AARHUS UNIVERSITET

3. Semesterprojekt

Smart Wagon

Projektrapport

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

AARHUS
UNIVERSITET

1 Abstract

The project *Smart Wagon* describes a 3rd semester project at the Aarhus School of Engineering, Aarhus University. The group consists of 6 members, all from the faculty "Software Technology". The development process in this project is based on the ASE model, and the product development is based on the iterative model *Scrum*. The purpose of the project *Smart Wagon* is to optimize the daily shopping trip, where the project group's implementation of Smart Wagon has consisted of implementing a beta version of Smart Wagon. The beta version contains the most important functionalities of Smart Wagon, such as a sensor in the form of a scale that weighs the item, whereafter Smart Wagon calculates and shows the price of the item, and subsequently the total sum for the shopping trip. The interaction between Smart Wagon and the customer is created via the implemented Graphical User Interface. Furthermore, the beta version consists of a DC motor that can rotate forward and stop. The start function on the Graphical User Interface can start the DC motor, which is essential for the full version of the Smart Wagon. With Covid-19 in mind, Smart Wagon is the optimal solution by reducing contact surfaces, which helps reducing the risk of infection during the shopping trip as the customer doesn't have to touch the shopping basket. Unfortunately a problem arose in regards to collecting the hardware components from the workshop, thus hindering a smoooth and ideal process.

2 Resumé

Projektet *Smart Wagon* beskriver et 3. semesterprojekt på Ingeniørhøjskolen, Aarhus Universitet. Projektgruppen består af 6 medlemmer, fra linjen "Softwareteknologi". Udviklingsprocessen i dette projekt er baseret på ASE-modellen, hvoraf at produktudviklingen er baseret på den iterative-model *Scrum*. Formålet med projektet *Smart Wagon* er at optimere den daglige handletur, hvor projektgruppens implementering af Smart Wagon har bestået i at implementere en betaversion af Smart Wagon, idet der opstod afhentningsproblemer ift. nødvendige hardwarekomponenter. Betaversionen indeholder de vigtigste funktionaliteter af Smart Wagon, såsom en sensor i form af en vægt, der vejer handlevaren, hvorefter Smart Wagon beregner og viser varens foreløbige pris, og efterfølgende den totale sum for handleturen. Denne interaktion mellem Smart Wagon og kunden oprettes via den implementerede *Graphical User Interface*. Derudover består betaversionen af en DC-motor, der kan rotere frem og stoppe. Smart Wagons DC motor kan startes via start-funktionen på Smart Wagons *Graphical User Interface*, hvilket er ideelt for den fulde version af Smart Wagon. I forhold til Covid-19, er det i denne tid optimalt at mindske berøring med kontaktoverflader, og derfor kan Smart Wagon bidrage til at mindske smitterisikoen under indkøbsturen.

3 Versionshistorik

Version	Dato	Beskrivelse
1	06-02-2021	Projektformulering
2	21-02-2021	Funktionelle krav
3	05-03-2021	Kravspecifikation, Accepttest og BDD
4	18-03-2021	Domænemodel, Risk Assessment, WebSocket
5	19-03-2021	Sekvensdiagrammer for handling
6	03-04-2021	Opdatering af Sekvensdiagrammer for handling, Opdatering af Domænemodel, Oprettet definitionsliste, Rettelse af Risk Assesment
7	22-04-2021	Applikationsmodeller med sekvensdiagrammer med metoder og tilhørende UML klassediagrammer
8	30-04-2021	Start af implementering af Smart Wagons delsystemer
9	25-05-2021	Teknologianalyse, Generel projektskrivning
10	26-05-2021	Hardware test, Opdatering af accepttest
11	28-05-2021	Modultest, Procesbeskrivelse
12	29-05-2021	Modultest/test af betaversion, Opdatering af projektrapport og bilagsrapport
13	31-05-2021	Test af betaversion, Rettelse af projektrapport og bilagsrapport
14	01-06-2021	Accepttest af betaversion, Rettelse af projektrapport og bilagsrapport
15	02-06-2021	Færdiggørelse af projektrapport, Opdatering af bilagsliste og referenceliste

Table 1: Versionshistorik

4 Definitionsliste

Forkortelse	Definition
BDD	Block Definitions Diagram
IBD	Internal Block Diagram
UC	Use Case
PSoC	Programmable System-on-Chip
GUI	Graphical User Interface
GFV	Grænseflader til den fysiske verden
ISU	Indlejret softwareudvikling
HAL	Hardware abstraktioner
MTBF	Mean Time Between Failures
RPi	Raspberry Pi
HW	Hardware
SW	Software
PWM	Pulse-width modulation

Table 2: Definitionsliste

5 Arbejdsfordeling

Arbejdsfordelingen i projektgruppen har været således, at alle medlemmer har været med til at udarbejde de overordnet rammer, krav og specifikationer for udarbejdelsen af Smart Wagon.

5.1 Oversigt over arbejdsfordeling

I tabel 3 kan en oversigt over den specifikke arbejdsfordeling ses. Tabellen viser hvad de enkle gruppemedlemmer har haft primært ansvar for.

Ansvarsområde	**Andreas**	**Asger**	**Azar**	**Christina**	**Johanne**	**Maagisha**
Design og implementering af Vægt	X				X	
Design og implementering af Motor		X	X			
Design og implementering af GUI				X		X

Table 3: Arbejdsfordeling i projektgruppen

AARHUS UNIVERSITET

Indholdsfortegnelse

6 Indledning

Projektet Smart Wagon er dannet på baggrund af en almindelig indkøbskurv, hvor der er fokuseret på en ny og bedre udgave, med en smart implementering. Formålet med denne smarte udgave er at mindske de lange køer ved kassebåndet, og dermed forbedre indkøbsoplevelsen.

Smart Wagon gør danskerens dagligvareindkøb til en behagelig oplevelse, dramafri fra dengser, drenge og døtre. Ved det simple design, som styres af kunden selv, vil den trælse oplevelse af en indkøbskurv med slidte hjul være forbi. Smart Wagon vil hele tiden være opdateret på det givne indkøb, både med de planlagte varer, men skam også de eftertragtede impulskøb. Det er nemt at styre indkøbets gang, via den integrerede brugergrænseflade, der kan tilgås fra flere forskellige enheder, samt Smart Wagon i sig selv. Scan varen og læg den ned, her vil indkøbet gå nemt og bekvemt. Ved brug af nemt tilgængelige forskellige typer af brugergrænseflader, er der ikke længere brug for upraktiske applikationer, der skal trækkes op og ned af lommen, samt låses og låses op, når hånden der skal bruges til at rage vare til sig, som kurven ellers så galant holder på.

Ved hjælp af sensorteknologi, vil Smart Wagon være i stand til både at scanne kunden og de aktuelle varer. Det lyder måske lidt uhyggeligt, men i virkeligheden er det yderst belejligt. Der sidder nemlig en sensor, der vil følge kunden rundt i butikken, så der ikke længere er behov for at slæbe den langs gulvet, eller gå med den, ubehageligt hængende på armen. Der befinder sig også en sensor, der er i stand til at kunne scanne varer, og i samarbejde med brugergrænsefladen, holde styr på indkøbslisten, uden kunden skal røre en finger. Herved opdateres listen over indkøbet også automatisk. Der sidder desuden en vægt i bunden af kurven, hvis funktion er at holde styr på ændring af varer i indkøbslisten. Dette gøres i samarbejde med produkt scanneren, for at holde styr på indkøbet. Her registreres der også, hvis noget skulle blive taget ud af Smart Wagon kurven.

Tankerne falder måske på Big Brother og dataudveksling. Snart vil alt indkøb være delt med alt og alle, og veganer farmor ville kunne se at skrabeæg bliver prioriteret, frem for de bæredygtige økologiske varianter. Men det er ikke situationen med Smart Wagon. Her ønskes blot at have fuldt fokus på kunden, dog kun under det aktuelle indkøb. Der vil altså ikke komme personlige tilbud eller forslag til aftensmaden. Der kræves ikke engang et login, blot en simpel kontakt mellem kunde og Smart Wagon, som kører over en sikker forbindelse. Det skal være sikkert, anonymt og trygt at købe ind.

Smart Wagon bidrager hermed til at give kunden en god, sikker og ikke mindst effektiv indkøbsoplevelse i de lokale supermarkeder. Hverdagen kan nu nydes som Dan Turèll beskriver den, til de lækre toner, komponeret af Halfdan E.

7 Projektformulering

I afsnit 7 vil der formuleres en beskrivelse af et 3. semesterprojekt, der beskriver det produkt der ønskes lavet. I projektet ønskes der udarbejdet en intelligent indkøbsvogn, kaldet *Smart Wagon*, som selvstændigt kan navigere rundt i et supermarked eller følge kunden rundt. Smart Wagon har funktionen at veje og scanne, og derved registrere produkter. Produktet registreres herefter af en computer, der behandler kundens indkøb løbende, som kan tilgås via en brugergrænseflade.

Smart Wagon skal gøre dagligdagen bedre for kunden, hvilket ønskes for at opnå et stadie, hvor Smart Wagon kan fungere som en smartere måde at håndtere kundens varer, og for at kunden automatisk kan følge med i sit køb. I en ideel verden ville Smart Wagon kende til kundens indkøbsliste, og skulle være i stand til at kunne guide kunden til placering af næste vare på indkøbssedlen. Denne funktion virker kun, når kunden følger indkøbslisten, da Smart wagon ikke kan tage højde for impulskøb. Heraf kan der integreres et andet system, der blot gør at Smart Wagon følger kunden. Dette gør, at kunden selv vil være i kontrol over indkøbet.

Smart Wagon skal kunne give plads til specialproducerede poser/net/tasker. Tasken skal kunne placeres i Smart Wagon. Dette er effektivt, da man blot kan samle tasken op til slut efter køb, og gå ud af butikken nemt og bekvemt.

Betaling via Smart Wagon gøres nemt og bekvemt ved, at der løbende holdes styr på antal og type af varer, ved hjælp af Smart Wagons centralsystem. Der vil også være en vægt, der kan holde styr på, om varerne bliver taget ud af Smart Wagon. Man kan da blot betale ved kassen, for at mindske ventetiden for betaling af indkøb.

For projektet kræves der, at det udviklede system skal interagere med omverden, have en brugergrænseflade, anvende en indlejret Linux platform og en PSoC platform, samt indeholde faglige elementer fra semesterets fag. Efter kravene er opfyldt, vil der være frie tøjler for projektets opbygning.

Der udvikles en grafisk brugergrænseflade, kaldet *GUI*, som skal være så intuitivt som muligt, så en kunde med mindst mulig viden om systemet, vil være i stand til at benytte Smart Wagon. Målet er, at flere forretninger vil kunne bruge Smart Wagon, og der vil kunne skabes en individuel GUI til den individuelle kundes butikskæde.

I en dagligdag, hvor der bruges meget tid i køer i dagens supermarkeder, vil Smart Wagon være behjælpelig med at begrænse tiden ved kassen, idet varerne allerede er scannet, og der er foretaget en automatisk debitering.

Projektet henvender sig hovedsageligt til kunder i større butikker, da der ikke er ligeså mange pladsproblemer, ift. en lidt mere kompakt butik, hvor der ikke er plads til Smart Wagon. Yderligere er der et langt større marked for en Smart Wagon i en stor butikskæde.

Betaversion

Projektet Smart Wagon udarbejdes med fokus på en betaversion, som skal indeholde de vigtigste elementer af Smart Wagon. Dermed vil denne version af Smart Wagon ikke indeholde det fulde system. Betaversionen er en version, der skal vise, hvad det givne produkt skal kunne i sin simpleste forstand. Her vil der være udviklet et system, der skal være i stand til at identificere forskellige varer, vise kunden, i form af en GUI, hvad der er aktuelt i indkøbet. Systemet skal kunne have en kørefunktion som viser, at der er bevægelse i forhold til indkøbet.

Identificering af varen vil være i form af en sensor, der ud fra en vægt, som måles i gram, i et forudbestemt interval, skal være i stand til at identificere, hvilken vare der detekteres. Der vil kun være 3 varer der kan blive identificeret, i intervallerne 1-300 (æble), 301-600 (smør) og 601-900 (rugbrød) målt i gram. Hvis vægten er uden for dette interval, vil intet produkt identificeres. Der detekteres kun 1 vare ad gangen.

For at få Smart Wagon til at kunne kører frem og stoppe, vil der være tilknyttet en DC-motor. Denne DC-motor styres af et PWM-signal, der kan bestemme om motoren er aktiv eller inaktiv. Motoren skal styres via GUI, og har ingen distancesensor. Den kan altså udelukkende kun styres af input fra RPi'en.

GUI'en for systemet er lavet som en hjemmeside, hvor der vil være forskellige funktioner som brugeren kan bruge, såsom "Start indkøb" og "Tilføj vare". GUI'en er sat op via en RPi, som er i stand til at

vise hjemmesider. RPi'en er hermed i stand til at kunne modtage data fra vægten, samt sende data til DC-motoren. Den kan behandle den data som fås fra vægten, til at sætte varen ind på indkøbslisten, og sende data der bestemmer hvordan motoren skal agere. RPi'en består deraf af tre dele, et hoved script, en hjemmeside og en serielforbindelse.

Skitsen af betaversionen af Smart Wagon, i figur 1, viser, at den fuldt implementerede Smart Wagon består af et køretøj, hvorpå der er en vægt og en kurv. I den udviklede betaversion vil GUI'en blive integreret på en computer.

Skitsen i figur 1 viser interaktionen og handlingsforløbet mellem kunden og Smart Wagon. Under indkøbsturen lægger kunden varer ned i Smart Wagon, hvorefter varetypen bestemmes ud fra varens vægt. Dette opdateres automatisk på GUI'ens indkøbskurv.

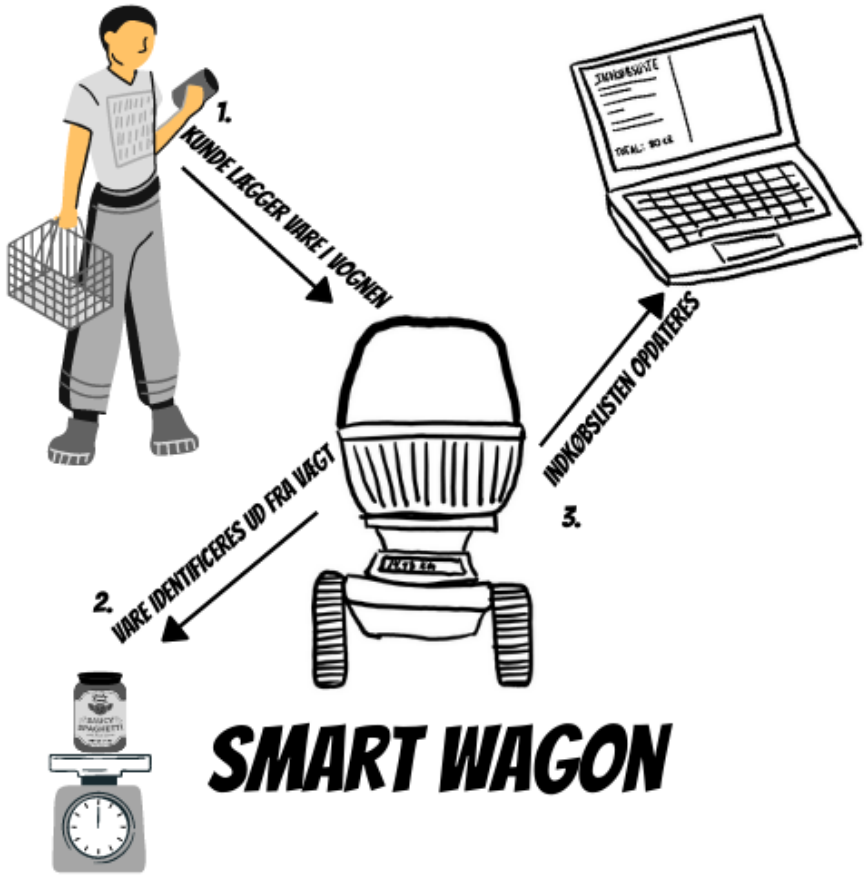

Figure 1: Det rige billede af Smart Wagon

8 Kravspecifikation

I dette afsnit forekommer systemspecifikation for Smart Wagons fulde version, 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 og FURPS analysemodellerne ud fra betaversionen.

8.1 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 2, beskriver interaktionen mellem systemets aktører.

Smart Wagon har kunden og personalet, som de to primære aktører. Grunden til at personalet optræder som en primær aktør, er for at dette personale igangsætter Use Case 4, som kan ses i figur 3, da personalet skal foretage fejlhåndtering på Smart Wagon.

Smart Wagon har en sekundær aktør, *Centralsystemet*, som bl.a. kan hjælpe den primære aktør i Use case 3, med at fjerne forskellige varer fra indkøbskurven.

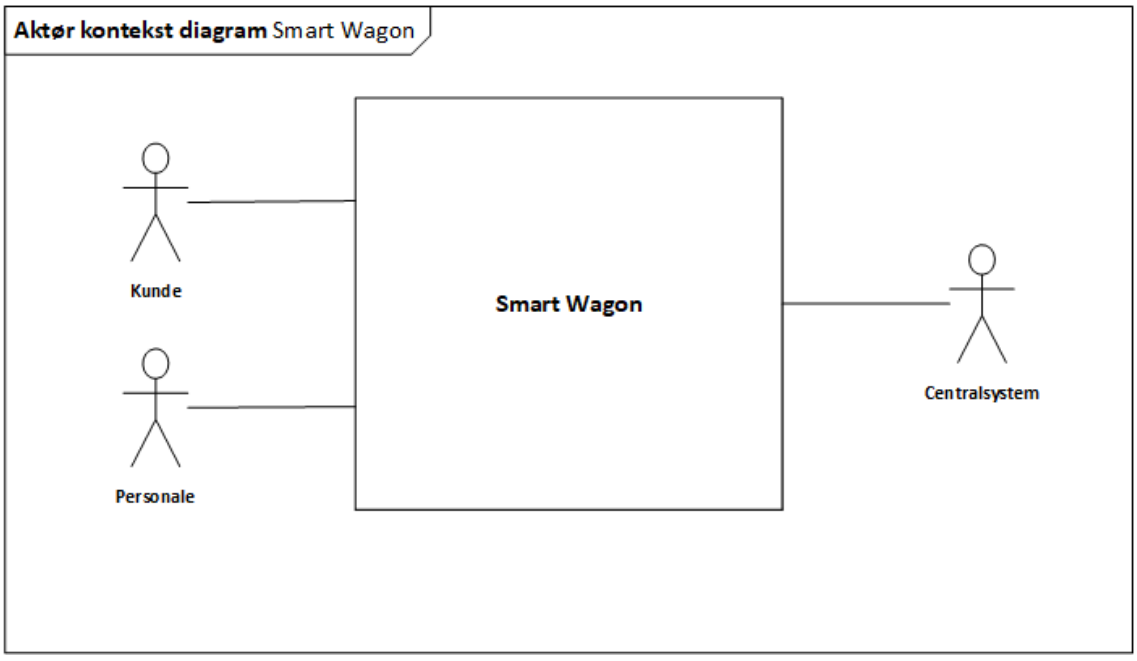

Figure 2: Aktør Kontekst Diagram

8.2 Funktionelle krav

Figur 3 viser et Use Case diagram for den fuldt 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, hvori der indgår to primære aktører og en sekundær aktør. Forbindelserne viser, hvilke aktører der har indflydelse på, hvilke Use Cases.

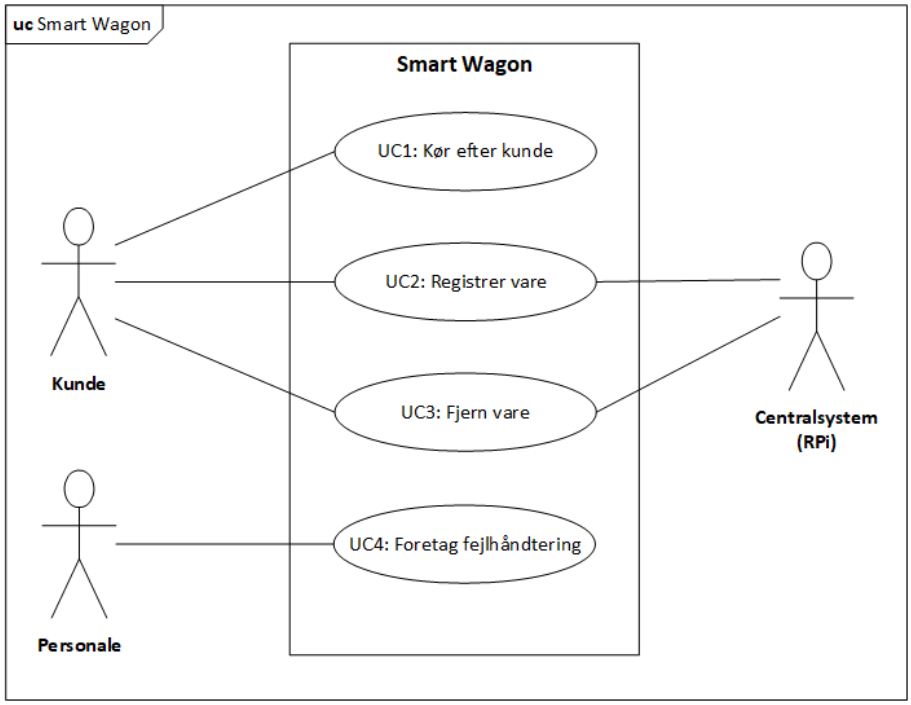

Figure 3: Use Case diagram for Smart Wagon

8.3 Fully dressed Use Case beskrivelser

I dette afsnit vil Smart Wagons Use Cases blive beskrevet, ved brug af fully dressed Use Case beskrivelser. Formålet med anvendelsen af denne model, er at danne en detaljeret beskrivelse af systemet og dets mulige mangler. 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.

Der vil igennem rapporten være specielt fokus på UC2: Registrer vare, da denne Use Case indholder de essentielle funktionaliteter for Smart Wagon. Heraf vil der kun medtages en fully dressed Use Case beskrivelse for UC2: Registrer vare. De resterende fully dressed Use Case beskrivelser kan findes i bilagsraporten under afsnittet for kravsspecifikation.

8.3.1 Use Case 1: Kør efter kunde

Use Case 1 beskriver Smart Wagons kørefunktion, 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. Se mere i bilagsraporten afsnit 1.3.1.

8.3.2 Use Case 2: Registrer vare

Use Case 2 beskriver hvordan Smart Wagon registrerer, 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 i overensstemmelse, således at varen kan registreres i Centralsystemet.
Initiering	Kunden scanner en vares stregkode med stregkodescanneren
Aktør	Primær: Kunde Sekundær: Centralsystem
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	1. Kunden scanner en vare 2. Smart Wagon registrerer varens stregkode 3. Centralsystemet 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 placer varen i kurven". [EXT. 2: Registrerer ingen vægt] 5.Smart Wagon registrerer om varens vægt passer med varens beskrivelse i centralsystemet [EXT. 3: Varens vægt stemmer ikke overens med systemet] 6. Smart Wagon tilføjer varen til kundens indkøbsliste 7. Smart Wagon er klar til at scanne en ny vare
Udvidelser/ undtagelser	EXT. 1: Kender ikke varen 1. Smart Wagon kan ikke koble stregkoden til en vare, der er registreret i centralsystemet. 2. Smart Wagon udskriver fejlbesked, "Fejl ved scanning af vare, prøv igen eller kontakt personale". 3. Hop til UC4: "Foretag fejlhåndtering" 4. UC2 afsluttes EXT. 2: Registrerer ingen vægt 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, placer varen i indkøbskurven eller kontakt personalet" 3. Hop til UC4: "Foretag fejlhåndtering" 4. UC2 afsluttes EXT. 3: Varens vægt stemmer ikke overens med systemet 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 4: Fully dressed Use Case beskrivelse for UC2

8.3.3 Use Case 3: Fjern vare

Use Case 3 beskriver, hvordan der kan fjernes en 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. Se mere i bilagsraporten afsnit 1.3.3.

8.3.4 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. Se mere i bilagsraporten afsnit 1.3.4.

Arbejdsprocessen af projektet for Smart Wagon har medført, at der udarbejdes en betaversion for prototypen af Smart Wagon, grundet Corona situationen, som uddybes yderligere i afsnit 12.7.

8.4 MoSCoW af betaversionen for de funktionelle krav

For at konkretisere systemets krav er der blevet udarbejdet en MoSCoW analyse. For 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 beskrives i dette afsnit for betaversionen.

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
- Smart Wagon skal beregne det totale beløb for varerne

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

8.5 Betaversionen af de ikke-funktionelle krav

Til udarbejdelse af betaversionen for de ikke-funktionelle krav, har projektgruppen valgt at anvende FURPS-metoden. FURPS-metoden består af inddelingerne: *functionality*, *usability*, *reliability*, *performance* og *supportability*.

Functionality:

- Smart Wagon kører via en DC-motor
- Smart Wagon registrerer varer 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 operative 83,33% af tiden, hvilket er udregnet med en MTBF analyse, som kan ses i bilagsrapporten under kravsspecifikationen afsnit 1.6.

Performance:

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

Supportability:

- Smart Wagon skal kunne genstartes af alle medarbejdere

8.6 MoSCoW af betaversionen for de ikke-funktionelle krav

De ikke-funktionelle krav for betaversionen prioriteres ved brug af MoSCoW analysen, 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 registrere varer 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

9 Afgrænsning

I afsnit 9 vil der beskrives, hvilke afgrænsninger der er dannet ift. kravspecifikationen, som skal specificere, hvilke områder af Smart Wagon, der medtages i implementationsfasen.

Der er på forhånd fra ingeniørhøjskolen, Aarhus Universitet blev stillet nogle fast bestemte krav, samt vurderinger for realisering af projektet som skal inkorporeres.

Ingeniørhøjskolens bestemte krav for 3. semesterprojekt er følgende:

- Projektet skal interagere 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 fastlægger fokuspunkter for projektet, på baggrund af ingeniørhøjskolens fastlagte krav samt bestemmelse af den generelle afgrænsning for Smart Wagon projektet, som beskrives herunder.

- Projektet interagerer med omverdenen via en sensor *Load Cell: 3-12V DC Digital Electronic Scale 1 Kg Weight Weighing Sensor Load Cell* og en aktuator *DC motor*. Aktuatoren er introduceret i GFV labøvelse *2: DC-motor control*, og sensoren er introduceret i labøvelse *4: Build a scale*.
- Projektet anvender en PSoC til at processere og videresende data henholdsvis mellem sensor og Centralsystemet, og mellem aktuator og Centralsystemet.
- 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 ekstern 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, HAL.

10 Metoder

Udviklingsprocessen af Smart Wagon projektet gennemføres på baggrund af ASE modellen, der ses i figur 4. ASE modellen er blevet introduceret på første og andet semester af Software Teknologi uddannelsen. Anvendelse af ASE modellen skal sørge for, at projektarbejdet består af vigtige faglige elementer, som danner et struktureret helhedsbillede af Smart Wagon.

Projektgruppen har fulgt ASE modellen ved samlet at udarbejde de grundlæggende projektdokumenter, såsom projektformulering, kravspecifikationen og arkitektur. Herefter har projektgruppen bevæget sig ind i den iterative arbejdsproces, som indeholder design og implementering for hardware og software moduler, samt forskellige itereringer af test.

Afslutningsvis består ASE modellens projektarbejde af at udføre integrationstest og accepttest i fællesskab med projektvejlederen.

10.1 ASE modellen

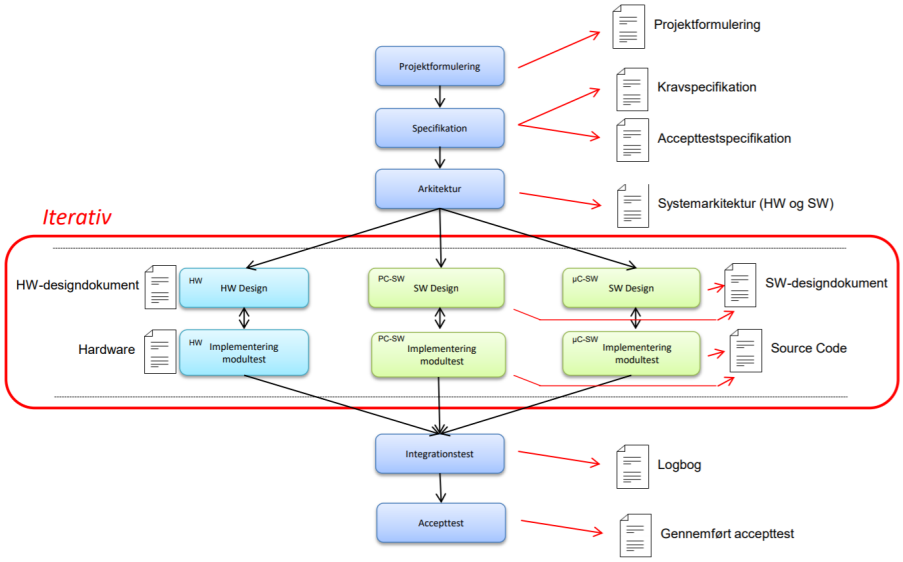

Figure 4: ASE modellen

Yderligere er Smart Wagon dannet på baggrund af den agile udviklingsproces, der illustreres i figur 5, hvor man udfører iterering af det forventede fulde produkt af Smart Wagon. Denne agile udviklingsproces er blevet anvendt for at kunne tilfredsstille butikskæder under delprocesserne af udviklingsprocessen. Betaversionen af Smart Wagon er et eksempel på, hvordan projektgruppen har designet og implementeret en version af Smart Wagon som indeholder de vigtigste dele af Smart Wagon, samt de mest basale principper. Heraf vil kunden kunne tilgå Smart Wagon via en brugergrænseflade, og efterfølgende vælge varer som kan placeres i handlekurven. Dertil, vil denne version af Smart Wagon ikke have den fulde implementering. Betaversionen vil indeholde de nødvendige funktionaliteter, for at butikskædens ønske kan opfyldes på længere sigt.

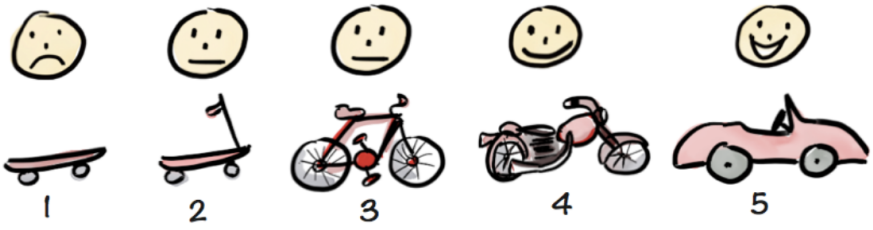

Figure 5: Den iterative udviklingsproces

Smart Wagon befinder sig i itereringsfase nr. 3, som illustreres af cyklen, på figur 5. Dette begrundes ved, at Smart Wagon indeholder diverse funktionaliteter, som giver kunden mulighed for den planlagte interaktion. Systemet er blevet samlet, således, at hardware modulerne for systemet kan analyseres og kontrolleres. Heraf gøres det muligt at bruge og processere den data som Centralsystemet (RPi) har brug for, for at kunne gøre systemet funktionelt. Yderligere er der opbygget en brugervenlig GUI, som giver kunden mulighed for at interagere med Smart Wagon, uden at have en dybere kendskab til baglandet for systemet, hvilket kan sammenlignes med cyklen. I dette tilfælde kan en given bruger nemt skifte gear, hvorimod det er de færreste, der har kendskab til det egentlige mekaniske skift.

10.2 Scrum

Projektgruppen anvender den agile metode scrum til at planlægge og strukturere projektarbejdet.

Figur 6 afbilder, hvordan en cyklus af scrum processen udarbejdes. Scrum cyklussen består af en fast defineret *product backlog*, som indeholder alle arbejdsopgaver, der danner delene af systemet Smart Wagon. Heraf tages de vigtigste opgaver fra product backloggen til sprint backloggen. Sprint backloggen indeholder opgaver der gennemføres til hver sprint, som typisk har en varighed på 2-4 uger. Derudover indeholder scrum daglige møder, hvor projektgruppens deltagere diskuterer arbejdsfremgang og kan revurdere arbejdsprocessen. Efterfølgende dannes der på baggrund af hver sprint, features, som dele af det færdige produkt, der delvist skal være klart til butikskæden.

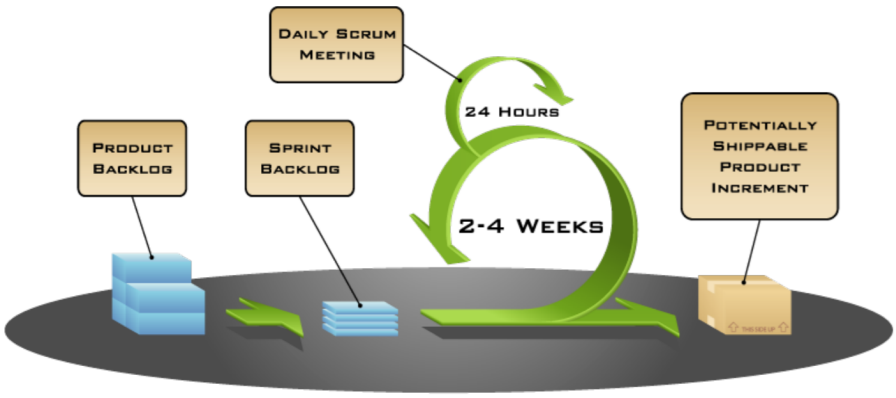

Figure 6: Scrum cyklus

Projektgruppen har udarbejdet en revurdering af scrum opbygningen, ved at anvende undervisernes forslag til udarbejdelsen af scrum, som har bestået i, at Johanne og Christina har fungeret som scrum masters, og ellers har alle projektgruppens deltager opfyldt rollen som scrum team.

Product backloggen er defineret ud fra Use Cases, som uddybes i afsnit 8, der beskriver kravspecifikationen. Funktionalitet af Use Casene danner backlog-items som er selve indholdet i product backloggen. De valgte backlog-items for hver sprint er udvalgt i samarbejde med projektvejlederen, som fungerer som produktejeren.

Til sammenligning med en alm. scrum arbejdsproces, har projektgruppen valgt at behandle 1-2 sprint-items under hvert sprint, som skal danne de færdige features. Hver scrum sprint har haft en varighed på 1 uge. Dertil er der blevet holdt ugentlige scrum møder, idet det ikke har været muligt at holde daglige scrum møder, som hænger godt sammen med at det er et skoleprojekt, og ikke et fuldtidsprojekt.

Projektgruppen har anvendt et Trello-board med stadige justeringer, som er dokumenteret i Logbogen, der kan findes i bilagsmappen. Dette Trello-board fungerer som det traditionelle scrum board, som gradvist indeholder backlog-items fra product backloggen og de udvalgte sprint-items i sprint backloggen.

I figur 7 ses en typisk opstilling af sprints-items samt enkelte backlog-items, som anvendes under de forskellige sprints.

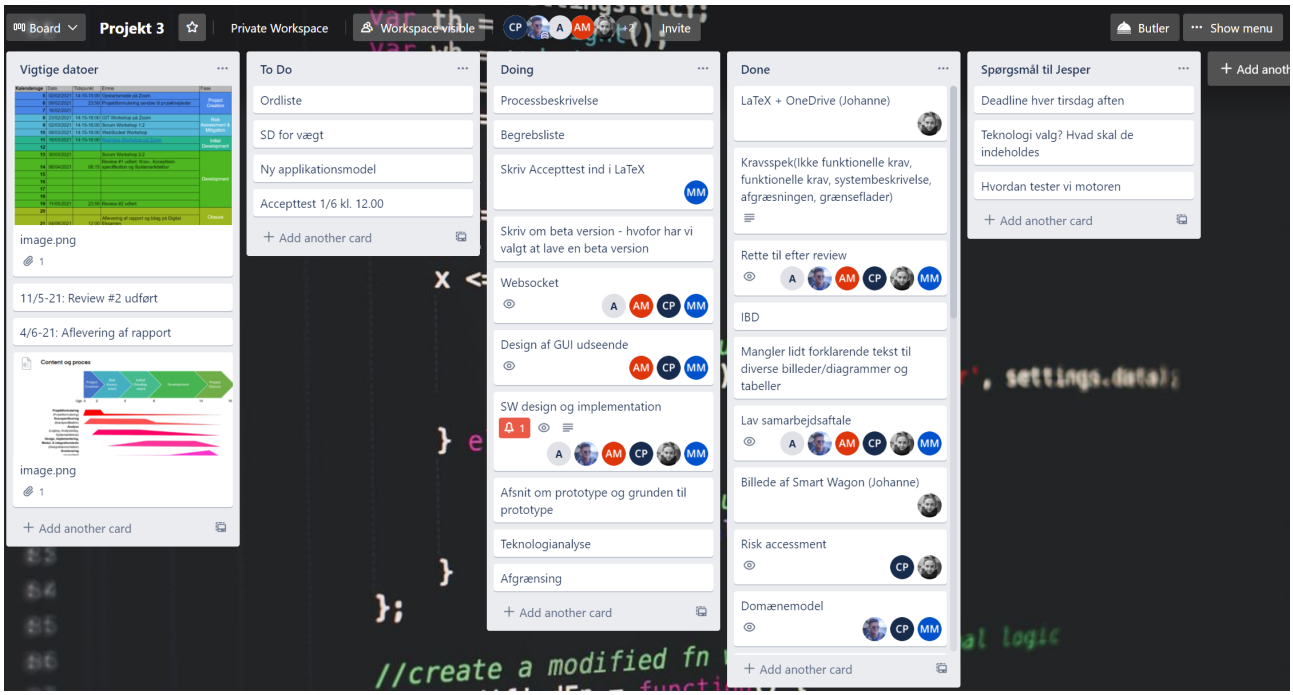

Figure 7: Trello board

Derudover, har scrum processen afviget fra den traditionelle scrum, ved at der har været et behov for, at revurdere tidligere backlog-items fra allerede afsluttede sprints. Dette skyldes, at udviklerne tilegner sig mere faglig viden i løbet af semesteret.

10.3 Versionsstyring af kode

Projektgruppen har valgt at anvende git som versionsstyringsværktøj, for at effektivt dele source kode af Smart Wagons moduler.

Git anvendes så en gruppe af udviklere kan samarbejde om at udvikle den samme software, til et system, på en effektiv måde. Heraf kan udviklerne hente de opdaterede ændringer ned til deres individuelle lokale repository, fra den delte repository. Dette er medvirkende til en struktureret implementeringsfase, som kommer til syne under projektarbejdet med Smart Wagon.

11 Udviklingsværktøjer

Her ses en oversigt over hvilke værktøjer, der er brugt i projektudviklingen af Smart Wagon.

Microsoft Visual Studio 2019/ Microsoft Visual Studio Code	Er brugt til udvikling af centralprocessor (RPi) hvoriblandt GUI indgår, og kommunikationen mellem RPi og PSoC.
Cypress PSoC Creator 4.3	Er brugt til udvikling af aktuator og sensor drivere, bruges til at programmere PSoC'en.
RealTerm: Serial Capture Program	Er brugt til integrationstest af PSoC drivere (motor og vægt)
VMware Workstation 15 Player	Er brugt til vores virtuelle indlejrede Linux system, hvor GUI og RPI kommunikerer igennem.
Microsoft OneDrive	Er brugt til deling af generelle filer blandt gruppens medlemmer, såsom billeder og UML diagrammer.
Git	Er brugt som værktøj til deling af source kode mellem udviklerne i projektet.
GitLab	Er en udbyder af git, som er brugt til deling af kode blandt gruppens medlemmer.
GitHub Desktop applikation	Er en git klient, som er brugt til at arbejde med projektgruppens repository fra GitLab.
Discord	Er brugt som kommunikationsværktøj til tale- og videoopkald, samt text chat.
Zoom	Er brugt som kommunikations værktøj til tale- og videoopkald under vejledermøder og review.
Facebook Messenger	Er brugt som kommunikationsværktøj til text chat blandt gruppens medlemmer.
Trello	Er brugt som værktøj til styring af scrum. Trello har givet mulighed for at opsætte en arbejdsplan med To-do, Doing og Done.
Online LaTeX Editor Overleaf	Er brugt til alt rapportskrivning.
Tables Generator	Er brugt til at generere scipts til tabelopbygning for LaTeX.
Microsoft Excel 365	Er brugt til at udarbejde tidsplanen.
Microsoft Visio Professional	Er brugt til at danne alle SysML og UML diagrammer.

Table 5: Oversigt over udviklingsværktøjer

12 Teknologianalyse

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

12.1 Valg af distancesensor 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 distancesensor 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 "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 distancesensor i betaversionen for Smart Wagon.

I bilagsrapporten, kan der ses en mere uddybende beskrivelse, samt billeder af de tre type sensorer der var til udsigt for Smart Wagon.

12.2 Valg af Centralsystem

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 CentralSystem, da den har mulighed for at oprette diverse forbindelser. En af disse forbindelser er en seriel forbindelse som f.eks. er blevet anvendt i udviklingen af Smart Wagons vægt og motor. Raspberry Pi er også i stand til at fortolke den data, som kommer ind fra Smart Wagons forskellige delsystemer, såsom den data der kommer fra vægten, eller den data der sendes til motoren, for at aktivere den. Yderligere har Raspberry Pi mulighed for at opbevare data.

12.2.1 Level Converter

Projektgruppen har ikke haft mulighed for at ændre på hardware processoren[1], grundet Corona. Dette har medført, at projektgruppen ikke har kunne lave en konvertering mellem 5V til 3.3V, og har hermed anvendt en Level Converter.

Ændringen har været nødvendig, idet PSoC'en som standard vil videregive 5V, mens RPi'en maksimalt kan håndtere 3.3V. Ved hjælp af den givne Level Converter, er det muligt at lave denne konvertering. Det vil sige, at der ved hjælp af denne, sendes 5V ind fra den ene side, og 3.3V i den anden. Herved kan de spændinger der transmitteres fra PSoC til RPi konverteres fra 5V til 3.3V, og omvendt, via de fire tilgængelige kanaler på Level Converteren. Dette er essentielt for, at kommunikationen kan finde sted.

12.3 Forbindelse imellem dataprocessoren og Centralsystemet

Projektgruppen gjorde sig en del overvejelser i forbindelse med, hvordan kommunikationen fra data processoren til Centralsystemet skal fungere.

Der var primært to løsninger, som gruppen overvejede:

- SPI driver
- Serielforbindelse

Ved samtale med en Lektor, som er tilknyttet faget HAL, blev der stillet spørgsmål om forbindelsen mellem PSoC og RPi, ift. SPI og en seriel forbindelse. Det blev gjort klart for gruppen, på baggrund af denne anbefaling, at en seriel forbindelse ville give en simplere implementeringsløsning. Fordelen ved seriel forbindelsen er simpliciteten, da RPi'en allerede har et integreret system til at kommunikere via en seriel forbindelse. Dette vil sige, at der ikke skulle skrives en SPI driver, der ellers var til overvejelse og blev forsøgt implementeret, dog uden held for funktionalitet. Seriel forbindelsen fungerer således, at der ved et UART modul på PSoC'en, kan sendes og modtages værdier mellem de to hardware enheder.

[1] PSoC

Der oprettes en interrupt rutine på PSoC'en, der gør det muligt at registrere input udefra, ved hjælp af denne omtalte UART. Kort sagt, kan der nemt og bekvemt sendes data via en Rx og Tx forbindelse, der kommunikerer mellem PSoC og RPi.

12.4 Valg af CPU til hardware delsystemer

Smart Wagon bygger på hardwaremoduler, såsom en sensor i form af en vægt, samt en aktuator i form af en DC-motor. Disse systemer kan Centralsystemet (RPi), 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 hardwaremodulerne. PSoC'en er her i stand til at behandle den data og videresende den til Centralsystemet (RPi).

12.4.1 Vægt

For at Smart Wagon skal kunne registrere, hvilken vare der tilføjes til indkøbskurven, er der gjort brug af en sensor, i form af en vægt, som forbindes til PSoC'en. PSoC'en behandler den data, i form af spænding, som kommer ind fra vægten. Heraf vil PSoC'en konvertere dataen til gram, hvilket kan bruges til at identificere hvilket produkt der er registreret.

Der gøres opmærksom på, at vægten ikke kan identificere flere produkter på en gang. Dette medfører en begrænsning til udførelsen af betaversionen, om at der kun kan være én vare i indkøbskurven ad gangen. Hvis vægten måler en spænding, der samlet svarer til 400 gram (smør), hvor der faktisk er 2 produkter af 200 gram (æbler), samtidig på vægten, vil dette blive registreret som én vare af 400 gram (smør).

Vægten har et off-set, på et interval mellem 800-810 mV. Dette betyder, at der korrigeres således, at der vil måles 0 gram ved 800 mV. Vægt konverteringen forgår således at per 1 mV, vil der måles en vægt på 0.8 gram. Vægten er valgt til betaversionen, da faget GFV har skabt basis kendskab til implementering af en vægt af denne type.

12.4.2 DC-motor

For at Smart Wagon skal være i stand til at bevæge sig rundt, og i fremtiden burde kunne køre efter kunden, har der været brug for en DC-motor. Denne motor skal kunne aktiveres på kommando. Denne kommando benytter en forbindelse mellem Centralsystem og motor via PSoC, der processerer den data, som motoren bruger. Det er altså muligt at sende information, et logisk '1' eller '0', fra RPi til PSoC, som via motorens switch case får motoren til at køre frem eller stoppe. Dette giver også mulighed for videreudvikling af systemet, da DC-motoren har mulighed for at skifte hastighed og retning, grundet PWM. Heraf skulle der sendes yderligere data, udover de logiske værdier.

Motoren kører via et PWM-signal, der via pins bliver aktiveret ved et signal på enten logisk '1' eller logisk '0'. Disse pins bestemmer, om det er muligt at sende data fra PSoC til L298 H-bro. H-broen aktiverer motoren, hvor motorens rotering afhænger af, hvilken vej strømmen løber gennem H-broen.

I projektet er der anvendt en DC-motor, hvor valget ellers kunne have været en stepper-motor. Grunden til at projektgruppen har valgt en DC-motor, er at betaversionen kun skal opfylde kravene om, at motoren skal kunne køre frem og stoppe. Her vurderes det, at DC-motoren er den mest ideelle, da en DC-motor oftest bruges til styrring af hjul, såsom en radiobil, hvorimod en Stepper-motor oftest har til funktion at kunne lave præcisionsarbejde, såsom f.eks. at stoppe ved en bestemt vinkel når man arbejder med robotarme.

12.5 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 mobilt skal kunne tilgå Smart Wagon.

Projektgruppens primære overvejelser i forbindelse med, hvordan brugergrænsefladen skulle være tilgængelig for kunden ses herunder.

- En smart phone applikation

- En hjemmeside
- En touchskærm monteret på Smart Wagon

Det er i projektgruppen blevet besluttet at implementere en brugergrænseflade via en hjemmeside ved brug af HTML, JavaScript og Cascading Style Sheets (CSS). Dette valg skyldes tankegangen om den agile udviklingsproces, som bliver illustreret i figur 5, hvor der er fokus på at en proces gennemgår forskellige udviklingsstadier, før den endelige version står færdig.

Projektgruppen vurderer implementeringen af brugergrænseflade via en hjemmeside som det første stadie i udviklingsprocessen, hvor implementering på en applikation eller en touchskræm ville være et mål for et færdigudviklet slutprodukt af Smart Wagon.

12.5.1 Valg af Raspberry Pi som host for hjemmesiden

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

12.6 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 kompendiet "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. Denne minimumsmodel ses implementeret i bilagsrapporten, tabel 8. Heraf vurderes sandsynlighed og konsekvens ved foranstaltningerne ud fra 3 scenarier.
Sandsynligheden S bedømmes ud fra niveau 1-3:

- 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 bedømmes ud fra niveau 1,3 eller 10:

- 1 = Konsekvensen kan løses med planens nuværende aktiveter
- 3 = Der skal gøres noget for at afbøde konsekvensen
- 10 = Planen skrider hvis dette sker

Produktet P af sandsynligheden og konsekvensen beregnes med udregningen $S \cdot K$

I den følgende tabel udtrykkes *risikomodellen* 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. Opfylde 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 scrummøder 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 de essentielle delsystemer ikke vil kunne udføre deres opgave.

12.7 Projektplan

I risikoanalysen er projektplanen blevet revurderet på grund af den igangværende Covid-19 situation.

Det er i den forbindelse blevet besluttet, at slutproduktet for Smart Wagon bliver nedskaleret. Det er blevet oplyst af Embedded Stock, at der på dette semester ikke vil være muligt at få udleveret HW komponenter. Det vil sige, det ikke er muligt for gruppen af få udleveret den bil med larvefødder, 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 betaversionen ved en integrationstest. Beslutningen er taget for at muliggøre krav og realisere accepttesten af betaversionen for projektet.

Bortset fra ovenstående revurdering af projektplan har projektets forløb været som planlagt.

13 Systemarkitektur

I dette afsnit vil Smart Wagons systemarkitektur blive redegjort i form af et BDD-diagram, et deployment view, en domænemodel og fire forskellige sekvensdiagrammer med handling. Disse redskaber bruges til at specificere og præcisere Smart Wagons hardware- og software-interface.

13.1 Block Definition Diagram

Block-diagrammet over Smart Wagon i figur 8 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 henholdsvis er en distancesensor og en vægt.

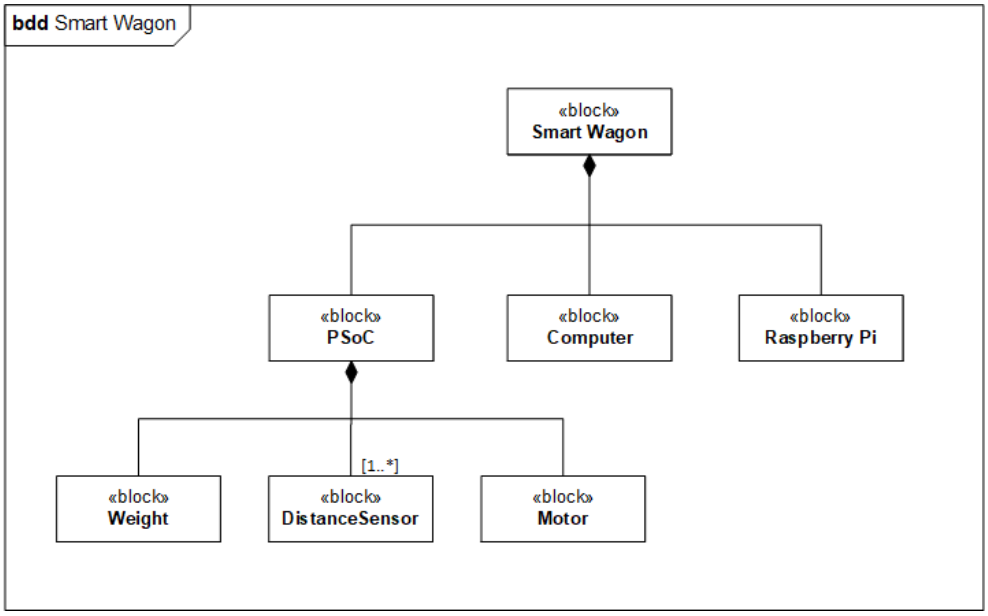

Figure 8: BDD for Smart Wagon

13.2 Deployment View

Ud fra BDD'et i figur 8 og IBD'et i bilagsrapporten er der udarbejdet et deployment view, som kan ses i figur 9. Deployment viewet viser Smart Wagons interne blokforbindelser, og hvor der er indsat software.

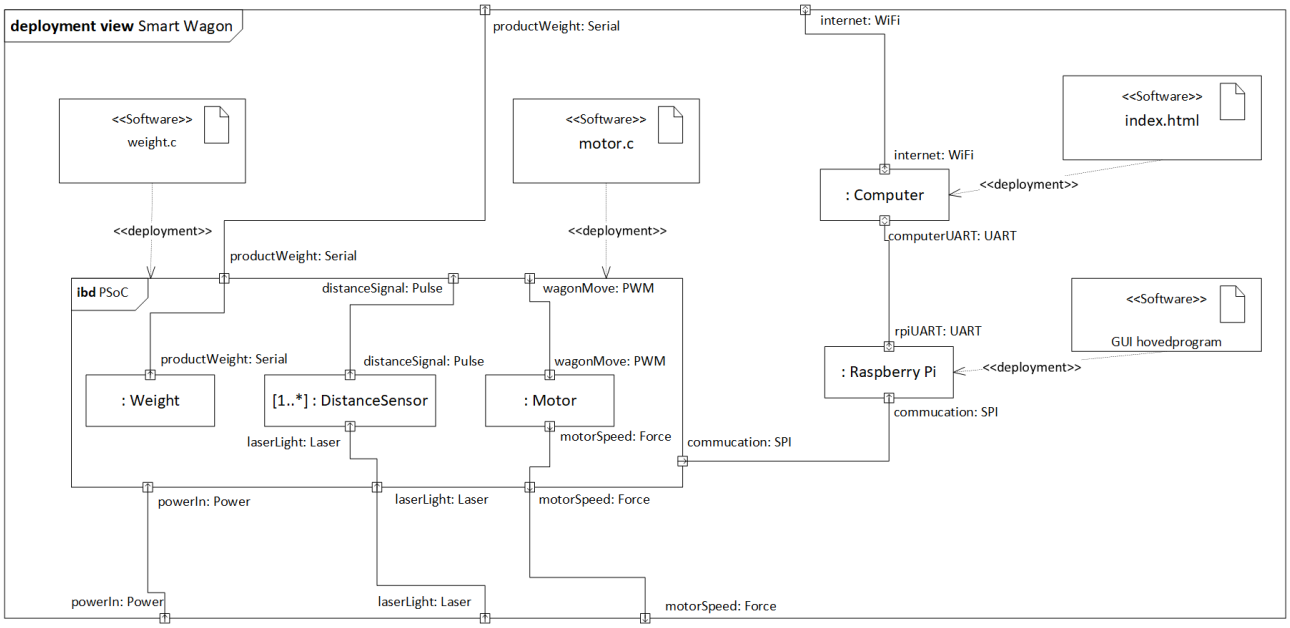

Figure 9: Deployment view af Smart Wagon

13.3 Domænemodel

For at få udarbejdet domænemodellen i figur 10 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.

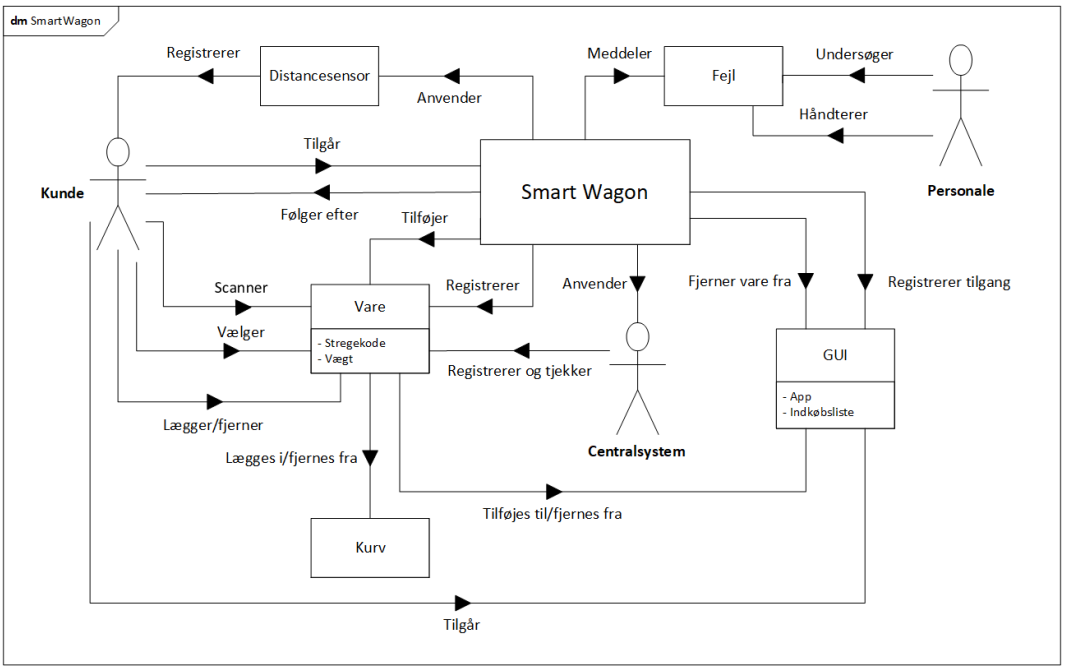

Figure 10: Domænemodel for Smart Wagon

13.4 Sekvensdiagrammer for handling

Der er blevet udarbejdet fire sekvensdiagrammer som beskriver Smart Wagons adfærd i form af handlinger ud fra de fire udformede Use Case diagrammer som er blevet præsenteret i afsnit 8, Kravspecifikation.

Idet der igennem rapporten har været fokus på UC2: Register vare vil det også kun være dette sekvensdiagram, som vil blive præsenteret i afsnittet. De resterende tre sekvensdiagrammer kan findes i bilagsrapporten under afsnittet for systemarkitetur.

13.4.1 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.

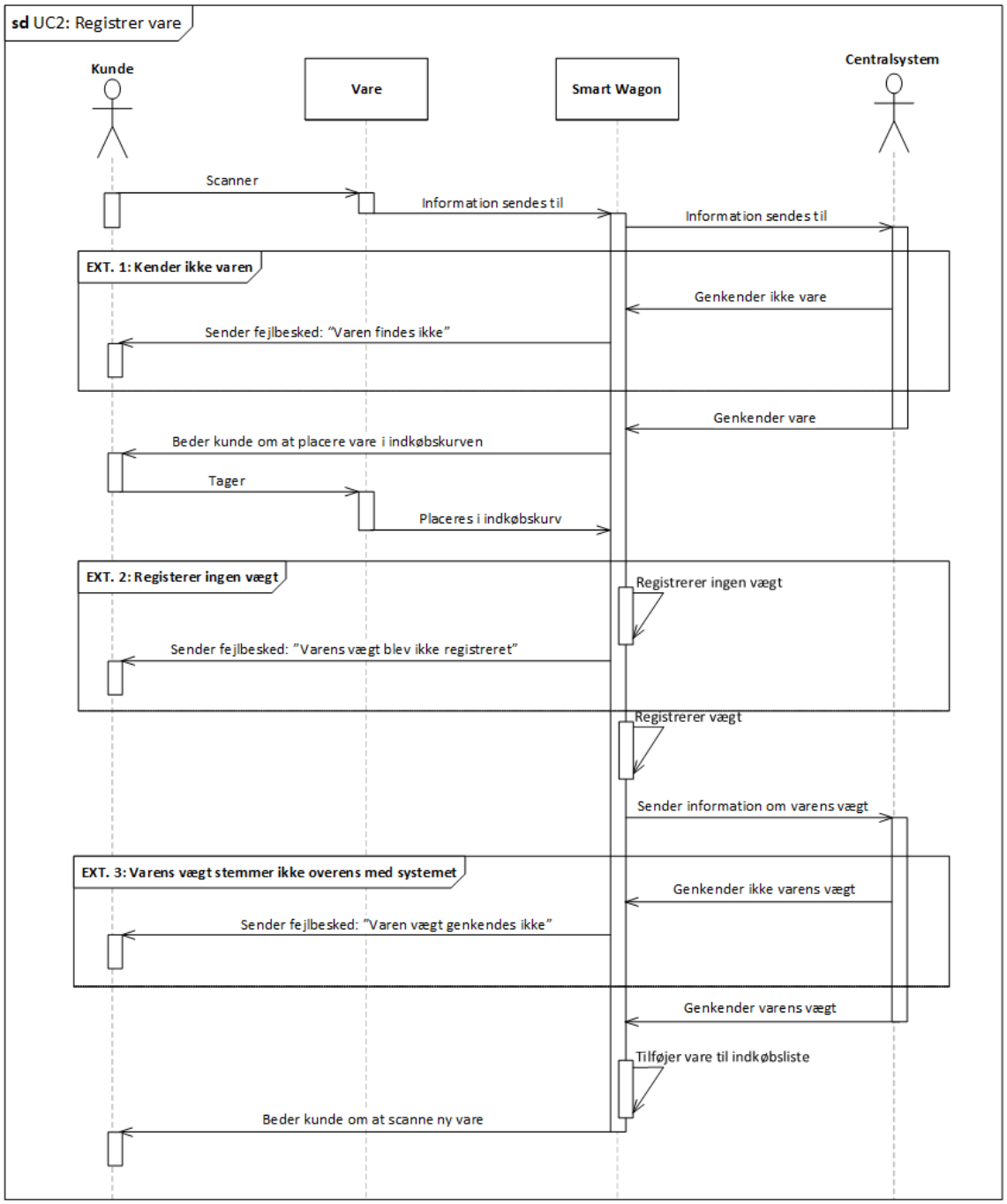

Figure 11: Sekvensdiagram for UC2: Registrer vare

14 Design

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

14.1 Hardware design

I dette afsnit vil Smart Wagons hardware design blive præsenteret. Dette vil gøres ved brug af en teoretisk opstilling og en fysisk opstilling af motoren og vægten, samt 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 mellem de forskellige hardware blokke, som kan ses i figur 12. Figurens formål er at give et overskueligt overblik over de forskellige forbindelser, så når der foretages forskellige test, vil det være nemmere for gruppens medlemmer at debugge systemet hvis fejl skulle opstå.

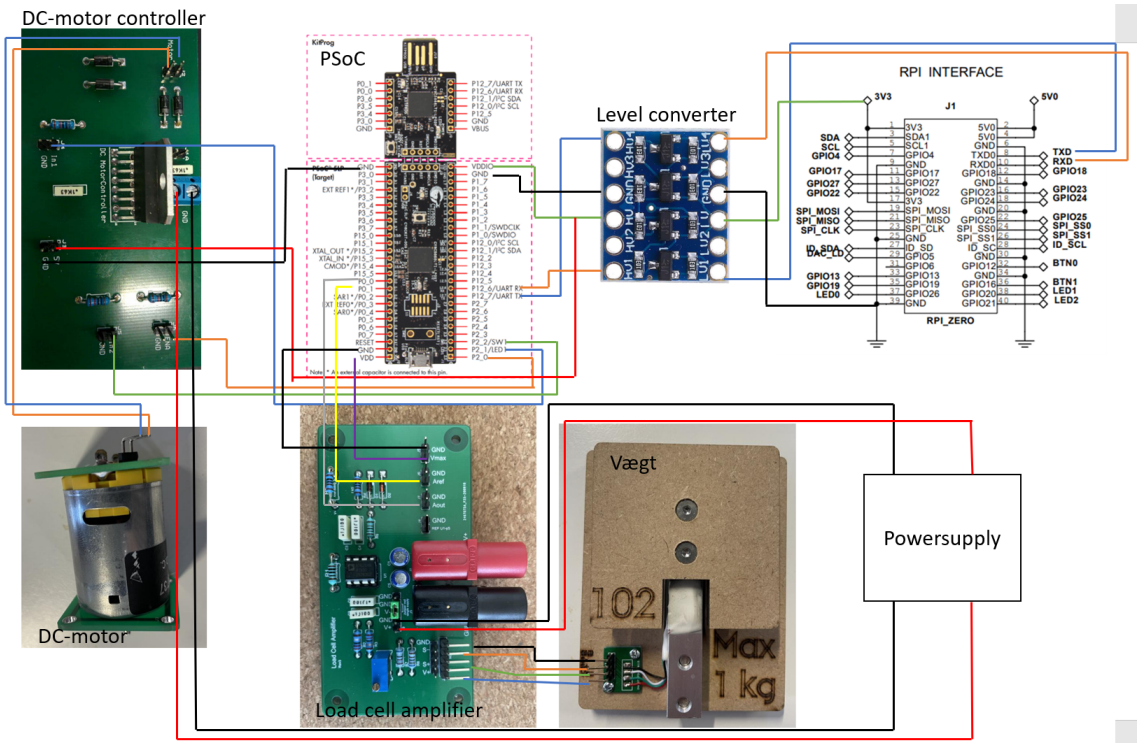

Figure 12: Fysisk hardware oversigt

Der er udarbejdet en tabel, som viser pin-forbindelserne mellem hardware komponenterne, i bilagsrapporten, *afsnit 10.1: Hardware design*. Denne tabel giver et præcist overblik over, hvilke forbindelser der skal laves for at få et operativt system for Smart Wagons betaversion.

14.1.1 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. Når 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 påvirking af strain gauge, giver større spænding, hvilket videresendes gennem *Load Cell Amplifieren*. Det vil sige, at designet består af en sensor, en *Load Cell Amplifier* og en PSoC.

14.1.2 Design af motor

Ved design af motoren, er der blevet gjort brug af en DC-motor, en H-bro og en PSoC. Motoren styres via et PWM signal, som driver motoren og giver mulighed for at kontrollere omdrejningshastigheden. PSoC'en skaber dette PWM signal, som sendes ud på to pins. Disse pins bestemmer, hvilken retning strømmen løber igennem H-broen. Heraf kan motorens retning styres via en H-bro.

14.1.3 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 imellem PSoC og RPi.

PSoC og RPi gør brug af en seriel forbindelse, således at der bliver lavet en krydsforbindelse, mellem Tx[2] og Rx[3] pins på henholdsvis PSoC og RPi. Der kan da sendes data begge veje, hvilket resulterer i, at RPi kan modtage data fra vægten, samt sende data til PSoC, der kan bruges til at aktivere DC-motoren.

Der bliver i forbindelse med seriel forbindelsen anvendt en Level Converter som skal sikre, at der kun bliver sendt 3.3V signaler til RPi'en, fra PSoC'ens 5V signaler for, at der ikke skabes overload, og RPi'en brænder sammen. Der kan læses mere om Level Converteren i afsnittet der omhandler teknologianalysen.

[2] Transmitter
[3] Receiver

14.2 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.

14.2.1 Applikationsmodel baseret på beta modellen af SmartWagon

I dette afsnit vil den udarbejdede applikationsmodel for Smart Wagons betaversion blive præsenteret. Det gøres i form af et udarbejdet klassediagram og sekvensdiagram med metoder for UC2: Registrer vare. Applikationsmodellerne for de resterende Use Cases ses i bilagsrapporten, i afsnit *Revurdering af applikationsmodellen for betaversionen af Smart Wagon.*

UML klassediagram for UC2: Registrer vare

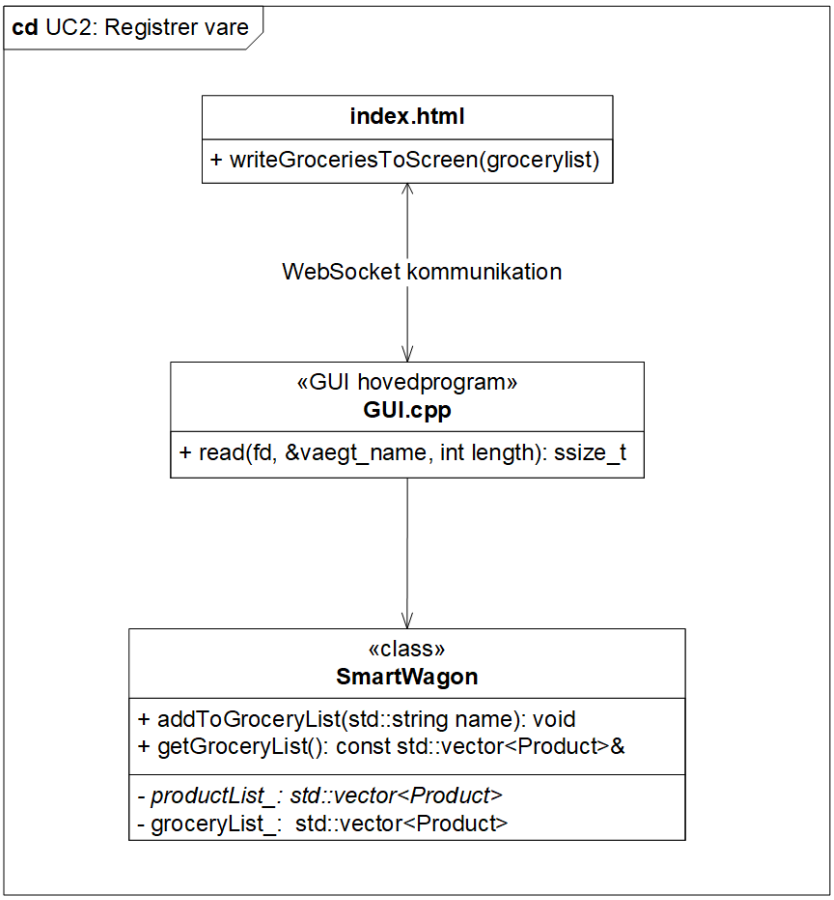

Figure 13: UML klassediagram for UC2: Registrer vare

Sekvensdiagram med metoder for UC2: Registrer vare

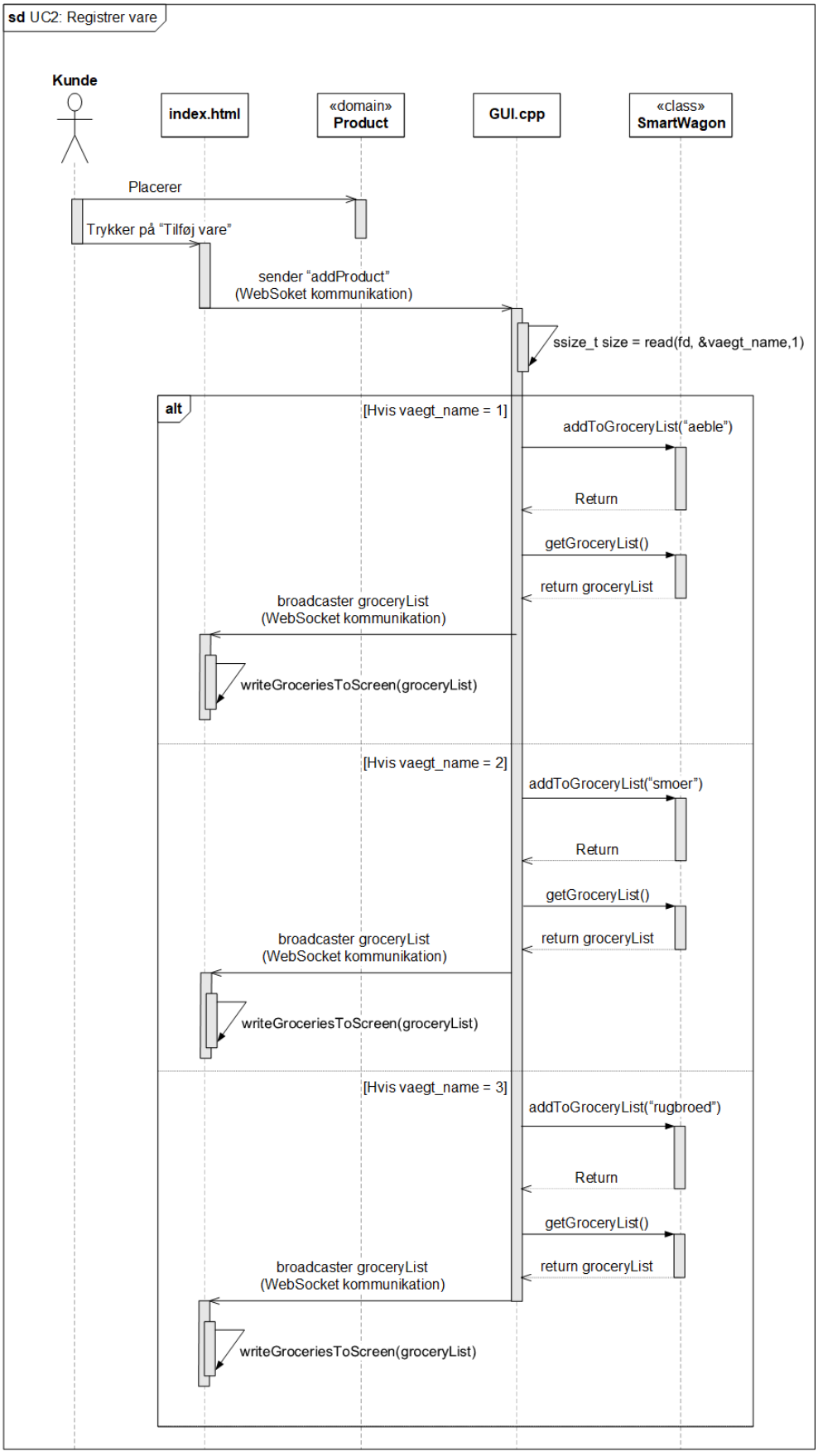

Figure 14: Sekvensdiagram med metoder for UC2: Registrer vare

15 Implementering

15.1 Hardware implementering

I dette afsnit vil hardware implementeringerne for Smart Wagons delsystem blive beskrevet

15.1.1 Implementering af motor

Til implementeringen af motoren er der taget udgangspunkt i den udarbejdede motor fra faget GFV [4] laboratorieøvelse kaldet *DC 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 kan styre motorens hastighed ved, at PWM signalets duty-cycle ændres. Motoren modtager et signal via en L290 H-bro, som sendes fra PSoC'en der kan styre, hvilken vej signalet går igennem DC-motoren, og hvilket PWM signal, som frekvensen kører med.

På UART modulet er der en interrupt rutine, ved navn isr_uart_rx, som er tilknyttet rx_interrupt delen på UART'en. Denne interrupt rutine har til formål at registrere, om der kommer et input. Dette bliver brugt i forbindelse med DC-motoren, hvor et tast på tastaturet vil sende en besked til UART'en, der aktiverer den givne interrupt rutine. Dette medfører en handling, der igangsætter et event, som forårsager at DC-motoren starter eller stopper.

15.1.2 Implementering af vægt

Til implementering af vægten er der taget udgangspunkt i den udarbejdede vægt fra laboratorieøvelse kaldet *Build a scale and determine how good it is*, i faget GFV. Denne øvelse er der blevet udarbejdet en vægt, som er i stand til at foretage A/D konvertering. Det vil sige, at vægten kan oversætte analoge input om til digitale output. Dette gør, at PSoC'en kan arbejde med en modtagende spænding, som modtages fra den givne sensor (vægten). Den modtagende data, der måles fra vægten, omdannes til en vægt målt i gram.

For at øge vægtens præcision, er der blevet foretaget en række målinger, hvor først vægtens linearitet er blevet undersøgt, hvorefter vægtens ikke-lineære område er blevet analyseret, og til sidst er vægtens sensitivitet blevet undersøgt. Der måles et off-set på 800-810 mV, og der er fundet en nøjagtighed, for 95 procent konfidensniveau, som er på +/-1.95 gram.

I bilagsrapporten, under afsnittet implementering, ses et billede af TopDesign for vægten, der er blevet opstillet i forbindelse med hardware modulerne på PSoC'en. Denne hardware fortæller PSoC'en, hvilke hardware moduler den arbejder med. Der er oprettet to hardware moduler, en UART og en ADC_SAR. UART modulet har to funktioner, hvoraf den ene del anvendes til integrationstest, og den anden anvendes til seriel forbindelsen.

Testdelen går ud på at kunne kommunikere direkte med PSoC'en, som er blevet gjort via konsolapplikationen 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.

Seriel forbindelsen, ift. vægt, anvendes for at skabe 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 til formål at modtage analog data og konvertere det til digtal data, som derefter kan behandles af PSoC'en. Denne analoge data sendes ind til PSoC'en via pin Pin_AC_in. Det er denne spænding, som software programmet kan behandle.

[4] Grænseflader til den fysiske verden

15.2 Software implementering

I dette afsnit vil software implementeringerne for Smart Wagons delsystemer blive beskrevet.

15.2.1 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*, der 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 15, der 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 af GUI'ens udseende. SmartWagon-klassen indeholder metoder, som indkapsler funktionalitet omhandlende tilføjelse og fjernelse af varer i Smart Wagons indkøbskurv, samt beregning af handleturens totale sum.

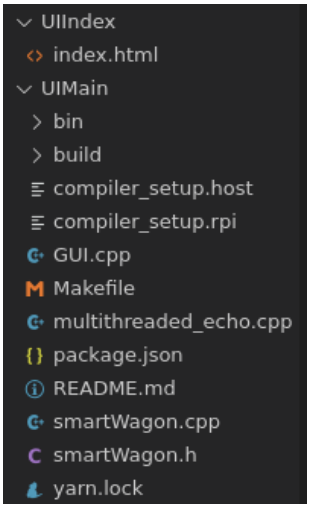

Figure 15: Udklip af filer fra repository

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 samt information af karakteren #, som danner linjeskiftene i udskriften på GUI'en.

GUI hovedprogrammet indeholder en main()-funktion, hvor oprettelse af alle produkterne i SmartWagons indbyggede 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.

I figur 16 ses den faktiske udskrift af indkøbskurvens indhold, som opdateres automatisk med både titel og pris, samt en foreløbig beløb og efterfølgende de resulterede beløb.

Figure 16: GUI for SmartWagon

15.2.2 Motor implementering

Motoren er blevet implementeret med en ISR[5], 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.

15.2.3 Implementering af vægt

Implementeringen af vægten er baseret på baggrund af en sensor, som måler en spænding ved hjælp af en ADC SAR[6]. SAR tjekker om der er noget at måle, hvis ja, vil den givne spænding konverteres, fra analog til digital. Vægten har et offset på 800-810 mV. PSoC'en omregner den målte spænding til en vægt, målt i gram. Dette sker via funktionskaldet "voltToGram". Funktionen "voltToGram" adderer 0.8 gram pr. målt spænding, som medfører at der vil være en vægt på 240 gram ved en måling 300 mV uden off-set.

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 genstand pr. sekund, eller ligende, vil vægten kunne være påvirket af udefrakommende forstyrrelser.

Efter målingen bliver der tjekket om der er blevet målt en ny vægt. For at der kan 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, hvilken vare der er på vægten, ud fra hvilket vægtinterval varen ligger i. Heraf vil

[5] interrupt service rutine
[6] Successive Approximation Register

funktionen ”setProduct” blive kaldt. Betaversionen indeholder tre varer, der ligger i intervallerne 1-300 (æbler, numeriske ASCII værdi: ’1’), 301-600 (smør, numeriske ASCII værdi: ’2’) og 601-900 (rugbrød, numeriske ASCII værdi: ’3’) gram. Når varen er registreret, vil den tilhørende ASCII værdi blive sat klar til transmittering.

Herefter vil denne information sendes fra PSoC’en til RPi’en, via en UART seriel forbindelse. Dette sker via transmission af data, ved hjælp af en Tx til Rx forbindelse. Denne transmission opstår samtidigt med registrering af varen, som er hvert 0.25 sekund. Dette er en yderligere grund til tidsintervallet, nævnt tidligere, da den data der skal sendes til RPi’en, sendes ofte nok til at kunne blive registreret. Produkternes ASCII værdier som sendes via transmissionen, modtages af GUI hovedprogrammet via WebSocket kommunikation. Dataen bliver sendt via UART funktionen *PutString()*. Hernæst gentages hele rutinen, så systemet igen vil transmittere data og tjekke efter vægt, og dermed også efter ny vare.

16 Accepttest for betaversionen

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.

16.1 Accepttest for UC2: 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 7: Accepttest for UC2: Registrer vare

16.2 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 de opstillede handlinger og forventede observationer passede med de faktiske observationer.

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

17 Konklusion

Projektet er startet med en ambition, om at gøre hverdagen for den gængse dansker nemmere. Heraf er der udarbejdet en betaversion af en smart indkøbskurv, som kan registrere hvilke varer, der bliver lagt i indkøbskurven.

Projektgruppen kan konkludere, at de er nået i mål med deres ønskede version for prototypen. Gruppen har satset på at opfylde alle de krav, der blev stillet af Aarhus Universitet til dette års semesterprojekt. Derved er der implementeret hardware og software til en aktuator i form af motorstyring, der kan få en DC-motor til at køre og stoppe brat. Ligeledes er der også udarbejdet hardware og software til implementeringen af en sensor i form af en vægt, der kan registrere hvilken af de tre testvarer der bliver sat på den, samt sende denne besked videre til det udarbejdede webbaserede GUI, der herefter tilføjer varen til kundens indkøbsliste.

Smart Wagons delsystemer er blevet udarbejdet med fokus på indlæring. Heraf er udviklingsprocessen udarbejdet i mindre grupper, således at projektgruppen har fået indsigt i en ingeniørmæssig arbejdsproces. Denne proces er udført på baggrund af fagene, der er blevet undervist i, i løbet af 3. semester. Fagene HAL og ISU har givet projektgruppen gode erfaringer med at arbejde med Linux platformen, og arbejde med RPi og drivere ud fra terminalkald, samt arbejdet med RPi'ens hjemmeside. GFV har lært gruppen at arbejde med PSoC, som er en anden form for microcontroller som supplement af RPi, hvilket gruppen har brugt en del i projektet, da alle drivere til motor og vægt er lavet og styret af PSoC. Denne proces har alle bidraget til, som gør, at selvom systemdesign og udførsel er blevet lavet i mindre grupper, har alle fået en samlet forståelse af, hvad systemet i helhed kan og skal.

Under udviklingsprocessen har der været et godt samarbejde blandt projektgruppens medlemmer, som har bygget på samarbejde og respekt. Projektgruppen har været gode til at snakke sammen og tilrettelægge projektarbejdet ved bl.a. at anvende den agile metode scrum.

Afslutningsvis, kan det konkluderes, at projektgruppen har formået at danne et tilfredsstillende produkt, som understøtter de fundamentale funktioner af Smart Wagon. Hertil kan der videreudvikles smartere funktioner, som opfylder de resterende krav, som beskrevet for den fulde version af Smart Wagon. Dette omtales nærmere i afsnit 18, der omhandler det fremtidige arbejde med Smart Wagon.

18 Fremtidig arbejde

Videreudvikling af systemet Smart Wagon vil bestå af færdiggørelse af de planlagte opgaver i den iterative arbejdsproces, som illustreres i figur 5. Heraf vil et begyndende skridt i udviklingsfasen være at implementere en fysisk indkøbskurv og en vægt som har en større tolerance end 1 kg. Herudover, skal Smart Wagon opretholde designkonceptet med, at der kan tilføjes og fjernes varer, hvor der allerede er varer i indkøbskurven. Hertil er det en forudsætning, at der implementeres en *tara* funktion til vægten, som er introduceret i GFV undervisningen. Derudover ønskes en Smart Wagon iterering, som skal tilbyde kunden at afbryde købet, hvis kunden ikke ønsker at fortsætte handleturen.

Efterfølgende videreudvikling vil indebære implementering af en distancesensor, stregkodescanner, drejefunktion, samt larvefødder. Udvikling af distancesensoren vil gøre det muligt for Smart Wagon at følge efter kunden under handleturen, som er handlingen for *Use Case 1: Kør efter kunde*. Udviklingen af drejefunktionen kan være en tilføjelse, der påbegyndes efter projektgruppen har udviklet en succesfuld distancesensor, som kan bidrage med en smidig kørsel under handleturen. Implementering af en stregkodescanner vil tillade en bedre præcision for registrering af varer i systemet, så der kan tilføjes flere forskellige varer en de tre der tages med i betaversionen. Itereringsprocessen af Smart Wagon kan afslutningsvis indeholde udvikling af en smartphone applikation, som vil danne en ekstern brugergrænseflade til Smart Wagon.

Afslutningsvis har projektgruppen dannet et vellykket projekt ift. de forudsatte projektkrav ved, at projektgruppen har dannet moduler, der opfylder krav om en brugergrænseflade, samt brug af en sensor og en aktuator. Projektarbejdet har medbragt god supplement til undervisningen på 3. semester, som har givet en bredere faglig forståelse af de ingeniørmæssige fag.

19 Opnået erfaringer

Projektet har givet en bedre forståelse for de fag som er blevet brugt i løbet af 3. semester. Ved brug af vægt og DC-motor, processeret af PSoC, har vi fået en bedre forståelse af faget GFV. Dette skyldes, at brugen af dem er blevet lavet om, således at systemet bygges op omkring dén brug, der kræves for Smart Wagon. Dette ses f.eks. ift. brug af vægt, hvor diverse produkter skal kunne identificeres ud fra den målte spænding. Herefter har projektgruppen videreudviklet en Tx/Rx seriel forbindelse mellem PSoC og RPi.

I projektets udarbejdelse har der været bølger af fremgang og modgang. Allerede tidligt oplevede vi problemer med hardware, da det ønskede behov for hardware, således at produktet, Smart Wagon, skulle kunne være i stand til at kunne bevæge sig, ikke var muligt at få fat på. Problemet opstod grundet Corona, hvor der ikke var mulighed for at få udleveret det ønskede hardware. Efterfølgende har projektgruppen opnået erfaring om hvor svært det er at opbygge det fulde system, uden at have alle de nødvendige komponenter, specielt da systemet skulle være bygget op omkring den hardware der ikke kunne udleveres. Løsningen har da været at opbygge produktet i dele, da der ikke har været mulighed for at samle alle delene.

Løbende i processen, har der været fokus på systemudvikling af betaversionen og udarbejdelse af SysML og UML diagrammer. Til start var fokuspunktet det fulde produkt, hvilket spejler sig i de diagrammer der opstår ift. Use Case's.

Udarbejdelsen af betaversionen har givet kendskab og bedre forståelse for den iterative arbejdsproces, som ses illustreret i figur 5. Derudover har projektgruppen dannet sig erfaring med at arbejde i mindre grupper, under implementeringsfasen og løbende sammensætte de færdige moduler til et samlet produkt.

Reviews har bidraget med konstruktiv kritik, der har haft indflydelse på projektgruppens udarbejdelse af projektrapporten. Dette har givet et indblik i, hvordan projektet betragtes udefra. Dette har bidraget med selvreflektion, som har gjort at projektet har udviklet sig i nye retninger, da der er blevet lagt fokus på nye områder. Yderligere er det erfaret, hvordan andre projektgrupper har udarbejdet deres projekt, som har bidraget med alternative løsninger af produktudvikling fra idé til færdigprodukt.

20 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_Referat.pdf
- Logbog.pdf
- Procesbeskrivelse.pdf
- Bilagsrapport.pdf
- Kode implementeringer
  - GUI_implementering.zip
  - PSoC_implementering.zip
- Accepttest1.png
- Accepttest2.png

21 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
- Information om ASE modellen
  `https://blackboard.au.dk/bbcswebdav/pid-2711466-dt-content-rid-8854723_1/\protect\@normalcr\relaxcourses/BB-Cou-UUVA-91809/I2ISE%20Lessons/Development%20Processes.pdf`
- Information om den agile udviklingsproces SCRUM
  `https://blackboard.au.dk/bbcswebdav/pid-2711468-dt-content-rid-8854735_1/\protect\@normalcr\relaxcourses/BB-Cou-UUVA-91809/I2ISE%20Lessons/Scrum.pdf`
- Artikel om den agile udviklingsproces "Making sence 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 MAXsonar-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`