappens tekst er "AKS-Off"]</p> <ol style="list-style-type: none"> 1. Systemet opdater AKS-status 2. System ændre knappen til "AKS-On"

Tabel 9: UC9: Tænd/sluk AKS

Use Case 10: Indstil makshastighed

Navn:	UC10: Indstil makshastighed
Mål:	At konfigurere bilens makshastighed
Initering:	Bruger
Aktører:	Bruger
Reference:	UC8: Konfigurer IP-adresse
Antal samtidige forekomster:	Én
Forudsætning:	UC1: Aktiver system er udført, bilen og PC er på samme netværk, at systemet viser "Hovedvindue" samt at systemet er operationelt
Resultat:	
Hovedscenarie:	<ol style="list-style-type: none"> 1. Bruger trykker på "Indstil makshastighed". 2. Systemet præsenterer menu makshastighed med mulighed for indtastning af makshastighed fra 1-10 km/t. 3. Menuen indikerer bilens nuværende makshastighed. 4. Bruger indtaster bilens nye makshastighed. 5. Bruger trykker på "Opdater". 6. "Hovedvindue" viser den nye værdi som makshastighed. <ul style="list-style-type: none"> • [Ext 1. "Hovedvindue" viser ikke den nye makshastighed]
Udvidelser:	<p>[Ext 1. Menuen indikerer ikke den nye makshastighed]</p> <ol style="list-style-type: none"> 1. Bruger går til UC8

Tabel 10: UC10: Indstil makshastighed

Use Case 11: Kalibrer styretøj

Navn:	UC11: Kalibrer styretøj
Mål:	At kalibrere systemet så bilen kører ligeud når brugeren slipper styrepinden på Xbox-360 controlleren
Initiering:	Bruger
Aktører:	Bruger
Reference:	Ingen
Antal samtidige forekomster:	Én
Forudsætning:	UC1: Aktiver system er udført, bilen og PC er på samme netværk, at systemet viser "Hovedmenu", at systemet er operationelt samt bilen holder stille
Resultat:	Bilens styretøj er kalibreret
Hovedscenarie:	<ol style="list-style-type: none"> 1. Bruger vælger "Kalibrer styretøj". 2. Systemet viser menu for Kalibrering med mulighed for indtastning af værdi mellem -50 og 50, hvor -50 svarer til fuldt udslag til venstre og 50 vil fuldt udslag til højre. 3. Bruger indtaster værdi. 4. Bruger trykker på "Gem". 5. Forhjulene drejer en absolut værdi mod enten, højre eller venstre: positiv værdi oversætte til højre, og negativ værdi oversættes venstre. 6. Systemet returnerer til "Hovedvindue"
Udvidelser:	

Tabel 11: UC11: Kalibrer styretøj

Use Case 12: Afbryd system

Navn:	UC12: Afbryd system
Mål:	At lukke systemet ned
Initering:	Bruger
Aktører:	Bruger
Reference:	Ingen
Antal samtidige forekomster:	Én
Forudsætning:	UC1: Aktiver system er udført, bilen og PC er på samme netværk, at systemet viser "Hovedvindue" samt at systemet er operationelt
Resultat:	Systemet er lukket sikkert ned og forsyning til batteriet er afbrudt
Hovedscenarie:	<ol style="list-style-type: none"> 1. Bruger lukker ned for softwaren på PC'en. <ul style="list-style-type: none"> • [Ext 1.a Bilen er løbet tør før strøm] 2. GUI'en sender shutdown kommando til bilen. 3. GUI'en modtager ACK fra bilen. <ul style="list-style-type: none"> • [Ext 3.a GUI'en modtager NACK] 4. Bruger skubber kontakten "ON/OFF" på undersiden af bilen til position "OFF". 5. Strømmen til bilen afbrydes.
Udvidelser:	<p>[Ext 1.a Bilen er løbet tør før strøm]</p> <ol style="list-style-type: none"> 1. GUI'en lukker korrekt ned. 2. Use Case 12 fortsætter fra punkt 3 i hovedscenariet. <p>[Ext 3.a GUI'en modtager NACK]</p> <ol style="list-style-type: none"> 1. GUI'en viser en advarsel med at bilen ikke kunne lukkes sikkert ned. 2. Usecasen gentages fra punkt 1.

Tabel 12: UC12: Afbryd system

3 Systemarkitektur

Version

Dato	Version	Initialer	Ændring
26. oktober	2	Alle	Første udkast
29. oktober	3	Alle	Rettelser fra vejleder
	4		

3.1 Indledning

Følgende afsnit beskriver arkitekturen for hardware- og software delene af projektet. BDD og IBD er lavet mhp. forståelse og indblik i systemet, således alle grænseflader og interne dele af systemet bliver forklaret. Til hvert diagram vil der være en kort forklaring, som beskriver det yderligere.

3.2 BDD for AU2

På figur ?? ses det overordnede blokdiagram for AU2. Der vises tilhørsforhold og sammenhæng med det samlede system, som senere biver uddybet mere.



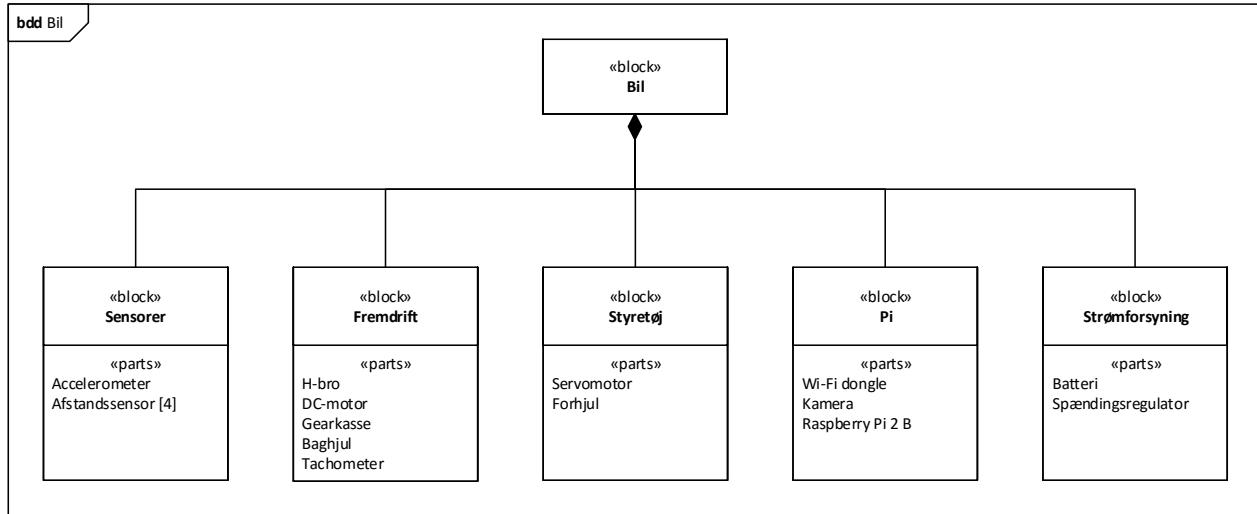
Figur 6: Overordnet BDD for AU2

3.3 Bil

3.3.1 Diagrammer for bil

BDD for bil

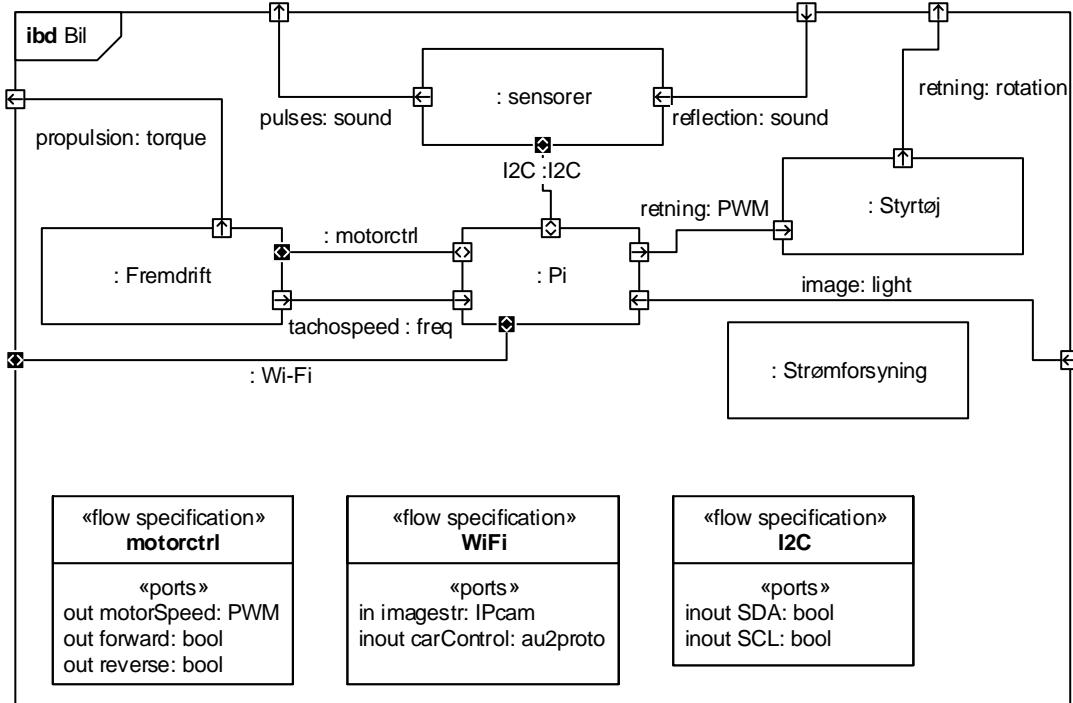
Dette diagram viser blokken bil fra figur ???. Blokken 'bil' skal forstås som alle mekaniske og elektriske som er fastgjort på køretøjet. Således er



Figur 7: BDD for bil

IBD for signaler i bil

På figur ?? ses de interne forbindelser for figur ???. Diagrammet skaber et overblik over hvilke signaler der sendes og modtages. Bemærk at alle forsyningerne ikke er taget med på diagrammet, men istedet er lavet i et diagram for sig. Forsyningerne kan ses på figur ??.



Figur 8: IBD for bil

Signalbeskrivelser for bilens signaler

Signal (navn: type)	Funktion	Tolerancer	Kommentarer
Propulsion: torque	Baghjulenes torque til underlaget.		
motorSpeed: PWM	Et PWM signal der bestemmer motorhastigheden.	Frekvens: 30kHz +/- 1kHz 0-5V +/- 0.2V	Logisk signal: Lav = 0V +/- 0.2V Høj = 5V +/- 0.2V
forward: bool	Kontrolsignal til H-bro.	0-5V ± 0.2V	Lav = 0V +/- 0.2V 'idle' Høj = 5V +/- 0.2V 'frem'
reverse: bool	Kontrolsignal til H-bro.	0-5V ± 0.2V	Lav = 0V +/- 0.2V 'idle' Høj = 5V +/- 0.2V 'tilbage'
tachoSpeed: freq	Digitalt signal med varierende frekvens afhængig af baghjulenes omdrehningshastighed.		Vejledende: 64Hz = 10Km/t Logisk signal: Lav = 0V +/- 0.2V Høj = 5V +/- 0.2V
Inout SDA: bool	I ² C dataline til sensorer herunder accelerometer og afstandssensorer.	0-5V ± 0.5V	Logisk signal: Lav = 0V ± 0.5V Høj = 7.2V ± 0.5V
Inout SCL: bool	I ² C clockline til sensorer herunder accelerometer og afstandssensorer.	0-5V ± 0.5V	Logisk signal: Lav = 0V ± 0.5V Høj = 7.2V ± 0.5V
Pulses: sound	Ultralydsbølger afsendt af sensor.	Jfv. Datablad (henvisning kommer senere)	
reflection: sound	Refleksionsbølge af udsendte ultralydsbølger.	Jfv. Datablad (henvisning kommer senere)	
retning: PWM	PWM signal der vha pulsbredden angiver hvilken retning servomotoren skal dreje og dermed hvilken retning bilen skal dreje.	Pulsbredde: 0.5ms – 2.5ms Freq = 360Hz 0.5ms = 18% Duty cycle (Venstre) 2.5ms = 90% Duty cycle (Højre)	
retning: rotation	Får bilen til at dreje.	30 grader til hhv. venstre og højre ± 5 grader	
imagestr: IPcam	Karsten... Hej		
carControl: au2proto	Få fingeren ud.		

Forsyninger

Diagrammet på figur ?? tilsvarer direkte figur ??, blot med beskrivelsen af forsyning. Dette giver forbedret overblik da de to diagrammer sat sammen bliver uoverskueligt.



Figur 9: IBD for bilens forsyninger

Signalbeskrivelse for bilens forsyning

Signal (navn: type)	Funktion	Tolerancer	Kommentarer
forsyning: VCC	Forsyningsspænding fra det tilkoblede batteri.	7.2V DC \pm 1V max. 20A	Aflæst på batteriet.
GND: GND	Reference.	0V	
styrtøj: 7.2V DC	Forsyningsspænding til styrtøj herunder servomotor.	7.2V DC \pm 0,5V max 400 mA	Fundet i databladet for servoen [?].
accelerometer: 3.3V DC	Forsyningsspænding til accelerometeret.	3.3V DC \pm 0.2V max 8 mA	Fundet i databladet for servoen [?].
afstandssensor: 5V DC	Individuel forsyningsspænding til afstandssensorerne.	5V DC \pm 0.5V max 400 mA	Fundet i databladet for sensorerne [?].
PI: 5V DC	Individuel forsyningsspænding til Pi.	5V DC \pm 0.5V max 1.8A	Aflæst i FAQ for Pi [?].
Tacho: 5V DC	Individuel forsyningsspænding til tachometeret.	5V DC \pm 0.5V max 8 mA	Aflæst i datablad for sensor [?].
motor: 7.2V DC	Individuel forsyningsspænding til motoren.	7.2V DC \pm 1V max 2A	Udregnet ud fra stall-test i laboratoriet.

3.3.2 Fremdrift

Bilens fremdrift forårsages af motoren samt tilhørende elektronik, hvilket er beskrevet på figur ???. Det skal igen noteres at forsyningen til H-broen ikke er på diagrammet, men findes på figur ???. Motoren trækker altså ikke sin strøm fra signalet motorCtrl.



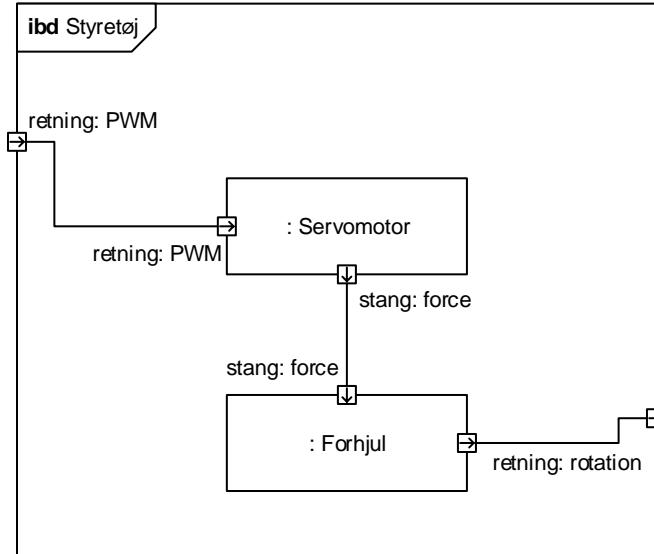
Figur 10: IBD for blokken fremdrift

Signalbeskrivelse for fremdrift

Signal (navn: type)	Funktion	Tolerancer	Kommentarer
motorSpeed: PWM	Et PWM signal der bestemmer motorhastigheden.	Frekvens: 30kHz +/- 1kHz 0-5V +/- 0.2V	Logisk signal: Lav = 0V +/- 0.2V Høj = 5V +/- 0.2V
forward: bool	Kontrolsignal til H-bro.	0-5V ± 0.2V	Lav = 0V +/- 0.2V 'idle' Høj = 5V +/- 0.2V 'frem'
reverse: bool	Kontrolsignal til H-bro.	0-5V ± 0.2V	Lav = 0V +/- 0.2V 'idle' Høj = 5V +/- 0.2V 'tilbage'
motorP: PWR	Et PWM med frekvens som motorSpeed, dog med mulighed for højere effekt. Dette signal forsyner motoren.	Frekvens: 30kHz +/- 1kHz 0-7,2V +/- 0.5V	Logisk signal: Lav = 0V +/- 0.5V Høj = 7,2V +/- 0.5V
motorM: PWR	Reference til motorP.	0V ± 0.5V	
fastRot: torque	Kraft der overføres fra motor til gearkasse via drivaksel.	-	
slowRot: torque	Kraft der overføres fra gearkasse til baghjul via drivaksel.	-	
: angularFreq	Hjulenes omdrejningshastighed.	-	
tachoSpeed: freq	Digitalt signal med varierende frekvens afhængig af baghjulenes omdrejningshastighed.	-	Vejledende: 64Hz = 10Km/t Logisk signal: Lav = 0V +/- 0.2V Høj = 5V +/- 0.2V
Propulsion: torque	Baghjulenes torque til underlaget.	-	

3.3.3 Styretøj

De interne signaler for blokken styretøj er beskrevet nedenfor i figur ??.



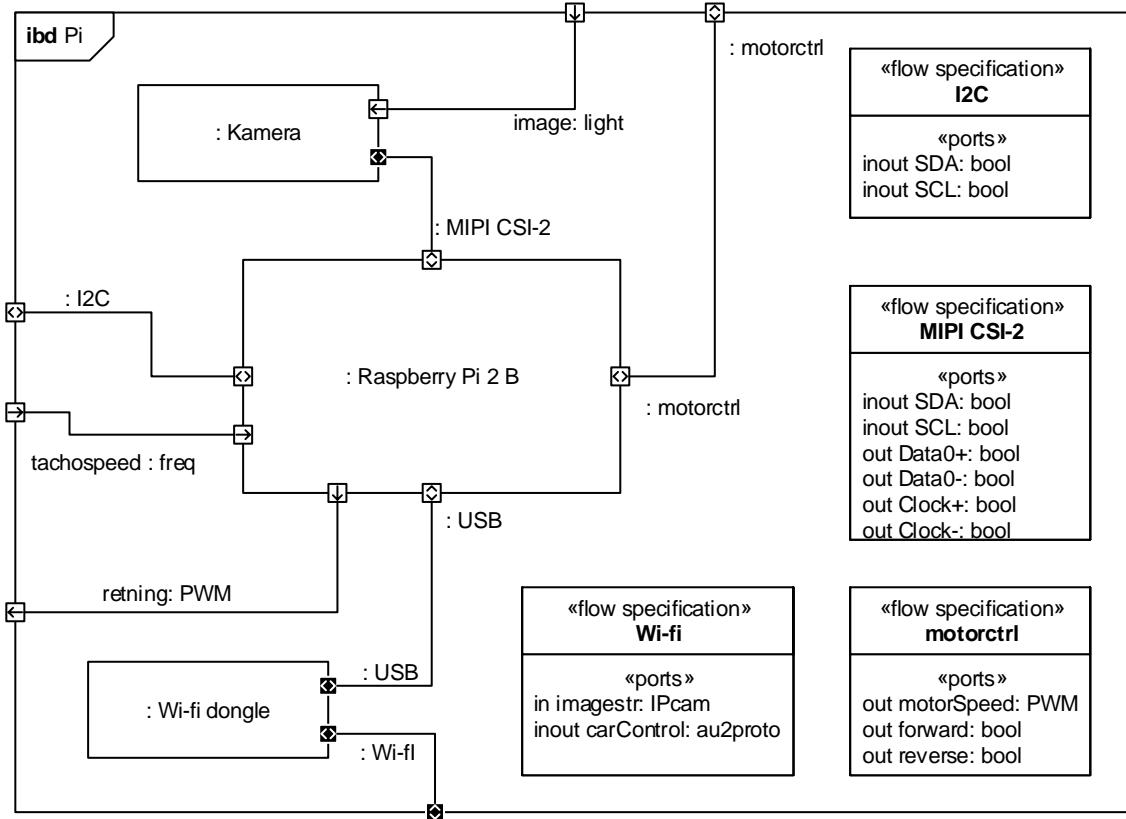
Figur 11: IBD for blokken styretøj

signalbeskrivelse for styretøj

Signal (navn: type)	Funktion	Tolerancer	Kommentarer
retning: PWM	PWM signal der vha pulsbredden angiver hvilken retning servomotoren skal dreje og dermed hvilken retning bilen skal dreje.	Pulsbredde: 0.5ms – 2.5ms Freq = 360Hz 0.5ms = 18% Duty cycle (Venstre) 2.5ms = 90% Duty cycle (Højre)	
stang: force	Skal overføre kraften fra servomotoren til forhjulene. Dette sker via en stang.	-	
retning: rotation	Får bilen til at dreje.	30 grader til hhv. venstre og højre ± 5 grader	

3.3.4 Pi

Her beskrives intern kommunikation for controlleren PI i systemet.



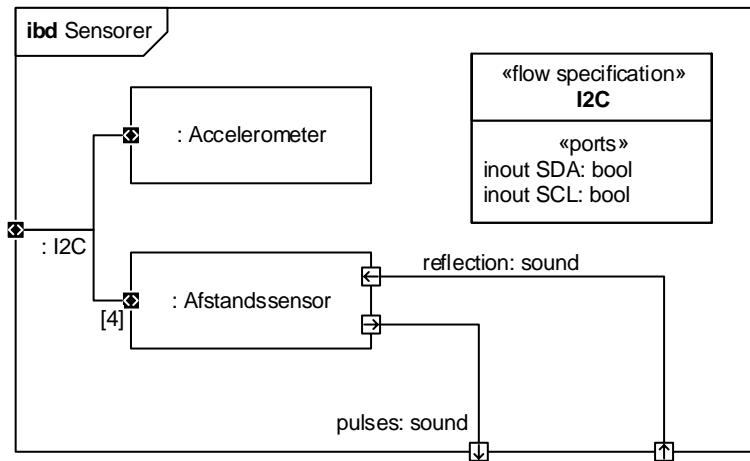
Figur 12: IBD for blokken PI

Signalbeskrivelse for Pi

Signal (navn: type)	Funktion	Tolerancer	Kommentarer
SDA: bool	I2C dataline til sensorer herunder accelerometer og afstandssensorer	0-5V±0.5V	Logisk signal: Lav= 0V±0.5V Høj= 7.2V±0.5V
SCL: bool	I2C clockline til sensorer herunder accelerometer og afstandssensorer	0-5V±0.5V	Logisk signal: Lav= 0V±0.5V Høj= 7.2V±0.5V
Image: light	Lysindfald til kamera-sensor	-	-
motorSpeed: PWM	Et PWM signal der bestemmer motorhastigheden.	Frekvens: 30kHz±1kHz 0-5V±0.2V	Logisk signal: Lav= 0V±0.2V Høj= 5V±0.2V
forward: bool	Kontrolsignal til H-bro	0-5V±0.2V	Logisk signal: Lav= 0V±0.2V "idle" Høj= 5V±0.2V "forward"
reverse: bool	Kontrolsignal til H-bro	0-5V±0.2V	Logisk signal: Lav= 0V±0.2V "idle" Høj= 5V±0.2V "back"
:USB	Serielforbindelse mellem Wi-fi dongle og Pi	VBUS = 5V±0.2V D- = 5V±0.2V D+ = 5V±0.2V GND = 0V	VBUS for Low-power port: Diff "1" (D+)-(D-) > 200 mV og D+>VIH (min) Diff "0" (D-)-(D+) > 200 mV og D->VIH (min)
imagestr: IPcam	Karsten... Dette er dit job	-	-
carControl: au2proto	Få fingeren ud Karsten.	-	-
: MIPI CSI-2	Camera serial interface	-	Se reference: [?]

3.3.5 Sensorer

På figur ?? ses de interne signaler for blokken Sensorer.



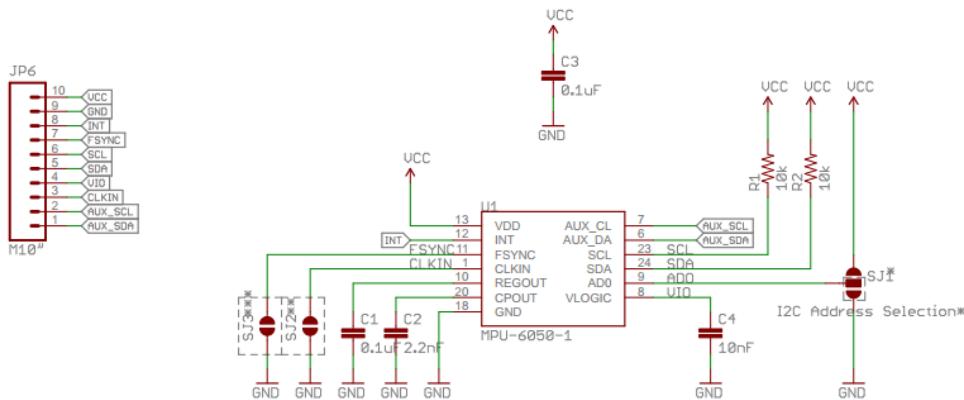
Figur 13: IBD for blokken sensorer

signalbeskrivelse for sensorer

Signal (navn: type)	Funktion	Tolerancer	Kommentarer
Inout SDA: bool	I ² C dataline til sensorer herunder accelerometer og afstandssensorer.	0-5V ± 0.5V	Logisk signal: Lav = 0V ± 0.5V Høj = 7.2V ± 0.5V
Inout SCL: bool	I ² C clockline til sensorer herunder accelerometer og afstandssensorer.	0-5V ± 0.5V	Logisk signal: Lav = 0V ± 0.5V Høj = 7.2V ± 0.5V
Pulses: sound	Ultralydsvagts afsendt af sensor.	Jfv. Datablad (henvisning kommer senere)	
reflection: sound	Refleksionsvagte af udsendte ultralydsvagter.	Jfv. Datablad (henvisning kommer senere)	

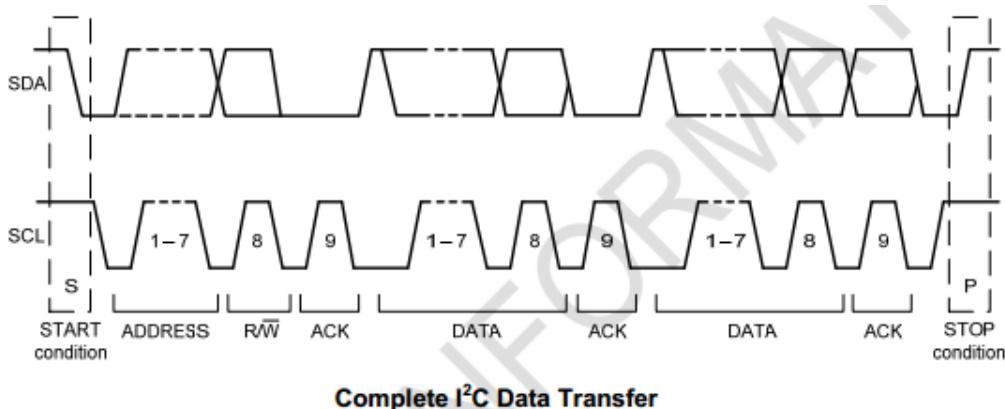
3.3.6 MPU-6050 Accelerometer/Gyroskop

MPU-6050 er en kombination af et accelerometer, et gyroskop, begge med 3 akser og et termometer. Det betyder for systemet at det er i stand til at registrere en ændring i acceleration og/eller orientering i alle retninger. Sensoren er blevet valgt til projektet på baggrund af et I²C interface, som derved kan tilkobles en samlet bus sammen med andre sensorer på bilen. Udover dette giver det konstruerede breakoutboard mulighed for nem tilslutning, men fortsat lille størrelse på sensoren. Sensoren fungerer altid som en slave med adressen 0b110100X, hvor X bliver bestemt af det logiske niveau på pin AD0, der som standard er lav.



Figur 14: MPU-6050 diagram

Diagrammet for sensoren er vist i figur ???. Maksimal bushastighed for MPU-6050 er 400kHz, men da der er andre sensorer som arbejder langsommere end MPU-6050 i systemet, passer den fint ind. Accelerometeret er en MEMS-type, hvor der er bygget mikroskopiske kondensatorer ind i chippen, som kan fjedre og bevæge sig, hvilket registreres som en ændring i kapacitans. Denne ændring kan omregnes til nogle brugbare værdier, og kan herfra anvendes til bl.a. retningsbestemmelse. Der er i alt 7 16-bits registre i sensoren, som hver især er tilknyttet til en ADC på hver akse, med undtagelse af register nr. 7, som er tilknyttet termometeret. Protokollen for kommunikation med sensoren ser således ud:



Figur 15: I²C protokol for MPU-6050

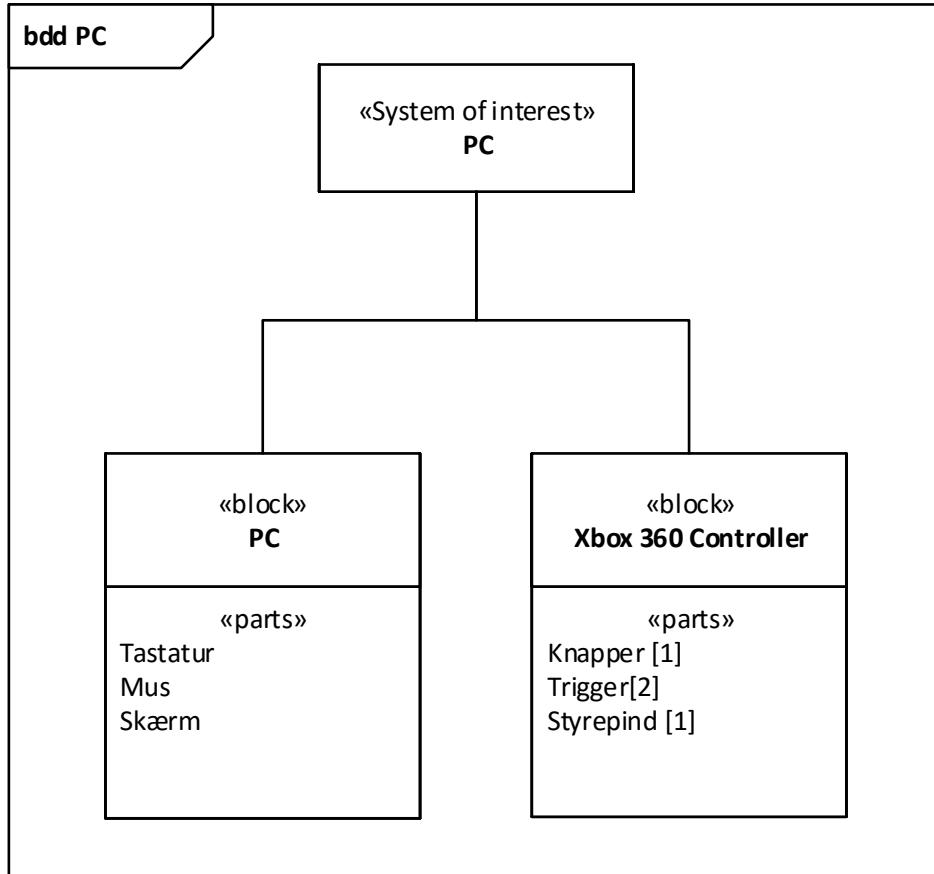
Som det ses i figur ???, starter masteren med at sætte en startsekvens ud på SDA (HIGH-to-LOW),

mens SCL er høj. Herefter betragtes bussen som optaget, indtil der bliver sendt en stopsekvens på SDA (LOW-to-HIGH) af masteren, mens SCL ligeledes er høj. Efter startsekvensen bliver der sendt en 7-bits adresse og en R/W bit. Data der bliver transmitteret over I²C bliver sendt i pakker af 8 bits. Når der først er sendt en startsekvens, er der ingen begrænsning på hvor meget data der må sendes, udover at der efter hver pakke, skal registreres en acknowledge. MPU-6050 indeholder desuden en DMP (Digital Motion Processor), som har til opgave at håndtere noget af dataprocesseringen fra selve MPU-6050.

3.4 PC

3.4.1 BDD for PC

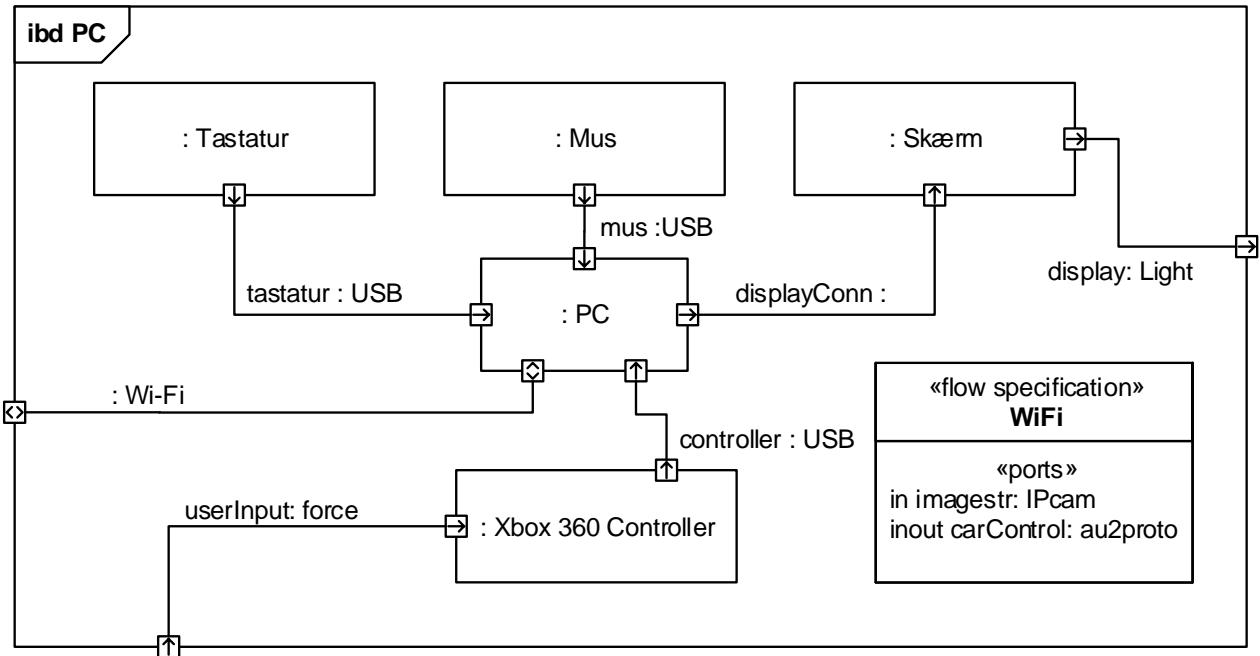
På figur ?? ses et overordnet blokdiagram for PC'en og dens interagerende dele.



Figur 16: BDD for PC kontekst

3.4.2 IBD for PC

På figur ?? ses alle interne signaler og en yderligere specificering af blokdiagrammet for pc på figur ??



Figur 17: IBD for blokken PC

3.4.3 Signalbeskrivelse for PC

Signal (navn: type)	Funktion	Tolerancer	Kommentarer
imagestr: IPcam	TODO Karsten		
carControl: au2proto	TODO Karsten		
tastatur: USB	Seriellkommunikation fra tastetur til PC	VBUS = 5V min. 4.40V max 5.25V GND = 0V D- = 5V +/- 0.2V D+ = 5V +/- 0.2V	VBUS for Low power port: Diff "1" $(D+) - (D-) > 200$ mV and D+ > VIH (min) Diff "0" $(D-) - (D+) > 200$ mV and D- > VIH (min)
mus: USB	Seriellkommunikation fra mus til PC	VBUS = 5V min. 4.40V max 5.25V GND = 0V D- = 5V +/- 0.2V D+ = 5V +/- 0.2V	VBUS for Low power port: Diff "1" $(D+) - (D-) > 200$ mV and D+ > VIH (min) Diff "0" $(D-) - (D+) > 200$ mV and D- > VIH (min)
displayConn:			
controller: USB	Seriellkommunikation fra Controller til PC	VBUS = 5V min. 4.40V max 5.25V GND = 0V D- = 5V +/- 0.2V D+ = 5V +/- 0.2V	VBUS for Low power port: Diff "1" $(D+) - (D-) > 200$ mV and D+ > VIH (min) Diff "0" $(D-) - (D+) > 200$ mV and D- > VIH (min)
userInput: force	Mekanisk input fra brugeren.		Dette input dækker over alle tryk/træk på Xbox 360 Controllerens knapper/styrepinde/triggers.
display: Light	Lys fra skærmen		Brugeren aflæser systemets interface via denne port.

3.5 Protokolbeskrivelse

3.5.1 Kamera

Kameraet er påmonteret Pi'en og der er installeret en virtuel driver kaldet uv4l. uv4l er en overbygning på den generelle driver således programmet motion kan finde kamaret i /dev/video0. Motion har den fordel at det kan sætte et IP videostream op hvorved Pi'en kan stremme videodata den vej igennem. På PC'en modtages videoen igennem GUI'en ved hjælp af opensource biblioteker fra VLC media player. Herved laves der en socket til at modtage videoen i. Adressen hvor i video stream modtages er Pi'ens Ip-adresse port 8081.

Kommando	Svar	Beskrivelse	Bitmønstre
openPlayer	Kontinuert stream fra motion	Motion streamer video data til PC'en	UDP socket forbindelse

Tabel 13: Kamera Protokol

Motion streamer video ligeså snart der er tændt for bilen. Der er derfor ikke nogen direkte kommunikation mellem PC'en og Pi'en igennem Kameraprotoollen.

3.5.2 GUI

PC'en og bilen kommunikerer igennem to TCP forbindelser. Xbox360 Controlleren har en TCP forbindelse hvor controller data bliver sendt kontinuert. GUI'en har en anden TCP forbindelse hvor data også udveksles kontinuert. Data består af makshastighed, afstand til forhindring, AKS-status, styretøj, acceleration og hastighed på bilen. TCP forbindelse til GUI'en sker på Pi'ens Ip-adresse port 1234 og controlleren har port 1235. I tabel ?? ses sekvensen over hver byte i GUI protokollen. Her er beskrevet hvilket bytenummer hver data byte har, samt enheden er. DB1 er fx. Hastighed som sendes i enheden $(\text{km}/\text{h}) \times 10$. Dette gøres for at vi derfor på PC siden kan modtage hastigheden fra 0-255 km/h. Når kommandoen shutDown sendes benytter den sig af samme protokol. Indholdet at bytes er i rækkefølgen "dwnnow"

Kommando	Svar	Beskrivelse	Bitmønstre
getData	sendStatus	GUI'en udveksler dens data med bilens	Ved getData sendes data som char array i rækkefølgen: makshastighed(km/h), hastighed((km/h) $\times 10$), afstand(m $\times 10$), acceleration(g $\times 10$), AKS-status(1/0) og styretøj(-50..50).
shutDown	shuttingDown	GUI'en sender besked til bilen om at den skal lukke dens software ned	shutDown sendes som en string "dwnnow" bilen sender "dwnnow" tilbage som ACK. Alt andet tolkes som NACK

Tabel 14: GUI Protokol

	DB0	DB1	DB2	DB3	DB4	DB5
Datatype	Makshastighed	Hastighed	Afstand	Acceleration	AKS-status	Styretøj
Enhed	km/h	(km/h)×10	m×10	G×10	0..1	-50..50

Tabel 15: GUI byte protokol

	DB0	DB1	DB2	DB3	DB4	DB5
Datatype	Shutdown	Shutdown	Shutdown	Shutdown	Shutdown	Shutdown
Enhed	'd'	'w'	'n'	'n'	'o'	'w'

Tabel 16: GUI shutdown protokol

På tabel ?? ses protokollen over Xbox360-controlleren.

Kommando	Svar	Beskrivelse	Bitmønstre
controllerFrem	<i>Intet svar</i>	Bruger har trykket på RT på controlleren	Unsigned char fra 0-255 afhængig af hvor hårdt der trykkes på knappen
controllerTilbage	<i>Intet svar</i>	Bruger har trykket på LT på controlleren	Unsigned char fra 0-255 afhængig af hvor hårdt der trykkes på knappen
controllerDrej	<i>Intet svar</i>	Bruger har ændret positionen af venstre styrepind, til højre eller venstre, på controlleren	Char fra -127 til 127 afhængig af positionen af styrepinden
controllerStop	<i>Intet svar</i>	Bruger har trykket på X på controlleren	Char værdi 1

Tabel 17: Xbox360-controller Protokol

	DB0	DB1	DB2	DB3
Datatype	Frem	Tilbage	Drej	Stop
Enhed	0..255 unsigned char	0..255 unsigned char	(-127)..127 char	0..1 char

Tabel 18: Xbox360-controller Protokol byte mønster

4 Hardwaredesign

Version

Dato	Version	Initialer	Ændring
xxx	1	Alle	Første udkast.

4.1 Strømforsyning

For at dimensionere strømforsyningen er der i første omgang udarbejdet et estimat for hvor meget strøm de forskellige blokke i bilen forbruger. Resultater af disse udregninger er indsat i tabel ?? på side ??.

Strømforsyningen er som udgangspunkt designet som en buck converter [?, afsnit 6.10]. Til at realisere dette er en TI LM26003 taget i brug som central enhed. I databladet for LM26003 [?] fremgår et typisk design, som stemmer godt overens med behovene for dette projekt.

Til selve dimensioneringen af kredsløbet er fremgangsmåden, der fremkommer i databladets punkt 9.2.1 fulgt. Der er lavet en mindre modifikation ift. ??, "Typical Application", for at give strømforsyningen en 3V udgang.

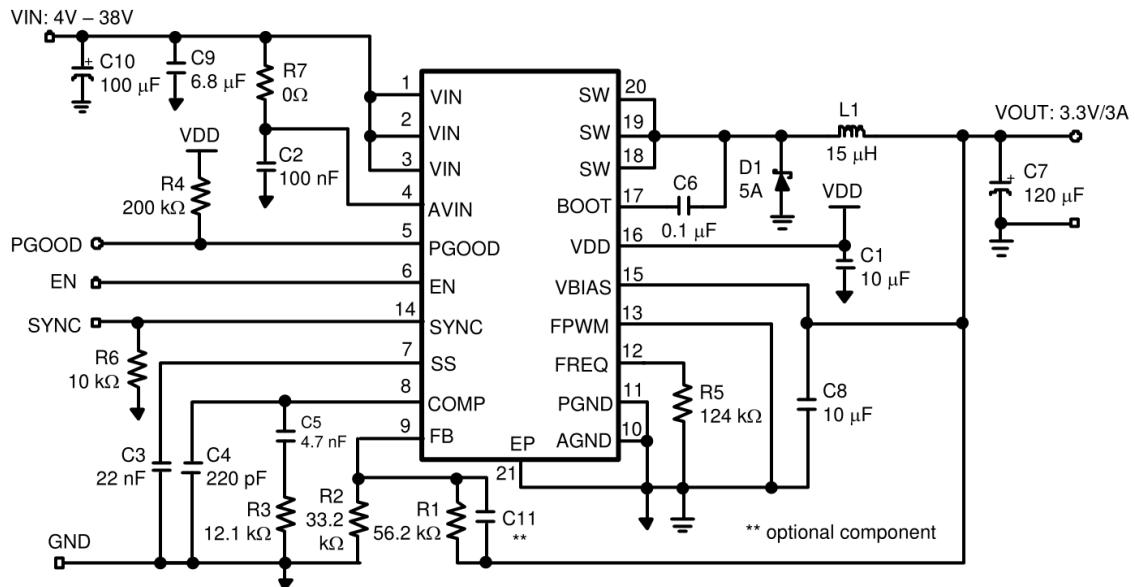
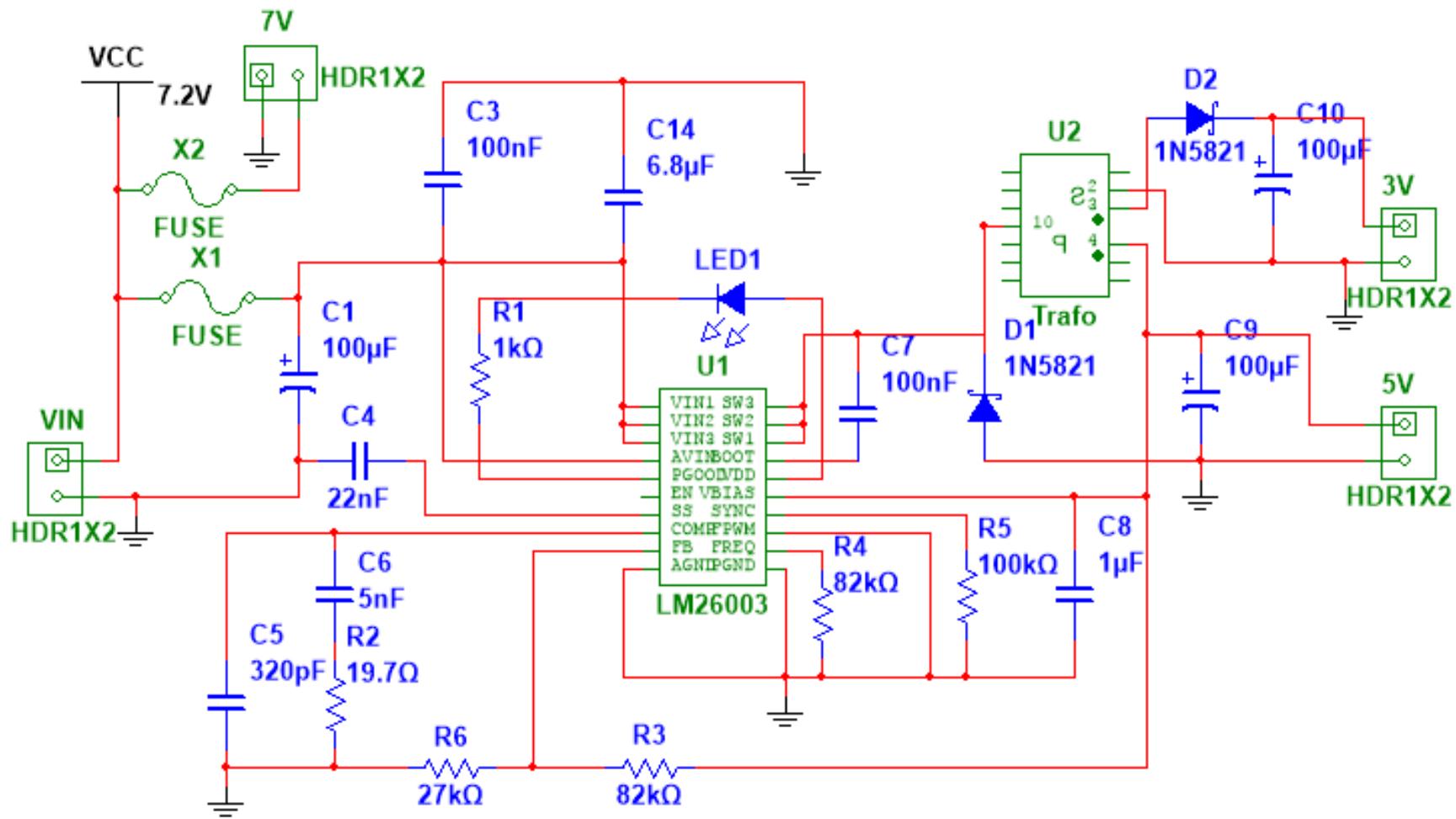


Figure 16. Example Circuit 3A, 300 kHz

Figur 18: Typisk kredsløb fra databladet for LM26003

I forbindelse med L1 er der på samme kerne viklet viklinger på for at danne en transformator, hvorved en 3V udgang er lavet. Denne ensrettes med en diode og filtreres med en kondensator.

U2 på figur ?? består af den førstnævnte egenviklede transformator, som er viklet på en axial spoleform. Denne kan ses i databladet for den anvendte spolekerne [?]. Samtlige beregninger for strømforsyningen fremgår i bilaget [?]. For at overholde kravene om en lav ESR, ækvivalent seriemarkant, for kondensatorerne C1, C9 og C10 er disse bestilt udefra, da de elektrolytkondensatorer, som forefindes på skolens komponentlager har langt for høj ESR. En for høj ESR vil medføre at polen for det pågældende lavpasfilter lægges for langt ned i frekvensen til at fjerne dynamikken i det pågældende signal.



Figur 19: Kredsløbsdiagram for strømforsyning i bilen

5 Softwaredesign

Version

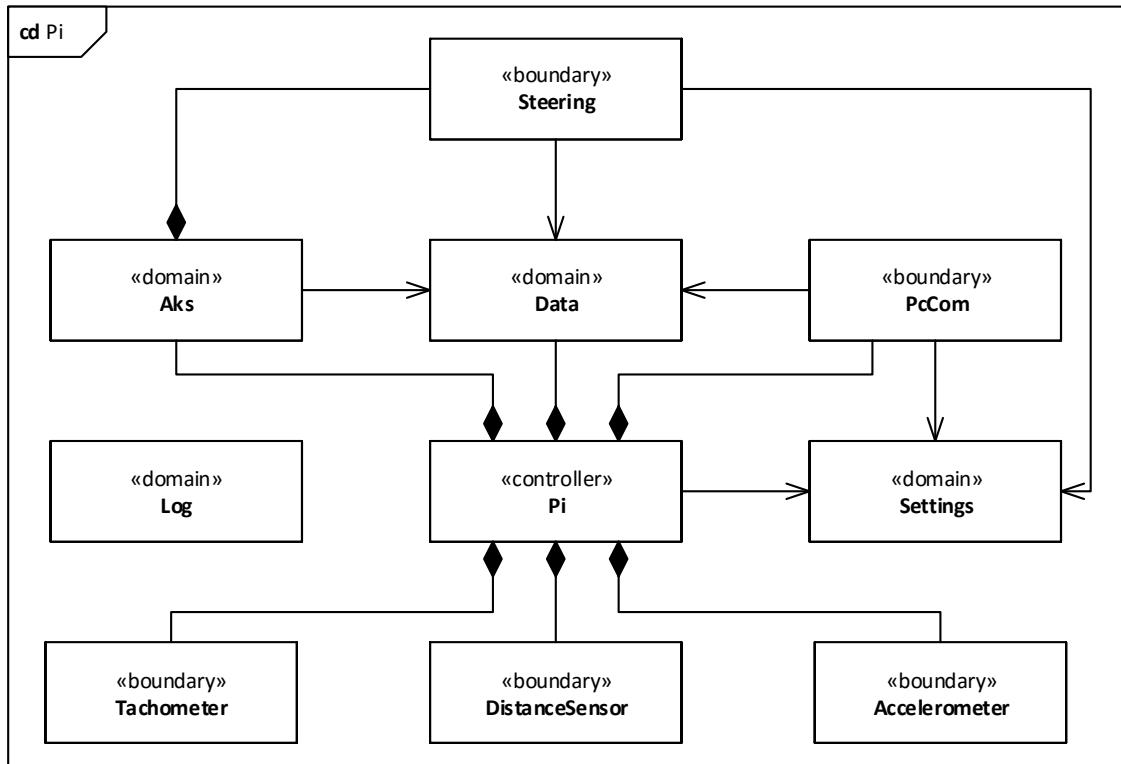
Dato	Version	Initialer	Ændring
xxx	1	-	Første udkast.
3. december	2	PKP	Opdateret data-klasse beskrivelse og tilføjet log-klasse beskrivelse.

5.1 Bil

5.1.1 DistanceSensor

5.1.2 Klassediagram

I dette afsnit beskrives det overordnede design på den software der kommer til at ligge på Pi. På figur ?? ses et klassediagram der opdeler funktionaliteten i klasser.



Figur 20: Klasse diagram over Pi

Controller-klasse: Pi

Controller-klassen Pi indeholder main funktionen og har derfor ansvaret for at styre slagets gang. Klassen skal derfor iværksætte initialisering af alle de klasser som den har ejerskab over. En af klassens ansvarsområder er at indsamle data fra sensorerne, og dette gøres ved at starte en særskilt tråd til dette. Denne tråd skal også sørge for at iværksætte Aks-klassen hver gang nye data er indsamlet.

Domain-klasse: Aks

Domain-klassen analyserer indkomne sensordata og i tilfælde at bilen er ved at køre ind i en forhindring, blokeres brugerinput og der styres udenom eller bremses.

Domain-klasse: Data

Denne klasse har til formål at indsamle alle sensordata i en datastruktur. Disse data gemmes i memory kan ikke overstige en defineret størrelse. Brugerinput gemmes ikke i denne klasse.

Domain-klasse: Log

Denne klasse har til formål at gemme samtlige systemhændelser i den fil, så kilden til eventuelle programcrash kan identificeres. Alle klasser på Pi har en reference til denne log, så de hver i sær kan skrive til den. På figur ?? er der undladt at lave pile fra alle klasser til denne, da dette vil gøre diagrammet uoverskueligt.

Domain-klasse: Settings

Settings er datastruktur der indeholder indstillinger for maksimal hastighed, AKS status, og styretøjets kalibrering. Indstillingerne er gemt i en fil som kan tilgås af Pi-klassen og Steering-klassen.

Boundary-klasse: PcCom

Boundary-klassen PcCom håndterer kommunikationen imellem PC og Bil. Denne kommunikation sker vha. UDP via Wi-Fi.

Boundary-klasse: Steering

I denne klasse styres bilens aktuatorer. Dette er altså en driver til både motoren der skaber fremdrift og servo-motoren der styrer forhjulene. Klassen tager ligeledes højde for brugers indstillinger.

Boundary-klasse: Tachometer

Denne klasse håndterer kommunikationen til bilens tachometer og konverterer sensordata til brugbar hastighedsmåling.

Boundary-klasse: DistanceSensor

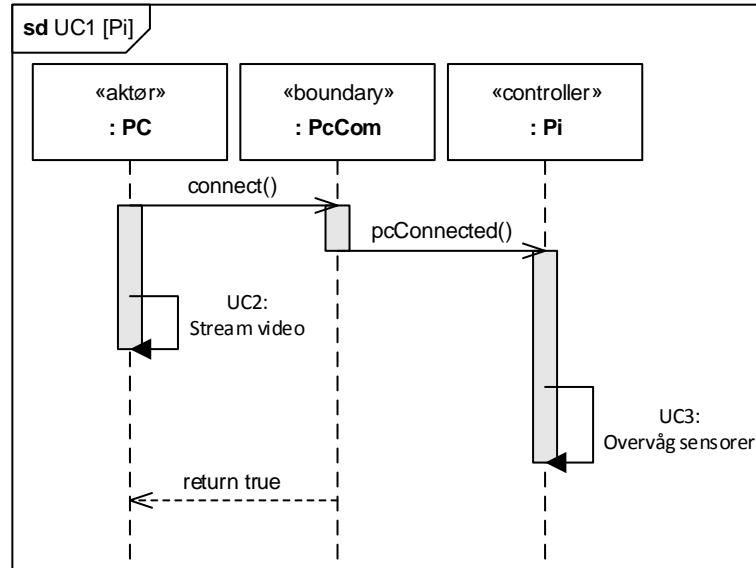
Denne klasse håndterer kommunikationen til bilens afstandssensorer og konverterer sensordata til brugbar distancemålinger. Klassen håndterer alle fire sensorer.

Boundary-klasse: Accelerometer

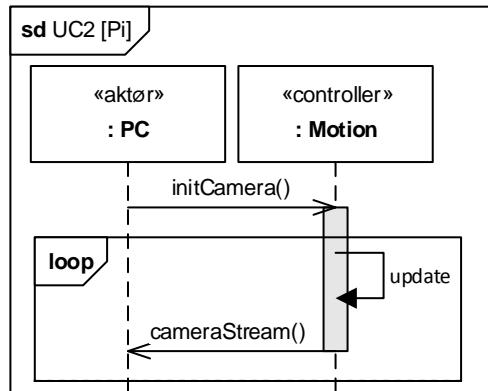
Denne klasse håndterer kommunikationen til bilens accelerometer og konverterer sensordata til brugbar g-måling.

5.1.3 Sekvensdiagrammer

Herunder er udarbejdet sekvensdiagrammer for den funktionalitet som Pi blokken på bilen skal have. Der er tage udgangspunkt i de tidligere fremstillede use cases. Sekvensdiagrammer for UC8 og UC12 er udeladt, da disse kun indeholder handlinger over interaktion imellem bruger og PC.

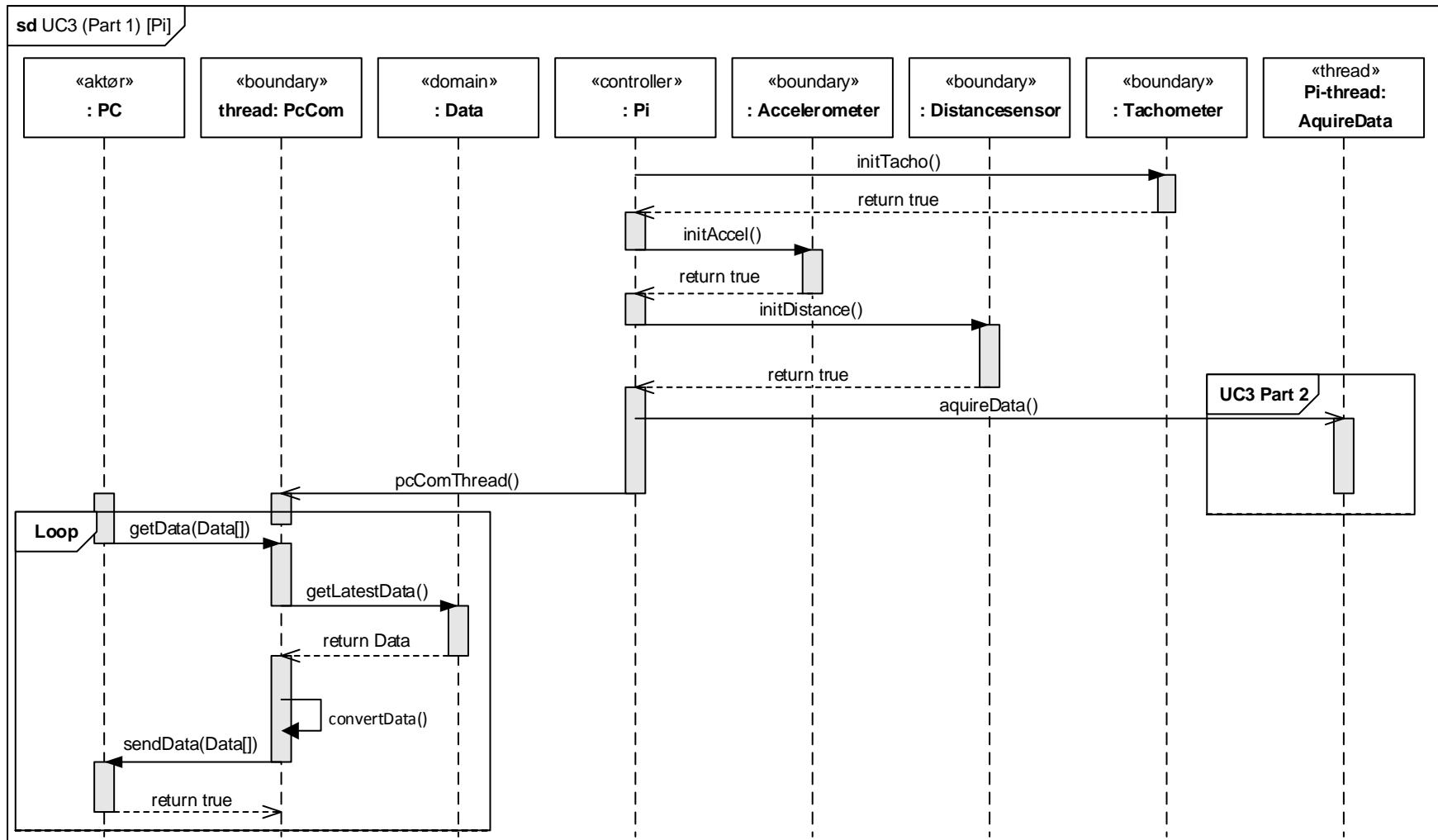


Figur 21: Sekvensdiagram over bilens funktionalitet i UC1: Aktiver system

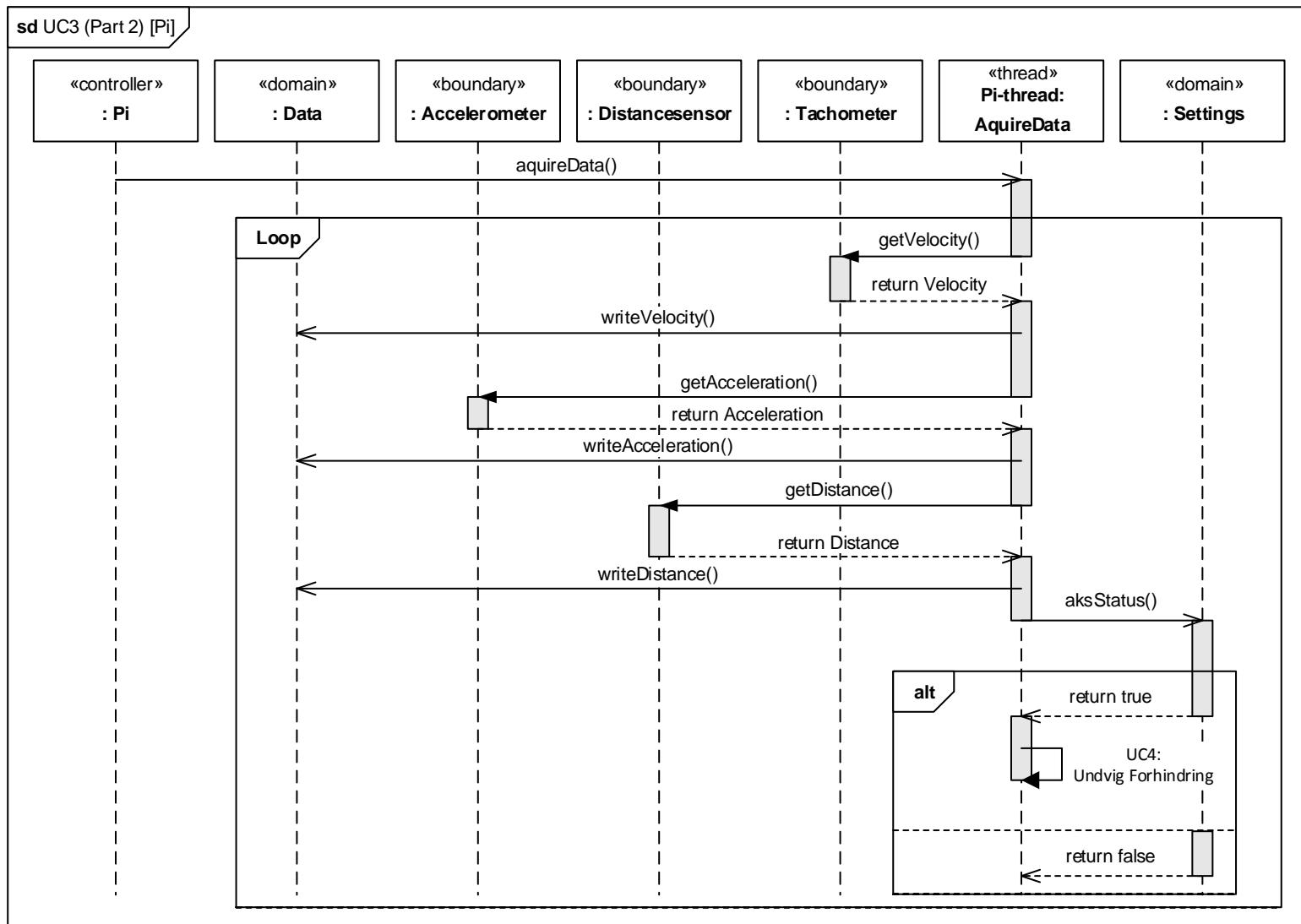


Figur 22: Sekvensdiagram over bilens funktionalitet i UC2: Stream video

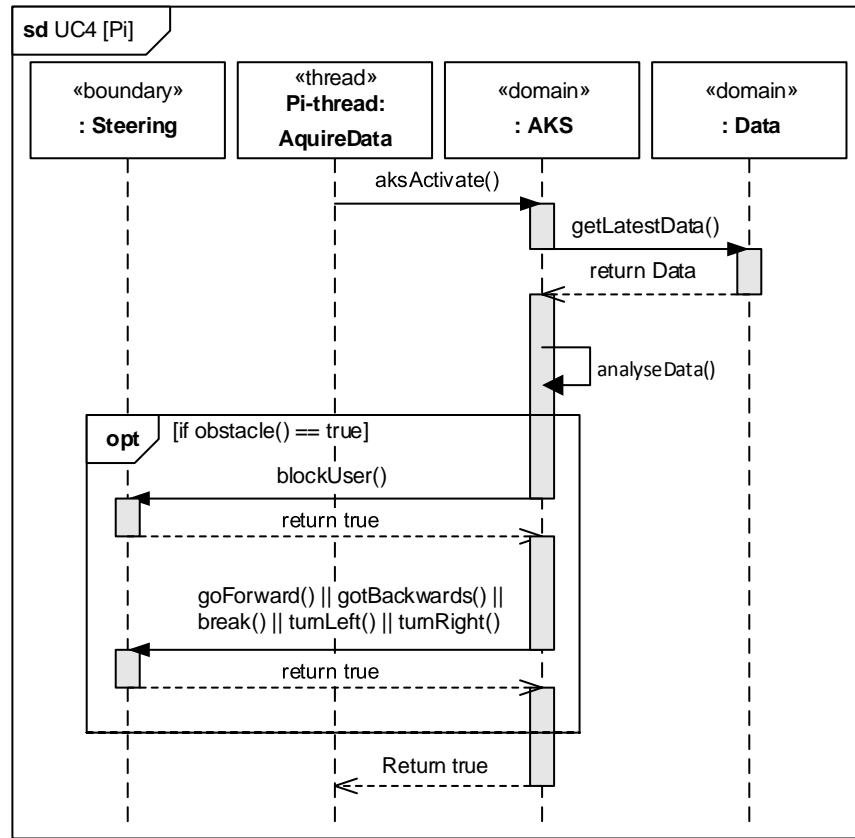
5.1. Bil



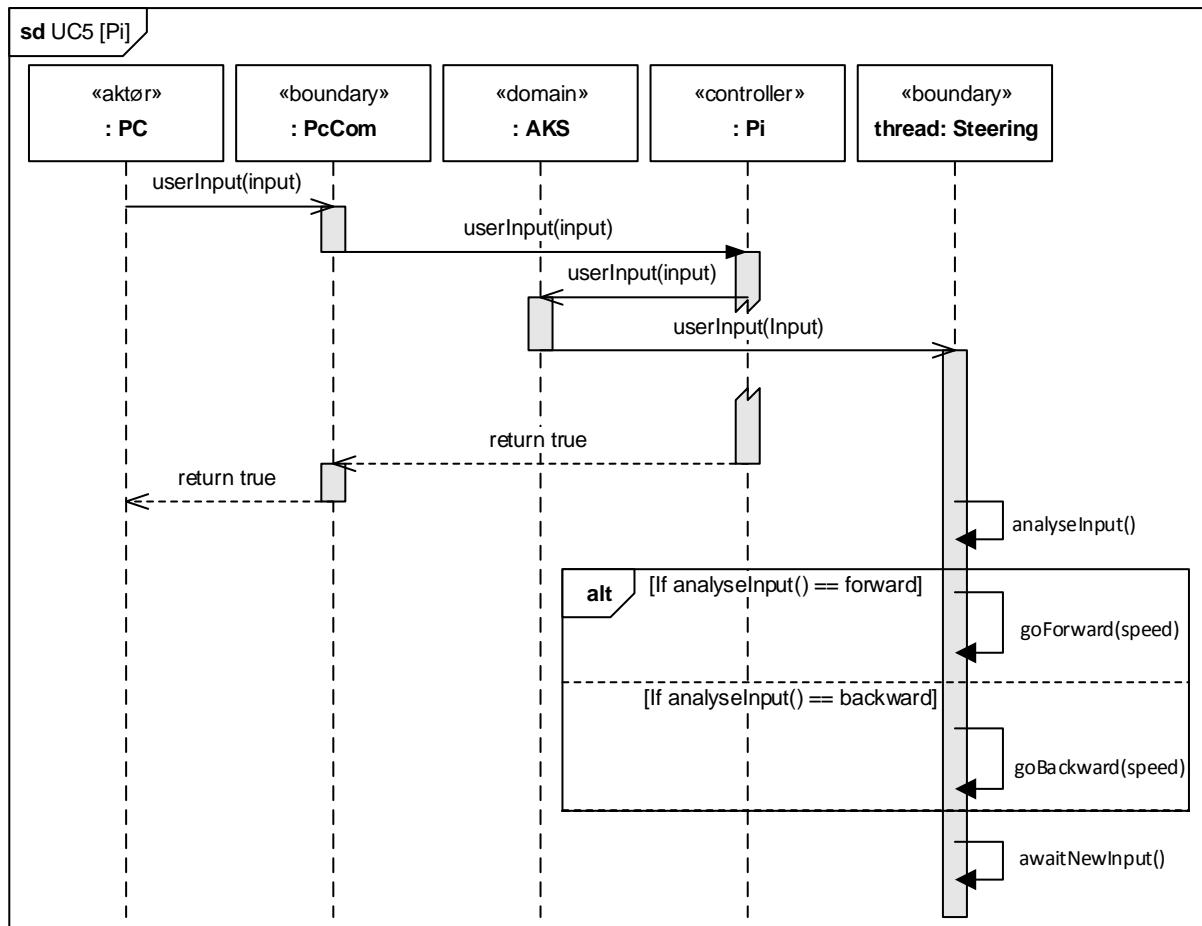
Figur 23: Sekvensdiagram over bilens funktionalitet i UC3: Overvåg sensorer - Del 1



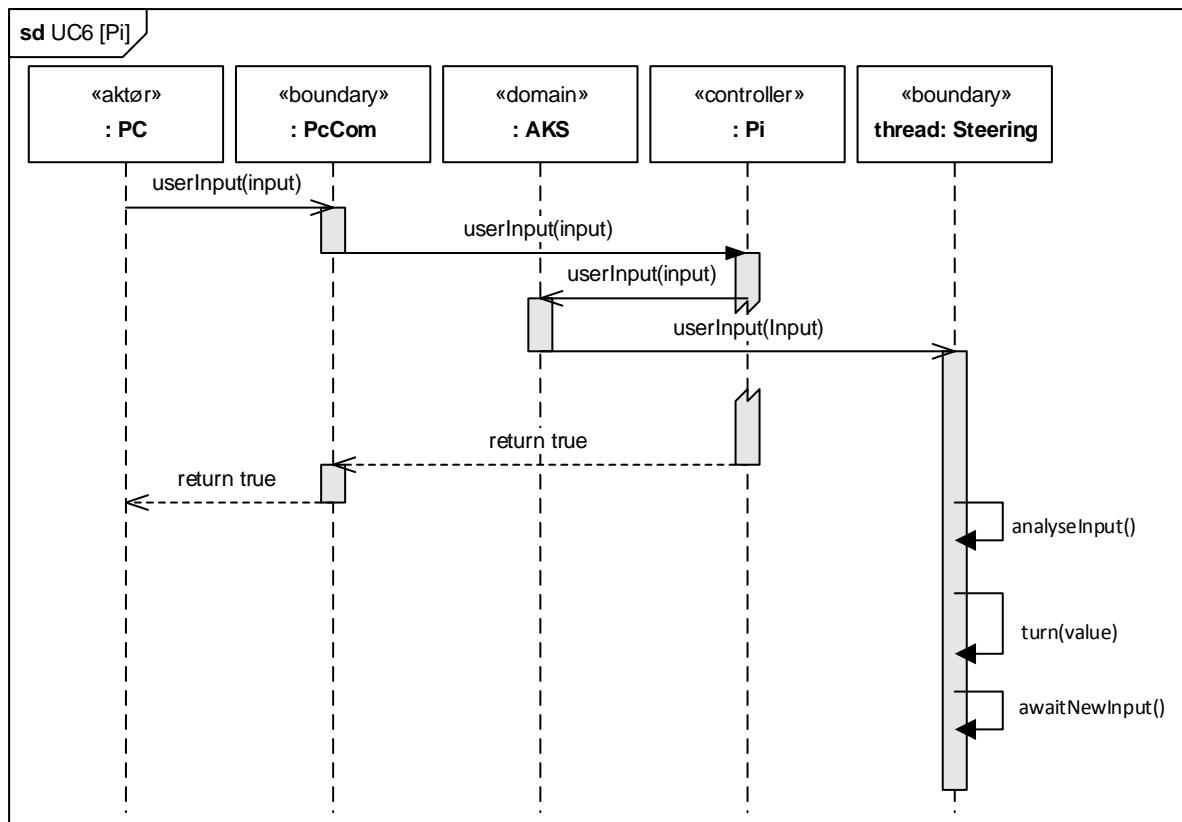
Figur 24: Sekvensdiagram over bilens funktionalitet i UC3: Overvåg sensorer - Del 2



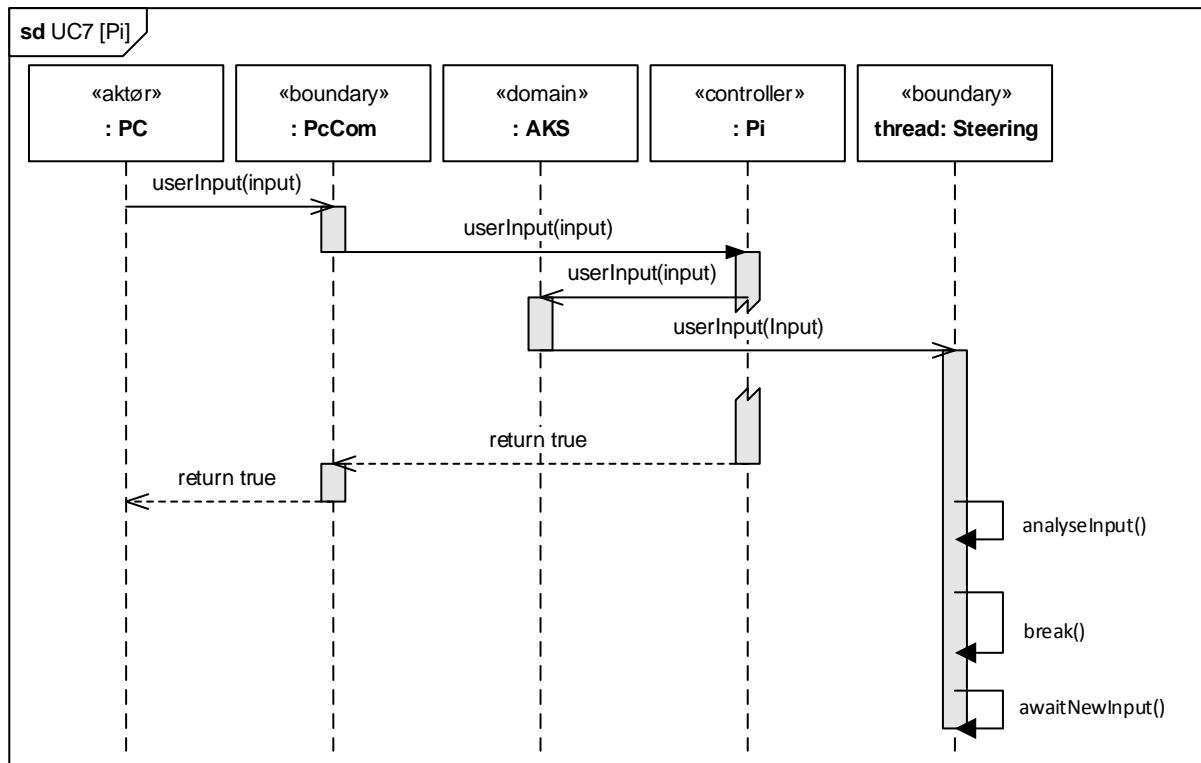
Figur 25: Sekvensdiagram over bilens funktionalitet i UC4: Undvig forhindring



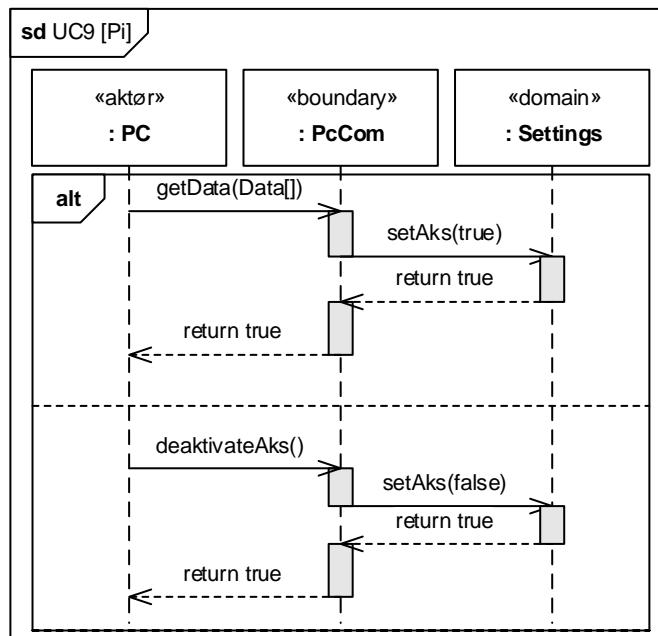
Figur 26: Sekvensdiagram over bilens funktionalitet i UC5: Kør bil frem/tilbage



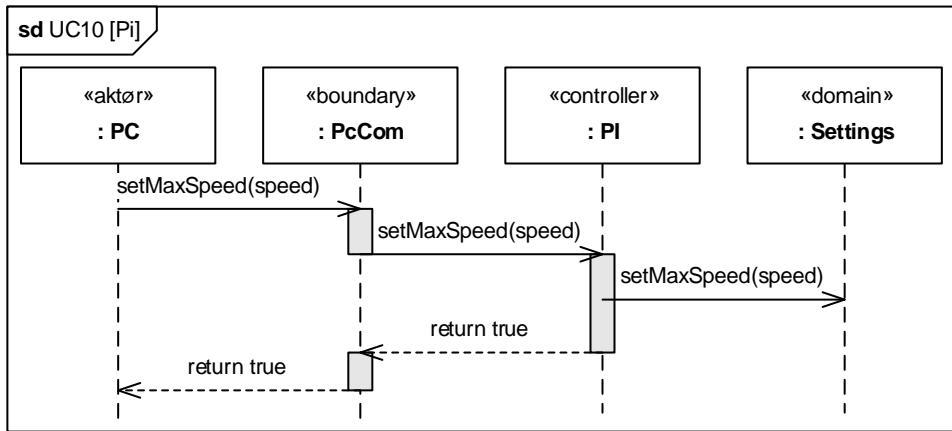
Figur 27: Sekvensdiagram over bilens funktionalitet i UC6: Drej til højre/venstre



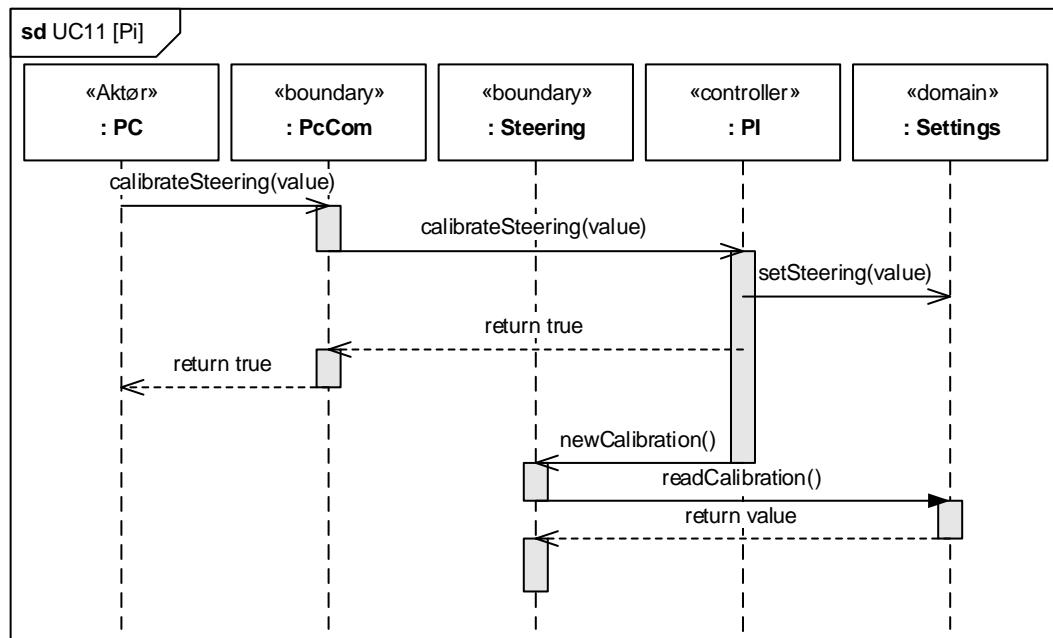
Figur 28: Sekvensdiagram over bilens funktionalitet i UC7: Brems bil



Figur 29: Sekvensdiagram over bilens funktionalitet i UC9: Tænd/sluk AKS



Figur 30: Sekvensdiagram over bilens funktionalitet i UC10: Indstil makshastighed

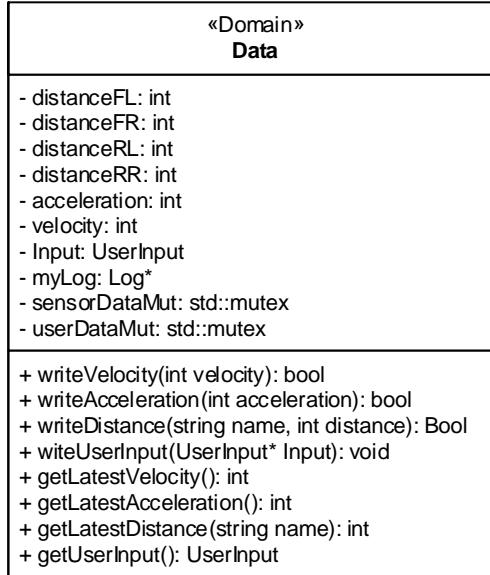


Figur 31: Sekvensdiagram over bilens funktionalitet i UC11: Kalibrer styretøj

5.1.4 Klassebeskrivelser

Domain-klasse: Data

Denne klasse fungerer som en datastruktur der indeholder den data, der er blevet indsamlet fra sensorer og brugerens input der er blevet sendt via PcCom-klassen. Data'en indlægges ved at kalde write-metoderne og læses ved at kalde get-metoderne. Klassen benyttes af flere forskellige parallelle tråde og skal derfor beskytte sine attributter ved brug af `std::mutex`.



Figur 32: Klassebeskrivelse for domain-klassen Data

Attributter

Navn	Type	Beskrivelse
distanceFL	int	Variabel der indeholder afstanden til en evt. forhindring ved forreste venstre hjørne af bilen i cm.
distanceFR	int	Variabel der indeholder afstanden til en evt. forhindring ved forreste højre hjørne af bilen i cm.
distanceRL	int	Variabel der indeholder afstanden til en evt. forhindring ved bageste venstre hjørne af bilen i cm.
distanceRR	int	Variabel der indeholder afstanden til en evt. forhindring ved bageste højre hjørne af bilen i cm.
acceleration	int	Variabel der indeholder bilens aktuelle acceleration i $G/10$.
velocity	int	Variabel der indeholder bilens aktuelle hastighed i $(km/t)/10$.
Input	UserInput	En struct der indeholder userinput til systemet. Denne struct er specifiseret i utilities.hpp.
myLog	Log*	En pointer til systemets log.
sensorDataMut	std::mutex	En mutex der anvendes som lås til de variable der skal indeholde data fra alle bilens sensorer.
userDataMut	std::mutex	En mutex der anvendes som lås til de variable der skal indeholde data om brugerens input.

Tabel 19: Attributter for klassen Data

Metoder

Prototype	<code>bool writeVelocity(int velocity)</code>
Parametre	<code>velocity</code> Den hastighed der skal indlæses.
Returværdi	<code>bool</code> Returnerer <code>true</code> hvis skrivningen er gået godt og <code>false</code> hvis skrivningen gik galt.
Beskrivelse	Metoden indlæser den nyeste værdi af hastigheden i datastrukturen.

Tabel 20: Metodebeskrivelse for `writeVelocity`

Prototype	<code>bool writeAcceleration(int acceleration)</code>
Parametre	<code>acceleration</code> Den acceleration der skal indlæses.
Returværdi	<code>bool</code> Returnerer <code>true</code> hvis skrivningen er gået godt og <code>false</code> hvis skrivningen gik galt.
Beskrivelse	Metoden indlæser den nyeste værdi af accelerationen i datastrukturen.

Tabel 21: Metodebeskrivelse for `writeAcceleration`

Prototype	<code>bool writeDistance(string name, int distance)</code>
Parametre	<code>name</code> Navnet på den sensor som der skal indlæses data fra. Kan være FL, FR, RL eller RR. <code>distance</code> Afstanden der fra den pågældende sensor der skal indlæses i datastrukturen.
Returværdi	<code>bool</code> Returnerer <code>true</code> hvis skrivningen er gået godt og <code>false</code> hvis skrivningen gik galt.
Beskrivelse	Metoden indlæser den nyeste værdi af fra en vilkårlig afstandssensor i datastrukturen.

Tabel 22: Metodebeskrivelse for `writeDistance`

Prototype	<code>void writeUserInput(UserInput* Input)</code>
Parametre	<code>Input</code> En pointer med den data som der skal gemmes i dataklassen.
Returværdi	<code>void</code>
Beskrivelse	Metoden indlæser den nyeste værdi af brugerens input i datastrukturen.

Tabel 23: Metodebeskrivelse for `writeUserInput`

Prototype	<code>int getLatestVelocity()</code>
Parametre	
Returværdi	<code>int</code> Den nyeste hastighedsmåling.
Beskrivelse	Metoden returnerer den nyeste hastighedsmåling der er indlæst i datastrukturen.

Tabel 24: Metodebeskrivelse for `getLatestVelocity`

Prototype	<code>int getLatestAcceleration()</code>
Parametre	
Returværdi	<code>int</code> Den nyeste accelerationsmåling.
Beskrivelse	Metoden returnerer den nyeste accelerationsmåling der er indlæst i datastrukturen.

Tabel 25: Metodebeskrivelse for `getLatestAcceleration`

Prototype	<code>int getLatestDistance(string name)</code>
Parametre	<code>name</code> Navnet på den sensor som der ønskes data fra. Kan være FL, FR, RL eller RR.
Returværdi	<code>int</code> Den nyeste afstandsmåling fra den angivne sensor.
Beskrivelse	Metoden returnerer den nyeste afstandsmåling fra den angivne sensor.

Tabel 26: Metodebeskrivelse for `getLatestDistance`

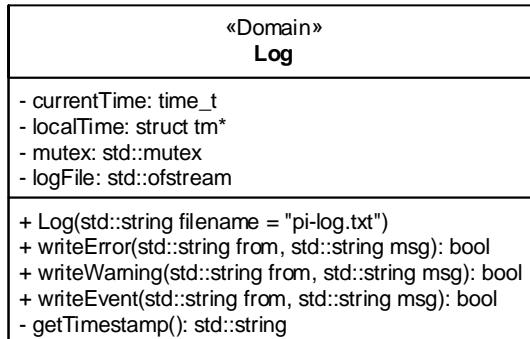
Prototype	<code>UserInput getUserInput()</code>
Parametre	
Returværdi	<code>UserInput</code> Den nyeste instans af brugerens indput til systemet.
Beskrivelse	Metoden returnerer de nyeste brugerinput.

Tabel 27: Metodebeskrivelse for `getUserInput`

Domain-klasse: Log

Loggen har til formål at kunne lokalisere og identificere fejl i systemet. Loggen skal oprettes mere eller mindre globalt og der skal efterfølgende medgives en pointer til samtlige klasser på Pi. Alle disse klasser skal således anvende loggen som debugging redskab. Der skal som udgangspunkt kun skrives i loggen hvis en fejl opstår, da loggen ellers bliver uoverskuelig. Når der skrives til loggen, anvender den pågældende tråd cpu-tid, hvilket ligeledes er en grund til at være opmærksom på hvornår det er smart at skrive til loggen.

For at forhindre at log-entries fra forskellige tråde sammenflettes, skal der i implementeringen anvendes `std::mutex` som lås når der skrives i loggen.



Figur 33: Klassebeskrivelse for domain-klassen Log

Attributter

Navn	Type	Beskrivelse
<code>currentTime</code>	<code>time_t</code>	Denne attribut bruges til at gemme det nuværende tidspunkt, når <code>getTimestamp</code> kaldes.
<code>localTime</code>	<code>struct tm*</code>	Anvendes til at holde tiden i et læseligt format.
<code>mutex</code>	<code>std::mutex</code>	Anvendes som lås i <code>std::lock_guard</code> der forhindrer flere tråde i at skrive i loggen på samme tid.
<code>logFile</code>	<code>std::ofstream</code>	File descriptor til logfilen.

Tabel 28: Attributter for klassen Log

Constructor

Prototype	<code>Log(std::string filename = "pi-log.txt")</code>
Parametre	<code>filename</code> Det ønskede filnavn til logfilen der oprettes. Hvis denne parameter udelades ved initialiseringen bliver objektet oprettet med filnavnet "pi-log.txt".
Beskrivelse	Constructoren opretter et objekt af Log klassen med det angivne filnavn.

Tabel 29: Beskrivelse af constructor for Log

Metoder

Prototype	<code>bool writeError(std::string from, std::string msg)</code>
Parametre	<p>from Denne streng skal udfyldes med den indbyggede identifier <code>"__PRETTY_FUNCTION__"</code> (GNU standard) der returnerer placeringen af det pågældende kald. Herved kan det i loggen ses præcis i hvilken klasse og metode log-beskeden kommer fra.</p> <p>msg Den besked der skal stå i loggen.</p>
Returværdi	<code>bool</code> Returnerer <code>true</code> hvis skrivningen er gået godt og <code>false</code> hvis skrivningen gik galt.
Beskrivelse	Metoden skriver en besked i loggen. Anvendes når der er sket en alvorlig fejl.

Tabel 30: Metodebeskrivelse for `writeError`

Prototype	<code>bool writeWarning(std::string from, std::string msg)</code>
Parametre	<p>from Denne streng skal udfyldes med den indbyggede identifier <code>"__PRETTY_FUNCTION__"</code> (GNU standard) der returnerer placeringen af det pågældende kald. Herved kan det i loggen ses præcis i hvilken klasse og metode log-beskeden kommer fra.</p> <p>msg Den besked der skal stå i loggen.</p>
Returværdi	<code>bool</code> Returnerer <code>true</code> hvis skrivningen er gået godt og <code>false</code> hvis skrivningen gik galt.
Beskrivelse	Metoden skriver en besked i loggen. Anvendes når der er sket en mindre alvorlig fejl.

Tabel 31: Metodebeskrivelse for `writeWarning`

Prototype	<code>bool writeEvent(std::string from, std::string msg)</code>
Parametre	<p><code>from</code> Denne streng skal udfyldes med den indbyggede identifier ”<code>__PRETTY_FUNCTION__</code>” (GNU standard) der returnerer placeringen af det pågældende kald. Herved kan det i loggen ses præcis i hvilken klasse og metode log-beskeden kommer fra.</p> <p><code>msg</code> Den besked der skal stå i loggen.</p>
Returværdi	<code>bool</code> Returnerer <code>true</code> hvis skrivningen er gået godt og <code>false</code> hvis skrivningen gik galt.
Beskrivelse	Metoden skriver en besked i loggen. Anvendes når der er hændelse, der ikke er en fejl, som skal skrives i loggen.

Tabel 32: Metodebeskrivelse for `writeEvent`

Prototype	<code>std::string getTimestamp()</code>
Parametre	
Returværdi	<code>std::string</code> Returnerer en streng med systemets indstillede tid og dato. Formatet er <code>ÅÅÅÅ-MM-DD TT:MM:SS</code> .
Beskrivelse	Metoden henter den nuværende tid i systemet og omdanner denne til en streng i en format der nemt kan anvendes i log-beskeder.

Tabel 33: Metodebeskrivelse for `getTimestamp`

Domain-klasse: Settings

Denne klasses formål er at gemme de nyeste data af hhv. bilens maksimale hastighed, kalibrering af styrtøj samt AKS status. Alle disse værdier har brugeren mulighed for at indstille fra PC'en. Disse data skal ligeledes være tilgængelige for andre klasser, der har brug for en eller flere af disse data. Da der arbejdes med mange threads i systemet, er det vigtigt at det ikke er muligt at skrive og læse fra de samme data af to forskellige threads på samme tid, til dette udnyttes mutex låse. På Figur ?? kan ses et klasse diagram over den ønskede klasse efterfulgt af funktionalitet.

«Domain» Settings	
<hr/>	
- maxVelocity_:	int
- AKSStatus_:	bool
- calibration_:	char
- logClassPtr_:	Log*
- maxVelocityMut:	mutex
- AKSStatusMut:	mutex
- calibrationMut:	mutex
<hr/>	
+ Settings(Log* logClassPtr):	void
+ setMaxSpeed(int maxVelocity):	void
+ setAKS(bool AKSStatus):	void
+ calibrateSteering(char calibration):	void
+ getMaxSpeed():	Int
+ getAKS():	bool
+ getCalibration():	char

Figur 34: Klassebeskrivelse for domain-klassen Settings

Attributter

Navn	Type	Beskrivelse
maxVelocity_	int	Variabel af typen int til indhold af nuværende maksimal hastighed på bil.
AKSStauts_	bool	Variabel af typen bool til indhold af nuværende tilstand af AKS.
calibration_	char	Variabel af typen char der indeholder nuværende kalibrering til styrtøj.
logClassPtr_	Log*	En pointer til et object af typen Log. Gør det muligt at kalde funktioner fra Log objektet.
maxVelocitMut	mutex	Mutex til kontrol af læsning og skrivning af/til maxVelocity_ variabel fra flere threads.
AKSStautsMut	mutex	Mutex til kontrol af læsning og skrivning af/til AKSStatus_ variabel fra flere threads.
calibrationMut	mutex	Mutex til kontrol af læsning og skrivning af/til calibration_ variabel fra flere threads.

Tabel 34: Attributter for klassen Settings

Metoder

Prototype	<code>void Settings(Log* logClassPtr)</code>
Parametre	<code>logClassPtr</code> En pointer til objektet af typen Log der bruges til at lave en association til objektet.
Beskrivelse	Constructor til klassen Settings.

Tabel 35: Metodebeskrivelse for constructoren af `Settings` klassen

Prototype	<code>void setMaxSpeed(int maxVelocity)</code>
Parametre	<code>maxVelocity</code> Den hastighed der ønskes sættet til maksimal hastighed for bil.
Returværdi	<code>void</code>
Beskrivelse	Denne funktion har til formål at skrive til variablen <code>maxVelocity_</code> .

Tabel 36: Metodebeskrivelse for `setMaxSpeed()`

Prototype	<code>void setAKS(bool AKSStatus)</code>
Parametre	<code>AKSStatus</code> Den ønskede tilstand af AKS status for bil.
Returværdi	<code>void</code>
Beskrivelse	Denne funktion har til formål at skrive til variablen <code>AKSStatus_</code> .

Tabel 37: Metodebeskrivelse for `setAKS()`

Prototype	<code>void calibrateSteering(char calibration)</code>
Parametre	<code>calibration</code> Den ønskede kalibrering af bilens styrtøj.
Returværdi	<code>void</code>
Beskrivelse	Denne funktion har til formål at skrive til variablen <code>calibration_</code> .

Tabel 38: Metodebeskrivelse for `calibratesteering()`

Prototype	<code>int getMaxSpeed()</code>
Parametre	<code>void</code>
Returværdi	<code>int</code> Returnerer variablen <code>maxVelocity_</code>
Beskrivelse	Denne funktion har til formål at give muligheden for at læse variablen <code>maxVelocity_</code> .

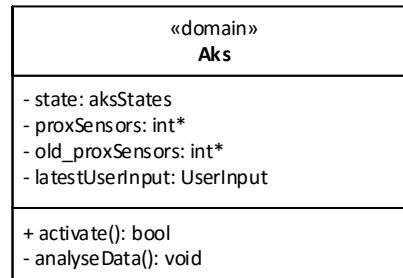
Tabel 39: Metodebeskrivelse for `getMaxSpeed()`

Prototype	<code>bool getAKS()</code>
Parametre	<code>void</code>
Returværdi	<code>bool</code> Returnerer variablen <code>AKSStatus_</code>
Beskrivelse	Denne funktion har til formål at give muligheden for at læse variablen <code>AKSStatus_</code> .

Tabel 40: Metodebeskrivelse for `getAKS()`

Prototype	<code>char getCalibration()</code>
Parametre	<code>void</code>
Returværdi	<code>char</code> Returnerer variablen <code>calibration_</code>
Beskrivelse	Denne funktion har til formål at give muligheden for at læse variablen <code>calibration_</code> .

Tabel 41: Metodebeskrivelse for `getCalibration()`

Domain-klasse: Aks

Figur 35: Klassebeskrivelse for domain-klassen Aks

Attributter

Navn	Type	Beskrivelse
MySteering	Steering	Styretøjsklassen, bruges når Aks skal påvirke bilens hastighed eller retning.
MyData	Data*	Pointer til bilens datastruktur.
MySettings	Settings*	Pointer til Settingsklassen.
MyLog	Log*	Pointer til loggen.
state	aksStates	Husker hvilket stadie bilen er i, kan skifte mellem at stå stille, køre fremad/bagud eller trille.
proxSensors	int*	Et array med nuværende værdier fra afstandssensorer
old_proxSensors	int*	Et array der holder de foregående værdier fra afstandssensorer
latestUserInput	UserInput	Gemmer de seneste input fra brugeren.

Tabel 42: Attributter for klassen Aks

Metoder

Prototype	<code>void aksActivate(void)</code>
Parametre	<code>void</code>
Returværdi	<code>bool</code> Returnerer TRUE hvis det gik godt og FALSE hvis der skete fejl undervejs.
Beskrivelse	Metoden kaldes når det automatiske anti-kollisionssystem skal aktiveres. Forhindrer samtidigt input fra brugeren kortvarigt.

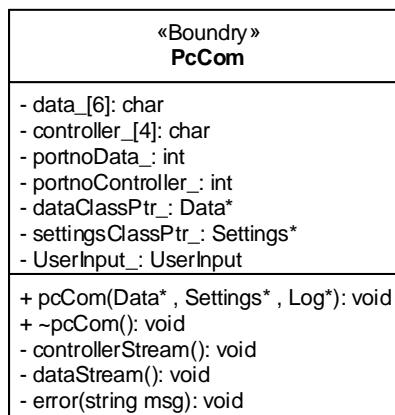
Tabel 43: Metodebeskrivelse for `aksActivate`

Prototype	<code>void analyseData(void)</code>
Parametre	<code>void</code>
Returværdi	<code>void</code>
Beskrivelse	Metoden analyserer indhentet data fra Data klassen og vurderer hvilken type af undvigelse der bedst passer. Aktiverer herefter Steering-klassen for at bilen skal undvige forhindringen.

Tabel 44: Metodebeskrivelse for `analyseData`

Boundary-klasse: PcCom

Denne klasse har til formål at styre alt kommunikation mellem PC og Bil, fra bilens side. Det er valgt at bruge TCP connections til dette og det er valgt at bruge to forskellige former for sockets således at det er muligt at sende vigtige data oftere end andre ikke nær så vigtige data. Denne klasse skal ligeledes oprette de to threads der skal styre disse sockets, således at disse TCP connections ikke stopper anden funktionalitet på bilen. På Figur ?? kan ses et klasse diagram over den ønskede klasse efterfulgt af funktionalitet.



Figur 36: Klassebeskrivelse for boundary-klassen PcCom

Attributter

Navn	Type	Beskrivelse
data_	char[6]	Array af typen char med variable der indeholder hhv. maksimal hastighed, nuværende hastighed, afstand til nærmeste forhindring, nuværende acceleration, AKS status og styretøjs calibrering.
controller_	char[4]	Array af typen char med variable der indeholder bruger input for hhv. frem, tilbage, drej og stop.
portnoData_	int	Variabel der indeholder portnummeret til den TCP socket der har med overførelse til og fra data_ array'et.
portno-Controller_	int	Variabel der indeholder portnummeret til den TCP socket der har med overførelse til og fra controller_ array'et.
dataClassPtr_	Data*	En pointer til det objekt af typen Data der ønskes læst og skrevet data til.
settings- ClassPtr_	Settings*	En pointer til det objekt af typen Settings der ønskes skrevet data til.
UserInput_	UserInput	En struct af typen UserInput der har fire chars, hhv. frem, tilbage, drej og stop.

Tabel 45: Attributter for klassen PcCom

Metoder

Prototype	<code>void PcCom(Data* dataClass, Settings* settingsClass, Log* logClass)</code>
Parametre	<p>dataClass Det objekt af typen Data der ønskes bearbjetet af PcCom.</p> <p>settingsClass Det objekt af typen Settings der ønskes bearbjetet af PcCom.</p> <p>logClass Det objekt af typen Log der ønskes skrevet til af PcCom.</p>
Returværdi	<code>void</code>
Beskrivelse	Constructor til klassen PcCom. Skal initialisere threads til funktionerne controllerStream() og dataStream().

Tabel 46: Metodebeskrivelse for constructoren af PcCom klassen

Prototype	<code>void ~PcCom()</code>
Parametre	<code>void</code>
Returværdi	<code>void</code>
Beskrivelse	Destructor til klassen PcCom.

Tabel 47: Metodebeskrivelse for destructoren af PcCom klassen

Prototype	<code>void controllerStream()</code>
Parametre	<code>void</code>
Returværdi	<code>void</code>
Beskrivelse	Denne funktion har til formål at initialisere og køre en TCP server. Denne server skal stremme hhv. frem-, tilbage-, drej- og stop-kommandoer fra PC'en til bilen. Herefter skal den sende den streamede data til Data klassen.

Tabel 48: Metodebeskrivelse for controllerStream()

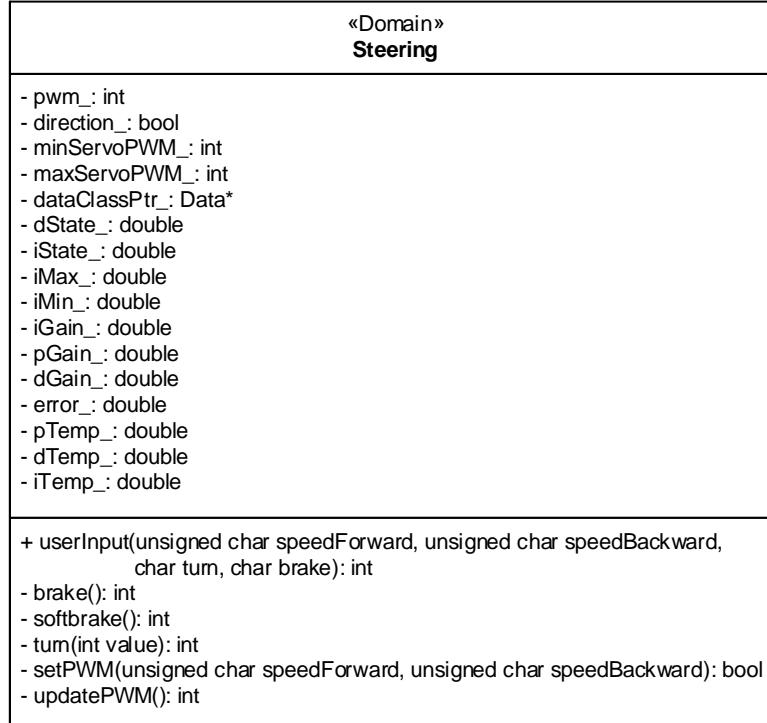
Prototype	<code>void dataStream()</code>
Parametre	<code>void</code>
Returværdi	<code>void</code>
Beskrivelse	Denne funktion har til formål at initialisere og køre en TCP server. Denne server skal stremme hhv. maksimal hastighed, nuværende hastighed, afstand til nærmeste forhindring, nuværende acceleration, AKS status og styretøjs calibrering fra PC'en til bilen. Herefter skal den sende den streamede data til Data og Settings klassen.

Tabel 49: Metodebeskrivelse for `dataStream()`

Prototype	<code>void error(string msg)</code>
Parametre	<code>msg</code> En pågældende fejl-besked i form af en string.
Returværdi	<code>void</code>
Beskrivelse	Denne funktion har til formål at skrive en fejl til log, hvis der opstår en fejl.

Tabel 50: Metodebeskrivelse for `error()`

Boundary-klasse: Steering



Figur 37: Klassebeskrivelse af boundary-klassen Sterering

Attributter

Navn	Type	Beskrivelse
pwm_	int	tekst.
direction_	bool	tekst.
minServoPWM_	int	tekst.
maxServoPWM_	int	tekst.
dataClassPtr_	Data*	tekst.
dState_	double	tekst.
iState_	double	tekst.
iMax_	double	tekst.
iMin_	double	tekst.
iGain_	double	tekst.
pGain_	double	tekst.
dGain_	double	tekst.
error_	double	tekst.
pTemp_	double	tekst.
dTemp_	double	tekst.
iTemp_	double	tekst.

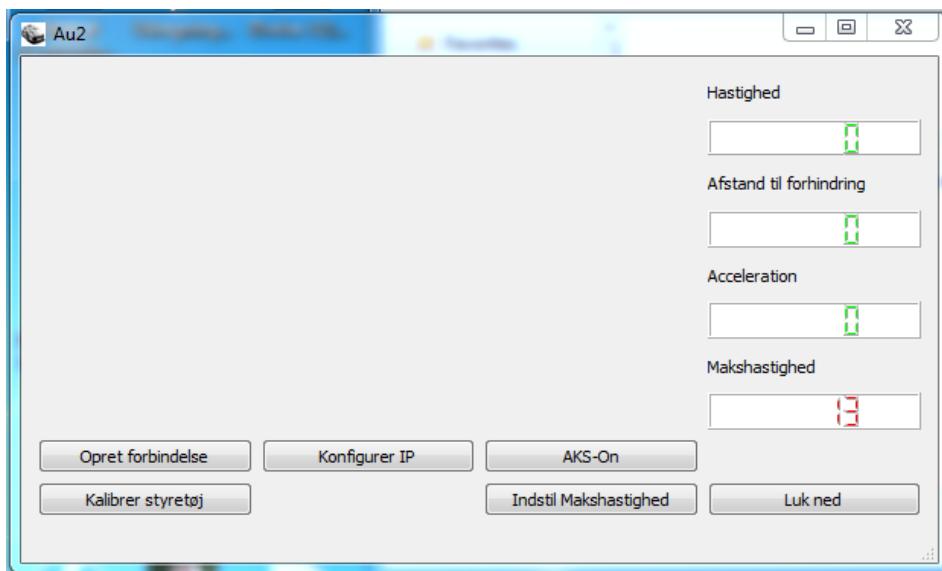
Tabel 51: Attributter for klassen Steering

Metoder

Prototype	<code>int userInput(unsigned char speedForward, unsigned char speedBackward, char turn, char brake)</code>
Parametre	<p><code>speedForward</code> Den ønskede hastighed fremad. 0..255</p> <p><code>speedBackward</code> Den ønskede hastighed bagud. 0..255</p> <p><code>turn</code> Den ønskede drejning på forhjul. -127..127</p> <p><code>brake</code> Der skal bremses. 0..1</p>
Returværdi	<code>int</code> 1 hvis alle operationer er gået okay ellers -1.
Beskrivelse	Metoden sørger for at opdaterer retning og hastighed på bilen.

Tabel 52: Metodebeskrivelse for `userInput`

5.2 PC

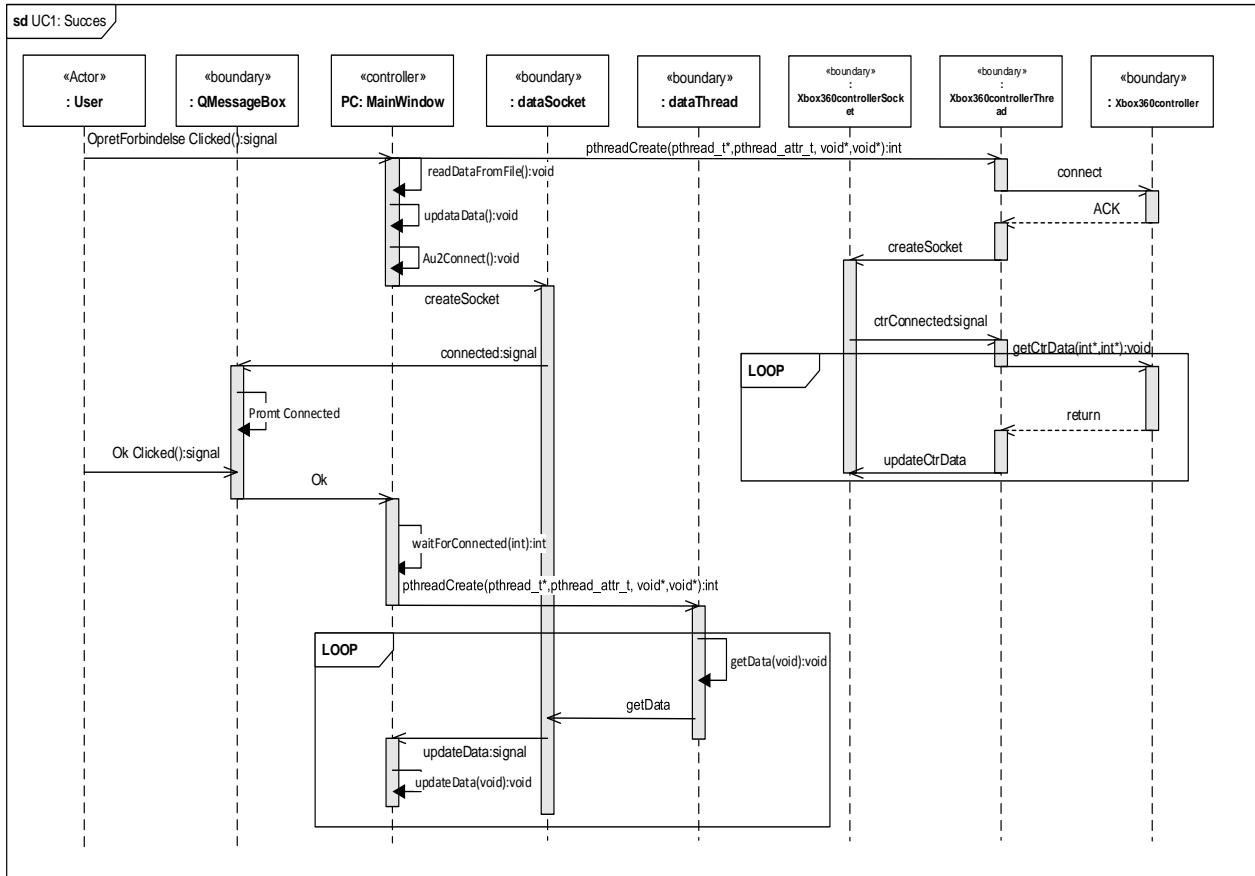


Figur 38: GUI hovedvindue

5.2.1 Sekvensdiagrammer

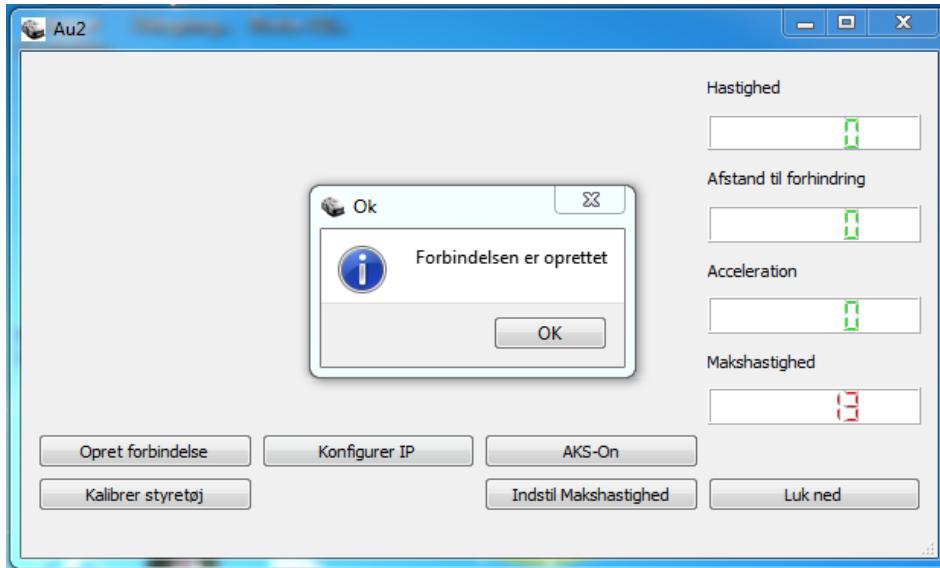
I denne sektion beskrives sekvensdiagrammer over usecasene for GUI'en. Der er valgt at lave i alt 4 scenarier af UC1 hvor 3 af dem inkluderer fejl. Dette er gjort for at gøre det mere overskueligt, da et sekvensdiagram med en masse undtagelser hurtigt bliver forvirrende. På figur ?? ses sekvensdiagrammet over UC1 scenario: succes.

UC1 Succes



Figur 39: Sekvensdiagram over UC1 scenarie:succes

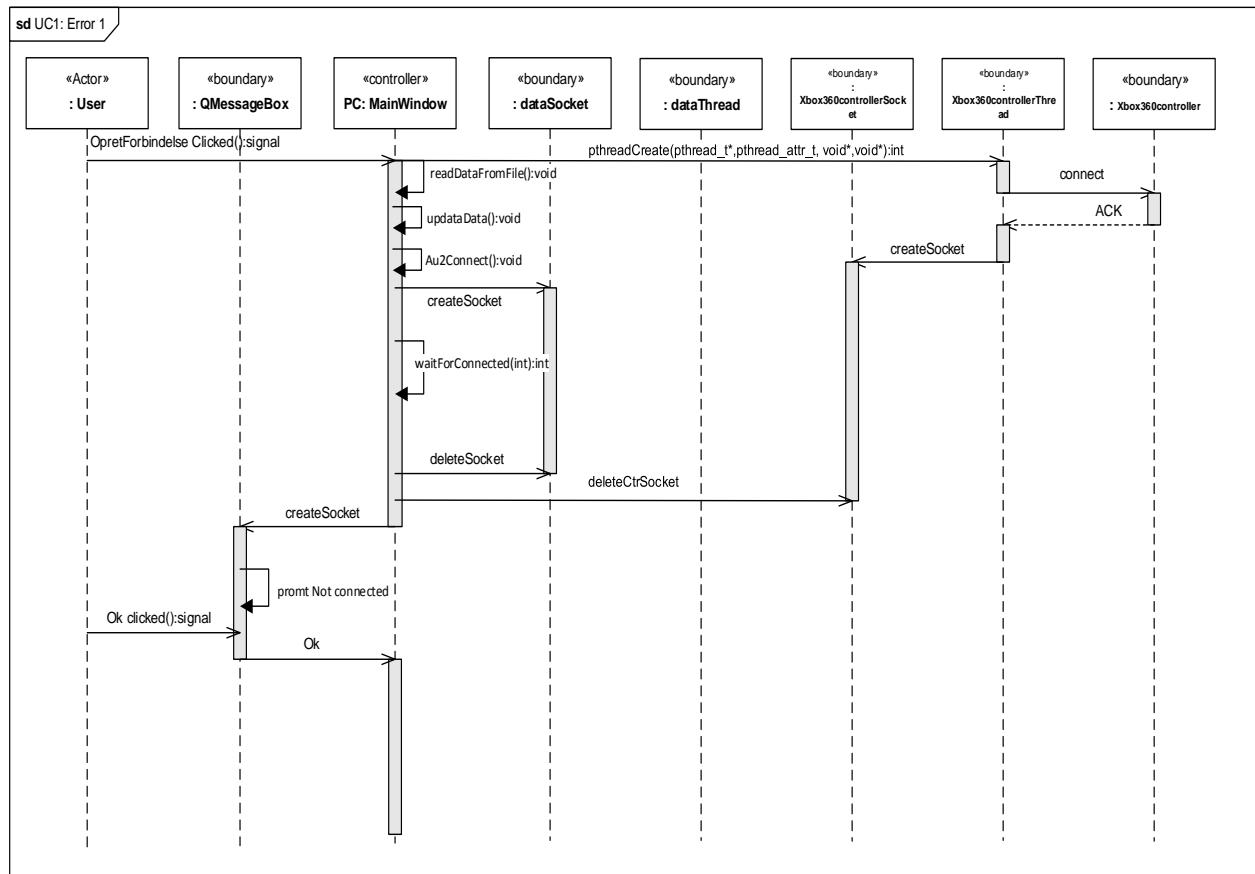
Når usecasen startes klikker brugeren på ”Opret forbindelse” på GUI’en. Herefter vil MainWindow starte Xbox360ControllerThread, som vil oprette en socketforbindelse mellem bilen og PC til at sende controllerdata. Efterfølgende vil MainWindow indlæse data, som blev gemt i en fil sidste gang programmet lukkede ned og derefter opdatere vinduet. Nu vil MainWindow oprette en socket mere til at sende og modtage data fra og til bilen. Når socketforbindelsen er oprettet vil dataSocket sende et signal til MainWindow, som derved vil promte brugeren at forbindelsen er oprettet. Se figur ???. Når brugeren lukker vinduet ved at klikke ”Ok” venter MainWindow på at der er forbindelse for at derefter oprette dataThread som efterfølgende vil stå for at opdatere vinduet med hastig, afstand, acceleration osv. Det virker måske lidt dumt at vente på at der er forbindelse efter at der er givet signal om der er forbindelse. Dette skyldes at når brugeren trykker på ”Ok” sættes en variabel til 1. Hvis forbindelsen senere mistes sættes denne variabel til 0, således dataThread ikke skriver til en socketforbindelse som ikke længere er aktiv. Ventetiden er for at sikre at signalet kommer inden for en given tid. Defor: Kommer signalet om at forbindelsen ikke er oprettet inden for en given tid, vil programmet lave en fejl som det ses i figur ???. Når dataThread har opdateret dataen, vil den give et signal til MainWindow om at der skal hentes nye data fra bilen. Det er derfor MainWindow som henter og sender data, men dataThread som opdatere selve dataen på GUI’en. Dette skyldes at QTcpsocket ikke kan køres i en separat tråd som det gøres med Xbox360ControllerThread hvis der skal gives signaler. Der sker derfor en fejl i programmet når Xbox360 controlleren er forbundet og TCP forbindelsen mistes. Der har desværre ikke været tid til at finde en løsning på dette problem.



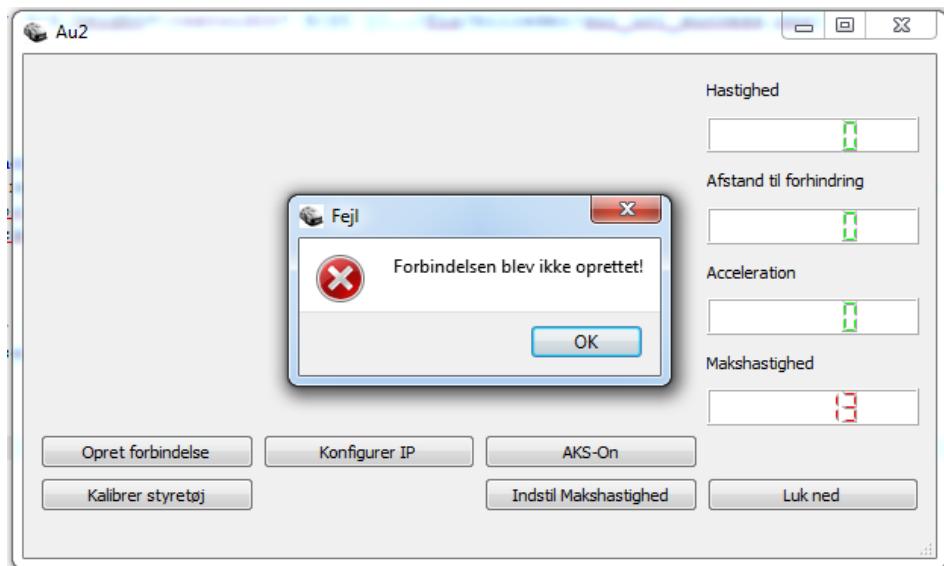
Figur 40: GUI UC1 succes

UC1 Error 1: Forbindelse kunne ikke oprettes

I denne sektion beskrives fejlen, at der ikke kan oprettes forbindelse til bilen. Dette bliver som før beskrevet konkluderet i `WaitForConnected()`, som venter 1 sekund. Er der ikke givet signal inden `delete`s `dataSocket` og `Xbox360ControllerSocket`. Brugeren promtes med at forbindelsen ikke kunne oprettes. Se sekvensdiagrammet i figur ?? og advarselsskiltet i figur ??



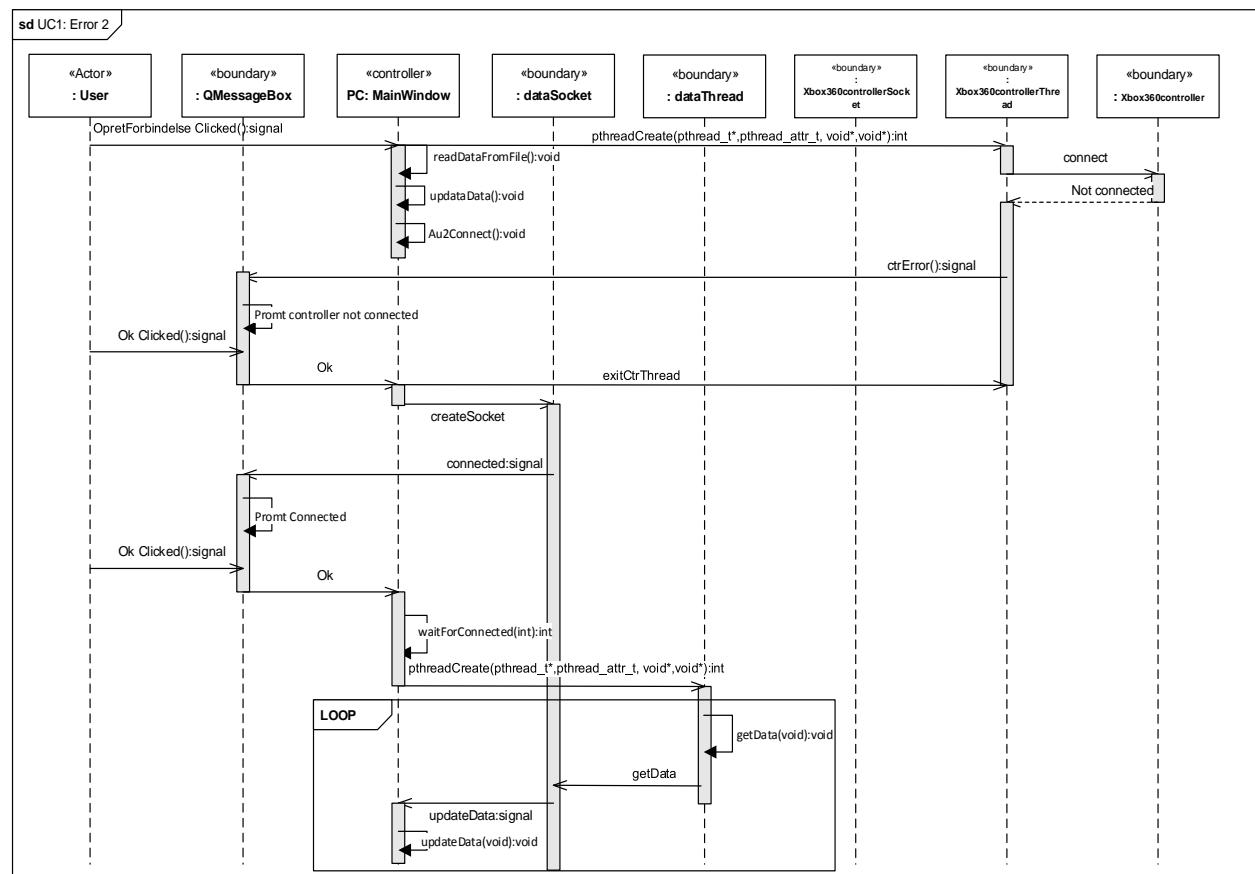
Figur 41: Sekvensdiagram over UC1 scenarie:Error 1



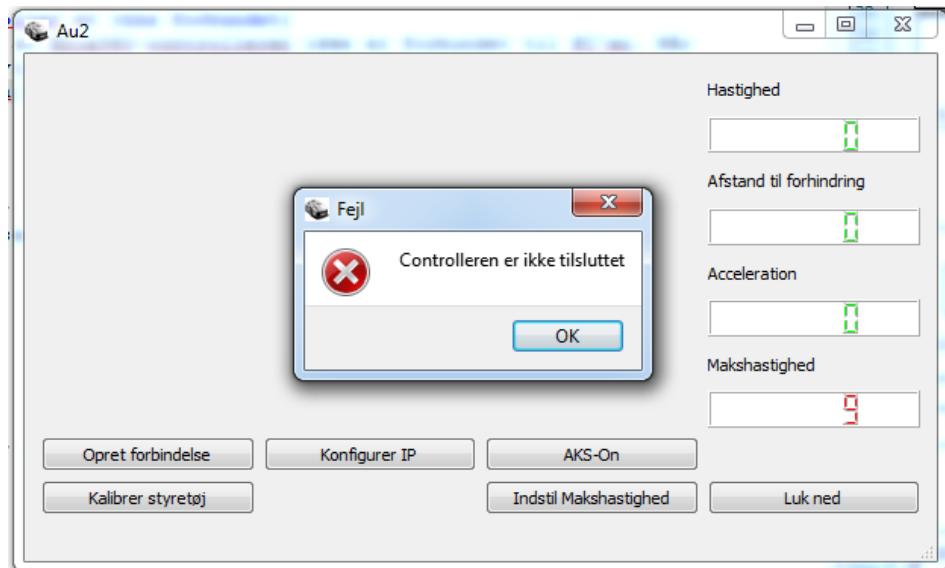
Figur 42: GUI UC1 Error 1

UC1 Error 2: Controller er ikke forbundet

I denne sektion beskrives fejlen, at Xbox360-controlleren ikke er forbundet til PC'en. Når Xbox360controllerThread oprettes vil den spørge om controlleren er tilsluttet. Hvis den ikke er promtes bruger med at controlleren ikke er forbundet. Se sekvensdiagrammet i figur ?? og advarselsskiltet i figur ??



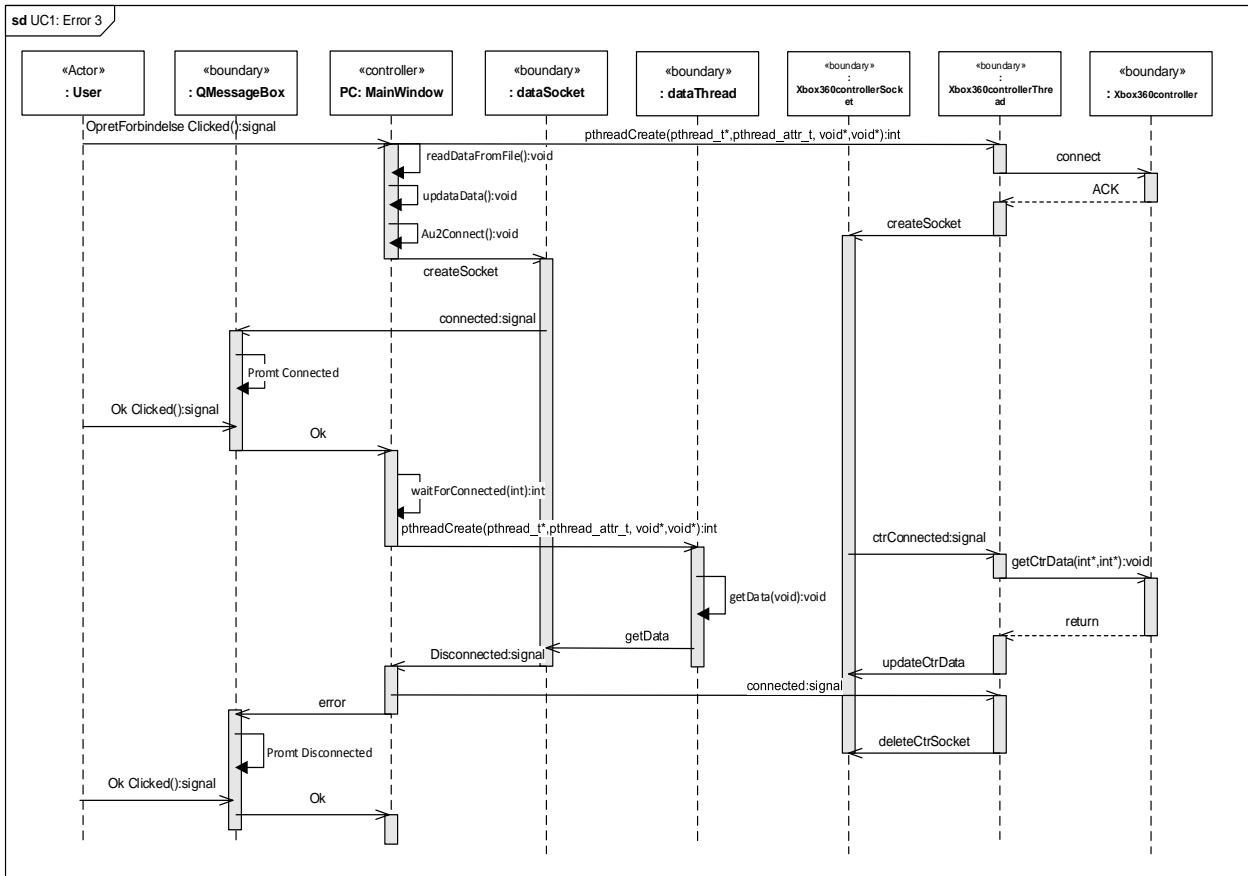
Figur 43: Sekvensdiagram over UC1 senarie:Error 2



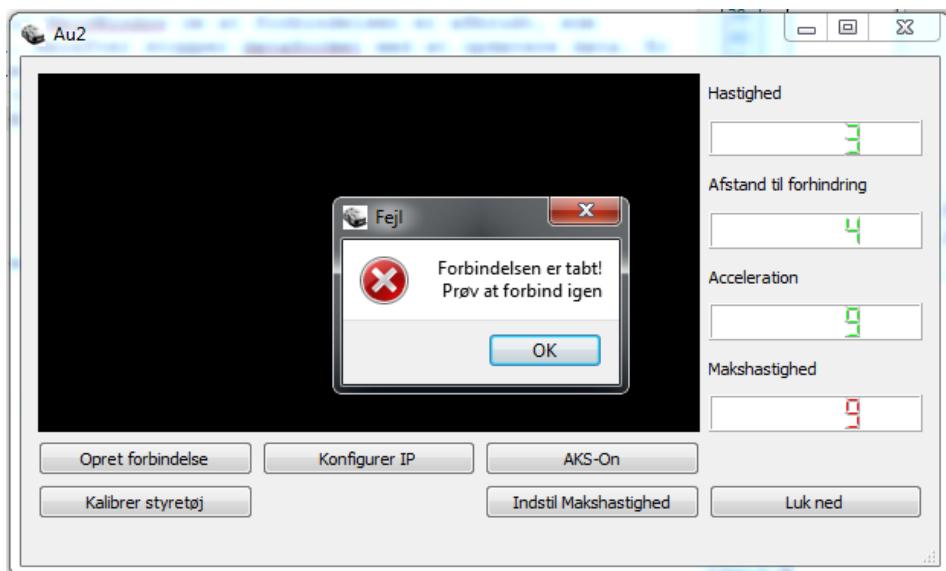
Figur 44: GUI UC1 Error 1

UC1 Error 3: Forbindelsen bliver afbrudt

I denne sektion beskrives fejlen, at forbindelsen mellem bil og PC pludselig bliver afbrudt. dataSocket giver signal til MainWindow om at forbindelsen er afbrudt, som herefter giver besked til brugeren. Herefter stopper dataSocket med at opdatere data. Er controlleren forbundet og bliver denne forbindelse også afbrudt crasher programmet desværre. Denne fejl er beskrevet tidligere. Se sekvensdiagrammet i figur ?? og advarselsskiltet i figur ??



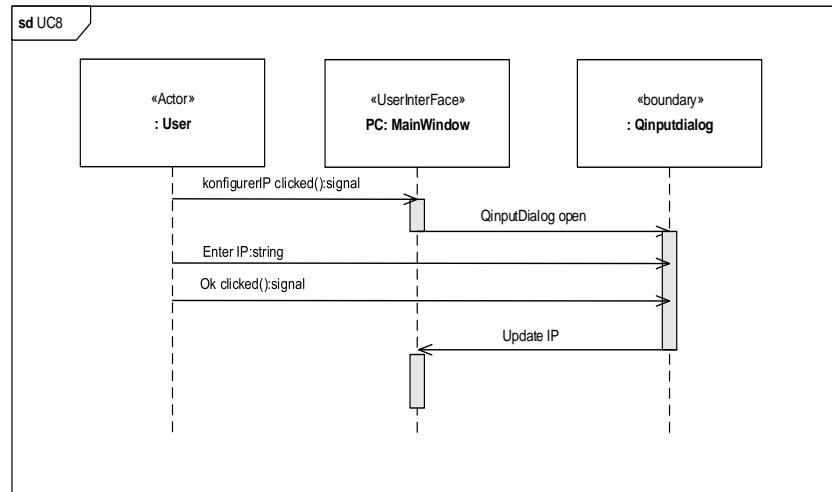
Figur 45: Sekvensdiagram over UC1 senarie:Error 3



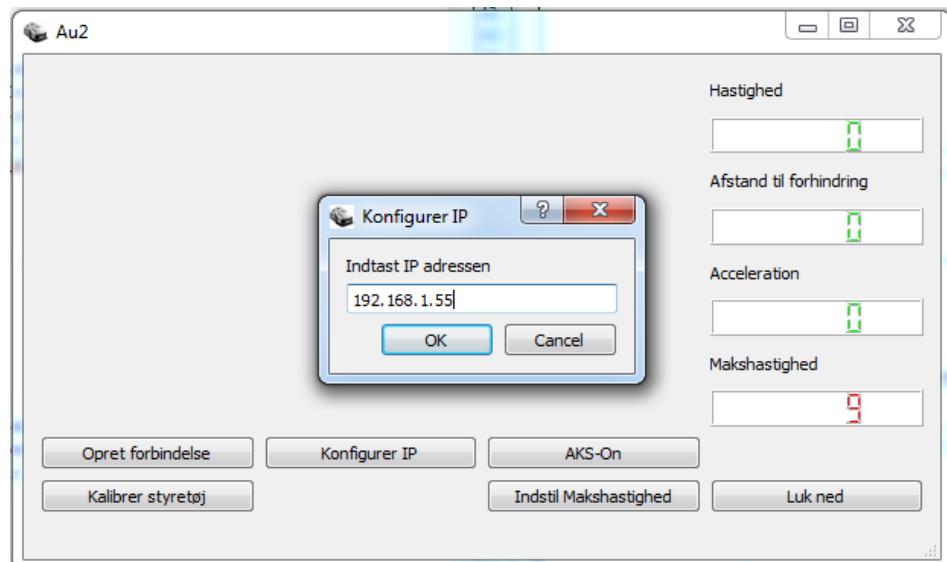
Figur 46: GUI UC1 Error 3

UC8: Konfigurer IP-adresse

For at aktivere usecase 8 trykker bruger på "Konfigurer IP". Herefter åbner MainWindow en inputdialog som bruger indtaster IP-adressen i og efterfølgende trykker "Ok". Se sekvensdiagrammet i figur ?? og indputdialogen i figur ??



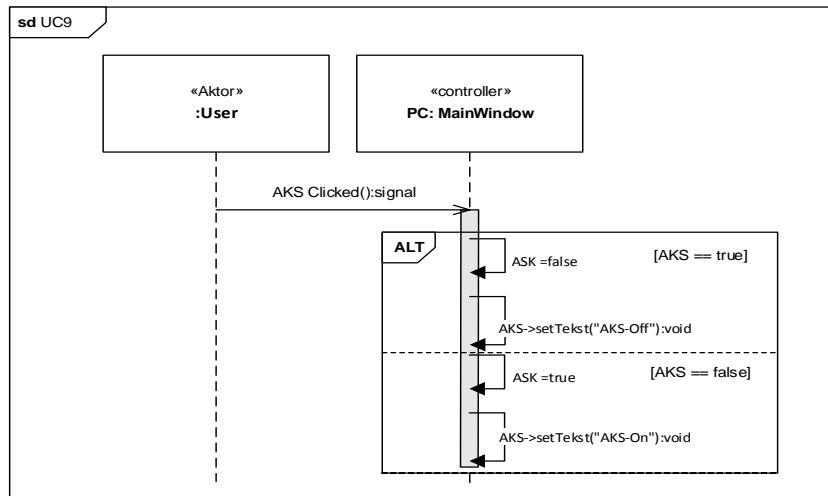
Figur 47: Sekvensdiagram over UC8



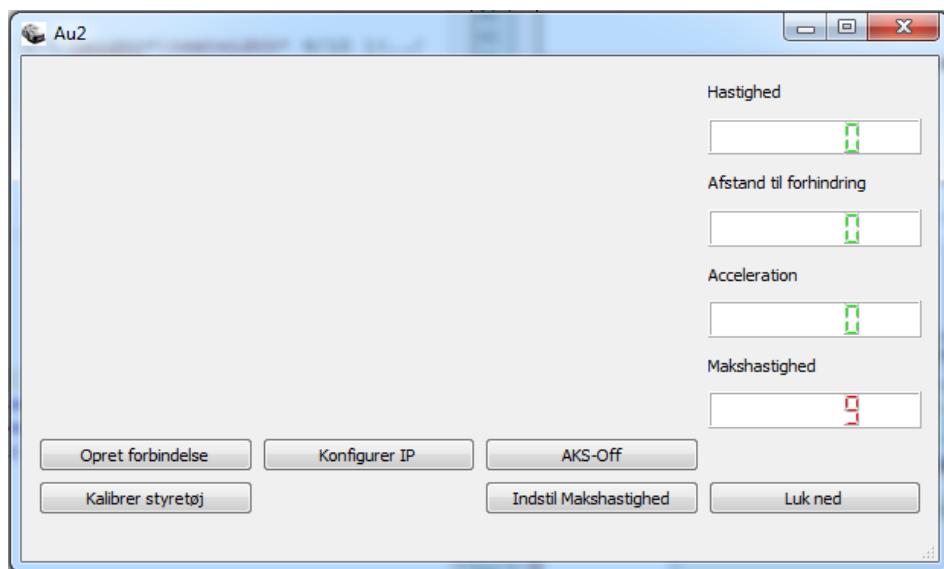
Figur 48: GUI UC8

UC9: Tænd/sluk AKS

For at aktivere usecase 9 trykker bruger på "AKS-On". Når bruger trykker på denne ændres teksten til "AKS-Off". Er teksten i forvejen "AKS-Off" ændres denne til "AKS-On". En variabel i MainWindow opdateres og gived med til bilen næste gang data bliver opdateret i UC1. Se sekvensdiagrammet i figur ?? og indputdialogen i figur ??



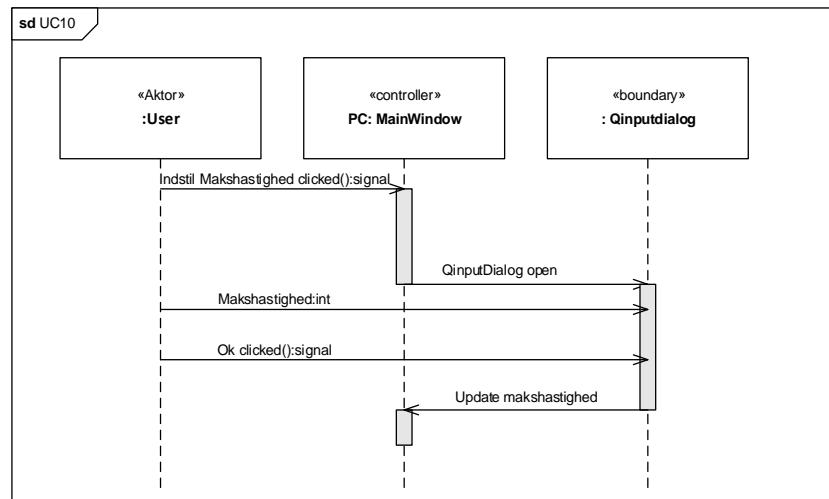
Figur 49: Sekvensdiagram over UC9



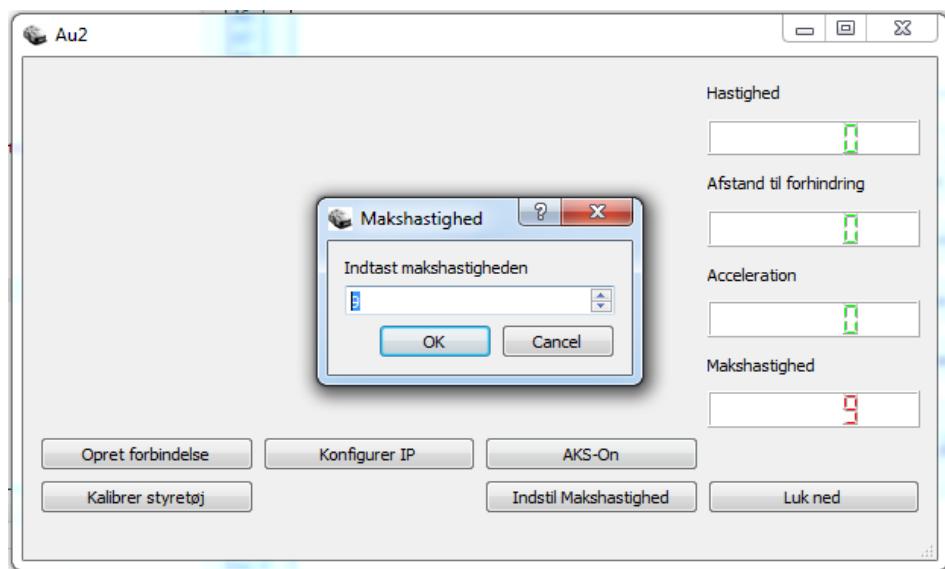
Figur 50: GUI UC9

UC10: Indstil makshastighed

For at aktivere usecase 10 trykker bruger på "Indstil makshastighed". MainWindow åbner en input-dialog hvor i bruger indtaster et tal mellem 0 og 10. Inputdialogen accepterer ikke indtastninger uden for dette interval. Se sekvensdiagrammet i figur ?? og indputdialogen i figur ??



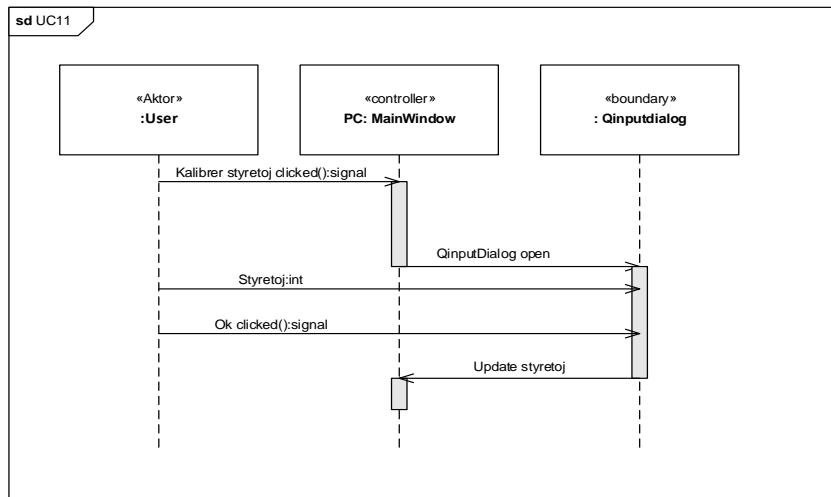
Figur 51: Sekvensdiagram over UC10



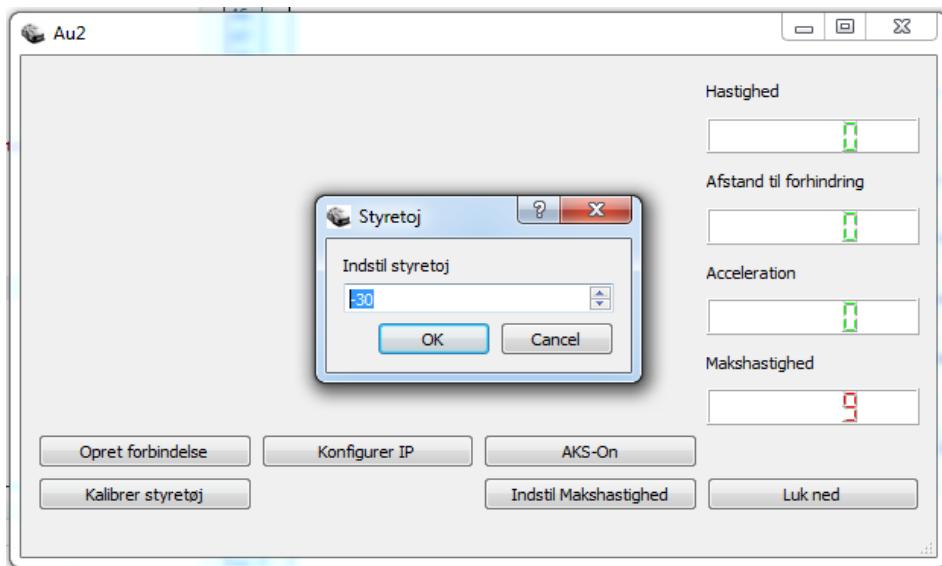
Figur 52: GUI UC10

UC11: Kalibrer styretøj

For at aktivere usecase 11 trykker bruger på "Kalibrer styretøj". MainWindow åbner en inputdialog hvor i bruger indtaster et tal mellem -50 og 50. Inputdialogen accepterer ikke indtastninger uden for dette interval. Se sekvensdiagrammet i figur ?? og indputdialogen i figur ??



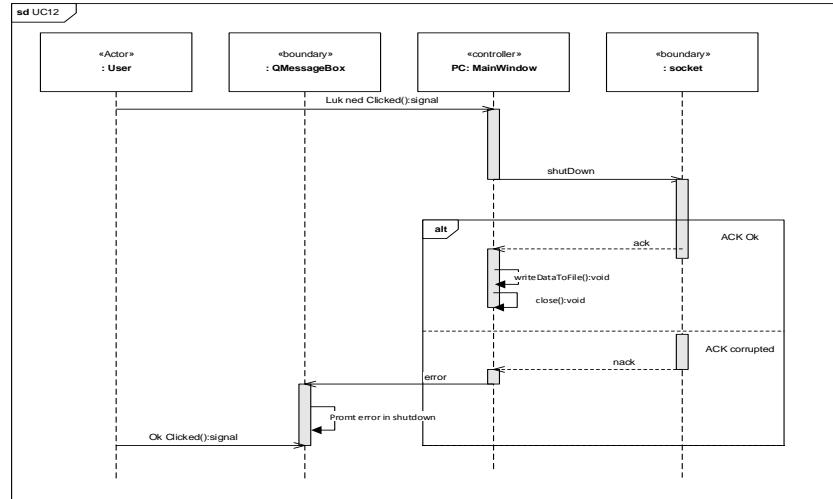
Figur 53: Sekvensdiagram over UC11



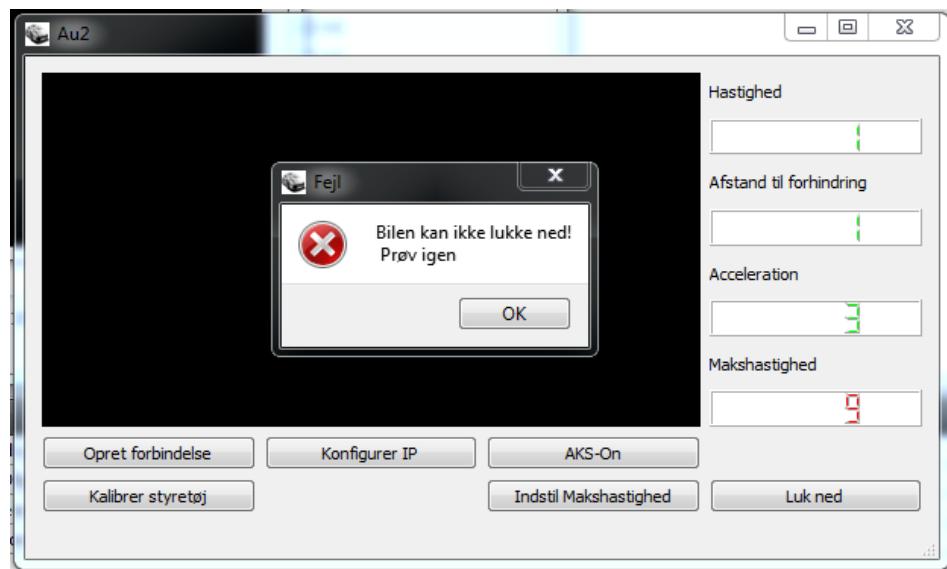
Figur 54: GUI UC11

UC12: Afbryd system

For at aktivere usecase 11 trykker bruger på "Luk ned". dataSocket beder bilen om at lukke ned og bilen svarer med et ACK. Modtages der ikke et ACK giver MainWindow bruger besked om at bilen ikke kan lukke ned. Se sekvensdiagrammet i figur ?? og advarslen i figur ??



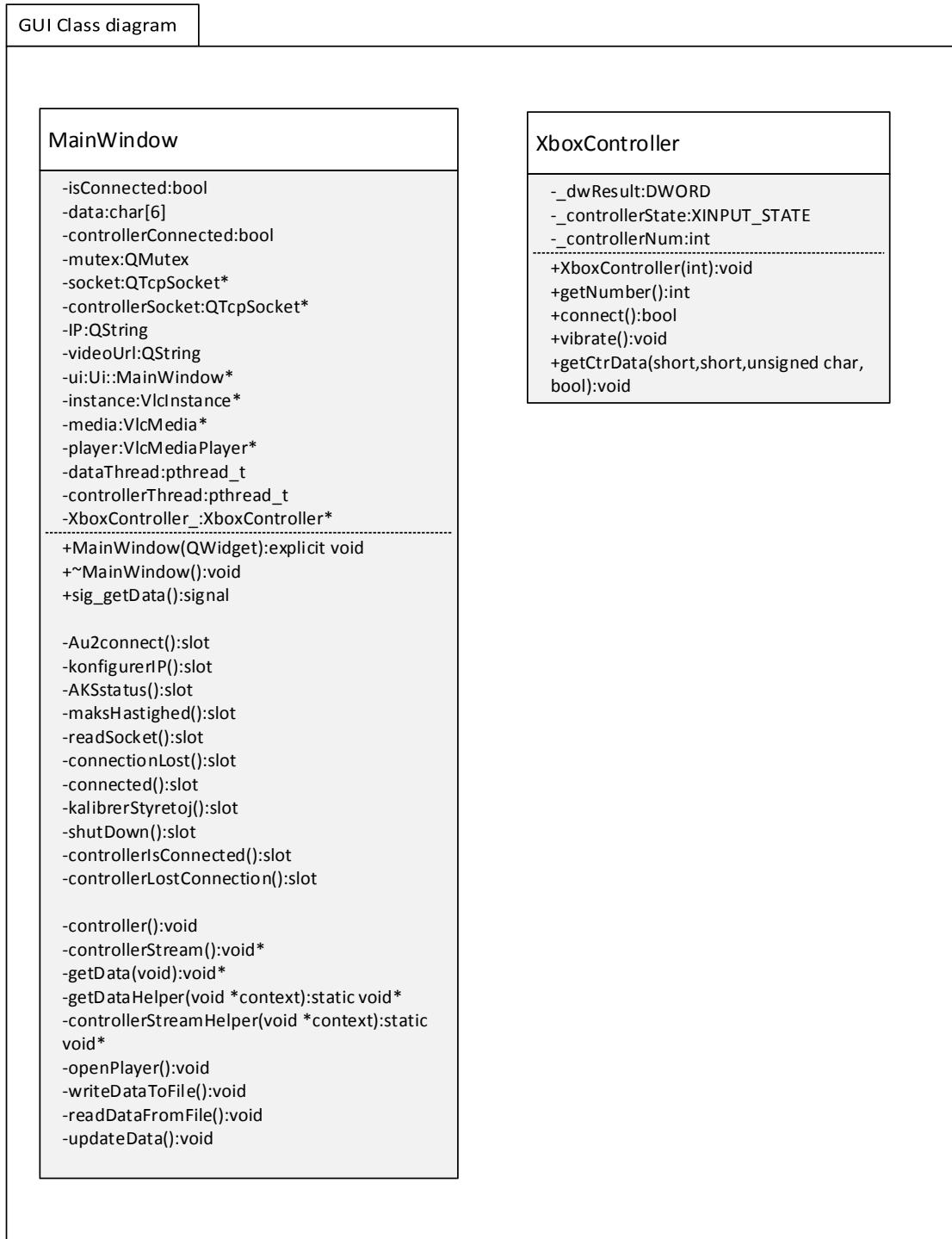
Figur 55: Sekvensdiagram over UC12



Figur 56: GUI UC12

5.2.2 Klassebeskrivelse

I denne sektion vil der blive beskrevet klassen MainWindow og XboxController. MainWindow er GUI'ens hoved klasse hvor i alt styres fra. XboxController er controllerens klasse som MainWindow bruger til at streame controllerdata fra.



Figur 57: Klasse diagram over GUI

Attributter for MainWindow

Navn	Type	Beskrivelse
isConnected	bool	Bliver brugt til at angive om socket har forbindelse til bilen eller ej
data	char [6]	Char array som indeholder: Hastighed, Makshastighed, AKS-status, Styretøj, Acceleration og Afstand
controller -Connected	bool	Bruges til at angive om Xbox360-controlleren er forbundet eller ej
mutex	QMutex*	Bruges til at låse således data kun kan tilgås af en tråd af gangen
socket	QTcpSocket*	Er selve datasocket instansen
controller -Socket	QTcpSocket*	Er selve controllerSocket instansen
IP	QString	Indeholder IP-adressen på bilen
videoURL	QString	Indeholder adressen på videotreamet fra bilen
ui::Ui	MainWindow*	Selve GUI'en
instance	VlcInstance*	Bruges til at afspille videotream
media	VlcMedia*	Bruges til at afspille videotream
player	VlcMediaPlay -er*	Bruges til at afspille videotream
dataThread	p_thread_t	Er selve data tråden
controller -Thread	p_thread_t	Er selve controller tråden
XboxControl -ler_	XboxControl -ler*	En instance af Xbox360-controlleren

Tabel 53: Attributter for klassen MainWindow

Metoder for MainWindow

Prototype	<code>explicit void MainWindow()</code>
Parametre	QWidget
Returværdi	
Beskrivelse	Er MainWindows constructor. Bruges til at indlæse data fra log filen så bruger kan se gamle indtastede værdier, samt opdatere hovedvinduet.

Tabel 54: Metodebeskrivelse for MainWindow

Prototype	<code>void ~MainWindow()</code>
Parametre	
Returværdi	
Beskrivelse	Er MainWindows destructor. Bruges til at slette oprettede instanser.

Tabel 55: Metodebeskrivelse for ~MainWindow

Prototype	<code>signal void sig_getData()</code>
Parametre	
Returværdi	
Beskrivelse	Bliver kaldt af dataThread når MainWindow skal opdatere data arrayet. Signalet fungerer ved at det eksekveres som en funktion i dataThread. Herved gives et signal til MainWindow som derved eksekverer funktionen readSocket.

Tabel 56: Metodebeskrivelse for `sig_getData`

Prototype	<code>slot void Au2connect()</code>
Parametre	
Returværdi	
Beskrivelse	Kaldes når bruger trykker på ”Opret forbindelse”. Bruges til at skabe forbindelse mellem bil og Pc samt oprette dataThread og Xbox360ControllerThread.

Tabel 57: Metodebeskrivelse for `Au2connect`

Prototype	<code>slot void konfigurerIp()</code>
Parametre	
Returværdi	
Beskrivelse	Kaldes når bruger trykker på ”Konfigurer IP”. Bruges til at opdatere variablen IP med brugerinput.

Tabel 58: Metodebeskrivelse for `konfigurerIp`

Prototype	<code>slot void AKSstatus()</code>
Parametre	
Returværdi	
Beskrivelse	Kaldes når bruger trykker på ”AKS-On” eller ”AKS-Off”. Bruges til at ændre status på AKS.

Tabel 59: Metodebeskrivelse for `AKSstatus`

Prototype	<code>slot void AKSstatus()</code>
Parametre	
Returværdi	
Beskrivelse	Kaldes når bruger trykker på ”Indstil makshastighed”. Bruges til at ændre makshastigheden på bilen.

Tabel 60: Metodebeskrivelse for `AKSstatus`

Prototype	<code>slot void readSocket()</code>
Parametre	
Returværdi	
Beskrivelse	Kaldes når dataThread vil have opdateret data fra bilen. dataThread giver signalet sig_getData(). Funktionen kører i MainWindow og opdaterer data arrayet.

Tabel 61: Metodebeskrivelse for `readSocket`

Prototype	<code>slot void connectionLost()</code>
Parametre	
Returværdi	
Beskrivelse	Kaldes når dataSocket mister forbindelsen til bilen. Variablen isConnected sættes til false.

Tabel 62: Metodebeskrivelse for `connectionLost`

Prototype	<code>slot void connected()</code>
Parametre	
Returværdi	
Beskrivelse	Kaldes når dataSocket har oprettet forbindelsen til bilen. Variablen isConnected sættes til true.

Tabel 63: Metodebeskrivelse for `connected`

Prototype	<code>slot void kalibrerStyretøj()</code>
Parametre	
Returværdi	
Beskrivelse	Kaldes når bruger trykker på "Kalibrer styretøj". Bruges til at kalibrere styretøjet på bilen. Funktionen opdaterer den respektive plads i data arrayet.

Tabel 64: Metodebeskrivelse for `kalibrerStyretøj`

Prototype	<code>slot void shutDown()</code>
Parametre	
Returværdi	
Beskrivelse	Kaldes når bruger trykker på "Luk ned" eller klikker i det røde kryds i hovedvinduet. Sørger for at sende besked til bilen om at lukke dens software sikkert ned, samt skrive data arrayet til log-filen.

Tabel 65: Metodebeskrivelse for `shutDown`

Prototype	<code>slot void controllerIsConnected()</code>
Parametre	
Returværdi	
Beskrivelse	Kaldes når der er forbindelse mellem Xbox360 controlleren og bilen. controllerConnected sættes til true.

Tabel 66: Metodebeskrivelse for `controllerIsConnected`

Prototype	<code>slot void controllerLostConnection()</code>
Parametre	
Returværdi	
Beskrivelse	Kaldes når forbindelse mellem Xbox360 controlleren og bilen afbrydes. controllerConnected sættes til false.

Tabel 67: Metodebeskrivelse for `controllerLostConnection`

Prototype	<code>void controller()</code>
Parametre	
Returværdi	
Beskrivelse	Kaldes når Au2Connect() vil oprette controllerThread. Funktionen sørger for at oprette controllerThread samt skabe controllerSocket. Funktionen retunerer hvis Xbox360 controlleren ikke er tilsluttet Pc'en.

Tabel 68: Metodebeskrivelse for `controller`

Prototype	<code>void* controllerStream()</code>
Parametre	
Returværdi	
Beskrivelse	Er selve funktionen der kontinuert bliver kørt af dataThread. Controller data bliver sendt til bilen med intervaller på 10ms.

Tabel 69: Metodebeskrivelse for `controllerStream`

Prototype	<code>static void* controllerStreamHelper()</code>
Parametre	<code>void* context</code>
Returværdi	Funktionspointer til controllerStream
Beskrivelse	Kaldes af pthread_create og retunerer funktionspointeren til controllerStream. Dette gøres fordi pthread kun kan tilgå static funktioner.

Tabel 70: Metodebeskrivelse for `controllerStreamHelper`

Prototype	<code>void* getData()</code>
Parametre	
Returværdi	
Beskrivelse	Er selve funktionen der opdaterer MainWindow med data. Giver signal til MainWindow om at hente nye data fra bilen hvert 100ms.

Tabel 71: Metodebeskrivelse for `getData`

Prototype	<code>static void* getDataHelper()</code>
Parametre	<code>void* context</code>
Returværdi	Funktionspointer <code>getData</code>
Beskrivelse	Kaldes af <code>pthread_create</code> og retunerer funktionspointeren til <code>getData</code> . Dette gøres fordi <code>pthread</code> kun kan tilgå static funktioner.

Tabel 72: Metodebeskrivelse for `getDataHelper`

Prototype	<code>void openPlayer()</code>
Parametre	
Returværdi	
Beskrivelse	Åbner instansen af VLC-player i hovedvinduet. Kaldes når <code>dataSocket</code> er forbundet.

Tabel 73: Metodebeskrivelse for `openPlayer`

Prototype	<code>void writeDataToFile()</code>
Parametre	
Returværdi	
Beskrivelse	Skriver data-arrayet til log filen når GUI'en lukkes ned.

Tabel 74: Metodebeskrivelse for `writeDataToFile`

Prototype	<code>void readDataFromFile()</code>
Parametre	
Returværdi	
Beskrivelse	Læser data-arrayet fra log filen når constructoren kaldes.

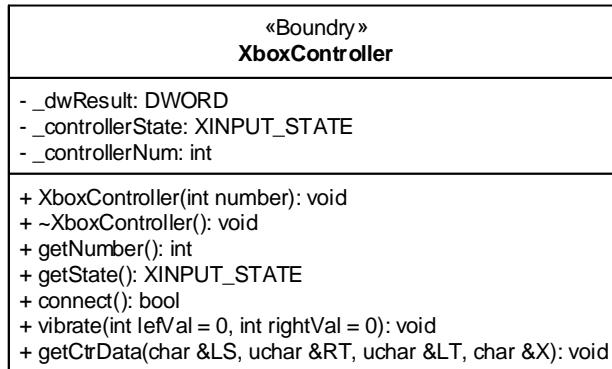
Tabel 75: Metodebeskrivelse for `readDataFromFile`

Prototype	<code>void updateData()</code>
Parametre	
Returværdi	
Beskrivelse	Opdaterer hovedvinduet med data. Kaldes i constructoren og senere af <code>dataThread</code> .

Tabel 76: Metodebeskrivelse for `updateData`

Boundary-klasse: XboxController

Denne klasse har til formål at agere API for Xbox Controlleren til PC softwaren. Til dette skal udnyttes standard biblioteket; XInput. Det skal være muligt at få alt det data der udnyttes til at bestemme bilens retning, ved et enkelt funktionskald for simplicitet. På Figur ?? kan ses et klasse diagram over den ønskede klasse efterfulgt af funktionalitet.



Figur 58: Klassebeskrivelse for boundary-klassen XboxController

Attributter

Navn	Type	Beskrivelse
_dwResult	DWORD	Variabel af typen DWORD, skal indeholde data om controllerens tilstand.
_controller-State	XINPUT_STATE	Struct af typen XINPUT_STATE. Indholder hhv. DWORD, der indeholder en værdi forskellig fra 0 hvis der er sket en ændring i controllerens tilstand, og XINPUT_GAMEPAD, der indeholder alle værdier som fortæller om controllerens nuværende tilstand.
_controller-Num	int	Variabel af typen int der indeholder controller nummer.

Tabel 77: Attributter for klassen XboxController

Metoder

Prototype	void XboxController(int number)
Parametre	number Det ønskede controller ID (1-4).
Returværdi	void
Beskrivelse	Constructor til klassen XboxController.

Tabel 78: Metodebeskrivelse for constructoren af XboxController klassen

Prototype	<code>void ~XboxController()</code>
Parametre	<code>void</code>
Returværdi	<code>void</code>
Beskrivelse	Destructor til klassen <code>XboxController</code> .

Tabel 79: Metodebeskrivelse for destructoren af `XboxController` klassen

Prototype	<code>int getNumber()</code>
Parametre	<code>void</code>
Returværdi	<code>int</code> Controllerens ID
Beskrivelse	Denne funktion har til formål at returnere objektets ID.

Tabel 80: Metodebeskrivelse for `getNumber()`

Prototype	<code>XINPUT_STATE getState()</code>
Parametre	<code>void</code>
Returværdi	<code>XINPUT_STATE</code> Struct af typen <code>XINPUT_STATE</code> . Indeholder hhv. <code>DWORD</code> og <code>XINPUT_GAMEPAD</code> .
Beskrivelse	Denne funktion har til formål at opdatere <code>_controllerState</code> som indeholder controllerens nuværende tilstand og info om hvorvidt den har skiftet tilstand.

Tabel 81: Metodebeskrivelse for `getState()`

Prototype	<code>bool connect()</code>
Parametre	<code>void</code>
Returværdi	<code>bool</code> Controllerens connection tilstand
Beskrivelse	Denne funktion returnerer hvorvidt der er forbindelse til controlleren.

Tabel 82: Metodebeskrivelse for `connect()`

Prototype	<code>void vibrate(int leftVal = 0, int rightVal = 0)</code>
Parametre	<p>leftVal En værdi der fortæller controllerklassen hvor hurtigt vibrater motoren i venstre side af controlleren skal køre (0 - 65535).</p> <p>rightVal En værdi der fortæller controllerklassen hvor hurtigt vibrater motoren i højre side af controlleren skal køre (0 - 65535).</p>
Returværdi	<code>void</code>
Beskrivelse	Denne funktion igangsætter højre og venstre vibrater med en styrke alt efter input parameterne leftVal og rightVal. Inputparameterne kan gå mellem 0 og 65535, hvor 0 er ingen vibrering og 65535 er maksimal vibrering. Hvis der ikke gives parametre med i funktionen, vil den automatisk sætte parameterne til 0 og dermed slukke for vibraterne.

Tabel 83: Metodebeskrivelse for `vibrate()`

Prototype	<code>void getCtrlData(char &LS, uchar &RT, uchar &LT, char X)</code>
Parametre	<p>LS En reference til den char der ønskes at funktionen indsætter data om controllerens "Left Stick" tilstand i.</p> <p>RT En reference til den unsigned char der ønskes at funktionen indsætter data om controllerens "Right Trigger" tilstand i.</p> <p>LT En reference til den unsigned char der ønskes at funktionen indsætter data om controllerens "Left Trigger" tilstand i.</p> <p>X En reference til den char der ønskes at funktionen indsætter data om controllerens "X-button" tilstand i.</p>
Returværdi	<code>void</code>
Beskrivelse	Denne funktion har til formål at returnere nye controller tilstande, for controllerens Left Stick, Right Trigger, Left Trigger og X-button, på de refererede parametres plads.

Tabel 84: Metodebeskrivelse for `getCtrlData()`

6 Hardware Implementering

6.1 Strømforsyning

7 Software Implementering

7.0.1 Bil

Camera

På Pi'en skal der installeres en virtuel driver til kameraet kaldet uv4l, før det vil virke med programmet motion. Cameraet der skal bruges er et **Raspberry Pi Camera Rev 1.3**. Kameraet forbindes med et fladkabel til connectoren på Pi'en som der er markeret med teksten "Camera". Når kameraet er forbundet, startes Pi'en op og der logges ind via en ssh-forbindelse. Efterfølgende indtastes:

```
1 $ sudo raspi-config
```

I denne menu skal cameraet aktiveres. Når kameraet er aktiveret trykkes exit og i terminalen skrives:

```
1 $ curl http://www.linux-project.org/listing/uv4l_repo/1rkey.asc | sudo apt-key add -
```

Herved åbned en fil som angiver hvilke url **apt-get** henter filer fra. Nederst i filen tilføjes:

```
1 deb http://www.linux-project.org/listing/uv4l_repo/raspbian/ wheezy main
```

Når filen er opdateret skrives:

```
1 $ sudo apt-get update
2 $ sudo apt-get install uv4l uv4l-raspicam motion
```

Nu er driveren uv4l samt motion installeret. Før vi kan begynde at streame skal der ændres i motion's config file. Filen åbnes ved at skrive:

```
1 $ sudo nano /etc/motion/motion.conf
```

I filen skal følgende variablerstå til:

```
1 daemon off
2 videodevice /dev/video0
3 width 640
4 height 480
5 framerate 25
6 quality 100
7 control_port 8080
8 webcam_port 8081
9 webcam_quality 100
10 webcam_localhost off
11 webcam_limit 0
12 threshold 9999999
```

Streamet kan nu startes ved følgende kommando:

```
1 $ uv4l --driver raspicam --auto-video_nr
2 $ LD_PRELOAD=/usr/lib/uv4l/uv4lext/armv6l/libuv4l.so motion -c ./motion.conf
```

Følgende findes også i scriptet startVidStream.sh som ligger i bilagende. Scriptet skal kopieres over i mappen **/home/pi/Documents/** For at streamet skal starte ved opstart, skal scriptet tilføjes til crontab. Dette gøres ved at åbne crontab og skrive stien til scriptet. Crontab åbnes:

```
1 $ sudo crontab -e
```

I slutningen af filen tilføjes:

```
1 $ /home/pi/Documents/startVidStream.sh
```

Settings

Settings er en klasse, som holder styr på systemets indstillinger. Klassens primære opgave er, at holde styr på brugerens indstillinger fra PC softwaren, hhv. calibrering af styrtøj, maksimal hastighed og hvorvidt AKS er slæt til eller fra. Der er flere threads der kan finde på, at kalde funktioner i Settings klassen på samme tid, skal klassen sørge for at der kun er en thread der læser eller skriver til dens data af gangen. Herunder vil koden blive forklaret i detaljer.

Settings Constructor er klassens constructor. Ud over at initialisere alle settings data til 0, sker der en linkning til et Log object, via association, da det ønskes at constructoren skriver til text logen når klassen initialiseres, for at kunne fejlfinde. Koden er vist i vist i Listing ??.

```

1 Settings :: Settings (Log* logClassPtr)
2 {
3     this->logClassPtr_ = logClassPtr;
4     this->maxVelocity_ = 0;
5     this->AKSStatus_ = 0;
6     this->calibration_ = 0;
7     this->logClassPtr_->writeEvent (__PRETTY_FUNCTION__, "Initialized Settings")
8 }
```

Listing 7.1: Settings Constructor

setMaxSpeed er en funktion til at skrive til klassens attribut, der indeholder den indstillede maksimale hastighed. Koden er vist i Listing ???. Ud over at skrive tallet, der er givet med som parameter, til klassens attribut, låses klassens attribut så det ikke er muligt for andre funktionskald at skrive eller læse attributen til at den er færdig med at skrive til den.

```

1 void Settings :: setMaxSpeed (int maxVelocity)
2 {
3     std :: lock_guard<std :: mutex> lock (this->maxVelocityMut);
4     this->maxVelocity_ = maxVelocity;
5     return;
6 }
```

Listing 7.2: Funktionen setMaxSpeed

setAKS er en funktion til at skrive til klassens attribut, der indeholder AKS status. Koden er vist i Listing ???. Denne funktion er implementeret på samme måde som setMaxSpeed, beskrevet tidligere, og vil derfor ikke blive gennemgået i detaljer her.

```

1 void Settings :: setAKS (bool AKSStatus)
2 {
3     std :: lock_guard<std :: mutex> lock (this->AKSStatusMut);
4     this->AKSStatus_ = AKSStatus;
5     return;
6 }
```

Listing 7.3: Funktionen setAKS

calibrateSteering er en funktion til at skrive til klassens attribut, der indeholder kalibrering af styrtøj. Koden er vist i Listing ???. Denne funktion er implementeret på samme måde som setMaxSpeed og setAKS, beskrevet tidligere, og vil derfor ikke blive gennemgået i detaljer her.

```

1 void Settings :: calibrateSteering( char calibration )
2 {
3     std :: lock_guard<std :: mutex> lock( this->calibrationMut );
4     this->calibration_ = calibration ;
5     return ;
6 }
```

Listing 7.4: Funktionen calibrateSteering

getMaxSpeed er en funktion til at læse og returnere klassens attribut, der indeholder den indstilte maksimale hastighed. Koden er vist i Listing ???. Ud over at læse attributen låses klassens attribut, så det ikke er muligt for andre funktionskald at skrive eller læse attributen, indtil at funktionen er færdig med at læse fra og returnere den.

```

1 int Settings :: getMaxSpeed()
2 {
3     std :: lock_guard<std :: mutex> lock( this->maxVelocityMut );
4     return this->maxVelocity_ ;
5 }
```

Listing 7.5: Funktionen getMaxSpeed

getAKS er en funktion til at læse og returnere klassens attribut, der indeholder AKS status. Koden er vist i Listing ???. Denne funktion er implementeret på samme måde som getMaxSpeed, beskrevet tidligere, og vil derfor ikke blive gennemgået i detaljer her.

```

1 bool Settings :: getAKS()
2 {
3     std :: lock_guard<std :: mutex> lock( this->AKSStatusMut ) ;
4     return this->AKSStatus_ ;
5 }
```

Listing 7.6: Funktionen getAKS

getCalibration er en funktion til at læse og returnere klassens attribut, der indeholder kalibrering af styrtøj. Koden er vist i Listing ???. Denne funktion er implementeret på samme måde som getMaxSpeed og getAKS, beskrevet tidligere, og vil derfor ikke blive gennemgået i detaljer her.

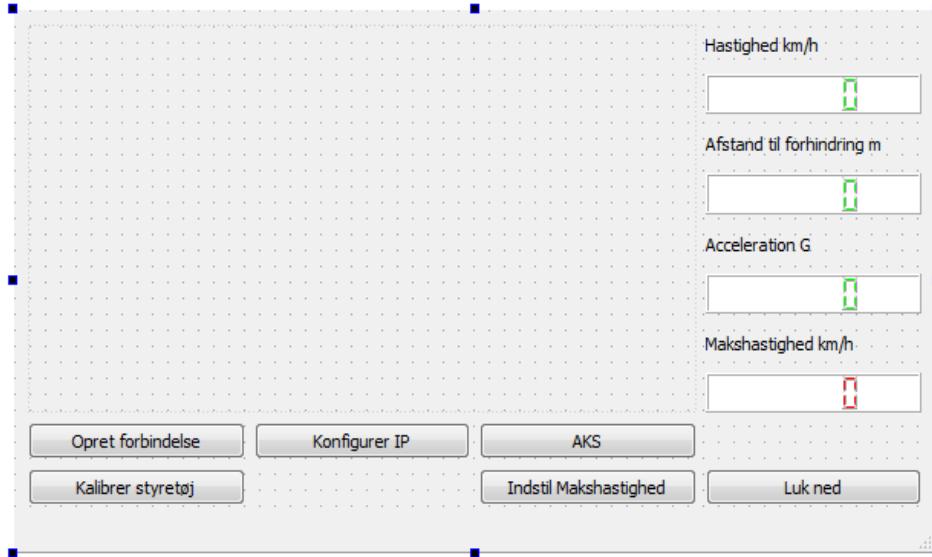
```

1 char Settings :: getCalibration()
2 {
3     std :: lock_guard<std :: mutex> lock( this->calibrationMut ) ;
4     return this->calibration_ ;
5 }
```

Listing 7.7: Funktionen getCalibration

7.1 Pc

7.1.1 GUI



Figur 59: GUI i implementerings processen

Udviklingsmiljøet som hele GUI'en er skrevet i er Qt version 5.5 [?]. En af fordelene ved QT er at man kan lave den grafiske del af GUI'en hurtigt og uden at skulle vide noget om hvordan koden bagved fungerer. Princippet er drag and drop og fungerer ved at man trækker de forskellige knapper og bokse ind i vinduet. Qt opretter selv en klasse kaldet MainWindow. I Main.cpp oprettes en instans af MainWindow som gør at hele programmet køres i MainWindow's constructor. Når programmet kører og der trykkes på en knap gives der et signal. Signalet forbindes til en slot i constructoren ved hjælp af funktionen `connect`.

```

1 connect(ui->OpretForbindelse, SIGNAL(clicked()), this, SLOT(Au2connect()));
2 connect(ui->KonfigurerIP, SIGNAL(clicked()), this, SLOT(konfigurerIP()));
3 connect(ui->AKS, SIGNAL(clicked()), this, SLOT(AKSstatus()));
4 connect(ui->IndstilMaksHastighed, SIGNAL(clicked()), this, SLOT(maksHastighed()));
5 connect(ui->KalibrerStyretoj, SIGNAL(clicked()), this, SLOT(kalibrerStyretoj()));
6 connect(ui->LukNed, SIGNAL(clicked()), this, SLOT(shutDown()));
7 connect(this, SIGNAL(sig_getData()), this, SLOT(readSocket()));

```

Listing 7.8: Forbindelse af signals og slots

I listing ?? ses fx. at når der klikkes på OpretForbindelse kaldes funktionen Au2connect som er funktionen der opretter forbindelsen til bilen.

7.1.2 VLC

Udviklingsmiljøet som hele GUI'en er skrevet i er Qt version 5.5. For at inkludere VLC, skal Qt være installeret [?]. For at kunne modtage video stream i GUI'en skal vi bruge en forbygget version af VLC til windows32 indeholdende .dll filer osv, samt bilitoteker til Qt. Dette gøres ved at udpakke Filen **vlc-2.0.7-win32.7z** som hentes fra en ftp server [?]. til destinationen **c:/Qt/**. Pakken indeholder rumtime-filerne som senere skal kopieres over i debugfolderen. Include filerne til Qt downloades **“Official VLC-Qt Windows SDK and Source Packages”** [?] og udpakkes i **c:/Qt/**. I denne

pakke ligger der et demoprojekt som der er hentet inspiration fra til projektet. Laves der et nyt projekt skal der inkluderes de rigtige filer til Qt. Dette gøres ved at åbne .pro filen i Qt og tilføje:

```
1 # Edit below for custom library location
2 LIBS      += -LC:\Qt\libvlc-qt\lib -lvlc-qt -lvlc-qt-widgets
3 INCLUDEPATH += C:\Qt\libvlc-qt
```

Når projektet er bygget kopieres filerne **libvlc-qt-widgets.dll** og **libvlc-qt.dll** fra **C:/Qt/libvlc-qt/bin** til build folderen. Efterfølgende kopieres filerne **axvlc.dll**, **libvlc.dll**, **libvlccore.dll** og **nvlc.dll** fra **C:/Qt/vlc-2.0.7** samt folderen **plugins** også til buildfolderen. Programmet skulle nu genre kunne køre. Hvis ikke kan der findes mere hjælp her [?]. Desværre er der nogle fejl i linket, som der gerne skulle være rettet i denne beskrivelse.

8 Accepttest

Version

Dato	Version	Initialer	Ændring
29. september	1	Alle	Første udkast.
26. oktober	2	PKP, KT og JEP	Mindre rettelser efter review
	3		
	4		

8.1 Funktionelle Krav

Fremgangsmåden for test af funktionelle krav er generelt taget udgangspunkt i Use Cases. I tabel ?? er vist en matrise der sammenholder Use Cases med funktionelle krav, der sikrer at alle krav bliver testet ved test af Use Cases. Der henvises til kravnumre i afsnit ?? på side ??.

Krav	1	2	3	4	5	6	7	8	9	10	11	12
UC1						X				X		
UC2										X		
UC3							X					
UC4	X	X	X				X	X	X			
UC5	X		X		X							
UC6		X			X							
UC7			X								X	
UC8						X						
UC9											X	
UC10			X	X								
UC11		X										
UC12												

Tabel 85: Use Case-krav matrise

Use Case 1: Aktiver system

Use case under test		UC1: Aktiver system		
Scenarie		Hovedscenarie		
Forudsætning		Netværksforbindelse er opsat og fungerende		
Step	Handling	Forventet Resultat	Resultat	Godkendt / Kommentar
1.1	Bruger sætter bilens "ON/OFF"-switch til "ON".	Visuel test: Lampe på bilens strømforsyning lyser.		
1.2	Bruger starter software på PC.	Visuel test: Hovedvinduet vises på skærmen.		
1.3	Bruger trykker på "Opret forbindelse".	Visuel test: Hovedvindue viser "Forbindelse oprettet".		
1.4	Bruger observerer hovedvinduet.	Visuel test: Videostream vises i hovedvinduet.		
1.5	Bruger observerer hovedvinduet.	Visuel test: Bilens aktuelle hastighed vises i hovedvinduet.		

1.6	Bruger observerer hovedvinduet.	Visuel test: Bilens aktuelle tyngdeacceleration vises i hovedvinduet.		
1.7	Bruger observerer hovedvinduet.	Visuel test: Data fra bilens afstandssensorer vises i hovedvinduet.		

Tabel 86: Accepttest for UC1: Aktiver system

Use Case 2: Stream Video

Use case under test		UC2: Stream Video		
Scenarie		Hovedscenarie		
Forudsætning		UC1 frem til punkt 5 er fuldført		
Step	Handling	Forventet Resultat	Resultat	Godkendt / Kommentar
2.1	Bruger åbner AU2 softwaren på PC'en og trykker "Opret forbindelse".	Visuel test: Der vises et live-feed fra bilens kamera.		
2.2	Bruger har Wireshark åbent på samme computer. Wireshark er opsat til at overvåge det pågældende netværk.	Visuel test: I Wireshark observeres der for overføring af pakker fra bilens IP-adresse til computerens IP-adresse.		

Tabel 87: Accepttest for UC2: Stream Video

Use Case 3: Overvåg sensorer

Use case under test		UC3: Overvåg sensor		
Scenarie		Hovedscenarie		
Forudsætning		UC1 frem til punkt 6 er fuldført		
Step	Handling	Forventet Resultat	Resultat	Godkendt / Kommentar
3.1	Bruger åbner programmet PUTTY.EXE og indtaster ssh -l pi IP _ ADRESS -p 22.	Visuel test: Terminalen spørger om et password”.		
3.2	Bruger indtaster 1234.	Visuel test: Terminalen viser pi@raspberry \$.		
3.3	Bruger indtaster nano /etc/var/log/au2log	Visuel test: Log filen viser Accelerometer initialisering.. Done. Tachometer initialisering.. Done. Distancesensors initialisering.. Done.		
3.4	Bruger kører en tur med bilen og observerer hovedmenu i softwaren på PC.	Visuel test: Bruger observerer at data for bilens hastighed, afstand til forhindring og acceleration fremgår af brugerfladen.		

Tabel 88: Accepttest for UC3: Overvåg sensor

Use Case 4: Undvig forhindring

Use case under test		UC4: Undvig forhindring		
Scenarie		Hovedscenarie		
Forudsætning		UC1 er gennemført, UC3 er gennemført.		
Step	Handling	Forventet Resultat	Resultat	Godkendt / Kommentar
4.1	Bruger styrer bilen fremad mod en forhindring på min. (30cm × 30cm) vinkelret på bilens kørebane vha. Xbox-360 controlleren, således at bilen er umiddelbart til venstre for objektet.	Visuel test: Bruger observerer at bilen ændrer kurs til højre på trods af brugerinput.		
4.2	Bruger tester om det er muligt at styre bilen igen med Xbox-360 controlleren.	Visuel test: Bruger observerer at bilen reagerer på brugerinput.		
4.3	Bruger styrer bilen fremad mod en forhindring på (30cm × 30cm) vinkelret på bilens kørebane vha. Xbox-360 controlleren, således at bilen er umiddelbart til højre for objektet.	Visuel test: Bruger observerer at bilen ændrer kurs til venstre på trods af brugerinput.		
4.4	Bruger styrer bilen fremad mod en forhindring på (30cm × 30cm) vinkelret på bilens kørebane vha. Xbox-360 controlleren, således at bilen har retning lige mod objektet.	Visuel test: Bruger observerer at bilen standser på trods af brugerinput.		

4.5	Bruger bakker mod en forhindring på (30cm × 30cm) vinkelret på bilens kørebane vha. Xbox-360 controlleren, således at bilen er umiddelbart til venstre for objektet.	Visuel test: Bruger observerer at bilen ændrer kurs til venstre på trods af brugerinput.		
4.6	Bruger bakker bilen mod en forhindring på (30cm × 30cm) vinkelret på bilens kørebane vha. Xbox-360 controlleren, således at bilen er umiddelbart til højre for objektet.	Visuel test: Bruger observerer at bilen ændrer kurs til højre på trods af brugerinput.		
4.7	Bruger bakker bilen mod en forhindring på (30cm × 30cm) vinkelret på bilens kørebane vha. Xbox-360 controlleren, således at bilen har retning lige mod objektet.	Visuel test: Bruger observerer at bilen standser på trods af brugerinput.		

Tabel 89: Accepttest for UC4: Undvig forhindring

Use Case 5: Kør bil frem/tilbage

Use case under test		UC5: Kør bil frem/tilbage		
Scenarie		Hovedscenarie		
Forudsætning		UC1: Aktiver system er fuldført og systemet er operationelt.		
Step	Handling	Forventet Resultat	Resultat	Godkendt / Kommentar
5.1	Bruger holder "RT" på Xbox-360 controlleren halvt nede.	Visuel test: Bruger observerer at bilen accelererer fremad til halv makshastighed og holder denne.		
5.2	Bruger holder "RT" på Xbox-360 controlleren helt nede.	Visuel test: Bruger observerer at bilen accelererer fremad til makshastighed og holder denne.		
5.3	Bruger holder "RL" på Xbox-360 controlleren halvt nede.	Visuel test: Bruger observerer at bilen accelererer bagud til halvdelen af makshastighed og holder denne.		
5.4	Bruger holder "RL" på Xbox-360 controlleren helt nede.	Visuel test: Bruger observerer at bilen accelererer bagud til makshastighed og holder denne.		

Tabel 90: Accepttest for UC5: Kør bil frem/tilbage

Use Case 6: Drej bil til højre/venstre

Use case under test		UC6: Drej bil til højre/venstre		
Scenarie		Hovedscenarie		
Forudsætning		UC1: Aktiver system er fuldført og systemet er operationelt		
Step	Handling	Forventet Resultat	Resultat	Godkendt / Kommentar
6.1	Bruger ændrer position af den venstre styrepind på Xbox360-controlleren til venstre yderposition.	Visuel test: Bilens forhjul drejer 30° til venstre i forhold til center.		
6.2	Bruger ændrer position af den venstre styrepind på Xbox360-controlleren halvvejs til venstre yderposition.	Visuel test: Bilens forhjul drejer 15° til venstre fra center.		
6.3	Bruger ændrer position af den venstre styrepind på Xbox360-controlleren til højre yderposition.	Visuel test: Bilens forhjul drejer 30° til højre fra center.		
6.4	Bruger ændrer position af den venstre styrepind på Xbox360-controlleren halvvejs til højre yderposition.	Visuel test: Bilens forhjul drejer 15° til højre fra center.		
6.5	Bruger ændrer position af den venstre styrepind på Xbox360-controlleren til center mellem højre og venstre yderposition.	Visuel test: Bilens forhjul går tilbage til center.		

Tabel 91: Accepttest for UC6: Drej bil til højre/venstre

Use Case 7: Brems bil

Use case under test		UC7: Brems Bil		
Scenarie		Hovedscenarie		
Forudsætning		UC1: Aktiver system er fuldført og systemet er operationelt.		
Step	Handling	Forventet Resultat	Resultat	Godkendt / Kommentar
7.1	Bruger trykker på "X" knappen på Xbox-360 controlleren.	Visuel test: Bruger observerer at bilens hastighed sænkes hvis i fart, ellers tændes bilens bremselflys blot.		

Tabel 92: Accepttest for UC7: Brems Bil

Use Case 8: Konfigurer IP-adresse

Use case under test		UC8: Konfigurer IP-adresse		
Scenarie		Hovedscenarie		
Forudsætning		UC1: Aktiver system er udført, bilen og PC er på samme netværk, at systemet viser "Hovedvindue" samt at systemet er operationelt.		
Step	Handling	Forventet Resultat	Resultat	Godkendt / Kommentar
8.1	Bruger trykker på "Konfigurer IP".	Visuel test: Konfigurations menuen for IP-adressen vises, og der er mulighed for at indtaste en IP-adresse.		
8.2	Bruger indtaster bilens IP-adresse. Og trykker "Gem".	Visuel test: Systemet viser "Hovedvindue".		
8.3	Bruger trykker på "Opret forbindelse".	Visuel test: Hovedmenuen viser et videobillede samt opdater variablerne Hastighed, Afstand, Acceleration og Makshastighed.		

Tabel 93: Accepttest for UC8: Konfigurer IP-adresse

Use Case 9: Tænd/Sluk AKS

Use case under test		UC9: Tænd/sluk AKS		
Scenarie		Hovedscenarie		
Forudsætning		UC1: Aktiver system er udført, bilen og PC er på samme netværk, at systemet viser "Hoved vindue" samt at systemet er operationelt.		
Step	Handling	Forventet Resultat	Resultat	Godkendt / Kommentar
9.1	Bruger trykker på "AKS-On".	Visuel test: Knappen ændres til "AKS-Off".		
9.2	Bruger trykker på "AKS-Off".	Visuel test: Knappen ændres til "AKS-On".		

Tabel 94: Accepttest for UC9: Tænd/sluk AKS

Use Case 10: Indstil maksimalhastighed

Use case under test		UC10: Indstil makshastighed		
Scenarie		Hovedscenarie		
Forudsætning		UC1: Aktiver system er udført, bilen og PC er på samme netværk, at systemet viser "Hovedvindue" samt at systemet er operationelt.		
Step	Handling	Forventet Resultat	Resultat	Godkendt / Kommentar
10.1	Bruger trykker på "Indstil makshastighed".	Visuel test: Hovedvindue viser menu med mulighed for at indtaste makshastighed fra 1-10 km/t.		
10.2	Menuen viser bilens nuværende makshastighed.	Den nuværende makshastighed vises.		
10.3	Bruger indtaster bilens ønskede makshastighed.	Menuen viser den ønskede makshastighed.		
10.4	Bruger trykker på "Ok".	Systemet viser den nye makshastighed.		
10.5	Bruger holder "RT" inde på Xbox 360 controlleren.	Bilen accelererer til den angivne maksimalhastighed.		

Tabel 95: Accepttest for UC10: Indstil makshastighed

Use Case 11: Kalibrer styrtøj

Use case under test		UC11: Kalibrer styretøj		
Scenarie		Hovedscenarie		
Forudsætning		UC1: Aktiver system er udført, bilen og PC er på samme netværk, at systemet viser "Hovedmenu", at systemet er operationelt samt bilen holder stille		
Step	Handling	Forventet Resultat	Resultat	Godkendt / Kommentar
11.1	Bruger vælger "Kalibrer styretøj"	Visuel test: Menu med mulighed for kalibrering fremkommer.		
11.2	Bruger indtaster en værdi mellem 50 og -50 for kalibrering.	Den ønskede værdi vises.		
11.3	Bruger trykker på "Ok".	Forhjulene drejer en absolut værdi mod enten højre eller venstre: positiv værdi giver udslag til højre, og negativ værdi giver udslag venstre.		
11.5	Systemet returnerer til "Hovedvindue"	Visuel test: "Hovedvindue" fremkommer		

Tabel 96: Accepttest for UC11: Kalibrer styretøj

Use Case 12: Afbryd system

Use case under test		UC12: Afbryd system		
Scenarie		Hovedscenarie		
Forudsætning		UC1: Aktiver system er fuldført, bilen holder stille og systemet er operationelt		
Step	Handling	Forventet Resultat	Resultat	Godkendt / Kommentar
12.1	Bruger trykker på ”Luk ned”.	Visuel test: Hovedvinduet forsvinder fra skærmen.		
12.2	Bruger venter på at lampen på strømforsyningen slukker	Visuel test: Lampe på strømforsyning slukker.		
12.3	Bruger skubber kontakten ”ON/OFF” på undersiden af bilen til position ”OFF”	Visuel test: Knappen står i position OFF.		

Tabel 97: Accepttest for UC12: Afbryd system

8.2 Ikke-funktionelle krav

Krav	Test	Forventet Resultat	Resultat	Godkendt/ kommentar
1.	<p>Der udmåles en strækning, på en vandret overflade, på 10 meter. Bilen startes således at maksimumshastighed er nået når den passerer startpunktet for den udmålte strækning. Der tages tid på bilen fra startpunkt til slutpunkt af strækningen.</p> <p>Herefter omregnes disse data til en hastighed.</p>	Der måles en maksimumshastighed på 10 km/t \pm 10%.		
2.	<p>Der udmåles en strækning, på en vandret overflade, på 1 meter. Bilen startes således at maksimumshastighed er nået når den passerer startpunktet for den opmålte strækning. Bilen sættes til at bremse indtil at bilen er i stilstand. Der observeres om bilen er kommet ud over slutpunkt for den udmålte strækning.</p>	Bil er i stilstand inden for den udmålte stræknings start- og slutpunkt.		

3.	Bilen sættes til at accelerere ligeud på en vandret overflade. Der tages tid fra start af bilens acceleration. Ved 6 sekunders mærket tages der to billeder af bilen med et kamera med en maks. lukketid på $1/100s$. Disse billeder bruges til at finde hastigheden af bilen, ved at aflæse længden bilen har flyttet sig og dividere det med tiden der er gået mellem de to billeder.	De beregnede data viser at bilen har nået maksimumhastighed på $10 \text{ km/t} \pm 10\%$.		
4.	Bruger trykker på "B"-knappen på Xbox-360 controlleren. Der måles en tid fra tryk på knap til test-LED på bilen lyser.	Den målte tid overstiger ikke $50ms$.		
5.	Der placeres en genstand der opfylder givne krav for forhindring ($30cm \times 30cm$) i afstanden $6m$ fra sensoren og der måles om sensoren detekterer pågældene genstand.	Der observeres at sensoren detekterer genstanden.		
6.	Bruger slukker for program på PC.	Der observeres at bilens bremse-LED lyser indenfor $50ms$.		
7.	Datablad for kamera aflæses.	Kameraet er detekteret til at have en opdateringshastighed på minimum 15 billeder i sekundet.		

8.	Der tages et screenshot af hovedvinduet. Selve videofeedet beskæres i mspaint.exe og måles.	Den målte oplosning er 640×480 pixels.		
9.	Wireshark anvendes til at aflæse antal kommandoer sendt per sekund.	Den aflæste mængde kommandoer er minimum 60.		
10.	Bruger kigger på HID.	HID består af Xbox-360 controller og tastatur.		

Tabel 98: Ikke funktionelle krav

Litteraturliste

- [1] Mipi Alliance: *Kamera interface standard.*
<http://mipi.org/specifications/camera-interface>. 2015.
- [2] RASPBERRY PI FOUNDATION: *Anbefalet PSU størrelse.*
<https://www.raspberrypi.org/help/faqs/#powerReqs>. 2015.
- [3] MaxBotix: *I2CXL-MaxSonar-EZ0 datablad.*
http://www.maxbotix.com/Ultrasonic_Sensors/MB1202.htm. 2015.
- [4] Corona: *CS238MG Metal Gear Servo datablad.*
<http://kurser.oha.dk/eit/eit-lab/embeddedStock/Datasheet/CS238MG.pdf>. 2015.
- [5] InvenSense: *MPU-6050 accelerometer datablad.*
<http://www.invensense.com/products/motion-tracking/6-axis/mpu-6050/>. 2015.
- [6] SIEMENS: *TLE4905L Hall Switch til tachometer.*
<http://www.alldatasheet.com/datasheet-pdf/pdf/45868/SIEMENS/TLE4905L.html>. 2015.
- [7] EPCOS: *ETD 29/16/10 spolekerne og form datablad.*
http://en.tdk.eu/inf/80/db/fer_13/etd_29_16_10.pdf. 2015.
- [8] Kristian Thomsen: *Beregninger for bilens strømforsyning.*
Bilag 01. 2015.
- [9] Tore Skogberg: *Analogteknik T-005 v1.2.*
Bilag 02. 2015.
- [10] Texas Instruments: *LM26003 datablad.*
<http://www.ti.com/product/lm26003>. 2015.
- [11] VLC mediaplayer plugins:
<ftp://ftp.videolan.org/pub/videolan/vlc/2.0.7/win32/>
- [12] Qt creator:
<http://www.qt.io/ide/>
- [13] vlc-qt-library:
<https://vlc-qt.tano.si/>
- [14] vlc using qt:
<http://derekmolloy.ie/custom-video-streaming-player-using-libvlc-and-qt/>
- [15] Siemens: *TLE4905 datablad.*
Bilag 03. 1997-09-1.
- [16] C++ GUI programming in Qt:
<http://www.bogotobogo.com/cplusplus/files/c-gui-programming-with-qt-4-2ndedition.pdf>
- [17] motion on raspberry:
<http://captain-slow.dk/2013/11/01/install-motion-on-a-raspberry-pi/>

[18] motion on raspberry:

<https://discourse.osmc.tv/t/trying-to-install-motion-missing-libs/5709/9>

[19] uv4l Camera driver:

<http://www.linux-project.org/modules/sections/index.php?op=viewarticle&artid=14>