

Broomba

3. SEMESTERPROJEKT

Aarhus University School of Engineering

ARKITEKTUR, FORANALYSE & DESIGN

EE3PRJ3 - GRUPPE 16

<i>Christoffer Broberg</i>	<i>201605997</i>
<i>Mathias Fredslund Jensen</i>	<i>201502302</i>
<i>Erik Kato Ipsen</i>	<i>201406553</i>
<i>Parweiz Haqshenas</i>	<i>201408675</i>
<i>Lars Holm</i>	<i>201304971</i>
<i>Rasmus Aabo Jørgensen</i>	<i>201303676</i>
<i>Minik Nathanielsen Olsen</i>	<i>201600341</i>

VEJLEDER

Søren Nielsen

Aarhus University School of Engineering

AFLEVERET

22-12-2017

Versionshistorie

Version	Dato	Navn	Beskrivelse
0.0.0	25.09.2017	Mathias Fredslund Jensen	Dokument oprettet - dokument opsætning
0.1.0	13.11.2017	Mathias Fredslund Jensen Rasmus Aabo Jørgensen	Oprettning og Rettelse af BDD og IBD diagrammer samt signalbeskrivelse. Gennemgå dokument for ordforklaringer.
0.1.1	13.11.2017	Software holdet	Applikationsmodeller, sekvens diagrammer og domænemodel tilføjet
0.1.2	17.11.2017	Hardware holdet	Oprettelse og udfyldelse af diverse hardware design
0.1.3	17.11.2017	Software holdet	Logical, Data, Deployment view tilføjet
0.1.4	13.12.2017	Hardware holdet	Rettelser og resterende af hardware design udfyldt
0.1.5	13.12.2017	Software holdet	Rettelser til Logical, Data, Deployment view
0.1.6	19.12.2017	Hardware holdet	Endelige rettelser i hardware design
0.1.7	19.12.2017	Software holdet	Endelige rettelser til N+1 view
1.0.0	20.12.2017	Hele gruppen	Gennemgang og sidste rettelser af dokument

Ordforklaring

Ord/Forkortelse	Beskrivelse
Loadcell	En loadcell er en transducer, der bruges til at skabe et elektrisk signal, hvis størrelse er direkte proportional med den kraft der måles.
BDD	Block Definition Diagram
IBD	Internal Block Diagram
SYSML	System Modeling Language
PSoC	Programmable System-on-Chip
RPI	Raspberry Pi Zero Wifi
GUI	Graphical user interface
mosFET	Metal Oxide Semiconductor Field Effect Transistor
HTML	HyperText Markup Language er et opmærknings-sprog der primært har til formål at få opsat billeder og tekst på en hjemmeside

Indhold

1 Arkitektur	6
2 Hardware Arkitektur	6
2.1 BDD	6
2.2 IBD	10
2.3 Signal Beskrivelse	13
3 Software Arkitektur	19
3.1 Domæne Model	19
3.2 Sekvensdiagram	19
3.3 N+1 - View	22
3.4 Logical view	23
3.4.1 PSoC	24
3.4.2 RPI	29
4 Foranalyse	33
4.1 Front Tryk	33
4.2 Væske	33
4.3 DC-motorer	33
4.4 Karosseri	34
4.5 Pumpe	34
4.6 SPI-kommunikation	34
4.7 GUI	34
4.8 Overvejelser om implementering af software	34
5 Hardware Design	35
5.1 Motor system	35
5.1.1 Pumpe til vand	35
5.1.2 H-Bro	36
5.1.3 DC motor skrubber	41
5.1.4 Stykliste	41
5.2 Batteri	42

5.2.1	Kort om spændingsregulator	42
5.2.2	5V spændingsregulator	43
5.2.3	6V spændingsregulator	44
5.2.4	Spændingsdeler	45
5.2.5	Stykliste	46
5.3	Front tryk system	47
5.3.1	Stykliste	47
5.3.2	Load cell	48
5.3.3	Stykliste	48
6	Hardware Implementering	49
6.1	Pumpe til vand implementering	49
6.2	H-bro implementering	50
6.3	DC motor skrubber implementering	51
6.4	Batterier implementering	52
6.5	Front tryk system implementering	53
6.6	Load cell implementering	54
7	Software Design	55
7.1	Process view	55
7.2	Data View	55
7.2.1	Dataoverblik	56
7.3	Deployment View	56
7.3.1	Oversigt over kommunikationsforbindelser	56
7.3.2	Kommunikation mellem RPI og PSoC	57
7.3.3	Inputs fra frontsensorer til PSoC	57
7.3.4	Inputs fra væskebeholders load cell til PSoC	57
7.3.5	Inputs fra batterimåling til PSoC	57
7.4	Inputs fra motorspænding til PSoC	57
7.5	Implementation View	58
7.5.1	Pinassignments	58
7.5.2	Topdesign	59

7.5.3	Webserver	59
7.5.4	Moduler - PSoC	60
7.5.5	MotorControl	60
7.5.6	LiquidSensor	60
7.5.7	LiquidControl	61
7.5.8	BatterySensor	62
7.5.9	RPICom	62
7.5.10	Brush	62
7.5.11	Interruptrutiner	62
7.6	Moduler - RPI	63

1 Arkitektur

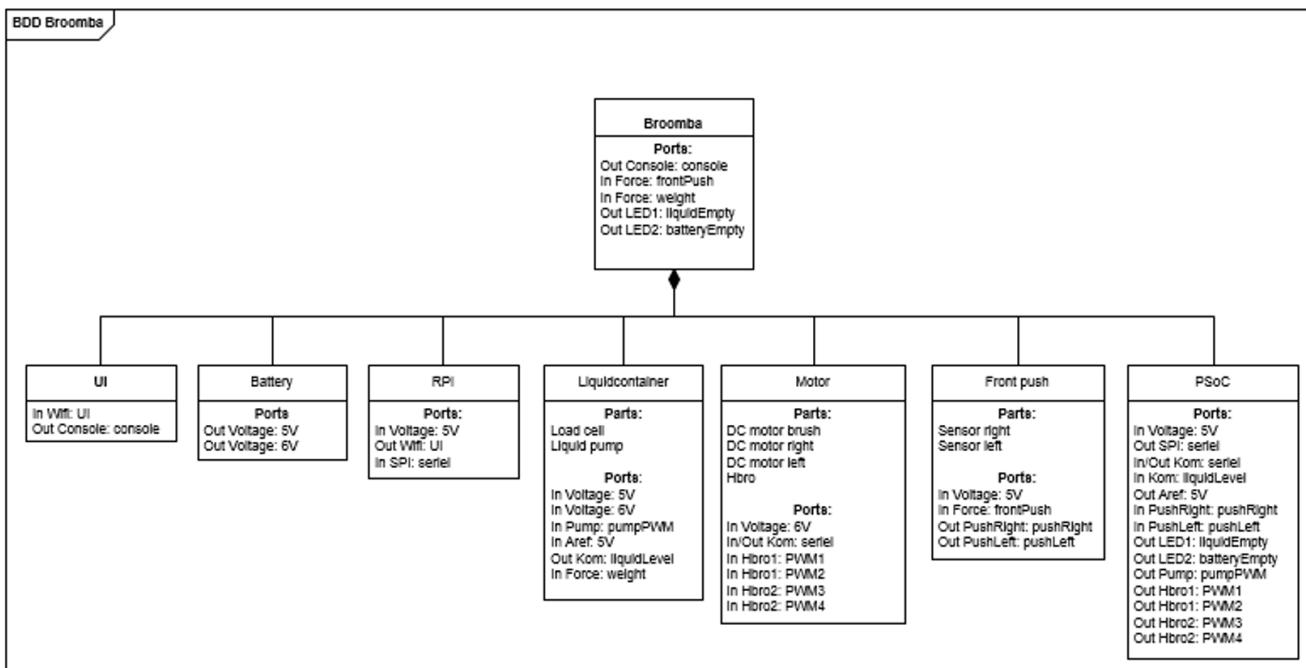
Systemarkitekturen illustrerer vha. SysML, som er hvordan systemet overordnet er designet. Systemarkitekturen for elektroniske systemer, som Broomba, kan overordnet opdeles i to hovedkategorier: hardware arkitekturen og software arkitekturen.

2 Hardware Arkitektur

Først anvendes BDD til at sætte systemet op som blokke, hvor det efterfølgende kan brydes ned i mindre blokke, indtil der opnåes tilstrækkelig overskuelighed. Hvert af disse blokke får en kort blokbeskrivelse. Til at illustrere hvordan blokkene er forbundet til hinanden, samt hvilke signaler der sendes mellem blokkene. anvendes IBD samt en signalbeskrivelses tabel, der forklare signalerne med ord.

2.1 BDD

Block Definition Diagram eller BDD angiver hvilke blokke et system består af. Blokkene er karakteriseret ved at besidde lav kobling og høj intern samhørighed.



Figur 1: Her ses det overordnet system af broombæn, som illustreres via et BDD diagram.

Blok beskrivelse Broomba

Det beskrevet system indeholder forskellige blokke. Disse blokke har forskellige funktioner, der udføres for systemet. BDD'et der er vist i figur 1, og er den første iteration af systemet. Nedenunder kan man se de beskrevet hardware

blokke der er i dette system.

Broomba

Denne blok er hovedblokken, der indeholder inputs og outputs der går til og fra systemet udefra, dette ses fra brugerens side.

UI

Denne blok er webserveren der bruges til at fortælle bruger om væske/batteri niveau.

Battery

Denne blok er spændingskilden i systemet, som forsyner de forskellige hardware dele med den korrekte spænding, enten 5V eller 6V.

RPI

RPI'en er mellem ledet imellem brugeren og PSoC'en. Den sørger for at formidle informationer der skal gives videre til UI fra PSoC'en, således at brugeren kan se væske- og batteri tilstand.

Liquidcontainer

VæskeBeholder har til formål at fortælle systemet, om der skal fyldes væske i beholderen, eller om den er tilstrækkeligt fyldt. Den skal derfor sende et signal til PSoC'en som sender en besked videre til RPI og videre til en webserver. PSoC'en skal også sende et output til en LED der illustrerer at der skal påfyldes væske.

Denne blok indholder også en peristaltisk pumpe, der ved hjælp af et PWM signal pumper væsken fra væskebeholderen og ud på gulvet.

Motor

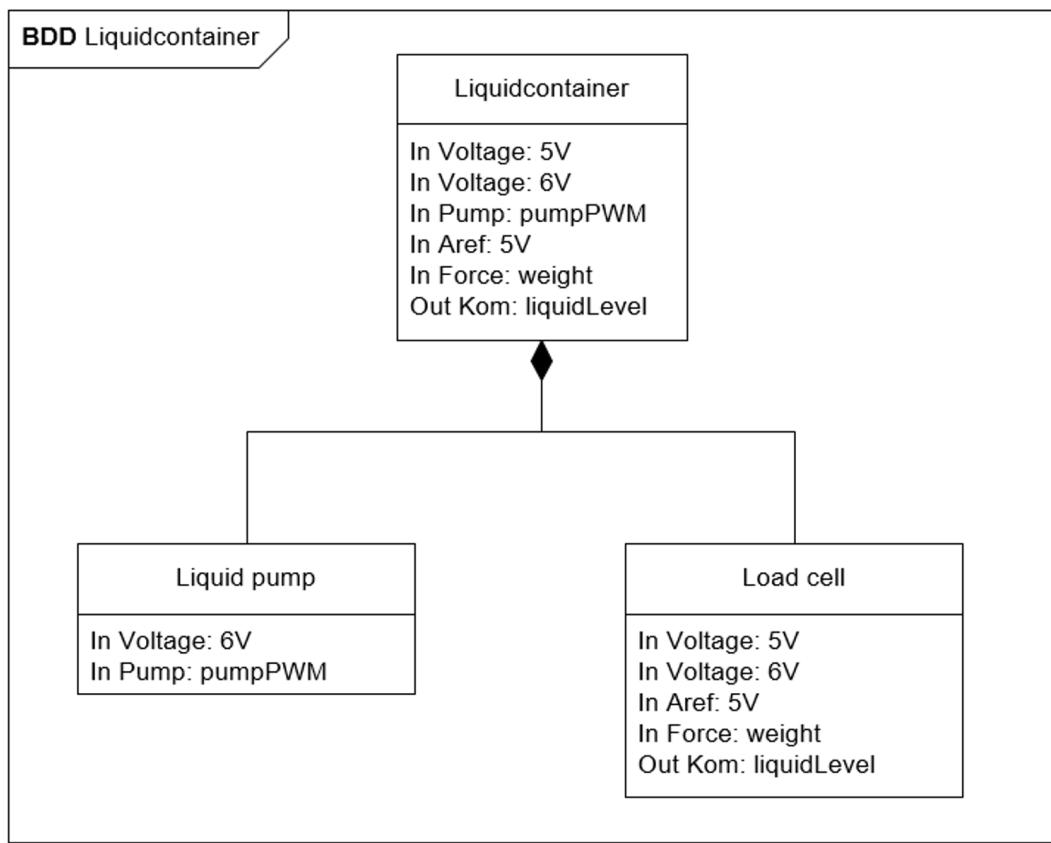
Motoren indeholder 3 DC motore, en motor til henholdsvis højre og venstre trækhjul, og en DC motor til børsten der skal vaske gulvet. De to trækmotorer skal kunne styre både retning og hastigheden som Broomba'en kører.

Front push

Det system indeholder 2 tryksensorer, som blot sender signal til PSoC'en når de bliver aktiveret af omgivelserne.

PSoC

PSoC'en er den der styre de 3 motorer og får inputs fra de forskellige sensorer der er i systemet, og videre giver disse informationer til henholdsvis bruger via RPI'en til webserveren.



Figur 2: Dette diagram viser væskebeholderens hardware dele.

Blok beskrivelse BDD Liquidcontainer

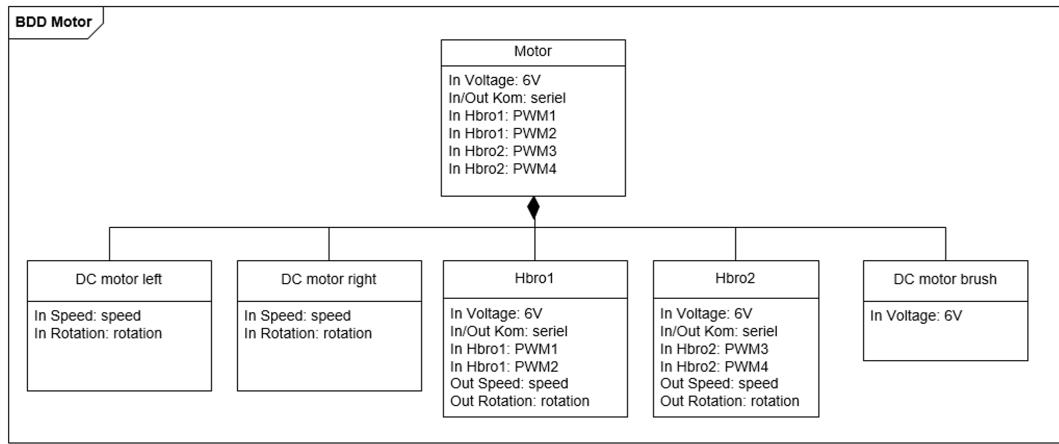
Denne blok indeholder flere dele for væskebeholderen.

Liquid pump

Liquid pump er en pumpe der skal pumpe vand fra væskebeholderen ud foran vores børste, dette gøres ved at tænde pumpen i et interval der gør at der kommer nok vand på gulvet.

Load cell

Load cellen er vægtsensoren, der mäter hvor meget væske der er i beholderen, og videregiver dette signal til PSoC'en.



Figur 3: Motorens harware dele ses på dette BDD diagram

Motor

indeholder alle underdelene som man ikke kan se i broomba BDD'et, og som har en ligeså stor vigtighed som det øvrige system.

DC motor left

DC motor left, er den venstre DC motor som bruges sammen med den højre DC motor, til frem/tilbage drift af broombaan.

DC motor right

DC motor right, er den højre DC motor som bruges sammen med den venstre DC motor til frem/tilbage drift af broombaan.

Hbro1

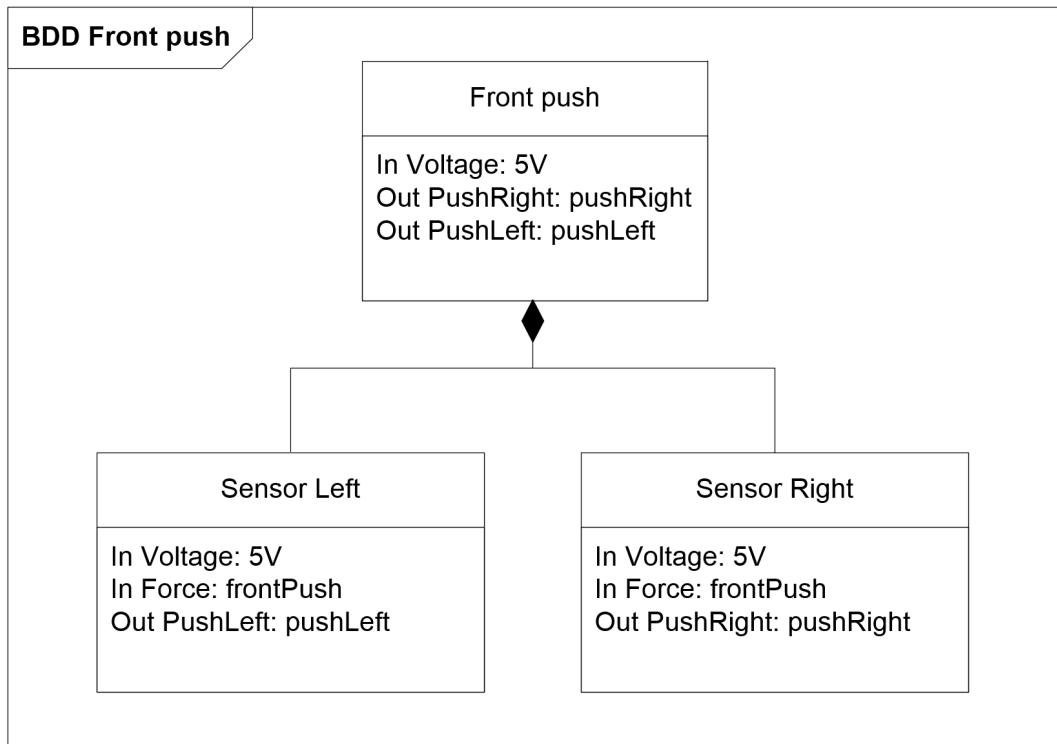
Hbro1 er styremodulet der skal styre hastigheden og rotationen af DC motor right ved hjælp af PWM signaler.

Hbro2

Hbro2 er styremodulet der skal styre hastigheden og rotationen af DC motor left ved hjælp af PWM signaler.

DC motor brush

DC motor brush er den DC motor der bruges til at vaske gulvet, som blot er en motor med en børste på en vinge der drejer rundt med en konstant fart når broomba kører.



Figur 4: De forskellige hardware dele, kan ses på dette diagram af front trykket.

Front push

Front push er baseret på de sensorer, der sidder forrest på Broombaen. Dvs. Når sensorerne bliver påvirket af omgivelserne, skal sensorerne sende et signal ind til PSoC'en. Dette er derfor en meget vigtig del for systemets måde at navigere på.

Sensor right

Denne sensor har til formål at sende signal til PSoC'en fra eventuelle tryk på front som Broomba'en skulle modtage.

Sensor left

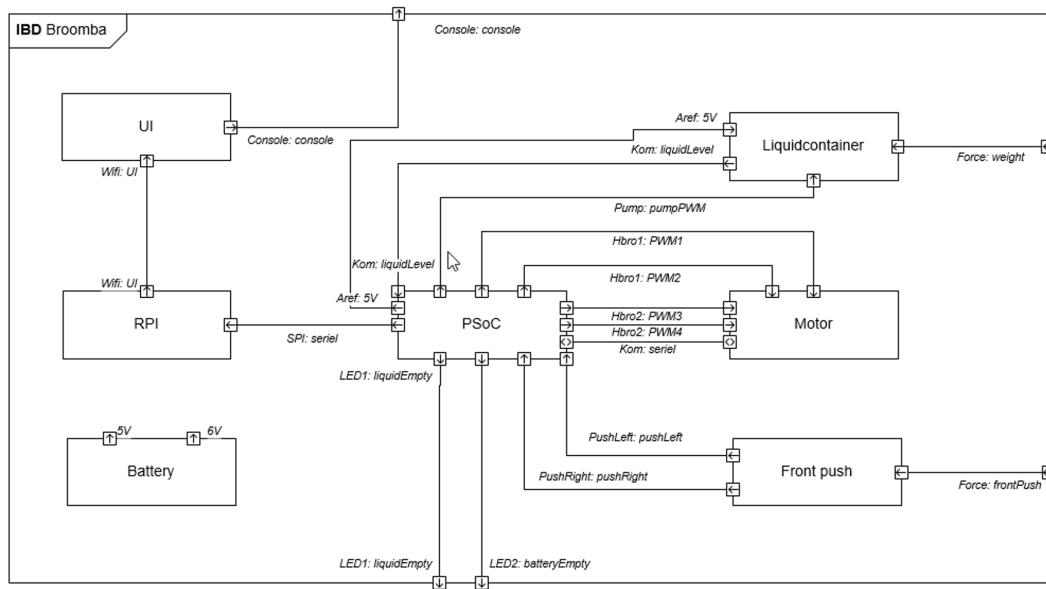
Denne sensor har til formål at sende signal til PSoC'en fra eventuelle tryk på front som Broomba'en skulle modtage.

2.2 IBD

Internal Block Diagram eller IBD angiver forbindelserne mellem systemets blokke defineret i BDD-diagrammerne fra Figur 1 til Figur 4, og suppleres af tabeller til beskrivelse af systemets signaler mellem blokkene, se Tabel 1 (2.3) til Tabel 17 (2.3).

IBD over Broomba systemet

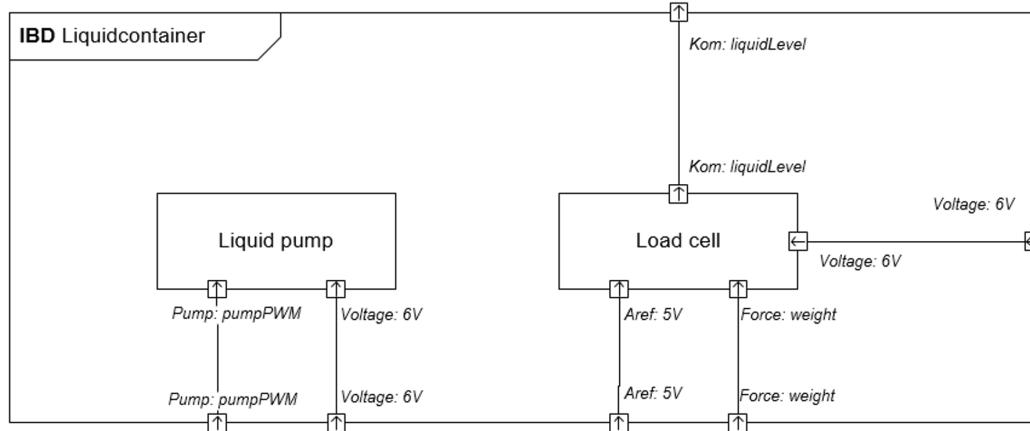
I Figur 5 kan man se det interne overblik over hele systemet.



Figur 5: Det interne system af Broombae kan ses i dette diagram.

IBD over Liquidcontainer blokken

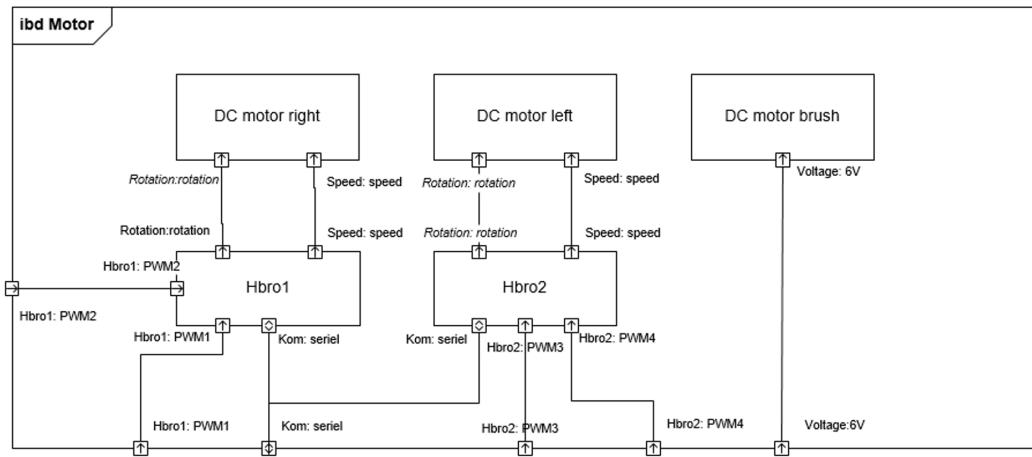
I Figur 6 kan man se det interne overblik over liquidcontainter blokken.



Figur 6: Det interne system af Væskebeholderen kan ses i dette diagram.

IBD over Motor blokken

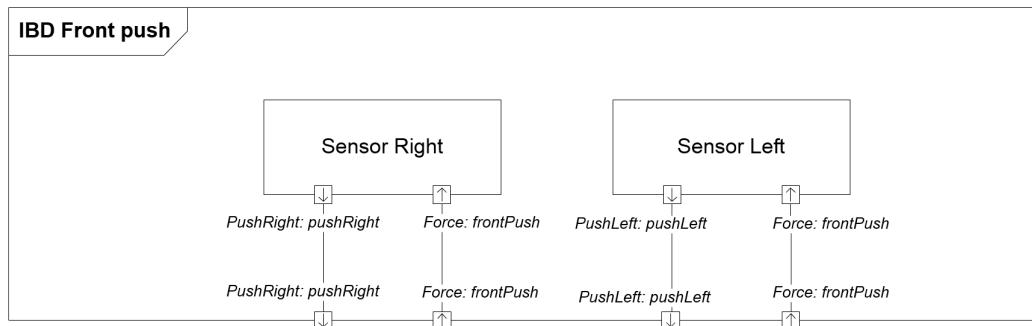
I Figur 7 kan man se det interne overblik over Motor blokken.



Figur 7: IBD diagrammet af motor arkitekturen.

IBD over Front push blokken

I Figur 8 kan man se det interne overblik over Front push blokken.



Figur 8: Front tryk arkitekturen illustreret via et IBD.

2.3 Signal Beskrivelse

I de følgende tabeller vil de enkelte blokke og signaler fra ovenstående afsnit forklares.

Broomba				
Funktionsbeskrivelse	Signaler	IN/OUT	Signalnavn	Signalbeskrivelse
Broomba signaler er signaler der kommer udefra systemet, eller fra systemet og ud	Console	Out	console	Kommunikation til brugeren via webserver
	Force	In	frontPush	Sensor aktivering af omgivelser
	Force	In	weight	Signal udefra når der påfyldes væske
	LED1	Out	liquidEmpty	Signal til en LED om væskebeholder lav niveau
	LED2	Out	batteryEmpty	Signal til en LED om batteri lav niveau

Tabel 1: signalbeskrivelse over Broomba

Battery				
Funktionsbeskrivelse	Signaler	IN/OUT	Signalnavn	Signalbeskrivelse
Batteriet skal forsyne Broombas system med Power	Voltage	Out	5V	5V spændingsoutput
	Voltage	Out	6V	6V spændingsoutput

Tabel 2: signalbeskrivelse over Battery

RPI				
Funktionsbeskrivelse	Signaler	IN/OUT	Signalnavn	Signalbeskrivelse
RPI er systemets styredel, der snakker sammen med webserveren og PSoC'en	Voltage	In	5V	5V spændingsforsyning
	Wifi	Out	UI	Kommunikationen fra RPi til webserver
	SPI	In	seriel	Kommunikation fra PSoC til RPi

Tabel 3: signalbeskrivelse over RPI

UI				
Funktionsbeskrivelse	Signaler	IN/OUT	Signalnavn	Signalbeskrivelse
UI er webserveren som bruges til at se når batteri eller væske niveau er lavt	Wifi	In	UI	Kommunikationen fra RPi til webserver
	Console	Out	console	Kommunikation til brugeren via webserver

Tabel 4: signalbeskrivelse over UI

Liquidcontainer				
Funktionsbeskrivelse	Signaler	IN/OUT	Signalnavn	Signalbeskrivelse
Væskebeholderen skal sende besked til webserveren når den er ved at løbe tør for væske	Voltage	In	5V	5V spændingsforsyning til print
	Voltage	In	6V	6V spændingsforsyning
	Pump	In	pumpPWM	PWM signal til pumpen
	Aref	In	5V	5V reference spænding til print
	Force	In	weight	Signal udefra når der påfyldes væske
	Kom	Out	seriel	signal ud til en LED der fortæller om der er nok væske

Tabel 5: signalbeskrivelse over Liquidcontainer

Load cell				
Funktionsbeskrivelse	Signaler	IN/OUT	Signalnavn	Signalbeskrivelse
Load cellen bruges til at måle niveauet af væske der er tilbage	Voltage	In	5V	5V spændingsforsyning til print
	Voltage	In	6V	6V spændingsforsyning
	Aref	In	5V	5V reference spænding til print
	Force	In	weight	Signal udefra når der påfyldes væske
	Kom	Out	liquidLevel	Signal til PSoC om væskenniveau

Tabel 6: signalbeskrivelse over Load cell

Liquid pump				
Funktionsbeskrivelse	Signaler	IN/OUT	Signalnavn	Signalbeskrivelse
Liquid pumpen bruges til at pumpe væske ned på gulvet	Voltage	In	6V	6V spændingsforsyning
	Pump	In	pumpPWM	PWM signal til pumpen

Tabel 7: signalbeskrivelse over Liquid pump

Motor				
Funktionsbeskrivelse	Signaler	IN/OUT	Signalnavn	Signalbeskrivelse
Forklare de forskellige signaler der går ind til motor modulet	Voltage	In	6V	6V spændingsforsyning
	Kom	In/Out	seriel	Kommunikation mellem motor og PSoC hvis broomba sidder fast
	Hbro1	In	PWM1	Første PWM signal til Hbro1
	Hbro1	In	PWM2	Andet PWM signal til Hbro1
	Hbro2	In	PWM3	Første PWM signal til Hbro2
	Hbro2	In	PWM4	Andet PWM signal til Hbro2

Tabel 8: signalbeskrivelse over Motor

DC Motor left				
Funktionsbeskrivelse	Signaler	IN/OUT	Signalnavn	Signalbeskrivelse
DC Motor left bruges til fremdriften af Broombaen's venstre hjul	Speed	In	speed	Fra Hbroen der bestemmer farten
	Rotation	In	rotation	Fra Hbroen der bestemmer retningen

Tabel 9: signalbeskrivelse over DC motor left

DC Motor right				
Funktionsbeskrivelse	Signaler	IN/OUT	Signalnavn	Signalbeskrivelse
DC Motor right bruges til fremdriften af Broombaen's højre hjul	Speed	In	speed	Fra Hbroen der bestemmer farten
	Rotation	In	rotation	Fra Hbroen der bestemmer retningen

Tabel 10: signalbeskrivelse over DC motor right

Hbro1				
Funktionsbeskrivelse	Signaler	IN/OUT	Signalnavn	Signalbeskrivelse
Hbro1 er styremodulet til motorerne	Voltage	In	6V	6V spændingsforsyning
	Kom	In/Out	seriel	Kommunikation mellem motor og PSoC hvis broomba sidder fast
	Hbro1	In	PWM1	Første PWM signal til Hbro1
	Hbro1	In	PWM2	Andet PWM signal til Hbro1
	Speed	Out	speed	Fra Hbroen der bestemmer farten
	Rotation	Out	rotation	Fra Hbroen der bestemmer retningen

Tabel 11: signalbeskrivelse over Hbro1

Hbro2				
Funktionsbeskrivelse	Signaler	IN/OUT	Signalnavn	Signalbeskrivelse
Hbro2 er styremodulet til motorerne	Voltage	In	6V	6V spændingsforsyning
	Kom	In/Out	seriel	Kommunikation mellem motor og PSoC hvis broomba sidder fast
	Hbro2	In	PWM3	Første PWM signal til Hbro2
	Hbro2	In	PWM4	Andet PWM signal til Hbro2
	Speed	Out	speed	Fra Hbroen der bestemmer farten
	Rotation	Out	rotation	Fra Hbroen der bestemmer retningen

Tabel 12: signalbeskrivelse over Hbro2

DC motor brush				
Funktionsbeskrivelse	Signaler	IN/OUT	Signalnavn	Signalbeskrivelse
DC motor brush bruges til at køre vores børste der skal vaske gulvet	Voltage	In	6V	6V spændingsforsyning

Tabel 13: signalbeskrivelse over DC motor brush

Front push				
Funktionsbeskrivelse	Signaler	IN/OUT	Signalnavn	Signalbeskrivelse
Front tryk er blokken der beskriver signaler der går ud fra sensor blokken	Voltage	In	5V	5V spændingsforsyning
	PushRight	Out	pushRight	Signal til PSoC om højre sensor er ramt
	PushLeft	Out	pushLeft	Signal til PSoC om venstre sensor er ramt

Tabel 14: signalbeskrivelse over Frontpush

Sensor right				
Funktionsbeskrivelse	Signaler	IN/OUT	Signalnavn	Signalbeskrivelse
Sensor right registrere når Broombæn rammer en forhindring i højre side	Voltage	In	5V	5V spændingsforsyning
	Force	In	frontPush	Sensor aktivering af omgivelser
	PushRight	Out	pushRight	Signal til PSoC om Højre sensor er ramt

Tabel 15: signalbeskrivelse over Sensor right

Sensor left				
Funktionsbeskrivelse	Signaler	IN/OUT	Signalnavn	Signalbeskrivelse
Sensor left registrere når Broombæn rammer en forhindring i Venstre side	Voltage	In	5V	5V spændingsforsyning
	Force	In	frontPush	Sensor aktivering af omgivelser
	PushLeft	Out	pushLeft	Signal til PSoC om venstre sensor er ramt

Tabel 16: signalbeskrivelse over Sensor left

PSoC				
Funktionsbeskrivelse	Signaler	IN/OUT	Signalnavn	Signalbeskrivelse
PSoC'en er styremodulet til sensorene og motorene	Voltage	In	5V	5V spændingsforsyning
	SPI	Out	seriel	Kommunikation mellem RPi og PSoC
	Kom	In/Out	seriel	Kommunikation mellem motor og PSoC hvis broomba sidder fast
	Kom	In	liquidLevel	Signal til PSoC om væskeniveau
	Aref	Out	5V	reference spænding til print
	PushRight	In	pushRight	Signal til PSoC om Højre sensor er ramt
	PushLeft	In	pushLeft	Signal til PSoC om venstre sensor er ramt
	LED1	Out	liquidEmpty	Signal til en LED om væskebeholder lav niveau
	LED2	Out	batteryEmpty	Signal til en LED om batteri lav niveau
	Pump	Out	pumpPWM	PWM signal til pumpen
	Hbro1	Out	PWM1	Første PWM signal til Hbro1
	Hbro1	Out	PWM2	Andet PWM signal til Hbro1
	Hbro2	Out	PWM3	Første PWM signal til Hbro2
	Hbro2	Out	PWM4	Andet PWM signal til Hbro2

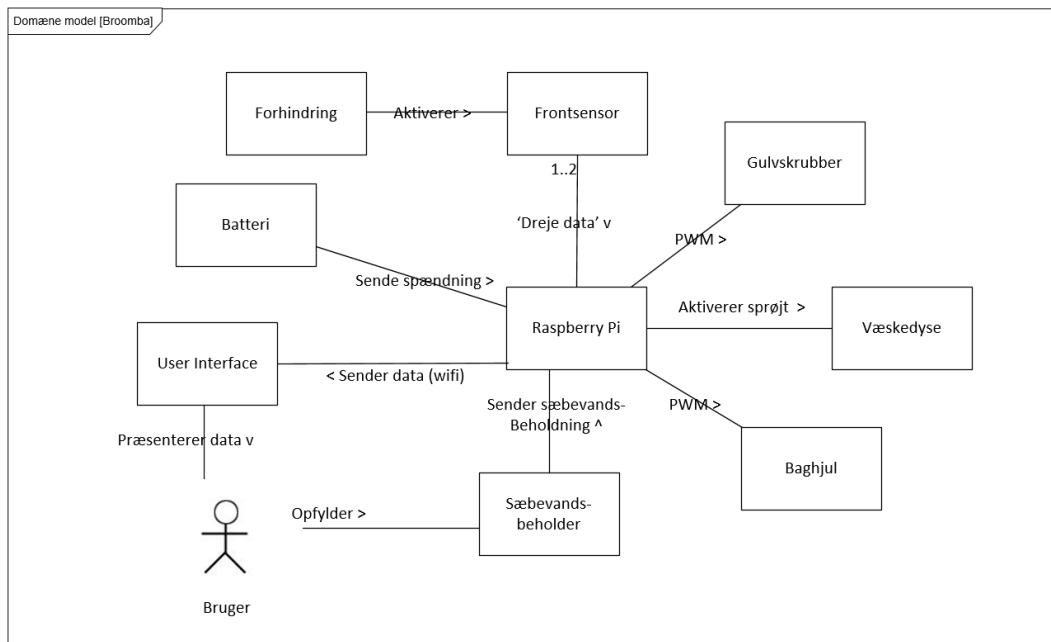
Tabel 17: signalbeskrivelse over PSoC'en

3 Software Arkitektur

Ud fra Broombas use cases udledes konceptuelle klasser, der danner grundlag for domænemodellen. Ud fra domænemodellen udarbejdes sekvensdiagrammer for hver use case. Ud fra sekvensdiagrammerne identificeres hver CPU. For hver CPU laves en applikationsmodel. Dette sker i det, der kaldes Logical View under det begrebsapparat, der kaldes N+1. Denne vil også kort blive introduceret. Metoderne udledt i applikationsmodellerne opsummeres til sidst i klassediagrammer.

3.1 Domæne Model

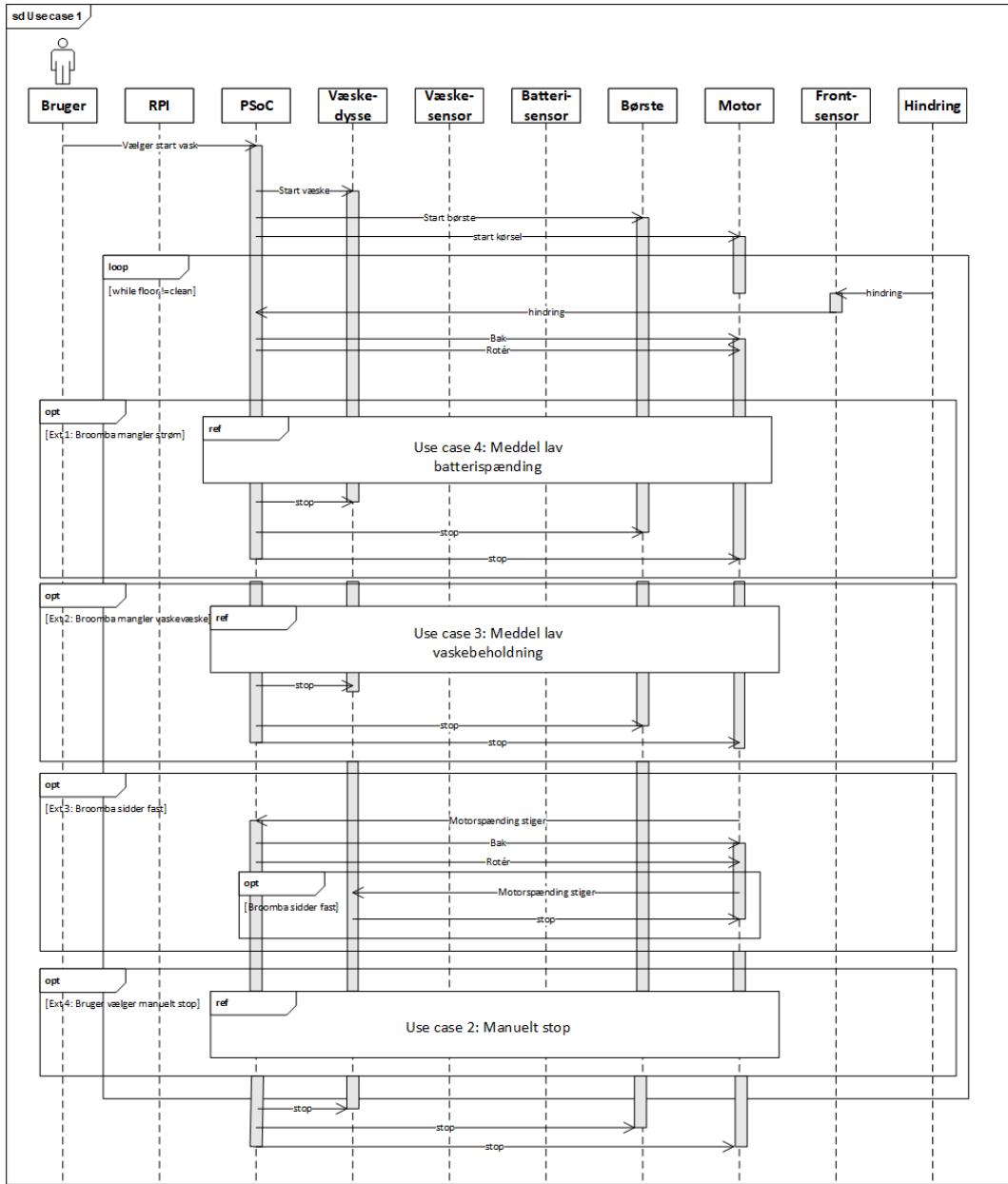
Domænemodellen illustrerer, hvordan de konceptuelle klasser overordnet interagerer med hinanden og bruges til at danne et idémæssigt overblik over systemet. De konceptuelle klasser udledes ved at identificere navneordene ud fra fully-dressed use cases[1], se Dokumentation kap.1. Det kan være nødvendigt at tilføje ekstra klasser for at varetage systemets funktioner.



Figur 9: Domæne Model

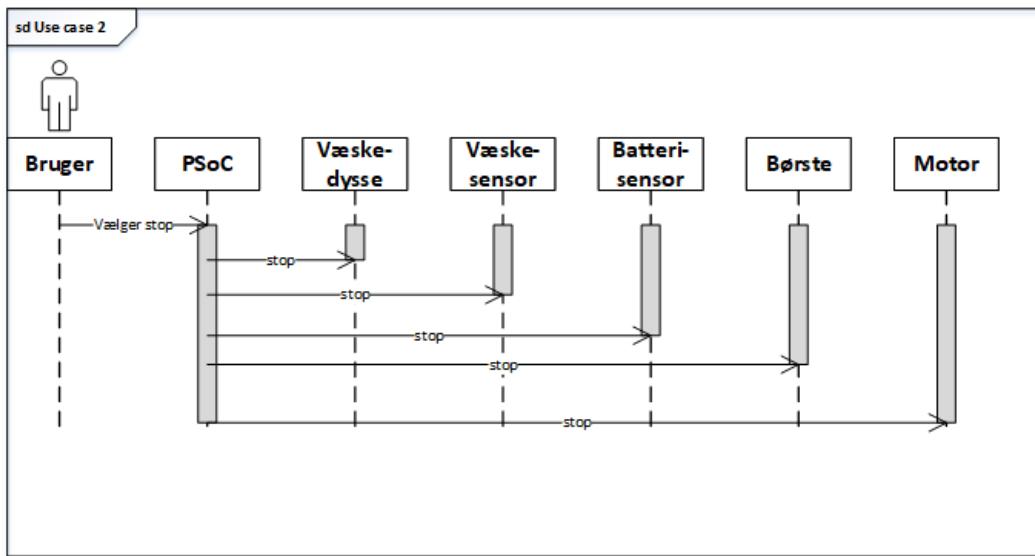
3.2 Sekvensdiagram

Sekvensdiagram illustrerer handlingsforløbet gennem systemet for hver use case mellem hver konceptuel blok fra domænemodellen i en story-line[2]. Handlingsforløbet tager sit udgangspunkt i fully dressed for hver use case, der kan ses i bilagsmaterialet (Kravspecifikation, s. 8).



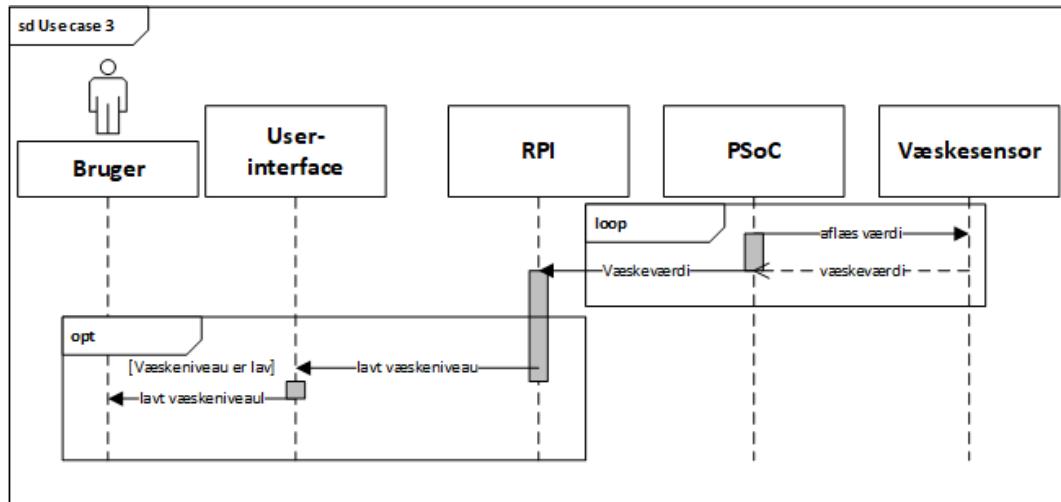
Figur 10: Sekvensdiagram Use case 1

Use case 1 er Broombas centrale Use case, og her interageres der med omverdenen via en række forskellige sensorer. Use case 1 bærer præg af, at den kan blive påvirket af de andre use cases. Disse præsenteres på de følgende diagrammer.



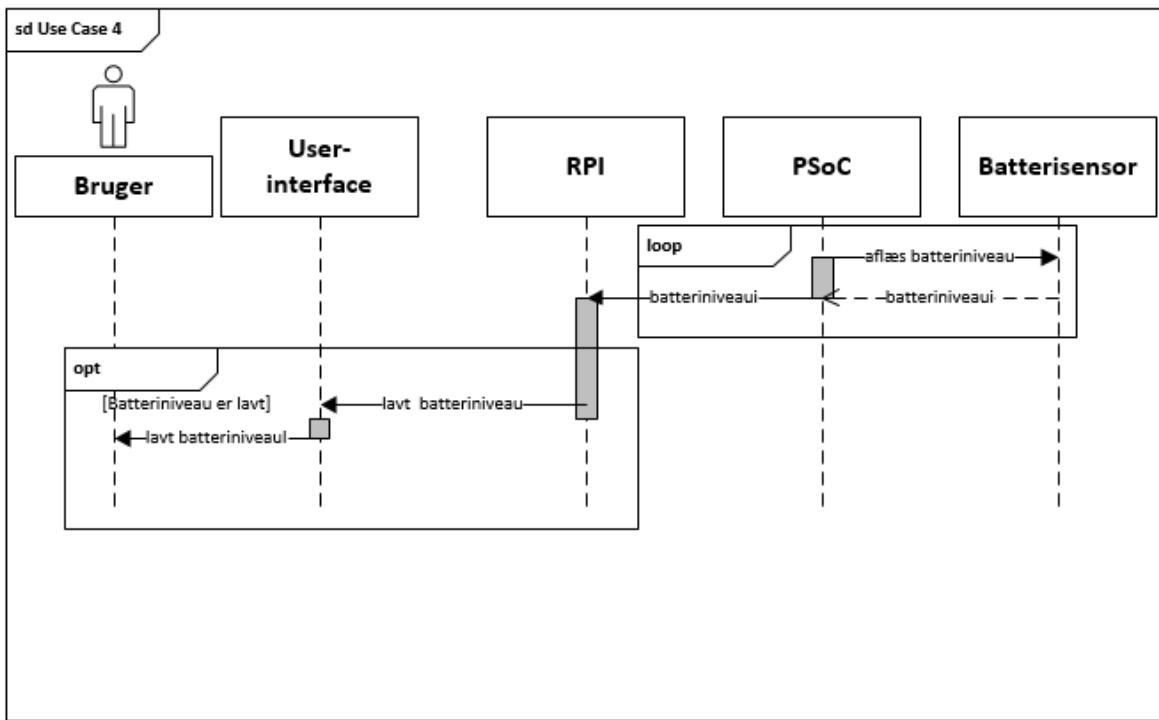
Figur 11: Sekvensdiagram Use case 2

Use case 2 handler om, at brugeren vælger manuelt stop. Her stoppes systemets sensorer og aktuatorer.



Figur 12: Sekvensdiagram Use Case 3

Use case 3 finder sted, når Broomba er ved at løbe tør for væske. Broomban meddeler via brugergrænsefladen til brugeren, at væskebeholderen er ved at være tom.



Figur 13: Sekvensdiagram Use Case 4

Use case 4 ligner Use case 3, men den finder sted, når Broomba er ved at løbe tør for strøm.

3.3 N+1 - View

N+1 er en model til at udvikle software. Modellen skaber overgangen fra softwarearkitektur til softwaredesign, idet man bevæger sig rundt i N antal forskellige ”views- perspektiver - der alle tager afsæt i de use cases, der er opstillet for systemet. Use casene er således ”+1”. Der er valgt følgende views, der vil blive præsenteret, når de dokumenteres:

- Logical view
- Process view
- Data view
- Deployment view
- Implementation view

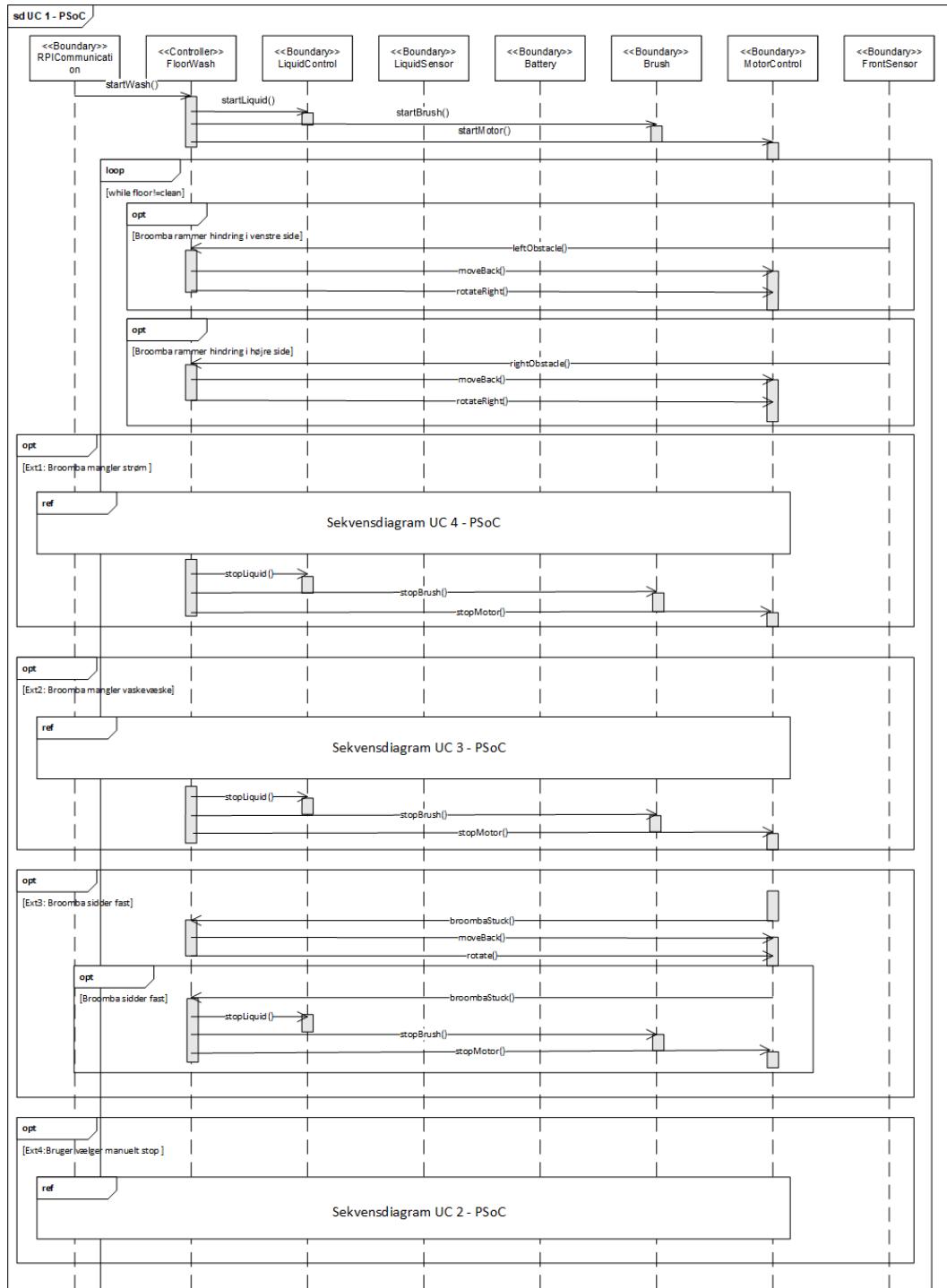
For yderligere information om N+1 henvises til bilagsmaterialet (N+1).

3.4 Logical view

Det første view, der vil blive taget i brug her, er Logical View. Logical View handler om at lave applikationsmodeller for hver CPU i systemet. Hver applikationsmodel består af ét sekvensdiagram og ét klassediagram, og formålet er at vise, hvilke metodekald, der er nødvendige mellem klasserne på en givet CPU til den pågældende use case. Klasserne karakteriseres som hhv. domain-, boundary- eller controller klasser. Domain er en klasse, der bruges i en anden klasse, og opbevarer information. Controller er den centrale styrekasse i use casen, og boundary er en klasse, hvorigennem controllerklassen interagerer med det omgivende system. Metodekaldende, der er udledt i sekvensdiagrammerne, opsamles for hver CPU i et klassediagram, hvor der også angives klasserelationer mellem hhv. domain-, boundary- og controllerklasserne. Klassediagrammerne anvendes, selvom der anvendes C-kode, og C-sproget ikke understøtter klasser. Ikke desto mere giver applikationerne en systemastisk afledning af relevante klasser, ligesom klassediagrammerne i sig selv giver en struktureret oversigt over, hvilken software, der skal laves. I sidste ende laves de enkelte klasser så som moduler. Der tages udgangspunkt i domænemodellen og systemsekvensdiagrammerne, når der skal identificeres relevante klasser.

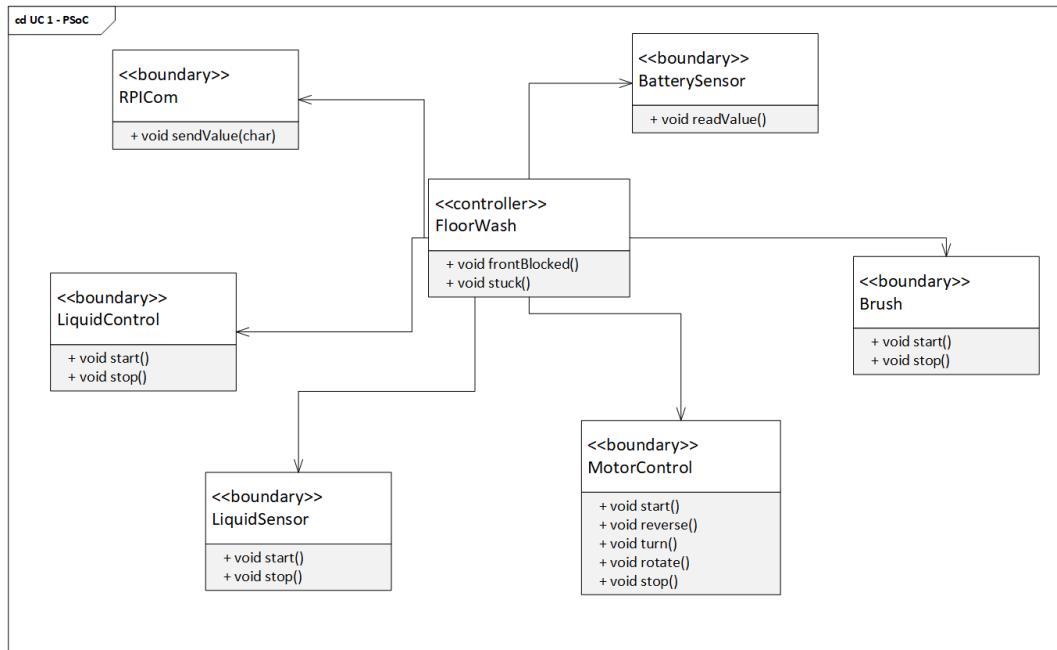
3.4.1 PSoC

Denne del viser viser applikationsmodeller for den software, der skal ligge på Broombaens PSoC.

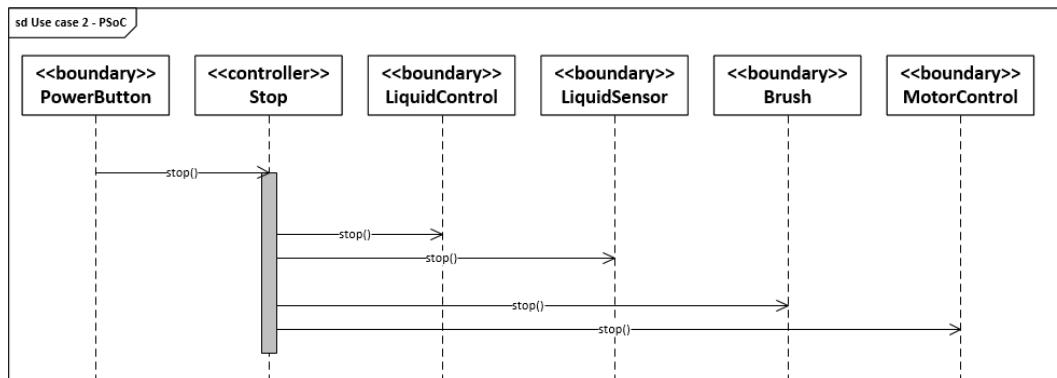


Figur 14: Sekvensdiagram for UC 1 - PSoC

For Use case 1 kan man se, at der er en boundary for hver sensor og aktuator, ligesom der også er en boundary, der forbinder til RPI'en. Metoderne i sekvensdiagrammet giver anledning til følgende klassediagram:

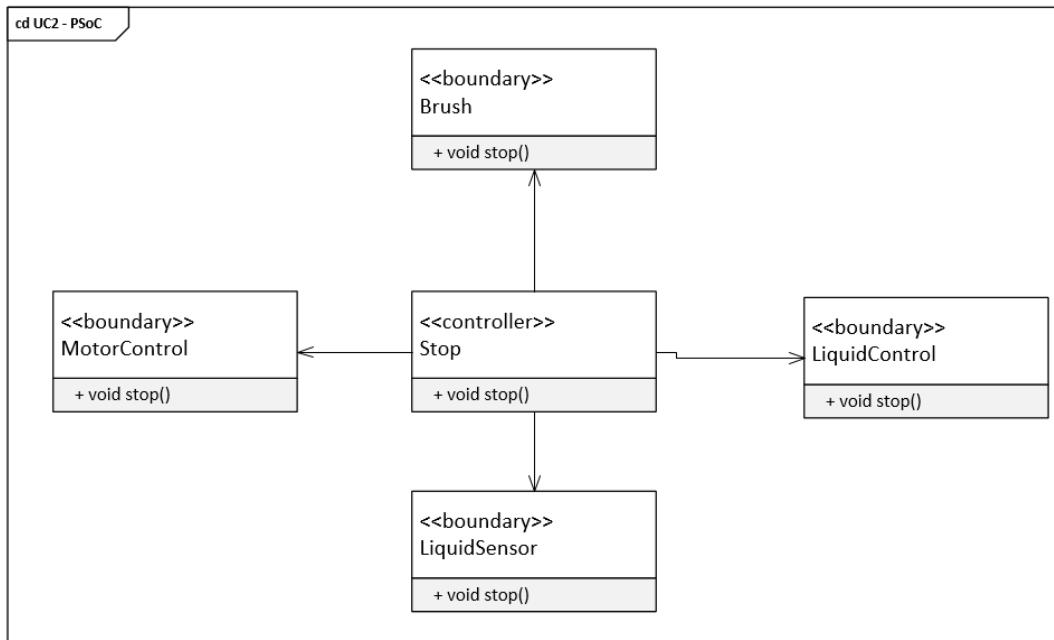


Figur 15: Klassediagram for UC 1 - PSoC

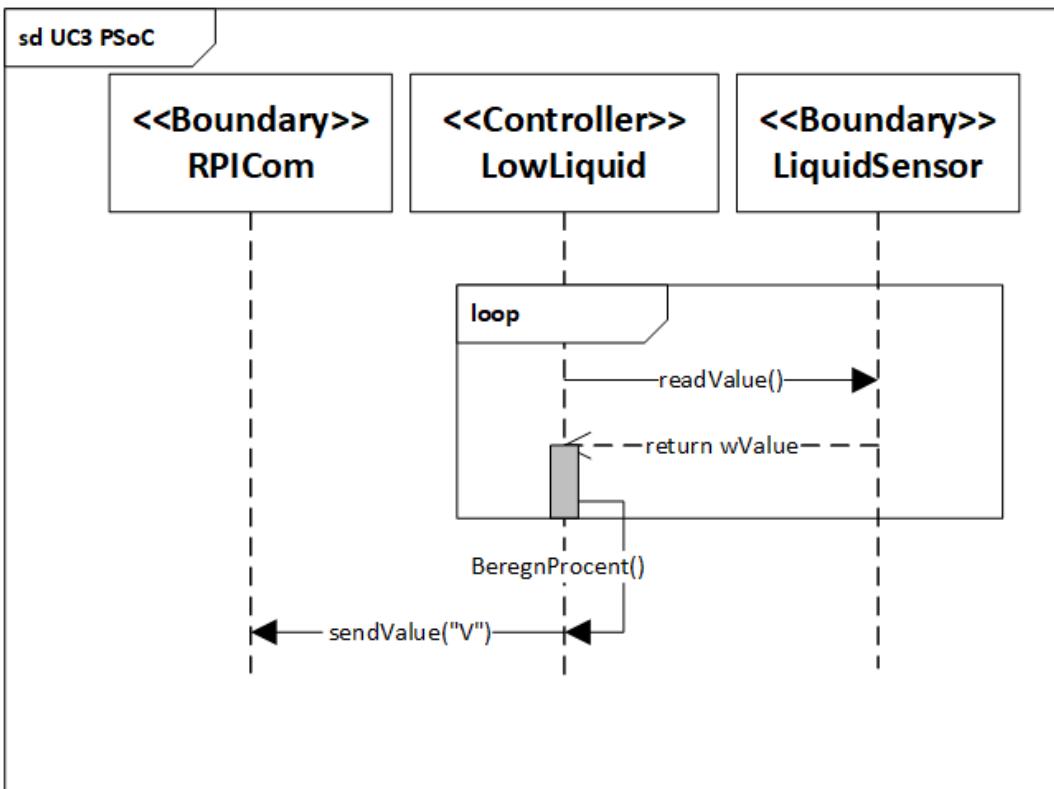


Figur 16: Sekvensdiagram for UC 2 - PSoC

I Use case 2 kalder controllerklassen stop-funktioner hos de forskellige sensorer og aktuatorer. Dette giver anledning til følgende klassediagram:

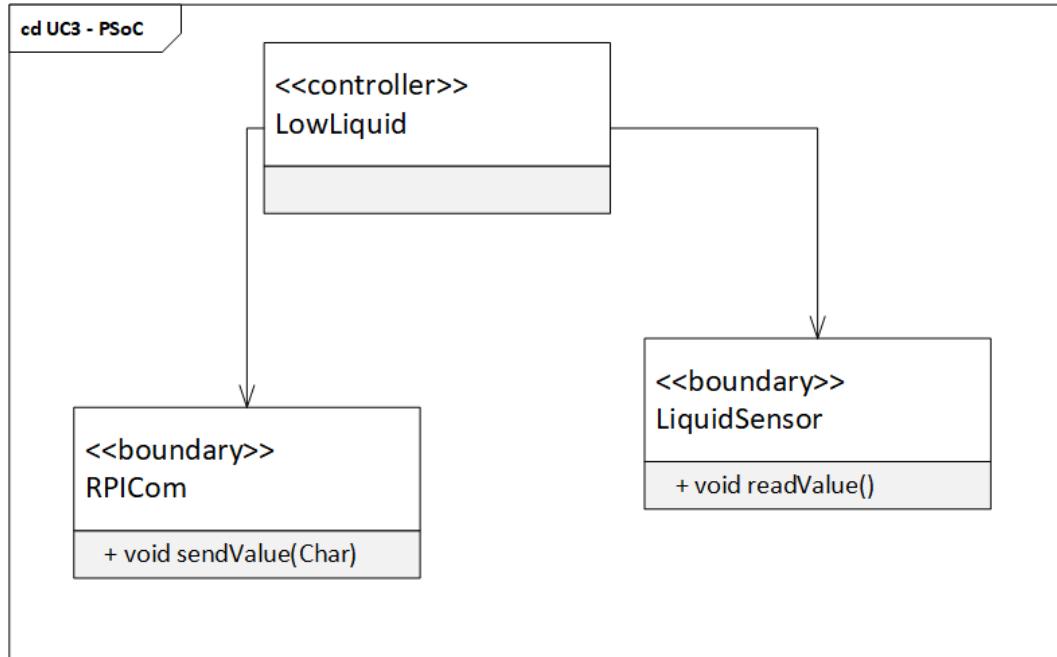


Figur 17: Klassediagramm for UC 2 - PSoC

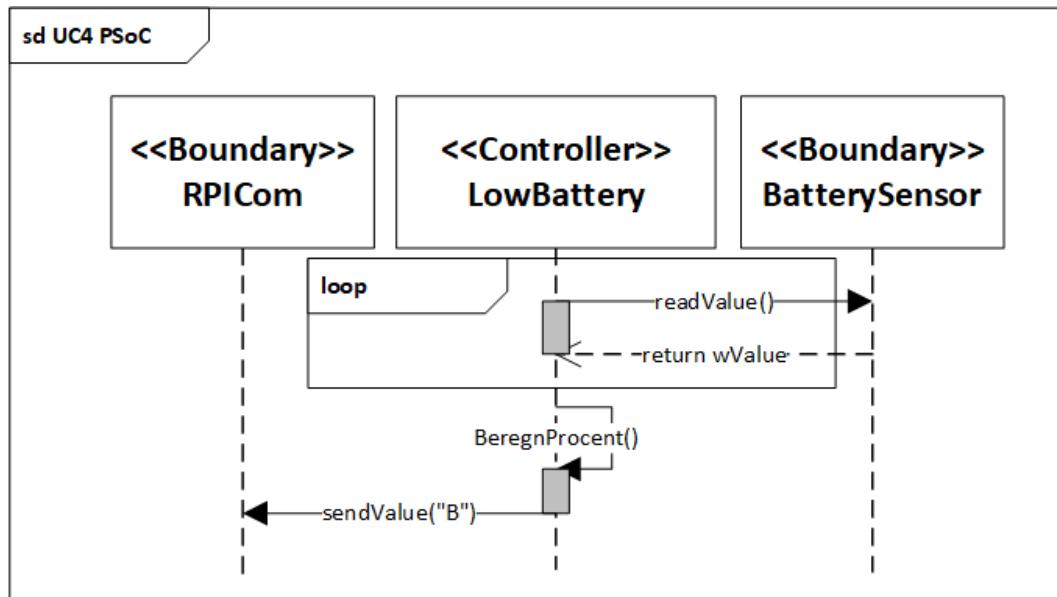


Figur 18: Sekvensdiagramm for UC 3 - PSoC

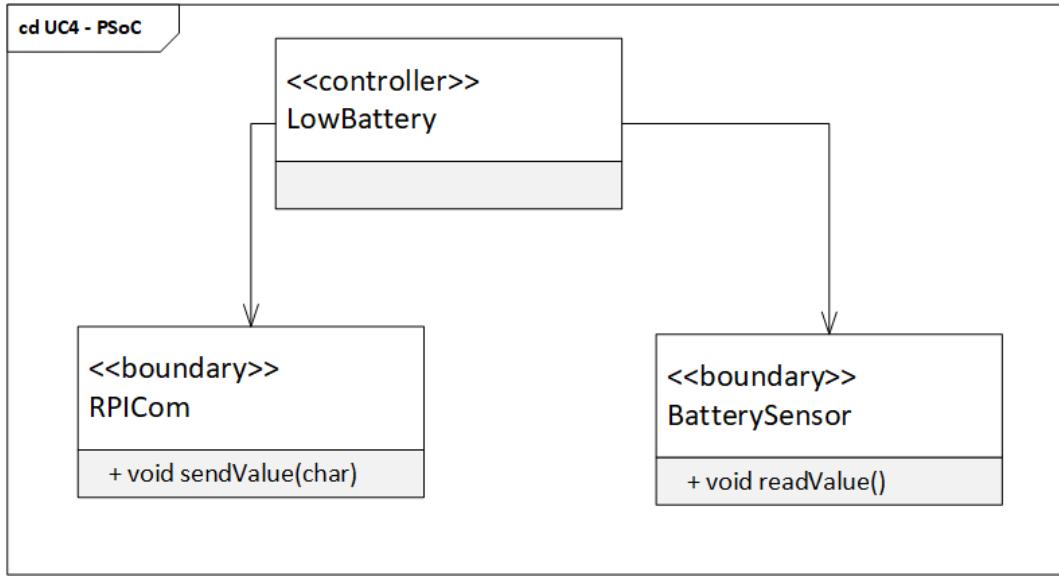
Use case 3 og 4 minder meget om hinanden, idet det er dem, der skal meddele brugeren om, at der mangler hhv. sæbevand og strøm. Her interagerer controllerklassen med hhv. væskesensoren og batterisensoren, mens der sendes til RPI'en, hvis sensorerne mäter lav værdi.



Figur 19: Klassediagram for UC 3 - PSoC

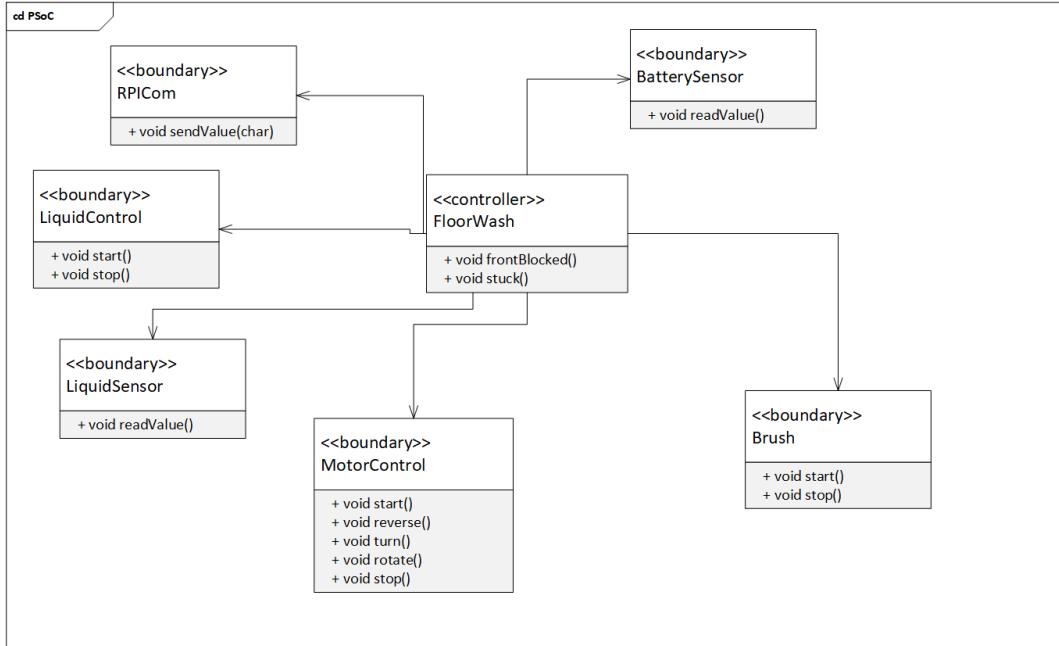


Figur 20: Sekvensdiagram for UC 4 - PSoC



Figur 21: Klassediagram for UC 4 - PSoC

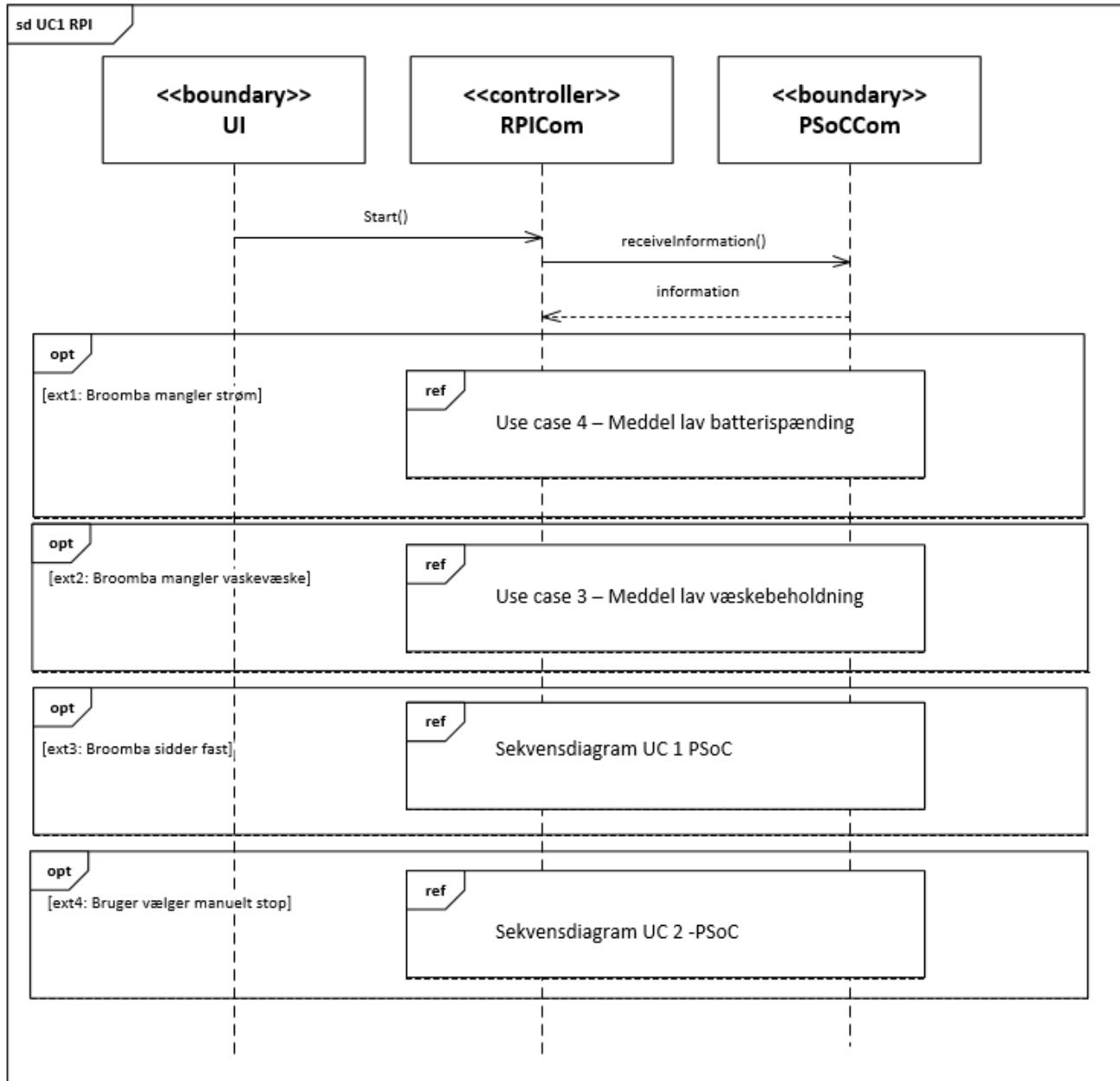
De enkelte applikationsmodeller lægges sammen til følgende samlede klassediagram for PSoC'en:



Figur 22: Klassediagram for PSoC

3.4.2 RPI

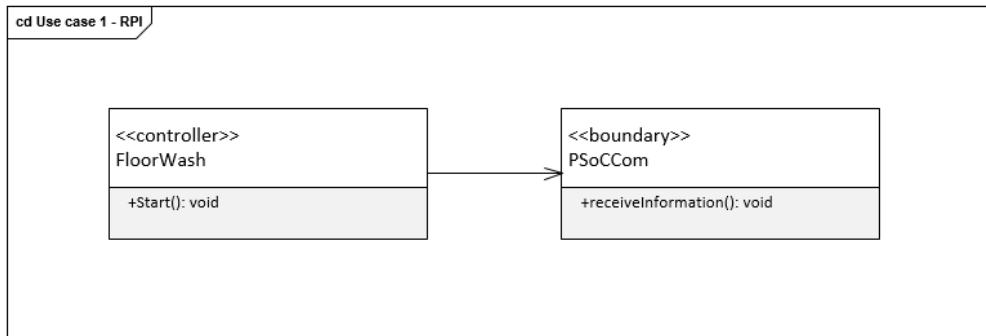
Denne del viser viser applikationsmodeller for den software, der skal ligge på Broombaens RPI.



Figur 23: Sekvensdiagram for UC 1 - RPI

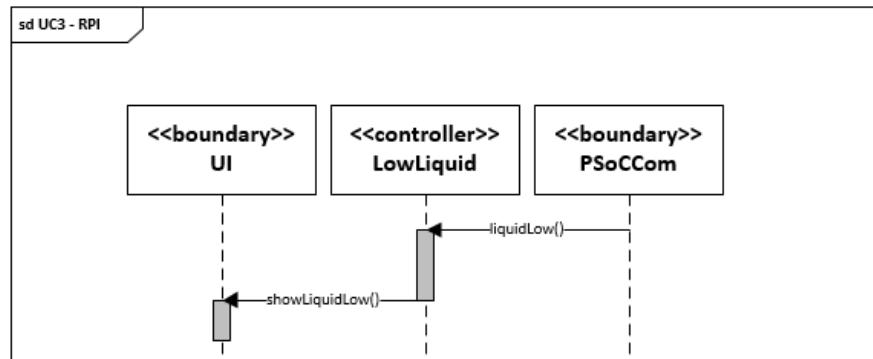
Sekvensdiagrammet for Use case 1 bærer præg af, at den kan påvirkes af de andre use cases. Use casen startes af et input fra brugergrænsefladen.

Sekvensdiagrammet giver anledning til følgende klassediagram:

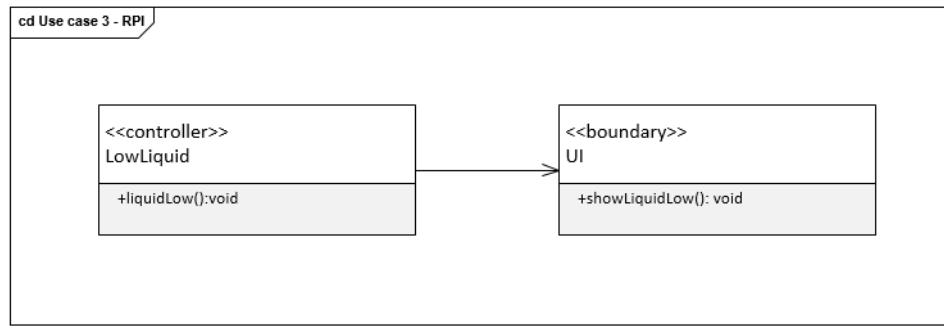


Figur 24: Klassediagram for UC 1 - RPI

RPI'en indgår ikke i Use case 2, så der er laves ikke applikationsmodel for RPI'en for denne Use case. For Use case 3 gælder, at der modtages et input fra PSoC'en, hvorefter controllerklassen kalder en metode i UI-boundary'en.

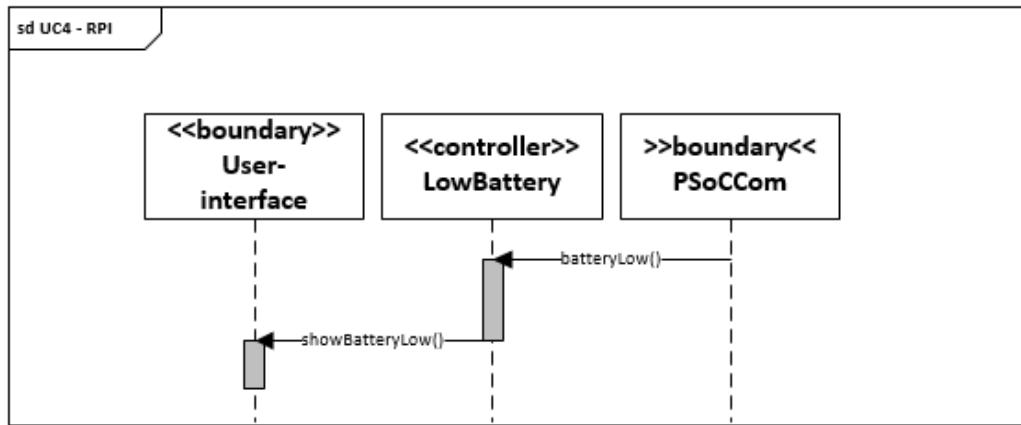


Figur 25: Sekvensdiagram for UC 3 - RPI

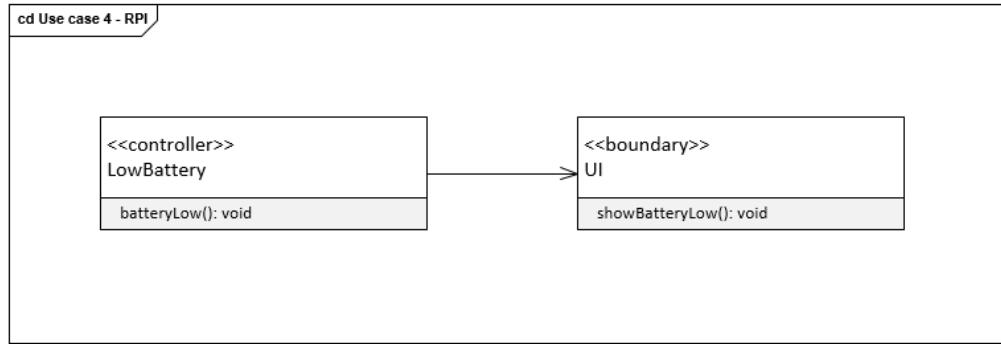


Figur 26: Klassediagram for UC 3 - RPI

Use case 4 er fuldstændig analog til Use case 4, men her modtages et andet input, hvorefter der kaldes en anden metode i UI-klassen.

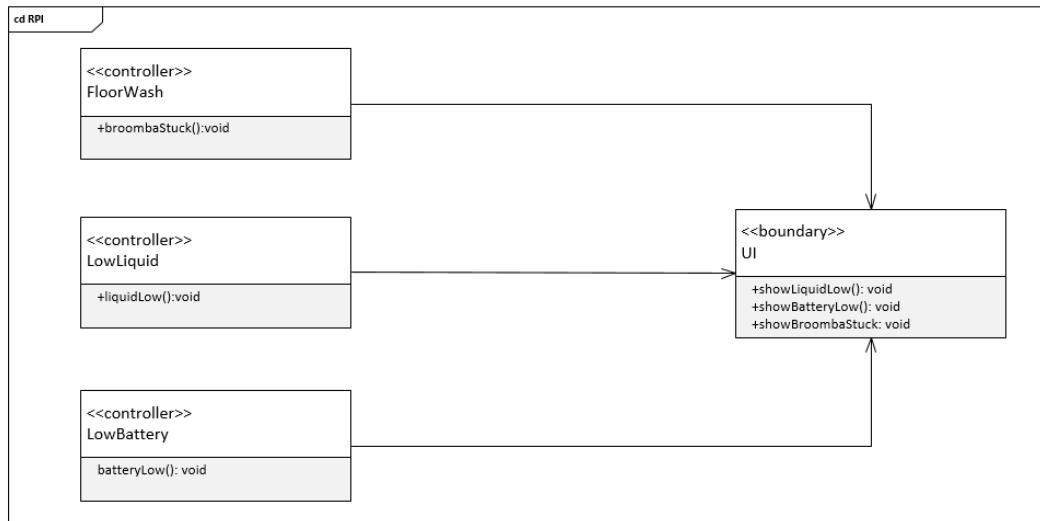


Figur 27: Sekvensdiagram for UC 4 - RPI



Figur 28: Klassediagram for UC 4 - RPI

Applikationsmodellerne giver anledning til et samlet klassediagram for RPI'en, der ses her:



Figur 29: Klassediagram for RPI

4 Foranalyse

I dette afsnit vil vores tanker omkring de forskellige dele af projektet være beskrevet, og hvordan vi har tænkt os at disse dele skal laves.

4.1 Front Tryk

Front trykkene er tænkt som en open/close switch, der sender 5V ind på et input på PSoC'en når switch'en (DM1[6]) bliver aktiveret, der kommer til at sidde én i hver side. Ifølge datasheetet for DM1 [6] switchen, kan den håndtere 5V DC, som passer fint med at vi bare sender 5V ud, og får en aktiv 5V tilbage.

4.2 Væske

Væskebeholder

Væskebeholderen er bare tænkt som en flaske hvor toppen bliver skåret af. Da dyse funktionen bliver som en pumpe der kommer ned i beholderen fra toppen af. Væskebeholderen skal også være beregnet til at kunne tages ud, da det ikke er særligt hensigtsmæssigt for kunden at skulle holde Broomba ind under en fx vask for at påfyldes væskebeholderen og derved risikere at der kommer vand i de elektroniske kredsløb.

Væskesensor

Væskesensoren bliver lavet som en load-cell som væskebeholderen står på. Load-cell måler beholderens vægt og giver signal tilbage til PSoC'en, hvor meget vægt der er på den, som svarer til den mængde væske, der er tilbage i beholderen.

4.3 DC-motorer

Der blev overvejet at montere baghjulene på en hjulaksel for derved at gøre baghjulene mere robuste, hvis vægten på Broomba pludselig blev øget fx. at nogen trådte på den. Drejningen af Broomba skulle varetages af en servomotor monteret på forhjulet, der både kunne dreje og køre selv. Det viste sig at være for besværligt at montere en hjulaksel samt at konstruere forhjulets funktioner med de motorer, vi havde til rådighed og besværliggjorde Broombas drejfunktion unødvendigt.

Til baghjulene blev i stedet anvendt to 6V DC-motorer én til hvert baghjul, som styrer fremdrift og drejning af Broomba. I starten var det tænkt at drejning skulle foretages af en servomotor på monteret på forhjulet, der kunne dreje til den ønskede vinkel. Det men det viste sig at være langt nemmere at implementere drejningen vha. et hjul fra en kontorstol som forhjul og øge PWM signalet på den ene motor ifht. den anden motor, så den kører hurtigere til den højre side eller til den dreje venstre side.

Til skrupperen anvendes en 6V 120 RPM DC-motor, som bare skal køre med de 120 RPM og derved blot have en 6V forsyning.

4.4 Karosseri

Karosseriet var i starten tiltænkt som en firkantet kasse, med nogle switches på, som skulle simulere fronttrykkene, da det var den nemmeste løsning på et karrosseri.

Det blev besluttet at 3D printe karosseriet, da det gav mulighed for den ønskede form. Printet bestod af en masse små dele, som et sammesæt, hvorefter det blev samlet da alle dele var blevet printet.

Da karosseriet er af plastik, og 3D-printet, gav det mulighed for at fastgøre de dele der ikke skulle kunne udskiftes.

4.5 Pumpe

Pumpen var i starten tænkt som en dyse, der skulle sprøjte sæbevand foran Broomba i doseret portioner foran børsten, når Broomba vasker gulv. Denne funktion kræver tilførelse af sæbevand fra beholderen som kontrolleres via en mekanisme, der kan åbne og lukke for tilførselen af sæbevand. Til formålet blev der overvejet en servomotor, der kunne klemme på en gummislange, men det blev vurderet at denne løsning var for utæt og ville give læk af sæbevand, hvorfor der i stedet blev hjemkøbt en 6V DC peristaltisk pumpe til formålet. [14].

4.6 SPI-kommunikation

Til brug til kommunikation mellem PSoC og RPI anvendes SPI-kommunikation. Dette skyldes for det første, at SPI-kommunikationen er let at skalere, således at der kan nemt kan tilføjes flere PSoC-slaveenheder, hvis der er brug for det. Derudover er SPI også funktionelt skalerbart, da det understøtter full-duplex. Bliver der behov for dette, kan det således lade sig gøre. For det andet er udviklingen af en SPI-driver en del af et andet af semestrets kurser, så det er oplagt at bygge videre på den.

4.7 GUI

Brugergrænsefladen og forbindelsen til en tablet vha. en applikation var først tiltænkt, men vi kunne konkludere at det var for meget arbejde. Løsningen var at lave en webserver, der kan tilgås fra alle enheder med internet forbindelse. Da vi lægger mere vægt på funktionen af brugergrænsefladen og forbindelsen til produktet, end hvilken enhed der bruges til at monitorere produktet. Derudover er det gjort overvejelser om GUI og Control klassen skal være sammen eller hver for sig, spørgsmålet var om vi skulle kunne tilgå controller klassen uden at tilgå GUI'en.

4.8 Overvejelser om implementering af software

Da både undervisningen i HAL og GFV har bestået af C-programmering, har vi valgt at skrive vores kode i C. Derudover er C også tættere på hardware end C++, hvilket fik os fast besluttet på at skrive i C. Vi har også haft overvejelser om, om hhv. målingen af batterispænding og vandbeholderen skal sættes op som interrupt, men der har vi valgt at bruge polling, da det vurderes, at høj responstid ikke er nødvendigt på nogen af disse. Begge dele polles således med et fast interval.

5 Hardware Design

I dette afsnit vil vi komme indover de forskellige hardware dele, og deres funktioner.

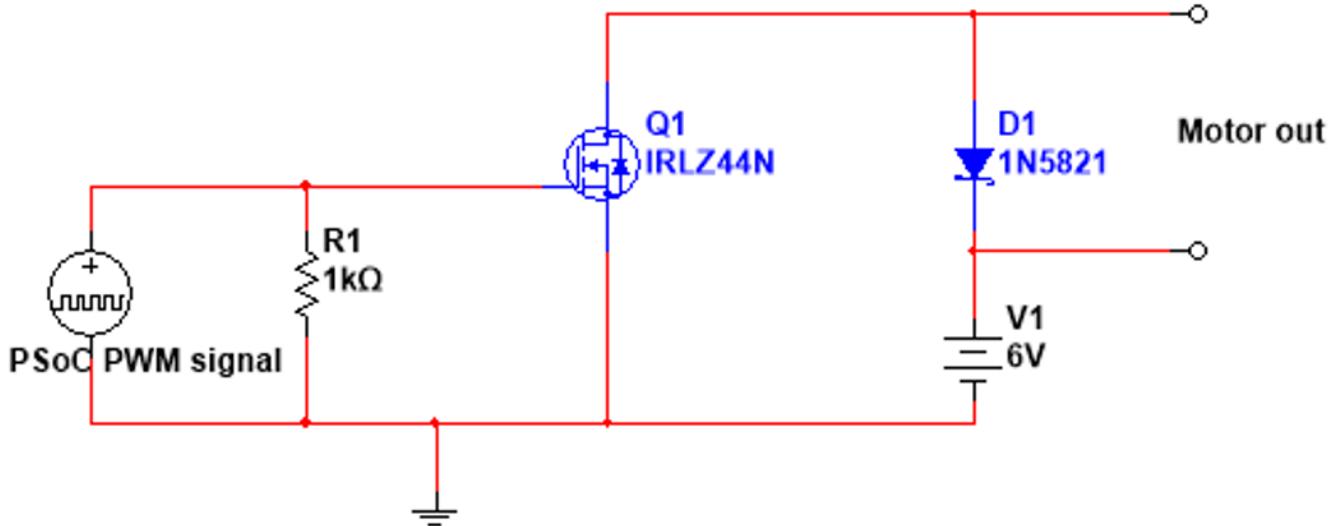
5.1 Motor system

Under motor system, vil vi forklare omkring designed af Pumpen der bruges til at pumpe væsken fra beholderen og ned på gulvet. H-broen, som bruges til at styre driftmotorene, samt en DC motor der bruges til at dreje nogle børster rundt på gulvet til vaskningen.

5.1.1 Pumpe til vand

Pumpen til at fragte vandet fra beholder til gulvet er en 6V DC DIY Dosing pump [14], som vi giver et fast PWM signal når broombaen kører. Dette PWM signal vil blive sat til at passe til mængden af vand, vi gerne vil have ud. Et PWM signal på 40% svarer til ca. 20RPM, som giver os et flow af vand på 20 mL/min, som vil svare til at Broombaen kan vaske gulv i 25 min, med 500mL væskebeholder. De 20mL/min er samtidig det mindste vand flow pumpen kan give i følge de specifikationer vi har kunnet finde på den [14]. Dette er dog alt for meget til det behov som Broomba har. Dette løses ved at sætte pumpen til at køre i intervaller. Tændt 1 min, slukket 2 min. Ved at gøre dette kan Broombaen vaske gulv i 3 gange så lang tid, og derved dække et større areal.

PWM signalet bliver kørt igennem en MOSFET IRLZ44[7] som regulerer hastigheden på motoren, i forhold til hvad PWM signalet er sat til.

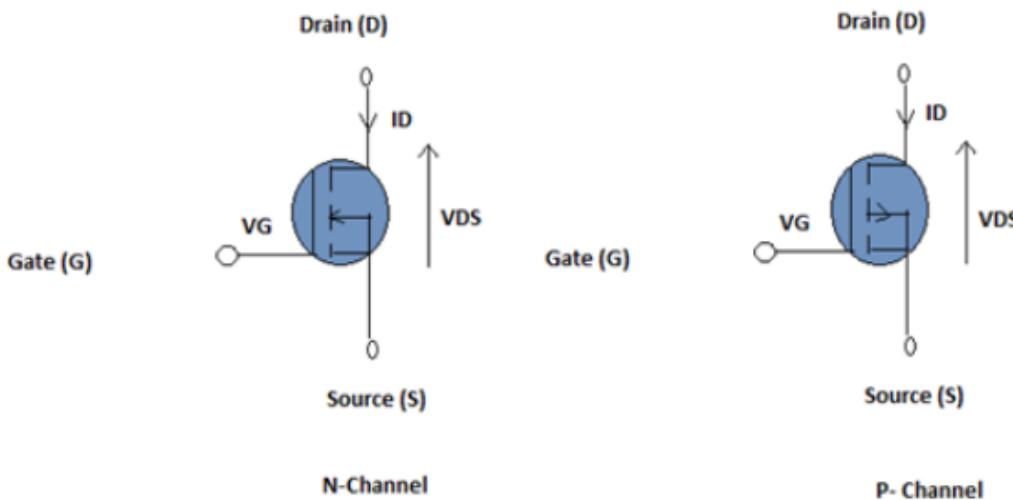


Figur 30: Multisim diagram over Pumpeprint

5.1.2 H-Bro

mosFET's

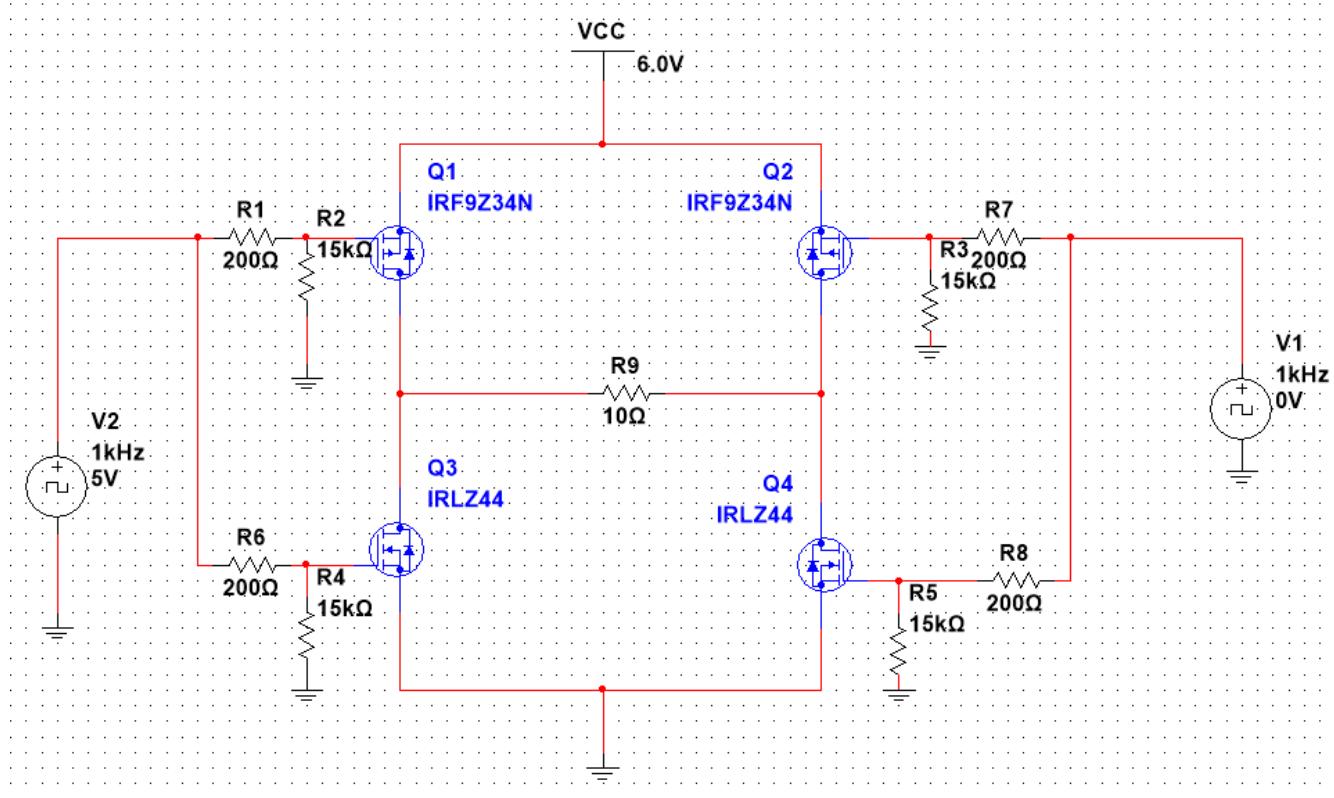
MosFET er en enhed som er vidt benyttet som en åbne og lukke mekanisme, eller til at forstærke elektroniske signaler i et givent apparat. MosFET'en har fire terminaler som er Source(S), gate(G), drain(D) og body(B). Bodien er typisk forbundet med source terminalen, således at den fungere som en tre vejs terminal. MosFET'en er langt den mest brugte transistor både i analoge og digitale kredsløb. MosFET virker på den måde, at man elektronisk varierer længden af dens kanal vha. spændingen som så også kontrollere åbningen af den. Dette resulterer at der kan løbe ladninger igennem fra source til drain alt efter reguleringen.



Figur 31: Et diagram over de to forskellige mosFETS[13] typer

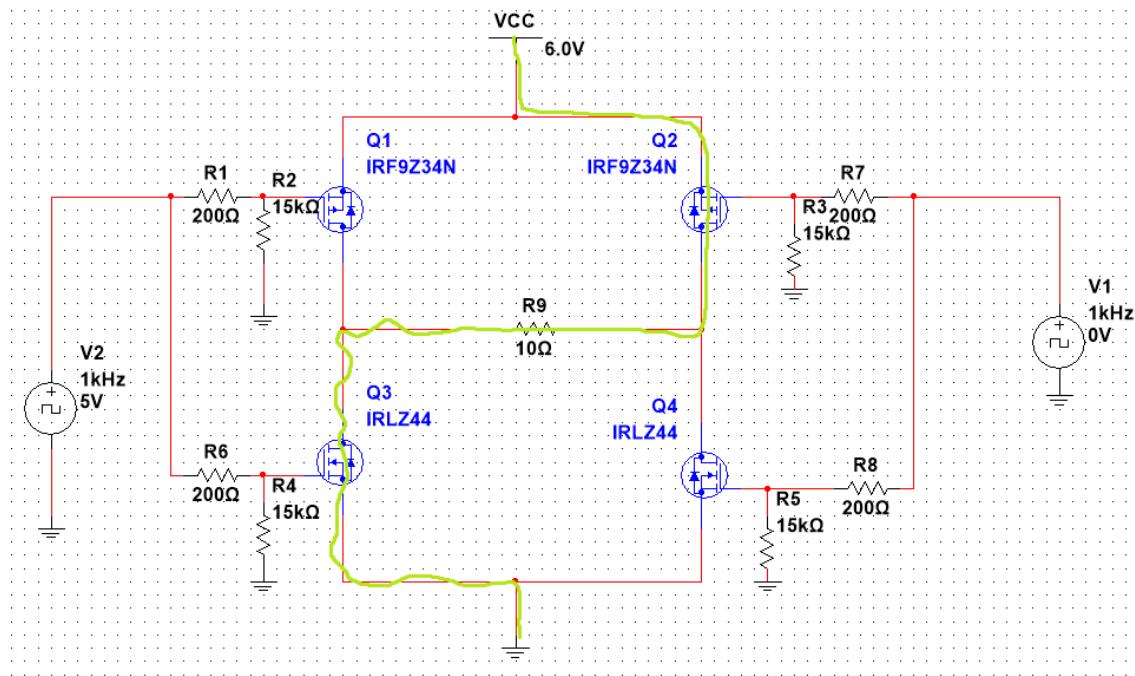
Hos mosFET'en er der tale om to forskellige typer N og P kanaler. P kanalen fungere på den måde, at får den ingen spænding så vil source og drain åbne op således at der vil løbe en ladning igennem. N kanalen fungere lige modsat og er kun åben hvis den får tilstrækkelig spænding. Den type der er valgt til netop dette design er valgt ud fra at $V_{GSTH} < 5V$, hvilket den er hvis men læser det vedlagte datasheet for IRF9Z34N [10] og IRLZ44N [7].

Design

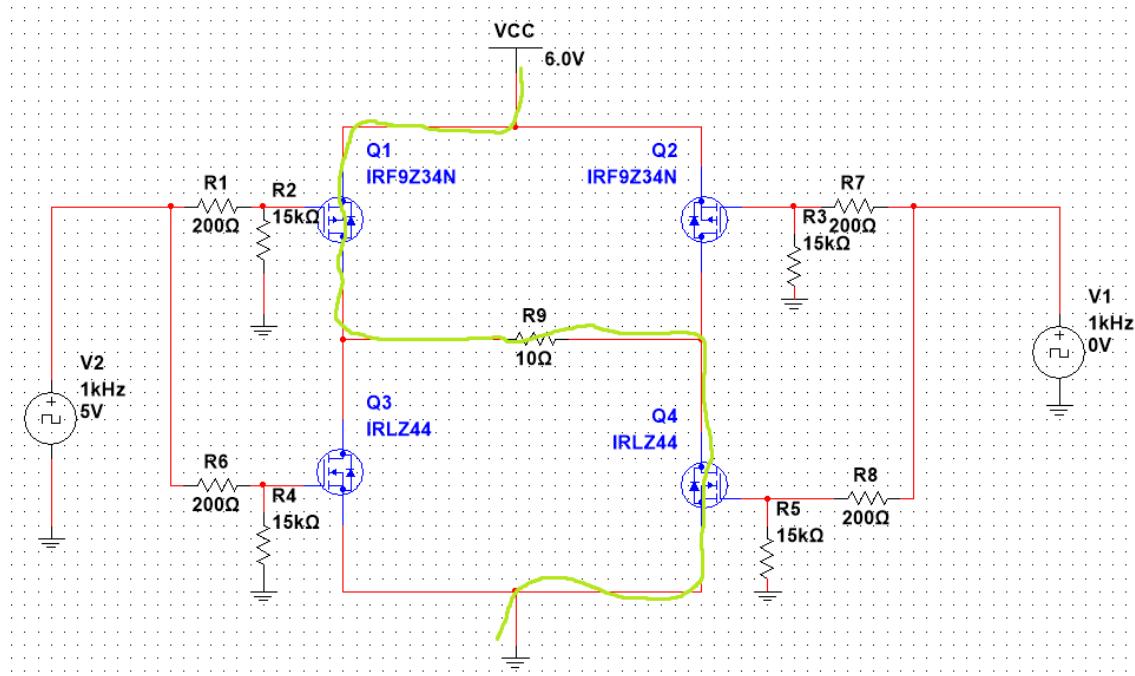


Figur 32: Et design af H-broen der er blevet brugt og illustreret i multisim

Når V2 er åbent og V1 er slukket, så kører motoren baglæns [9]. Det vil sige at Q1 er lukket og Q3 er åbent som vist på figur 33, da der er tale om P og N kanaler. Så vil de altid være åbent modsat af hinanden, når de er koblet som på. Er V1 tændt og V2 slukket, så vil den kører fremad, som vist på figur 34, og gøre det modsatte af hvad der allerede er beskrevet. Altså er Q2 lukket, Q4 åben og det hele kan kører på tværs.



Figur 33: Et design af H-broen som er blevet brugt, dette er illustreret i multisim.



Figur 34: H-broen med ladningens løberetning der er vist med en grøn streg.

Modstande

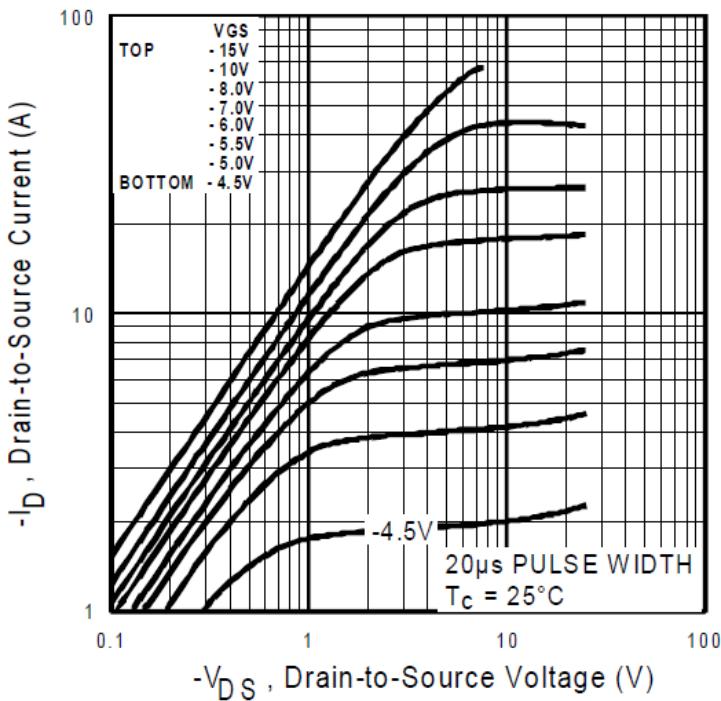
For at beskytte PSoC'en er der valgt at sætte en modstand på efter PSoC'en og før mosFET'en. Kigger man på databladet for PSoC'en så kan en pin på PSoC'en kun klare $25mA$ det vil sige vi er interesseret i at begrænse den strøm som kan løbe tilbage.

$$R = \frac{V}{I} \quad (1)$$

$$R = \frac{5V}{25mA} = 200\Omega \quad (2)$$

Design forklaring

I virkeligheden er designet af H-broen lavet således at det er blevet overdimensioneret. Dvs. at mosFET'en er administreret på den måde, at den kan klare op til 4 ampere, men motoren skal slet ikke bruge så meget. Dog er der ingen idé om hvor meget strøm den skal bruge, da der ingen tilhørende datasheet er for den. Derfor er designet lavet således, at motoren selv tager den strøm den har brug for. På den måde kan man anse motoren som en variabel modstand, der varierer strømmen som den har behov for, alt efter motor belastningen. Kigger man på figur 35, kan man se at hvis den ikke overstiger 1 A, så vil spændingstabet over mosFET'en ikke være mere end 0.1V. Det vil gøre at man mere eller mindre, kan anse det som et spændingstab på 0V. Derfor er det vigtigt at vide hvor stor en belastningsstrøm der når moteren oplever maksimal modstand. Da man på den måde så vil kunne få en konstant 6V spænding over modstanden.



Figur 35: Her ses Drain to source current som funktion af voltage[10].

For at være sikker på at den ikke overstiger 1 Ampere, er motorne blevet testet vha. manuel modstand. Dvs at der er blevet brugt en hånd til at øge belastningen på den, således at den vil øge strømmen til denne. Den belastning der er blevet anset som fornuftig er blevet vurderet efter bedste evne.



Figur 36: Her ses den maksimale belastningsstrøm.

Kølelegeme

Det er også relevant at undersøge, om hvorvidt der skal et kølelegme på flere mosFET's. Dette kan gøres ved at undersøge hvor meget effekt der bliver omsat, og om dette stemmer overens med hvad mosFET'en kan klare. Her kræver det $200mA$ at starte motoren med, men væsentlig mindre at holde den kørende. Derfor er I_{motor} faldet på $300mA$ bare for at være sikker at den kan håndtere den nødvendige effekt. Desuden er der valgt en pulldown modstand, der sørger for at mosFET'en ikke svæver. Hvis den er i den tilstand har den med at opsamle interferens, hvilket gør at den kan en hvilket som helst spænding, som gør at den er åben, lukket eller delvist åbent.

$$I_{Motor} = 300mA$$

$$R_{DsOn} = 0.10\Omega$$

$$T_{MaxJunction} = 175^{\circ}C$$

$$T_A = 25^{\circ}C$$

$$R_{\theta JA} = 62^{\circ}C$$

$$Pulldown = 15k\Omega$$

Først er det nødvendigt at vide hvilken effekt motoeren drives med, og da strømmen for den allerede er fastlæjt, kan resten findes i datablade.

$$P_{Used} = R_{DsOn} * I_{Motor}^2 = 9mW \quad (3)$$

For at vurdere om den kan håndtere denne effekt, så skal datasheet'et for mosFET'en undersøges, hvor Max temperaturen for junction bruges. Desuden skal junction til ambient celcius pr watt benyttet. Alt dette giver følgende udtryk:

$$P_{max} = \frac{T_{MaxJunction} - T_A}{R_{\theta JA}} = 2.419W \quad (4)$$

Kigger man på de 2 resultater for hhv equation 3 og equation 4 så er den brugte effekt ikke i nærheden af den maksimale effekt der kan bruges. Derfor er det ikke nødvendigt med et kølelegme.

5.1.3 DC motor skrubber

Der er ingen hardware til Skrubber motoren, da det er en 6V 120RPM motor, og vi gerne vil have at den vasker gulv med den hastighed som den har ved fast forsyning.

DC motoren får derved bare en forsyning fra spændingsregulator printet på 6V.

Motoren bruger **ampere** uden belastning, og **ampere** ved en lille belastning, som skal illustrere belastningen når der bliver skrubbet på gulvet.

5.1.4 Stykliste

- 8 x Modstande 200Ω

- 1 x Modstand $1\text{k}\Omega$
- 8 x Modstande $15\text{k}\Omega$
- 1 x 1N5821 Power diode
- 4 x IRF9Z34N[10]
- 5 x IRLZ44[7]

5.2 Batteri

Valget på batteriforsyningen var en powerbank vi fik udleveret fra elektronikværkstedet. Ifølge powerbankens påklistrede oplysninger, er batteriets nominelle spænding på 9.6V, og nominel kapacitet på 2700 mAh. Af praktiske grunde valgte vi at bruge det ene batteri: Kun et batteri skal genoplades, og fylder mindre.



Figur 37: 9.6V batteri

Da PSoC'en højest tåler 5.5V [17] og RPI højest tåler 5.1V [15], og motorerne drives med 6V skulle vi derfor designe 2 spændingsregulatorer.

Efter at have gennemgået værkstedets udvalg af spændingsregulatorer, valgte vi LM317T til DC motorerne og L7805SCV til microcontrollerne.

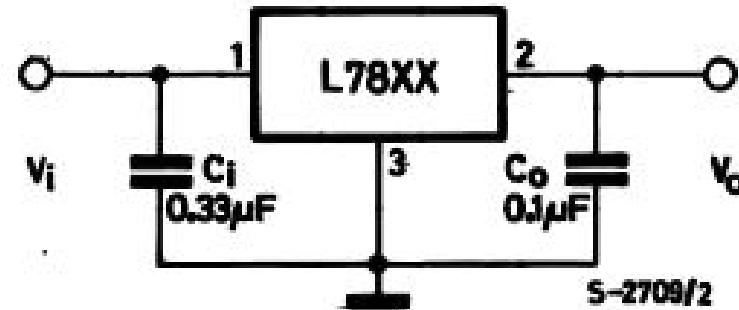
5.2.1 Kort om spændingsregulator

En spændingsregulator modtager en DC spænding på indgangen og omsætter den til en DC spænding på udgangen. Vores 9.6V batteri har en spænding på 11.28V når det er fuldt opladt. En typisk opstilling med en serieregulator nedsætter spændingen fra en ustabil effektforsyning til en stabil og kortslutnings-sikker forsyningslinje- Udgangen

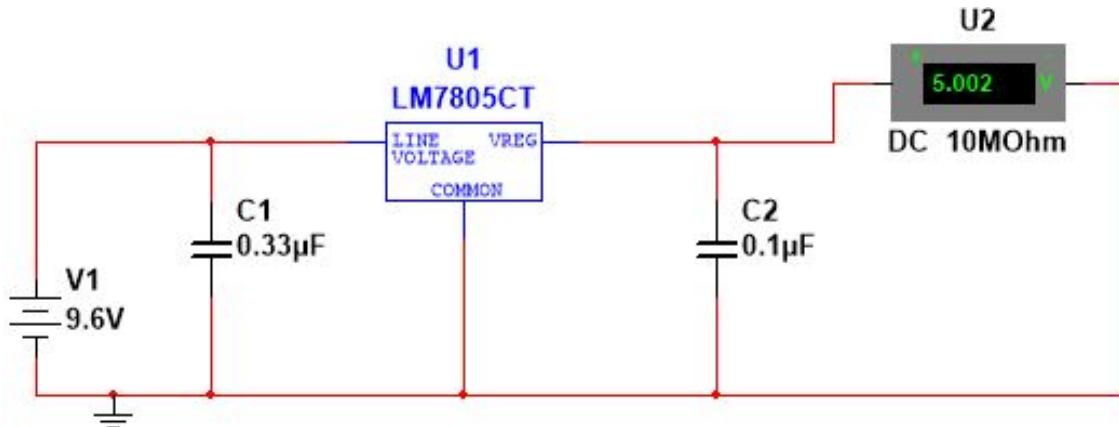
får en fast spændingsværdi, hvilket er især vigtigt i vores PSoC og RPI, der kan brænde af hvis de får over 5V spænding. Kondensatorerne er derfor krævet af hensyn til stabilitet. Afhængig af formålet, anvendes passende kondensatorer med tilhørende passende værdier, som vi finder i regulatorernes respektive datablade. [8]

5.2.2 5V spændingsregulator

I dette afsnit ser vi kort på 5V spændingsregulatoren L7805SCV. L7800-serien kan levere faste spændingsværdier på udgangen. L7805 leverer en fast spænding på 5V og kan levere op til 1.5A på udgangsstrømmen, hvilket gør den ideel til forsyning af vores 5V microcontrollere [11].



Figur 38: L7805 spændingsregulator fra datablad
[11]



Figur 39: Multisim simulation - 5V spændingsregulator

Selvom en kondensator strengt taget ikke er nødvendig for udgangens stabilitet, forbedrer kondensatoren transientesponsen [11]. Det er vigtigt for vore microcontrollere, da de tåler højst 5.5V. Simulationen i Multisim viser, at udgangsspændingen (repræsenteret ved U2) nu er på 5.002V, hvilket er acceptabelt til vores formål. Vi kan derfor bygge regulatoren på et print.

5.2.3 6V spændingsregulator

For at forsyne motorerne med 6V, anvender vi spændingsregulatoren LM317. LM317 er en justerbar regulator, som forsyner op til 1.5A og udgangsspænding i spændeviddet 1.25V-37V. På udgangen sidder der en modstand med en fast værdi på 240Ω , og en justerbar modstand, der regulerer udgangsspændingen.

Vi har taget inspiration fra LM317Ts datablad Figur 9 [12]. Den justerbare modstand R_2 i kredsløbet bestemmer udgangsspændingen. [12] Udgangsspændingen findes ved:

$$V_O = V_{REF}(1 + R_2/R_1) + (I_{ADJ} * R_2) \quad (5)$$

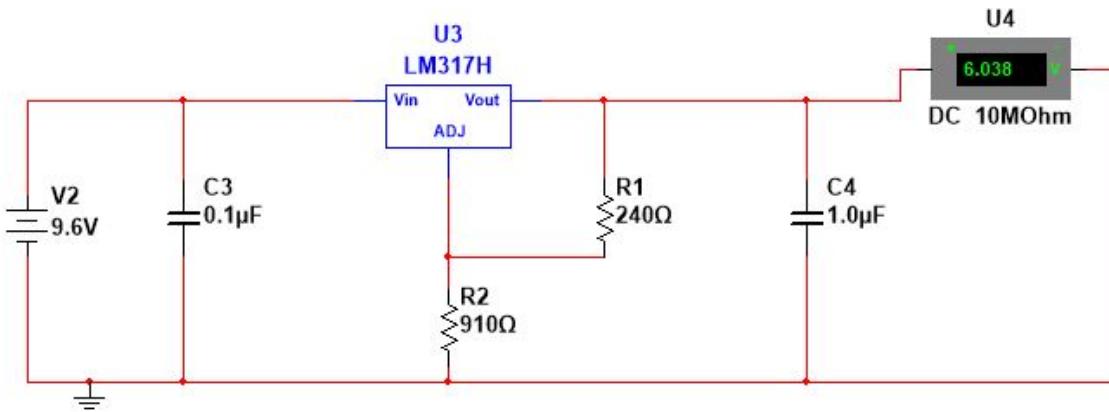
Da I_{ADJ} typisk er på $50\mu A$, og er ubetydelig i de fleste applikationer, udelades den fra udregningen. Vi regner os frem til R_2 ved at anvende tilgængelige modstandsværdier fra værkstedet:

$$\begin{aligned} V_R &:= 1.25 \text{ V} & R_1 &:= 240 \Omega & R_2 &:= 910 \Omega \\ V_O &:= V_R \cdot \left(1 + \frac{R_2}{R_1}\right) \\ V_O &= 5.99 \text{ V} \end{aligned}$$

Figur 40: Udregning i MathCAD Prime 4.0

Modstand R_2 skal derfor være på 910Ω , da V_O er på 5.99V, som er tættest på den tilstræbte værdi på 6V.

Nu kan kredsløbet endelig simuleres i Multisim:



Figur 41: Multisim simulation - 6V spændingsregulator

Simulationen i Multisim viser nu en udgangsspænding på 6.038V, hvilket er nok til DC motorerne.

5.2.4 Spændingsdeler

Da batteriets nominelle spænding er på 9.6V, og 11.28V fuldt opladet, har vi besluttet at implementere en spændingsdeler Forrest i kæden på printet. Det skyldes, at PSoC, som skal måle batteriets spændingsstatus, har en maksimal indgangsspænding på 5.5V, og en spændingsdeler er derfor nødvendig. [17]. Det matematiske udtryk for spændingsdeler er Kirchhoff's Spændingslov (KVL): [8]

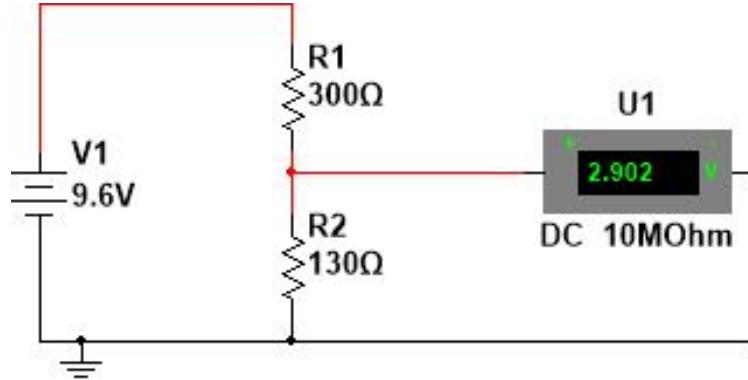
$$V_O = \frac{R_2}{(R_1 + R_2)} * V_{in} \quad (6)$$

Vi tilstræber en udgangsværdi under 5.5V:

$V_{in} := 9.6 \text{ V}$	$R_1 := 300 \Omega$	$R_2 := 130 \Omega$
$V_{out} := \frac{R_2}{(R_1 + R_2)} * V_{in} = 2.902 \text{ V}$		

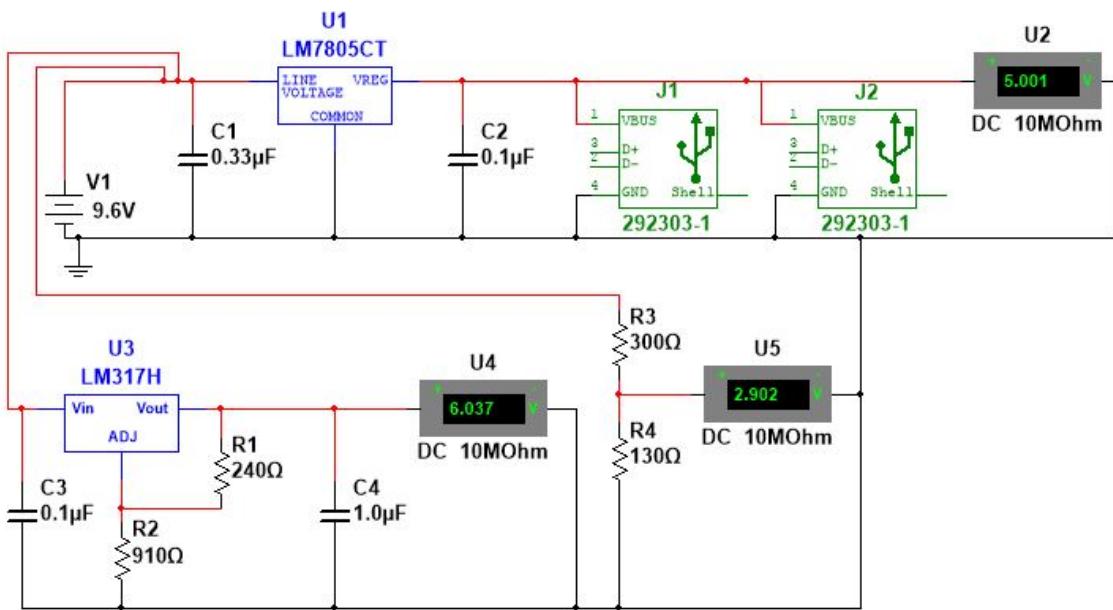
Figur 42: Spændingsdeler udregning i Prime

2.902V er under 5.5V, og vi kan nu simulere i Multisim.



Figur 43: Multisim simulation af spændingsdeler

Simulationen viser samme spænding som det analytiske udtryk. Vi samler alle 3 kredsløb i et kredsløb og simulerer i Multisim:



Figur 44: Multisim simulation af vores print

Multisim simulationen viser, at printet kan realiseres. Vi kan nu samle spændingsregulator og spændingsdeler i et testprint. Det kan ses i Modul og integrationstest-dokumentet.

5.2.5 Stykliste

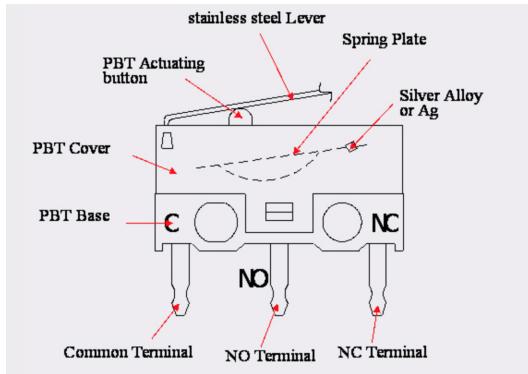
- 1 x 9.6V batteri
- 1 x $0.33 \mu F$ kondensator (C1)
- 1 x L7805SCV spændingsregulator
- 2 x $0.1 \mu F$ kondensator (C2 og C3)
- 1 x LM317T spændingsregulator
- 1 x 240Ω modstand (R_1)
- 1 x 910Ω modstand (R_2)
- 1 x $1.0 \mu F$ kondensator (C4)
- 1 x 300Ω modstand (R_3)
- 1 x 130Ω modstand (R_4)

5.3 Front tryk system

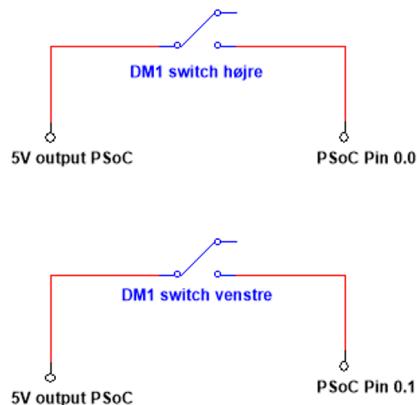
Front trykket et lavet som en DM1[6] switch i hver side.

Da det er et 5V signal vi gerne vil have tilbage på PSoC'en, får PBT base benet på DM1[6] 5V, og derved når den bliver aktiveret ved et tryk fra omgivelserne, sender den et 5V signal tilbage på NO terminalen. Som er den udgang vi bruger fra switchen. NO betyder at den er normalt er åben, som vist i Figur 46

Da switchen blot skal styre et signal, behøver vi ikke tænke på om den kan håndtere en bestemt strøm, men ved 5V DC som vi bruger, kan den holde til 30mA, hvilket den langt fra vil blive belastet med.



Figur 45: DM1 switch



Figur 46: Multisim diagram over Fronttrykkene

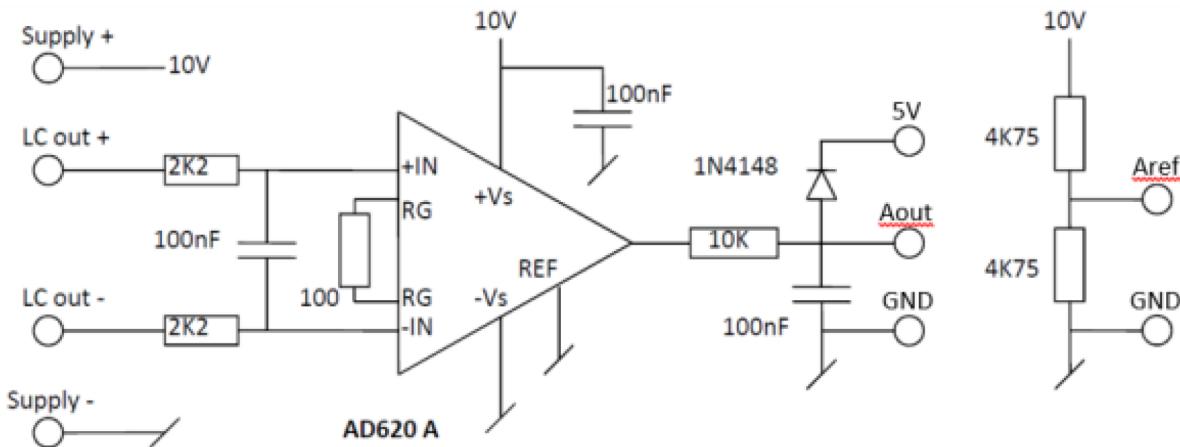
5.3.1 Stykliste

- 2 x DM1[6] switch
- 3 x pins på PSoC

5.3.2 Load cell

Load cellen er et færdigt produkt, der er blevet taget fra en øvelse i faget ”Grænseflader til den fysiske verden”. Her er der erfaret at den kan bruges til at forstærke en meget lille spænding til en tilpas stor spænding, således at det kan benyttes i ADC'en på PSoC'en. Her skulle det så være muligt at måle vægten, efter passende kalibrering og dermed mængden af væske i Liquidcontaineren. I følge øvelses vejledningerne [3] & [4] har vi brugt den tidligere erfaret viden, til at finde ud af præcis hvad vi skal bruge den til i vores projekt.

På Figur 47 ses en skabelon over kredsløbet. Og Aref bruges til radiometrisk måling.



Figur 47: Diagram over Load cell print

5.3.3 Stykliste

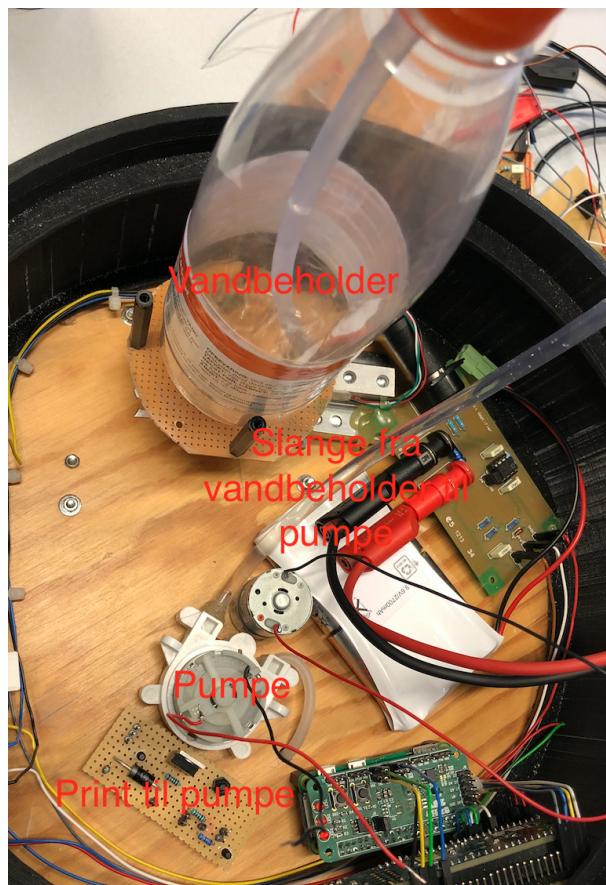
- Load cell 1kg
- AD620
- Modstand 100 Ω
- 2 x Modstand 2.2k Ω
- 2 x Modstand 4.75k Ω
- Modstand 10k Ω
- 3 x Kondensator 100nF
- 1N4148 diode

6 Hardware Implementering

6.1 Pumpe til vand implementering

Hardwaren til den peristaltiske pump, er implementeret med et print til styring af pumpen. En PVC slange som er limet fast til låget af en 500mL flaske fra indgangen af pumpen. Og fra udgangen af pumpen en slange der bliver ført ud gennem bunden af broomba. Printet er lavet med inspiration fra en GFV øvelse [5]

Alle tingene er enten limet fast eller skruet fast med bolt og møtrik, så det er stabilt når broomba køre ind i væggen og giver et bump.



Figur 48: Pumpe, print & væskebeholder

På Figur 49 ses slangen der går fra Pumpen og ud til gulvet.



Figur 49: Udgangen på slangen set fra bunden af broomba.

6.2 H-bro implementering

Efter at der er blevet lavet en velfungerende H-bro på fumlebræt blev der lavet en kredsløbtegning, som kan findes under vedlagt bilag. Herfter blev der loddet 2 test broer ud fra tegningen, hvor det ene virkede og det andet ikke. Til sidst blev der loddet et tilsvarende kredsløb af det der virkede, på samme print, således at hver motor har deres egen H-bro. Printet blev skåret til, så den havde en god pasform i Broombaen. Resultatet af dette kan ses i figur ??



Figur 50: To H-broer på samme print

6.3 DC motor skrubber implementering

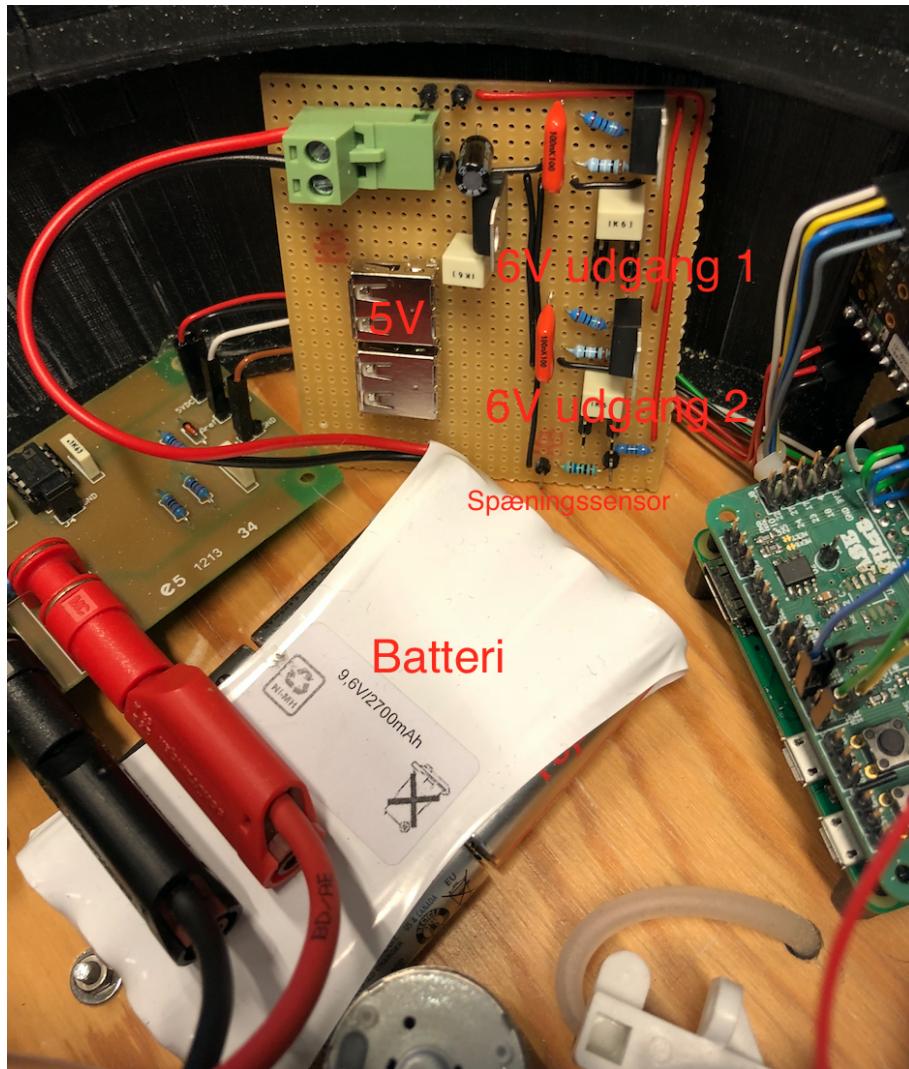
Skrubberen er en DC motor der er skruet fast til bunden af broomba med aksen igennem bundet. Der er her lavet en forlænger til akslen, med et T-stykke på hvor skrubber delen bliver monteret på som vist på Figur 51. DC motoren har ikke noget hardware lavet til sig, da den blot skal have fast 6V spænding.



Figur 51: Skrubber fra undersiden af DC motoren

6.4 Batterier implementering

Vores batteri giver en nominel spænding på 9.6V. Den forsyner vores prints og microcontrollere med deres respektive påkrævede spænding. Da de forskellige elementer af vores system kræver forskellig spænding, har vi bygget 3 spændingsregulatorer: 2 med 6V udgang, og 1 med 5V udgang. Selve batteriet er monteret centralt på bunden af karosserigulvet. Batteriets grønne Euroblock-stik anvendes, når batteriet skal genoplades med opladeren som fulgte med batteriet.



Figur 52: 9.6V batteriet. Stikket er sat til vores spændingsregulator prototype.

6.5 Front tryk system implementering

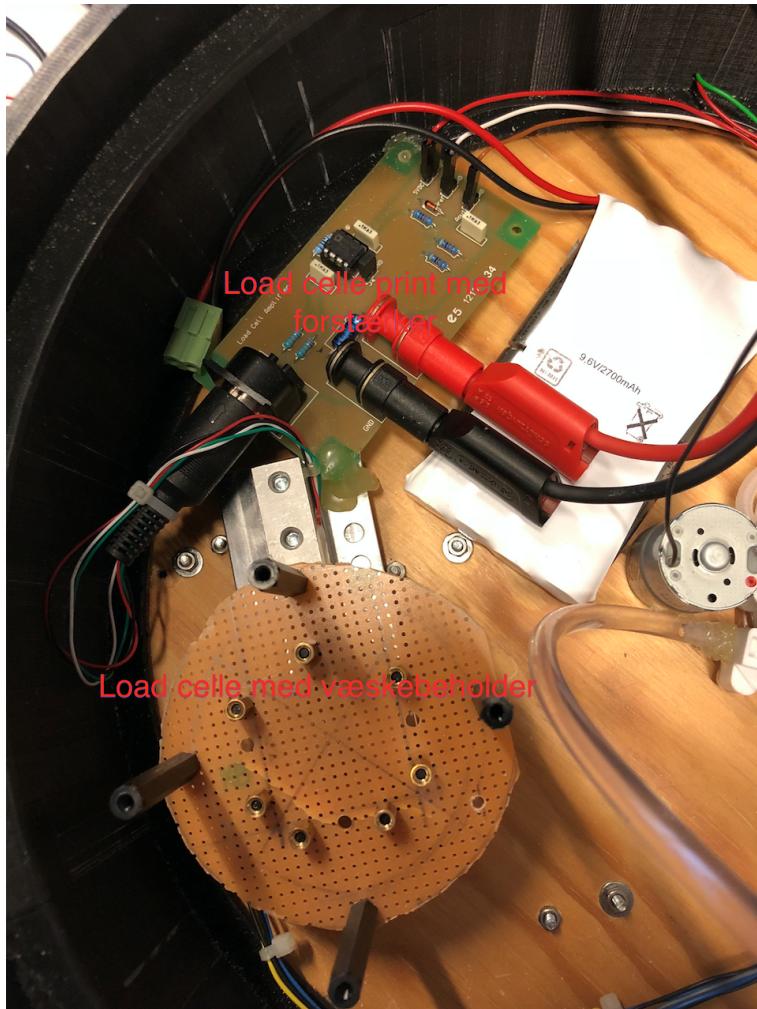
Front trykene sidder på ydersiden af karosseriet, og bliver påvirket af et tryk af en bøje der sidder 2 cm fra ydersiden af karosseriet. Dette gør at kontaktfladen vil dække et større område så broomba'en kan køre ind i ting med flere vinkler.



Figur 53: Front tryk med indgang og udgang signal

6.6 Load cell implementering

Load cellen er implementeret ved at selve loadcellen er skruet fast til bunden af broomba for stabilitet, og printet er blevet limet fast til kanten også for stabilitet. Der er herefter lavet en platform til væskebeholderen. Load cell printet er ”købt” fra anden part.



Figur 54: Loadcelle med platform og print

7 Software Design

Softwaredesignet vil fortsætte N+1 viewet, hvor arkitekturen slap. I denne del vil Process View, Data View og Deployment View blive anvendt og præsenteret.

7.1 Process view

Dette view omhandler de forskellige processer på de enkelte CPU'er og deres ansvarsområder samt kommunikationen mellem CPU'erne.

PSoC:

PSoC'en håndterer alle sensorer og aktuatorer. Frontsensorene består af to tryksensorer på Broombaen til at registrere, når Broombaen kører ind i en forhindring.

Derudover styrer den en vægtsensor i form af en loadcell, der kan måle op til 1 kg. Loadcellen bruges til at måle vægten af væskebeholderen, så der gives besked når beholderen er ved at være tom.

Dysen til væsken, der sprøjtes på gulvet inden der bliver skrubbet, er også styret af PSoC'en.

Der er to motorer, der styrer hvert sit hjul, samt en DC-motor der er sat på skrubberen til at vaske med. Sluttligt er der også en pumpe, der også skal styres af PSoC'en.

Kommunikation og ansvar:

Vi har to forskellige kommunikationstyper i Broombaen. For det første er der SPI mellem RPI og PSoC. PSoC'en har tre kommandoer, den kan sende til RPI'en: Lav batterispænding, lav væskemængde samt når Broomba sidder fast. Derudover er der også kommunikation til webserveren fra RPI'en. PSoC'en har ansvar for alle eksternt forbundne komponenter, såsom motorer, sensorer og aktuatorer. Hvorimod RPI er ansvarlig for al kommunikation til og fra brugeren.

RPI

RPI'en har kun én tråd, men er interruptstyret. Den kan interruptes både fra PSoC'en og brugergrænsefladen. Broombaen kan startes samt slukkes fra brugergrænsefladen, dvs. interrupte på beskeder fra brugergrænsefladen. Derudover får den et interrupt, når der bliver sendt data fra PSoC'en. Da der kun er én tråd, er der ikke nogen prioritet på, hvilke interrupts der kommer først i tilfælde af, at der skabes flere interrupts på én gang.

7.2 Data View

Dette view omhandler den data, der opbevares og behandles i systemet, herunder hvilken data der skal sendes mellem hvilke enheder. Dette skal give et afsæt til Deployment View, der handler om hvordan forskellige enheder i systemet kommunikerer med hinanden.

7.2.1 Dataoverblik

Den data, der florerer i Broombas system florerer mellem PSoC'en og dens boundaries. Boundary-klasserne er de klasser, der interagerer med forskellige sensorer. Systemet opbygges således, at der ikke sendes variable mellem RPI og PSoC. Der skal i stedet implementeres en beskedstruktur til brug i mellem de to CPU'er, så RPI'en ved, hvordan de beskeder, den får, skal fortolkes. De inputs, PSoC'en får fra dens boundaryklasser ses i nedenstående tabel:

Navn	Type	Beskrivelse
batteryValue	int	En integer der indeholder batteriets nuværende spændingsniveau
liquidValue	int	En integer der indeholder batteriets nuværende sæbevandsniveau
leftFrontPushed	bool	En bool der bliver true, når venstre frontsensor trykkes ind
rightFrontPushed	bool	En bool der bliver true, når højre frontsensor trykkes ind

Tabel 18: Databeskrivelse

Værdierne på variablene skal ikke sendes rundt i systemet. I stedet bruges ovenstående variable til at lade PSoC'en vide, hvornår den skal sende hvilke beskeder.

7.3 Deployment View

Deployment View fokuserer på kommunikationen mellem systemets forskellige enheder. Fokus vil her være på protokoller i mellem disse enheder. Det vil således blive taget afsæt i hhv. applikationsmodellerne fra Logical View, der har klarlagt hvilke boundary-klasser, de enkelte controllere skal kommunikere med, samt Data View, der har klarlagt hvilken data, der skal sendes rundt i systemet.

7.3.1 Oversigt over kommunikationsforbindelser

Vha. Logical view kan følgende grænseflader findes, hvor der skal sendes information til CPU'erne:

- Half-duplex mellem RPI og PSoC
- Inputs fra frontsensorer til PSoC
- Inputs fra væskebeholders load cell til PSoC
- Inputs fra batterimåling til PSoC
- Inputs fra motorspænding til PSoC

7.3.2 Kommunikation mellem RPI og PSoC

Denne del skal være systemets primære kommunikationsforbindelse, idet den forbinder de to CPU'er, systemet indeholder. Broombas styringssoftware placeres på PSoC'en, så forbindelsen til RPI'en handler primært om at sende statusmeddelelser til brugergrænsefladen via RPI'ens wifi.

Beskedstruktur

Strukturen af beskederne er opbygget således, at kun det mest nødvendige bliver sendt mellem RPI og PSoC. Beskederne består af én char-værdi, som er listet i tabellen herunder:

Char	Beskrivelse
'B'	Batteriniveau lav
'V'	Væskeniveau lav
'O'	Broomba sidder fast
'I'	Broomba sidder ikke fast og mangler ikke batteri

Tabel 19: Besked struktur fra PSoC til Raspberry

Da der ikke benyttes flere slaveenheder, så vil der ikke indgå adressering i beskedstrukturen.

7.3.3 Inputs fra frontsensorer til PSoC

PSoC'en skal modtage inputs fra frontsensorerne, når disse aktiveres af tryk. Denne funktion ligges ind som interruptfunktion, så PSoC-programmet ikke spilder CPU-tid på polling af sensorerne.

7.3.4 Inputs fra væskebeholders load cell til PSoC

Driveren er en ADC der laver en måling på en strain gauge, hvor efter værdien konverteres til mV ud fra en referencespænding. Derefter bruger vi en formel fundet fra skalaen, til at konvertere fra mV til gram. Når vægten er tilstrækkelig lav gives der et signal til RPI'en.

7.3.5 Inputs fra batterimåling til PSoC

Batterimåling sker også vha. en AD-converter, der sender information om batteriets niveau til PSoC'en. Det er denne værdi, der gemmes som variablen "batteryValue".

7.4 Inputs fra motorspænding til PSoC

Broomba skal registrere, at den sidder fast via spændingen i motoren. Denne findes også vha. en AD-converter.

7.5 Implementation View

Implementation view er det sidste view fra N+1, der medtages i dette projekt. Dette view ligger på grænsefladen mellem design og implementering. Her vil der blive beskrevet hvilke pins, der bruges til hvilke formål, samt hvordan softwaren sættes op og allokeres på de enkelte CPU'er. Bemærk at selve implementeringen af koden ikke vil være vist her. Den kommenterede kode kan ses i bilagsmaterialet (Source kode).

7.5.1 Pinassignments

I dette afsnit sættes der fokus på pins, og hvad deres formål er ift. systemet. Beskrivelse af pins kan ses herunder:

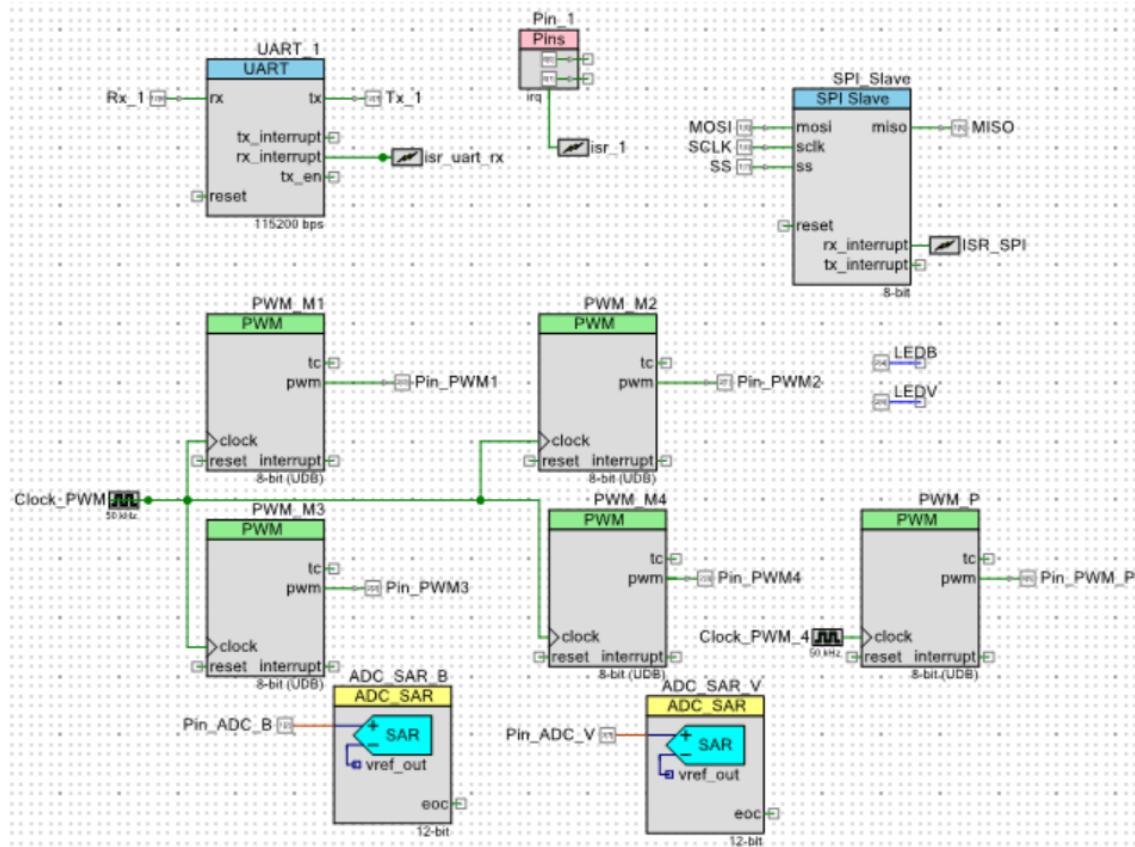
Navn	Pins	Beskrivelse
ADC_SAR_B: ExtVref	P0[2]	reference Pin til batterysensor
ADC_SAR_V: ExtVref	P0[4]	reference Pin til Væskesensor(loadcell)
Pin1[1:0]	P0[1:0]	pins til frontsensor 1 og 2
LEDB	P2[4]	Pin til batteri LED
LEDV	P2[5]	Pin til væske Led
MISO	P1[5]	Master input, slave output
MOSI	P1[6]	Master output, slave input
PIN_ADC_B	P1[2]	Pin til batteri ADC
PIN_ADC_V	P3[7]	Pin til væske ADC
PIN_PWM1	P2[0]	Pin til PWM_Motor_1
PIN_PWM_2	P2[1]	Pin til PWM_Motor_2
PIN_PWM3	P2[2]	Pin til PWM_Motor_3
PIN_PWM4	P2[3]	Pin til PWM_Motor_4
PIN_PWM_P	P0[5]	Pin til PWM_Pumpe
RX_1	P12[6]	Rx pin til UART
SCLK	P1[4]	Pin til serial clock
SS	P1[7]	Pin til slave select
TX_1	P12[7]	Tx pin til UART

Tabel 20: Pinassignment - PSoC

På RPI'en benyttes de pins, der er dedikeret til SPI-kommunikation på det extension module, der er udleveret fra skolens side ved semesterstart. Databladet ses i bilagsmaterialet(ase-fhat-schematic).

7.5.2 Topdesign

Vi bruger PSoC-creator til at programmere PSoC'en, hvilket er et stykke software der forenkler programmeringen. PSoC-creator er sat op som et "drag and drop" værktøj, hvor man finder den ønskede komponent i en søgeliste, og derefter placerer den på sit Top-design. Topdesignet for vores program ses i Figur 55.



Figur 55: Topdesign fra PSoC-creator

7.5.3 Websværver

Brugergrænsefladen sættes op som en webserver, der sættes op til at kunne give besked til brugeren, når væskebeholderen og batterispændingen er ved at være lav. RPI'en driver en webserver, der kan håndteres ved at skrive til den fil, der lægger på serveren.

7.5.4 Moduler - PSoC

Med afsæt i klassediagrammerne under logical view kan der opstilles følgende moduler, der skal implementeres på PSoC'en:

- MotorControl
- LiquidSensor
- LiquidControl
- BatterySensor
- RPICom
- Brush

Dette er de klasser i klassediagrammerne, der ikke er controllerklasser. Da der anvendes C-kode kaldes modulernes metoder direkte fra main-programmet, og der oprettes derfor ikke controller-moduler.

7.5.5 MotorControl

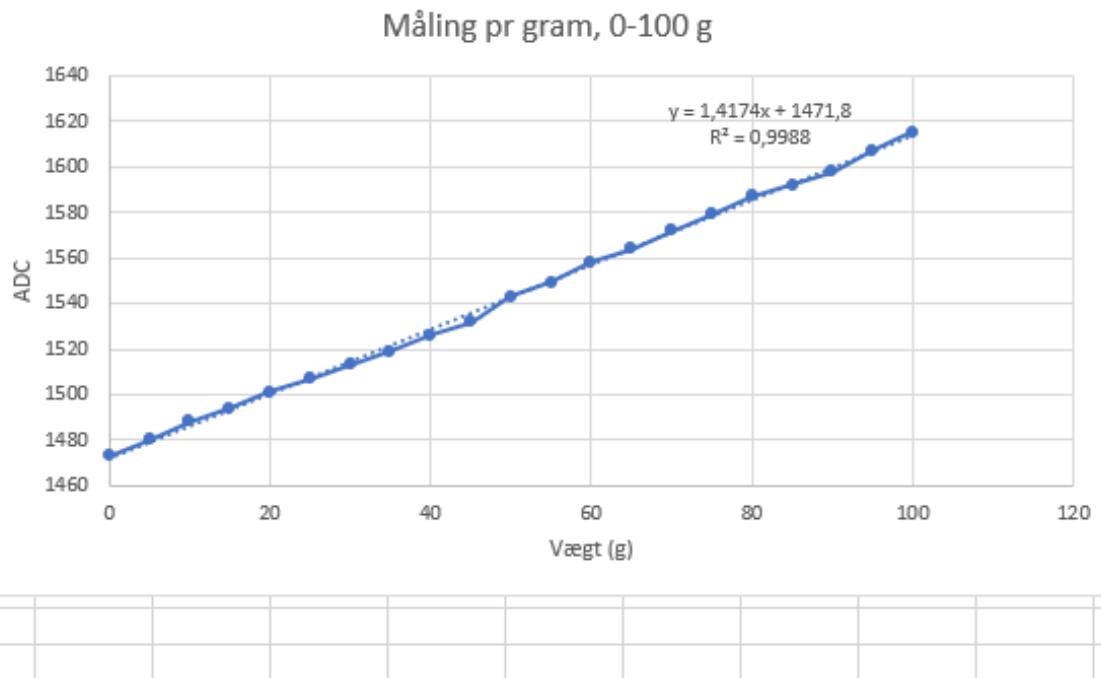
Dette modul har ansvaret for at styre Broombaens motorer. Det er et PWM-signal, der styrer hastigheden på motorerne, alt efter hvilken duty-cycle PWM-signalet har. For at styre retningen på motoren bruges en H-bro, som skal have et signal til at ændre retningen. Dette bliver lavet ved at sætte en pin høj og slukke for en anden. Derudover skal motordriveren også benyttes til at styre Broombas køreretning. Der laves ikke et justerbart førehjul, der skal kunne vælge Broombas retning. I stedet justeres retningen ved at give de to motorer forskellige hastigheder, så Broomba vil dreje mod den langsomme side.

7.5.6 LiquidSensor

PSoC'en bruger sin AD-converter til at konvertere den ADC værdi vi får fra en loadcell, som væskebeholderen er placeret på, til mV. Derefter konverteres mængden af mV til gram, ud fra en formel fundet ved kalibrering af loadcellen. Formlen viser, hvor meget loadcellen skal preloads med, inden den begynder at blive stabil i ADC-værdierne. Efter at vi har fundet preloadet laves der lineær regression på grafen for at komme frem til vores konverteringsformel.

Figur 56 viser grafen, samt formlen for loadcellens konverteringsformel:

Vægt (g)	ADC
0	1473
5	1480
10	1488
15	1494
20	1501
25	1507
30	1513
35	1519
40	1526
45	1532
50	1543
55	1549
60	1558
65	1564
70	1572
75	1579
80	1587
85	1592
90	1598
95	1607
100	1615



Figur 56: ADC's værdi pr. g vægt

På trods af frygten for, at grafen ville være ustabil, så kan man se, at der ikke er behov for at pre-loade loadcellen. Den lineære regression følger den blå linje meget godt og udtrykker følgende lineære formel: $y = 1.4174x + 1471.8$, hvor x er den målte masse i gram, og y er AD-converterens måling (ADC-måling) i forhold til offset. Det vil så sige, at hvis man ønsker at finde vægten, så hedder formlen $x = (y - 1471.8)/1.4174$, som også er den formel, der anvendes til at konvertere til gram:

```
//Convert to grams using linear function
float convertedResult = (result - 1471.8)/1.4174;
```

Figur 57: Codesnippet der sørger for at konvertere til gram

7.5.7 LiquidControl

Dette modul skal via et PWM-signal styre, hvor meget vand der pumpes ud på gulvet.

7.5.8 BatterySensor

Til at måle spændingen på batteriet anvendes en ADC på PSoC'en. Batteriets spænding justeres ned til 1/3 ved hjælp af en spændingsdeler, således at PSoC ikke overbelastes af en spænding på 9.6V fra batteriet, når det er fuldt opladt. Når ADC komponenten på PSoC læser en værdi, der er under 20% af spændingen på batteriet ved en fuld opladning, er spændingen på batteriet for lav til at Broomba kan fortsætte rengøringen tilfredsstillende, og der sendes efterfølgende besked til RPI om at slukke for Broomba.

7.5.9 RPICom

Dette modul skal facilitere SPI-kommunikation til RPI'en.

7.5.10 Brush

Dette modul skal via et PWM-signal styre, hvor hurtigt Broombas gulvskrubber skal rotere.

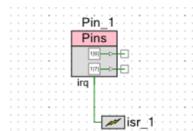
7.5.11 Interruptrutiner

Frontsensorer Sensorerne er forbundet til to kontakter på højre og venstre side forrest på Broomba. Hver kontakt er forbundet til PSoC med hver sin pin. Når kontakterne påtrykkes og kortsluttes, aktiveres en ISR på PSoC'en. Afhængig af om det er venstre eller højre kontakt, der påtrykkes, vil Broomba hhv. dreje til højre eller venstre. Sensorerne er active-high, så interruptet sættes op til at trigge på rising edge.

```
19 //Check which pin caused the interrupt by reading interrupt status register
20 if(Pin_1_INTSTAT & (0x01 << Pin_1_SHIFT))
21 {
22     turn = LEFT;
23 }
24 else
25 {
26     turn = RIGHT;
27 }
```

Figur 58: Kode for ISR

Højre og venstre kontakt er sat op på samme pin-komponent på PSoC. For at tjekke hvilken kontakt, der blev påtrykket anvendes linie 20 [Figur: 58] [16] på interrupt-komponenten isr_1 [Figur: 59] til at tjekke, om der blev fremkaldt interrupt på det første ben på pin-komponenten, hvor den vil dreje til venstre, ellers vil den dreje til højre.



Figur 59: Pin til interrupts

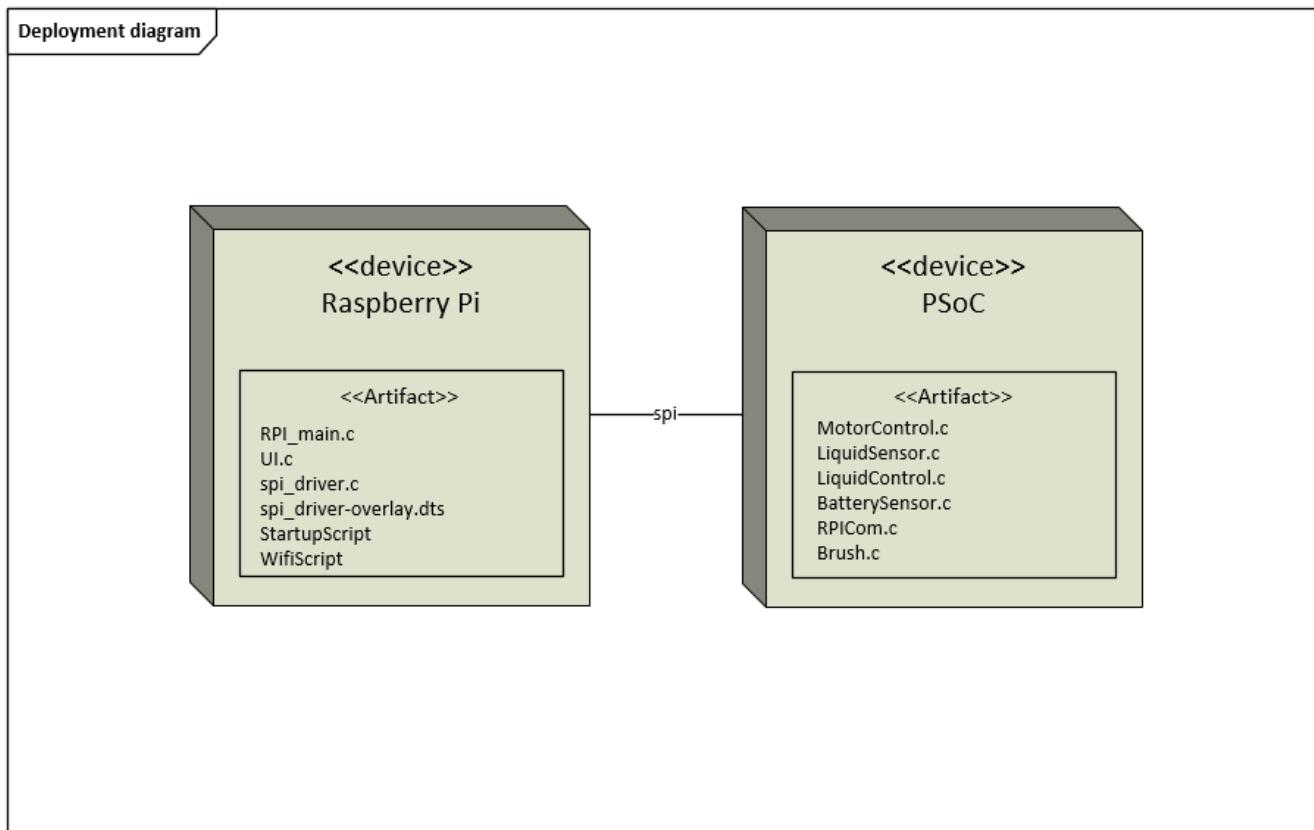
7.6 Moduler - RPI

På baggrund af RPI'ens samlede klassediagram afledes følgende moduler, der skal implementeres:

- UI

Derudover er der behov for hhv. en SPI-driver, et opstartsscript og et wifi-script. Disse er nødvendige for hhv. at muliggøre SPI-kommunikation på RPI'en samt sørge for, at RPI'ens program køres ved opstart, og at RPI'en forbindes til et trådløst netværk.

Allokeringen af softwaren på de forskellige CPU'er illustreres på deploymentdiagrammet i Figur 60.



Figur 60: Deployment diagram

SPI-driver

Bemærk at SPI-driveren bygger videre på et stykke kode, der er udleveret i undervisningen i faget Hardware Abstraktioner. Derfor er det ikke hele driveren, der er udviklet i af udviklingsteamet bag Broomba. Gruppen har selv skrevet read-funktionen og remove-funktionen. I readfunktionen anvendes metoder fra biblioteket linux/spi/spi.h. Til driveren laves en overlay-fil, så driveren kan skrives til RPI'ens device tree, således at den kan "hotplugges". Overlay-filen er vedlagt i bilagsmaterialet. På

Koden til de enkelte moduler er vedlagt i bilagsmaterialet i mappen ”Source kode”. Modultesten af de enkelte moduler kan findes i bilagsmaterialet i dokumentet ”Modul og integrationstest”.

Referencer

- [1] Kim Bjerge. *Specifikation Part 2 Use Cases*. URL: [Bilag/Projekt%20bilag/ISE/Use%20Cases](#).
- [2] Kim Bjerge. *SysML Behavioural Diagrams Sequence Diagrams*. URL: [Bilag/Projekt%20bilag/ISE/SysML%20Behavioural%20Diagrams%20-%20Sequence%20Diagrams](#).
- [3] Aarhus University school of engineering. *A/D D/A Lab experiment*. URL: [Bilag/Projekt%20bilag/Datablade/ADDA-Lab%20excercise](#).
- [4] Aarhus University school of engineering. *GFI Lab experiment*. URL: [Bilag/Projekt%20bilag/Datablade/Lab-ADC%20and%20signal%20conditioning](#).
- [5] Aarhus University school of engineering. *Lab eksperiment*. URL: [Bilag / Projekt % 20bilag / Datablade / Motorcontrol](#).
- [6] MULTICOMP. *Front tryk switch*. URL: http://www.farnell.com/datasheets/1662391.pdf?_ga=2.46451495.1020926447.1511351936-168769226.1494422046.
- [7] International rectifier. *MOSFET*. URL: https://www.mouser.com/catalog/specsheets/international%5C%20rectifier_irlz44n.pdf.
- [8] Tore Skogberg. *T-005 Analogteknik v. 2.0*. URL: <http://www.torean.dk/artikel/Analogteknik.pdf>.
- [9] Unknown. *H bridge Basics*. URL: <http://www.modularcircuits.com/blog/articles/h-bridge-secrets-h-bridges-the-basics/>.
- [10] Unknown. *IRF9Z34N Datasheet*. URL: <https://cdn.instructables.com/ORIG/FNM/2CI9/I5ZZY6BI/FNM2CI9I5ZZY6BI.pdf>.
- [11] Unknown. *L7805 datasheet*. URL: <http://pdf.datasheetcatalog.com/datasheet2/8/0ishsf7y9sp31h690e60g8gclc3y.pdf>.
- [12] Unknown. *LM317T datasheet*. URL: <http://www.ti.com/lit/ds/symlink/lm317.pdf>.
- [13] Unknown. *Mosfet*. URL: <https://www.elprocus.com/mosfet-as-a-switch-circuit-diagram-free-circuits/>.
- [14] Unknown. *Peristaltisk pumpe*. URL: https://www.aliexpress.com/item/10Pcs-lot-For-Aquarium-Lab-Analytical-Water-6V-DC-DIY-Dosing-pump-Peristaltic-Dosing-Head-Free/32796649542.html?spm=2114.search0306.3.9.cuL7Ar&ws_ab_test=searchweb0_0,searchweb201602_0_10152_10151_10539_10596_10059_10312_10314_10534_10313_10084_100031_10083_10107_10304_10307_10604_10603_10605_10341_10065_10142_10340_10068_10343_10541_10342_10345_10103_10344_10302,searchweb201603_0,ppcSwitch_0&algo_pvid=df100a46-2343-4001-b860-0a2c81f1fd34&algo_expid=df100a46-2343-4001-b860-0a2c81f1fd34-4.
- [15] Unknown. *POWER SUPPLY*. URL: <https://www.raspberrypi.org/documentation/hardware/raspberrypi/power/README.md>.
- [16] Unknown. *PSoC® 4 and PSoC Analog Coprocessor – Using GPIO Pins*. URL: <http://www.cypress.com/file/127101/download>.
- [17] Unknown. *PSoC® 5LP Prototyping Kit Guide*. URL: <http://www.cypress.com/file/157971/download>.