BEERTRESS

Systemarkitektur

Semesterprojekt 3, Gruppe 8

Peter Wann
201907121

August Hjerrild Andersen
201907251

Simon Phi Dang
201705957

Henry Pham
201606071

Alexander Flarup Wodstrup
201810602

Lucas Friis-Hansen
201811527

Jim Sørensen
201602614

Shynthavi Prithviraj
201807198

17. december 2020

Indhold

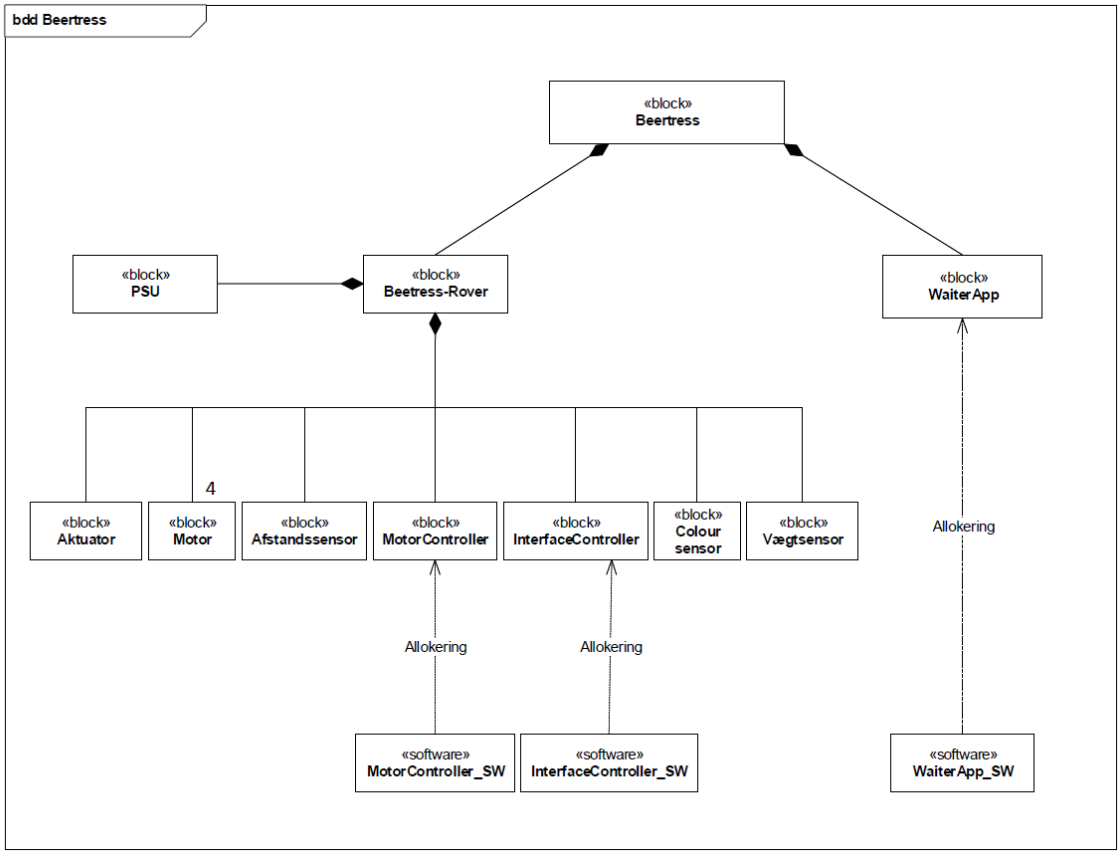
1 System arkitektur

1.1 Overblik over systemet

I dette afsnit vil figurer som BDD og sekvensdiagrammer give et overordnet overblik over systemet og kommunikationen imellem systemet. Disse vil blive understøttet af beskrivende tekster for at få en større forståelse for systemet.

1.1.1 Blok definitions diagram

Figur 1 viser det overordnede BDD for Beertress. Beertress består af 3 enheder som har forskellige software pakker.

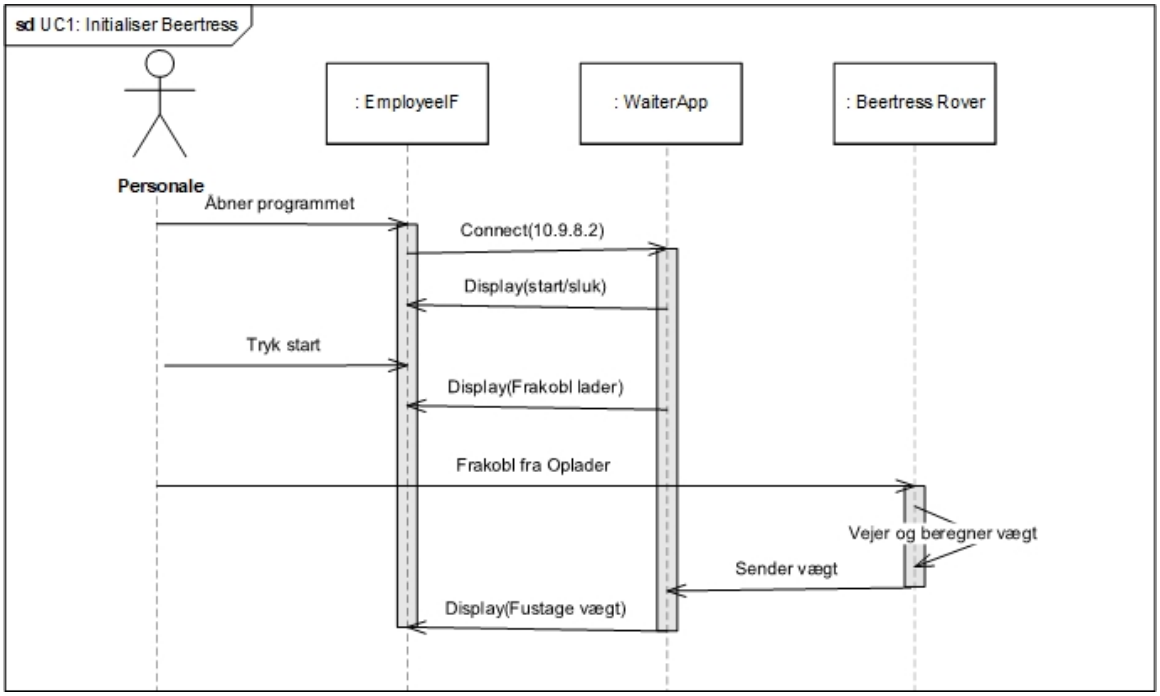


Figur 1: Overordnet BDD for Beertress

1.1.2 Systemsekvensdiagrammer

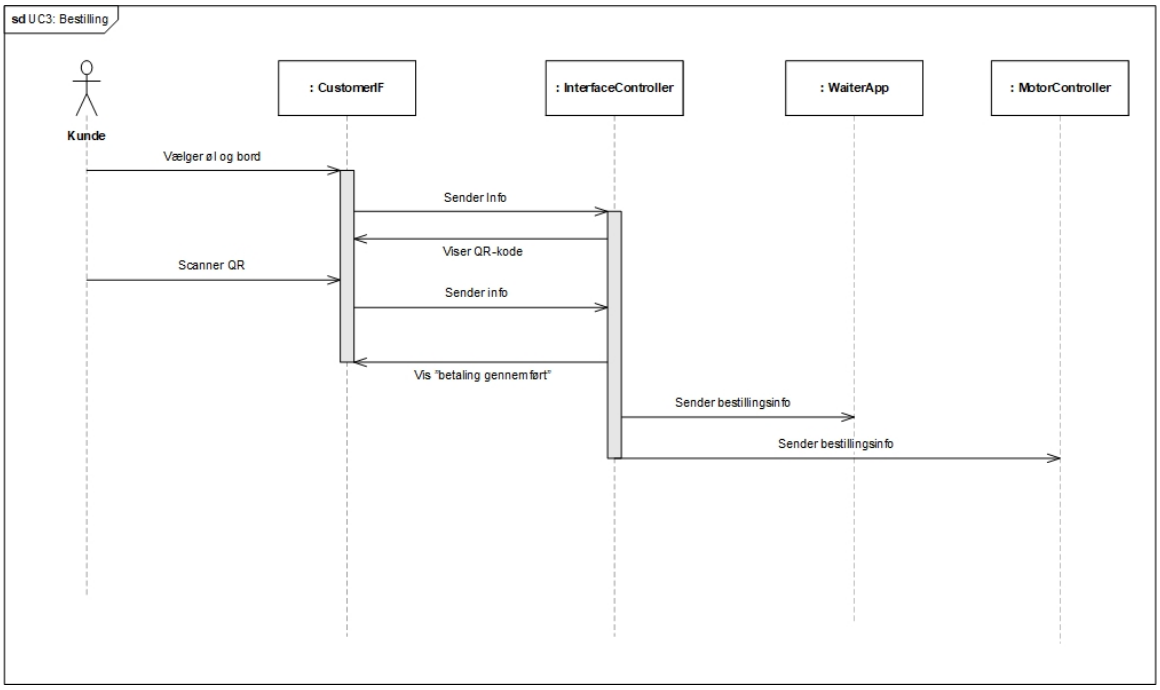
I de næste afsnit vil systemsekvensdiagrammerne for vores Use Case 1, 3 og 4 gennemgås, for at få et idé om kommunikationen i systemet. Der vil ikke være systemsekvensdiagrammer for Use Case 2 og 5, da de ikke fortæller det store om kommunikationen i systemet, som ikke sker i UC1, 3 og 4.

Sekvensdiagram for UC1 Figur 2 viser sekvensdiagrammet for UC1, som handler om opstart af Beertress. Dette sker fra WaiterApp som er en PC applikation.



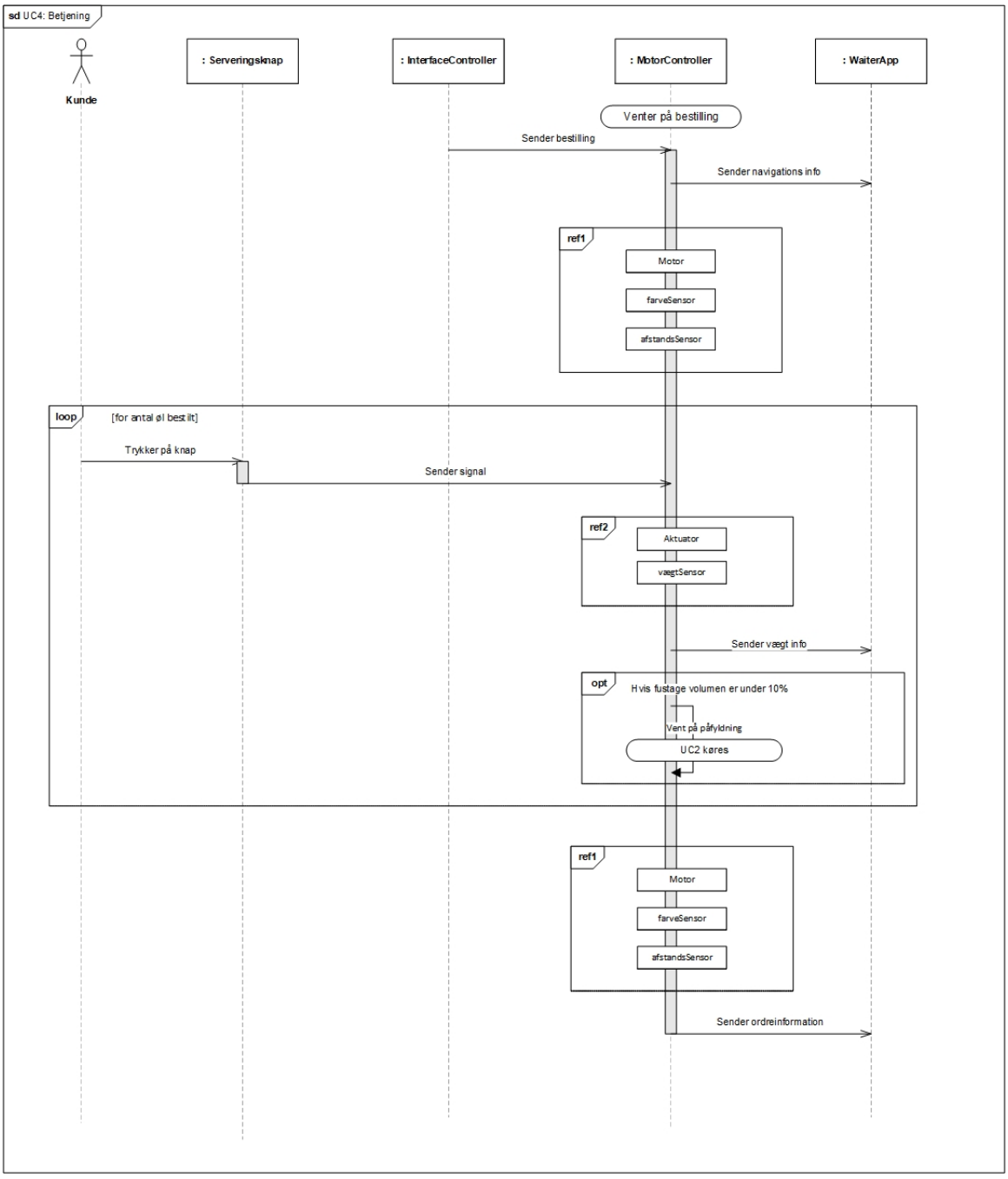
Figur 2: Sekvensdiagram for UC1

Sekvensdiagram for UC3 Figur 3 viser processens udfoldelse for UC3, som omfatter bestilling af øl. Kunden vælger antal øl samt hvilket bordnummer de sidder ved. Når Websiden har registreret disse oplysninger, sendes de til interface controlleren. Der sendes derefter en QR-kode ud på Websiden, hvorefter kunden scanner denne. Nå kunden har scannet QR-koden, sendes der oplysninger tilbage til interface controlleren om at betaling er gennemført. Der sendes herefter en besked ud på websiden, om at betalingen er gennemført. Use Casen afsluttes ved at interface controlleren sender bestillingsinfo til hhv. PC og Motor controlleren. Under PC findes en log som lagrer disse data.



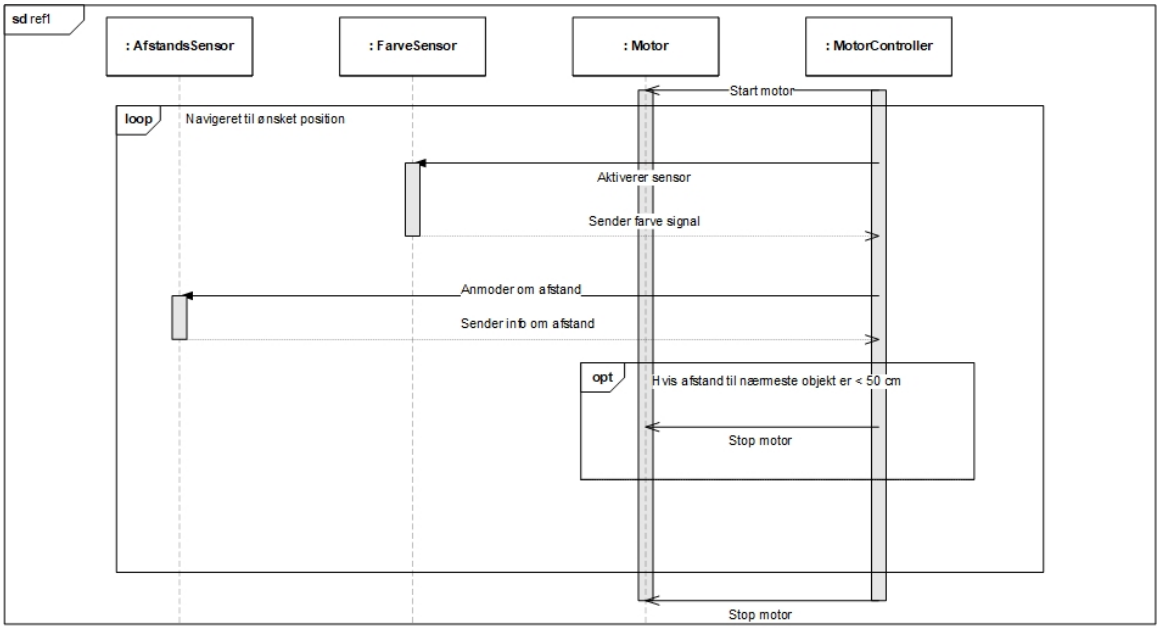
Figur 3: Sekvensdiagram for UC3

Sekvensdiagram for UC4 Figur 4 viser sekvensdiagrammet for UC4. Vores Motor controller modtager først en bestilling fra vores Interface controller, som har modtaget en bestilling fra kunden, hvilket kunne ses i sekvensdiagrammet for UC3. Af overbliksmæssige hensyn er der lavet yderligere sekvensdiagrammer som viser forbindelserne mellem vores Motor controller og de forskellige sensorer som systemet består af. Se ref1 og ref2. Til sidst sendes informationerne om ordren til PC, som gemmer disse i en log.



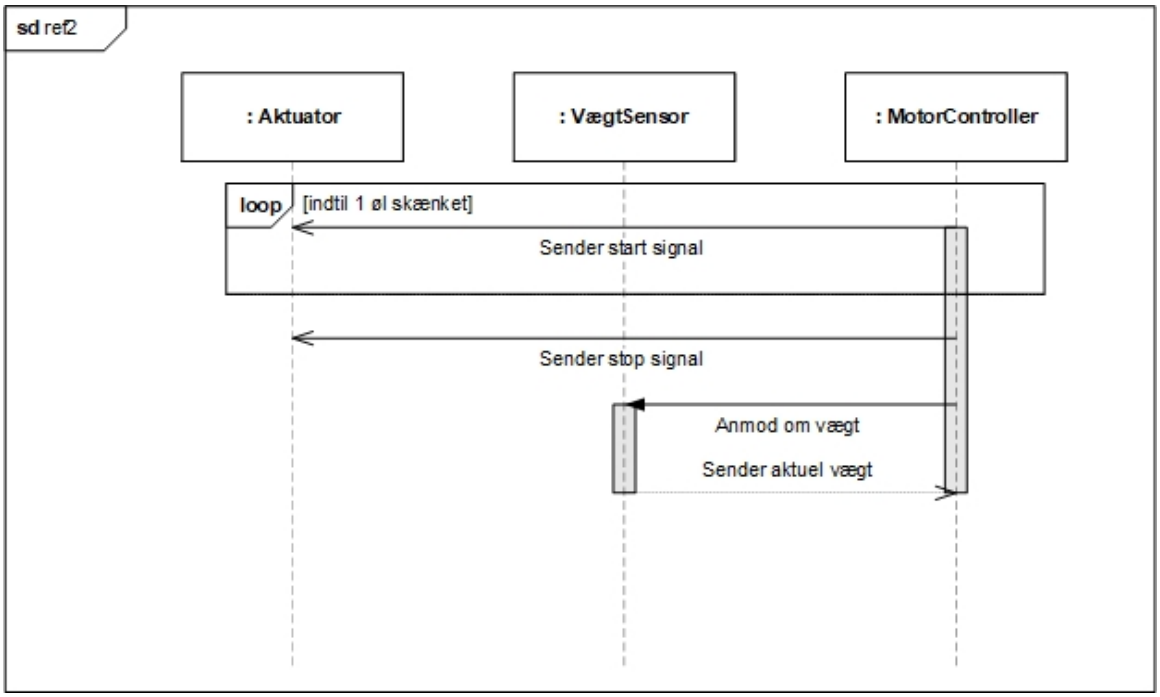
Figur 4: Sekvensdiagram for UC4

Sekvensdiagram for ref1 Figur 5 beskriver hvordan Motor controlleren kommunikerer med vores sensorer. Alt efter hvilket input der er kommet fra Interface controlleren i figur 4, skal Beertress køre til det ønskede bord eller startposition. Afstandsensoren og Motor Controlleren snakker konstant sammen og holder øje med, at objekter ikke kommer foran Beertress. Hvis dette sker, stoppes motoren. Farvesensoren fortæller om Beertress befinder sig på ruten, samt om den når til et bord(disse er markeret med en farve).



Figur 5: Sekvensdiagram for ref1

Sekvensdiagram for ref2 Figur 6 fortæller om situationen, når øllen skal udskænkes. Som ses i figur 4 trykker kunden på en knap som aktiverer aktuatoren. Når øllen er skænket(vi vil tjekke dette via timing, fx kunne det tage 5 sek at skænke en øl), sender vægtsensoren vægten på fustagen til Motor controlleren, så systemet ved, om der skal påfyldes en ny fustage.

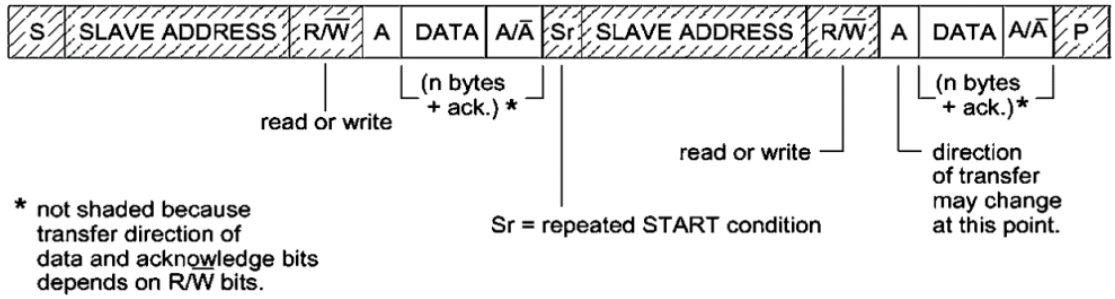


Figur 6: Sekvensdiagram for ref2

2 Grænseflade protokoller

2.1 Beertress Protokol

Som kommunikation mellem vores Motor controller(PSoC) og Interface controlleren(RPi), er der valgt at benytte I2C. I2C kommunikerer via en I2C bus med 2 pins, SDA og SCL, hvor forskellige devices er koblet på. Der vil være en master og x antal slaver, i vores tilfælde én master og én slave. SDA er dataen og SCL er clocken.



Figur 7: I2C - Transmission af data(**?**)

Figur 7 viser kommunikationen mellem en master og en slave. Der laves en start sekvens(S) af masteren hvor både SDA og SCL trækkes lav. Dernæst sendes slave adressen(adressen til vores PSoC), en bit hvor man enten bestemmer om den skal læse eller skrive(i vores tilfælde vil den starte med at skrive), og til sidst sendes et ACK fra slaven, hvis slaven kan godkende adressen. Dernæst sendes eller læses selve dataen til receiveren. Til slut trækkes begge pin høj, hvilket indikerer stop-sekvensen.

Selve protokollen er følgende: Der sendes 3 bytes med data (udover den første byte som angiver slave adressen). Den første byte bestemmer hvilken ordre der er kommet. Det kunne være, at kunden har bestilt øl, cider, kaffe eller noget helt andet. I vores nedskalerede tilfælde vil det altid være øl, men med denne protokol er det blevet skalérbart, så det vil være enkelt at udvide til andre væsketyper. Der vil kunne blive op til 255 forskellige væsketyper. Det andet byte angiver hvilket bord der har ønsket betjening. Der kan være op til 255 borde. Det tredje byte angiver antal af den ønskede væske som bliver angivet i byte 1. Der kan igen være op til 255.

Tabel 1 angiver 3 forskellige væsketyper. Dette kan som sagt udvides til mange flere. Vi bruger kun øl i vores projekt.

	Øl	**Cider**	**Kaffe**
Dataframe:	0b00000001	0b00000010	0b00000011

Tabel 1: Byte 1, RPI-PSOC protokol

Tabel 2 viser 2 eksempler på protokollen. I det første eksempel bestilles der 5 øl til bord 2. I det andet eksempel bestilles 3 cider til bord 1.

	Type	**Bord**	**Antal**
Dataframe, eks1:	0b00000001	0b00000010	0b00000101
Dataframe, eks2:	0b00000010	0b00000001	0b00000011

Tabel 2: Eksempler på dataoverførsel via protokol

Tabel 3 viser de 6 mulige dataoverførsler i dette projekt, da der er blevet nedskaleret ned til max 2 borde og max 3 øl.

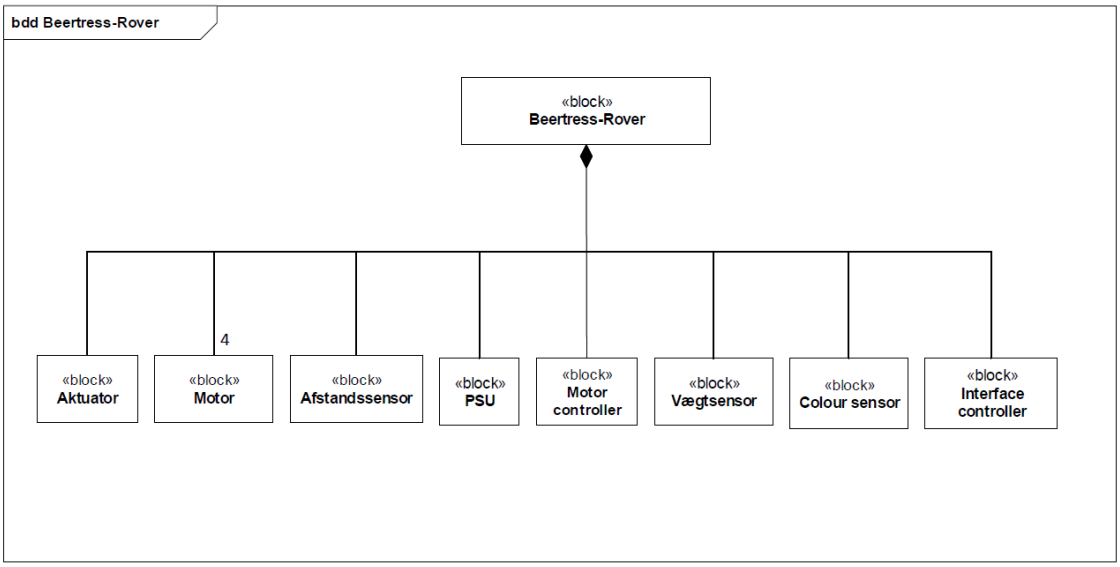
	Type	**Bord**	**Antal**
Bord 1, 1 øl:	0b00000001	0b00000001	0b00000001
Bord 1, 2 øl:	0b00000001	0b00000001	0b00000010
Bord 1, 3 øl:	0b00000001	0b00000001	0b00000011
Bord 2, 1 øl:	0b00000001	0b00000010	0b00000001
Bord 2, 2 øl:	0b00000001	0b00000010	0b00000010
Bord 2, 3 øl:	0b00000001	0b00000010	0b00000011

Tabel 3: Mulige dataoverførsler i nedskaleret udgave

3 Hardware Arkitektur

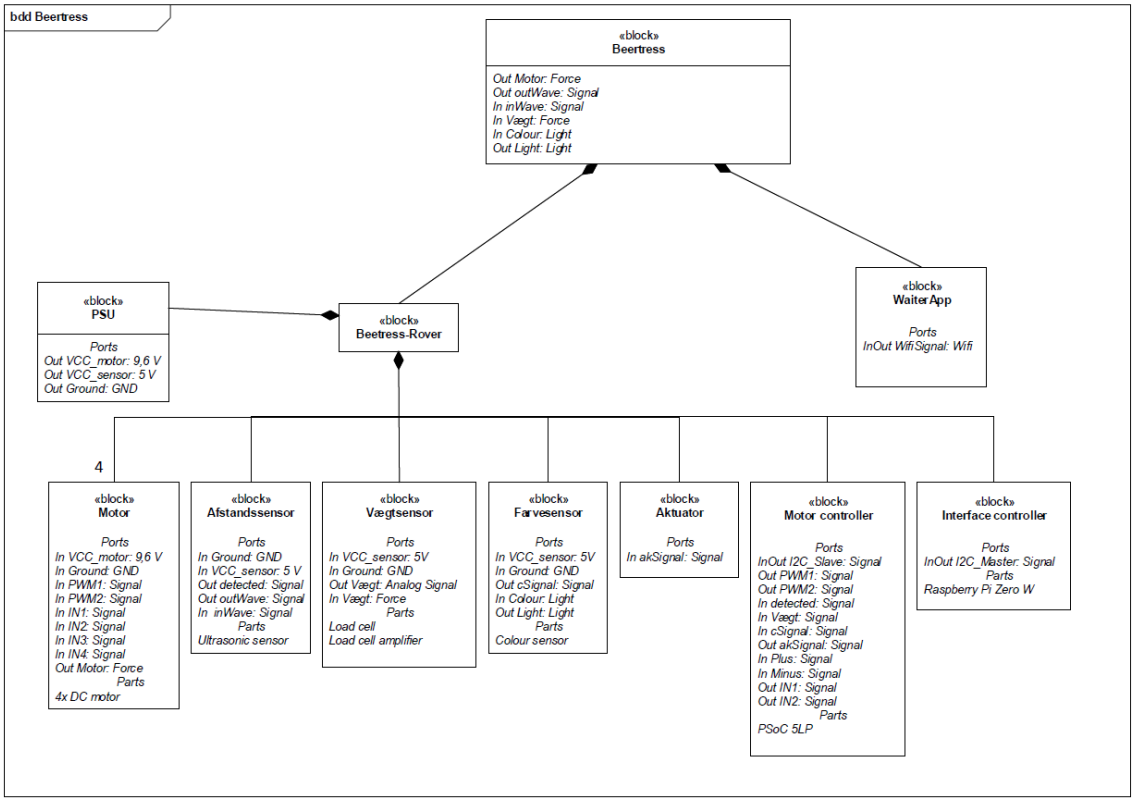
3.1 Block Definition Diagram

På figur 8 kan der ses det overordnet hardware BDD uden porte. Her skabes overblik over alle de forskellige hardware komponenter der findes på Beertress. Her er der blevet udeladt webside og PC i forhold til 1, da der ikke skal bygges en webside eller PC. Webside foregår rent SW, og PC bruges som adgang til personalegrænsefladen, og skal ikke bygges fra bunden. Derfor finder vi ikke dette relevant for Hardware arkitekturen.



Figur 8: Overordnet BDD for Beertress

På figur 9 ses et detaljeret BDD, hvor de forskellige blokkes porte er inkluderet. *Ports* henviser til de forskellige inputs og outputs systemet består af. Der er inkluderet Parts i motor controller, Interface controller og Motor, for at beskrive hvilken enhed der bruges. Der skal altså bruges en PSoC 5LP til at kontrollere alt ved motoren og en Raspberry Pi Zero W til interfacet(websiden og brugergrænsefladen).



Figur 9: BDD med ports for Beertress

3.1.1 Blokbeskrivelse

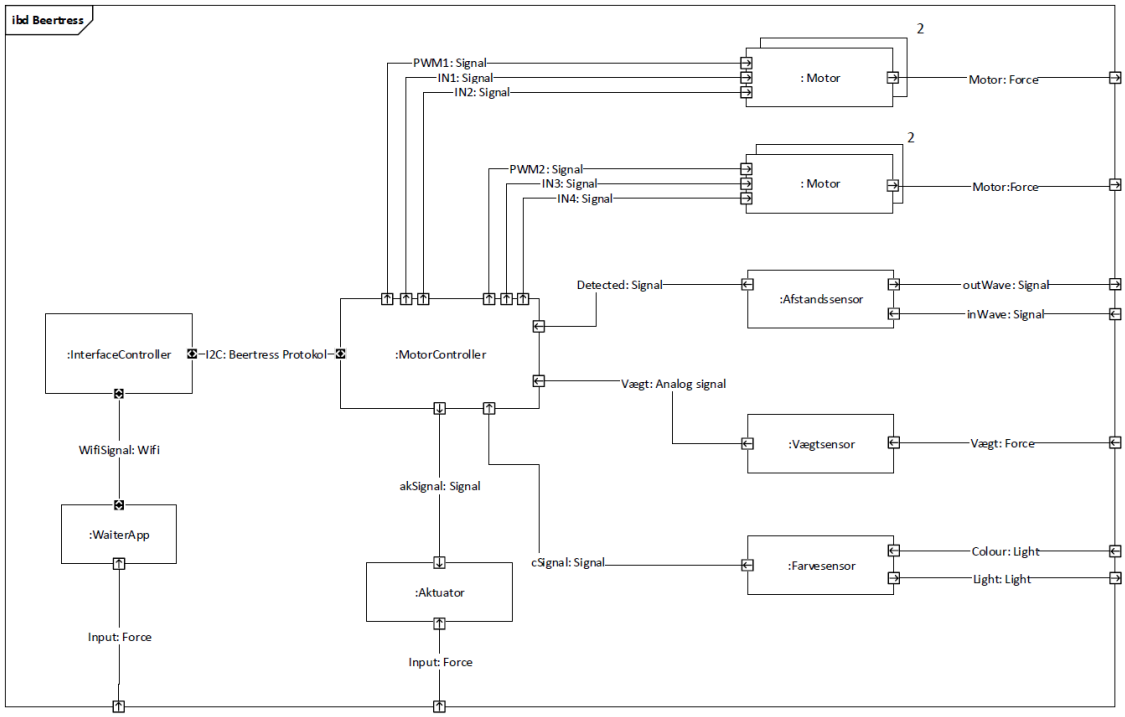
Bloknavn:	**Beskrivelse:**
Motor controller	Proccessering af input på en PSoC 5LP som har inputs og outputs til Motor, Afstandssensor, Vægtsensor og Farvesensor og sender data til interface controller via I2C. Motor Controlleren er slaven i forhold til Interface controlleren.
Interface controller	RPi som håndterer inputs fra Websiden, som er en del af controlleren selv, via requestData og kommunikerer med PC via WiFi. Output sendes til Motor controller via I2C. Interface controlleren er masteren i forhold til Motor controlleren.
Motor	Styring af de 4 forskellige DC motorer ved hjælp af h'broer. Får VCC med 9.6 V fra PSU'en.
Afstandssensor	Detektering af forhindringer foran Beertress. Sender konstant informationer til Motor controlleren.
Vægtsensor	Afvejning af vægt på Øl-indhold. Efter hver skænkning opdateres volumen og sendes til Motor Controlleren. Får 5V VCC fra PSU
Farvesensor	Navigering af Beertress ved hjælp af farvet tape. "Banen"er lavet via farvet tape på gulvet, og farvesensoren holder øje med denne og sender informationer til Motor Controlleren. Får 5V VCC fra PSU.
Aktuator	Åbning og lukning af øl-beholder. Aktiveres når bruger trykker på udskænkningsknap.
PSU	Strømforsyning til de forskellige hardware dele.
WaiterApp	Personale Applikation til styring af Beertress-Rover. Styring fungerer over WiFi.

Tabel 4: BDD blokbeskrivelse

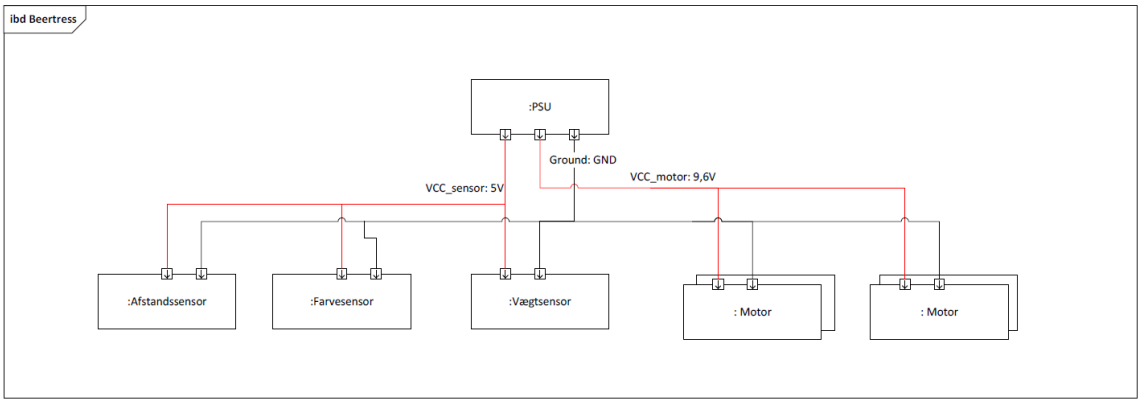
3.2 Internal Block Diagram

3.2.1 IBD diagram

På figur 10 vises IBD'et for Beertress. De interne kommunikationer mellem de forskellige blokke kan ses, og herved fås en dybere forståelse for hvorledes de enkelte dele i Beertress kommunikerer med hinanden. PSU (Power Supply Unit) er blevet udeladt i IBD'en, da det ikke vil give mening at vise, at den forbindes til begge controllers, motor og alle sensorer. Der er blevet landet et seperat IBD som viser hvordan PSU forbindes til de relevante komponenter der kan ses på 11 . Forbindelserne kan ses i figur 9.



Figur 10: Overordnet IBD for Beertress



Figur 11: IBD for PSU forbindelser

3.2.2 Signalbeskrivelser

I dette delafsnit beskrives signalerne fra det interne blok diagram. Der dannes et overblik over signalernes type og funktion.

Bloknavn	**Funktionsbeskrivelse**	**Portnavn**	**Type**	**Port-specifikation**
Motor controller	*Beertress*'s PSoC der styrer sensorer og motor.	PWM	Signal	PWM signal der sendes til DC motorer
		Detected	Signal	Signal der sender om forhindringer opstået
		Vægt	Signal	Signal der informerer den nuværende vægt
		cSignal	Signal	Signal der fortæller nuværende position
		akSignal	Signal	Signal der giver aktuator besked når der skal serveres øl
		I2C	Signal	I2C forbindelse mellem PSoC og RPI

Tabel 5: Signalbeskrivelse for Motor controller

Bloknavn	Funktionsbeskrivelse	Portnavn	Type	Port-specifikation
Interface controller	*Beertress*'s RPI der håndtere grænseflader og kommunikation til Motor controller	I2C	Signal	I2C forbindelse med Motor controller
		requestData	data	Webside data information omkring bestiling
		Wifi	Signal	Signal fra Interface controller til PC ved hjælp

Tabel 6: Signalbeskrivelse for Interface controller

Bloknavn	Funktionsbeskrivelse	Portnavn	Type	Port-specifikation
Motor	*Beertress*'s Motor der håndtere bevægelse.	PWM	Signal	PWM signal fra Motor controller der bestemmer hastighed
		Motor	Force	Mekanisk kræft som motoren producere

Tabel 7: Signalbeskrivelse for Motor

Bloknavn	Funktionsbeskrivelse	Portnavn	Type	Port-specifikation
Afstandssensor	*Beertress*'s sensor som detektere forhindringer foran den.	Detected	Signal	Signal som sensor sender til Motor control når der detekteres en forhindring
		outWave	Signal	Lydbølge som sendes ud
		inWave	Signal	Lydvbølgen som var sendt ud der blevet reflekteret tilbage

Tabel 8: Signalbeskrivelse for Afstandssensor

Bloknavn	**Funktionsbeskrivelse**	**Portnavn**	**Type**	**Port-specifikation**
Vægtsensor	*Beertress*'s sensor som detektere vægten på ølindholdet	Vægt	Analog Signal	Analog Signal som sensor sender til Motor control der giver info omkring vægt
		Vægt	Force	Vægten som bliver sat på loadcell

Tabel 9: Signalbeskrivelse for vægtsensor

Bloknavn	**Funktionsbeskrivelse**	**Portnavn**	**Type**	**Port-specifikation**
Farvesensor	*Beertress*'s sensor som fortæller position.	Colour	Light	Farve der detekteres i sensoren
		Light	Light	Lys der sendes ud fra sensoren
		cSignal	Signal	Signal med nuværende position der sendes til motor control

Tabel 10: Signalbeskrivelse for Farvesensor

Bloknavn	**Funktionsbeskrivelse**	**Portnavn**	**Type**	**Port-specifikation**
WaiterApp	*Beertress*'s PersonaleApp.	WifiSignal	Wifi	Wifi signal der forbindet Applikation med InterfaceController
		Input	Force	PersonaleApp Input

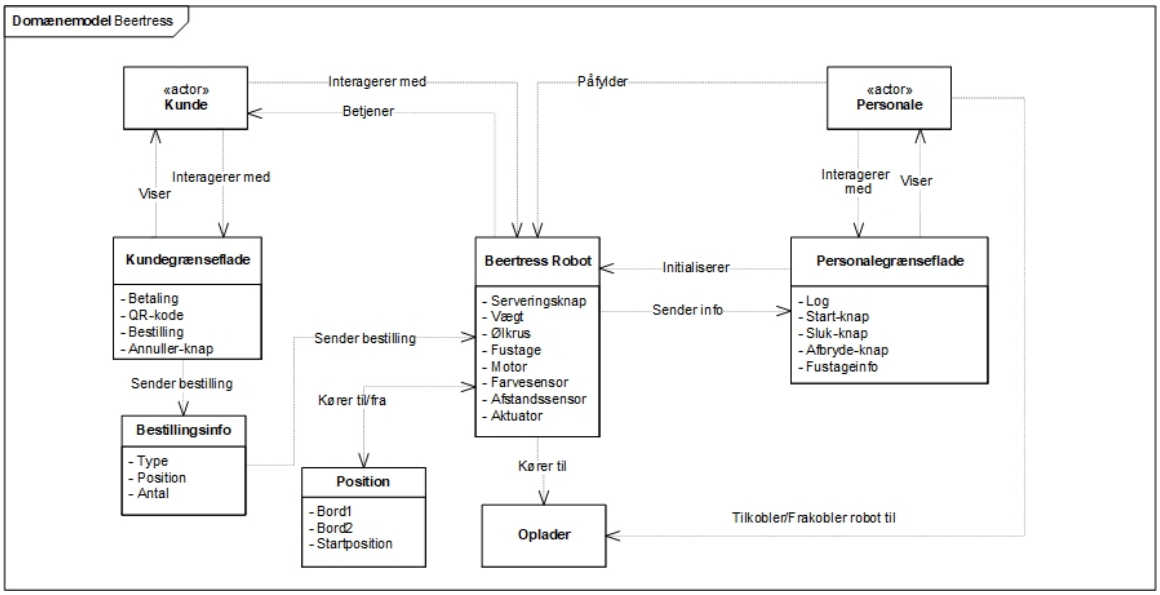
Tabel 11: Signalbeskrivelse for WaiterApp

4 Software Arkitektur

I de følgende afsnit vil software arkitekturen blive beskrevet. Først indledes der med en domænemodel for at vise et overblik over systemet og dernæst applikationsmodeller for hver block i BDD, på figur 4, der skal have software allokering.

4.1 Domænemodel

Figur 12 viser en domænemodel for systemet Beertress. Denne har til formål at vise et overblik over systemet. Denne er blevet udarbejdet ved hjælp af vores Use Cases ved at kigge efter navne- og udsagnsord.



Figur 12: Domænemodel for Beertress

4.2 Software Allokering

Som der kan ses i BDD på figur 4 skal der software allokeres til tre blokke: InterfaceController, MotorController og WaiterApp

WaiterApp_SW: Denne software skal være et program der kan bruges på f.eks. personalets PC, så personalet kan starte og slukke Beertress Rover, samt se en log over bestillinger og volume på

fustage som bliver opdateret af InterfaceController.

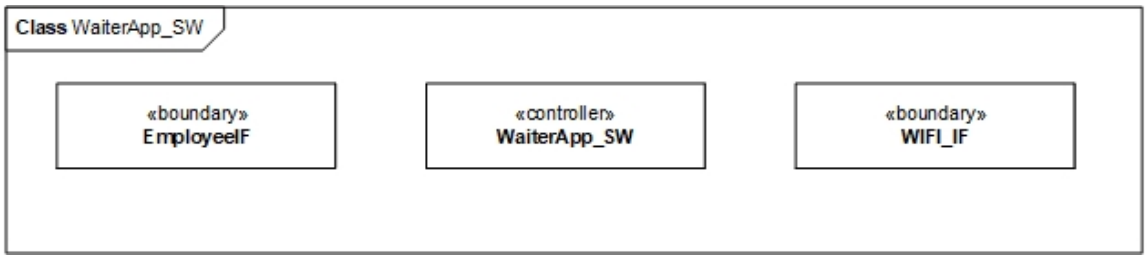
InterfaceController_SW: Raspberry Pi skal have software så en hjemmeside kan tilgås af kunden via WiFi hotspot, hvor kunden kan bestille til sit ønsket bord. Dernæst skal den kunne kommunikere med MotorControlleren via I2C, så MotorControlleren kan navigere ned til bordet og servere bestillingen, samt opdatere loggen som WaiterApp kan tilgå.

MotorController_SW: PSoC skal have software så den kan håndtere motor, farvesensor, afstandsensor, vægtsensor og aktuator så den kan navigere hen til det korrekte bord og servere bestillingen. Den skal kunne modtage ordre fra InterfaceController og sende information tilbage når ordre er færdig og om fustage eventuelt skal genopfyldes.

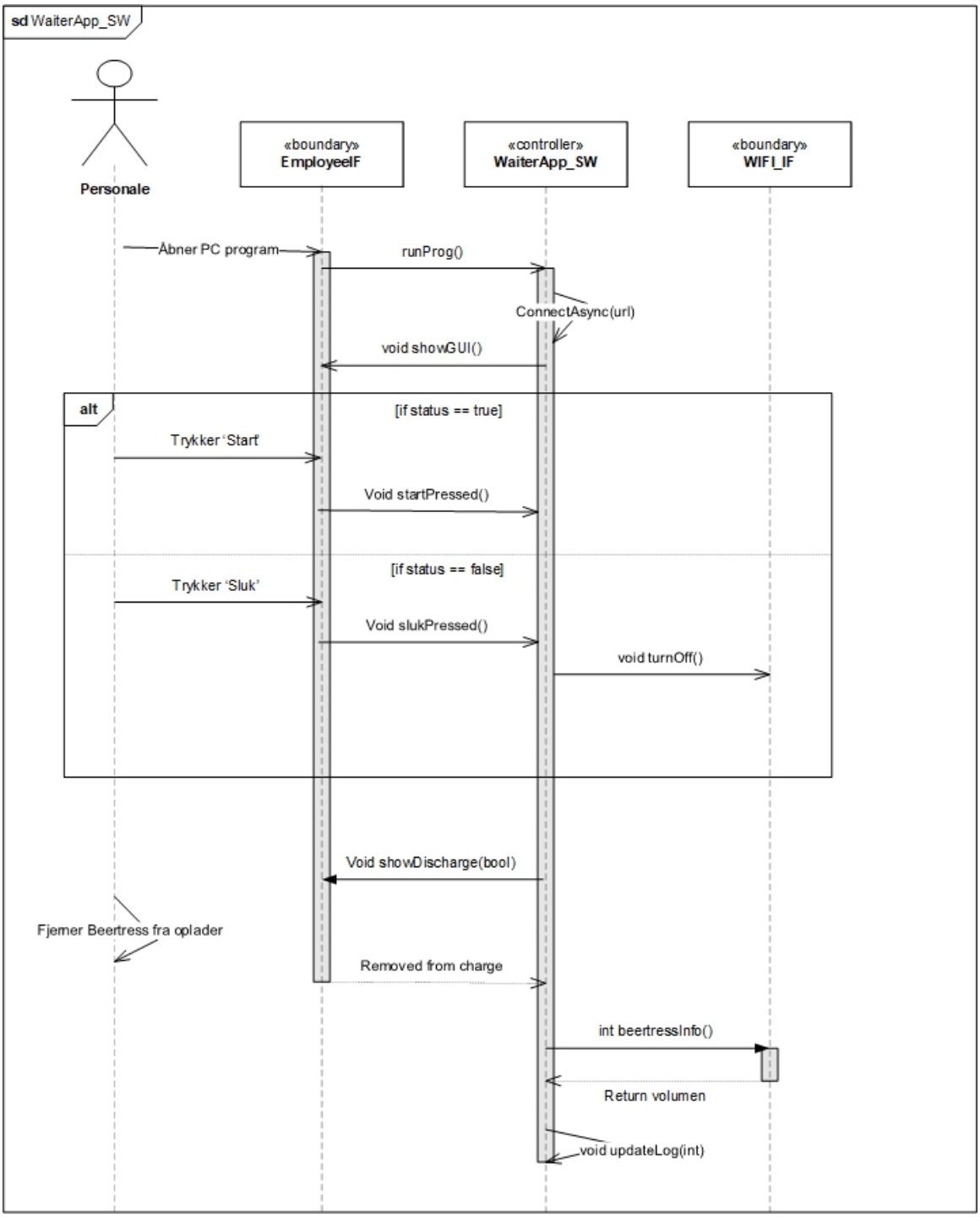
4.2.1 Applikationsmodel WaiterApp_SW

I det følgende afsnit beskrives den overordnede software arkitektur for WaiterApp_SW ved brug af klassediagrammer og sekvensdiagrammer. Disse er blevet udarbejdet med udgangspunkt i Use Case 1 samt domænemodellen.

Klassediagrammet kan ses på figur 13 og har en controllerklasse der aktiveres når personalet tænder programmet. Boundaryklassen EmployeeIF sørger for at vise det grafiske interface til personalet hvorpå de kan starte, slukke og se logbeskeder. Boundaryklassen WIFI_IF sørger for kommunikationen til InterfaceController via WiFi, så InterfaceController kan hente information om volumen samt opstarte og nedlukke MotorController (og dermed Beertress Rover). Sekvensdiagrammet med metodekald kan ses på figur 14.

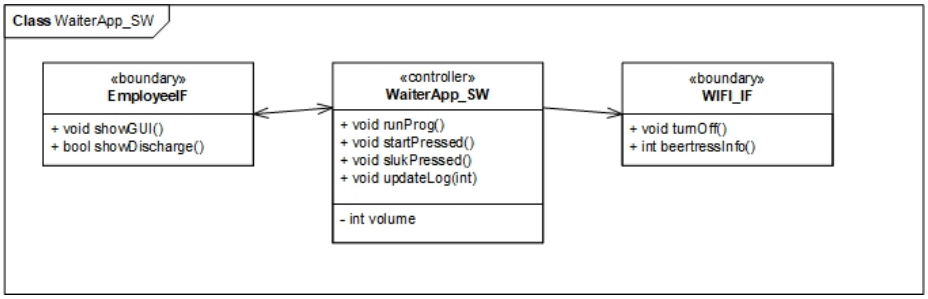


Figur 13: Klassedigram for WaiterApp_SW



Figur 14: Sekvensdiagram for WaiterApp_SW

Udfra sekvensdiagrammet kan vi nu færdiggøre klassediagrammet med metoder og variabler. Disse kan ses på figur 15

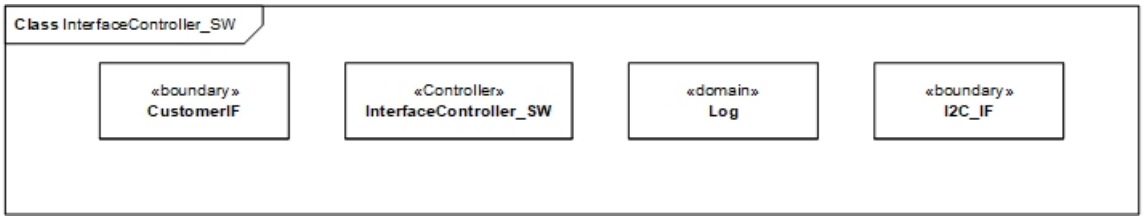


Figur 15: Klassediagram for WaiterApp_SW med metodekald

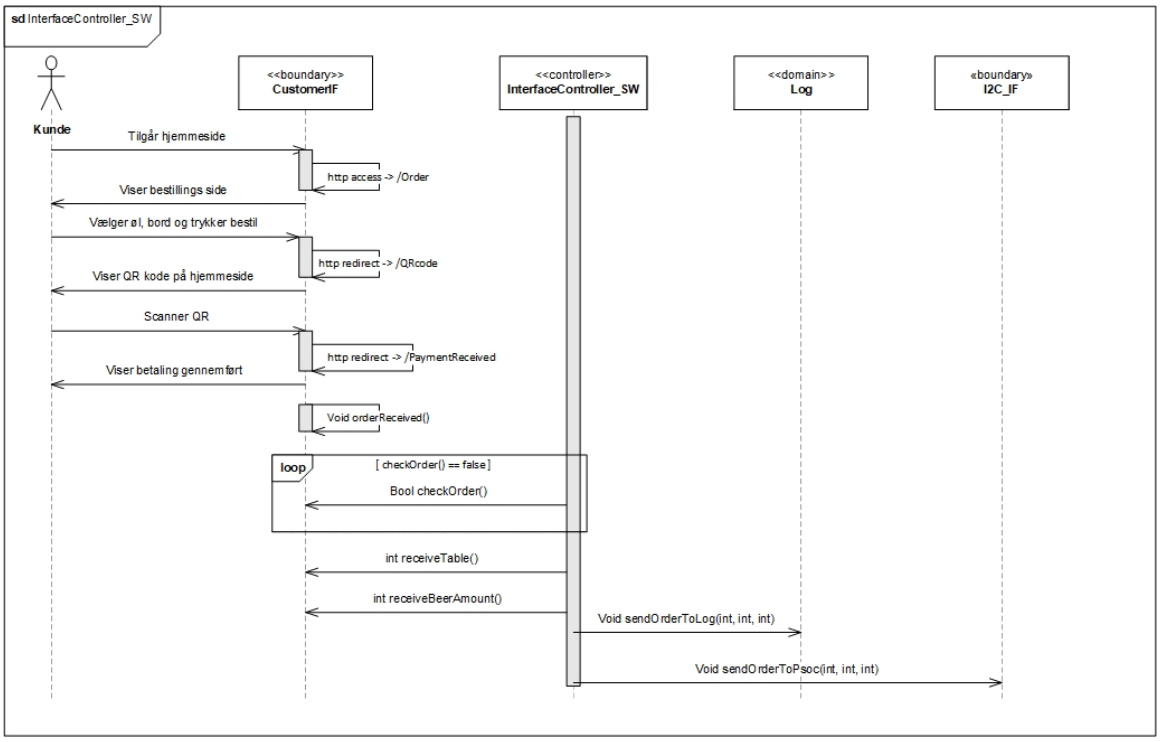
4.2.2 Applikationsmodel InterfaceController_SW

I det følgende afsnit beskrives den overordnede software arkitektur for InterfaceController_SW ved brug af klassediagrammer og sekvensdiagrammer. Disse er blevet udarbejdet med udgangspunkt i Use Case 3 samt domænemodellen.

Klassediagrammet kan ses på figur 16 og har en controllerklasse der modtager ordre hos boundaryklassen CustomerIF og henter information om bestillingen. Den sørger for dette bliver registreret i domæneklassen Log og sender bestillingsinformationen videre til MotorController via boundaryklassen I2C_IF. Sekvensdiagrammet med metodekald kan ses på figur 17.

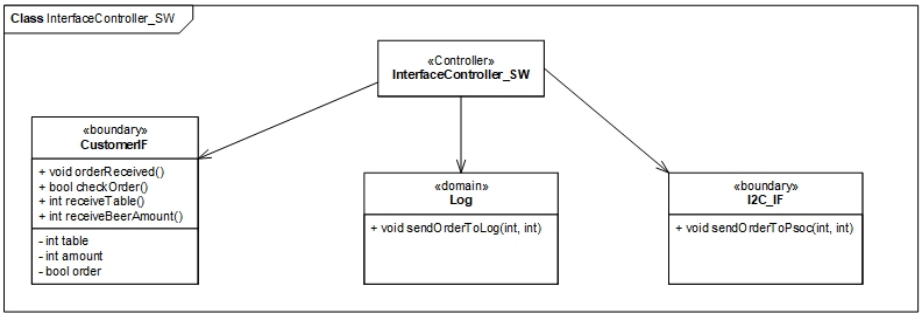


Figur 16: Klassedigram for InterfaceController_SW



Figur 17: Sekvensdiagram for Interfacecontroller_SW

Udfra sekvensdiagrammet kan vi nu færdiggøre klassediagrammet med metoder og variabler. Disse kan ses på figur 18

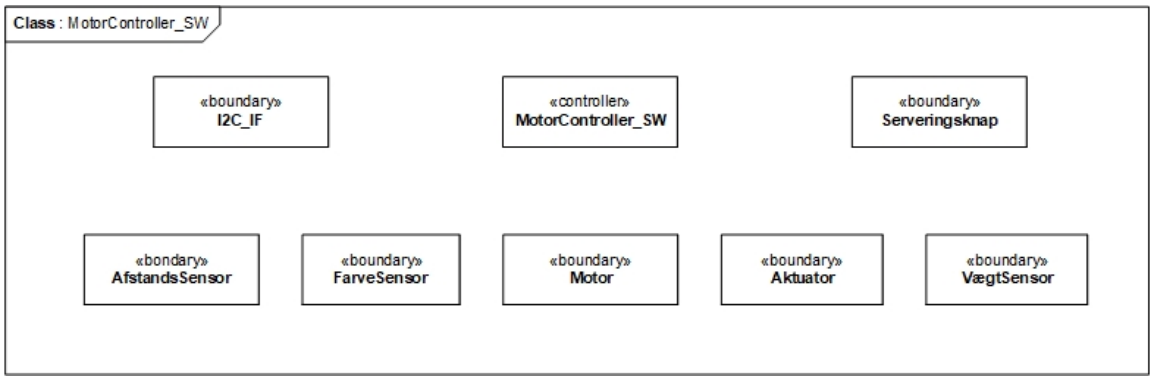


Figur 18: Klassediagram for InterfaceController_SW med metodekald

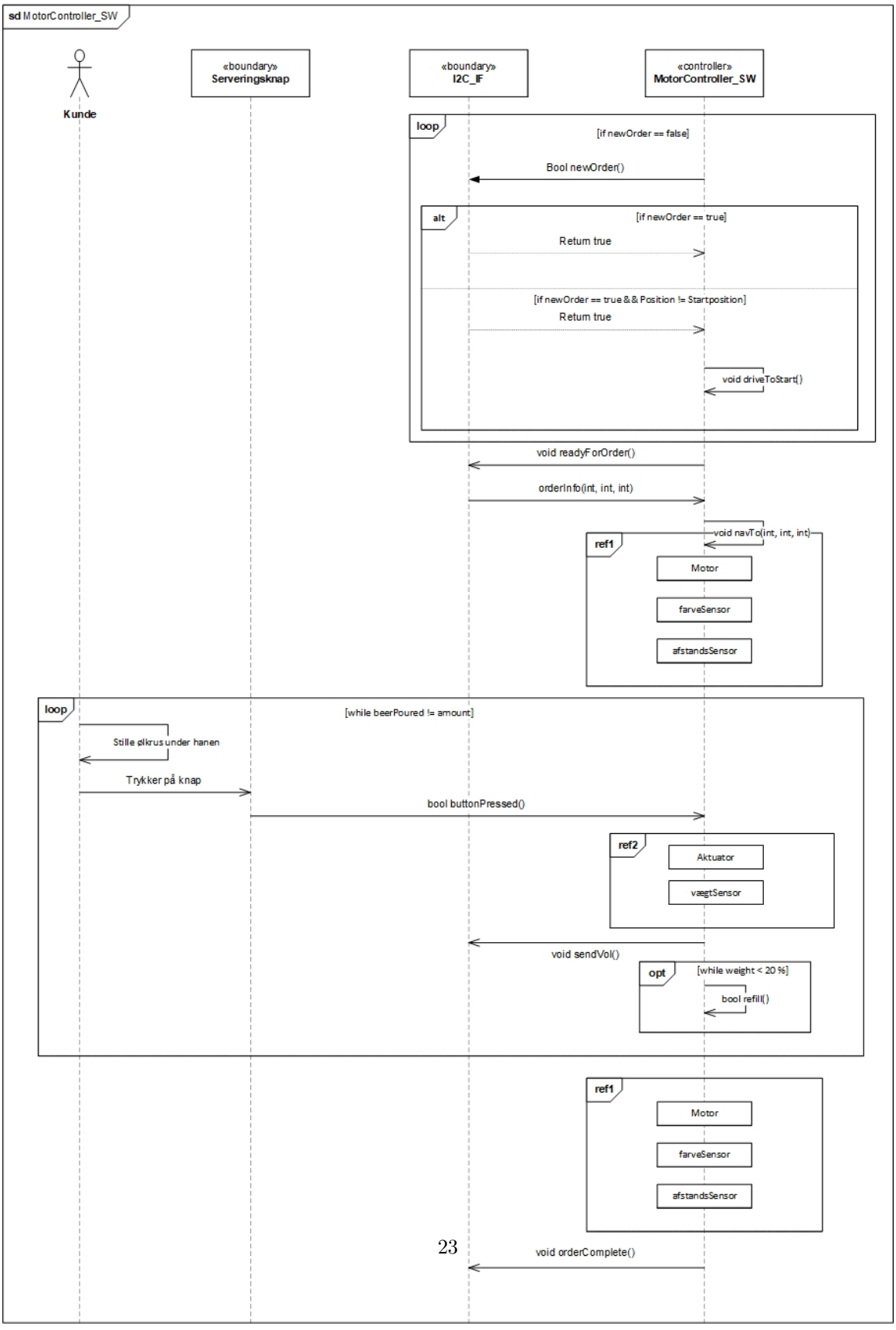
4.2.3 Applikationsmodel MotorController_SW

I det følgende afsnit beskrives den overordnede software arkitektur for MotorController_SW ved brug af klassediagrammer og sekvensdiagrammer. Disse er blevet udarbejdet med udgangspunkt i Use Case 4 samt domænemodellen.

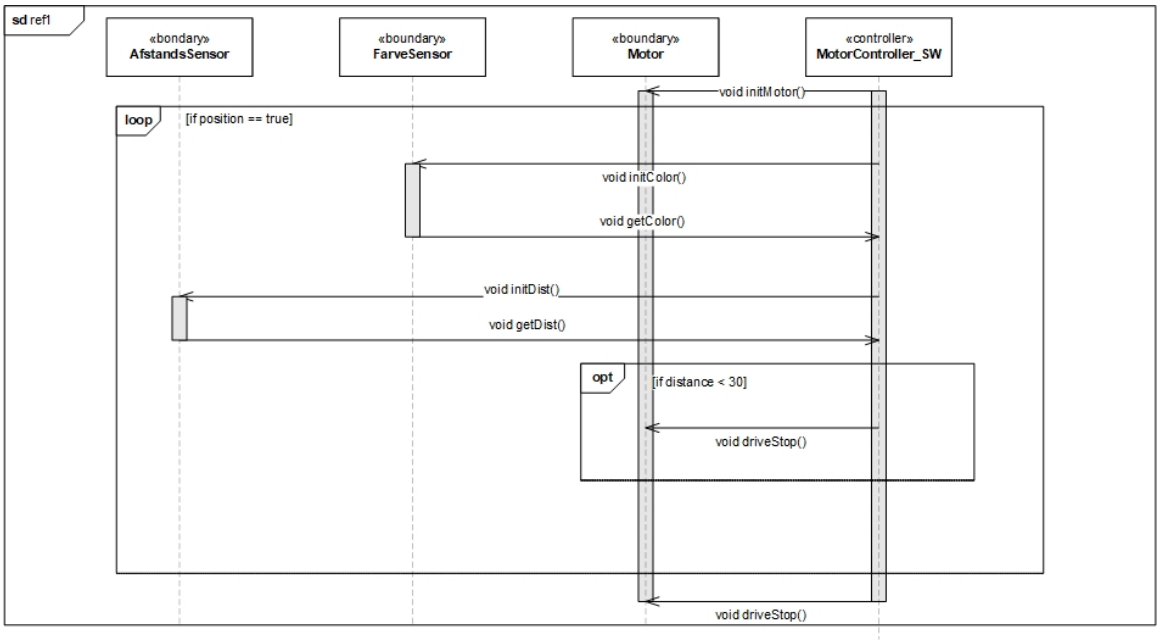
Klassediagrammet kan ses på figur 19 og har en controllerklasse der modtager bestillingsinformation fra InterfaceController via boundaryklassen I2C_IF og sender information tilbage igen når ordren er afsluttet. Dernæst navigerer MotorControlleren banen og serverer bestillingen til kunden ved hjælp af de andre boundaryklasser. Sekvensdiagrammet for MotorController_SW samt ref1 og ref2 kan ses på figur 20, 21 og 23 med metodekald.



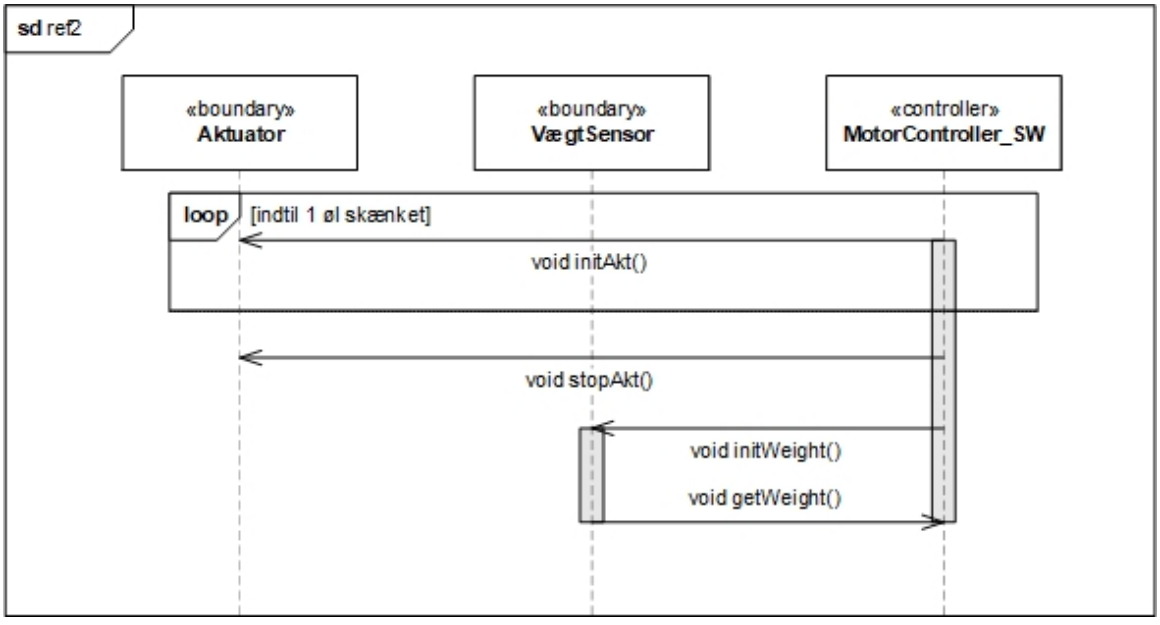
Figur 19: Klassediagram for MotorController_SW



Figur 20: Sekvensdiagram for MotorController_SW



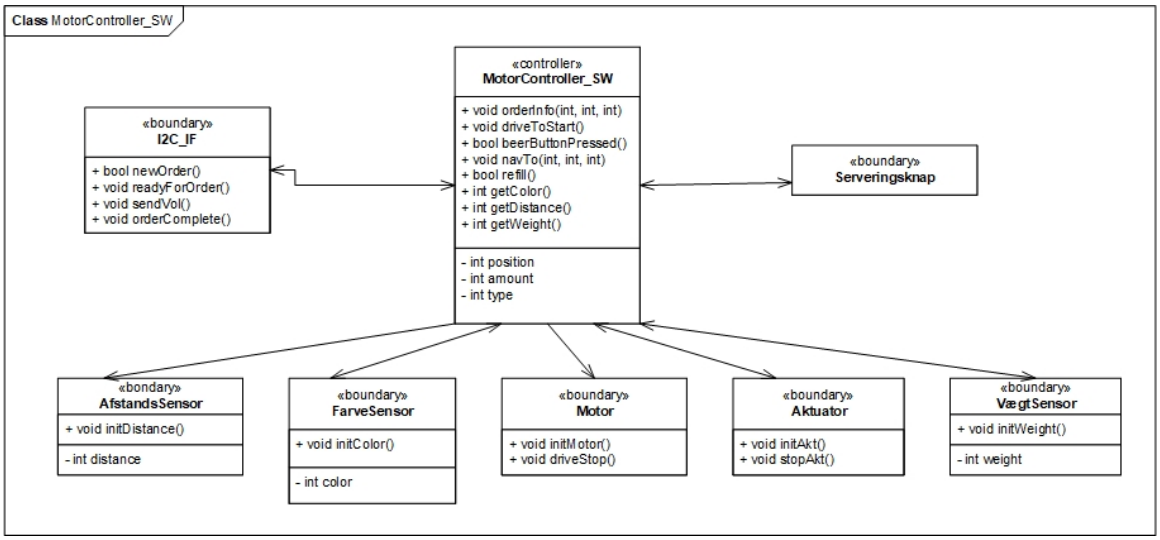
Figur 21: Sekvensdiagram for ref1



Figur 22: Sekvensdiagram for ref2

Udfra sekvensdiagrammet kan vi nu færdiggøre klassediagrammet med metoder og variabler. Disse

kan ses på figur 18



Figur 23: Klassediagram for MotorController_SW med metodekald