I3/E3PRJ3: Semesterprojekt 3

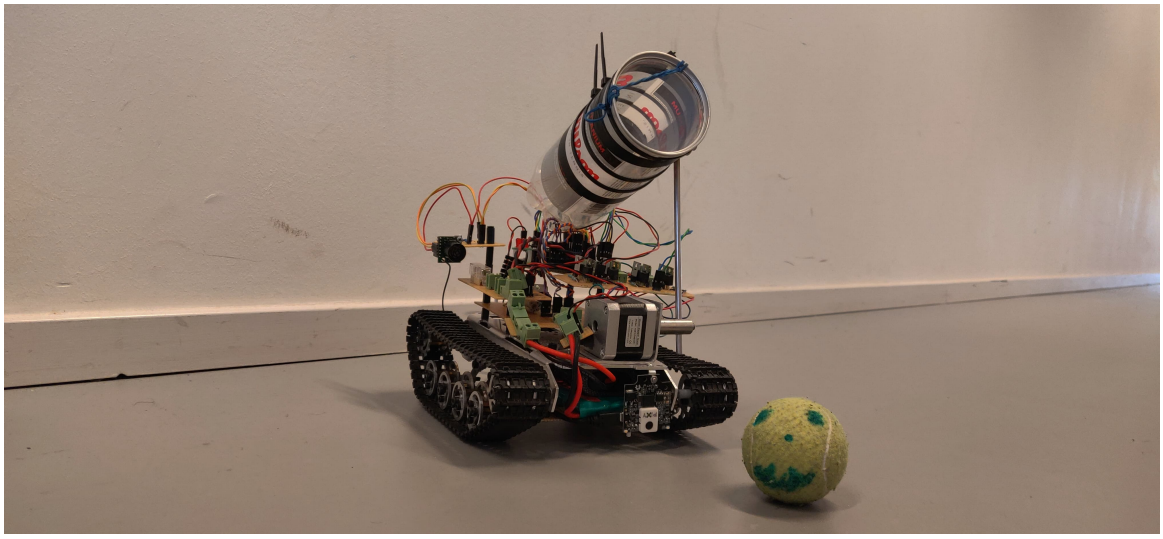
# BoldBot


Gruppe 9

201711532 Andreas Elgaard Sørensen
201707772 Andreas Ellegaard Svendsen
201704259 Christian Karl Oscar Lind Vie Madsen
201711525 Christian Olsen
201710688 Frederik Kronvang Gade
201710709 Mads Stengaard Jørgensen
201710702 Mark Højer Hansen
201710717 Mathias Tørnes Pedersen

Vejleder
Michael Sørensen Loft
ml@ase.au.dk


29. juni 2019



# Abstract

This paper is made by IKT-students (Information- and communication technology engineer) and E-students (Electronics engineer) of Aarhus School of Engineering as a project assignment on the 3. semester in the spring of 2019. It concerns the development of an automated tennisball collector named BoldBot. BoldBot must be able to drive on a tenniscourt, then navigate the court and be able to find and collect tennisballs. This report explains and specifies the development of the working product ”BoldBot”which can locate and collect tennisballs on a closed area.

# Resumé

Denne rapport er udviklet af IKT-studerende (Information- og kommunikations teknologi) og E-studerende (Elektronikingeniør) fra Aarhus Universitet som en projektrapport på 3. semester. Rapporten omhandler udviklingen af en automatisk robot kaldet 'BoldBot. BoldBot skal kunne køre og navigere på en tennisbane og kunne opsamle tennisbolde. Denne rapport forklarer udviklingen af produktet. Under udviklingsprocessen er der anvendt Scrum som arbejdsmetode, den er blevet tilpasset til vores behov og krav som ingeniørstuderende.

# Ansvarsområder

| Initialer | Navn | Initialer | Navn |
|---|---|---|---|
| **AS** | Andreas Elgaard Sørensen | **AES** | Andreas Ellegaard Svendesen |
| **CO** | Christian Olsen | **CM** | Christian Lind Vie Madsen |
| **FK** | Frederik Kronvang Gade | **MJ** | Mads Steengaard Jørgensen |
| **MH** | Mark Højer Hansen | **MT** | Mathias Tørnes Pedersen |

| **Ledelse** | **AS** | **CO** | **CM** | **AES** | **MT** | **MJ** | **MH** | **FK** |
|---|---|---|---|---|---|---|---|---|
| Scrum master | | | | | X | | | |
| Mødeleder | X | | | | | | | |
| Referant | | | | | | | X | |
| **Blok** | | | | | | | | |
| HW Motormodul | | X | X | | | | | X |
| HW Opsamlingsmodul | | X | X | | | | | X |
| HW Spændingsregulator | | X | X | | | | | X |
| HW Batteriindikator | | X | X | | | | | X |
| SW RPi drivers | X | | | | | X | X | |
| SW Webside | | | | | X | | | |
| SW PixyCam | | | | | X | X | | |
| SW RPi | X | | | | X | X | X | |
| SW PSoC Motorstrying | | X | X | | | | | X |
| SW PSoC Opsamlingsmodul | | X | X | | | | | X |
| SW PSoC main | X | | | | | | X | |
| SW Sonar | | | | X | | | | |
| **Rapport** | | | | | | | | |
| Resume og Abstract | | | | | | X | | |
| Forord | | | | | | | X | |
| Problemformulering | X | X | X | X | X | X | X | X |
| Krav | X | X | X | X | X | X | X | X |
| Afgrænsning | X | | | | | | | |
| Metode | | | | | X | X | | |
| Analyse | X | X | X | | X | X | X | X |
| HW Arkitektur | | X | X | | | | | X |
| SW Arkitektur | X | | | | X | X | X | |
| HW Design | | X | X | | | | | X |
| SW Design | X | | | | X | X | X | |
| Test | X | X | X | | X | X | X | X |
| Resultater | X | X | X | | X | X | X | X |
| Diskussion | X | | | | | | | |
| Fremtidigt arbejde | | | | | | | X | |
| **Procesrapport** | X | X | X | | X | X | X | X |

Tabel 1: Tabel over ansvarsområder for alle gruppemedlemmer, hvor X indikerer ansvar for område.

# Indhold

Ansvarsområder . . . . . iv
Ordforklaring . . . . . vi
1 Forord . . . . . 7
2 Projektformulering . . . . . 8
2.1 Løsning . . . . . 8
3 Metode . . . . . 9
3.1 Udviklingsmetoder . . . . . 9
3.2 Proces . . . . . 10
4 Krav . . . . . 12
4.1 Overordnede krav . . . . . 12
4.2 Funktionelle krav . . . . . 12
4.3 Ikke funktionelle krav . . . . . 14
5 Afgrænsning . . . . . 16
6 Analyse . . . . . 17
6.1 Hardware . . . . . 17
6.2 Software . . . . . 18
7 Arkitektur . . . . . 21
7.1 Overordnet system arkitektur . . . . . 21
7.2 Hardware arkitektur . . . . . 24
7.3 Software . . . . . 29
8 Design og implementering . . . . . 31
8.1 Hardware . . . . . 31
8.2 Software . . . . . 41
9 Test . . . . . 52
9.1 Modultest . . . . . 52
9.2 Integrationstest . . . . . 56
9.3 Accepttest . . . . . 60
10 Resultater . . . . . 61
10.1 Modultest . . . . . 61
10.2 Integrationstest . . . . . 62
10.3 Accepttest . . . . . 63
11 Diskussion . . . . . 65
12 Konklusion . . . . . 67
13 Fremtidigt Arbejde . . . . . 68
Bibliografi . . . . . 69

# Ordforklaring

| Forkortelse | Forklaring |
|---|---|
| Analog Discovery | Multiværktøj, med oscilloskop, forsyning, funktionsgenerator og logic analyzer. |
| CPHA | Bestemmer timingen af data bitene relateret til pulsen fra klokken. |
| CPOL | Bestemmer polariteten af klokken. |
| GUI | Graphical User Interface. |
| GPIO | General Purpose Input/Output. |
| GPS | Globale positioning system. |
| Homebase | Den position hvor BoldBot skal tømme de indsamlede tennsibolde. |
| LUT | Look Up Table. |
| PixyCam | Kamera brugt til detektering af tennisbol. |
| PixyMon | Software til kalibering af PixyCam. |
| PO | Product Owner. |
| Realterm | terminal vindue til PSoC. |
| Rpi | Raspberry Pi Zero Wifi. |
| SM | Scrum Master. |
| SPI | Seriel Peripheral Interface. |
| UART | Universal asynchronous receiver/transmitter. |
| USB | Universal Serial Bus. |
| WiFi | Standard for trådløst datanet. |

Tabel 2: Ordforklaring

## Læsevejledning

Der vil i igennem hele rapporten, bilagsrapporten og procesrapporten blive anvendt 'Harward-modellen' til brug af referencer i teksten. Referencen vil bestå af en parentes hvori referencen til en kilde vil ligge. En dybdegående gennemgang af den pågældende reference vil blive placeret i slutningen af hver rapport i alfabetisk orden.

# 1. Forord

Den følgende rapport er skrevet i faget PRJ3. Semesterprojektet er udarbejdet af uddannelserne IKT- og elektronikingeniør på Aarhus Universitet. Rapporten er udarbejdet af projektgruppe 9, som består af 5 IKT studerende og 3 elektronikstuderende, samt vejledning af Michael Sørensen Loft. Rapporten er det endelige produkt udarbejdet igennem semestret. Den består af 3 dele: projektrapporten, en processrapport og en billagsrapport. Udover dette er et fysisk produkt også blevet udviklet, som består af både software og hardware.
Rapporten har afleveringsfrist den 29/05/2019, og efterfølgende bedømmelse med forsvar den 25/06/2019. Den endelige karakter bedømmes ud fra rapporten, og vægtes også ud fra den efterfølgende samtale.

# 2. Projektformulering

Tennis er en sport der kræver mange bolde, som er til stort besvær at samle. Det er et langsommeligt arbejde der tager tid fra spillerne, til deres træningssessioner. Vi vil derfor udvikle en automatisk robot, som bliver kaldt Boldbot. Robotten har til formål at opsamle bolde på en tennisbane. Det er tiltænkt at den skal kunne opsamle bolde i pauserne og når den pågældende træning er ovre. Det er derfor vigtigt at robotten hurtigt og effektivt opsamler samtlige bolde på en tennisbane, inden den nye træning begynder. Der vil i forbindelse med udviklingen af BoldBot også udvikles en brugergrænseflade, som gør det overskueligt for brugeren at bruge produktet i praksis. Brugergrænsefladen skal på en nem og intuitiv måde kunne gøre det muligt for brugeren at aktivere og deaktivere BoldBotten. Det vil derfor være muligt for brugeren at aktivere BoldBotten når der ikke er nogle spillere på banen, og når træningen begynder igen kan den deaktiveres og fjernes fra området. (**Interview**; **Alexander Lykou**)

## 2.1 Løsning

BoldBot vil blive udviklet således, at der sidder en afstandssensor på siden af bilen, der gør det muligt for den at vide hvornår den skal dreje i en ny retning, når den møder en forhindring i form af net eller hegn. Endvidere vil robotten kunne registre bolde på banen, så den målrettet kan køre efter dem, og dermed ikke køre tilfældigt rundt. Det vil gøre indsamlingen af bolde hurtigere og langt mere effektivt. Derudover vil der, hvis tiden tillader det, blive udviklet endnu en sensor der skal kunne registrere antal opsamlede bolde. Når der ikke kan opsamles flere bolde, vil brugeren, via brugergrænsefladen, få en besked om, at den skal tømmes. Derudover vil det også være muligt for brugeren at kunne se, vha. LED på et printboard, hvornår batteriet på BoldBot er fuldstændig afladet. BoldBot vil blive styret vha. Raspberry Pi og en PSoC, som skal kunne kontrollere diverse DC motorer og sensorer som robotten måtte indeholde. Til sidst skal BoldBot have en funktion, der kan gøre det muligt at opsamle bolde på banen, og herefter kunne holde på boldene i en lille beholder.

# 3. Metode

## 3.1 Udviklingsmetoder

I projektet er der taget udgangspunkt i ASE-modellen(**Gennemførelse af semesterprojekt**) til at dokumentere, designe og implementere produktet i semesterprojektet. Dog er der over hele udviklingsmodellen blevet arbejdet agilt. Dette er blevet gjort ved at anvende Scrum til at administrere den iterative arbejdsgang, se afsnit 1.8 på side 9 'Scrum' i procesrapporten. Dette er blevet gjort ved at anvende værktøjerne i Scrum, herunder sprint planlægning, sprint, sprint review og sprint retrospective. Således er projektforløbets iterationer blevet afviklet i sprints, hvorefter hvert sprint er blevet reviewet og reflekteret over ved sprint retrospective, se 1.8 på side 9 'Scrum' i procesrapporten. Dette betyder altså i stedet for at fastlægge sig på arkitektur, implementering og test i starten af projektet, er dette blevet lavet løbende i de forskellige iterationer. Herudover er der i projektforløbet løbende blevet lavet reviews af arkitektur og valg af teknologier. Hvor feedback fra andre er blevet givet i forhold til den udarbejdede arkitektur og valg af teknologier.

### 3.1.1 SysML og UML

Udarbejdelsen af arkitekturdelen har taget udgangspunkt i SysML og UML. SysML er blevet anvendt til beskrivelse af systemarkitektur og design af modulerne i systemet. Herunder er der blevet anvendt BDD og IBD diagrammer, til at beskrive systemets forskellige blokke via SysML standarden. UML er blevet anvendt til at beskrive systemets opførelse, herunder software delene. Her er der blevet benyttet flowchart-, sekvens-, state machine- og klasse diagrammer til visuelt at beskrive den interne interaktion mellem klasserne for produktet. Disse diagrammer er udarbejdet på baggrund af usecasene se bilagsrapport afsnit 1.3 på side 5 'Funktionelle Krav'. (**SysML og UML**)

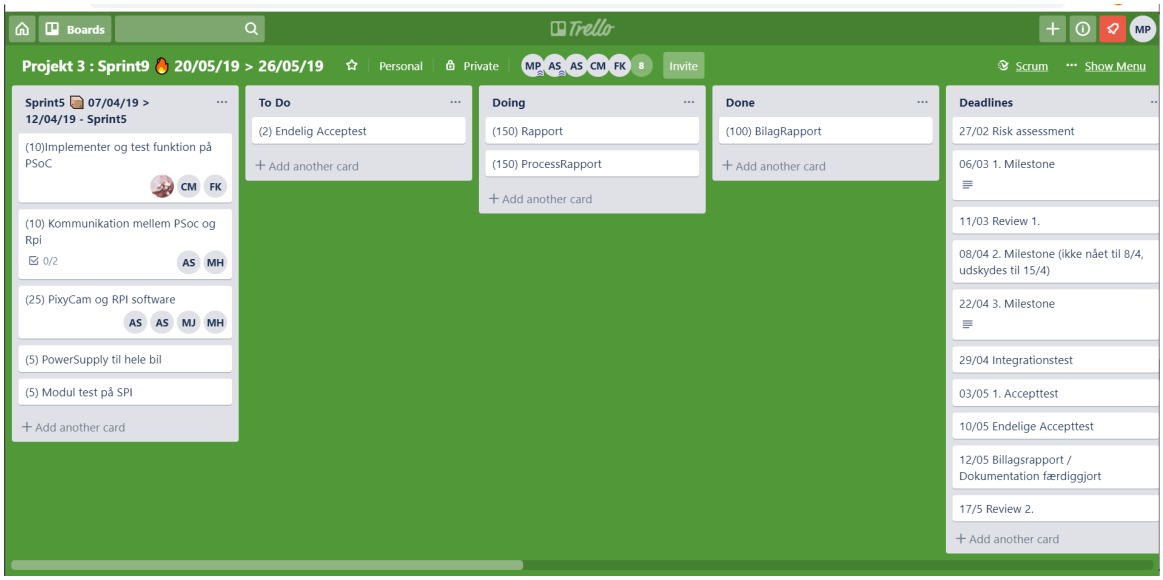
### 3.1.2 MoSCoW og (F)urps+

Til at beskrive systemets funktionelle og ikke-funktionelle krav er der blevet anvendt MoSCoW og (F)urps+ analyserne. MoSCow analysen består af at opstille produktets opførelse, altså de funktionelle krav, i en række prioriteringer i form af Must, Should, Could og Won't. På denne måde indedeles prioriteterne for de funktionelle krav i forhold til produktet. (F)urps+ analysen består af at finde de ikke-funktionelle krav til produktet. Her stilles der spørgsmål til systemets funktionalitet, anvendelighed, pålidelighed, ydeevne og kompatibilitet, hvoraf disse afgrænser systemets omfang. (**FURPS+**)
( **MoSCoW - metoden**)

## 3.2 Proces

Projektgruppen er dannet på baggrund af ønsker imellem de respektive gruppemedlemmer, der på forhånd havde tilkendegivet hvem de ønskede at arbejde sammen med. Projektgruppen blev sammensat af to mindre grupper, bestående af fem IKT studerende og tre E studerende. Der er i sammensætningen af dette semesterprojekt ikke taget højde for personlighedsprofiler, hvilket afspejler sig i manglen af visse roller i gruppen. Der blev i starten af projektforløbet udviklet en samarbejdskontrakt, der blev lavet i fællesskab af gruppen. Samarbejdskontrakten blev lavet for at beskrive gruppens regler for fremmøde, deadlines, arbejdsindsats osv. Se processrapport, afsnit 3 på side 15 'Samarbejdskontrakt'.

Planlægningen af arbejdsforløbet i projektet er lavet vha. en tidsplan, med forskellige milestones for produktet. Milestones på tidsplanen er blevet lavet ud fra krav og forudsætninger til produktet. Se tidsplan i processrapporten, afsnit 1.7 på side 5 ' Planlægning'. Til planlægning og administration af projektets arbejdsopgaver er platformen Trello blevet anvendt, se figur 3.1. Trello har haft til formål at fungere som Scrum-board for hvert igangværende sprint. Hvert gruppemedlem har haft mulighed for at tage en eller flere arbejdsopgaver på Scrum-boardet. Ønskes der en nærmere forklaring, se procesrapport afsnit 1.7.3 på side 6 'Trello.




Figur 3.1: Scrum board via. trello.

Igennem projektforløbet er der ugentligt blevet holdt et fast møde, hvor projektgruppen har mødtes med vejleder, for at få svar på eventuelle spørgsmål, eller diskutere udfordringer/problemstillinger. Typisk har der efter vejledermødet været et længere møde i projektgruppen, hvor der har været review af det forhenværende sprint, og planlægning af et nyt sprint. Udover det faste ugentlige møde, er der blevet holdt stand-up møder in-

ternt i projekt gruppen ifht. Scrum. Stand-up møderne har ligget på to faste dage i ugen, hvor de har haft til formål at oplyse de andre medlemmer i gruppen, om hvor langt man var med sit arbejde og om der var opstået problemer, der forhindrede i at arbejdet kunne forsætte. En nærmere gennemgang af Scrum-forløbet kan ses i procesrapporten i afsnit 1.8 på side 9 'Scrum'.

# 4. Krav

## 4.1 Overordnede krav

Som grundsten for dette projekt, er der blevet opstillet en række krav som projektets produkt har skulle indeholde. Kravene er som følgende:

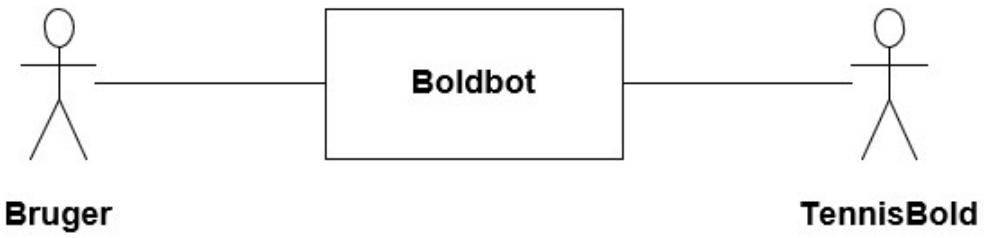
**Projektkrav**

- Systemet skal via sensorer/aktuatorer interagere med omverdenen.
- Systemet skal have et brugerinterface.
- Systemet skal indeholde faglige elementer fra semestrerets andre fag.
- Systemet skal anvende en indlejret Linux platform og en PSoC platform.

Disse krav udgør tilsammen de bestanddele produktet BoldBot skal indeholde. Udover disse krav, har gruppen selv opstillet en række funktionelle og ikke funktionelle krav for projektets produkt.

## 4.2 Funktionelle krav

Dette afsnit indeholder de krav som gruppen har sat for BoldBot. Systemets aktører beskrives med et aktør kontekst diagram, og med den primære Use Case, Use Case 1 [Start/Kør].
På figur 4.1 ses aktør kontekst diagrammet for systemet. Dette giver et overblik over aktører der indgår i systemet. På tabel 4.1 og tabel 4.2 ses aktørbeskrivelser for begge aktører. Yderligere information om de funktionelle krav kan findes i bilagsrapport afsnit 1.3 på side 5 'Funktionelle Krav'.




Figur 4.1: Aktør kontekst diagram

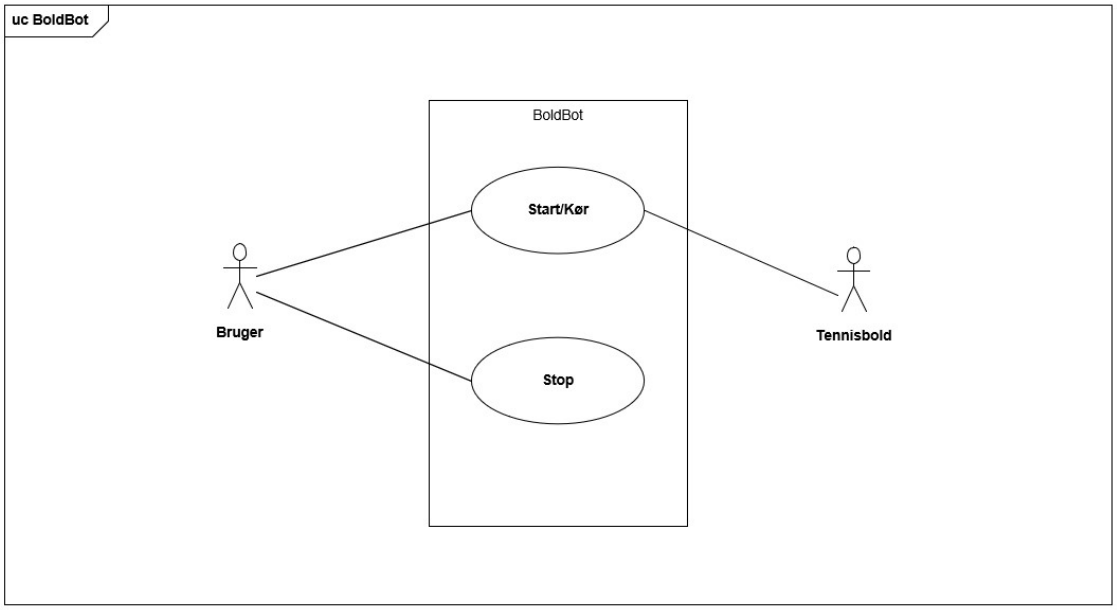
| | |
|---|---|
| Aktørnavn | Bruger |
| Rolle | Primær |
| Beskrivelse | Brugeren skal kunne starte og stoppe BoldBot igennem interfacet. |

Tabel 4.1: Aktørbeskrivelse af brugeren.

| | |
|---|---|
| Aktørnavn | Tennisbold |
| Rolle | Sekundær |
| Beskrivelse | Bolden skal kunne findes og opsamles af BoldBot. |

Tabel 4.2: Aktørbeskrivelse af tennisbold

På figur 4.2 kan use case diagramet for alle use cases i projektet ses. Der er valgt at de kun bliver inkluderet UC1 i rapporten (tabel 4.3). Derudover kan der i bilagsrapporten afsnit 1.3 på side 5 'Funktionelle Krav' ses en fullydressed use case til UC2 'Stop'.




Figur 4.2: Use Case diagram

Use Case 1 - Start/Kør

| **Navn** | **Start/Kør** |
|---|---|
| Mål | At BoldBot opsamler en tennisbold. |
| Initiering | BoldBot initieres på webside. |
| Aktører | Primær: Bruger.<br>Sekundær: Tennisbold. |
| Antal samtidige forekomster | Ingen. |
| Prækondition | BoldBot er inaktiv. |
| Postkondition | BoldBot har samlet bold op. |
| Hovedscenarie | 1. Brugeren starter BoldBot fra webside.<br>2. BoldBot modtager start signal fra webside.<br>3. BoldBot søger efter tennisbolde.<br>4. BoldBot registrerer en tennisbold.<br>[Extension 1: BoldBot registrerer en forhindring]<br>5. BoldBot opsamler tennisbold.<br>[Extension 2: BoldBot fejler opsamling af tennisbold] |
| Extensions | [Extension 1: Forhindring]<br>1. BoldBot møder en forhindring<br>2. BoldBot bakker tilbage.<br>3. Der fortsættes fra trin 3.<br>[Extension 2: Fejl i opsamling]<br>1. BoldBot bakker væk fra bolden.<br>2. Der fortsættes fra trin 3. |

Tabel 4.3: Use Case 1 - Start/Kør

## 4.3 Ikke funktionelle krav

Dette afsnit opstiller de ikke funktionelle krav for BoldBot systemet. Kravene er opstillet efter (F)urps+ modellen, som deler kravene efter deres kvaliteter. Disse krav er med til at lægge et grundlag for accepttesten, som kan findes i bilagsrapporten kapitel 8 på side 102 'Accepttest'.

**Functionality**

1. BoldBot må ikke overstige 40x40x40 cm $\pm 2cm$ .
2. Batteriindikatoren skal kunne vise 5 forskellige batteriniveauer.

**Usabillity**

3. BoldBot skal kunne startes af en bruger.
4. BoldBot skal kunne styre uden om eventuelle uventede forhindringer.

**Reliability**

5. BoldBot skal kunne rydde en tennisbane for bolde, med mindre genstande forhindrer dette.
6. BoldBot skal kunne indikere når spændingsniveauet er under 20%.

**Performance**

7. BoldBot skal kunne samle 1 bold op på mindst 5 min.
8. BoldBot skal kunne opbevare mindst 1 bold før den skal tømmes.
9. BoldBot skal kunne registrere et ukendt objekt (alt andet end en tennisbold) 45 cm fra sensor $\pm 2cm$.
10. Boldbot skal kunne køre mindst 15 minutter på en fuld opladning.

**Supportability**

11. BoldBot skal kunne testes på andet end en tennisbane for at se om den kan detektere bolde og samle dem op.

# 5. Afgrænsning

I starten af projektforløbet, var det tydeligt at projektet skulle afgrænses, for at det ikke blev for uoverskueligt. Projektet indeholder mange forskellige moduler, hvoraf nogle af disse naturligvis skulle begrænses. Første valg der afgrænser projektet omhandler opbygningen af opsamlermodulet. Her er det valgt at nedprioritere effektiviteten af modulet fra et 'rullebånd' der skulle transportere tennisbolde fra forenden af bilen, til bagenden i en lille beholder. I stedet er der blevet implementeret et tennisrør foran bilen, der skal bruges til at samle bolde op.
Derudover var det også klart at projektet stod overfor en stor udfordring i forhold til at registre tennisbolde på en tennisbane. Hvis projektgruppen selv skulle have udviklet et kamera der kunne genkende tennisbolde, vil dette modul have fyldt alt for meget, som vil have ført til at der ikke var blevet lavet andet. I stedet blev det besluttet at købe et kamera, der allerede havde implementeret software, som kunne genkende objekter vha. farver. Denne beslutning har betydet, at der kunne flyttes fokus til andre moduler og udfordringer.
Et tredje valg der har afgrænset projektet var fraværet af en 'home base', BoldBot selv skulle navigere hen til når den enten var løbet tør for strøm eller når BoldBot havde opnået en fuld kapacitet i boldbeholdning. At BoldBot selv skulle kunne navigere hen til en destination, ville betyde at der skulle implementeres en GPS med pathfinding der ville tage fokus de andre moduler i projektet. Til sidst blev der også fravalgt at implementere endnu en sensor, som kunne registrere hvor mange tennisbolde der var i opsamlingsmodulet og som herefter kunne sende et signal til BoldBot at den skulle navigere tilbage til en 'Home Base'.

# 6. Analyse

I analyseafsnittet vil en gennemgang af udviklingsprocessen af systemet fremgå samt udtænkningen af designet og hvilke overvejelser der er blevet gjort under udførelsen af projektet. Der bliver også vist alternativer og andre måder løsningen kunne laves på.

## 6.1 Hardware

### 6.1.1 Opsamlingsmodul

Til design af opsamlingsmodulet blev gjort flere overvejelser i forhold til valg af motor. Der blev overvejet forskellige løsninger til opsamlingsmodulet - DC-motor, stepper-motor og servo-motor. DC-motoren blev hurtigt udelukket, da BoldBot's opsamlingsmodul krævede en vis præcision i forhold til vinklen på løftearmen - denne løsning har den ulempe i denne konkrete sammenhæng, at man ikke kan kende positionen uden at skulle lave ekstra hardware, evt. med sensorer, eller skulle lave en anden mekanisk mekanisme der kunne bruges til løftearm. I forhold til valget mellem stepper-motor og servo-motor, blev servo-motoren valgt fra pga. tilgængelighed - stepper-motor er mindre præcis, men i og med at opsamlingsmodulet ikke krævede så høj en præcision, og stepper-motoren var meget nemmere at placere, pga. indpakningen (bla. stærkere bygget og kunne monteres på den ønskede måde på chassiet til BoldBot), på bilen. Der findes også et bredere udvalg af stepper-motorer med den krævede løftekapacitet (torque) af det udvalg der var let tilgængeligt. Hvis man skulle have brugt mere præcision eller skulle have haft en anden type opsamlingsmodul end den måde modulet er på BoldBot, så havde servo-motor også været en udmærket løsning.

### 6.1.2 Batteriindikator

Til designet af batteriindikatoren er valgt en IC med fire komperatorer i én, fordi der netop skulle bruges præcis fire, og derfor kunne spares på pladsen på printet, da der kun skulle én IC på, i forhold til at skulle have 4 mindre IC'er. Da der skulle bruges komperatorer og ikke en anden type forstærkning, blev komperator-IC'en brugt, da den er specifikt designet til at sammenligne spændinger, dette kommer bl.a. til udtryk ved at den har lavere voltage offset i forhold til andre regulære OP-amps.

### 6.1.3 Motormodul

Til design af motormodulet blev en IC med to h-broer brugt. Dette betød at motor-driver printet kunne designes meget mindre i forhold til alternativet, der var at bygge h-broerne med MOSFETS, dioder og modstande. Desuden er der indbygget sikkerhed i IC'en, der

sørger for at h-broerne ikke kan aktiveres i begge retninger på samme tid. Da der jo også skulle bruges to h-broer var denne IC det oplagte valg. Da chassiet til BoldBot var præ-monteret med DC-motorer, og dermed ikke kunne ændres, skulle der bruges en h-bro og ikke en anden type motor-driver. Ydermere betyder brug af h-bro IC'en lavere omkostninger, da den er billigere at benytte.

### 6.1.4 Sonarsensor

Det blev valgt at afstandssensoren til BoldBot skulle være en sonar. Valget af sonarsensor blev af modellen I2CXL-MaxSonar MB1202. Denne sensor har en rækkevidde fra 25 cm. til 765 cm. og kan registrere næsten alt, også blødere objekter som stof. Sonarsensoren bruger i dens kommunikation I2C hvilket der allerede er undervist i, så det var rimelig nemt at gå til. For at kunne registrere noget lige foran BoldBot var det nødvendigt at montere den i den bagerste del. Sonarsensoren var derfor et godt valg, da den kan måle afstanden, og ikke krævede meget mere ekstra viden at gå til. Alternativer til en sonar-sensor kunne være lyssensor eller lasersensor. Det blev dog sonar da det umiddelbart var godt og ikke for besværligt at gå til.

### 6.1.5 Levelconverter

For at kommunikere mellem PSoC og Rpi er der blevet gjort brug af en levelconverter, eftersom RPi har 3.3V logik og PSoC har 5V logik. Levelconverteren gav en del støj og forstyrring af signalet, som gjorde at PSoC ikke ville kunne modtage det korrekte signal. Derfor er der blevet monteret en pullup modstand på ben 'OE' og 'VA'.

## 6.2 Software

### 6.2.1 Webside

Til valg af GUI for produktet BoldBot, har det været vigtigt at udvikle noget der kan styre BoldBot fra et fjernt sted. Hele idéen med BoldBot er at den skal køre på en tennisbane, og styres af de spilleren der benytter tennisbanen. En tennisbane vil som regel altid have et klubhus i nærheden, hvor det er tænkt at en PC kunne stå, hvorfra man kan styre BoldBot. Udfra disse krav, faldt overvejelserne til GUI'en på enten en webside eller terminal GUI. Fordelene ved at vælge en terminal GUI, der kører direkte på RPi'en, ville være at det er noget udviklerne har erfaring med fra tidligere semester projekt, så det ville være forholdsvis nemmere at udvikle. Dog er en terminal GUI svær at gøre brugervenlig, og designet er fastlåst i forholdt til hvordan terminal vinduet ser ud. Med dette i mente, faldt valget af GUI på en webside. Fordelen ved en webside GUI er at den er nem for brugeren at benytte, samt det er muligt at køre BoldBot programmet parallelt med websiden, hvilket har været essentielt for vores produkt. Udviklingen af websiden

har derimod været udfordrende, da der ingen grundlæggende viden har været indenfor dette område.
Websiden består af tre overordnede dele, et interface design, client og server. Interfacet er udviklet i HTML/CSS og client samt server er udviklet i JavaScript. Disse udviklingsmiljøer er hovedsageligt blevet valgt da de er de meste gængse udviklingssprog indenfor webudvikling. Ønskes en længere forklaring på valg af udviklingssprog se bilag afsnit 2.2.7 på side 12 'Webside'.

### 6.2.2 RPI

Softwaren til RPi er blevet udviklet i C++ og de implementerede kernemoduler er blevet skrevet i C. RPi har til ansvar at håndtere den modtagne data fra det valgte kamera (PixyCam), hvorefter RPi sender tilsvarende reaktion på det modtagne data over den serielle kommunikation til PSoC. Den implementerede serielle kommunikation mellem PSoC og RPi er blevet implementeret med SPI. SPI protokollen anvendes da denne understøtter en høj overførelses hastighed af den sendte data. Til at modtage den sendte data fra PixyCam på RPi er den serielle kommunikation USB blevet anvendt. USB protokollens høje kompleksitet ville normalt gøre denne protokol uanvendelig, samtidig med at denne protokol er ukendt teknologi, men da protokollen allerede er en integreret del af PixyCam er det oplagt at anvende USB protokollen. Alternativt kunne den serielle kommunikation mellem PixyCam og RPi være blevet implementeret med SPI eller I2C, dog er USB protokollen væsentligt hurtigere når det kommer til overførelse hastigheder. Yderemere læses der på GPIO17 på RPi, som bliver sendt fra PSoC, hvilket angiver dataen fra sonarsensoren. Grunden til at der er valgt at læse fra en GPIO-pin, skyldes at der udregnes en række værdier på PSoC'en omkring afstandsbestemmelsen, således GPIO pinen sættes til logisk 1 når en forhindring er inden for den korrekt værdi. For mere information omkring RPi analysen henvises der til bilagsrapporten afsnit 2.2.8 på side 13 'Software på RPi'.

### 6.2.3 PSoC

Alt software til PSoC blev udviklet i PSoC Creator 4.2. I PSoC creator, er det kun muligt at kode i sproget C, derfor blev der ikke overvejet at skrive i andre sprog. Der har været fokus på at lave et program som hele tiden tjekker på det som modtages fra RPi, og tilsvarende kalder den korrekte funktion på baggrund af det sendte data fra RPi. Det er opbygget så RPi fortæller PSoC om den skal køre frem, stoppe, dreje eller samle en bold op. Derfor har vi valgt at lave et switch case, som tjekker på den modtagede værdi fra RPi, og kører den givne funktion. Et alternativ til dette ville være at opbygge softwaren som if/else sætninger. Men switch case blev priotet pga. effektivisering, hurtighed og

overskuelighed. For mere information omkring PSoC analyse henvises til billagsrapporten afsnit 2.2.10 på side 13 ’PSoC’.

# 7. Arkitektur

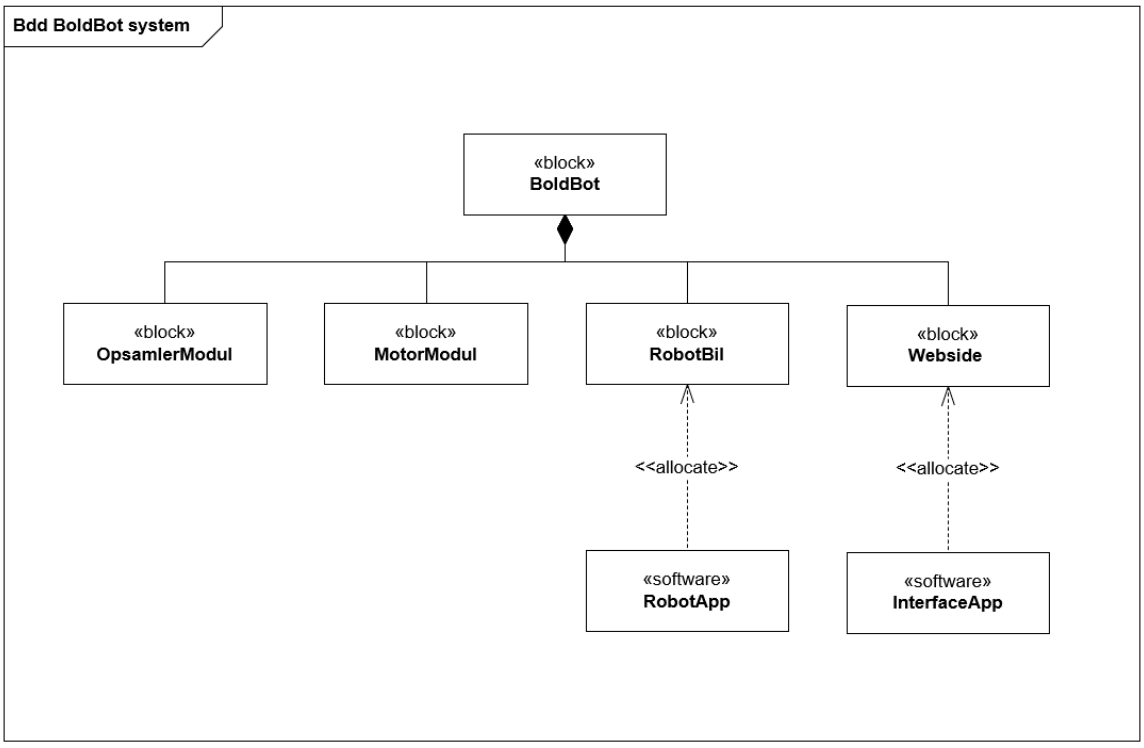
I dette afsnit vil arkitekturen for BoldBot blive gennemgået. Først gennemgås en overordnet arkitektur for systemet. Herefter dykkes der ned i de forskellige moduler i systemet, hvor der ved hjælp af SysML BDD og IBD diagrammer vil blive redegjort for deres funktionalitet og sammensætning.

## 7.1 Overordnet system arkitektur

I dette afsnit gennemgås den overordnet system arkitektur for systemet.

### BoldBot BDD

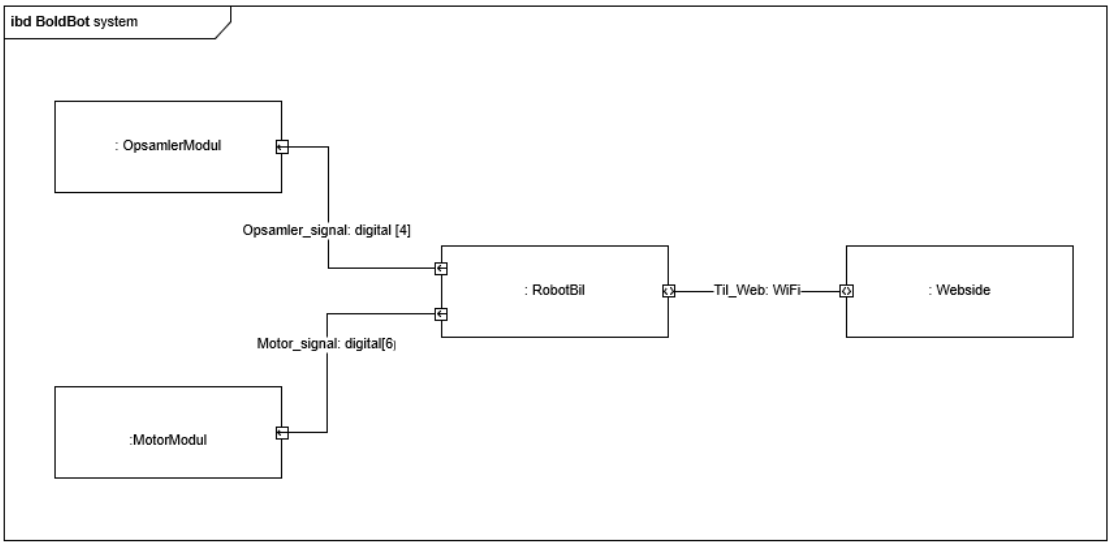
På figur 7.1, ses det overordnede BDD for BoldBot. Selve BoldBots blokke, består af et opsamlermodul, et Motormodul, en Robotbil og en webside. På RobotBil og webside, bliver der allokeret software i form af en RobotApp og en InterfaceApp. Disse blokke vil der i de efterfølgende afsnit blive gået i dybden med.




Figur 7.1: BDD af hele BoldBot system.

### 7.1.1 BoldBot IBD

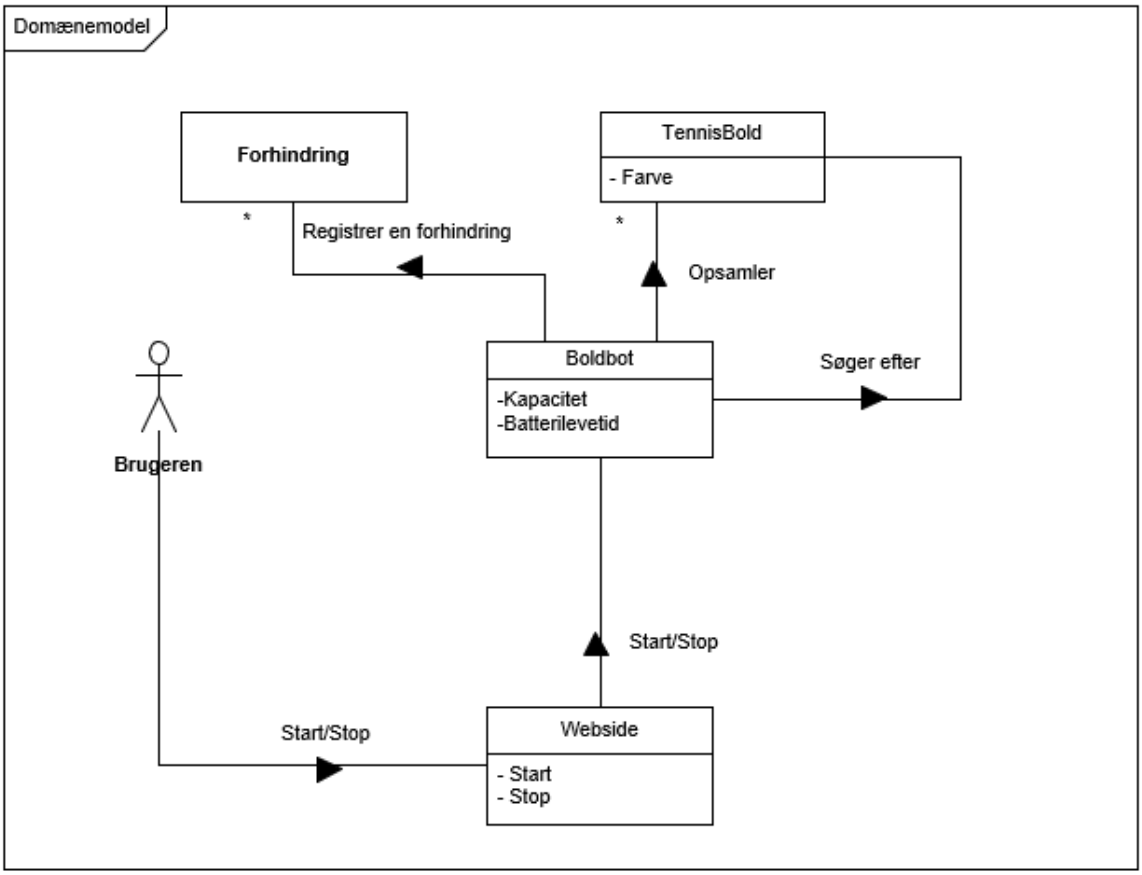
På figur 7.2 er et IBD udarbejdet fra det forrige BDD på figur 7.1 IBD'et viser de overordnede blokkes interne forbindelse. IBD'et kan med fordel læses fra højre, hvor webside blokken er. Websiden kommunikerer med RobotBil blokken, der herefter kommunikerer med opsamlermodul og motormodul.




Figur 7.2: IBD af hele BoldBot system.

### 7.1.2 BoldBot Domænemodel

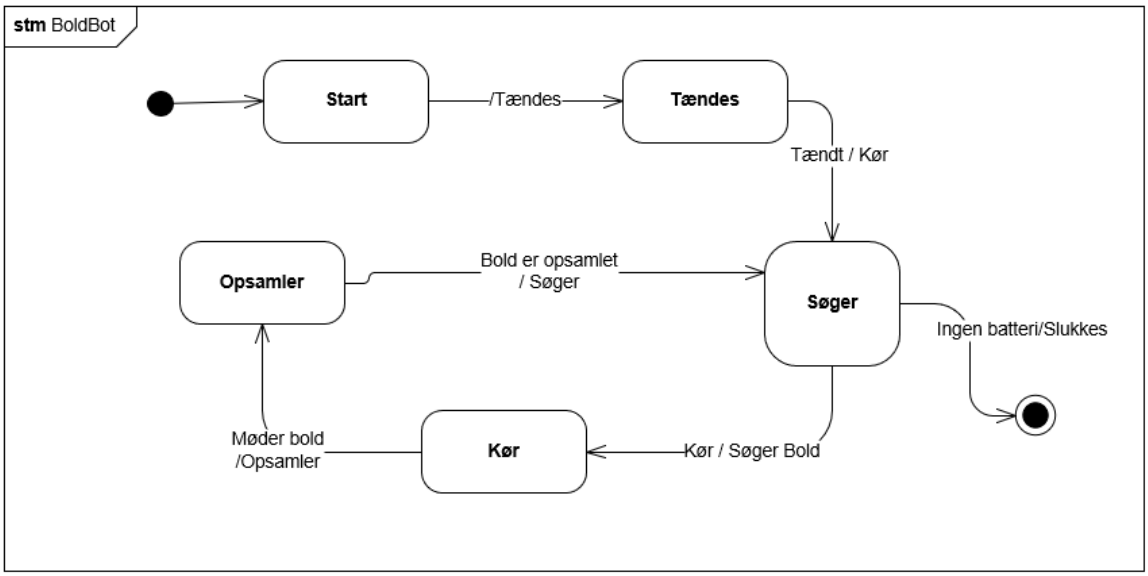
Domænenmodellen for BoldBot på figur 7.3, har til formål at illustrere systemets opbygning, samt dets anvendelse. Da BoldBot'en er et autonomt system, ses der på domænemodellen ikke alle moduler som indgår i systemet. Domænemodellen repræsenterer i stedet systemets funktionalitet på baggrund af usecases.
Brugeren starter/stopper BoldBot via. en webside der kommunikerer med BoldBotten, den interne funktionalitet i BoldBot er ikke vist her, da dette foregår i webside blokken.




Figur 7.3: Domænemodel af hele BoldBot system.

### 7.1.3 BoldBot Statemachine

Statemachine diagrammet på figur 7.4, har til formål at give et overblik over de forskellige processer der sker inde i selve BoldBot systemet. BoldBot tændes og startes, hvorefter den påbegynder søgning efter en tennisbold. Detekterer BoldBot en tennisbold, går den ind i kør, hvor den kører imod tennisbolden, indtil den når position for opsamling af tennisbold. Efter forsøg på at opsamle tennisbold, går BoldBot igen ind i søger processen. Dette loop forsætter BoldBot i, indtil den slukkes, eller løber tør for batteri. Sonarsensorens funktionalitet er udeladt fra dette diagram, på baggrund af manglende integrering i systemet, uddybende forklaring kan ses under afsnit 9.2.



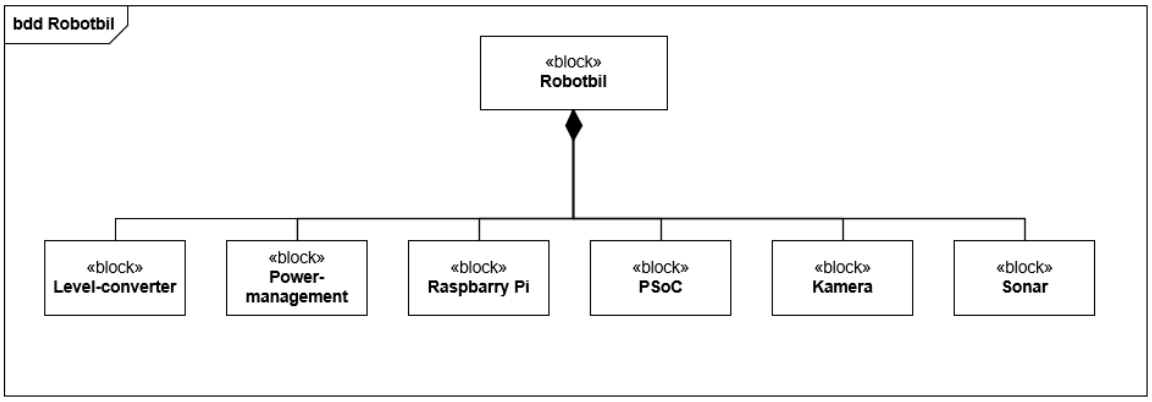

Figur 7.4: State machine digram af hele BoldBot system.

## 7.2 Hardware arkitektur

Som vist i den overordnede systemarkitektur, består BoldBot af 3 Hardware moduler, hhv. robotbil, opsamlermodul og motormodul. I dette afsnit gennemgås modulernes Arkitektur modulvis.

### 7.2.1 Robotbil

Figur 7.5 viser et BDD over modulet robotbil. Her ses Robotbil modulet bestående af 6 blokke.




Figur 7.5: BDD af robotbil.

### Blokbeskrivelse

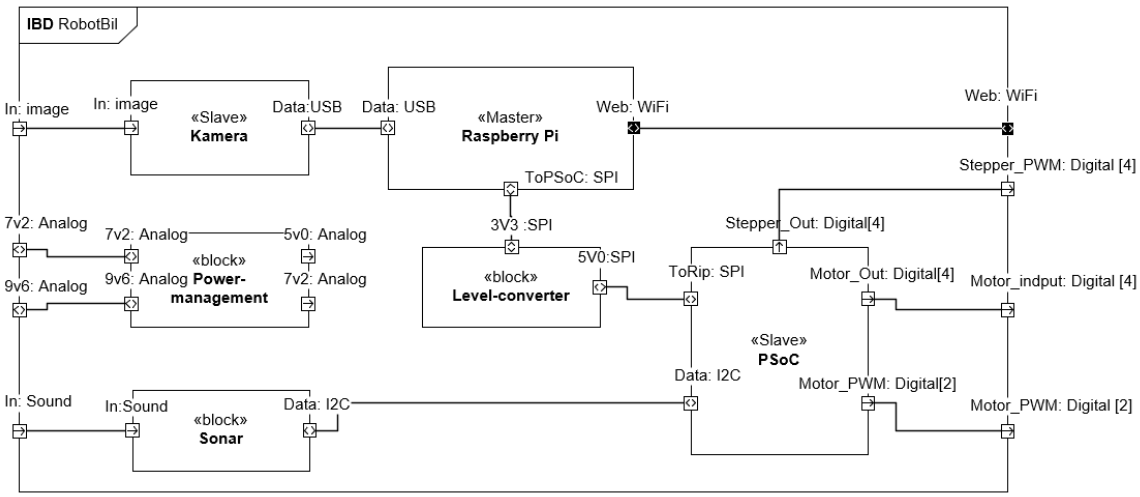
Tabel 7.2.1 viser en blokbeskrivelse af de forskellige blokke i robotbil modulet

| Bloknavn | Funktion |
|---|---|
| Level-converter | Level-converteren har til opgave at konvertere de to logiske niveauer fra PSoC og Raspberry Pi, så de kan kommunikere med hinanden |
| Power-management | Power-management blokken har til opgave at regulere batteriets spændingsniveau, samt at overvåge batteriernes niveauer |
| Raspbarry Pi | Raspberry Pi'en har til opgave at styre BoldBot ud fra de data den modtager fra Kameraet |
| PSoC | PSoC'en har til opgave at styre opsamlermodul og motormodul |
| Kamera | Kameraet har til opgave at lokalisere en tennisbold og videresende dens position til Raspberry PI'en |
| Sonar | Sonar har til opgave at stoppe BoldBot fra at køre ind i objekter |

Tabel 7.1: Blokbeskrivelse for robotbil

### Robotbil IBD

Figur 7.6 viser et IBD over robotbil modulet. Her ses hvordan de forskellige blokke i modulet er forbundet med hinanden og andre moduler i systemet. For flere detaljer omkring Signalbeskrivelse og portspecfikation se Internal Block Diagram i bilagsrapporten afsnit 3.2.2 på side 21 'Internal Block Diagram'.

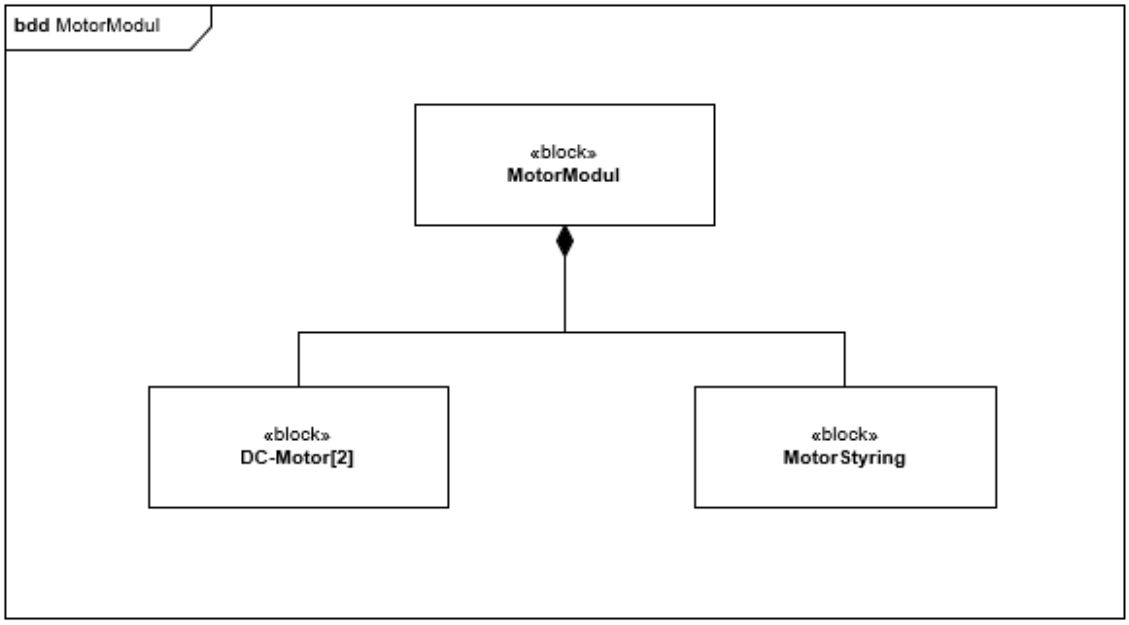



Figur 7.6: IBD af robotbil

### 7.2.2 Motor modul

I dette afsnit bliver arkitekturen for motormodulet gennemgået.

Figur 7.7 viser et BDD over motormodulet. Her kan det ses at modulet består 2 blokke, en Motorstyring og 2 DC-motorer.




Figur 7.7: BDD af hele Motor modul.

#### Blokbeskrivelse

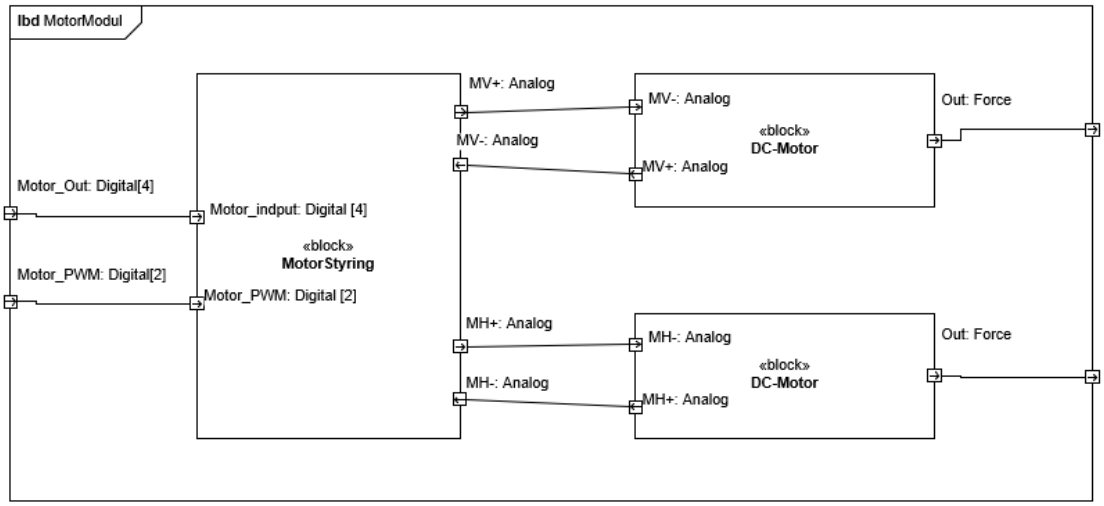
Tabel 7.2.2 viser en blokbeskrivelse af det forskellige blokke i motor modulet

| Bloknavn | Funktion |
|---|---|
| DC-Motor | Har til opgave at få BoldBot til at bevæge sig |
| MotorStyring | MotorStyringens funktion, er at sikre at fart og retningen af de to DC-motorer kan styres med PWM-signal fra PSoC'en. Samt at der trækkes strøm fra et batteri, i stedet for fra PSoC'en |

Tabel 7.2: Blokbeskrivelse for motor modul

#### Motormodul IBD

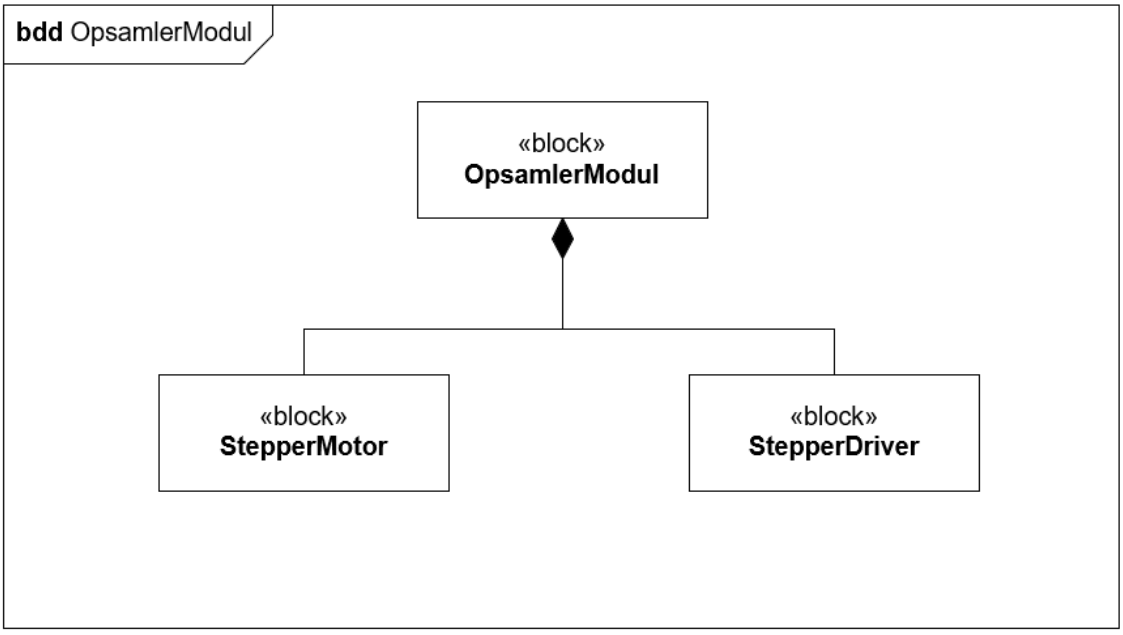
Figur 7.8 viser et IBD over motormodul. Her kan det ses hvordan blokkene i modulet er forbundet med hinanden og andre moduler i systemet. For flere detaljer se Internal Block Diagram i bilagsrapporten afsnit 3.2.2 på side 21 'Internal Block Diagram'.




Figur 7.8: IBD af Motor modul.

### 7.2.3 Opsamler modul

I dette afsnit gennemgås arkitekturen for Opsamlermodul




Figur 7.9: BDD af hele opsamler modul.

### Blokbeskrivelse

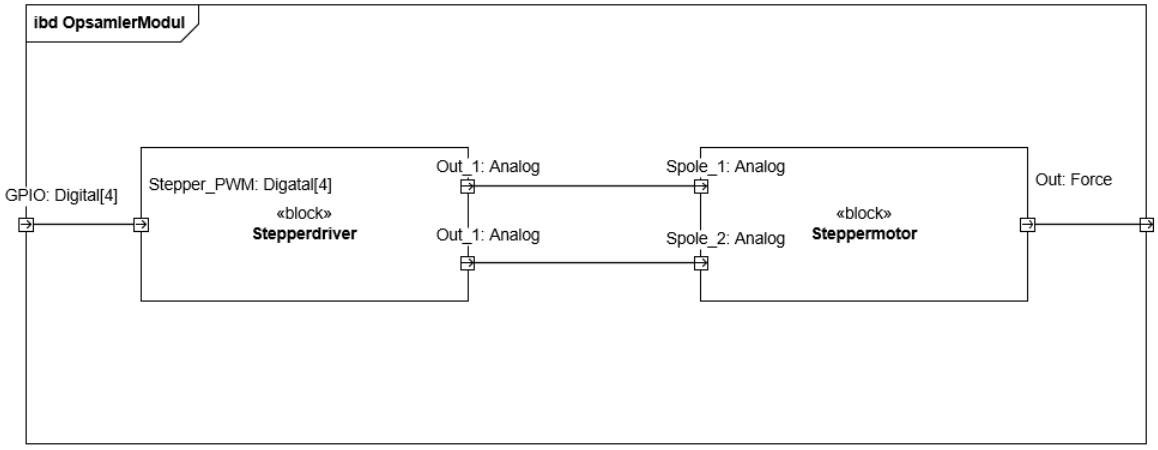
Tabel 7.2.3 viser en blokbeskrivelse af det forskellige blokke i opsamlermodulet modulet

| Bloknavn | Funktion |
|---|---|
| StepperMotor | Steppermotoren har til opgave at bevæge opsamlerarmen |
| StepperDriver | stepperdriveren her til opgave at sikre at steppermotoren kan styres med en PSoC, ved hjælp af PWM-signaler |

Tabel 7.3: Blokbeskrivelse for opsamler modul

### IBD opsamlermodul

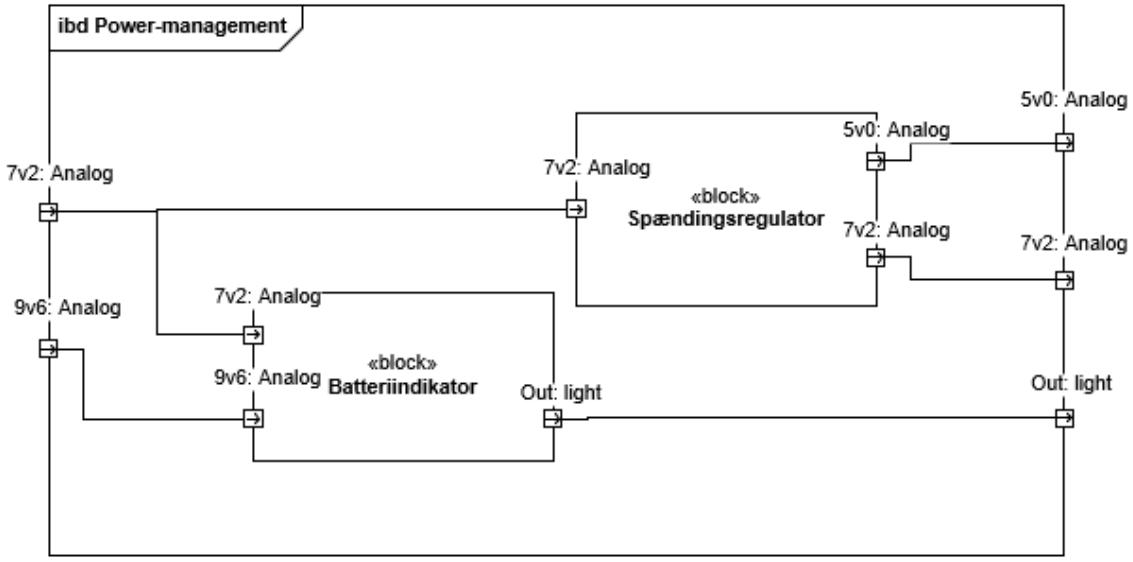
Figur 7.10 viser et IBD over opsamlermodulet. Her kan det ses hvordan blokkene i modulet er forbundet med hinanden og andre moduler i systemet. For flere detaljer se Internal Block Diagram i bilagesrapporten 3.2.2 på side 21 'Internal Block Diagram'.




Figur 7.10: IBD af opsamler modul.

## 7.2.4 Power-management

Figur 7.11 viser et IBD over Power-management blokken. Her ses det at den består at en spændingsregulator og en batteriindikator. Her kan det også ses hvordan blokkene er forbundet med de andre moduler. For flere detaljer om dette, se Internal Block Diagram i bilagesrapporten 3.2.2 på side 21 'Internal Block Diagram'.




Figur 7.11: IBD af Power-management.

### 7.2.5 Signalbeskrivelse

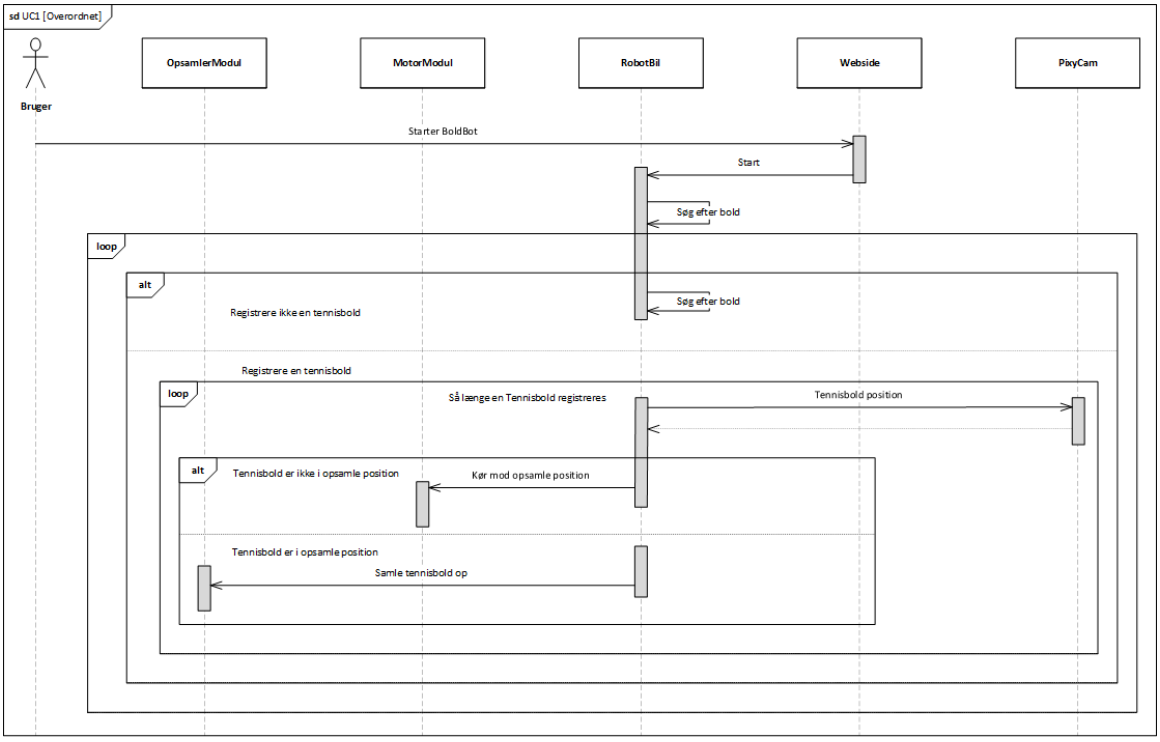
Signalbeskrivelsen kan findes i bilagsrapporten afsnit 3.2.2 på side 21 'Internal Block Diagram'.

## 7.3 Software

Dette afsnit omhandler den overordnede software arkitektur for systemet. Her bliver der med UML-diagrammer, herunder sekvensdiagrammer, beskrevet systemets overordnede funktionalitet ud fra de to forskellige usecases. For at se de to Usecases henvises der til afsnit 1.3 på side 5 'Funktionelle Krav' i bilagsrapporten. De anvendte moduler i nedstående sekvensdiagrammer fremgår af IBD'et 7.2.

#### Use case 1

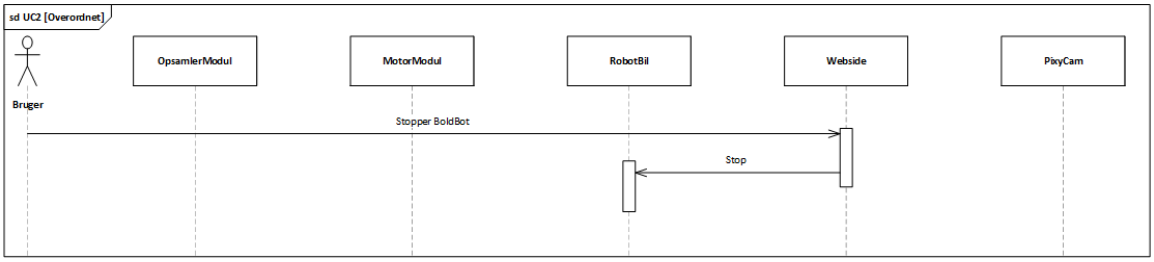
På figur 7.12 fremgår det overordnede sekvensdiagram for UC1. Af sekvensdiagrammet fremgår de blokke, som BoldBot består af. Dette er dog undtagen blokken "pixyCam,"som der ikke selv er blevet implementeret, men anvendt. Sekvensdiagrammet beskriver hvordan de forskellige blokke kommunikerer, når det kommer til funktionaliteten af systemet specificeret i UC1 se tabel 4.3. For mere dybdegående gennemgang af hvordan dette er blevet designet, henvises der til afsnit 8.2.




Figur 7.12: Det overordnet sekvensdiagram for UC1.

Use case 2

Af figur 7.13 fremgår den overordnede funktionalitet af systemet, specificeret ud fra UC2. Af sekvensdiagrammet ses de blokke, som BoldBot består af, som nævnt i ovenstående afsnit. Herudover beskriver sekvensdiagrammet kommunikationen mellem de forskellige blokke i forhold til UC2, se tabel 1.2 på side 6 'Use Case 2 - Stop' i bilagsrapporten for hele UC2.




Figur 7.13: Det overordnet sekvensdiagram for UC2.

# 8. Design og implementering

Der vil i dette kapitel komme en gennemgang af implementeringen og design af både hardware og software. Hvert afsnit er inddelt i de forskellige moduler, som indeholder en kort beskrivelse. En nærmere beskrivelse af design og implementeringen kan ses i bilagsrapporten kapitel 4 på side 40 og kapitel 5 på side 61.

## 8.1 Hardware

### 8.1.1 Indledning

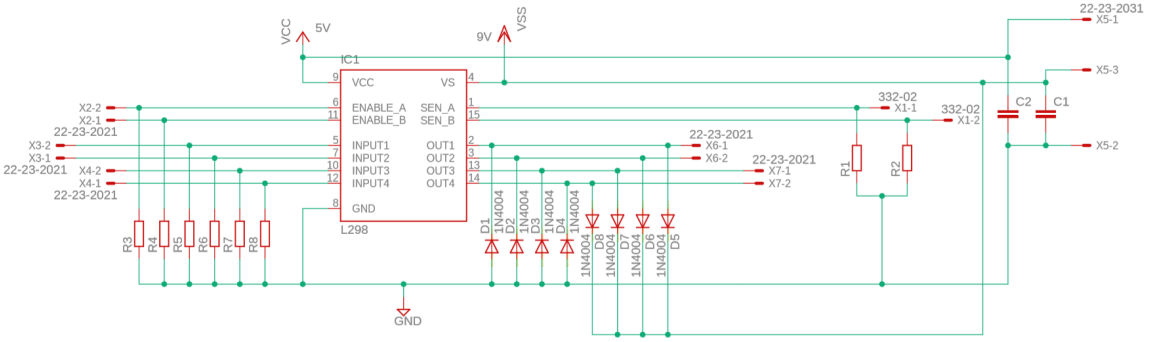
Dette afsnit beskriver designet af hardwaremodulerne, der er udarbejdet til udførelsen af BoldBot. Der vil blive refereret til mere udførlig dokumentation i bilagsrapportens kapitel 4 på side 40 'Hardware design og implementering', hvor der også kan findes styklister med komponenter for de forskellige moduler til at reproducere produktet. Designet er delt op for hvert hardware modul på BoldBot, som igen er delt op i hardware design og PSoC design.

### 8.1.2 Motormodul

I dette afsnit beskrives det hvordan hardwaren for motorstyringen på BoldBot er designet. Der bliver redegjort for funktionen af komponenter. Funktionsbeskrivelsen for modulet kan findes i tabel 7.2.2 Blokbeskrivelse .
Til motorstyringsprintet bliver der brugt en PSoC som står for styringssignalerne til motorprintet. I denne del er der brugt Eagle til tegning af kredsløbsdiagrammer og Veecad til layout af print.

#### Hardwaredesign

BoldBot består af to DC-motorer der hver står for at drive én side af larvefødderne. For at kunne både køre frem, bakke og dreje, skal der derfor laves et kredsløb som kan styre både hastighed og retningen på de to motorer. Begge motorer skal styres uafhængigt af hinanden. Dette gøres ved at bruge en h-bro til hver motor. Valget er faldet på en L298 IC. Argumentation for dette valg kan findes i afsnit 6.1.3 'Motormodul'

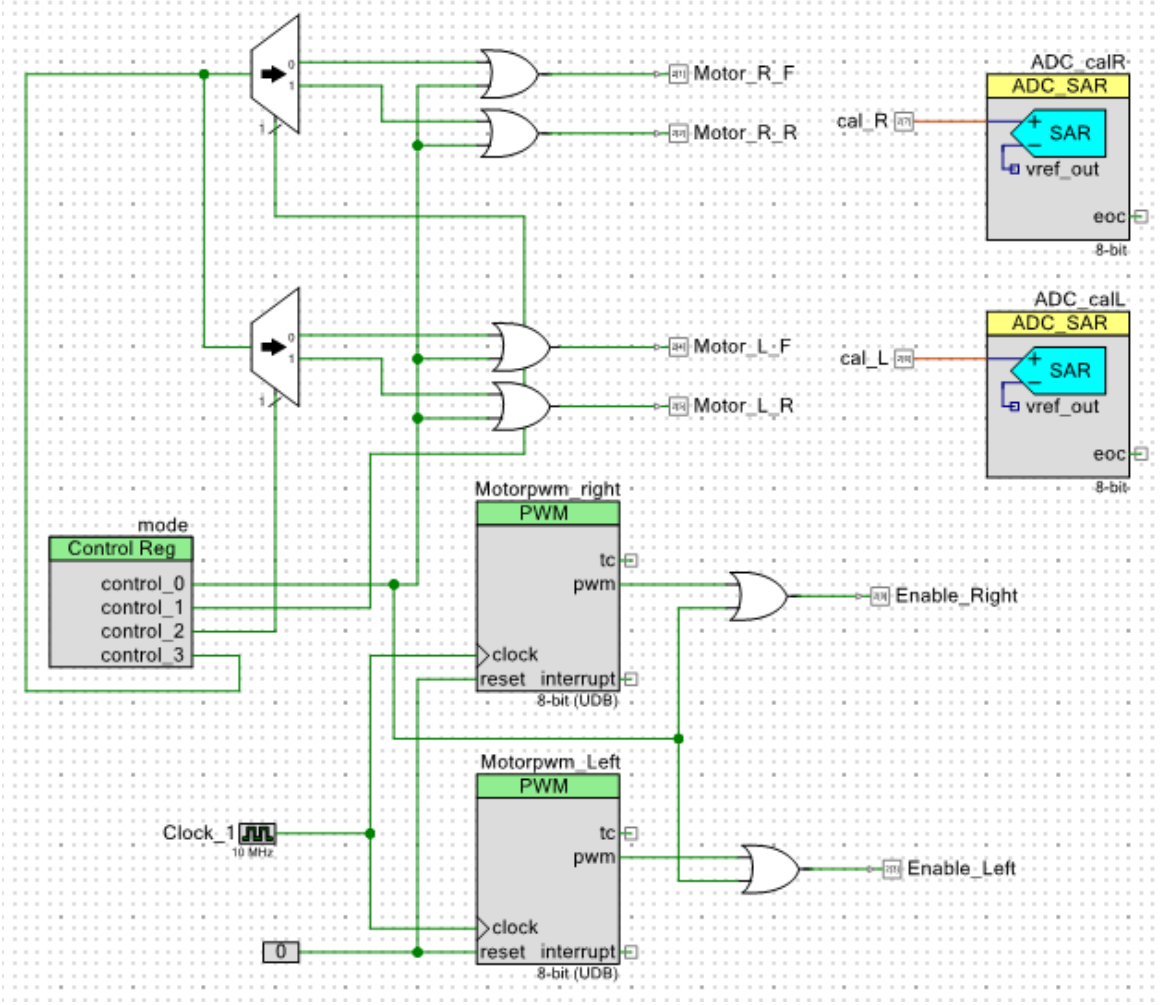



Figur 8.1: Kredsløbsdiagram for motorstyringsprintet

Som det kan ses på Figur 8.1 består L298 IC'en af totalt 6 indgange. Alle indgange fra X2 til X4 har en pulldown modstand som er modstandene R3 til R8, for at sikre mod at indgange skulle svæve og derved fejltænde motorerne som er forbundet på udgangene X6-1, X6-2, X7-1 og X7-2. Ligeledes er der 6 dioder på udgangene for at beskytte L298 IC'en mod overspænding genereret af motorne.
Modstandende R1 og R2 er begge effektmodstande på hhv. 1 ohm og 6W og som hver især sidder i serieforbindelse med en af motorne. Disse gør det muligt at måle strømforbruget på hver motor. Det skal dog siges at der højest kan løbe en strøm til hver motor på 2 ampere hvilket udgør en effekt igennem hver modstand på 5W. Derfor er der valgt en modstand på 6W så de med sikkerhed ikke brænder af.
Stikket X5 er forsyningen til både den logiske del i L298 og forsyningspændingen for motorerne. Her er der også sat nogle kondensatorer i forbindelse med de to forsyninger som hhv. er C1 for 9V forsyningen og C2 for 5V forsyningen som er den logiske del i L298.

## PSoC Design

For at kunne drive motorstyringsmodulet med en PSoC, skal der først oprettes noget hardware i PSoC'ens PLD som er en programmerbar logisk enhed. For at gøre dette muligt laves først et topdesign i PSoC Creator og derefter programmeres dette design via C-kode.




Figur 8.2: Kredsløbsdiagram for motorprintet

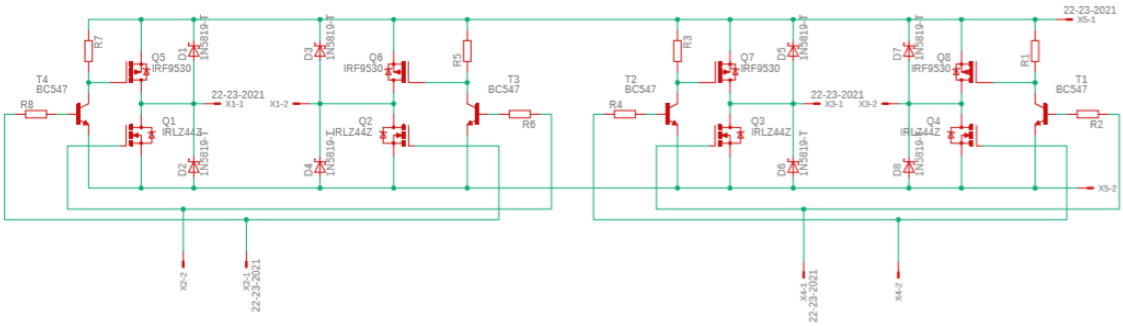
Topdesignet på Figur 8.2 for motorstyringen består af 2 SAR'er, der fungerer som kalibrering af hastigheden på hhv. højre og venstre side af Boldbotten. Det er hhv. "ADC calR"som kan finjusterer hastigheden på højre side og "ADC calL"der finjusterer hastigheden på venstre side.
Motorpwm right og Motorpwm left bruges til at give motorstyringsprintet PWM signal på de to Enable pins. Frekvensen af begge PWM signaler er 50kHz, da dette ligger uden for det hørbare spektrum for mennesket og er også denne frekvens hvor motorerne på BoldBot øjensynligt opererer bedst. Dog er der koblet en OR-gate imellem på hver udgang for at kunne bruge "Fast-stop"mode på motorstyringsprintet, der gør at BoldBot kan benytte aktiv bremsning.

### 8.1.3 Opsamlingsmodul

Opsamlingsmodulet på BoldBot består af en steppermotor og et driverprint til at drive steppermotoren fra den samme PSoC der også står for motorstyringen i motormodulet.

## Hardwaredesign

For at styre den bipolære steppermotor der bruges i dette modul, bruges to h-bro'er, da der inde i steppermotoren er to spoler, som man skal vende polariteten på for at få steppermotoren til at dreje. Grundet tidspres ift. levering af komponenter blev denne dobbelte h-bro lavet ud af transistorer og mosfets. Kredsløbsdiagram over den dobbelte h-bro kan ses på Figur 8.3.

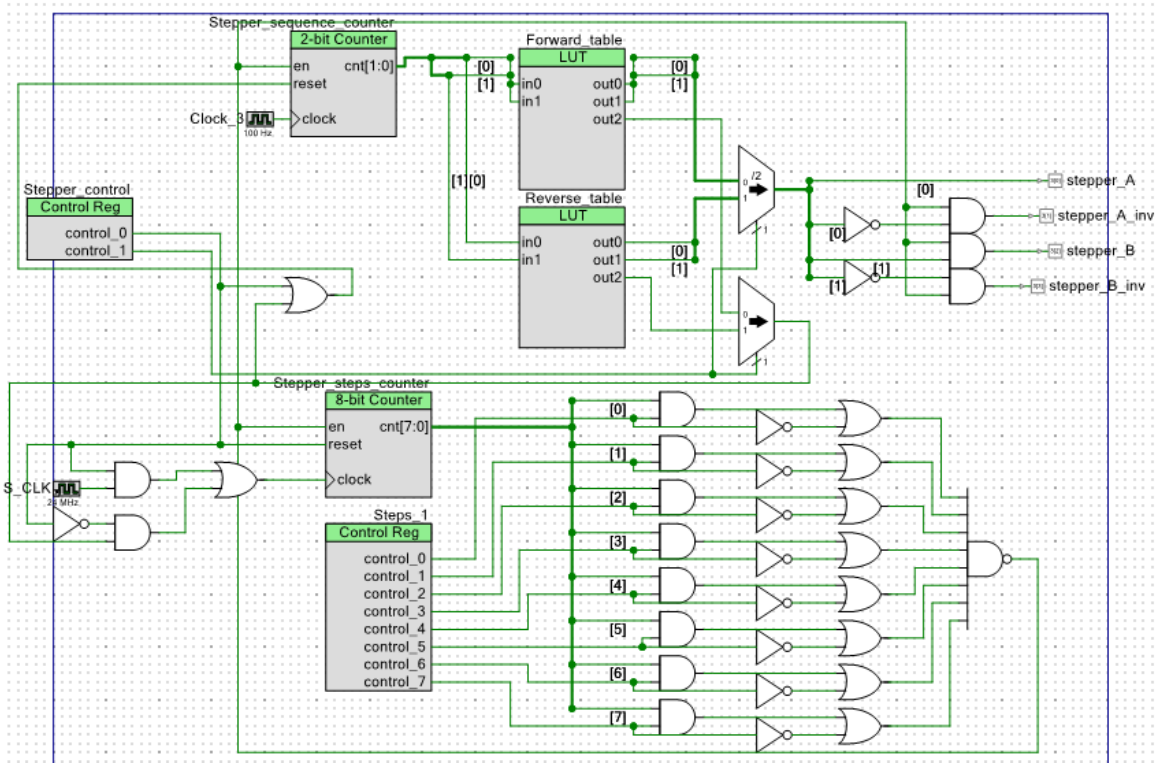



Figur 8.3: Kredsløbsdiagram for opsamlingsmodul

Den dobbelte h-bro er opbygget ved hjælp af henholdsvis N og P channel MOSFET's. Der bruges en N-channel MOSFET for at afbryde og tilslutte ground til udgangsterminalerne X1-1 og X1-2 samt X3-1 og X3-2. Den N-channel der er brugt er en IRFZ44Z hvilket er Q1, Q2, Q3 og Q4 på Figur 8.3.
Se begrundelse for valg af MOSFET i bilagsrapportens afsnit 4.2.2 på side 43 'Design af opsamlingsmodul'.
Der bliver brugt P-channel MOSFETS til at afbryde og tilslutte VCC til udgangsterminalerne X1-1 og X1-2 samt X3-1 og X3-2.
Se begrundelse for valg af MOSFETs i bilagsrapportens afsnit 4.2.2 på side 43 'Design af opsamlingsmodul'. I samme afsnit kan også findes beregninger for base-modstand til transistorerne.

## PSoC Design

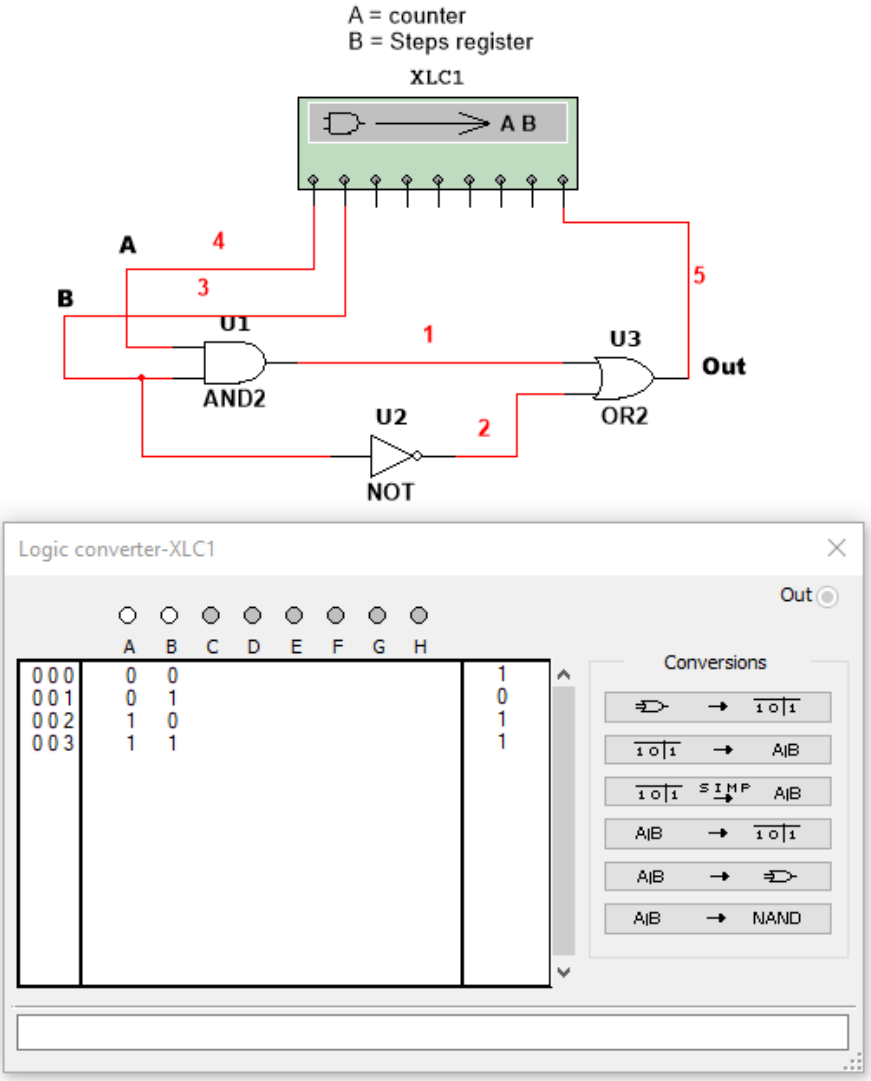
På samme måde som ved motormodulet er der for opsamlingsmodulet lavet et topdesign i PSoC for at kunne drive steppermotor driver printet.




Figur 8.4: Topdesign i PSoC for opsamlingsmodul

På figur 8.4 ses topdesignet for opsamlingsmodulet, der består af en ”Stepper sequence counter”, som egentlig er en 2-bit bineær counter. Denne counter får et 100 Hz clock signal som angiver hastigheden på stepper motoren, altså hvor mange steps motoren tager i sekundtet. Counteren er koblet til henholdsvis to seperate LUT’er, hvor hver LUT er en sandhedstabel som beskriver tændingsrækkefølgen af steppermotoren.
Den LUT der bruges afhænger af hvilken rotationsretning steppermotoren skal have. LUT ”Forward table”for frem og LUT ”Reverse table”for modsat retning. Udgangene af begge LUT’er bliver derefter ført over i to seperate multiplexere hvoraf den øverste vælger den valgte tændingssekvens (rotationsretning) mens den nederste vælger det rette reset signal for den valgte LUT sandhedstabel.
Reset signalet er også koblet på ”Stepper steps counter”som tæller en op når den første counter ”Stepper sequence counter”er nået igennem en af de to LUT tabeller en enkelt gang. Dette gør at det er muligt at tælle antal steps som steppermotoren tager. Outputtet af ”Stepper steps counter”bliver derefter splittet op i 8 forbindelser hvor hver bit bliver sammenlignet med controlregisteret ”Steps”ved hjælp af en AND, NOT og OR gate. Dette gøres ved at når der fx står ’0’ på Control0’s plads i Steps registeret, så bliver udgangen af den øverste OR-gate altid 1 uanset hvad den øverste AND-gate får fra indputtet. Hvis der derimod står ’1’ på Control0’s plads, får bit 0 af ”Stepper steps counter”lov til at passere. Alle 8 signaler bliver derefter NAND’et sammen, og derefter sendt ud til enable pinsne på begge countere. Dette bliver illustreret bedre på Figur 8.5 som er en sandhedstabel

over kredsløbet.




Figur 8.5: Sandhedstabel for det lille gate kredsløb

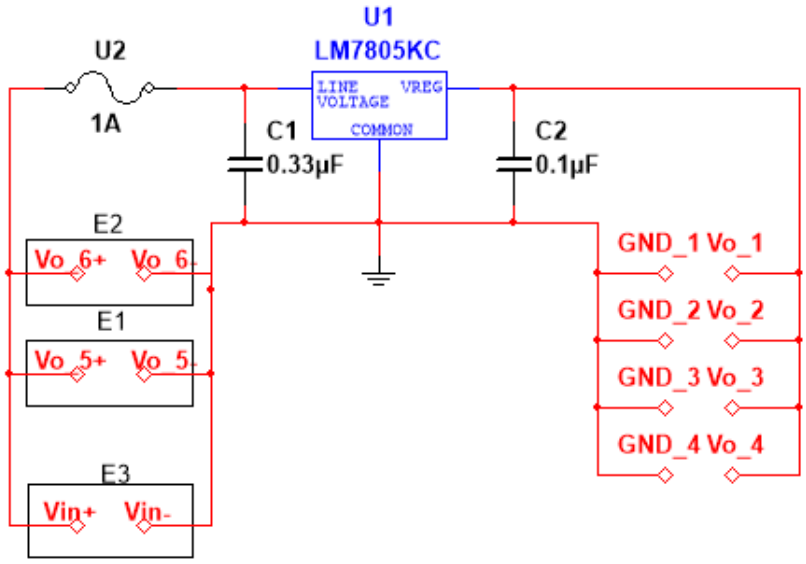
Her ses på sandhedstabellen, at når B er nul, er outputtet altid 1. Men når B er et, så får udgangen A lov til at bestemme outputtet. Dette gør at man kan holde øje med hver af de der bliver sendt ud af den 8-bit counter ved navn ”Stepper steps counter”. Efter de tre gates, bliver alle 8 signaler NAND’et sammen og hvis alle indgange er 1 så bliver begge countere deaktiveret. Dette gør at når counteren er har talt lige så meget som der står i Steps registeret, så stopper begge countere og dermed også steppermotoren i vores system.

### 8.1.4 Spændingsregulator

For at kunne regulere spændingen ned fra den givne batterispænding til 5 V forsynings-spænding, som de interne moduler skal bruge, designes der et spændingsregulator print.

Til dette formål bliver der valgt en LM7805 spændingsregulator sat op som en 'Fixed Output' Voltage Regulator. (**lm7800**) Denne leverer et spændingsoutput på typisk 5 V og samt maks og min på 5,2 V og 4.8 V. Hvilket er inden for de definerede grænser for 5 V forsyning, på 5 V +- 0.5 V. Denne komponent kan levere 1.5 A i den valgte opsætning. Dette er mere end det beregnede strømtræk. For tabel over det samlede strømtræk for systemet, se bilagsrapportens tabel 4.3.2 på side 48. Der er derfor mulighed for at tilføje flere dele til BoldBot uden dette giver andledning til problemer i forhold til strømtræk. Herudover er der også en intern strømbegrensning, på typisk 2.4 A, til hvis der opstår en kortslutning på udgangen. Dette gør den sikker, hvis der sker en fejl et sted i systemet.

Regulatoren holder til en indgangsspænding på 35 V. Derfor er der også rig mulighed for at udskifte batteriet til et med en højere spænding, hvis behovet opstår på et senere tidspunkt. På bilagsrapportens Figur 4.7 på side 49 findes desuden også en testsimulering af strømregulatoren og på nedenstående Figur 8.6 findes et kredsløbsdiagram over den endelige opsætning for spændingsregulatoren.



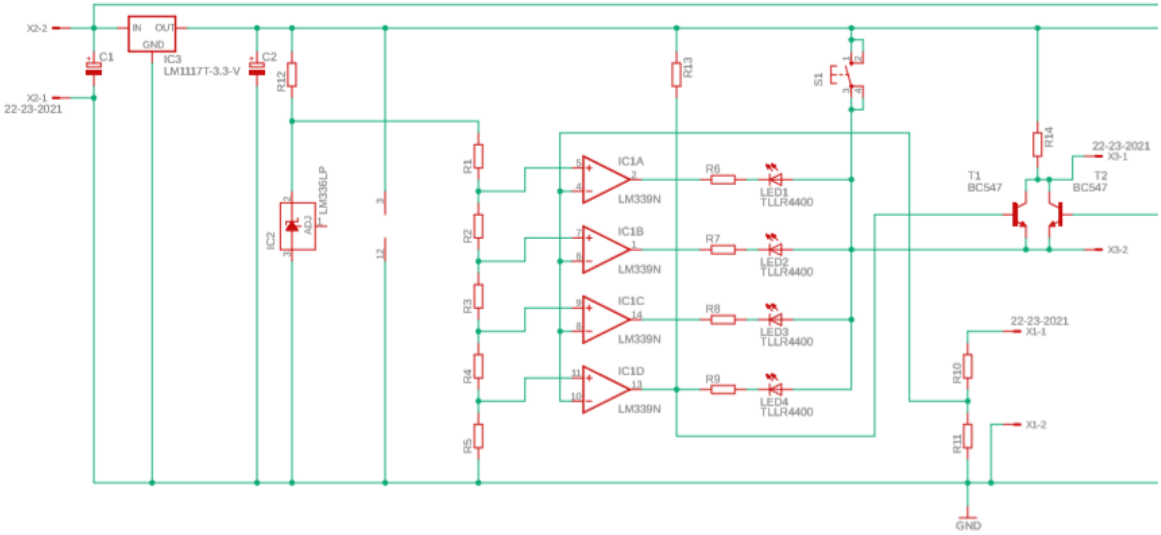

Figur 8.6: Kredsløbsdiagram for spændingsregulator

### 8.1.5 Batteriindikator

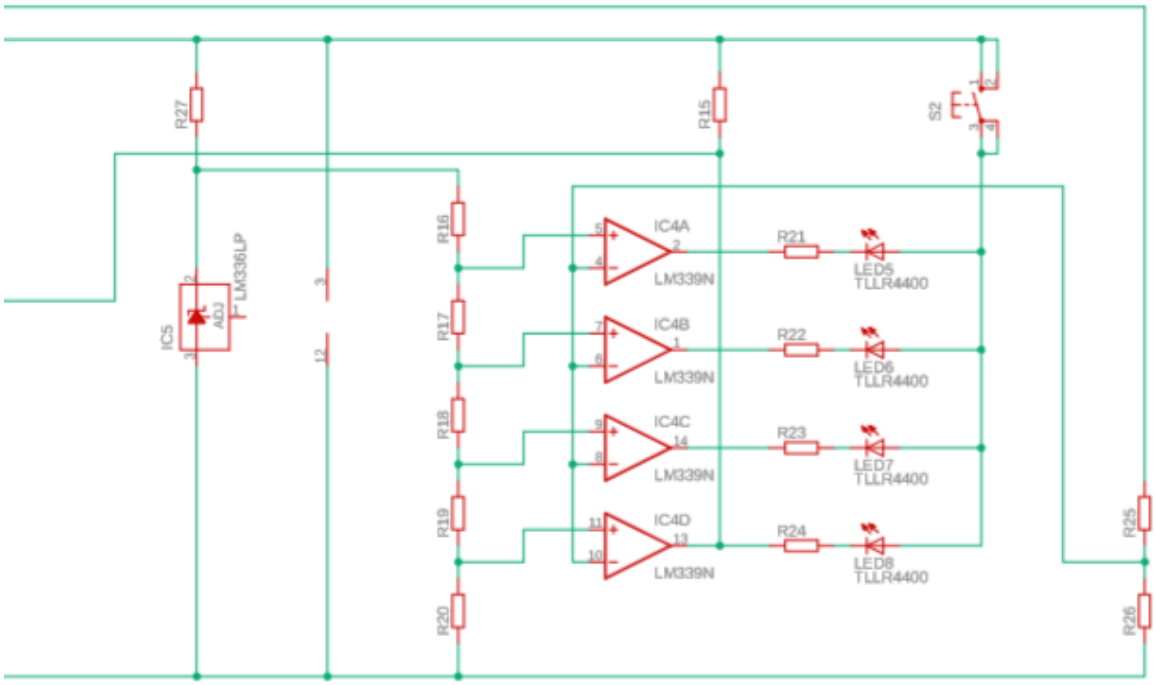
I dette afsnit beskrives der hvordan hardwaren for batteriindikatoren på BoldBot er designet. I denne del er der brugt Eagle til tegning af kredsløbsdiagrammer og Veecad til layout af print. For mere information omkring valg af komponenter, se analyseafsnit 6.1.2. 'Batteriindikator'

For overhovedet at kunne designe batteriindikatoren er målinger på batterierne nødvendige. For at kunne definere spændingsniveauer med modstande er et forsøg med batterierne designet. Der er her lavet afladningskurver for at se hvordan spændingsniveauet på batterierne udvikler sig tidsmæssigt, når de belastes. Dette skal gøres fordi et batteri ikke aflader lineært, hvilket gør at batteriet pludselig kan aflade meget hurtigt. Dette er gjort ved at aflade begge batterier hver for sig, ved at trække konstant 1A fra hvert batteri og plotte spændingen over tid. Det er besluttet for batteri-indikatoren at den skal vise følgende batteri niveauer på hhv. 20,40,60 og 80 procent. (**batteritest**).

## Hardwaredesign




Figur 8.7: Venstre side af kredsløbsdiagram for batteriindikatoren




Figur 8.8: Højre side af kredsløbsdiagram for batteriindikatoren

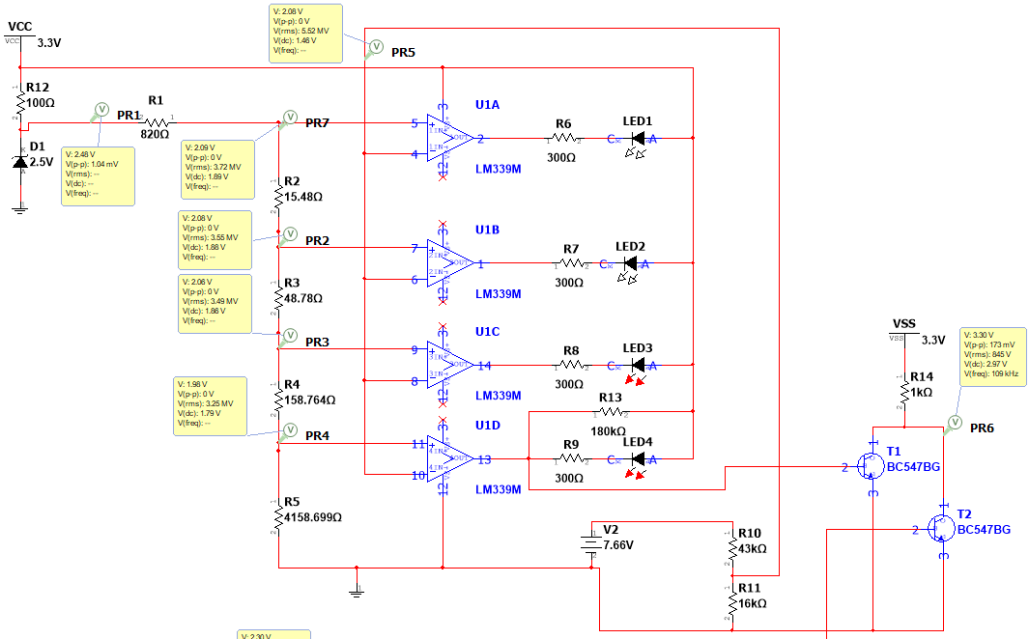
På Figur 8.7 og 8.8 kan kredsløbsdiagrammet for batteriindikatoren ses. Batteriindikatoren er opbygget med en spændingsregulator på 3.3 V til at forsyne de otte komperatorer af typen LM339, som bruges til at måle de forskellige spændingsniveauer på de to batterier. Denne type komperator kan køre på 3.3 V og så har den en lav offsetspænding på indgangene hvilket gør den ideel til vores brug. For at komperatorne kan vurdere hvilke spændingsniveauer batterierne har, er det nødvendigt at bruge en præcis spændingsreference. Til det bruges der to LM336 Zenerdioder hhv. IC2 og IC5 på Figur 8.7 og 8.8, en til hvert batteri. Foran de to zenerdioder sidder der hver en modstand for at begrænse strømmen som løber igennem den individuelle diode. På bilagsrapportens Figur 4.10 på side 52 kan udregningen for begge modstande R12 og R27 ses. Grunden til der er valgt 3.3 V forsyningsspænding er, fordi batteriindikatoren er designet til at kunne snakke med RPI'en (der kører 3.3 V logik), så den kan fortælle RPI'en hvornår der er lavt batteriniveau.

Efter spændingsreferencen bliver spændingen delt ned ved hjælp af en stor spændingsdeler som består af R1, R2, R3, R4, R5 for det ene batteri på 9.6 V og R16, R17, R18, R19, R20 for det andet 7.2 V batteri. Værdierne for disse modstande findes vha. afladningskurverne batteri-afladningsjournalen(**batteritest**).

På bilagsrapportens Figur 4.12 på side 55 kan beregningen af de forskellige procent-

niveauer ses og dertil også beregning af spændingsdeler til brug som referencespænding senere i dette afsnit. Bilagsrapportens Figur 4.14 på side 57 viser på samme måde udregninger for 9.6 V batteriet.

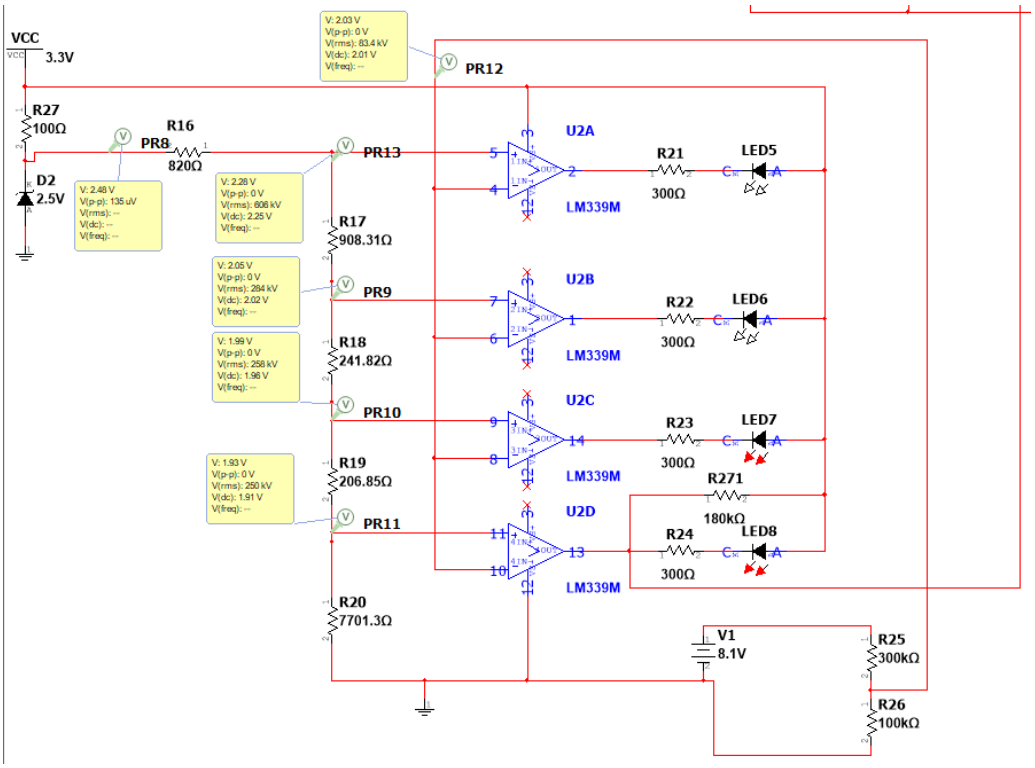
Figur 8.9 viser den ene del af det færdige kredsløb for batteriindikatoren som indikerer batteriniveauet for 7.2 volt batteriet.




Figur 8.9: Simulering af batteriindikator øverste del

Samtidig viser Figur 8.9 også en simulering over kredsløbet, der bliver tilført en batterispænding på 7.66 volt. Ifølge beregningen for 7.2 V batteriet i bilagsrapporten på Figur 4.12 på side 55 kan det ses at dette niveau ligger over 40 procent men under 60 procent, derfor forventes det at led'erne for 20 og 40 procent lyser. Hvilket kan ses på Figur 8.9 at de gør.

Den anden del af kredsløbet er identisk med det første, bortset fra nogle af modstandsværdierne. Da dette kredsløb måler batteriniveauet på 9.6 volt batteriet i stedet for 7.2 volt batteriet. Kredsløbet kan ses på Figur 8.10 hvor det er simuleret med en batterispænding på 8.1 volt. På bilagsrapportens Figur 4.14 på side 57 kan det ses øverst at 8.1 volt ligger over 40 procent, men under 60 procent. Derfor burde de 2 nederste LED'er lyse, hvilket de også gør hvis man ser på Figur 8.10.




Figur 8.10: Simulering af batteriindikator nederste del

I dette projekt er batterimåleren designet så den kan kobles til den RPi som bruges på BoldBot. Derfor er der lavet et lille transistorkredsløb som levere et højt signal på 3.3 V når batteriniveauet er over 20 procent på begge batterier og et lavt signal når batteriniveauet på et af de to batterier er under 20 procent. Beregning af transistorernes formodstand R13 og R15 er den samme som beregningen i bilagsrapporten på Figur 4.4 på side 44 i design af opsamlingsmodul.

### 8.1.6 Sonarmodul

Dette afsnit bliver ikke gennemgået grundet udeblivelse af gruppemedlem.

## 8.2 Software

Nedenstående afsnit gennemgår software design for systemet. Herunder gennemgangen af diverse UML-diagrammer, som inkluderede sekvensdiagrammer og klassediagrammer til at beskrive software designet ud fra de anvendte use cases. Software designet tager udgangspunkt i UC1, se afsnit 1.3 på side 5 'Funktionelle Krav' i bilagsrapporten. Herudover fremgår de anvendte klasser i systemet. Disse er inddelt i kategorierne domæne, boundary og controller som specificerer hvordan de forskellige klasser interagerer med hinanden. Ud fra hver subsystem i UC1 er der blevet udarbejdet en software applikations model, som bliver gennemgået i dette afsnit.

### 8.2.1 Softwareallokering

Som det kan ses på figur 7.1, bliver der allokeret software til følgende af systemets moduler:

- RobotApp: RobotBilen skal have software, der skal styre RobotBilen imod en tennisbold efter hvilke koordinater RobotApp modtager for tennisbolden. RobotApp skal også styre Opsamlermodulet, der sørger for at opsamle tennisbolden, når RobotBilen har opnået den korrekte position ifht. tennisbolden.
  RobotApp softwaren er delt ud på to microcontrollere, en RPi, og en PSoC.

- InterfaceApp: Dette er softwaren der bliver allokeret på websiden. Softwaren står for interfacet, der gør det muligt for brugeren at kommunikere med RobotBilen igennem en browser.
  Softwaren for InterfaceApp bliver allokere på samme RPi, som RobotApp.

### 8.2.2 Applikationsmodeller

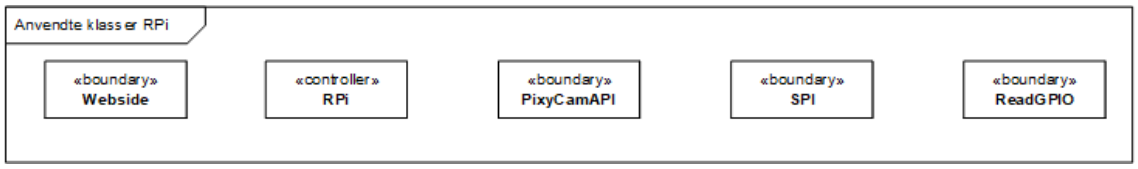
I nedenstående afsnit gennemgås det hvad der er specificeret i UC1 at systemet skal kunne og hvordan dette er blevet designet i forhold til den implementeret software. For hvert subsystem er der blevet udarbejdet en applikationsmodels ud fra de funktionelle krav, der er specificeret i UC1, se tabel 4.3.

#### Applikationsmodel RobotApp

Dette afsnit gennemgår applikationsmodellerne for RobotApp. Dette består af to subsystemer, hvor softwaren er allokeret på microcontrollerne RPi og PSoC. Dette forklares med detaljeret sekvens og klassediagrammer med udgangspunkt i UC1. Ønskes det at se tilsvarende applikationsmodel til RobotApp for UC2 henvises der til afsnit **??** på side **??** '**??** i bilagsrapporten.

#### Anvendte klasser RPi

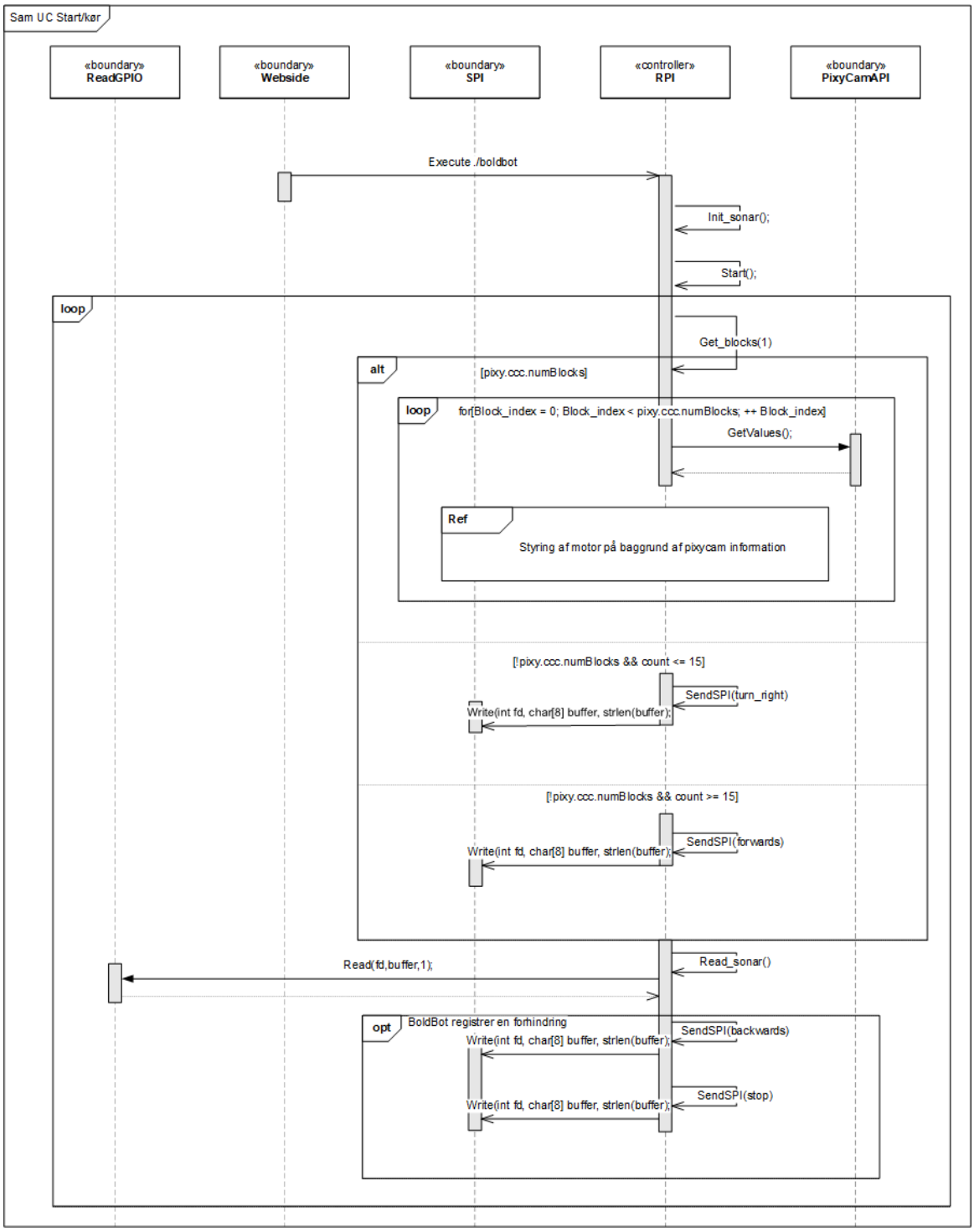
På figur 8.11 fremgår de anvendte klasser, som udgør subsystemet for RPi. Webside klassen er grænseflade mellem den implementerede webside og klassen RPi. RPI står for at modtage data fra pixyCamAPI klassen, og ud fra dette fortolke hvad der skal sendes over SPI klassen. ReadGPIO anvendes til at læse data fra en GPIO pin på RPi. For en mere detaljeret gennemgang af RPi softwaren henvises der til afsnit 5.4 på side 66 'RPI' i bilagsrapporten.




Figur 8.11: De anvendte klasser som indgår i subsystemet for RPi.

### Sekvensdiagram RPi

På figur 8.12 fremgår sekvensdiagrammet for APi. Her fremgår de forskellige funktionskald mellem modulerne i subsystemet. På sekvensdiagrammer er der for overskueligheden lavet en referance til et andet sekvensdiagram, for flere detaljer om referancen henvises der til afsnit 3.3.4 på side 34 'Sekvensdiagrammer' i bilagsrapporten.

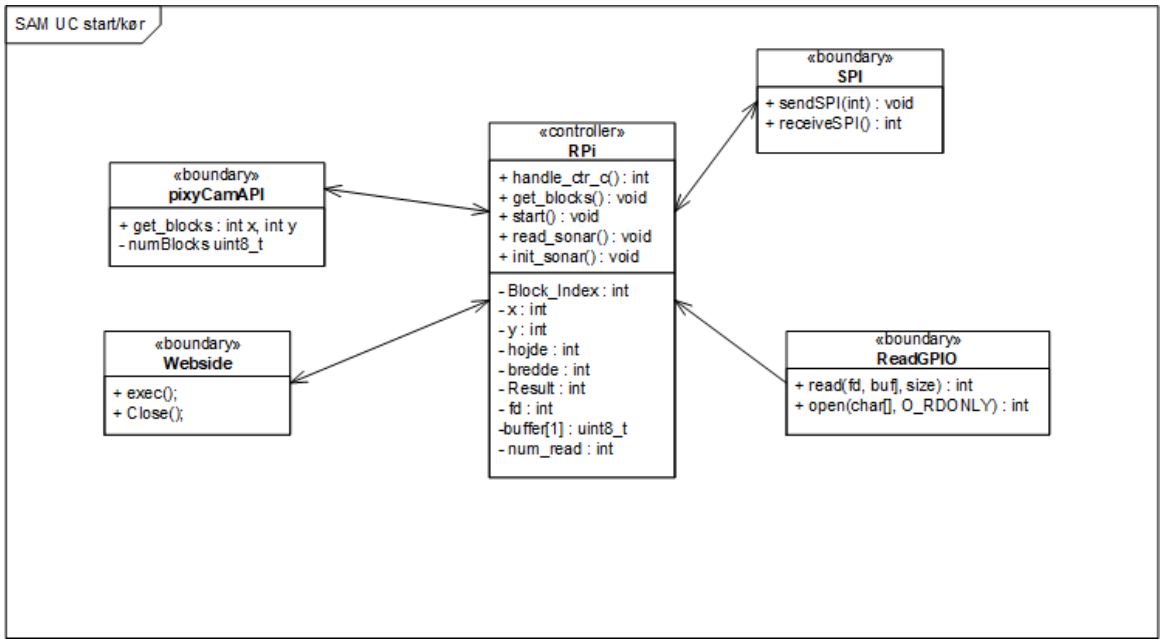



Figur 8.12: Sekvensdiagram for RPi.

Controller modulet startes fra websiden. Herefter initieres de forskellige klasser, hvorefter controller modulet på baggrund af dataen fra pixyCamAPI, sender en funktion videre over SPI. Hvad der sendes over SPI bestemmes i referencen ”Styring af motor på baggrund af pixycam information”blokken på sekvensdiagrammet. Hvis en forhindring registreres ved aflæsning af GPIO pinnen med klassen ReadGPIO, sender controller modulet den tilsvarende funktion over SPI.

### Opdateret Klassediagram RPi

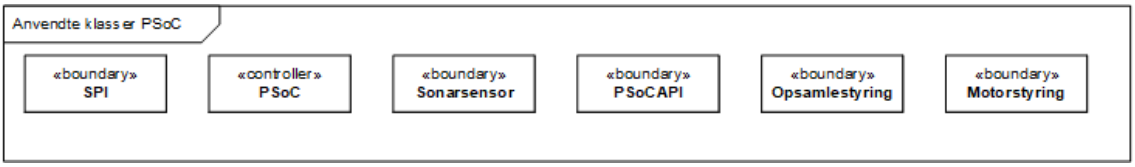
På figur 8.13 fremgår klassediagrammet med funktioner og attributter. Hvor de anvendte klasser fra figur 8.11 er blevet opdateret med de anvendte funktioner på baggrund af figur 8.12. For en mere detaljeret gennemgang af funktionerne i klassediagrammet for RPi henvises der til afsnit 3.3 på side 26 'Software arkitektur' i bilagsrapporten.




Figur 8.13: Klassediagram for RPi

### Anvendte klasser PSoC

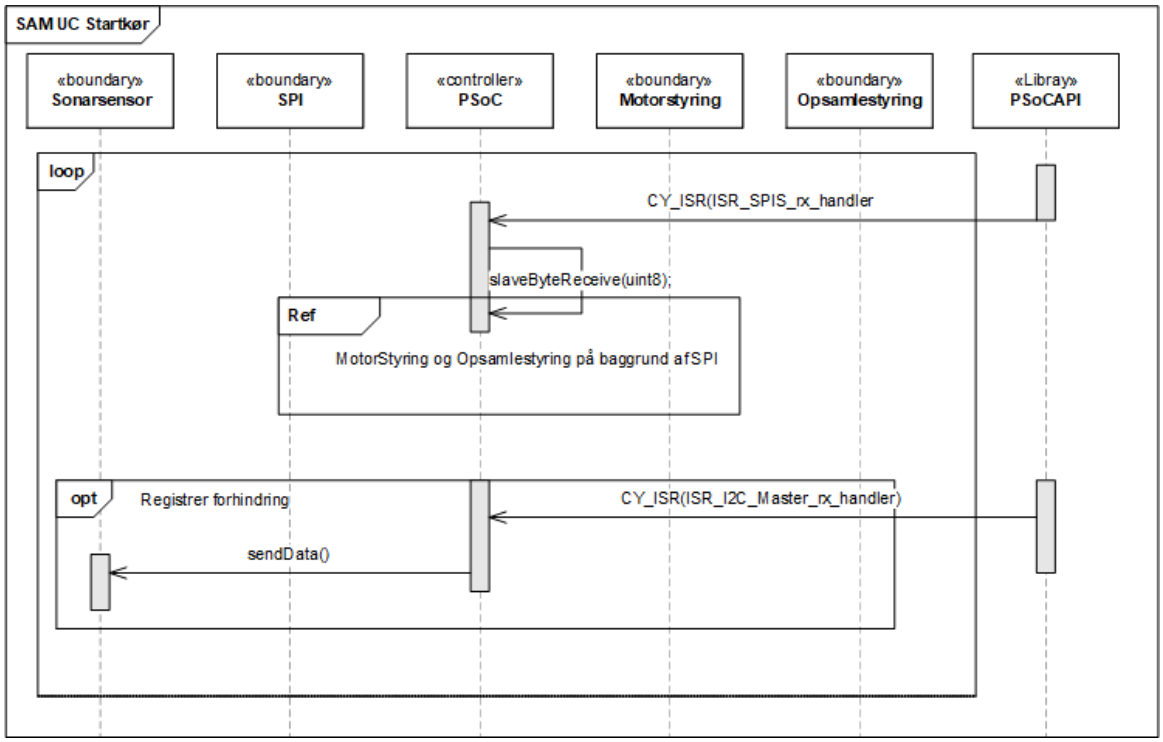
På figur 8.14 fremgår de anvendte klasser, som udgør subsystemet for PSoC. PSoC modtager data over klassen SPI, hvor den modtagne data anvendes til motorstyring og opsamlestyring. PSoCAPI anvendes til brugen af interrupt rutinerne hvor der kommer et interrupt på det modtagne data fra sonarsensoren og SPI. Sonarsensor klassen modtager data fra den anvendte sonarsenor til at registrere en forhindring. For en mere detaljeret gennemgang af PSoC softwaren henvises der til afsnit 5.2 på side 63 'PSoC' i bilagsrapporten.




Figur 8.14: De anvendte klasser som indgår i subsystemet for PSoC.

### Sekvensdiagram PSoC

På figur 8.15 fremgår sekvensdiagrammet for PSoC. Her fremgår de forskellige funktionskald mellem modulerne i subsystemet. På sekvensdiagrammer er der for overskueligheden lavet en reference til et andet sekvensdiagram, for flere detaljer om referencen henvises der til afsnit 3.3.4 på side 34 'Sekvensdiagrammer' i bilagsrapporten.

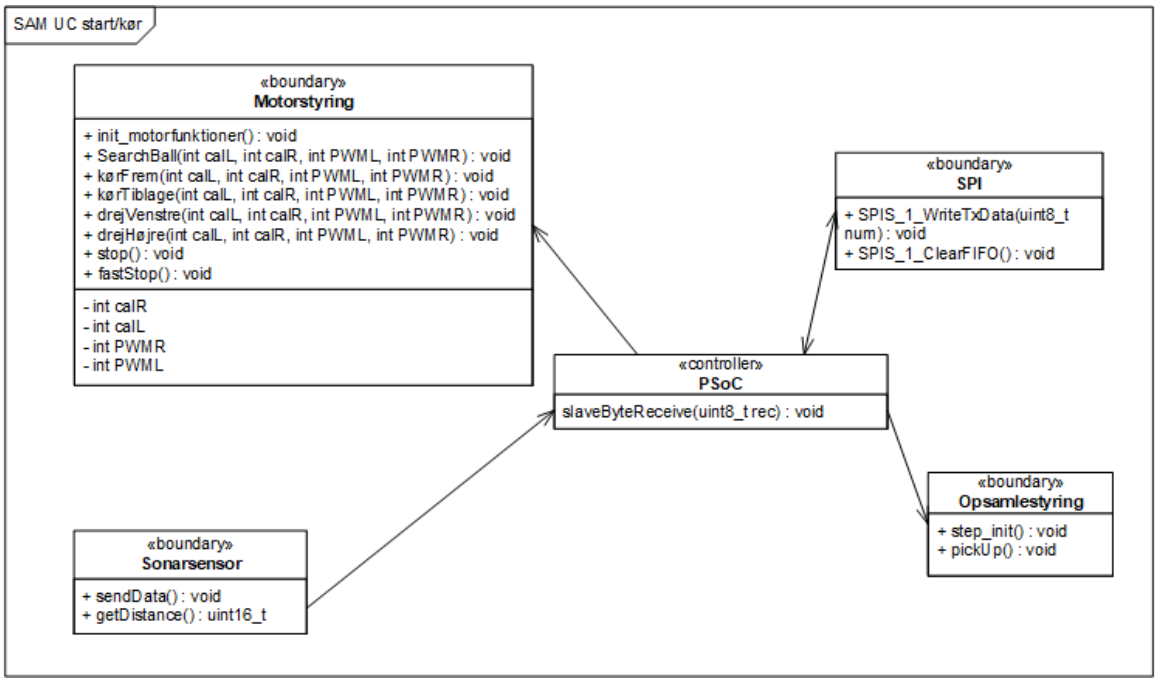



Figur 8.15: Sekvensdiagram for PSoC.

Controller modulet modtager data fra SPI klassen, hvorefter der på baggrund af den modtagne data fra SPI, kaldes en tilsvarende motorstyrings eller opsamlestyrings funktion. Hvilke funktioner der anvendes til styring af motoren og opsamlemodul fremgår af reference blokken "Motorstyring og Opsamlestyring på baggrund af SPI."Samtidig modtages der data fra sonarsensoren, hvis afstanden til en forhindring kommer under en predefineret afstand, sætter Sonarsensoren en pin høj med et funktionskald.

### Opdateret Klassediagram PSoC

På figur 8.16 fremgår klassediagrammet med funktioner og attributter. 8.15. Her er de anvendte klasse fra figur 8.14 blevet opdateret med de anvendte funktioner fra sekvensdiagrammer fra figur 8.15. For en mere detaljeret gennemgang af funktionerne i klassediagrammet for PSoC henvises der til afsnit 3.3 på side 26 'Software arkitektur' i bilagsrapporten.




Figur 8.16: Klassediagram for PSoC.

### Applikationsmodel InterfaceApp

Dette afsnit gennemgår software applikations modellen for InterfaceApp. Der vil blive opstillet et klassediagram for dette modul, hvor der efterfølgende vil blive opstillet sekvensdiagrammer og et opdateret klassediagram. Der vil kun blive redegjort for UC1 ifht. diagrammer. Ønskes der at se diagrammer for UC2, henvises der til bilagsrapporten afsnit 3.3 på side 26 'Software arkitektur'.

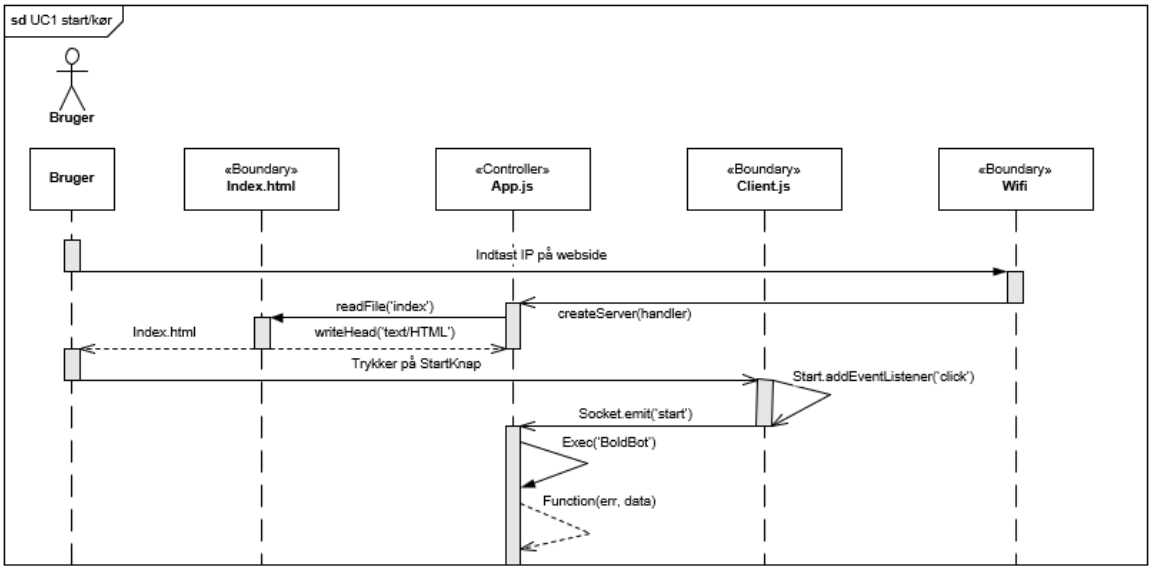
### Anvendte klasser for InterfaceApp

På figur 8.17 ses de anvendte klasser for softwaren InterfaceApp, som er en del af Webside modulet. Hver klasse har deres egen funktionalitet i forhold til websidens funktionalitet. Client.ja står for at modtage og sende beskeder fra index.html til app.ja. App.ja står for kommunikationen med RobotBilen. Index.html står for det grafiske interface, der fremgår for brugeren.


Figur 8.17: anvendte klasse for InterfaceApp UC1.

### Sekvensdiagram InterfaceApp

UC1 omhandler initieringen af BoldBot. Initieringen foregår i InterfaceApp'en som vist på figur 8.18, hvor brugeren starter med at indtaste en gyldig IP på sin browser. Browseren vil over Wifi kommunikere med RPi'en hvor InterfaceApp er allokeret. På InterfaceApp'en vil App.js oprette serveren, der henter den grafiske interface Index.html ned på serveren, som så bliver vist til brugeren. Brugeren kan herefter vælge start eller stop på websiden. I UC1 trykker brugeren på start, hvorefter Client.js bliver notificeret om modtaget besked og sender et respons til App.js, via funktionen socket.emit(''), der indeholder en specifik besked som parameter. App.js vil så starte programmet efter at have modtaget 'start' som parameter. For en mere detaljeret beskrivelse af SD diagrammet henvises til bilagsrapport afsnit 5.5 på side 67 InterfaceAPP'.

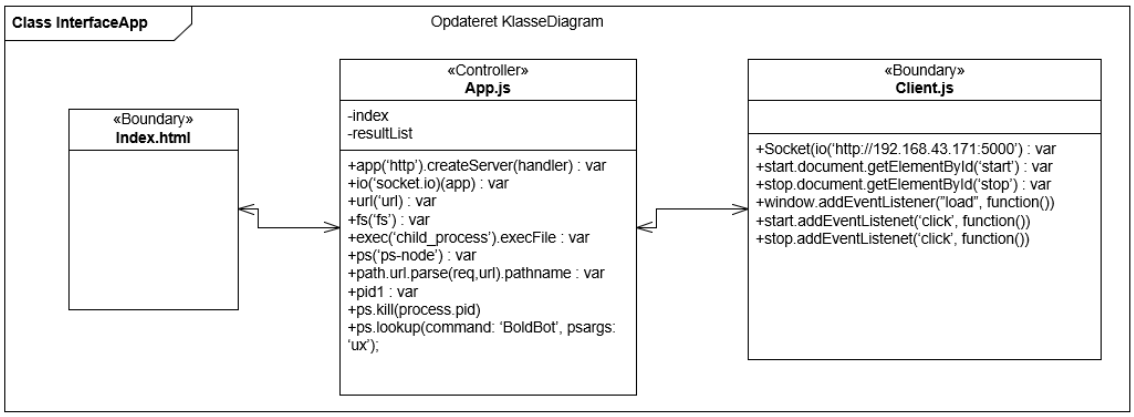



Figur 8.18: SD for InterfaceApp UC1.

### Opdateret Klassediagram InterfaceApp

Efter at have gennemløbet UC1, opdateres de anvendte klasser fra figur 8.17, med de anvendte funktioner for processen. For at gøre SD diagrammet på figur 8.18 mere overskueligt, er der valgt ikke at medtage alle de anvendte funktioner som ses på det opdateret klassediagram figur 8.19.

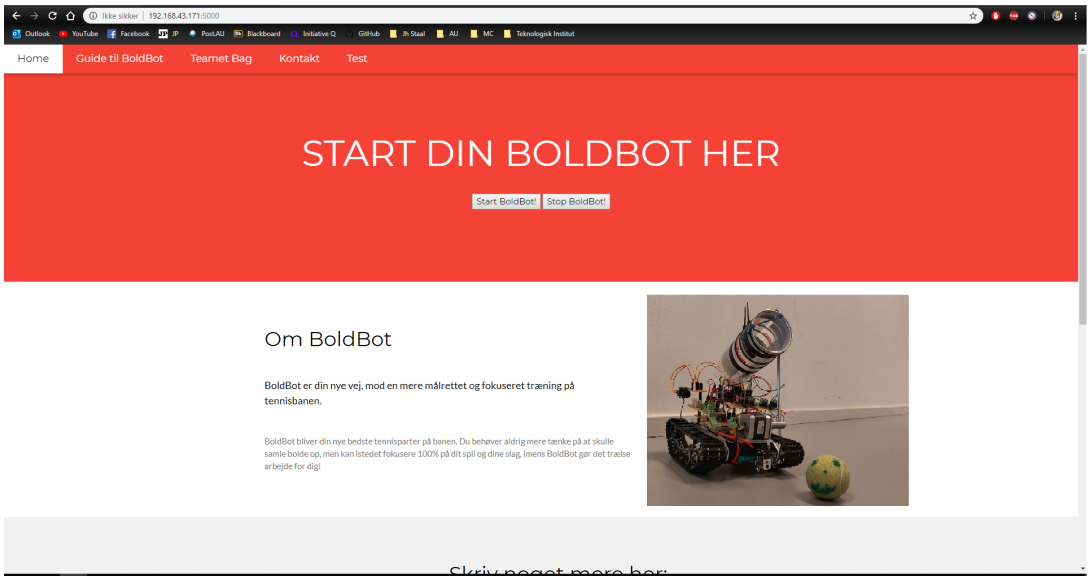
Ønskes en dybere forklaring af funktionerne på figur 8.19, henvises der til bilagsrapport afsnit 3.3.3 på side 32 ''. Implementeringen af InterfaceApp er blevet inspireret af Webserver with WebSocket (**Webserver with WebSocket**).




Figur 8.19: Opdateret Klassediagram for InterfaceApp.

### 8.2.3 Grafisk design af InterfaceApp

På figur 8.20, ses designet af websiden som brugeren interagere med. Websidens indhold, er hele indholdet af index.html klassen, som nævnt før i dette afsnit.




Figur 8.20: Grafisk design af InterfaceApp, index.html.

### 8.2.4 Protokol

I projektet er det besluttet at arbejde med SPI kommunikation mellem PSoC og RPi. Kommunikationen er opbygget med protokollen som ses på tabel 8.1. RPi sender en værdi til PSoC, som vha. en metode i main finder en case, der passer til den tilsvarende værdi, som er sendt over SPI. Mere om dette design kan ses i bilagsrapporten afsnit 5.2.5 på side 64 'Main'.

| **Kommando** | **Funktion** |
|---|---|
| 0b00000001 | searchBall |
| 0b00000010 | forwards |
| 0b00000011 | backwards |
| 0b00000100 | stop |
| 0b00000101 | turn_right |
| 0b00000110 | turn_left |
| 0b00000111 | fast_stop |
| 0b00001000 | pickup_down |
| 0b00001001 | pickup_up |
| 0b00001010 | pickup_stop |

Tabel 8.1: Protokol for SPI kommunikation

#### SPI Driver

På RPi er der blevet udviklet en driver, som gør det muligt at overføre data fra PSoC til RPi og omvendt vha. af SPI. Driveren er blevet implementeret som et kernemodul på RPi. Kernemodulet består desuden også af et overlay, som opretter et device, når RPi blivet bootet, samt allokeret major og minor nummer hertil. En mere dybdegående gennemgang af alle metoderne og det tilhørende overlay, samt hvordan det er blevet implementeret kan ses i bilagsrapporten afsnit 5.1.2 på side 61 'Design af SPI-kernemodul og device tree overlay'. Det skal bemærkes at der er blevet taget inspiration fra faget 'Hardware Abstraktioner på 3. semester. (**Devicemodel_a**; **Peter Høgh**) (**Devicemodel_b**; **Peter Høgh**)

#### GPIO Driver

På RPi er der til kommunikation mellem PSoC og RPi lavet endnu en driver. Denne driver er ligeledes også implementeret som et kernemodul. Dette kernemodul gør det kun muligt for PSoC at sende data til RPi, ved at sætte en pin til logisk 1. I den nuværende version er der kun oprettet et enkel device (gpio17) (**ase fhat schematic**), som gør det muligt for driveren at læse på denne gpio indgang. Kernemodulet bliver, ligesom

SPI kernemodulet, automatisk bootet når RPi starter, samt allokeret major og minor nummer hertil. En mere dybdegående gennemgang af alle metoderne og det tilhørende overlay, samt hvordan det er blevet implementeret kan ses i bilagsrapporten afsnit 5.1.3 på side 62 'Design af GPIO driver-kernemodul og device tree overlay'. Det skal bemærkes at der er i udviklingsfasen brugt kilder fra faget 'Hardware Abstraktioner' på 3. semester. (**Devicemodel and Busses**; **Peter Høgh**)

# 9. Test

Der vil i dette kapitel laves en kort gennemgang af alle de test der er lavet i udviklingsprocessen af BoldBot. Først i udviklingsforløbet blev alle moduler testet selvstændigt af de/den person(er) som har udviklet modulet. Herefter blev modulerne integreret i en integrationstest ét ad gangen. I alt er der lavet 6 forskellige integrationstest hvor der er blevet tilføjet et nyt modul for hvert scenarie. Til sidst er der gennemført en accepttest af begge use cases samt ikke funktionelle krav.

## 9.1 Modultest

Dette afsnit viser en kort gennemgang af modultest for alle modulerne i projektet. Der er blevet lavet en mere dybdegående test for samtlige moduler med dokumentation (billeder, grafer og figurer), som ses i bilagsrapporten afsnit 6 på side 69 'Modultest'.

| **Enhed** | **Forventet Resultat** | **Beskrivelse** |
|---|---|---|
| Batteriindikator | Der forventes, at batteriindikatoren ved tryk på to forskellige switches (en til hvert batteri) kan vise batteriniveauet for det batteri, som den pågældende switch tilhører. Dettes vises vha. fire LED'er. Niveauet skal kunne vises i 5 forskellige niveauer. | Testen blev lavet visuelt ved at sende de spændinger ind på batteriindgangene, hvorefter LED'erne tjekkes ved et tryk på switch. Det viste sig, at LED'erne ikke lyste ved de forventede spændingsniveauer, og derfor fejlede testen. Der er også beskrevet i konklusionen for modultesten fundet i bilagsrapporten afsnit 6.10.4 på side 83 'Konklusion', at der pga. tidspres ikke vil være mere fejlfinding på modulet. |

| | | |
|---|---|---|
| Levelconverter | Der forventes, at når der bliver sendt data fra RPi til PSoC, vil signalet blive konverteret fra 3.3 V til 5 V logik. Derudover forventes det, at når der bliver sendt data fra PSoC vil signalet blive konverteret fra 5 V til 3.3 V logik. | Inden levelconverteren blev taget i brug skulle det først sikres, at den omdanner signalet til den ønskede logik. Testen blev udført som en blackbox test, hvor vi sendte en puls igennem kredsløbet og afventede outputet. |
| SPI Driver | Der forventes, at der bliver sendt fra RPi data ud vha. SPI på de dertilhørende pins på RPi. | Denne test blev lavet som en blackbox test, hvor vi målte på SPI pins med Analog Discovery. Der var lavet et testprogram hvor data blev inkrementeret hver gang der blev sendt 8-bit. Denne type test gjorde det muligt at teste MOSI. MISO blev testet i forbindelse med integrationstesten mellem PSoC og RPi(se bilagsrapport afsnit 7.2 på side 96 'Scenarie 2'). |
| GPIO Driver | Der forventes, at driveren kan aflæse et 3.3 V logik signal sendt til GPIO 17. | Denne test blev udført som en blackbox test, hvor der blev vha. Analog Discovery sendt et pulssignal på GPIO17. Driveren printede herefter enten '0' eller '1' ud på terminalen. Det kunne gøres ved at aflæse på det oprettede device (cat dev/gpio17). |
| PSoC software | Det forventes, at når PSoC modtager et signal der matcher en case, vil den dertilhørende metode blive kaldt. | Denne test er lavet som en blackbox test, hvor vi tilsluttede PSoC til en computer hvor Realterm var åbnet. Realterm er forbundet med UART til PSoC, så afhængig af hvilken kommando brugeren indtaster, vil den dertilhørende metode blive kaldt i den korrekte switch-case sætning på PSoC. |

| | | |
|---|---|---|
| Sonarsensor | Det forventes at når sonarsensoren registrerer en forhindring inden for den korrekte afstand, hvilket modtages på PSoC | Denne test blev udført som en blackbox test, hvor der vha. Realterm blev tjekket at PSoC modtog den korrekte information fra sonarsensoren ved at udskrive værdierne til terminalen med den implementerede UART på PSoC. |
| Motormodul | Der forventes at motormodulet kan styre retning og hastighed på DC-motorer. | Testen er udført ved at tilslutte modulet til en Analog Discovery. Herefter bliver der testet ved at sætte PWM signaler ind, samt at toggle de 4 forskellige muligheder for indput på motor ind portene. |
| Spændingsregulator | Der forventes at spændingsregulatoren kan nedregulere fra ca. 7,2 V til ca. 5 V | Testen er udført ved at sende en 7,3 V spænding på indgangen af spændingsregulatoren, hvor batteriet skal tilkobles, og samtidig måles på udgangen af spændingsregulatoren med et oscilloskop. |
| Opsamlingsmodul | Der forventes under testen, at stepperdriver printet kan få steppermotoren til at dreje 90°. | Testen af opsamlingsmodulet er udført vha. nogle simple funktioner på en PSoC. Steppermotoren forbindes til PSoC'en og stepperdriver print som set i bilagsrapportens afsnit 6.19 på side 80 'Opstilling til modultest af opsamlingsmodul'. |
| PixyCam | Der forventes at PixyCam kan registrere bolden med den korrekte signatur. | Denne test er udført som en blackbox test, hvor PixyCam er tilsluttet til en computer hvor PixyMon er åbnet. En tennisbold placeres foran PixyCam der tjekkes om bolden registreres i PixyMon |

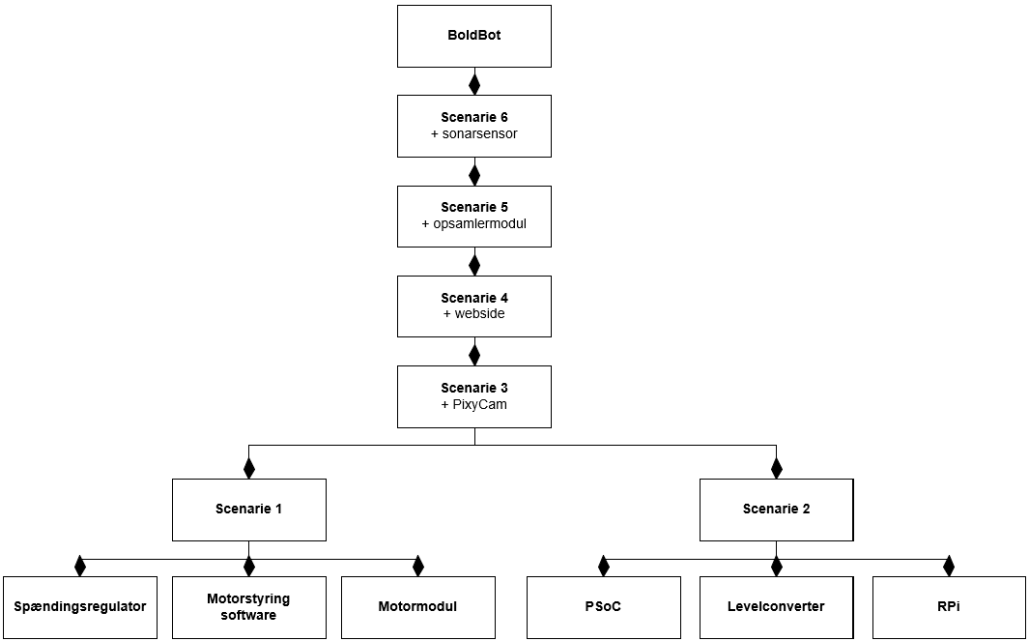
| | | |
|---|---|---|
| Webside | Det forventes i testen, at en webside med med IP-adresse (http://192.168. - 43.171:5000) kan åbnes vha. en webbrowser. Derudover skal websiden kunne starte og stoppe BoldBot vha. knapper på hjemmesiden. | Testen blev udført som en blackbox test, hvor brugeren først aktiverede app.js på RPi og herefter afventede en visuel bekræftelse på terminalvinduet på RPi. Herefter blev det testet, om det kørende program på RPi kunne termineres. |
| RPi software | Der forventes at RPi regsitrerer bolden, og printer korrekte koordinater ud, afhængigt af boldens placering. | Til test af RPi software, kobles PixyCam til RPi, og et testprogram opstartes. En bold sættes derefter foran PixyCam og koordinater bliver registreret. Bolden bliver derefter rykket på, for at se om koordinater ændrer sig korrekt. |

Tabel 9.1: Modultest fortsat

## 9.2 Integrationstest

Dette afsnit viser en kort gennemgang af integrationstesten for alle scenarierne. Vi har anvendt bottom-up metoden til at teste alle vores moduler i en struktureret og præcis rækkefølge(**Systemtest**; **ASE**). En mere dybdegående test for scenarierne hvori dokumentation (billeder, grafer og figurer) for testene kan ses i bilagsrapporten afsnit 7 på side 94 'Integrationstest'.

Figur 9.1 viser en model af hvordan det er valgt at lave integrationtesten. Der er først valgt at samle to større moduler, hvorefter der er blevet tilføjet ét modul ad gangen, for at gøre det lettere at fejlfinde på evt. fejl i både kode og hardware.
Det skal bemærkes at der ikke er lavet en integrationtest på batteriindikatoren, eftersom den ikke har bestået modultesten.




Figur 9.1: Integrationtest illustration

| Enheder | Forventet Resultat | Beskrivelse |
|---|---|---|
| **Scenarie 1**<br>Motormodul<br>+ Spændingsregulator<br>og motorstyring software | Der forventes at der vha. kald af funktioner fra Realterm bliver kaldt funktioner, som aktiverer både DC-motor og steppermotor. | **Opstilling:** Motorprint, strømforsyning og PSoC med motorstyring software monteres på BoldBot. Herefter tilsluttes PSoC til en computer, som kalder alle funktioner som skal testes.<br>**Test:** Funktionerne kaldes vha. Realterm med kommandoerne 1-10. Der anvendes UART frem for SPI, da det er lettere at kalde den korrekte funktion, samt at kunne fejlfinde korrekt. Motorstyrings funktionerne til styring af DC-motorne bliver kaldt i en systematisk orden. Efter en funktion bliver kaldt bliver motorne kalibreret, således at den bakker, kører frem og drejer korrekt i et passende tempo. Der er fire måder at justere en funktionerne på: PWMR, PWML, calR og calL. |
| **Scenarie 2**<br>PSoC<br>+ RPi<br>og levelconverter. | Der forventes at et 8-bit signal bliver sendt over SPI. Levelconverteren skal ændre logikken fra 3.3 V til 5 V, og omvendt. Det sendte signal, der bliver sendt fra RPI vha. et testprogram, skal være det samme som bliver modtaget i PSoC. | **Opstilling:** PSoC og RPi bliver forbundet til levelconverteren. PSoC bliver forbundet til B indgangen og RPi blev forbundet til A-indgangen (**txs0108e**).<br>**Test:** Der laves et testprogram på RPi (**Bilag**), hvori en variabel bliver inkrementeret hvergang der bliver sendt 8-bit data. Modtaget data bliver printet ud i terminalvinduet på Realterm. |

Tabel 9.2: Integrationstest - fortsætter på næste side.

| | | |
|---|---|---|
| **Scenarie 3**<br>Motormodul<br>+ Spændingsregulator<br>+ motorstyring software<br>+ RPI<br>+ levelconverter<br>og PixyCam. | Der forventes, at Pixy kan registrere en tennisbold og herefter sende tennisbolds koordinater, som RPi fortolker og herefter kalder funktioner på RPi vha. SPI. | **Opstilling:** PSoC, RPi, levelconverter og PixyCam bliver monteret på BoldBot, hvor alle moduler fra scenarie 1 allerede er placeret.<br>**Test:** Der laves en visuel test, hvor en bold placeres indenfor BoldBots rækkevide. Der testes flere forskellige scenarier, hvor det eneste der ændres er boldens placering. |
| **Scenarie 4**<br>Motormodul<br>+ Spændingsregulator<br>+ motorstyring software<br>+ RPI<br>+ levelconverter<br>+ PixyCam<br>og webside. | Det forventes at BoldBot kan startes fra websiden, hvor Boldbot har denne samme funktionalitet som i scenarie 3. Hvor dette tidligere i de andre scenarier er blevet gjort i et terminal vindue. | **Opstilling:** Boldbot monteres med samme moduler som i scenarie 3.<br>**Test:** Der laves en visuel test hvor det ses om BoldBot kan opstartes fra websiden. |
| **Scenarie 5**<br>Motormodul<br>+ Spændingsregulator<br>+ motorstyring software<br>+ RPI<br>+ levelconverter<br>+ PixyCam<br>+ webside<br>og opsamlermodul. | Det forventes at BoldBot kan opsamle en tennisbold med opsamlemodulet på baggrund af den korrekte information fra RPi over SPI efter at blive tændt fra websiden. | **Opstilling:** Opsamlermodulet bliver monteret på BoldBot, hvor alle moduler fra scenarie 4 allerede er placeret på BoldBot.<br>**Test:** Der laves en visuel test hvor det ses om BoldBot opsamler en tennisbold, hvis tennisbolden er i den korrekte position. |

| | | |
|---|---|---|
| **Scenarie 6**<br>Motormodul<br>+ Spændingsregulator<br>+ motorstyring software<br>+ RPI<br>+ levelconverter<br>+ PixyCam<br>+ webside<br>+ opsamlermodul og sonarsensor. | Det forventes at BoldBot kan registrere en forhindring, hvorefter at have registreret en forhindring, og herefter køre baglæns. | **Opstilling:** Sonarsensoren modulet monteres på BoldBot sammen med de andre moduler fra scenarie 5.<br>**Test:** Der laves en visuel test af sonarsensoren hvor en fod placeres foran sonarsesoren, hvor det ses om BoldBot reagere som forventet. |

Tabel 9.2: Integrationstest fortsat

## 9.3 Accepttest

I det kommende afsnit vil accepttest for version 1.2 kort forklares. En mere dybdegående accepttest for både version 1.1 og 1.2 kan ses i bilagsrapporten afsnit 8 på side 102 'Accepttest'.

| **Enheder** | **Beskrivelse** |
|---|---|
| UC1 | **Opstilling:** Accepttesten for UC1 har samme opstilling som scenarie 6 i integrationstesten, hvor samtlige moduler er integreret.<br>**Test:** Testen er foregået som beskrevet i accepttesten version 1.2 i bilaget. |
| UC2 | **Opstilling:** Accepttesten for UC2 har samme opstilling som scenarie 6 i integrationstesten, hvor samtlige moduler er integreret.<br>**Test:** Testen er foregået som beskrevet i accepttesten version 1.2 i bilaget. |
| Ikke funktionelle krav | **Opstilling:** Accepttesten for ikke funktionelle krav har samme opstilling som scenarie 5 i integrationstesten, hvor samtlige moduler er integreret.<br>**Test:** Testen er foregået som beskrevet i accepttesten version 1.2 i bilaget. |

Tabel 9.3: Accepttest

# 10. Resultater

I det kommende kapitel vil alle resultater som er opnået i løbet af projektet blive fremført. Der vil først blive præsenteret resultater for alle gennemførte modultest. Herefter vil resultaterne for de seks scenarier i integrationstesten (afsnit 9.2 på side 56 blive præsenteret. Til sidst i afsnittet vil resultaterne for accepttest (version 1.2) fremføres.

## 10.1 Modultest

Tabel 10.1 viser resultaterne opnået i modultesten. Der er fremvist resultater for samtlige modultest, som er fundet frem til i afsnit 9.1.

| **Enhed** | **Observation** |
|---|---|
| Levelconverter | Levelconverteren gør det muligt for PSoC og RPi at kommunikere med hinanden vha. SPI. Logikken bliver korrekt konverteret fra 3.3 V til 5 V og omvendt. Signalet bliver desuden heller ikke forstyrret eller på anden måde ændret. |
| SPi Driver | SPI driveren på RPi gør det muligt at sende data fra Master(RPi) til slave(PSoC). Desuden kan driveren også modtage data fra slaven. Kernemodulet samt overlay bliver automatisk bootet ved opstart af RPi, hvor både major- og minor nummer bliver tildelt. |
| GPIO driver | GPIO driveren aflæser korrekt om der bliver sendt 1/0 til gpio17 på RPi. Kernemodulet samt overlay bliver automatisk bootet ved opstart af RPi, hvor både major- og minor nummer bliver tildelt. |
| PSoC software | De korrekte metoder bliver kaldt på PSoC vha. en switch-case løkke, som bliver kaldt i main. |
| Sonarsensor | Sonarsensoren kunne registrere et objekt tættere på den bestemte afstand på 40 cm. Dette blev udskrevet på Realterm med UART på PSoC. |
| Motormodul | Motorprintet gør det muligt at justere fart og retning for de to DC motorer. |
| Spændingsregulator | Spændingsregulator printet formår at nedregulere de ca. 7 V DC til ca. 5 V DC. |
| Batteriindikator | Batteriindikatoren formår ikke at visualisere de korrekte niveauer for batteriniveauet. Den kan godt vise forskellige niveauer, men det er ikke de korrekte. |

| | |
|---|---|
| Opsamlingsmodul | Steppermotoren i opsamlingsmodulet kan flytte sig det antal grader, som er ønsket til at bevæge armen på opsamlingsmodulet. |
| PixyCam | PixyCam kan registrere tennisbolde korrekt, og sende koordinater vha. en USB forbindelse til en computer. |
| Webside | En webside for BoldBot kan åbnes vha. en webbrowser. Websiden viser to 'knapper' som enten kan starte eller stoppe BoldBot. Data bliver overført vha. WiFi. |
| RPi software | BoldBot kører korrekt imod BoldBot, når bolden er lige for. Under alle testscenarier udfører BoldBot korrekte funktioner. |

Tabel 10.1: Modultest resultater

## 10.2 Integrationstest

I tabel 10.2 er alle resultater fremgjort. Resultaterne bygger på observationer af tests, der er fundet i afsnit 9.2 på side 56 'Integrationstest'.

| **Enheder** | **Observation** |
|---|---|
| Scenarie 1 | Testen viste at motorfunktionerne korrekt kan kaldes fra PSoC. Derudover blev begge DC-motorer også kalibreret korrekt, således at BoldBot kører i passende tempo i lige linje. |
| Scenarie 2 | Testen viste at RPi og PSoC kan kommunikere med hinanden. Testen viste desuden også, at der kan kommunikeres begge veje. |
| Scenarie 3 | Testen viste at RPi på baggrund af data fra PixyCam, sendte det korrekt over SPI, Hvorefter BoldBot håndterede de forskellige positioner for tennisbolden korrekt, via den sendte data fra RPi til PSoC. |
| Scenarie 4 | Testen viste at BoldBot kunne opstartes med websiden og herefter eksekvere på samme måde som i scenarie 3. Derudover kunne BoldBot ligeledes blive stoppet via websiden. |
| Scenarie 5 | Testen viste at BoldBot kunne opsamle en tennisbold med opsamlermodulet, da tennisbolden var i den korrekt position i forhold BoldBot. |
| Scenarie 6 | Testen viser at når sonarsensoren var integreret med resten systemet, virkede det ikke korrekt. |

Tabel 10.2: Integrationstest resultater

## 10.3 Accepttest

Efter en fuldendt integrationstest, vil resultaterne for accepttesten blive præsenteret i tabel 10.3. De viste resultater stammer fra version 1.2. I tabel 10.4 ses den fulde accepttest for UC1 hovedscenarie.

| **Unit under Test** | **Observation** |
|---|---|
| Use Case 1 | BoldBot kan korrekt søge efter tennisbolde og og herefter lave en succesfuld opsamling. På tabel 10.4 ses resultatet af accepttest for UC1 hovedscenarie. Dog kan extension 1 ikke udføres korrekt, pga manglende modul integration, se tabel 10.2. |
| Use Case 2 | Brugeren kan slukke for BoldBot succesfuldt efter BoldBot er startet. |
| Ikke funktionelle krav | Alle ikke funktionelle krav er efterlevet, hvilket fremgår af afsnit 4.3. BoldBot har den korrekte boldkapacitet, samt opfylder de korrekte højde, bredde og længde mål. Dog kan BoldBot ikke korrekt registrere ukendte objekter uden det forstyrrer resten af systemet. |

Tabel 10.3: Accepttest resultater for version 1.2

| **Use case under test** | | Start/Kør | | |
|---|---|---|---|---|
| **Sceanrie** | | Hovedscenarie | | |
| **Prækondition** | | BoldBot er inaktiv | | |
| **No.** | **Handling** | **Forventet resultat** | **Faktisk resultat** | **Vurdering** |
| 1 | Bruger indtaster 192.168.43.171:5000 på en valgfri browser. | Webside vises på skærmen. | Webside ses i browser. | OK |
| 2 | Bruger trykker på 'Start Boldbot' knappen på hjemmesiden. | BoldBot tænder og begynder at søge efter tennisbolde. | BoldBot begynder at søge efter tennisbolde. | OK |
| 3 | Bruger placerer tennisbold 100cm $\pm 2cm$ foran BoldBot. Afstanden måles med et målebånd med nøjagtighed på $\pm 1cm$ | BoldBot kører mod bolden og opsamler i beholder. | BoldBot kører hen til tennisbolden og opsamler bolden. | OK |

| | | | | |
|---|---|---|---|---|
| 4 | Bruger placerer tennisbold 60 cm $\pm 2cm$ foran BoldBot. Afstanden måles med et målebånd med nøjagtighed på $\pm 1cm$ | BoldBot kører mod bolden og opsamler i beholder. | BoldBot kører hen til tennisbold og opsamler bolden. | OK |

Tabel 10.4: Accepttest af Use Case 1 - Start/Kør

# 11. Diskussion

Efter at både modultest og integrationtest for samtlige moduler er gennemført og resultaterne er blev gjort op, kan det ses at den overordnede funktionalitet beskrevet i projektformuleringen er opnået. Der er blevet udviklet et produkt som løser den beskrevne projektformulering i et tilfredsstillende omfang, dog med få mangler. Derudover har den foreslåede løsning, der blev beskrevet først på semestret, været den løsning der er blevet arbejdet ud fra i løbet af hele semestret. Løsningen var præcis, da den var passende i størrelse for et projekt af denne størrelse. Det betyder at projektet har passet perfekt i forhold til den tid som er blevet afsat til et projekt af denne størrelse. En sådan præcis planlægning skyldes hovedsageligt god kommunikation blandt gruppemedlemmerne samt en tidsplan, som er blevet overholdt. Det betyder dog ikke at den ikke er blevet skubbet og finjusteret, men gruppen har altid været opmærksom på, at der var deadlines som skulle overholdes, og hvis dette ikke var muligt, skulle det vendes med resten af gruppen.

De succesfulde resultater skyldes hovedsageligt dybdegående test af samtlige moduler. Alle modultests har fundet de fleste fejl i modulet. Denne gennemgang af fejl i modulerne har gjort, at integrationtesten har forløbet uden større problemer og de fejl som har opstået har været forholdsvis lette at lokalisere, eftersom modulerne er blevet tilføjet i en systematisk rækkefølge. Denne faste struktur af at udføre en integrationstest har haft en stor betydning for de opnåede resultater, og at de fleste krav er opfyldt. Hvis der kigges nærmere på de enkelte scenarier, kan det ses i resultaterne for de fem første scenarier at de passer overenes med de forventede resultater beskrevet i afsnit 9.2 på side 56 'Integrationstest', men i det 6. scenarie opstår der problemer. Derfor kan det hurtigt ses, at det er i det sidste tilføjede modul i integrationstesten (sonarsensor) der kommer problemer i systemet, hvilket er mærkeligt eftersom modultesten endte med korrekte resultater. Det at den ikke virker korrekt når den bliver integreret med resten af systemet, kan skyldes mangel af fejlfinding af koden på PSoC, men da den pågældende person som har haft ansvaret for dette modul ikke har haft tid, blev det valgt ikke at lave modulet færdigt. Ydermere kan det ses for resultatet til batteriindikatoren, at den ikke virker korrekt i modultesten. Eftersom den ikke virker korrekt heri, er det valgt ikke at prøve at tilføje den i en integrationstest, da den ikke vil blive opfyldt.
Resultaterne fra scenarie 1-5 har generelt været det som er blevet forventet i henholdsvis modultests integrationtests.

Overordnet er det meget succesfulde resultater. Projektet er endt ud i et funktionelt produkt, de eneste funktionaliteter som ikke er blevet integreret i systemet er sonar sensoren og batterindikatoren. Ser man borte fra dette, er der blevet udviklet et fuldt funktionelt system, som har en vis begrænsning. Dette afspejler sig også i accepttesten, hvor 13 ud af de 15 krav er vurderet til at være OK.

Projektet har været et betydningsfuldt fag på 3. semester, eftersom der er blevet anvendt teori fra fag brugt på 3. semester. Disse fag har givet en basis forståelse for bl.a. forskellige kommunikationsformer, så der kunne blive taget en oplyst beslutning om hvilken kommunikationsmetode der skulle anvendes mellem de forskellige moduler. Derudover er en stor del af den implementerede hardware bygget op omkring viden som er kommet fra fag på både 2.- og 3. semester.

# 12. Konklusion

Dette projekt har haft fokus på implementering af et autonomt system, der muliggør opsamlingen af tennisbolde på en tennisbane. Til dette er der blevet udarbejdet en prototype af et system/produkt til at opfylde kravene opstillet i starten af projektet med MoSCoW analysen. Ydermere til at implementere produktet har der været taget højde for læringsmålene for semesterprojektet, hvor produktet er blevet implementeret med en brugergrænseflade, samt de øvrige krav omkring anvendelsen af en RPi og en PSoC. Det kan konkluderes at BoldBot lever op til 'Must' kravene opstillet i MoSCoW analysen, altså den funktionalitet produktet skulle have. De andre krav herunder should, could og won't er der ikke blevet arbejdet yderligere på og disse funktionaliteter er ikke blevet implementeret, da der har været fokus på systemets "Must"krav. Derfor står de andre krav åben til fremtidigt arbejde. Med en accepttest blev de endelige kravspecifikationer for produktet testet og resultaterne ses i afsnit 10 på side 61. Resultaterne fra accepttesten viste overordnet at produktet opfyldte de fleste af de funktionelle og ikke-funktionelle krav sat til produktet. Udover at den implementerede sonarsenor ikke virkede, opfyldte produktet de øvrige funktionelle og ikke-funktionelle krav.
Problemet som dette produkt skulle løse ifht. projektformuleringen, er løst til en vis grad. En person vil stadig være i stand til at samle bolde hurtigere end det udviklede produkt, da produktet mangler en del forbedring på selve opsamlermodulet, og dets generelle effektivitet.
Generelt har indsatsen i forhold til projektarbejdet og procesarbejdet fungeret godt. Den agile arbejdsmetode har været en af grundende til det gode projektarbejde, med scrum som tovholder for den iterative arbejdsgang. Dette har betydet at det har været muligt løbende at kunne ændre i løsningerne i forhold til de opstillede krav til produktet, således projektet blev sluttet med fine resultater fra accepttesten.

# 13. Fremtidigt Arbejde

Fremtiden for BoldBot vil være at få implementeret nye funktionaliteter i systemet. Visionen er at få et fuldt automatiseret system, som inkluderer at BoldBot ikke bare skal kunne navigere en tennisbane, men at der skal være en homebase. Homebase skal fungere som en en tømningsstation og en ladestation, hvor Boldbot skal kunne køre hen og tømme sig selv eller påbegynde ladning når batteriet er lavt. Ladningen vil optimalt skulle foregå trådsløst, så der ikke er behov for en bruger til at tilkoble nogle ledninger. Hvis BoldBot autonomt skal kunne finde denne homebase, vil der også være behov for at implementere et GPS system, så den ved præcis hvor den befinder sig, for at kunne finde den hurtigste vej tilbage.
Udover disse nye funktionaliteter, ville et forbedret opsamlingsmodul være høj priotet. Da opsamlingsmodulet i dette stadie kun kan opsamle en enkelt bold, på grund af stepper motorens styrke, vil en stærkere steppermotor være en ideel forbedring, sammen med et større rør til opsamlingsmodulet. Dette ville gøre det muligt for BoldBot at opbevare og opsamle flere bolde, så der ikke er behov for tømning så ofte som der er nu. Disse forbedringer vil tilsammen skabe en ny og forbedret version af BoldBot, som vil være fuldt autonomt.

# Bibliografi

[1] PRJ3, GR9 *Bilagsrapport*.

Dette dokument indeholder Krav, Arkitektur, Design og Test i detaljer.

[2] *Datablad: Ase fhat schematic, ASE, 2019.*

[3] *Datablad: lm7800, Texas Instruments, 2016.*

[4] *Datablad: RPI ZERO V1.3 - reduced, ASE, 2019.*

[5] *Datablad: txs0108e, Texas Instruments, 2019.*

[6] *Interview: Alexander Lykou, Aarhus 1900, Højbjerg, 20/02/2019.*

[7] *Journal: Batteriafladning, Christian Karl Oscar Lind Vie Madsen, 2019*

[8] PRJ3, GR9 *Procesrapport*.

Indeholder en procesbeskrivelse med overvejelser og valg i projektet forløb fra start til slut.

[9] *PDF: SystemTest - Bottom-up, Forfatter: I2ISE, Sti: Kilder/ASE/SystemTest - Bottom-up.pdf*.

Dette dokument indeholder en beskrivelse af gennemgang af integrationstest og bottom-up test.

[10] *PDF: Vejledning_for_gennemfoerelse_af_projekt_2_V1_00, Forfatter: KBE, Dato: 20-05-2015, Sti: Kilder/ASE/Vejledning_for_gennemførelse_af_projekt_2_v1_00*.

Dette dokument indeholder vejledning til gennemførelse af 2. semester projekt.

[11] *PDF:06a_Device_Model - agenda.pdf, Forfatter: Peter Høgh, Sti: Kilder/A-SE/06a_Device_Model*.

Dette dokument indeholder en beskrivelse af udvikling af en platform driver (GPIO driver).

[12] *PDF:06b_Device_Tree - agenda.pdf, Forfatter: Peter Høgh, Sti: Kilder/A-SE/06b_Device_Tree - agenda.pdf*.

Dette dokument indeholder en beskrivelse af udvikling af en platform driver (GPIO driver).

[13] *PDF:07_Device_Model_and_Busses - agenda.pdf, Forfatter: Peter Høgh, Sti: Kilder/ASE/07_Device_Model_and_Busses - agenda.pdf.*

Dette dokument indeholder en beskrivelse af udvikling af et SPI kernemodul (SPI driver).

### 13.0.1 Weblinks

[14] *FURPS+ - metoden*

https://en.wikipedia.org/wiki/FURPS

[15] *MoSCoW - metoden*

https://en.wikipedia.org/wiki/MoSCoW_method

[16] *SysMl og UML*

https://en.wikipedia.org/wiki/Systems_Modeling_Language

[17] W3Schools, Webserver with WebSocket. Tilgængelig på: https://www.w3schools.com/nodejs/nod