



3. Semesterprojekt

Smart Wagon

Bilagsrapport

Projektgruppe 2

Forår 2021

Deltagere		
Studienummer	Navn	Studieretning
201705103	Andreas Stavning Erslev	Softwareteknologi
201807859	Asger Busk Breinholm	Softwareteknologi
199981001	Azar Kwame Martin	Softwareteknologi
201911338	Christina Boll Pedersen	Softwareteknologi
201904202	Johanne Berg	Softwareteknologi
201910327	Maagisha Mahenthirarajan	Softwareteknologi

Vejleder	
Jesper Michael Kristensen	jmkr@ece.au.dk



Indholdsfortegnelse

1 Kravspecifikation	3
1.1 Systemspecifikation	3
1.2 Funktionelle krav	4
1.3 Fully dressed Use Case beskrivelser	4
1.3.1 Use Case 1: Kør efter kunde	5
1.3.2 Use Case 2: Registrer vare	6
1.3.3 Use Case 3: Fjern vare	7
1.3.4 Use Case 4: Foretag fejlhåndtering	8
1.4 MoSCoW ift. Funktionelle krav	8
1.5 MoSCoW af betaversionen for de funktionelle krav	9
1.6 Ikke funktionelle krav	9
1.7 Betaversionen af de ikke-funktionelle krav	10
1.8 MoSCoW ift. ikke Funktionelle krav	10
1.9 MoSCoW af betaversionen af de ikke-funktionelle krav	11
2 Afgrænsning	12
3 Teknologianalyse	13
3.1 Valg af afstandssensor til Smart Wagons kørefunktion	13
3.2 Valg af Centralsystem	14
3.2.1 Level converter	14
3.3 Forbindelse imellem dataprocesseren og Centralsystemet	14
3.4 Valg af CPU til hardware delsystemer	14
3.4.1 Vægt	14
3.4.2 DC-motor	15
3.5 Valg af brugergrænseflade	15
3.5.1 Valg af Raspberry Pi som host for hjemmesiden	15
4 Risikoanalyse	16
4.1 Projektplan	17
5 Systemarkitektur	18
5.1 Block Definition Diagram	18
5.2 Internal Block Diagram	19
5.3 Deployment view	19
5.4 Domænemodel	20
5.5 Sekvensdiagrammer for handling	20
5.5.1 Sekvensdiagram for UC1: Kør efter kunde	21
5.5.2 Sekvensdiagram for UC2: Registrer vare	22
5.5.3 Sekvensdiagram for UC3: Fjern vare	23
5.5.4 Sekvensdiagram for UC4: Foretag fejlhåndtering	24
6 Design	25
6.1 Hardware design	25
6.1.1 Design af vægten	27
6.1.2 Design af motor	27
6.1.3 Serielforbindelse mellem PSoC og RPi	28
6.2 Komponentliste	28
6.3 Software design	29
6.3.1 Applikationsmodel baseret på beta modellen af SmartWagon	29
6.3.2 Klassediagram for det samlede system	35
6.3.3 Revurdering af applikationsmodellen for betaversionen af Smart Wagon	36
7 Implementering	42
7.1 Hardware implementering	42



7.1.1	Implementering af motor	42
7.1.2	Implementering af vægt	43
7.2	Software implementering	44
7.2.1	Implementering af GUI	44
7.2.2	Motor implementering	47
7.2.3	Implementering af vægt	48
8	Accepttest	50
8.1	Accepttest for betaversionen af de funktionelle krav	50
8.1.1	Accepttest for Use Case 1: Kør efter kunde	50
8.1.2	Accepttest for Use Case 2: Registrer vare	50
8.1.3	Accepttest for Use Case 3: Fjern vare	51
8.1.4	Accepttest for Use Case 4: Foretag fejlhåndtering	51
8.2	Accepttest for betaversionen af de ikke-funktionelle krav	51
8.3	Konklusion for accepttesten	52
9	Bilagsliste	53
10	Referenceliste	54

1 Kravspecifikation

I dette afsnit forekommer systemspecificeringen, hvoraf systemet vil blive beskrevet ud fra et aktør-kontekstdiagram og Fully dressed Use Case beskrivelser. Herudover vil Smart Wagons funktionelle og ikke funktionelle krav blive beskrevet og analyseret med MoSCoW analysemodellen.

1.1 Systemspecifikation

Smart Wagon er en indkøbsvogn designet til at kunne identificere forskellige varer ud fra deres vægt. I Smart Wagon er der en applikation indbygget, hvorpå en indkøbsliste automatisk bliver opdateret, således at kunden hele tiden kan holde øje med, hvilke varer der er blevet handlet og summen af det totale beløb. Aktør kontekstdiagrammet, i figur 1 beskriver interaktionen mellem systemets aktører.

Smart Wagon har to primære aktører, kunden og personalet. Grunden til at personalet opträder som en primær aktør er fordi, at dette personale igangsætter Use Case 4, som kan ses i figur 2, da personalet skal foretage fejlhåndtering på Smart Wagon.

Centralsystemet er Smart Wagons sekundære aktør og dens opgave er at sætte en sammenhæng mellem Smart Wagons individuelle delsystemer. Delsystemerne består af en sensor i form af en vægt, en aktuator i form af en DC motor og en hjemmeside som symbolisere en brugergrænseflade. Centralsystemet har yderligere en database som opbevarer informationer om de forskellige varer.

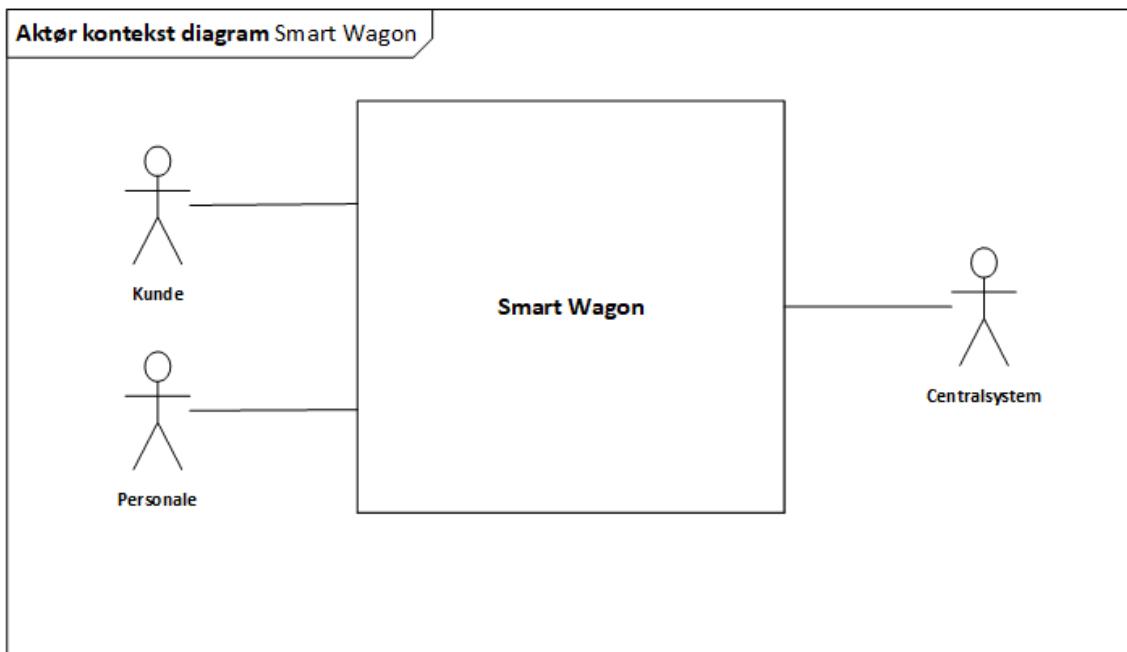


Figure 1: Aktør Kontekst Diagram

1.2 Funktionelle krav

Figur 2 viser et Use Case diagram for den fulde funktionelle Smart Wagon, som viser hvilke tiltænkte interaktioner der er mellem Smart Wagon og de tilknyttede aktører. Det ses, at Smart Wagon er bygget op af fire Use Cases, samt to primære aktører og en sekundær aktør. Forbindelserne viser, hvilke aktører der har indflydelse på hvilke Use Cases.

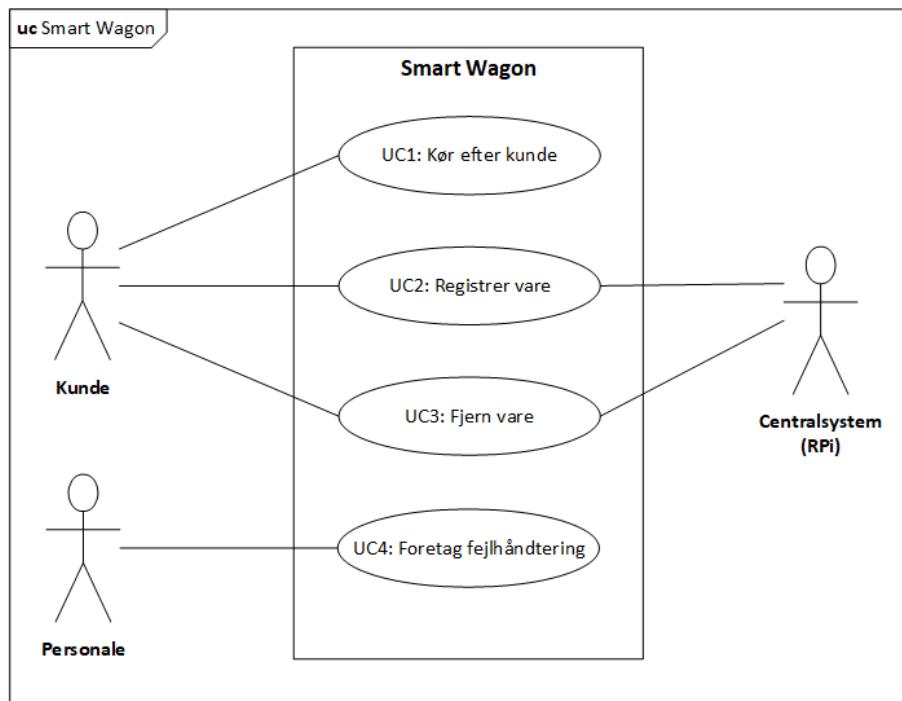


Figure 2: Use Case diagram for Smart Wagon

1.3 Fully dressed Use Case beskrivelser

I dette afsnit vil Smart Wagons Use Cases blive beskrevet ved brug af fully dressed Use Case beskrivelse. Formålet med anvendelsen af denne model er at detaljeret kunne beskrive systemets funktioner og mulige fejl. Der gøres opmærksom på at de fully dressed Use Cases beskrivelser beskriver de funktioner som var tiltænkt Smart Wagon før det blev besluttet at implementere en betaversion.



1.3.1 Use Case 1: Kør efter kunde

Use Case 1 beskriver Smart Wagons køre funktion, hvor Smart Wagon kan følge efter kunden, når den registrerer at kunden er mere end 1 meter væk, samt stoppe, når kunden kommer indenfor 1 meters afstand.

Navn	Kør efter kunde
Mål	Smart Wagon følger efter kunden og kan stoppe når kunden stopper
Initiering	Kunde tilgår Smart Wagon via GUI
Aktør	Primær: Kunde
Antal samtidige forekomster	Ingen
Prækondition	Smart Wagon står ved indgangen til butikken.
Postkondition	Smart Wagon har fulgt efter kunden og er afsat ved kassen i butikken.
Hovedscenarie	<ol style="list-style-type: none">1. Kunden tilgår Smart Wagon via GUI2. Smart Wagon registrerer tilgang3. Kunden begynder at gå4. Smart Wagon anvender ”distancesensor” for at registre om kunden er mere end 1 meter væk, og Smart Wagon kører frem, for at følge efter kunden. [EXT. 1: Smart Wagon nærmer sig en forhindring]5. Kunden stopper6. Smart Wagon registrerer at afstanden til kunden er mindre end en halv meter og bremser op7. Punkt 3-5 gentages indtil kunden er færdig med at handle8. Kunden afleverer Smart Wagon ved kassen.
Udvidelser/undtagelser	EXT. 1: Smart Wagon nærmer sig en forhindring 1. Smart Wagon registrerer at der er en forhindring i vejen og stopper for at finde ny rute.

Table 1: Fully dressed Use Case beskrivelse for UC1



1.3.2 Use Case 2: Registrer vare

Use Case 2 beskriver hvordan Smart Wagon registrere, hvilken vare kunden lægger i kurven, ud fra en indbygget vægt i Smart Wagon. Hvis Smart Wagon ikke genkender varens vægt, skal den kunne give denne besked videre til kunden, og bede kunden om at løse problemet eller tilkalde hjælp.

Navn	Registrer vare
Mål	Smart Wagon har scannet og vejet varen for at sikre at varens stregkode og vægt er overensstemmelse således at varen kan registreres i Centralssystemet.
Initiering	Kunden scanner en vares stregkode med stregkodescanneren
Aktør	Primær: Kunde Sekundær: Centralssystem
Antal samtidige forekomster	Ingen
Prækondition	Smart Wagon er klar til at scanne en vare.
Postkondition	Smart Wagon har scannet og registreret at en vare er lagt ned i kurven.
Hovedscenarie	<ol style="list-style-type: none">1. Kunden scanner en vare2. Smart Wagon registrerer varens stregkode3. Centralssystemet registrerer og tjekker om varen er i systemet [EXT. 1: Kender ikke varen]4. Smart Wagon beder kunden om at lægge varen i kurven, "Venligst placér varen i kurven". [EXT. 2: Registrerer ingen vægt]5. Smart Wagon registrerer om varens vægt passer med varens beskrivelse i centralssystemet [EXT. 3: Varens vægt stemmer ikke overens med systemet]6. Smart Wagon tilføjer varen til kundens indkøbsliste7. Smart Wagon er klar til at scanne en ny vare
Udvidelser/ undtagelser	<p>EXT. 1: Kender ikke varen</p> <ol style="list-style-type: none">1. Smart Wagon kan ikke koble stregkoden til en vare der er registreret i dens system.2. Smart Wagon udskriver fejlbesked, "Fejl ved scanning af vare, prøv igen eller kontakt personale".3. Hop til UC4: "Foretag fejlhåndtering"4. UC2 afsluttes <p>EXT. 2: Registrerer ingen vægt</p> <ol style="list-style-type: none">1. Smart Wagon har ikke registreret at en vare er blevet lagt i kurven indenfor 10 sekunder efter scanningen.2. Smart Wagon sender fejlbesked: "Kunne ikke registrere varen, placere varen i indkøbskurven eller kontakt personalet"3. Hop til UC4: "Fortag fejlhåndtering"4. UC2 afsluttes <p>EXT. 3: Varens vægt stemmer ikke overens med systemet</p> <ol style="list-style-type: none">1. Smart Wagon registrerer ikke den korrekte vægt og udskriver en fejlbesked, "Fejl ved registrering af vægt, venligst kontroller varen eller kontakt personale".2. Hop til UC4: "Foretag fejlhåndtering"3. UC2 afsluttes

Table 2: Fully dressed Use Case beskrivelse for UC2



1.3.3 Use Case 3: Fjern vare

Use Case 3 beskriver hvordan der kan fjernes vare fra indkøbslisten. Dette gøres ved at kunden tilgår indkøbslisten, der fungerer som en vareoversigt, over varene der ligger i Smart Wagon, og tillader kunden at kunne fjerne en vare fra Smart Wagon. Indkøbslisten forekommer på en brugergrænseflade på en skærm.

Navn	Fjern vare
Mål	Der er fjernet en vare fra Smart Wagon og indkøbslisten er opdateret
Initiering	Kunden tilgår indkøbslisten på GUI
Aktør	Primær: Kunde Sekundær: Centralsystem
Antal samtidige forekomster	Ingen
Prækondition	Smart Wagon er klar til brug
Postkondition	Der er blevet lavet en ændring i indkøbslisten
Hovedscenarie	<ol style="list-style-type: none">1. Kunden tilgår indkøbslisten2. GUI viser indkøbslisten3. Kunden vælger hvilken vare som skal fjernes4. Smart Wagon registrerer den valgte vare, som skal fjernes5. Bruger tager vare op af Smart Wagon. [EXT. 1: Vare bliver ikke fjernet]6. Smart Wagon registrerer at varens vægt er blevet fjernet fra kurven [EXT. 2: Fjernet vægt stemmer ikke med varens registrerede vægt]7. Smart Wagon fjerner varer fra indkøbsliste.
Udvidelser/ undtagelser	<p>EXT. 1: Vare bliver ikke fjernet</p> <ol style="list-style-type: none">1. Smart Wagon meddeler en fejlmeddeelse, "Venligst fjern vare fra Smart Wagon"2. Hop til UC4: "Foretag fejlhåndtering"3. UC3 afsluttes <p>EXT. 2: Fjernet vægt stemmer ikke med varens registrerede vægt</p> <ol style="list-style-type: none">1. Smart Wagon meddeler en fejlmeddeelse, "Forkert vægt fjernet fra Smart Wagon"2. Hop til UC4: "Foretag fejlhåndtering"3. UC3 afsluttes

Table 3: Fully dressed Use Case beskrivelse for UC3



1.3.4 Use Case 4: Foretag fejlhåndtering

Use Case 4 beskriver hvordan systemet tillader personalet at fortage fejlhåndtering på Smart Wagon i de tilfælde hvor kunden ikke selv vil kunne løse eventuelle udfordringer med indkøbsvognen.

Navn	Foretag fejlhåndtering
Mål	Personale har lavet fejlhåndtering
Initiering	Personale bliver informeret om fejl
Aktør	Primær: Personale Sekundær: Centralsystem
Antal samtidige forekomster	Ingen
Prækondition	Centralsystem er aktivt
Postkondition	Fejl er blevet håndteret
Hovedscenarie	1. Personale undersøger fejl. 2. Personale håndterer fejl. [EXT 1: Personale kan ikke håndtere fejl]
Udvidelser/ undtagelser	EXT. 1: Personale kan ikke håndtere fejl 1. Personalet tager Smart Wagon til videre reparation. 2. UC4 afsluttes.

Table 4: Fully dressed Use Case beskrivelse for UC4

1.4 MoSCoW ift. Funktionelle krav

For at konkretisere systemets krav er der blevet udarbejdet en MoSCoW analyse. For at bedst muligt at kunne beskrive systemet er kravene blevet delt ind i fire underkategorier: *Must have*, *Should have*, *Could have* og *won't have*. De funktionelle krav som beskrives i dette afsnit for den fulde version som var til Smart Wagon.

Must have:

- Smart Wagon skal have en brugergrænseflade i form af en GUI.
- Smart Wagon skal veje et antal varer, der svarer til Smart Wagons vægtpasitet.
- Smart Wagon skal køre frem og tilbage.
- Smart Wagon skal beregne det samlede beløb for varene.

Should have:

- Smart Wagon bør have operationsfunktionalitet i minimum 2 timer.
- Smart Wagons implementerede sprog på GUI bør være på dansk.

Could have:

- Smart Wagon kan dreje.
- Smart Wagon kan selv køre efter en bruger.
- Smart Wagon kan have en smartphone applikation.

Won't have:

- Smart Wagon vil ikke være en drone.
- Smart Wagon vil ikke selv kunne tage varer.
- Smart Wagon vil ikke kunne tage imod betaling.
- Smart Wagon vil ikke kunne tænde og slukke selv.



Arbejdsprocessen af projektet for Smart Wagon har medført at der udarbejdes en betaversion for prototypen af Smart Wagon, grundet Corona situationen, som uddybes nærmere i afsnit 4.1.

1.5 MoSCoW af betaversionen for de funktionelle krav

I dette afsnit beskrives de funktionelle krav i form af en MoSCoW analyse for Smart Wagons betaversion. Denne version af de funktionelle krav er blevet udarbejdet for at konkretisere betaversions systemkrav. For at bedst muligt at kunne beskrive systemet er kravene blevet delt ind i fire underkategorier: *Must have*, *Should have*, *Could have* og *won't have*.

Must have:

- Smart Wagon skal have en brugergrænseflade i form af en GUI (hjemmeside)
- Smart Wagon skal kunne veje 1 vare
- Smart Wagons DC motor skal kunne køre frem og tilbage
- Smart Wagon skal beregne det totale beløb for varene

Should have:

- Smart Wagons brugergrænseflade sprog bør være på dansk

Could have:

- Smart Wagon kan dreje

Won't have:

- Smart Wagon vil ikke køre efter en kunde
- Smart Wagon vil ikke have en smartphone applikation
- Smart Wagon vil ikke have betalingsservice

1.6 Ikke funktionelle krav

Til de ikke funktionelle krav, har vi valgt at lave dem ved hjælp af FURPS-metoden, der står for og bygger på functionality, usability, reliability, performance og supportability. De udarbejdet ikke funktionelle krav beskriver de opstillede krav for den fulde version for Smart Wagon.

Functionality:

- Smart Wagon kører via en DC-motor.
- Smart Wagon registrerer vare via en sensor i form af en vægt.
- Smart Wagon udregner pris baseret på antal registrerede varer.

Usability:

- Kunden kan interagere med en UI på Smart Wagon.
- Kunden kan placere en pose i Smart Wagon.
- Smart Wagon skal kunne betjenes af folk i alle aldre.
- Kunden skal være i stand til at kunne fortryde indkøbet.

Reliability:

- Smart Wagon skal kunne følge efter kunden med en afstand på op til ca 1 meter.
- Smart Wagon skal kunne registrere alle typer af forhindringer indenfor 0.5 meter.
- Smart Wagon skal kunne være operationel i 83,33% af tiden, hvilket er udregnet ud fra MTBF:



$$MTBF = \frac{uptime}{uptime + downtime} \cdot 100\%$$

$$MTBF = \frac{25 \text{ dage}}{25 \text{ dage} + 5 \text{ dage}} \cdot 100\% = 83,33\%$$

Performance:

- Smart Wagon skal kunne være operativ under hele indkøbet (minimum 2 timer inden opladning).
- Smart Wagon skal have en maks hastighed på 6 km/t.
- Smart Wagon skal være stor nok til at kunne holde 54L.
- Smart Wagon skal have en maks højde på 70 cm.
- Smart Wagon skal have en maks vægt på 10 kg.
- Smart Wagon skal kunne holde til en maks vægt på 30 kg.

Supportability:

- Smart Wagon skal kunne genstartes af alle medarbejdere.
- Smart Wagon skal være funktionsdygtig i mindst 1 år, inden den går i stykker (brug 12 timer om dagen, 365 dage om året).

1.7 Betaversionen af de ikke-funktionelle krav

Til udarbejdelse af betaversionen for de ikke-funktionelle krav, har projektgruppen valgt at lave dem ved hjælp af FURPS-metoden, der står for og bygger på functionality, usability, reliability, performance og supportability.

Functionality:

- Smart Wagons kører via en DC motor
- Smart Wagon registrerer vare via en sensor i form af en vægt
- Smart Wagon udregner pris baseret på den registrerede vare

Usability:

- Kunden kan interagere med Smart Wagons GUI
- Kunden kan placere en vare på Smart Wagons vægt

Reliability:

- Smart Wagons moduler skal kunne være operationel 83,33% af tiden, hvilket er udregnet med en MTBF analyse.

Performance:

- Smart Wagons vægt skal kunne holde til en maksimum vægt på 1 kg

Supportability:

- Smart Wagon skal kunne genstartes af alle medarbejdere

1.8 MoSCoW ift. ikke Funktionelle krav

Ud fra vores FURPS-metode inddeling, opdeler vi yderligere disse krav i prioritet ved hjælp af en MoSCoW inddeling som har underkategorierne: *Must have*, *Should have*, *Could have* og *won't have*. Denne opdeling afspejler også de udarbejdet krav for Smart Wagons fulde version.

Must have:

- Smart Wagon skal kunne kører med en DC-motor.



- Smart Wagon skal kunne registrer vare på baggrund af vægt.
- Smart Wagon skal kunne udregne den totale pris af antal varer.

Should have:

- Kunden bør kunne interagere med Smart Wagon via en UI.
- Smart Wagon bør kunne betjenes af folk i alle aldre.
- Smart Wagon skal kunne have plads til en pose.
- Kunden skal være i stand til at kunne fortryde indkøbet.
- Smart Wagon skal kunne genstartes af alle medarbejdere.

Could have:

- Smart Wagon kunne registrere alle typer af forhindringer indenfor 0.5 meter.
- Smart Wagon skal kunne følge efter kunden med en afstand på op til ca 1 meter.
- Smart Wagon skal have en maks højde på 70 cm.
- Smart Wagon skal have en maks vægt på 10 kg.

Won't have:

- Smart Wagon skal have en maks hastighed på 6 km/t.
- Smart Wagon skal kunne holde til en maks vægt på 30 kg.
- Smart Wagon skal være stor nok til at kunne holde 54L.
- Smart Wagon skal være funktionsdygtig i mindst 1 år, inden den går i stykker (brug 12 timer om dagen, 365 dage om året).

1.9 MoSCoW af betaversionen af de ikke-funktionelle krav

Ud fra vores FURPS-metode inddeling, opdeler vi yderligere disse krav i prioritet ved hjælp af en MoSCoW inddeling som har underkategorierne: *Must have*, *Should have*, *Could have* og *won't have*.

Must have:

- Kunden skal kunne interagere med Smart Wagon via en GUI
- Smart Wagons DC motor skal kunne køre
- Smart Wagon skal kunne registrer vare på baggrund af vægt
- Smart Wagon skal kunne udregne den totale pris for antal varer

Should have:

- Smart Wagon bør kunne genstartes af alle medarbejdere

Could have:

- Kunden kan være i stand til at kunne fortryde indkøbet

Won't have:

- Smart Wagon vil ikke have en fysisk indkøbskurv



2 Afgrænsning

I afsnit 2 vil der beskrives, hvilke afgrænsninger der er dannet ift. kravspecifikationen, som skal specificere hvilke områder af Smart Wagon, der delvist ikke medtages i implementationsfasen.

Projektafgrænsningen for Smart Wagon indeholder nogle fast bestemte krav fra ingeniørhøjskolen, Aarhus Universitet, samt vurderinger af realisering af dette projekt.

Ingeniørhøjskolens bestemte krav for 3. semesterprojekt er følgende:

- Projektet skal interagerer med den omkringliggende verden vha. sensorer eller aktuatorer.
- Projektet skal anvende en PSoC
- Projektet skal implementeres med en indlejret Linux platform
- Projektet skal have en brugergrænseflade
- Projektet skal yderligere indeholde faglige elementer fra semesterets fag

Projektgruppen vurderer på baggrund af ingeniørhøjskolens fastlagte krav og bestemmelse af den generelle afgrænsning for Smart Wagon projektet beskrives i de følgende fokuspunkter.

- Projektet interagerer med omverdenen via en sensor *Load Cell: 3-12V DC Digital Electronic Scale 1Kg Weight Weighing Sensor Load Cell* og en aktuator *DC motor*. Aktuatoren er introduceret i GFV labøvelse 2: *DC motor control* og sensoren er indtroduceret i labøvelse 4: *Build a scale*.
- Projektet anvender en PSoC til at processerer og videresende data henholdsvis mellem sensor og Centralstystemet, og mellem aktuator og Centralstystemet.
- Projektet implementeres med en indlejret Linux platform i form af Ubuntu systemet, som er introduceret i ISU og HAL undervisningen.
- Projektet har en brugergrænseflade som implementeres på Raspberry Pi's hjemmeside på Ubuntu platformen, som også anvender WebSocket for at opretholde kommunikation mellem source koden for GUI og HTML implementeringen. Anvendelsen af WebSocket er baseret på extern information givet i forbindelse med projekt workshop. Heraf skal det pointeres at der ikke har været nogen form for undervisning af WebSocket på 3. Semester, så denne del er selektiv læring.
- Projektet indeholder faglige elementer fra fagene GFV, ISU og HAL.

3 Teknologianalyse

I dette afsnit vil der redegøres for det teknologiske valg projektgruppen har fortaget i forbindelse med udviklingen af Smart Wagon. I analysen vil de teknologiske overvejelser og valg blive fremlagt.

3.1 Valg af afstandssensor til Smart Wagons kørefunktion

Der blev tidligt i forløbet ønsket, at Smart Wagon selv skulle kunne manøvrere sig rundt. Derfor var der brug for en afstandssensor, for at den ikke skulle bevæge sig ind i ting. Med dette ønske blev der kigget på 3 forskellige type sensorer, og valget endte med at falde på "VL53L0X Time of Flight Distance Sensor". Denne sensor ville ved brug af laser være i stand til at måle afstanden fra Smart Wagon til et givent objekt af interesse, som i dette tilfælde er kunden. Efter dette valg blev der meddelt, at der ikke kunne hentes hardware fra Embedded Stock, hvilket gjorde, at der ikke blev brug for en afstandssensor i betaversionen for Smart Wagon.

Billeder af de forskellige sensorer kan ses herunder:



Figure 3: VL53L0X Time of Flight Distance Sensor



Figure 4: MB1202 I2CXL-MaxSonar-EZ0



Figure 5: Infrared Proximity Sensor - Short Range - 2Y0A41SK

Der kom hurtig en enighed om, at "Infrared Proximity sensor", der kan måle en afstand på 4-30 cm ikke egnede sig til dette projekt, da der skulle bruges en sensor, der kunne måle mindst fra 50 cm til i hvert fald 100 cm, så denne sensor ville ikke opfylde kravet. Derudover blev det kigget på "Maxsonar"-sensoren. Denne sensor kunne måle en afstand fra 25 cm til 765 cm, hvilket ville være en oplagt mulighed for at opfylde kravet for afstandssensoren. Denne sensor havde dog et problem, når den skulle detektere kunden, da der kunne ende med at komme både bevægelser, men også statiske objekter ind i dens måleområde, hvilket den så skulle forholde sig til, og dette ville være svært at tage højde for. Til sidst blev der kigget på



”Time of flight”-sensoren, der kunne måle en afstand fra 30 mm - 1000 mm. Denne ville være en brugbar sensor i dette tilfælde, da den både kan få målt fra 50 cm til 100 cm, som er kravet for afstandssensoren i dette projekt. Med dens laser vil den kunne give signal om, Smart Wagon er for tæt på objekter eller for langt fra kunden.

3.2 Valg af Centralssystem

I udvikling af Smart Wagon har der været brug for et system som både kan sende, modtage og processere data. Derfor har projektgruppen valgt at anvende Raspberry Pi som Centralssystem, fordi den har mulighed for at oprette diverse forbindelser, heraf en seriell forbindelse som f.eks. er blevet anvendt i udviklingen af Smart Wagons vægt og motor. Raspberry Pi er også i stand til at forholde den data som kommer ind fra Smart Wagons forskellige delssystemer, såsom f.eks. den data der kommer fra vægten eller den data der skal sendes til motoren, for at aktivere den. Yderligere har RPi'en en mulighed for at opbevare data.

3.2.1 Level converter

Da der ikke var mulighed for, grundet Corona, at ændre på hardware processoren (PSoC'en) for at kunne lave en konversion mellem 5V til 3.3V. Ændringen er påkrævet, da PSoC'en som standard vil videregive 5V, mens RPi'en maksimum håndtere 3.3V. Ved hjælp af den givende Level Converter, er det muligt at lave denne konvertering. Det vil altså sige, at der ved hjælp af denne, sendes der 5V ind fra den ene side, og 3.3V i den anden. Herved kan der via 4 forskellige channels, kunne konvertere de spændinger der transimitteres. Dette bruges da de omtalte Rx og Tx bruger 5V på PSoC og 3.3V på RPi'en. For at kommunikationen altså kan forgå, kræves der at de rigtige spændinger måles af de to hardware systemer.

3.3 Forbindelse imellem dataprocesseren og Centralssystemet

Projektgruppen gjorde sig en del overvejelser i forbindelse med hvordan kommunikationen fra data processeren (PSoC) til Centralssystemet (RPi).

Der var primært to løsninger som gruppen overvejede:

- SPI driver
- Serielforbindelse

Ved samtal med en Lektor, hvor der blev udspurgt om forbindelse mellem PSoC og RPi. Det blev også hurtigt gjort klart for gruppen, på baggrund af anbefaling fra Lektor, at en seriell forbindelse ville give en simplicerer implementeringsløsning. Fordelen ved den serielle forbindelse er simpliciteten, da RPi'en allerede har et integreret system, til at kommunikere via denne forbindelse, hvilket vil sige, at der ikke skulle skrives en SPI driver, der ellers var til overvejelse og blev forsøgt implementeret, dog uden held for funktionalitet. Den serielle forbindelse fungere således, at der ved et UART modul på PSoC, kan sendes værdier videre mellem de to hardware enheder. Der oprettes en interrupt rutine på PSoC'en, der gør det muligt at registrerer input udefra, ved hjælp af denne omtalte UART. Kort sagt kan der nemt og bekvæmt sendes data via en Rx og Tx værdi, der kommunikere begge veje mellem PSoC og RPi.

3.4 Valg af CPU til hardware delssystemer

Smart Wagon bygger på hardware moduler, såsom en sensor, i form af vægt, og en aktuator, i form af en DC-motor. Disse systemer kan RPi'en, altså Centralssystemet, ikke kommunikere direkte med. Der er altså brug for en processor, der skal være i stand til at registrere og behandle data i forbindelse med de hardware moduler der findes i projektet. PSoC'en er her i stand til at behandle den data og videresende den til RPi'en.

3.4.1 Vægt

For at Smart Wagon skal kunne registrere hvilken vægt der tilføjes til indkøbet, er der gjort brug af en sensor, i form af en vægt. Her bruges en vægt, der er forbundet til en PSoC. Ud fra en forbindelse mellem



PSoC og vægten, behandles data fra vægten, der kommer ind som en spænding. Dette data omskriver PSoC'en til gram, som bruges til at bestemmer hvilket produkt der er registreret.

Her opstår dog et problem, da vægten ikke kan identificere flere produkter på en gang. Hvis vægten mäter en spænding, der svarer til 400 gram, hvor der opstår 2 produkter af 200 gram. Dette burde registreres som 2 æbler, men registreres som en pakke smør. Vægten har desuden et off-set, på et interval mellem 800-810 mV. Dette betyder at der korregeres, således at der vil mæles 0 gram ved 800 mV.

Yderligere er der bestemt at målingen af gram bestemmes, således at der ved en stigning på 1 mV, vil der ske en stigning på 0.8 gram. Vægten er valgt til betaversionen, da der allerede skulle udvikles en i faget GFV på tredje semester.

3.4.2 DC-motor

For at Smart Wagon skulle kunne være i stand til at bevæge sig rundt, og i fremtiden skulle kunne køre efter kunden, har der være brug for en motor. Denne motor skal kunne aktiveres på kommando, og heraf bruges en forbindelse mellem Centralsystem og motor, via PSoC, der processere den data som motoren bruger. Det er altså muligt at sende information, her et 1 eller 0, fra RPi til PSoC, som heraf kan få motoren til at køre.

Motoren kører via et PWM-signal, således ved et signal på enten 1 eller 0, vil PWM aktiveres, via pins. Disse pins bestemmer om det er muligt at sende data fra PSoC til L298 H-bro. H-broen aktiverer motoren, afhængigt af hvilken vej spændingen sendes igennem H-broen, og derved får den til at køre.

I projektet er der blevet brugt en DC-motor, hvor valget ellers kunne have været en stepper-motor. Grunden til at der er blevet valgt en DC-motor, er at der i betaversionen af projektet kun har skulle være funktion at kunne køre frem og stoppe igen. Her vurderes at DC-motoren er den mest ideelle at bruge da DC-motor frem for en stepper-motor oftest bruges til styrring af hjul, så som en elektrisk bil, da stepper-motoren er bedre præcisions arbejde, når man fx skal have motoren til at stoppe ved en bestemt vinkel. Ved videreudvikling af systemet ville det, via PWM, være muligt at ændre hastigheden på Smart Wagon.

3.5 Valg af brugergrænseflade

Der har været en række designovervejelser i forbindelse med udviklingen af Smart Wagons brugergrænseflade. Formålet med brugergrænsefladen, er at kunden nemt og mobil skal kunne tilgå den indkøbsliste som vil blive vist der på.

Projektgruppen havde primært følgende overvejelser i forbindelse med hvordan brugergrænseflade skulle være tilgængelig for kunden.

- En smart phone applikation
- En hjemmeside
- En touchskærm monteret på Smart Wagon

Det er i projektgruppen blevet besluttet at implementere brugergrænsefladen på en hjemmeside ved brug af HTML, JavaScript og Cascading Style Sheets (CSS). Dette valg skyldes tankegangen om den agile udviklingsprocess, som bliver illustreret i figur ??, hvor der er fokus på at en process gennemgår forskellige udviklingsstadier før den endelige version står færdig.

Gruppen ser implementeringen af brugergrænseflade på en hjemmeside som det første stadie i udviklingsprocessen, hvor implementering på en applikation eller en touchskærm ville være et mål for et færdigudviklet slutprodukt for Smart Wagon.

3.5.1 Valg af Raspberry Pi som host for hjemmesiden

Som en naturlig forlængelse i valget om at implementere brugergrænsefladen på en hjemmeside er det blevet besluttet at anvendes Raspberry Pi'en som en host for Smart Wagon hjemmeside. Dette valg skyldes at Raspberry Pi allerede har en integreret webhostnings funktion hvor der kan implementeres en hjemmeside.



4 Risikoanalyse

I dette afsnit bliver der udarbejdet et risikoanalyse, hvor formålet er at afdække de forskellige mulige risici som kunne indtræffe i afviklingen af projektet.

For at udarbejde risikoanalysen er ISE kompendiumet ”I2ISE Compendium, Revision 2, June 2012” blevet anvendt. Heraf er der taget udgangspunkt i afsnit 4 ”Retningslinjer for projektledelse”, side 277-279.

Der er blevet udarbejdet en minimumsmodel hvor de forskellige foranstaltninger vil blive vurderet ud fra sandsynlighed, konsekvens og en samlet vurdering af produktet af de to parametre.

I minimumsmodellen vurderes sandsynlighed og konsekvens ved foranstaltningerne ud fra 3 scenarier. Dette bliver uddybet i nedenstående tabel.

Sandsynlighed 'S'	1 = det vil sandsynligvis ikke ske, men det er dog muligt 2 = det er lige så sandsynligt, at det sker som at det ikke sker 3 = det vil sandsynligvis ske
Konsekvens 'K'	1 = Konsekvensen kan løses med planens nuværende aktiviteter 3 = Der skal gøres noget for at afbøde konsekvensen 10 = Planen skrider hvis dette sker
Produkt af sandsynlighed og konsekvens 'P'	$S \cdot K$

Table 5: Beskrivelse af sandsynlighed, konsekvens og produktet af sandsynlighed og konsekvens

I den følgende tabel udtrykkes *risikomodelen* for Smart Wagon

Risici	S	K	P	Mulige foranstaltninger
1. Ændringer i kravspecifikationen i sene faser	3	3	9	Revurdere behovet for ændringerne, samt en tilrettelæggelse af projektet
2. Opfyld krav ved accepttest	3	3	9	Revurdering af kravspecifikationen, udfra en beta prototypemodel
3. Implementationsmangler i subdele	1	3	3	Fejlfinde og forklare manglerne
4. Funktionel samlet prototype for det planlagte system	2	10	20	Implementering og test af subdele af 'beta prototypen' for Smart Wagon
5. Samarbejdsudfordringer i projektgruppen	2	3	6	Åben og produktiv dialog, samt en demokratisk proces
6. Tidsbegrænsning ift. projektudarbejdelsen	3	3	9	Ugentlig scrum møder og forebyggende revurderinger
7. Manglende forbindelse mellem PSoC og RPi	2	10	20	Individuel test af PSoC og RPi
8. Defekt vægt	1	10	10	Forsøge at fejlfinde og forklare manglerne
9. Defekt DC motor	1	10	10	Forsøge at fejlfinde og forklare manglerne
10. Beskadiget hardware	2	3	6	Udskifte de beskadiget hardware dele
11. Kommunikationsfejl mellem RPi og GUI	1	1	1	Debug af kode og-eller erstatter den pågældende RPi
12. Den fulde funktionalitet af GUI	2	1	2	Debug og videreudvikle kodeimplementering

Table 6: Risikomodel for Smart Wagon

De vurderinger med høj risiko i overstående tabel, er alle punkter, hvor et fuldt fungerende system er essentielt. Heraf vil konsekvensen være at der ikke er et fungerende slutprodukt. Dette vil påvirke en evt. accepttest, idet at de essentielle delsystemer ikke vil kunne udføre deres opgave.



4.1 Projektplan

I risiko analysen er projektplanen blevet revurderet på baggrund af den igangværende Covid-19 situation.

Det er i den forbindelse blevet besluttet at slutresultatet for Smart Wagon blevet nedskaleret. Da det er blevet oplyst af Embedded Stock at det på dette semester ikke vil være muligt at få udleveret HW komponenter, det vil sige at det ikke er muligt for gruppen af få udleveret den bil med larveben som ellers var planlagt.

Det er derfor blevet besluttet af projektgruppen at der ikke udvikles en fuld funktionel prototype, men at der i stedet udarbejdes en betaversion af prototypen. Dette betyder at der ikke længere stræbes efter, at de forskellige HW blokke vil kunne fungere i samspil med hinanden. Det er dog blevet muligt at lave et sammenspil mellem de systemer der er påkrævet af Beta versionen, ved en integrationstest. Beslutningen er taget for at muliggøre krav og realisere accepttesten af betaversionen for projektet.

Bortset fra ovenstående revurdering af projektplan forventes det at projektet forløber som planlagt.

5 Systemarkitektur

I dette afsnit vil Smart Wagons systemarkitektur blive redegjort i form af et BDD-diagram, et IBD-diagram, en domænemodel samt sekvensdiagrammer for hver Use Case. Disse redskaber bruges til at specificere og præcisere Smart Wagons hardware og software interface.

5.1 Block Definition Diagram

Block-diagrammet over Smart Wagon i figur 6 er udarbejdet for at gøre det muligt at identificere, hvilke blokke systemet består af. I diagrammet fremgår det, at øverst i hierarkiet ligger systemets tre overordnede delsystemer som hver er udstyret med en CPU: en *PSoC*, en *Computer* og en *Raspberry Pi*. Under *PSoC'en* ligger de to blokke som repræsenterer Smart Wagons to sensorer, der er henholdsvis en distancesensor og en vægt.

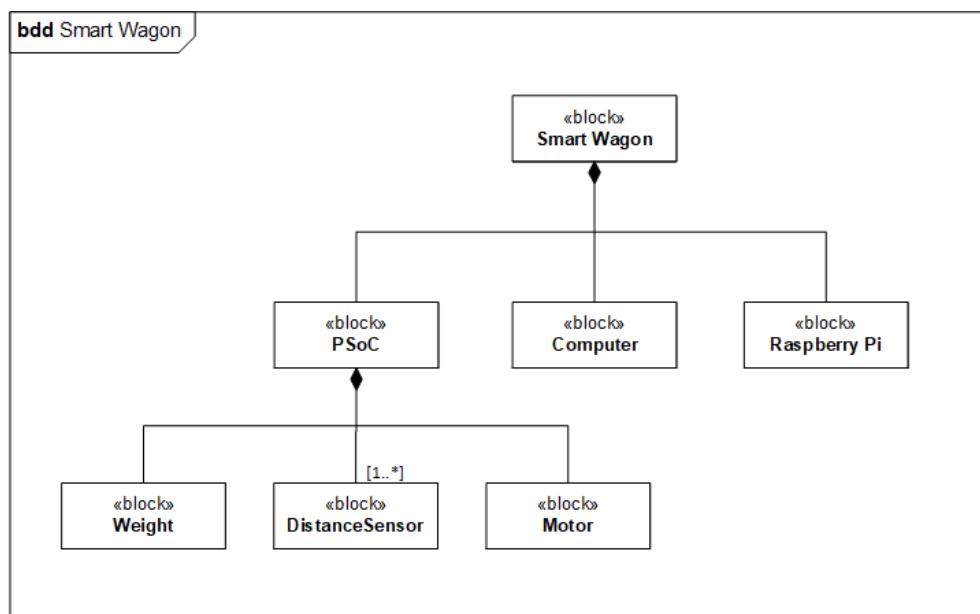


Figure 6: BDD for Smart Wagon

5.2 Internal Block Diagram

Ud fra BDD'et i figur 6 er der udarbejdet et IBD over Smart Wagon, hvilket ses i figur 7. Dette IBD indeholder de interne forbindelser i Smart Wagon, uden yderligere specificering af forbindelserne, samt de interne forbindelser for IBD'et af PSoC blokken.

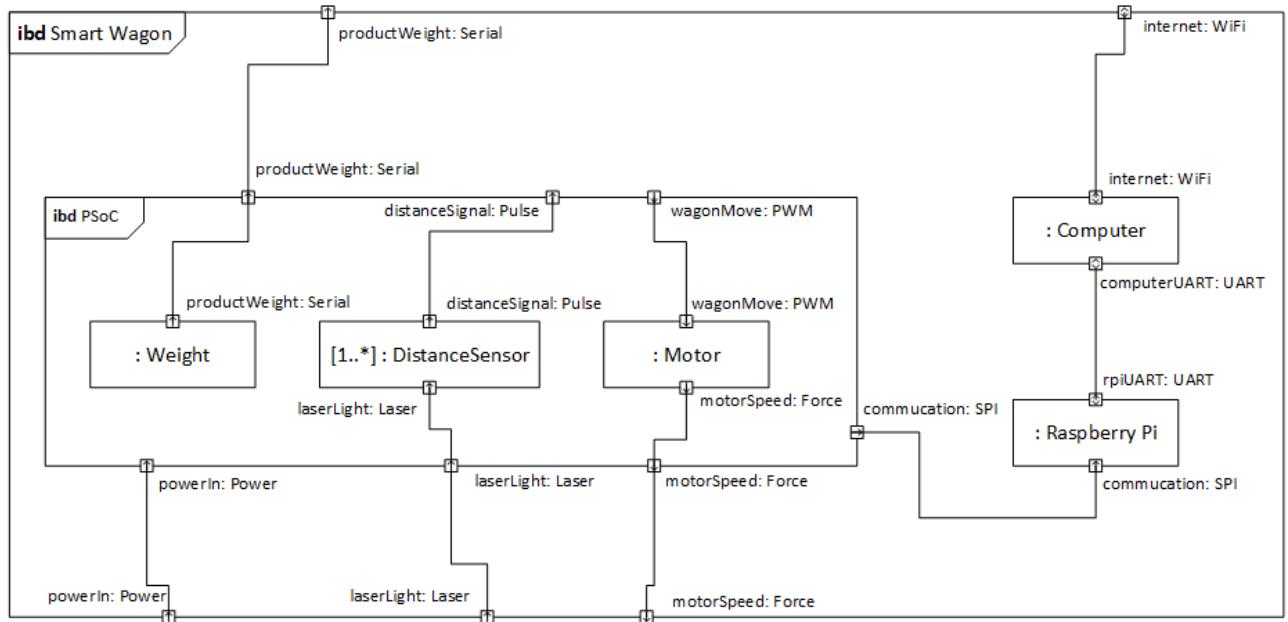


Figure 7: IBD for Smart Wagon

5.3 Deployment view

Ud fra IDB'et i figur 7 er der udarbejdet et deployment view, som kan ses i figur 8 som både viser hvorpå der er software i delsystemerne og de interne blokforbindelser.

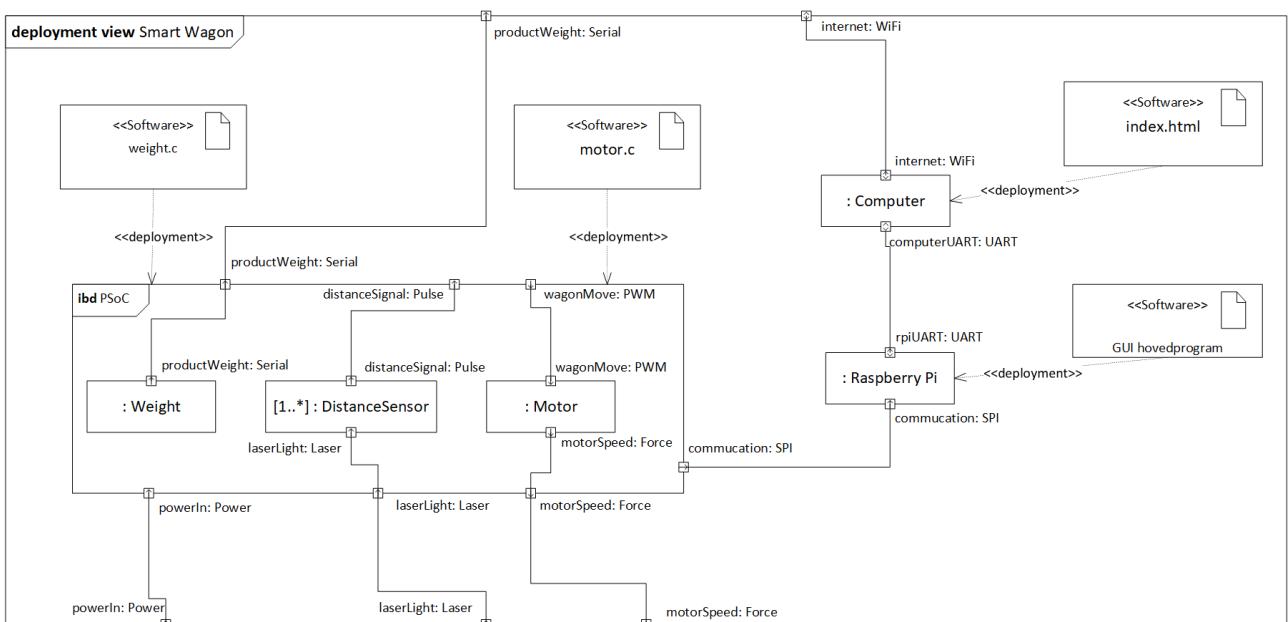


Figure 8: Deployment view af Smart Wagon

5.4 Domænemodel

For at få udarbejdet domænemodellen i figur 9 er der først lavet en system-domæneanalyse. Formålet med analysen er at afgrænse Smart Wagons vigtigste begreber og relationer. Domænemodellen beskriver relationerne imellem Smart Wagon og omverdenen, samt afbilder systemets begreber.

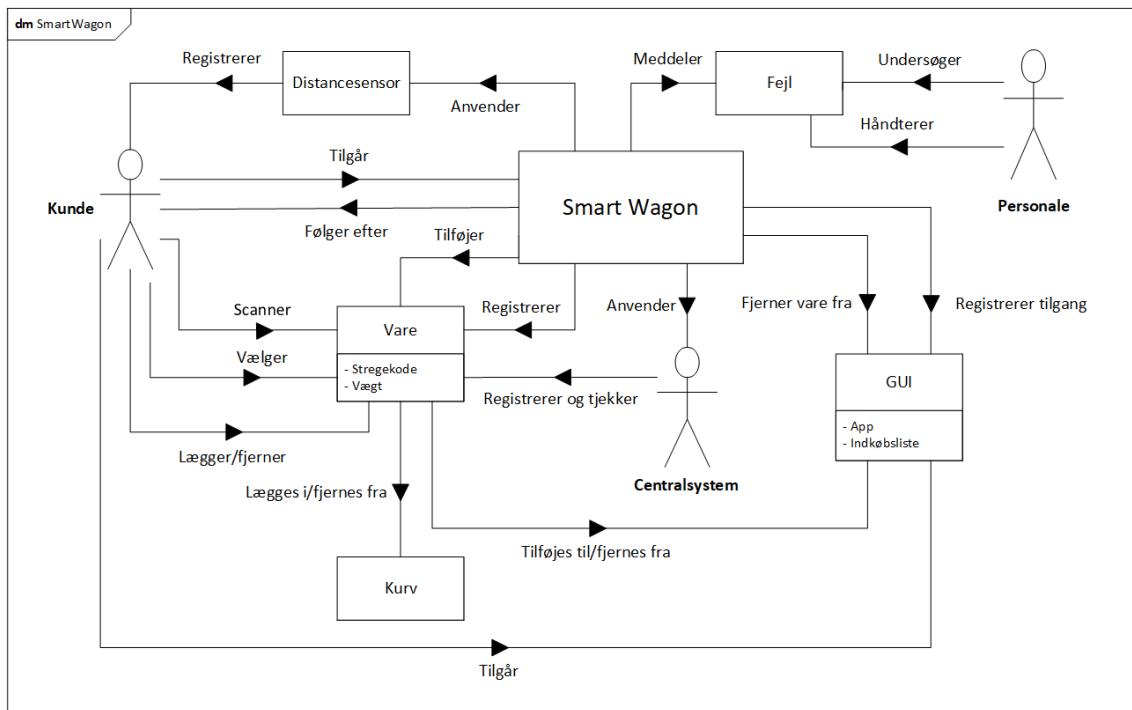


Figure 9: Domænemodel for Smart Wagon

5.5 Sekvensdiagrammer for handling

I efterfølgende sekvensdiagrammer beskrives Smart Wagons adfærd i form af handlinger ud fra de fire udformede Use Case diagrammer som er blevet præsenteret i afsnit 1, Kravspecifikation.

5.5.1 Sekvensdiagram for UC1: Kør efter kunde

I figur 10 kan det udarbejdede sekvensdiagram for Use Case 1 ses. Det beskrives i scenariet, hvordan kunden interagerer med Smart Wagon når indkøbsvognen skal køre efter kunden.

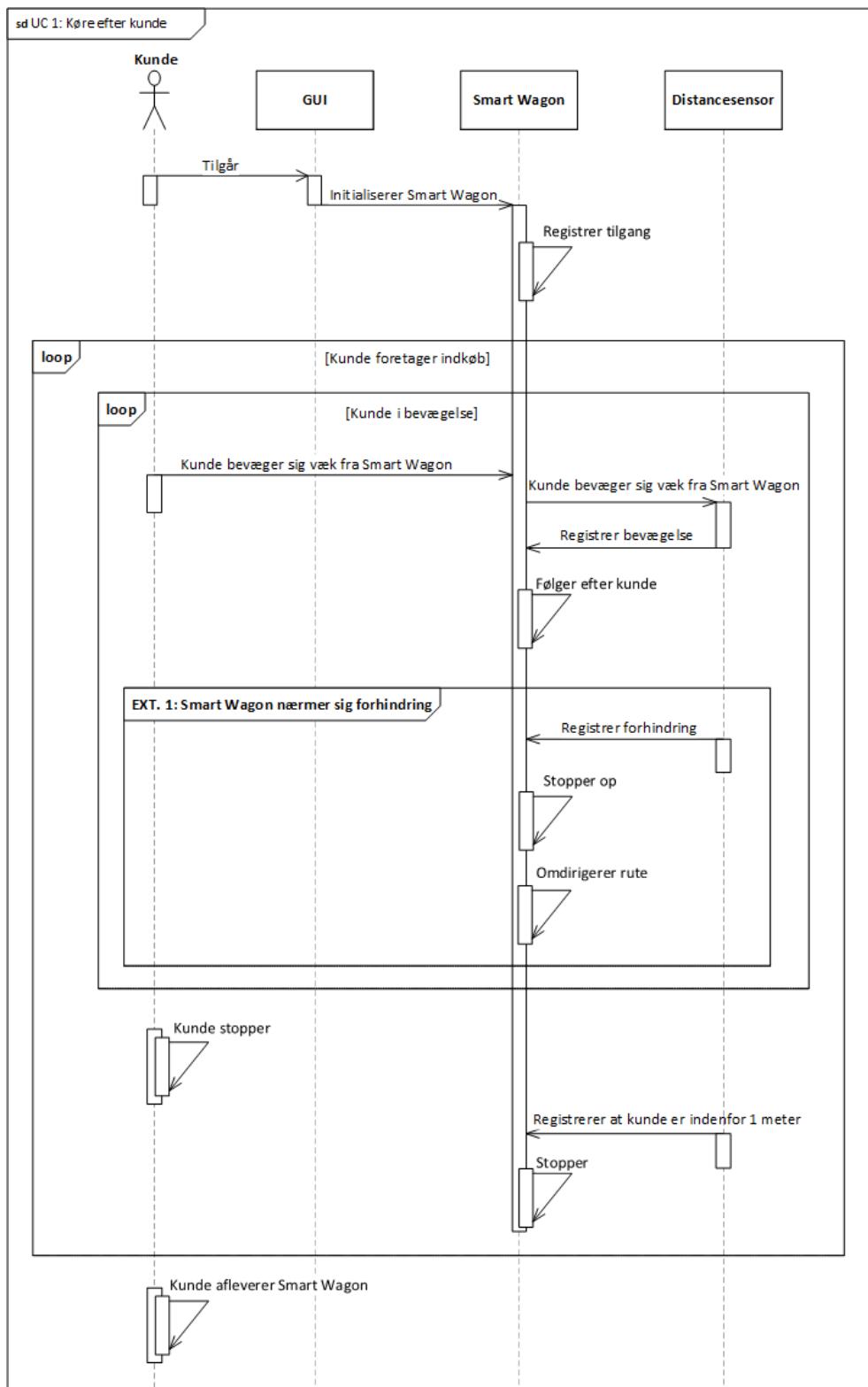


Figure 10: Sekvensdiagram for UC1: Køre efter kunde

5.5.2 Sekvensdiagram for UC2: Registrer vare

I figur 11 kan det udarbejdede sekvensdiagram for Use Case 2 ses. Det beskrives i scenariet, hvordan kunden interagerer med Smart Wagon når der skal registreres en vare.

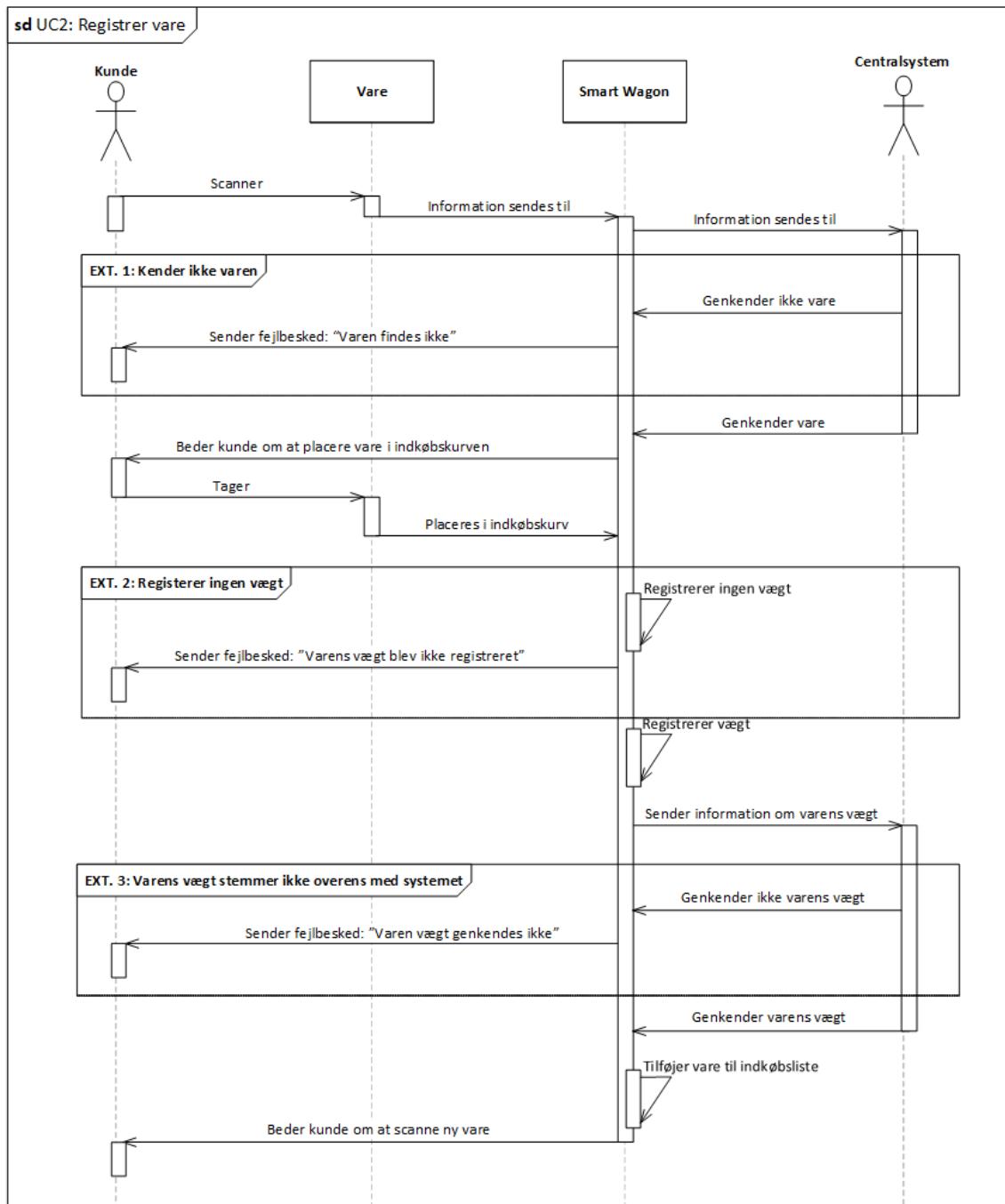


Figure 11: Sekvensdiagram for UC2: Registrer vare

5.5.3 Sekvensdiagram for UC3: Fjern vare

I figur 12 kan det udarbejdede sekvensdiagram for Use Case 3 ses. Det beskrives i scenariet, hvordan kunden interagerer med Smart Wagon når der skal fjernes en vare fra indkøbslisten.

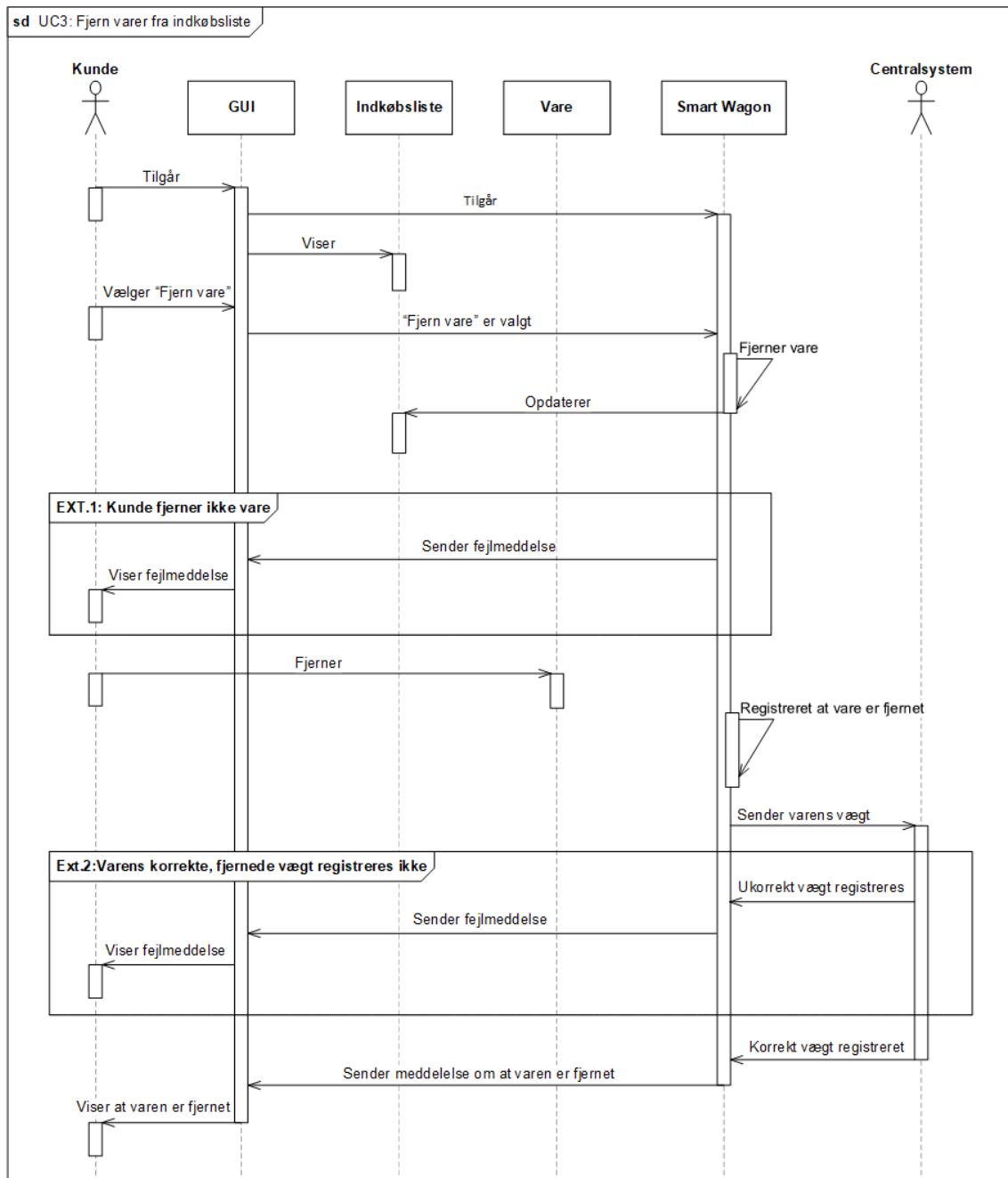


Figure 12: Sekvensdiagram for UC3: Fjern vare

5.5.4 Sekvensdiagram for UC4: Foretag fejlhåndtering

I figur 13 kan det udarbejdede sekvensdiagram for Use Case 4 ses. Det beskrives i scenariet, hvordan kunden interagerer med Smart Wagon når der skal foretages en fejlhåndtering på indkøbsvognen.

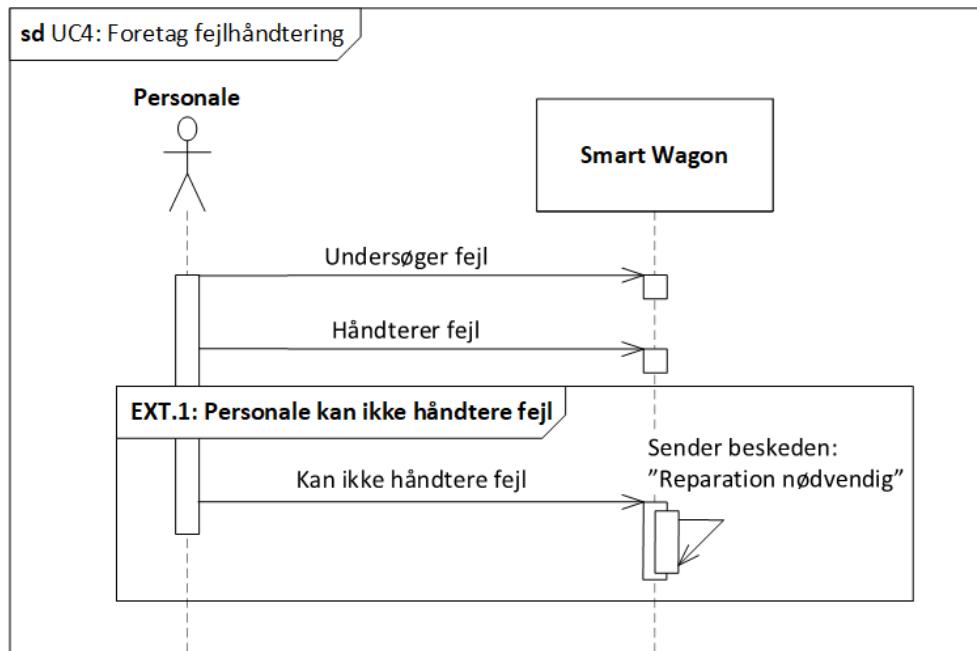


Figure 13: Sekvensdiagram for UC4: Foretag fejlhåndtering

6 Design

I design afsnittet vil Smart Wagons software og hardware design blive præsenteret. Software designet vil blive beskrevet i form af en applikationsmodel for Smart Wagons betaversion. Hardwarens design vil beskrives ud fra en samlet opstilling for Smart Wagons motor og vægt.

6.1 Hardware design

I dette afsnit vil Smart Wagons hardware design blive præsenteret. Dette vil gøres ved brug at en teoretisk opstilling og en fysisk opstilling af motoren og vægten, samt at den serielforbindelse som sender data i mellem hardware blokkene PSoC og Raspberry Pi, vil blive beskrevet.

Der er i forbindelse med udarbejdelsen af Smart Wagon blevet implementeret en fælles hardware opstilling. Der er derfor blevet lavet en oversigt over de forbindelser der er i mellem de forskellige hardware blokke som kan ses i figur 14. Figurens formål er at give en overskueligt overblik over de forskellige forbindelser så når foretages forskellige test vil gøre det nemmere for gruppensmedlemmer at debugge systemet hvis fejl skulle opstå.

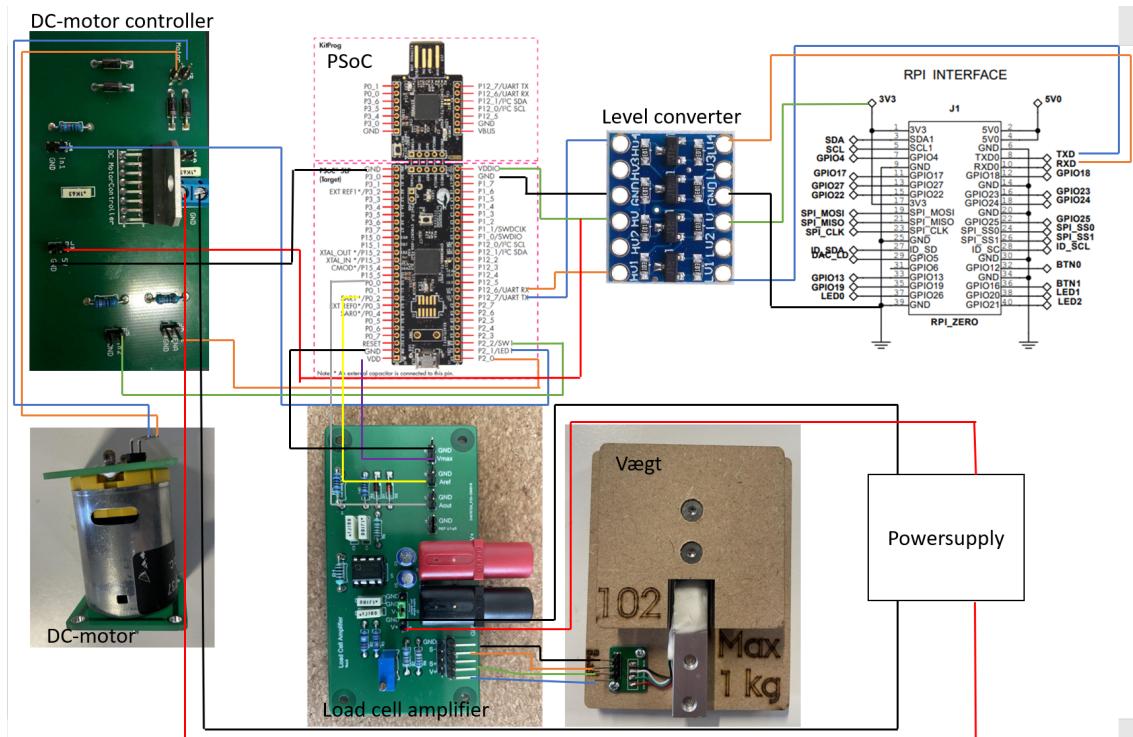


Figure 14: Hardware pin oversigt

I tabel 7 kan en oversigt over forbindelserne i mellem hardware blokken ses. Denne tabel giver et præcist overblik over hvilke forbindelser der skal laves for at få et operativt system for Smart Wagons betaversion



Hardware blok 1	Signal 1	Signal 2	Hardware blok 2
DC-motor controller	Motor +/-	DC-motor	DC-Motor
DC-motor controller	J4/In1	P2_1	PSoC
DC-motor controller	J3/5 V	HV	Level converter
DC motor controller	J5/In2	P2_2	PSoC
DC-motor controller	J6/ENA	P2_0	PSoC
DC-motor controller	GND	GND	PSoC
DC-motor controller	+/-	+/-	Powersupply
Load Cell Amplifier	GND	GND	PSoC
Load Cell Amplifier	V_max	VDD	PSoC
Load Cell Amplifier	A_ref	P0_2	PSoC
Load Cell Amplifier	A_out	P0_0	PSoC
Load Cell Amplifier	V +	+	Powersupply
Load Cell Amplifier	GND	-	Powersupply
Load Cell Amplifier	GND	GND	Vægt
Load Cell Amplifier	S-	S-	Vægt
Load Cell Amplifier	S+	S+	Vægt
Load Cell Amplifier	V+	V+	Vægt
Level Converter	HV1	P12_6	PSoC
Level Converter	GND	GND	PSoC
Level Converter	HV4	P12_7	PSoC
Level Converter	LV1	TXD	RPi
Level Converter	GND	GND	RPi
Level Converter	LV	3V3	RPi
Level Converter	LV4	RXD	RPi

Table 7: Forbindelser i mellem hardware blokkene for Smart Wagon

Billedet i figur 15 viser den fysiske opstilling for motoren og vægten. Her kan det blandt andet ses at der bliver oprettet forbindelse imellem PSoC og RPi via en level converter som er opstillet på et fumlebræt. Det har være muligt at kombinere Smart Wagons to primære hardware blokke, nemlig vægten og motoren, samt at sende data i mellem PSoC og RPi ved brug af en seriel forbindelse.

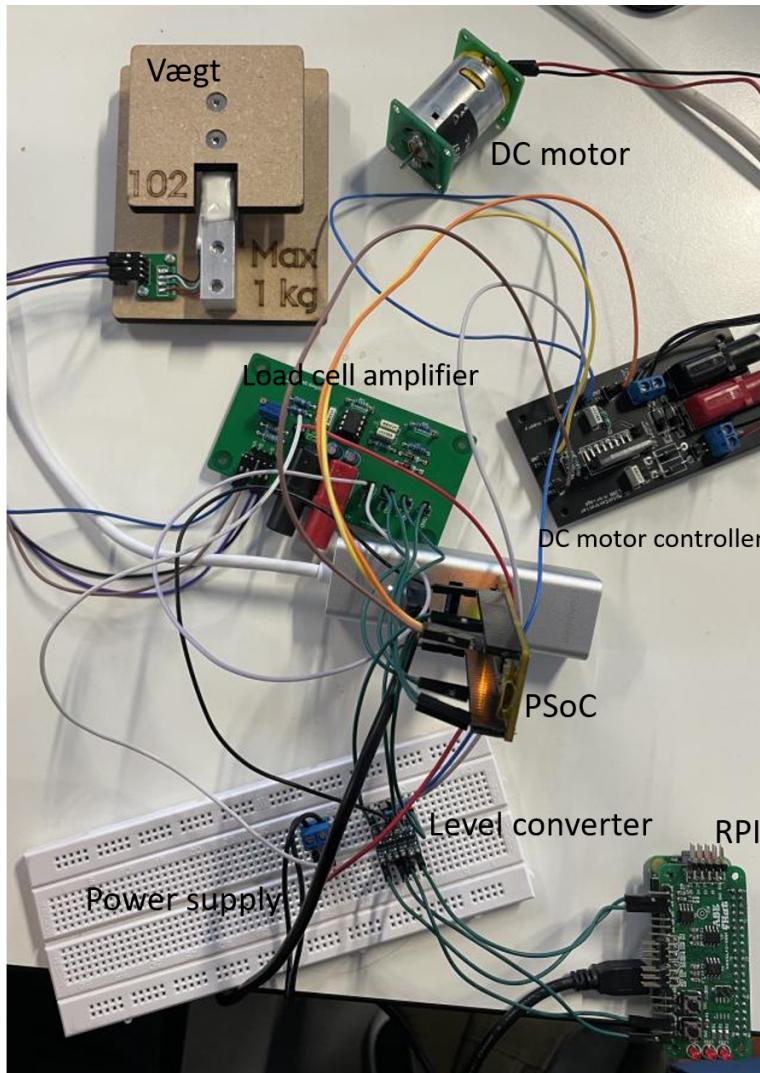


Figure 15: Fysisk opstilling af hardware

6.1.1 Design af vægten

Ved design af vægten, er der blevet gjort brug af en "Load Cell Amplifier", som modtager signaler fra vægten, og videresender det til PSoC'en, så den er i stand til at behandle data. Vægten er designet således, at hvis en kunde placerer en genstand på vægten, vil vægten måle en ny spænding, baseret på den strain gauge, det "stræk", som vægten påvirkes af. Denne påvirkning af strain gauge, giver stærkere spænding, hvilket videresendes gennem "Load Cell Amplifier" printet. Det vil sige at designet består af en sensor, en "Load Cell Amplifier" og en PSoC.

6.1.2 Design af motor

Ved design af motoren, er der blevet gjort brug af en DC-motor, "DC-motor controller" og en PSoC. Motoren er designet således at den styres igennem et PWM signal, som driver motoren og giver mulighed for at kontrollere omdrejningshastigheden, samt er i stand til at styre retningen på motoren. PSoC'en skaber dette PWM signal som sendes ud på to forskellige pins. Disse pins bestemmer hvilken retningen strømmen løber igennem H-broen.



6.1.3 Serielforbindelse mellem PSoC og RPi

I udviklingen af hardware designet er det blevet besluttet at der skal bruges en serielforbindelse til at sende data frem og tilbage i mellem PSoC og RPi.

Der bliver i forbindelse med den serielleforbindelse anvendt en level converter som sikre sig at der kun bliver sendt 3.3 V til RPi'en for ikke at brænde enheden af. Der kan læses mere om level converteren i afsnittet der omhandler teknologianalySEN.

De to hardware blokke bliver forbundet ved at krydsforbinde i mellem PSoC'ens Tx-pin¹ med RPi'ens Rx-pin² for at sende den indkommende data fra vægten igennem PSoC og videre til RPi'ens.

For at der kan sendes data fra RPi'ens videre til PSoC'ens og tilsidst den til DC-motoren krydsforbindes PSoC'ens Rx-pin med RPi'ens Tx-pin.

6.2 Komponentliste

I nedenstående tabel kan komponentlisten for Smart Wagon. Komponentlisten giver et overblik over hvilke hardware komponenter der skal anvendes for at teste og bygge betaversionen for Smart Wagon.

Navn	Rolle	Antal
PSoC	Data processer	1
Raspberry Pi Zero W	Centralsystem	1
Load cell amplifier (AMP PCB)	Vægt	1
DC motor controller (L298 PCB)	Motor	1
DC-motor (238-9737)	Motor	1
Load cell (max 1 kg)	Vægt	1
Level converter 5V - 3.3 V (4 channel, bidirectional)	5V - 3.3V converter	1
Power Supply (PSU 9 V)	System spænning	1

Table 8: Komponentliste for Smart Wagon

¹Transmitter

²Receiver

6.3 Software design

I afsnittet for Sofware design, er der blevet lagt fokus på, at give visuelle fremstillinger af Smart Wagons funktioner. Dette gøres ud fra applikationsmodeller.

6.3.1 Applikationsmodel baseret på beta modellen af SmartWagon

I dette afsnit vil den udarbejdet applikationsmodel for Smart Wagons betaversion blive præsenteret. Det gøres i form at udarbejdet klassediagrammer og sekvensdiagrammer ud fra de forskellige Use case diagrammer.

Klassediagram for UC1: Kør efter kunde

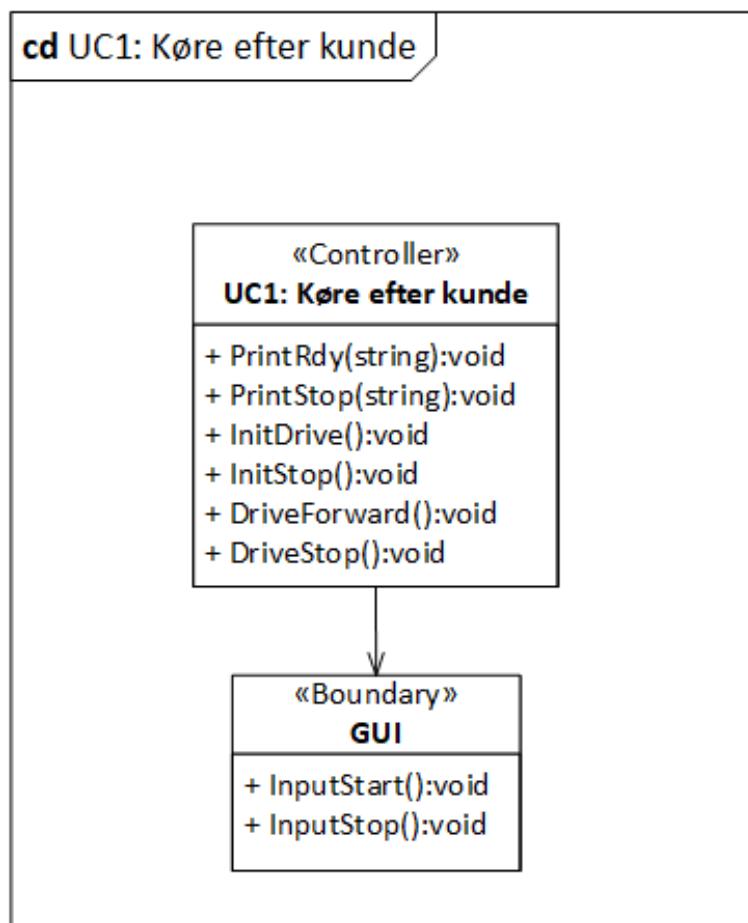


Figure 16: Klassediagram for UC1: Kør efter kunde

Sekvensdiagram med metoder for UC1: Kør efter kunde

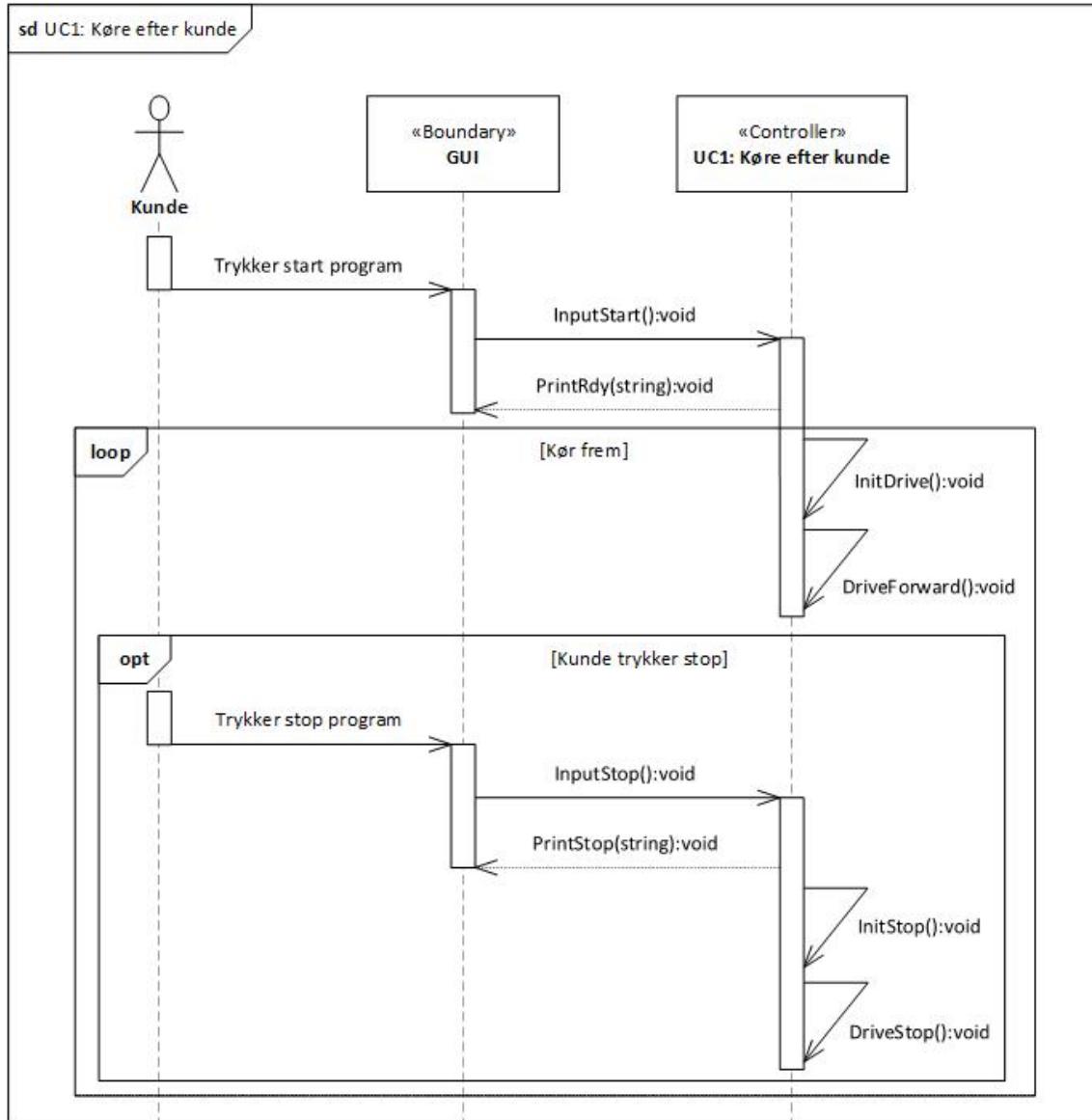


Figure 17: Sekvensdiagram med metoder for UC1: Kør efter kunde

Klassediagram for UC2: Registrer vare

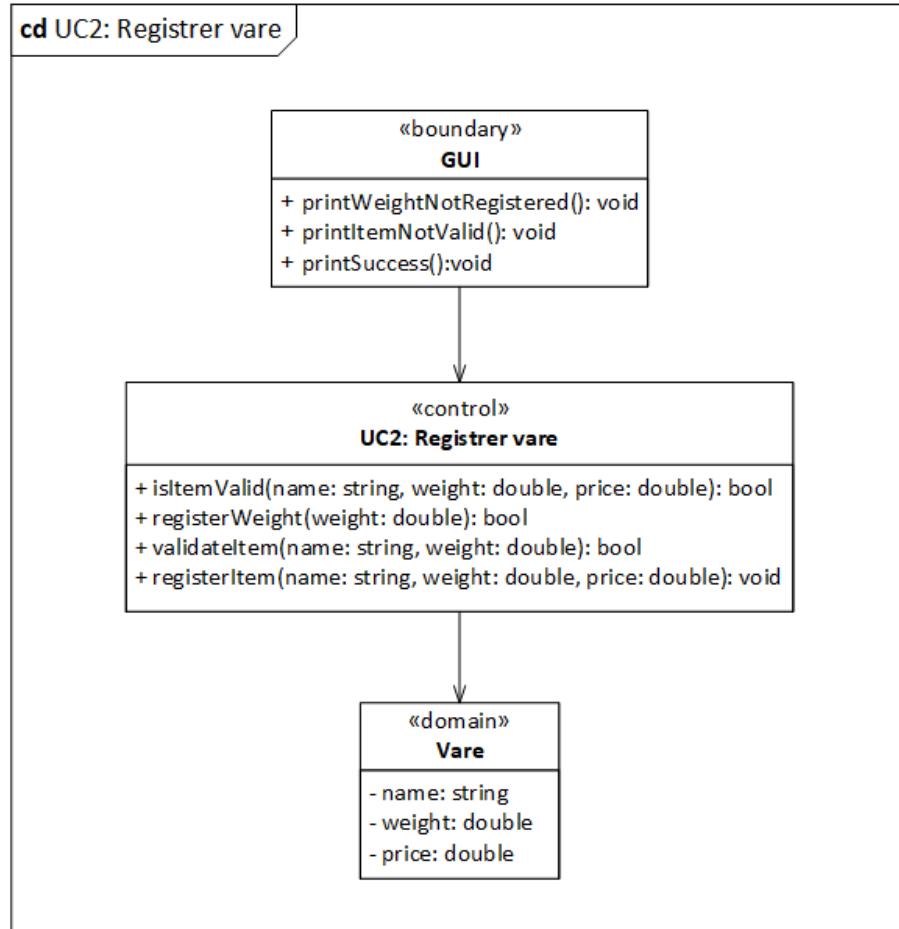


Figure 18: Klassediagram for UC2: Registrer vare

Sekvensdiagram med metoder for UC2: Registrer vare

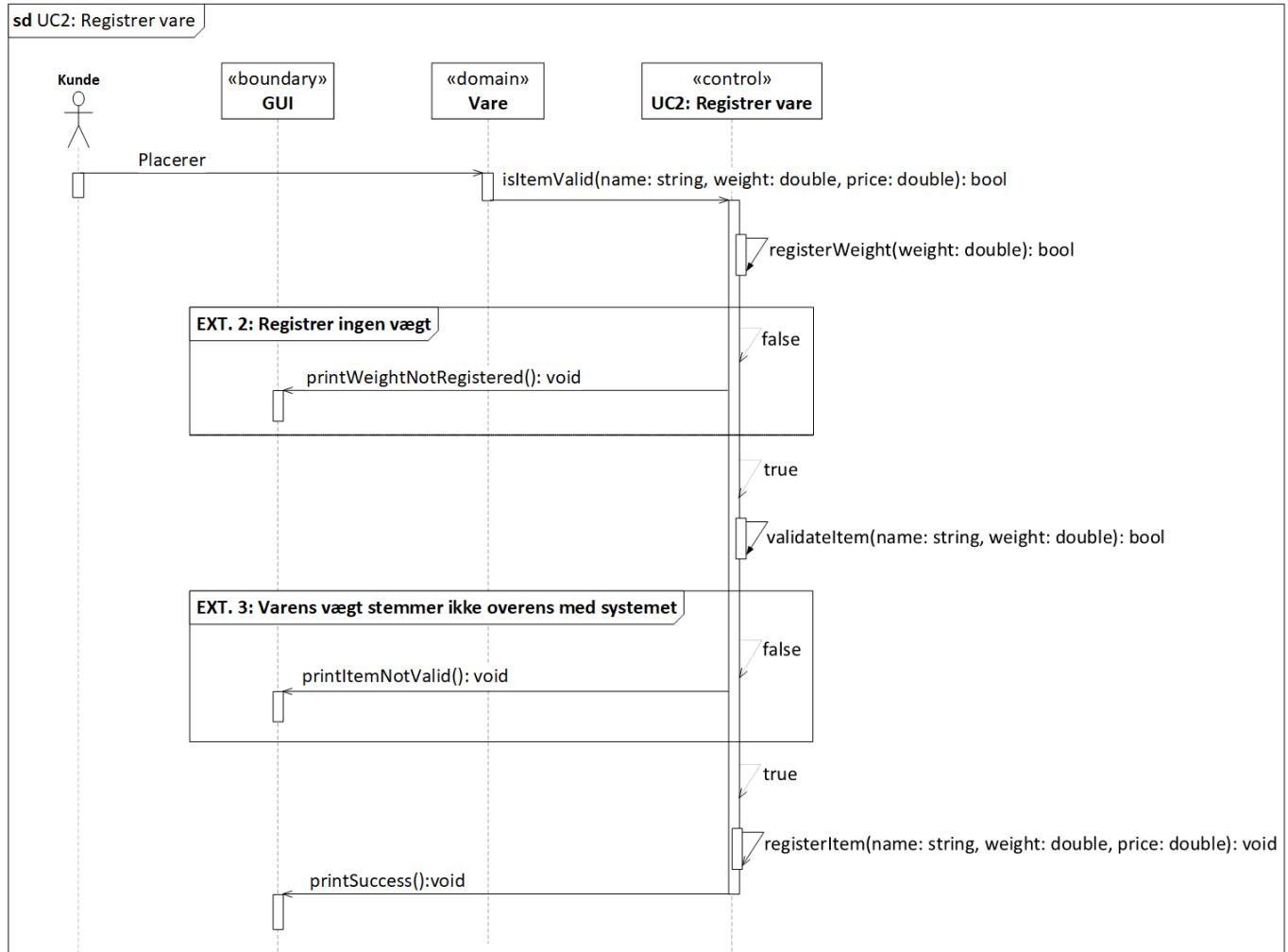


Figure 19: Sekvensdiagram med metoder for UC2: Registrer vare

Klassediagram for UC3: Fjern vare

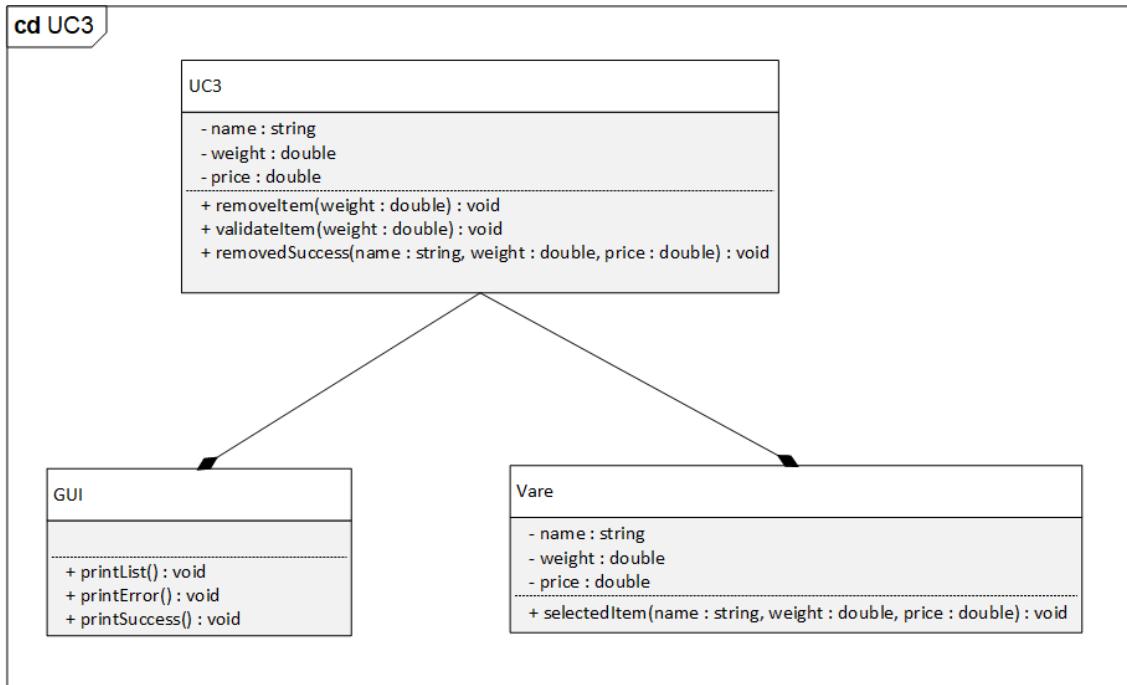


Figure 20: Klassediagram for UC3: Fjern vare

Sekvensdiagram med metoder for UC3: Fjern vare

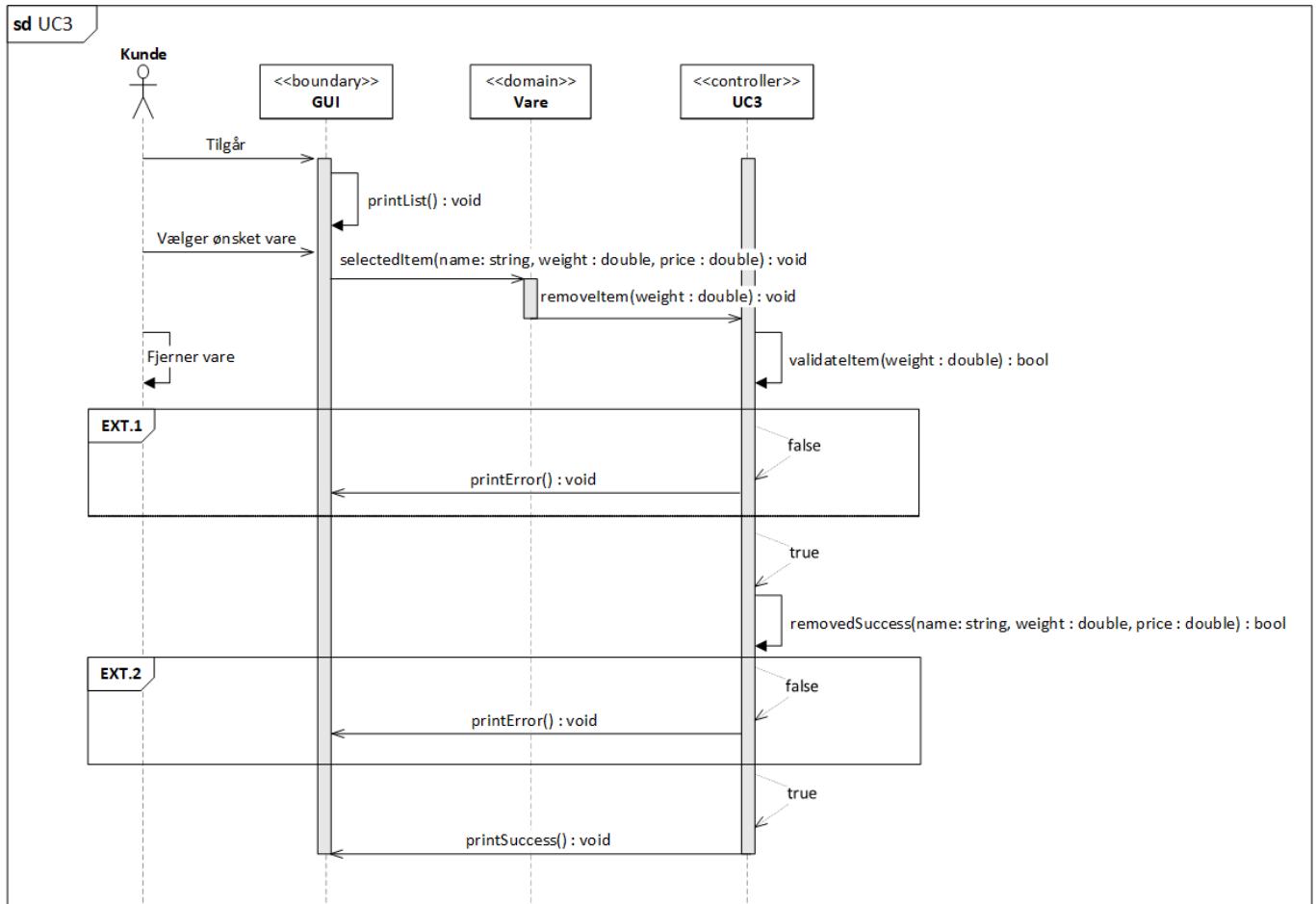


Figure 21: Sekvensdiagram med metoder for UC3: Fjern vare

6.3.2 Klassediagram for det samlede system

I figur 22 ses interaktionerne mellem klasserne *GUI*, *SmartWagon* og *Vare*, som er baseret på klassediagrammerne i figur 16, figur 18 og i figur 20, som er dannet i samspil med sekvensdiagrammerne med metoder, i figur 17, figur 19 og figur 21.

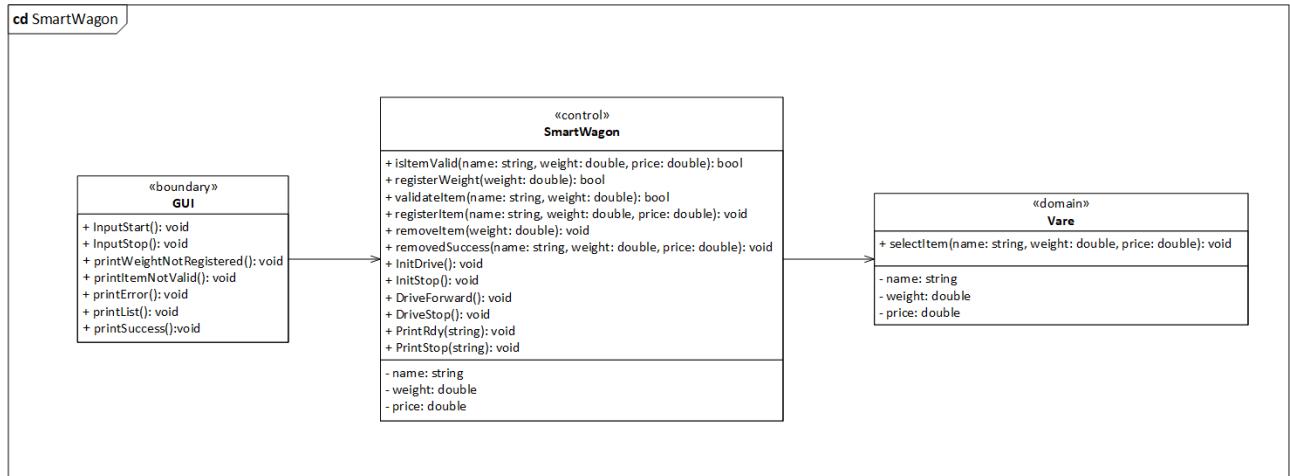


Figure 22: Klassediagram for det samlede system

6.3.3 Revurdering af applikationsmodellen for betaversionen af Smart Wagon Applikationsmodellen for UC1

UML klassediagram for UC1: Kør efter kunde

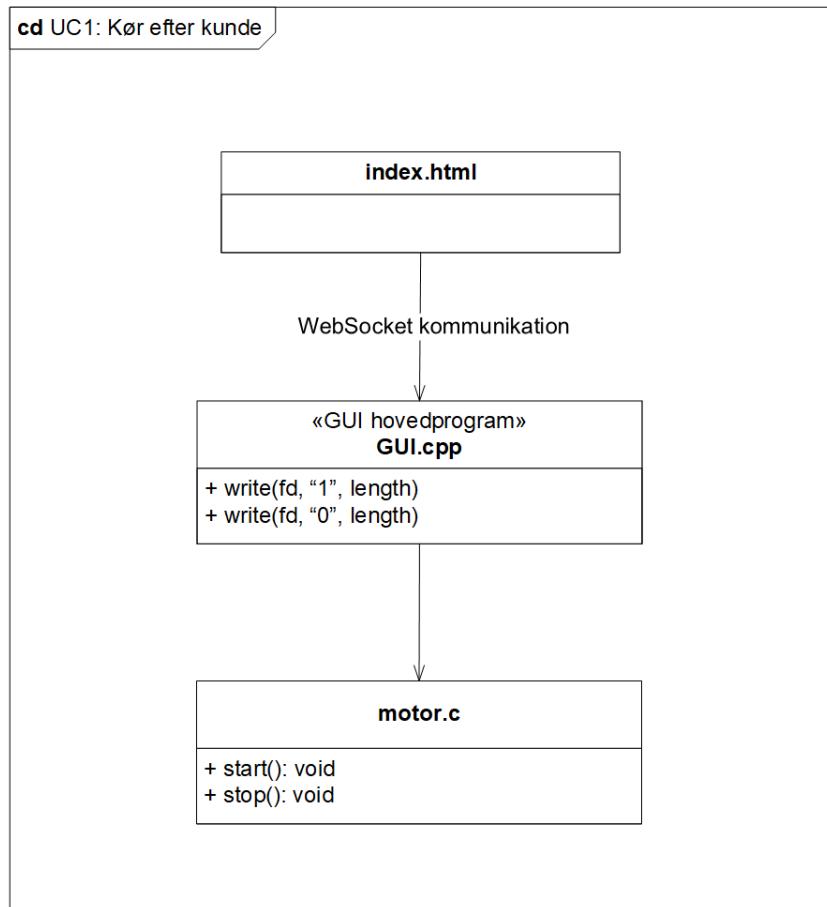


Figure 23: UML klassediagram for UC1: Kør efter kunde

Sekvensdiagram for UC1: Kør efter kunde

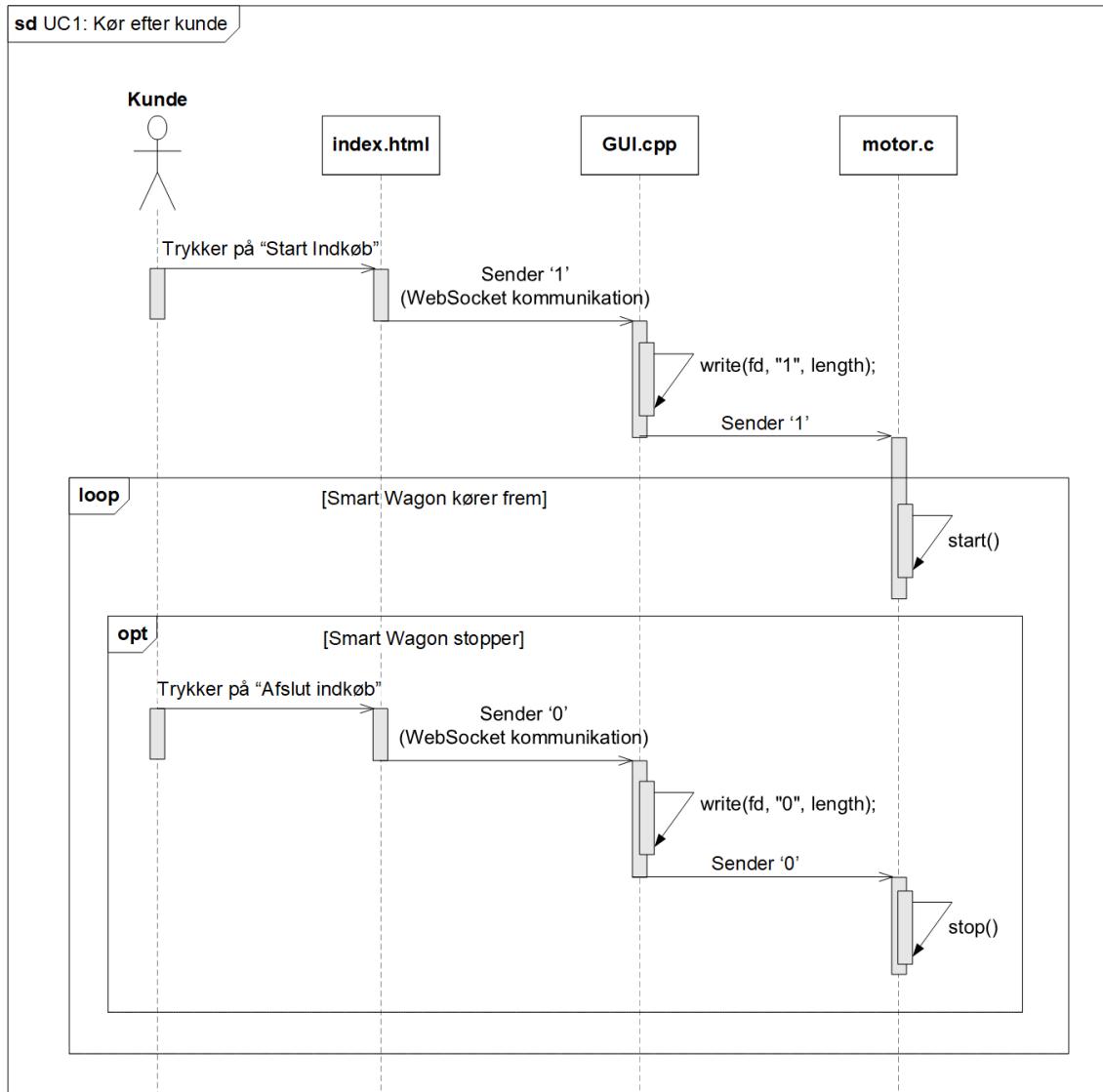


Figure 24: Sekvensdiagram med metoder for UC1: Kør efter kunde

Applikationsmodellen for UC2

UML klassediagram for UC2: Registrer vare

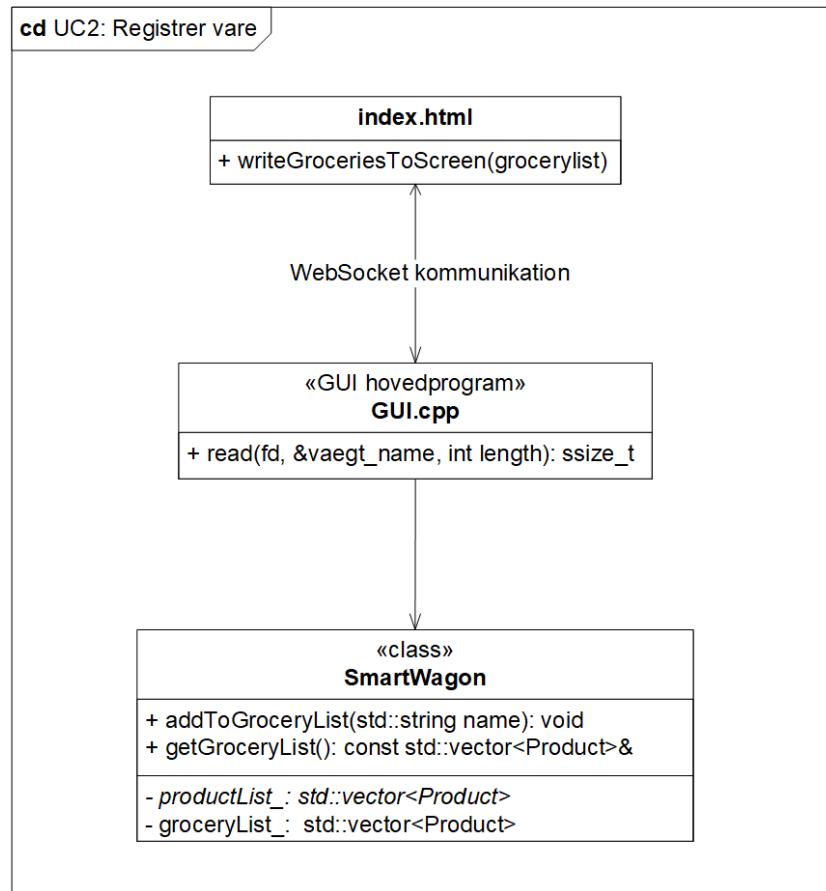


Figure 25: UML klassediagram for UC2: Registrer vare

Sekvensdiagram for UC2: Registrer vare

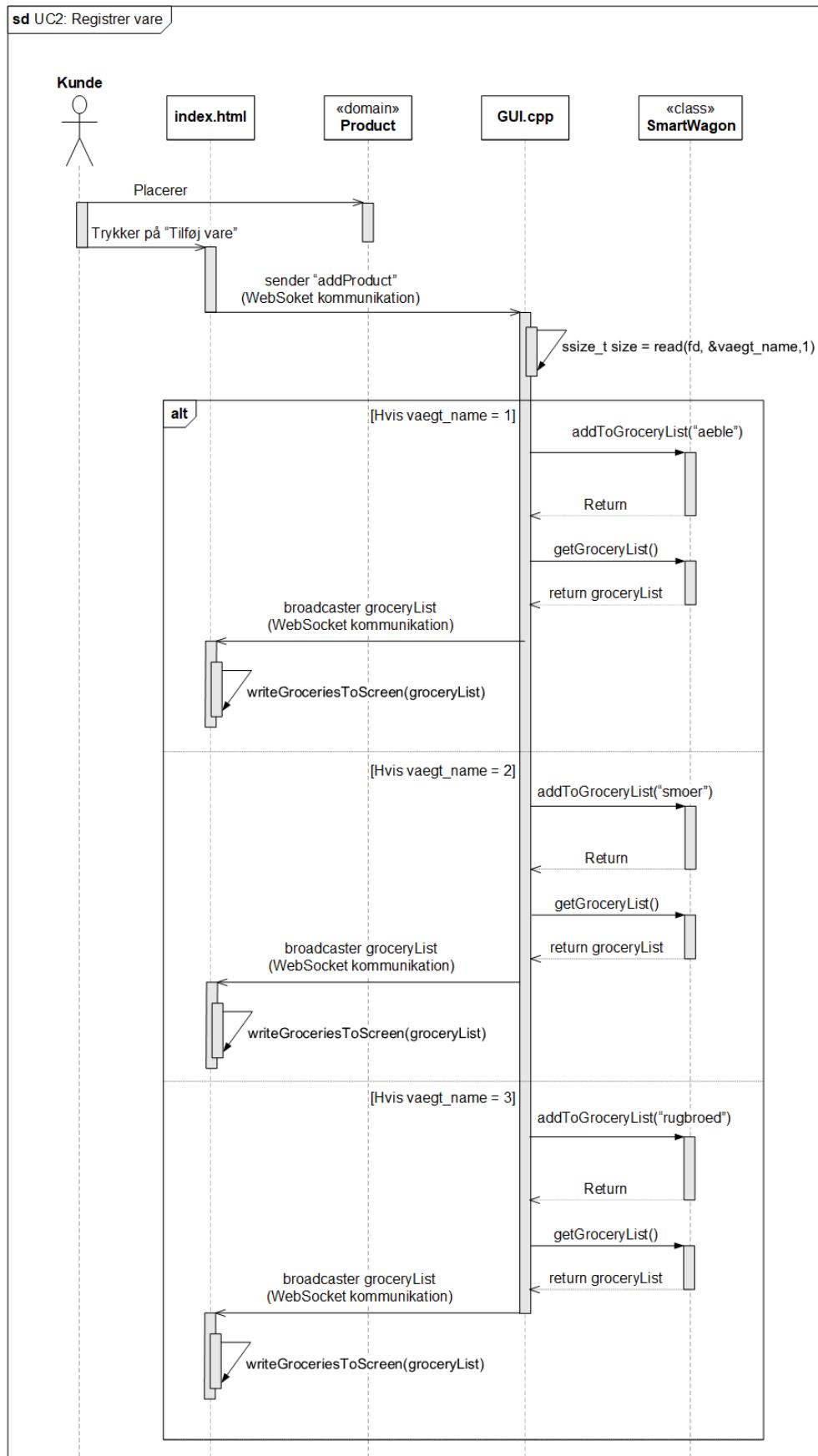


Figure 26: Sekvensdiagram med metoder for UC2: Registrer vare

Applikationsmodellen for UC3

UML klassediagram for UC3: Fjern vare

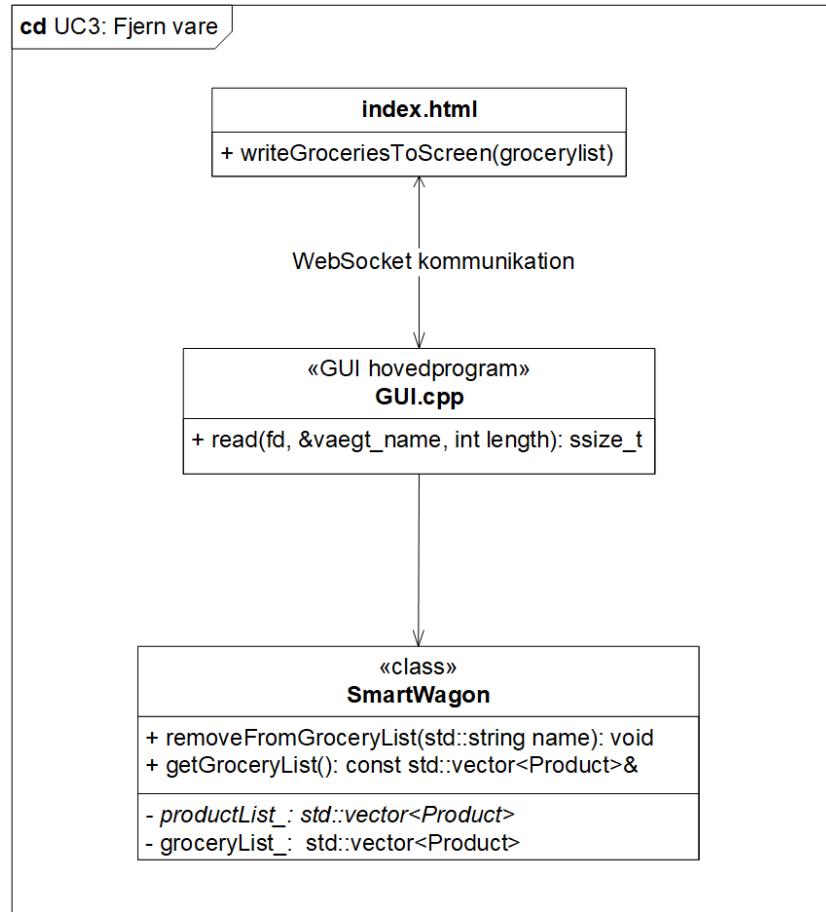


Figure 27: UML klassediagram for UC3: Fjern vare

Sekvensdiagram for UC3: Fjern vare

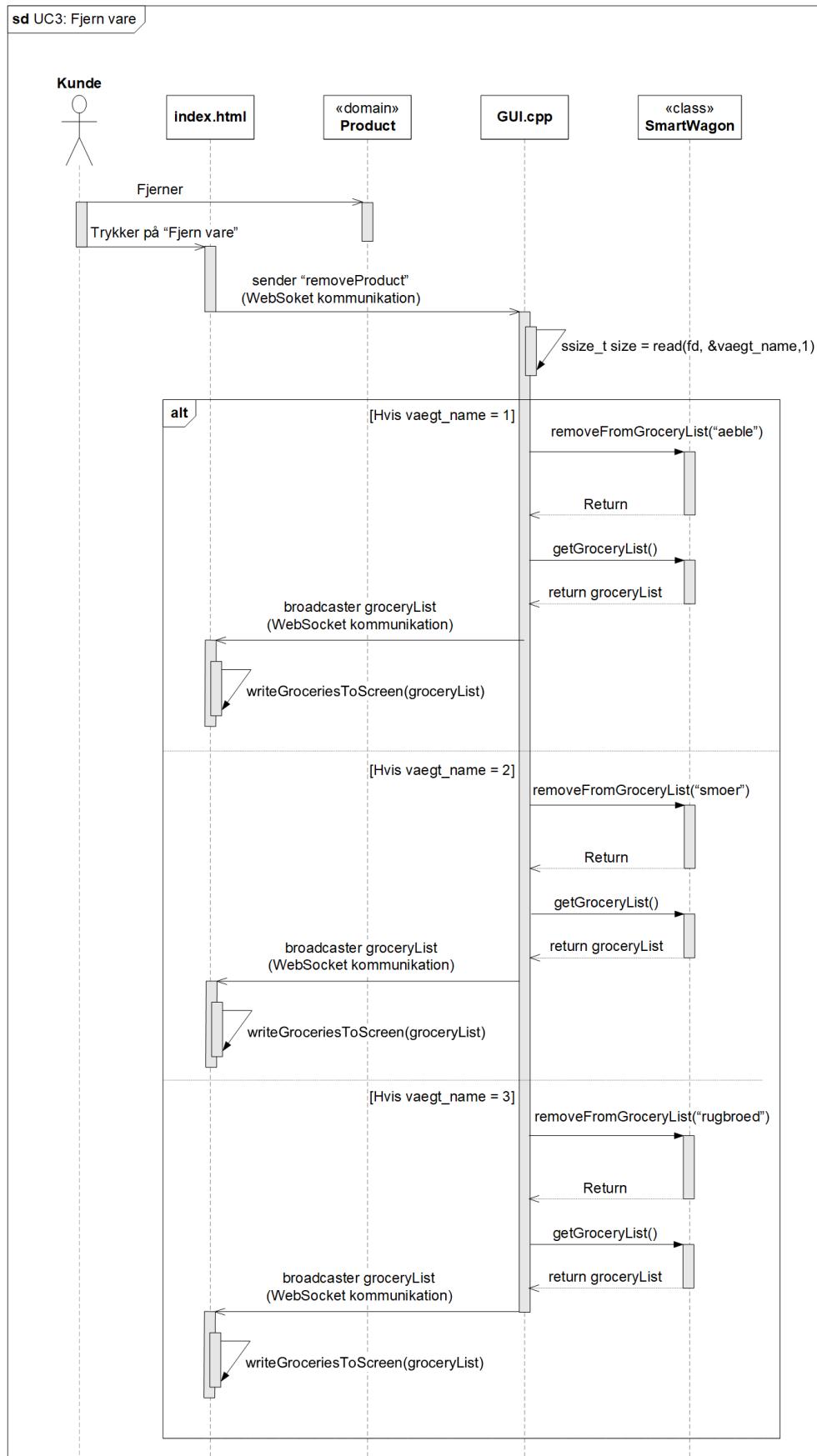


Figure 28: Sekvensdiagram med metoder for UC3: Fjern vare

7 Implementering

I dette afsnit vil implementeringen for hardwaren og softwaren blive beskrevet. Det vil i afsnittet blive beskrevet hvordan de forskellige implementeringer fungere og hvad deres formål er.

7.1 Hardware implementering

I dette afsnit vil der kigges nærmere på, hvordan de forskellige hardware moduler er implementeret.

7.1.1 Implementering af motor

Til implementeringen af motoren er der taget udgangspunkt fra den udarbejdet motor fra faget GFV³ laboratorie øvelse kaldet *Motor control*.

I betaversionen af Smart Wagon er der blevet udarbejdet en iterering af Smart Wagon, hvor DC motoren kun vil køre en vej, idet der er fokuseret på at køre fremad og at stoppe. DC motoren er styret af et PWM (Pulse-width modulation) signal, der eventuelt ville kunne styre motorens hastighed ved at ændre PWM signalets duty-cycle. Motoren får sit signal og sin spænding igennem en L290 H-bro og en PSoC, der via PSoC'en vil kunne styre, hvilken vej signalet går igennem DC motoren, og hvilket PWM signal frekvensen kører med.

I motor systemet indgår der to delsystemer: En Raspberry Pi (RPi) og en Cypress Programmable System-on-Chip L5 (PSoC). RPi'en vil sende en kommando igennem til PSoC'ens Rx input, for at kunne starte og stoppe DC motoren.

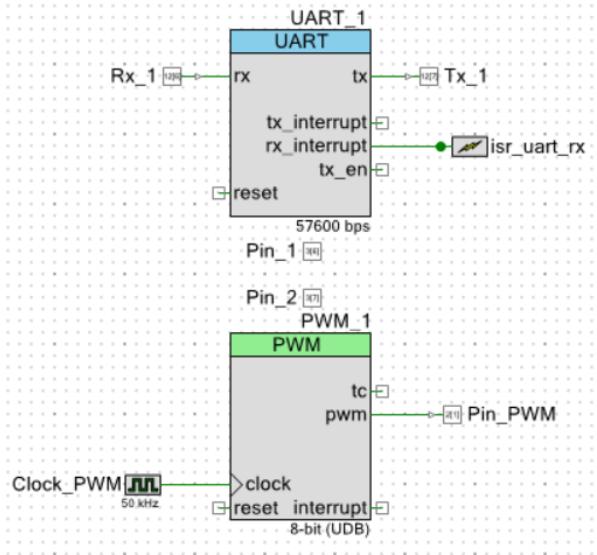


Figure 29: Top Design for DC motor

På UART modulet, er der en interrupt rutine, ved navn **isr_uart_rx**, som er tilknyttet rx_interrupt delen på UART'en. Denne har til formål at registrere om der kommer et input udefra. Dette bliver brugt i forbindelse med DC-motoren, hvor et tast på tastaturet, vil sende en besked til UART, der aktivere den givende interrupt rutine, og der udføres en handling, heraf i betaversionen af Smart Wagon, enten en start eller stop funktion af DC-motoren.

³Grenseflader til den fysiske verden

7.1.2 Implementering af vægt

Til implementeringen af vægten er der taget udgangspunkt den udarbejdet vægt fra faget GFV laboratorie øvelse kaldet *Build a scale and determine how good it is*. I denne øvelser der blevet udarbejdet en vægt som er i stand til at fortage A/D conversion. Det vil sige at vægten kan oversætte analoge input om til digitale output. Dette gør, at PSoC'en kan arbejde med en spænding, som den modtager fra det givende sensor, altså vægten. Denne data der måles fra vægten, i form af spænding, kan da omskrives, ved hjælp af noget software kode, til en vægt målt i gram.

For at øge vægtens præcision der er blevet fortaget en række målinger hvor først vægtenes linearitet er blevet undersøgt, herefter er vægtenes ikke-lineære område blevet analyseret og til sidst vægten sensitivitet blevet undersøgt.

Labratorieøvlesen konkluderer at der haves et OFF-set på 800 og at nøjagtigheden for det 95 procent konfidensniveau er +/- 1,96 gram.

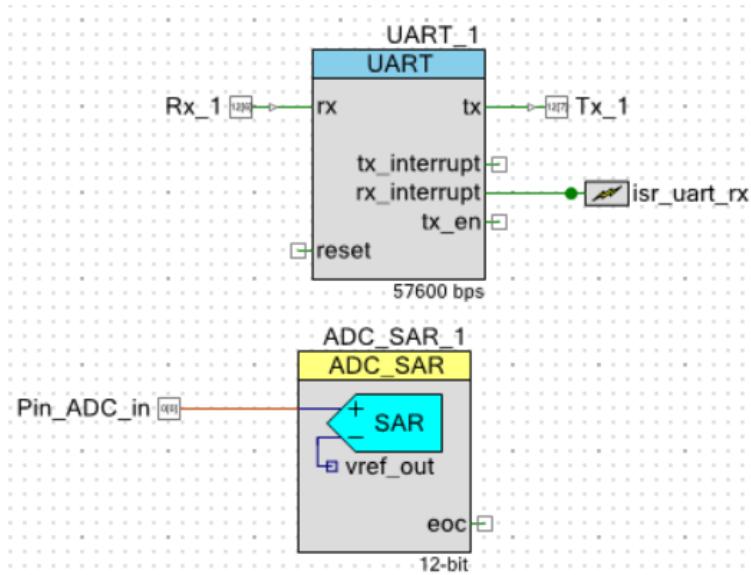


Figure 30: Top Design for vægten

På figur 30 ses det TopLevel design, der er blevet opstillet, i forbindelse med hardwaren på PSoC'en. Denne hardware fortæller PSoC'en hvilke hardware moduler den skal gøre brug af. Her ses det at der er oprettet en UART og en ADC_SAR. UART har to funktioner, hvor den ene del er beregnet til test, mens den anden er beregnet til en seriell forbindelse.

Test delen går ud på at kunne kommunikere direkte med PSoC'en, som er blevet gjort via konsol applikationen RealTerm. Heri er der blevet printet beskeder, som har været påvirket af hvordan vægten er blevet påvirket. Såsom ændring i vægt, og hvilken vægt ift. spænding der måles.

Den serielle forbindelse, ift. vægt, bliver brugt således, at der kan skabes en kontakt mellem RPi'en og PSoC'en. Denne forbindelse oprettes, således at der kan sendes data, via en Tx til Rx forbindelse. Denne forbindelse gør at der kan sendes data, vedr. hvilket produkt der skal registreres af RPi'en.

ADC_SAR modulet har till funktion at modtage analog data og konvertere det til digital data, som da kan måles og behandles af PSoC'en. Der måles en spænding over vægten, som da sendes data ind, via pin **Pin_AC_in**, som da omregnes til en digital værdi, som PSoC'en måler som en spænding. Det er denne spænding som software programmet kan behandle.

7.2 Software implementering

I dette afsnit vil software implementeringerne for Smart Wagons delsystemer blive beskrevet. Smart Wagon består af implementeringer for GUI, vægt og motor, som skal danne den samlede implementering for Smart Wagon systemet.

7.2.1 Implementering af GUI

Implementeringen af GUI blokken for SmartWagon består af C++ hovedprogrammet GUI, som anvender en instans af SmartWagon-klassen og HTML-filen *index.html*, som kommunikerer via WebSocket. Den anvendte WebSocket forbindelse danner kommunikation mellem client *index.html* og server *Raspberry Pi serveren*. Denne kommunikation dannes ved at der anvendes en instans af klassen *uWS::Hub*, som bruges til at kalde broadcast funktion. GUI hovedprogrammet fungerer som publisher, der signalerer til klienten *index.html*, som fungerer som subscriberen.

Den generelle mappestruktur for GUI implementeringen ses i figur 31, som giver et overblik over mapperne *UIIndex* og *UIMain*, hvor *UIIndex* indeholder *index.html* og *UIMain* indeholder GUI hovedprogrammet. *index.html* fungerer som HTML opsættet der danner GUI hjemmesiden for Smart Wagon, og består af tre programmeringsprog HTML, JavaScript og CSS. JavaScript bruges for at implementere handlingen der optræder som reaktion på kundens interaktion med GUI'en, og kommunikationen fra GUI hovedprogrammet. CSS anvendes til design at GUI'ens uddseende. SmartWagon-klassen indeholder metoder som indkapsler funktionalitet omhandlende tilføjelse og fjernelse af varer i Smart Wagons indkøbskurv, samt beregning af handleturens totale sum.

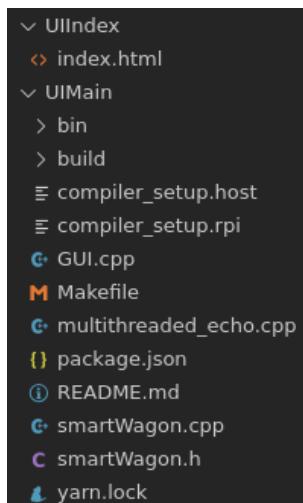


Figure 31: Udklip af filer fra repository

Anvendelse af broadcast funktionen kommer til udtryk i GUI klassens metode *sendGroceryListToGUI*, som skal modtage en instans af SmartWagon klassen og en reference til en *uWS::hub* som parametre. Broadcast funktionen anvendes til at sende indkøbslisten med de valgte produkter til *index.html* filen. Dette registreres i *index.html* filen, ved at de mulige modtagende strings skal starte med "GroceryList:". Herefter kaldes *writeGroceriesToScreen* med parameteren "list" som indeholder henholdsvis de valgte produkter og information af karakteren #, som danner linjeskiftene i udskriften på GUI'en.

GUI hovedprogrammet indeholder en *main()*-funktion, hvor oprettelse af alle produkterne i SmartWagons indebyggede produktliste optræder. SmartWagon klassen består af alle de nødvendige metoder som omhandler tilføjelse og fjernelse af produkter i produktlisten og indkøbslisten for alle varer som placeres i SmartWagons indkøbskurv.

Kodeudsnit af GUI implementeringen

I figur 32 ses implementeringen af kommunikationen af produktinformation fra GUI klassen til *index.html*

filen. Heraf venter index.html filen på at modtage tekststringen "GroceryList:", hvorefter at den udskriver produkternes titel og pris.

```
void sendGroceryListToGUI(SmartWagon& wagon, uWS::Hub& hub)
{
    // Sender den fulde indkøbslisten til GUI hjemmesiden
    const std::vector<Product>& list = wagon.getGroceryList();

    // Sender listen til GUI, så GUI kan vise listen
    std::ostringstream ss;
    ss << "GroceryList:";
    double sum = 0;
    // For hvert produkt i listen, skrives navn og pris til ostream "GroceryList"
    for(unsigned long i = 0; i < list.size(); i++)
    {
        ss << "Vare: " << list[i].name_ << std::fixed << std::setprecision(2)
           << "      Pris: " << list[i].price_ << " kr." << "#";
        sum = sum + list[i].price_;

        if(i != list.size() - 1)
        {
            ss << "Det nuværende beløb: " << std::fixed << std::setprecision(2)
               << sum << " kr." << "#";
        }
    }
    ss << "Det totale beløb: " << std::fixed << std::setprecision(2)
       << wagon.totalSum() << " kr.";

    // Sender hele indkøbslisten til GUI, så GUI kan vise indkøbslisten
    hub.broadcast(ss.str().c_str(),ss.str().length(), uWS::OpCode::TEXT);
}
```

Figure 32: Udklip af GUI klassen

I figur 33 ses kommunikationen mellem index.html og metoden sendGroceryToGUI, fra klassen GUI, samt anvendelsen af klassen SmartWagon.

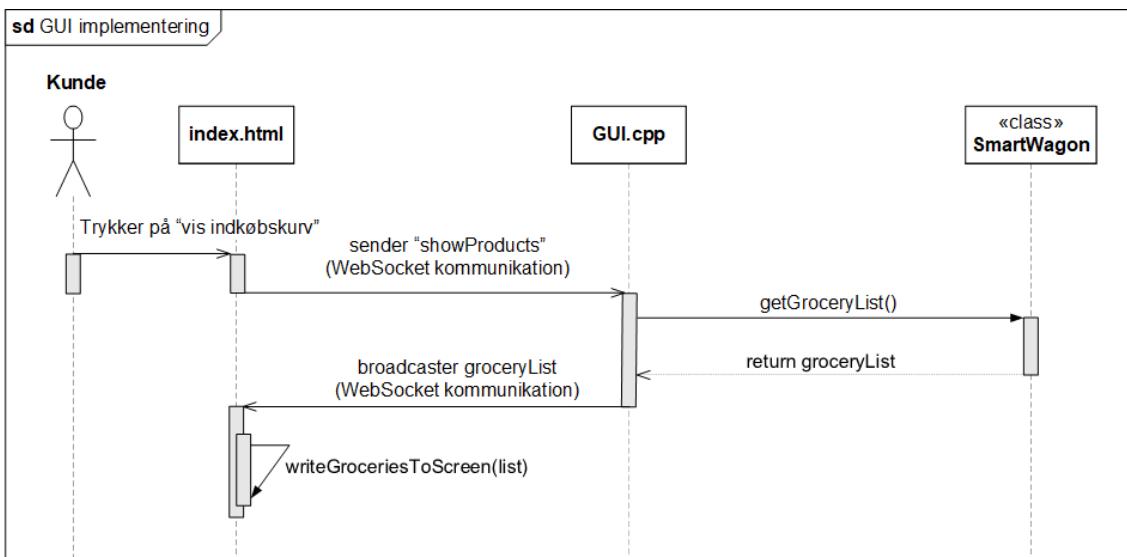


Figure 33: Sekvensdiagram for kommunikation ift. metoden sendGroceryToGUI

I figur 34 ses den faktiske udskrift af handlingen der beskrives i figur 33, hvor indkøbskurvens indhold opdateres automatisk med både titel og pris, samt en foreløbig beløb og efterfølgende de resulterede beløb.

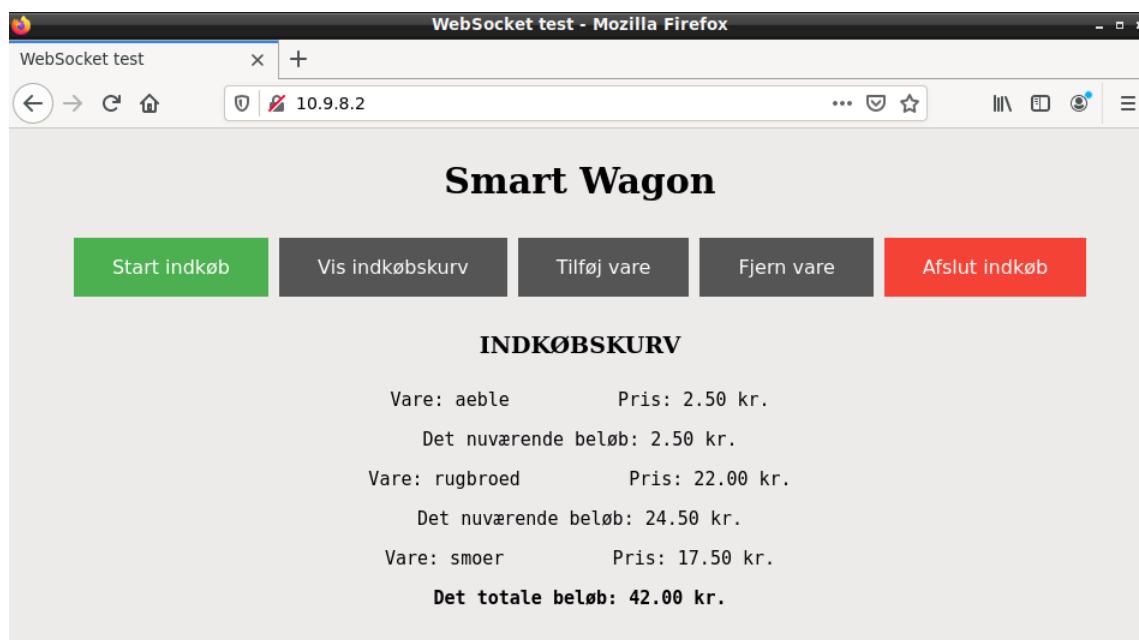


Figure 34: GUI for SmartWagon

I figur 35 ses konsoludskriften på RPi, for kørsel af GUI hovedprogrammet, når der tilføjes og fjernes produkter fra indkøbskurven. Dette viser den bagvedliggende funktionalitet for SmartWagon.

```
root@raspberrypi0-wifi:~# ./UI
Added product with name: aeble, weight: 170 and price: 2.5
Added product with name: smoer, weight: 500 and price: 17.5
Added product with name: rugbroed, weight: 750 and price: 22
Initializes Smart Wagon
Terminates Smart Wagon
Initializes Smart Wagon
Terminates Smart Wagon
Initializes Smart Wagon
Adds product to grocery list
Added product to grocery list with name: aeble
Adds product to grocery list
Added product to grocery list with name: rugbroed
Adds product to grocery list
Added product to grocery list with name: smoer
Terminates Smart Wagon
Initializes Smart Wagon
Prints grocery list
Removes product from grocery list
Removed product with name: rugbroed
Removes product from grocery list
Removed product with name: smoer
Removes product from grocery list
Removed product with name: rugbroed
Terminates Smart Wagon
```

Figure 35: GUI for SmartWagon

7.2.2 Motor implementering

Motoren er blevet implementeret med en ISR⁴, der kan registrere, om der kommer et input fra RPi. Dette input er tilkoblet PSoC'en således, at PSoC'en kan behandle data og herved give kommandoer, som bestemmer DC-motorens adfærd.

Dataen som registreres af ISR, er i dette tilfælde enten et logisk '1' eller '0'. Denne værdi bestemmer, om den givne pin er aktiv, hvilket styrer om DC-motoren kører frem eller stopper. Der bliver opsat et PWM signal med en duty cycle som styrer hvor stor del af tiden, DC-motoren modtager den maksimale spænding, og bestemmer hastigheden for DC-motoren. I betaversionen vil PWM pulsbredden, være på en konstant tilstand, med en pulsbredde på 50 %.

I praksis vil værdierne som bliver sendt fra RPi'en, være information som detekteres fra brugergrænsefladen, som sendes via en Tx/Rx forbindelse. Når kunden igangsætter indkøbet via brugergrænsefladen, vil DC-motoren bliver aktiveret og når indkøbet afsluttes vil motoren slukkes. Det vil sige at motorens adfærd igangsættes af kunden.

```
10 void handleByteReceived(uint8_t byteReceived)
11 {
12     switch(byteReceived)
13     {
14         case '1' :
15         {
16             start();
17         }
18         break;
19         case '0' :
20         {
21             stop();
22         }
23         break;
24         default :
25         {
26         }
27         break;
28     }
29 }
```

Figure 36: Motorkode Switch Case Handler

⁴interrupt service routine

```

31 void start()
32 {
33     PWM_1_WriteCompare(Speed);
34
35     IN1 = 1;
36     IN2 = 0;
37
38     Pin_1_Write(IN1);
39     Pin_2_Write(IN2);
40     //UART_PutString(" - Motor start\r\n");
41     for(;Speed <= 50;Speed += 5)
42     {PWM_1_WriteCompare(Speed);
43     CyDelay(150);
44     }
45 }
46
47 void stop()
48 {
49     IN1 = 0;
50     IN2 = 0;
51
52     Pin_1_Write(IN1);
53     Pin_2_Write(IN2);
54     //UART_PutString(" - Motor stop\r\n");
55     Speed = 0;
56 }
```

Figure 37: Motor styring

7.2.3 Implementering af vægt

Implementeringen af vægten er baseret på baggrund af en sensor, som mäter en spænding, ved hjælp af en ADC SAR⁵. Først vil SAR tjekke om der er noget at mæle, hvis ja, vil den givne spænding konverteres, fra analog til digital. Vægten har et offset på 800 mV. Det vil altså sige, at når der omskrives fra mV til gram, vil der ved 0 gram være en spænding på 800 mV. PSoC'en omregner den målte spændingen om til vægt, målt i gram. Dette sker i funktionen "voltToGram". "voltToGram" adder 0.8 gram pr. målte spænding. Altså vil der være en vægt på 240 gram ved 300 mV (1100 hvis offset tages med i beregningerne). Der bliver lavet 6 målinger, med 0.25 sekunders mellemrum pr. måling (1.5 sekunder i alt), af den genstand der sættes på vægten. Dette skyldes sikring af korrekt vægt. Hvis der blot vejes en pr. sekund, eller ligende, vil vægten kunne være påvirket af udefrakommende forstyrrelser.

```

66 float voltToGram(uint16_t volt) {
67
68     float gram = 0;
69
70     for (uint16_t convert = 810; convert < volt; convert++) {
71         gram += 0.8;
72     }
73
74     return gram;
75 }
76 }
```

Figure 38: Kode for funktionen voltToGram

Efter målingen bliver der tjekket efter, om der er blevet målt en ny vægt. For at der registreres en ny vægt, skal der være et skift på mere eller mindre end 10 gram. Hvis vægten defineres som ny, vil PSoC'en bestemme hvilket produkt der er på vægten, ud fra hvilket interval målingen ligger i. Dette sker i funktionen "setProduct". Der er i betaversionen 3 varer, der spænder mellem intervallerne 1-300

⁵Successive Approximation Register

(æbler, numerisk værdi: 1), 301-600 (smør, numerisk værdi: 2) og 601-900 (rugbrød, numerisk værdi: 3) gram. Når der er blevet bestemt hvilken vægt (og derved varer) der er registreret, vil produktnummeret blive sat klar til transmittering.

```

51
52     if (resultGram > checkWeight + 10 || resultGram < checkWeight - 10) {
53
54         product = setProduct(resultGram);
55
56         checkWeight = resultGram;
57
58         sprintf(uartBuffer, sizeof(uartBuffer), "Check: %d - result: %2.f\r\n", checkWeight, resultGram);
59         UART_PutString(uartBuffer);
60
61     }
62

```

Figure 39: Kode for at tjekke vægt

Heresfter vil denne informationen sendes fra PSoC'en til RPi'en i systemet, via en UART seriel forbindelse. Dette sker via transmission af data, ved hjælp af en Rx til Tx forbindelse. Denne transmission opstår samtidigt med registrering af varen, altså, hvert 0.25 sekund. Dette er en yderligere grund til tidsintervallet, nævnt tidligere, da den data der skal sendes til at RPi'en, skal sendes ofte nok til at kunne blive registreret. Der sendes de numeriske værdier for produkterne, som registreres af WebSocket. I tilfælde af at kunden bruger funktionen ”Tilføj vare”, vil der på brugergrænsefladen blive informeret om at en vare er blevet tilføjet. Yderligere kan en vare fjernes, alt afhængigt af hvilken vægt der er registreret. Herefter vil brugergrænsefladen slette produktet. Dette sker via UART funktionen ”UART_PutString(”numerisk produkt værdi”);”. Hernæst gentages hele rutinen, så systemet igen vil transmittere data og tjekke efter vægt, og derved også efter ny vare.

```

7  uint16_t setProduct(uint16_t weight) {
8
9     char uartBuffer[256];
10
11    uint16_t product = 0;
12    uint16_t aeble = 1;
13    uint16_t smoer = 2;
14    uint16_t rugbroed = 3;
15
16    // Varer sættes klar til transmittering
17    if (weight > 0 && weight <= 300) {
18
19        // Sætter produkt type
20        product = aeble;
21
22        // Klargøre tekst til print i konsol
23        sprintf(uartBuffer, sizeof(uartBuffer), "Aeble er tilføjet til kurven\r\n");
24
25    }

```

Figure 40: Kode for funktionen setProduct



8 Accepttest

Dette afsnit indeholder den udarbejdede accepttest som er blevet lavet for Smart Wagons betaversion og en konklusion for den udførte accepttest. Accepttesten er tilegnet en tilfældig kunde, for at se om kunden kan bruge systemet som forventet.

8.1 Accepttest for betaversionen af de funktionelle krav

I dette afsnit præsenteres de fire udarbejdede accepttest af de funktionelle krav for Smart Wagons betaversion. Accepttesten er opstillet ud fra de opstillede Use Cases.

8.1.1 Accepttest for Use Case 1: Kør efter kunde

Use Case under test	UC1: Kør efter kunde			
Scenarie	Hovedscenarie			
Prækondition	Smart Wagons DC motor og PSoC er tilsluttet strøm			
Nr.	Handling	Forventet observation	Faktisk observation	Vurdering
1	Kunden trykker på "Start indkøb"	Smart Wagons DC motor starter, og der udskrives "INDKØB STARTES" på Smart Wagons GUI.	Smart Wagons DC motor starter, og der udskrives "INDKØB STARTES" på Smart Wagons GUI, når der trykkes på "Start indkøb"	OK
2	Kunden trykker på "Afslut indkøb" på Smart Wagons GUI	Smart Wagons DC motor stopper, og der udskrives "INDKØB AFSLUTTES" på Smart Wagons GUI	Smart Wagons DC motor stopper, og der udskrives "INDKØB AFSLUTTES" på Smart Wagons GUI, når der trykkes på "Afslut indkøb"	OK

Table 9: Accepttest for UC1: Kør efter kunde

8.1.2 Accepttest for Use Case 2: Registrer vare

Use case under test	UC2: Registrer vare			
Scenarie	Hovedscenarie			
Prækondition	Smart Wagon er klar til at scanne en vare ved, at der er trykket på "Start indkøb" på GUI			
Nr.	Handling	Forventet observation	Faktisk observation	Vurdering
1	Kunden placerer en vare på vægten, i indkøbskurven, og kunden trykker på "Tilføj vare" på Smart Wagons GUI	Smart Wagon tilføjer varen til indkøbskurven på Smart Wagons GUI	Smart Wagon tilføjer varen til indkøbskurven på Smart Wagons GUI, når der sættes en vare på vægten, og der trykkes på "Tilføj vare" på Smart Wagons GUI	OK

Table 10: Accepttest for UC2: Registrer vare



8.1.3 Accepttest for Use Case 3: Fjern vare

Use Case under test	UC3: Fjern vare			
Scenarie	Hovedscenarie			
Prækondition	Smart Wagons GUI og vægt er klar til brug ved, at der er trykket på ”Start indkøb” på GUI			
Nr.	Handling	Forventet observation	Faktisk observation	Vurdering
1	Kunden trykker på ”Vis indkøbskurv” på Smart Wagons GUI	Smart Wagons GUI viser indkøbskurvens indhold	Smart Wagons GUI viser indkøbskurvens indhold, når kunden trykker på ”Vis indkøbskurv”	OK
2	Kunden placerer en vare på vægten og trykker på ”Fjern vare” på Smart Wagons GUI	Smart Wagon fjerner den vare, der er på vægten i GUI’ens indkøbskurv	Smart Wagon fjerner den vare, der er på vægten i GUI’ens indkøbskurv, når kunden trykker sætter en vare på vægten og trykker på ”Fjern vare”	OK

Table 11: Accepttest for UC3: Fjern vare

8.1.4 Accepttest for Use Case 4: Foretag fejlhåndtering

Use Case under test	UC4: Foretag fejlhåndtering			
Scenarie	Hovedscenarie			
Prækondition	Smart Wagons Centralssystem (RPi), vægt og GUI er klar til brug			
Nr.	Handling	Forventet observation	Faktisk observation	Vurdering
1	Personalet frakobler Centralssystemet (RPi) fra GUI	Fejlbeskedden ”Forbindelse til Centralssystemet (RPi) er afbrudt” udskrives til Smart Wagons konsolvindue	Intet at fejlteste	FEJL
2	Personalet håndterer fejlen	Personalet tilkobler Centralssystemet (RPi) til GUI og genstarter Centralssystemet (RPi)	Intet at fejlteste	FEJL
3	Centralssystemet (RPi) udskriver en besked til Smart Wagons konsolvindue, om at den er operationel	Smart Wagon er klar til brug	Intet at fejlteste	FEJL

Table 12: Accepttest for UC4: Foretag fejlhåndtering

8.2 Accepttest for betaversionen af de ikke-funktionelle krav

I dette underafsnit præsenteres accepttesten for de ikke-funktionelle krav for Smart Wagons betaversion.

Nr.	Krav	Test	Forventet observation	Faktisk observation	Vurdering
1	Smart Wagon udregner varens pris baseret på antal registerede varer	Visuel test: Smart Wagon beregner varens pris, som vises i GUI’ens indkøbskurv	Den udregnede pris vises på GUI	Den udregnede pris for antal varer i indkøbskurven, når man tilføjer eller fjerner varer på GUI	OK
2	Kunden interagerer med en GUI på PC	Visuel test: Kunden kan se GUI på PC	Kunden kan se og interagere med GUI på PC	Kunden har mulighed for at interagere med Smart Wagons GUI	OK
3	Smart Wagon skal kunne genstartes af alle gruppemedlemmer	Fysisk test: Centralssystemet (RPi) genstartes af en tilfældig gruppemedlem	Smart Wagon genstartes succesfuldt	Smart Wagon kan genstartes af alle medlemmer i projektgruppen	OK

Table 13: Accepttest for de ikke-funktionelle krav



8.3 Konklusion for accepttesten

Projektgruppen har, i samarbejde med vejleder, afholdt accepttest tirsdag d. 01-06-2021. Formålet med accepttesten var at få testet de implementerede krav, som blev opstillet for Smart Wagons betaversion.

Det konkluderes, at accepttesten gik som gruppen forventede, idet accepttest for Use Case 1, 2, 3 og testen for de ikke funktionelle krav blev godkendt. Det vil sige, at ved de opstillede handlinger blev det forventede observeret.

Accepttesten for Use Case 4 blev, som forventet, ikke godkendt, fordi muligheden for fejlhåndteringen ikke er blevet implementeret i betaversionen. Implementeringen af denne Use Case blev nedprioriteret grundet tidsmangel. Projektgruppen vurderede, at det ville være vigtigere for slutproduktet at Use Case 1-3 blev færdig implementeret fremfor.



9 Bilagsliste

- Information om ASE modellen
DevelopmentProces.pdf
- Information om den agile udviklingsproces SCRUM
Scrum.pdf
- Load Cell Amplifier datablad
AD620.pdf
- ASE fHat datablad
ase_fhat_schematic.pdf
- DC-motor datablad
DC Motor IGARASHI 24475.pdf
- L298 H-bro datablad
L298.pdf
- Load Cell datablad
Load Cell.pdf
- Raspberry Pi Zero w datablad
RPI-ZERO-V1_3_reduced.pdf
- Forventede tidsplan.pdf
- Opdateret tidsplan.pdf
- Samarbejdsaftale.pdf
- Dagsorden_Rerferat.pdf
- Logbog.pdf
- Procesbeskrivelse.pdf
- Kode implementeringer
 - GUI_implementering.zip
 - PSoC_implementering.zip
- Acceptttest1.png
- Acceptttest2.png



10 Referenceliste

- I2ISE Compendium, Revision 2, June 2012, Aarhus Universitet Ingeniørhøjskolen.
Heraf refereres det til
 - Craig Larman
 - Applying UML and Patterns
 - Prentice Hall PTR, 3.ed.
 - Chapter 5, Requirements, pp 54-57
- Artikel om den agile udviklingsproces ”Making sense of MVP” af Henrik Kniberg
<https://blog.crisp.se/2016/01/25/henrikkniberg/making-sense-of-mvp>
- Information om den anvendte metode Git <https://en.wikipedia.org/wiki/Git>
- Information om MAX sonar-sensoren
https://www.maxbotix.com/ultrasonic_sensors/mb1202.htm
- Information om infrarød-sensoren
https://global.sharp/products/device/lineup/data/pdf/datasheet/gp2y0a41sk_e.pdf
- Information om Time of flight-sensoren
<https://www.adafruit.com/product/3317>
- CY8CKIT-059 PSoC® 5LP Prototyping Kit datablad
<https://www.cypress.com/file/157971/download>
- Logic level converter
<https://learn.sparkfun.com/tutorials/bi-directional-logic-level-converter-hookup-guide>