

I3PRJ3: Semesterprojekt 3

Bilag

Gruppe 9

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

Vejleder:

Michael Sørensen Loft

ml@ase.au.dk

29. juni 2019

Indhold

1	Kravspecifikation	3
1.1	Systembeskrivelse	3
1.2	MoSCoW	3
1.3	Funktionelle Krav	5
1.4	Ikke-Funktionelle Krav	7
1.5	FURPS	7
1.6	Udviklingsværktøjer	8
2	Analyse	9
2.1	Hardware	9
2.2	Software	11
3	Arkitektur	14
3.1	Overordnet systemarkitektur	14
3.2	Hardware arkitektur	17
3.3	Software arkitektur	26
4	Hardware design og implementering	40
4.1	Motormodul	40
4.2	Opsamlingsmodul	42
4.3	Spændingsregulator	47
4.4	Batteriindikator	50
4.5	Sonarmodul	60
5	Software design og implementering	61
5.1	RobotApp	61
5.2	PSoC	63
5.3	PixyCam	65
5.4	RPI	66
5.5	InterfaceAPP	67
6	Modultest	69
6.1	PSoC motorstyring	69
6.2	Levelconverter	69
6.3	SPI Driver	71
6.4	GPIO driver	72
6.5	PSoC software	74
6.6	Sonarsensor	75
6.7	Motormodul	75
6.8	Spændingsregulator	78

6.9	Opsamlingsmodul	80
6.10	Batteriindikator	81
6.11	PixyCam	83
6.12	Webside	84
6.13	RPi software	86
7	Integrationstest	94
7.1	Scenarie 1	94
7.2	Scenarie 2	96
7.3	Scenarie 3	98
7.4	Scenarie 4	99
7.5	Scenarie 5	100
7.6	Scenarie 6	101
8	Acceptttest	102
8.1	Acceptttest 1.1	102
8.2	Acceptttest 1.2	104
	Bibliografi	106

1. Kravspecifikation

1.1 Systembeskrivelse

BoldBot er et system bestående af 3 overordnede moduler. Det første modul er websiden som er brugergrænsefladen, det andet er Robotbil som dækker styringen af BoldBot efter start og det sidste er opsamlingsmodulet.

Brugeren åbner en browser og går til websiden. På websiden bliver to muligheder præsenteret, Start og Stop BoldBot. Brugeren trykker på Start BoldBot og websiden starter Robotbil programmet på RPi. Så begynder BoldBot at søge efter en bold, ved at dreje om sig selv og køre fremad. Hvis BoldBot ser en bold, vil den køre forsigtigt imod bolden indtil den når en specifik opsamlingsposition. Når opsamlingspositionen er indtaget vil opsamlingsmodulet aktiveres, og et rør føres ned over bolden og den opsamles. I tilfældet at BoldBot ikke rammer bolden med røret bakker den tilbage og prøver igen. Hvis BoldBot møder en forhindring på vej imod bolden, vil den bakke tilbage og begynde at søge igen.

Når brugeren ønsker at stoppe BoldBot, kan dette gøres ved at igen åbne en browser og gå til websiden. Denne gang trykkes der på knappen "Stop" og derefter termineres Robotbil programmet på RPi.

1.2 MoSCoW

Nedstående MoSCoW analyse anvendes til at definere de funktionelle krav til projektet. Analysen består af at opsætte en række funktionelle krav og inddæle disse i prioriteringer. Dette gøres ved at indele de funktionelle krav i kategorierne must, should, could og won't. Endvidere defineres de funktionelle krav i use cases, som yderligere definere systemets opførelse.

M – Must

- BoldBot skal kunne orientere sig på tennisbanen.
- BoldBot skal kunne opsamle bolde fra tennisbanen.
- BoldBot skal kunne aktiveres via en webside.

- BoldBot skal have funktionalitet der gør det muligt for robotten at køre fremad, bakke og dreje.
- BoldBot skal detektere boldene på banen.
- Boldbot skal have en batteriindikator, der viser batteriniveau for begge batterier.

S – Should

- BoldBot burde kunne finde tilbage til home base.

C - Could

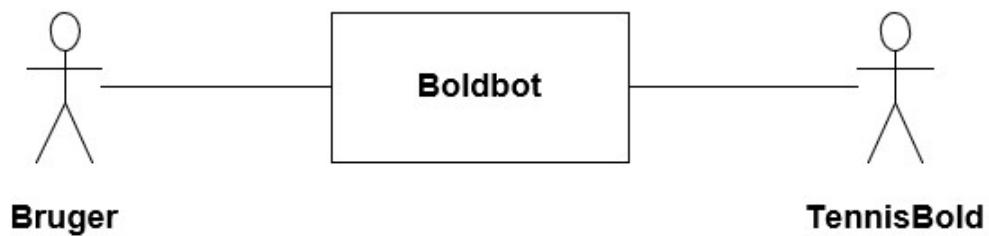
- BoldBot burde kunne detektere antallet af indsamlede bolde.
- BoldBot kunne have et aflæsningssted til aflevering af boldene.

W - Wont

- BoldBot vil ikke have et advarselssystem der informerer bruger hvis noget er galt.

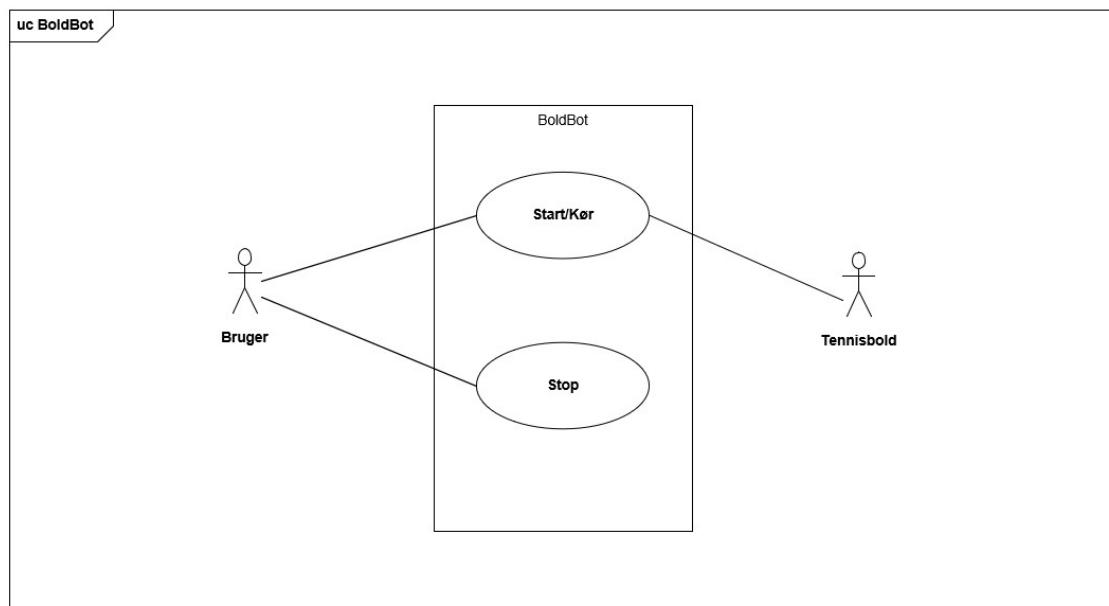
1.3 Funktionelle Krav

1.3.1 Aktør Beskrivelse



Figur 1.1: Aktør kontekst daigram for systemet

1.3.2 Use Case Diagram



Figur 1.2: Use Case diagram

1.3.3 Fully-dressed Use Cases

Use Case 1 - Start/Kør

Navn	Start/Kør
Mål	At BoldBot er startet og søger efter bolde
Initiering	BoldBot initieres på webside.
Aktører	Primær: Bruger Sekundær: Tennisbold
Antal samtidige forekomster	Ingen
Prækondition	BoldBot er inaktiv.
Postkondition	BoldBot har samlet bold op.
Hovedscenarie	<ol style="list-style-type: none"> 1. Brugeren starter BoldBot fra webside. 2. BoldBot modtager start signal fra webside. 3. BoldBot søger efter tennisbolde. 4. BoldBot registrerer en tennisbold. [Extension 1: BoldBot registrerer en forhindring] 5. BoldBot opsamler tennisbold. [Extension 2: BoldBot misser tennisbold]
Extensions	<p>[Extension 1: Forhindring]</p> <ol style="list-style-type: none"> 1. BoldBot møder en forhindring 2. BoldBot bakker tilbage. 3. Der fortsættes fra trin 3. <p>[Extension 2: Fejl i opsamling]</p> <ol style="list-style-type: none"> 1. BoldBot bakker væk fra bolden. 2. Der fortsættes fra trin 3.

Tabel 1.1: Use Case 1 - Start/Kør

Use Case 2 - Stop BoldBot

Navn	Stop BoldBot
Mål	At BoldBot stopper kørsel.
Initiering	BoldBot stoppes på websiden.
Aktører	Primær: Bruger
Antal samtidige forekomster	Ingen
Prækondition	BoldBot er startet og kører.
Postkondition	BoldBot er stoppet.
Hovedscenarie	<ol style="list-style-type: none"> 1. Bruger trykker på stop, på websiden. 2. BoldBot modtager stop signal fra webside. 3. BoldBot stopper.

Tabel 1.2: Use Case 2 - Stop

1.4 Ikke-Funktionelle Krav

1.5 FURPS

Nedstående FURPS analyse anvendes til at definere de ikke-funktionelle krav til projektet. Her bliver kvaliteterne og kriterierne udtrykt for systemet. Dette gøres via FURPS ved at inddæle de ikke-funktionelle krav i usability, reliability, performance og supportability, hvorfra F'et i FURPS fremgår af de funktionelle krav.

Functionality

- BoldBot må ikke overstige 60x60x60 cm.
- Batteriindikatoren skal kunne vise 5 forskellige batteriniveauer.

Usability

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

Reliability

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

Performance

- BoldBot skal kunne samle 1 bold op på mindst 5 min.
- BoldBot skal kunne opbevare mindst én bolde før den skal tømmes.
- BoldBot skal kunne registrere et ukendt objekt (alt andet end en tennisbold) 40 cm fra sensor.
- Boldbot skal kunne køre mindst 15 minutter på en fuld opladning.

Supportability

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

1.6 Udviklingsværktøjer

Nedstående afsnit indeholder de anvendte udviklingsværktøjer der er blevet anvendt til udvikling af produktet.

- Visual studio code text editor.
- Atom text editor.
- PSoC creater 4.2 .
- Multisim 14.0.
- Nano text editor.
- Visio.
- Trello.
- Google drive.

2. Analyse

2.1 Hardware

2.1.1 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 chassiset 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.

2.1.2 Batteriindikator

Til designet af batteriindikatoren er valgt en IC med fire komperatore 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 komperatore 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.

2.1.3 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 chassiset 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.

2.1.4 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 rimeligt til 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 sonarsensor kunne være lyssensor eller lasersensor. Det blev dog sonar da det umiddelbart var godt og ikke for besværligt at gå til.

2.1.5 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'.

2.2 Software

2.2.1 Protokol

Nedenstående afsnit kommer omkring de anvendte kommunikationsprotokoller, som er anvendt til at kommunikere mellem de forskellige enheder.

2.2.2 I2C

I2C protokollen bliver specifikt anvendt til kommunikationen med sonar sensor modulet. Anvendelsen af I2C protokollen fremfor andre kommunikationsprotokoller f.eks. SPI for sensor modulet er pga. af at sonar sensoren komminkerer over I2C, det er derfor ikke muligt at bruge andre kommunikationsformer, hvis man gerne vil bruge denne type sensor. Hvis man ville bruge andre kommunikationsformer skulle man derfor havde valgt en anden sonar sensor. Dette skaber grundlaget for valget af kommunikationsprotokollen I2C til at kommunikere med sensormodulet. I2C er også meget praktisk, da der kun bruges to ledninger til kommunikationen der også er to-vejs.

2.2.3 SPI

Der er valgt at anvende SPI, til kommunikationen mellem PSoC og RPi. Denne kommunikationsform er oplagt at bruge i projektet, eftersom den har en hurtig overførsel af data, samt der let kan implementeres et kernemodul så RPi bliver implementeret som SPI master. SPI kommunikation er forholdsvis let at implementere på RPi, eftersom der allerede er dedikeret specielle PINS til denne kommunikationsform. Fremfor SPI kommunikation mellem de to enheder, kunne der blive anvendt I2C eller UART som kommunikation. Disse metoder ville have været pålidelige og fornuftige. Men der er valgt SPI kommunikation fordi at hurtighed er blevet prioriteret, for at opnå optimal kommunikation mellem RPi og PSoC.

2.2.4 GPIO driver

Det blev belstuttet, at lave endnu et kernemodul (platform driver), som skulle læse fra en gpio på RPi. Kernemodulet blev først implementeret som et interrupt der skulle lave et interrupt på gpio17 når sonar sender højt signal hertil, men her kunne det hurtigt ses, at denne implementering ville gøre at koden hele tiden ville tjekke om gpio17 går højt, som dermed gør at koden aldrig ville komme videre. Dermed blev det besluttet at lave det uden et interrupt. Et alternativ til denne løsning, var at anvende den allerede implementerede kernemodul til SPI, og dertilhørende spiReceive(), som læser på kernemodulet hvad der bliver sendt fra slave (PSoC) til Master (RPi). Endnu et alternativ

kunne være at anvende en ny protokol som I2C, som også bliver anvendt mellem PSoC og sonarsensor.

2.2.5 USB

USB protokollen bliver anvendt til at etablere kommunikation mellem det anvendte kamera (Pixycam2) og RPizw. Anvendelsen af USB protokollen specifikt, som kommunikationsprotokol er grundet at den allerede alloverket software på Pixycam. Her er USB protokollen allerede en integreret del af Pixycam2, hvilket har gjort at valget af denne protokol oplagt til at kommunikere mellem Pixycam2 og RPizw. Dette udelukker dog ikke andre kommunikations protokoller, da Pixycam2 også understøtter SPI og I2C. Hvis USB ikke allerede havde været en integreret del af Pixycam2, ville det have været oplagt at anvende en anden protokol grundet USB protokollens kompleksitet, hvor I2C og SPI havde været mere idealt at anvende. (**USB protokollen**)

2.2.6 Websocket

Websocket protokollen er blevet anvendt til give full-duplex kommunikation over én TCP forbindelse mellem server og client på den implementeret website. Grundlaget for valg af websocket, som kommunikations protokol grunder i at der ikke før er blevet arbejdet med webdesign, hvor en kort introduktion er givet omkring websockets hvilket har gjort denne kommunikationsprotokol oplagt. Dog kunne andre kommunikations protokoller også have været anvendt til kommunikation over nettet såsom HTTP. Websocket tilbyder dog fortsættende forbindelse mellen client og server, hvor data kan blive sendt begge veje hele tiden, hvilket man ikke kan med HTTP. I forhold til det opnåede resultat i projektet, vil valget af enten half-duplex eller full-duplex i forhold til kommunikations protokol, ikke have haft nogen betydning. (**Websocket**)

2.2.7 Website

Til udviklingen af websiden er sproget JavaScript, samt HTML blevet anvendt til at opnå en funktionel website. Alternativt kunne der være blevet anvendt C++, Python, eller C. Disse sprog blev valgt fra da der efter nærmere undersøgelse, ikke var behov for at opnå den funktionalitet, som disse sprog udbyder fremfor JavaScript, som er et mere simpelt sprog. Der har ikke været noget undervisning i udviklingen af websider, så derfor har der skulle opnås ny viden indenfor den anvendte teknologi uanset hvilket sprog vi anvendte, og da JavaScript og HTML tilbød den mest simple tilgang til netop at løse vores krav til websiden, faldt valget på dette. Til kommunikations protokollen imellem RPi og websiden, er der brugt websocket protokollen. Alternativt kunne HTTP protokollen være blevet anvendt, men da websocket tilbyder en full-duplex kommunikation, der tillader

os at kommunikere med to-vejs kommunikation samtidigt, er valget af protokol faldet på websocket. Alternativt ville HTTP ikke kunne håndtere kommunikation imellem client serveren på RPi'en og websiden samtidigt, da HTTP kun tilbyder half-duplex. Til front-end delen af websiden er der anvendt HTML og CSS. Disse to udviklingssprog er anvendt da de er meget simple og nemme at lære. Dette var vigtigt, da der ingen viden om front-end udvikling er givet på kurset indtil videre. Da der ingen særlige krav var til front-end delen af websiden, uover at den skulle indeholde en start og en stop knap, var der heller ikke store krav til hvad de valgte udviklingssprog skulle kunne, i forholdt til funktionalitet.

2.2.8 Software på RPi

Til udvikling på RPi'en er det blevet valgt at der skrives i C++. Grunden til dette er at undervisningen understøtter dette. Et alternativt kunne have været C hvilket der også har været undervisning i, men C++ tilbyder mere funktionalitet og derfor er dette valgt. Der er andre alternativer f.eks. Python, som i tiden er meget populært, dette er dog ikke valgt da der ingen undervisning har været i dette, og derfor ville kræve mere arbejde at sætte sig ind i. C++ var det oplagte valg og derfor blev det programmeringssproget på RPi'en. Dog er der lavet kernemoduler hvilket bliver skrevet i C, da det er hvad der er blevet undervist i, i ISU.

2.2.9 Udviklingsværktøjer

Udviklingen af softwaren, udviklet i dette projekt, bygger på en arkitektur der er udarbejdet ved hjælp af UML, og derunder udvidelsen SysML. SysML er blevet brugt til udviklingen af størstedelen af arkitekturen, BDD'er IBD'er som beskriver systemets struktur og opbygning. Derudover bruges SysML også til design af sekvensdiagrammer og state-machines som beskriver systemets opførelse.

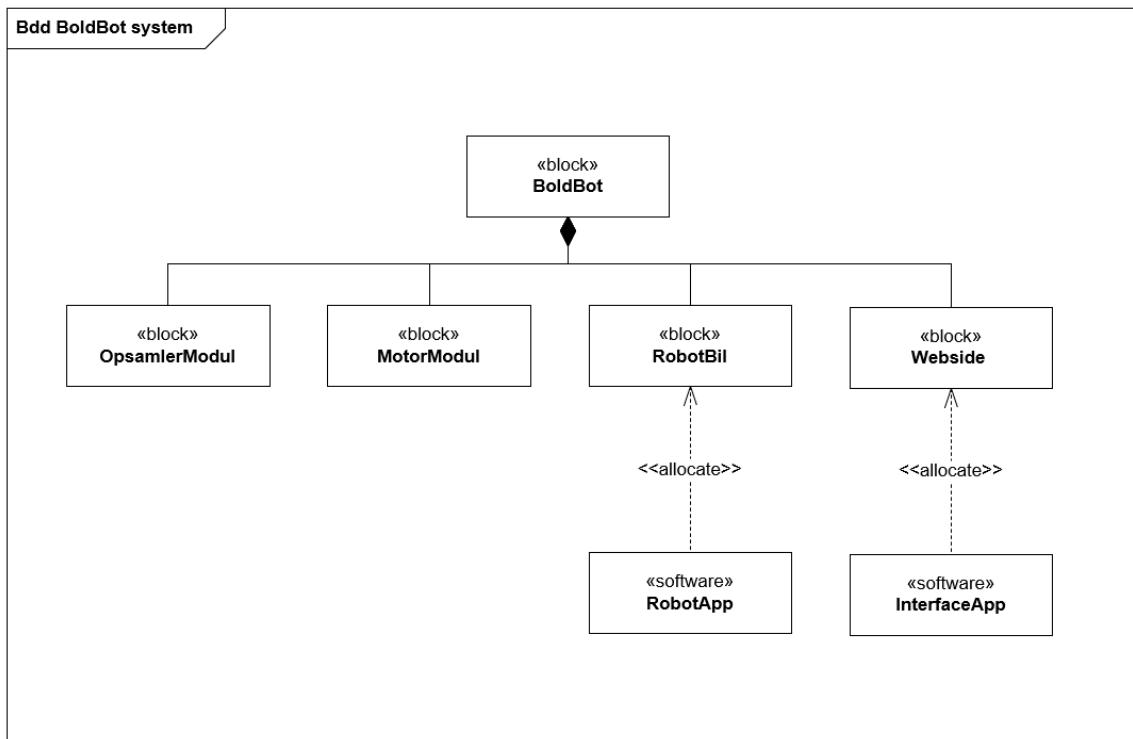
2.2.10 PSoC

Der er valgt at bruge PSoC Creator 4.2 til udviklingen af software til PSoC 5LP. Det er et Integrated Design Environment, som gør det muligt at programmere PSoC microcontrolleren. Der er valgt at bruge dette værktøj, delvist fordi det er det som er brugt i forbindelse med undervisning i faget GFV, og delvist fordi at dette er et værktøj lavet specifikt til at programmere på PSoC enheder. Derudover er PSoC Creator også ekstremt brugervenligt, det giver muligheder for at "drag and drop" mange forskellige komponenter til projekterne, som fx en SPI slave til kommunikation mellem RPi og PSoC. Det er en nødvendigt at bruge C til at programmere i PSoC creator, og hvilket også er grunden til at softwaren er skrevet i sproget C.

3. Arkitektur

3.1 Overordnet systemarkitektur

3.1.1 Hardware systemarkitektur



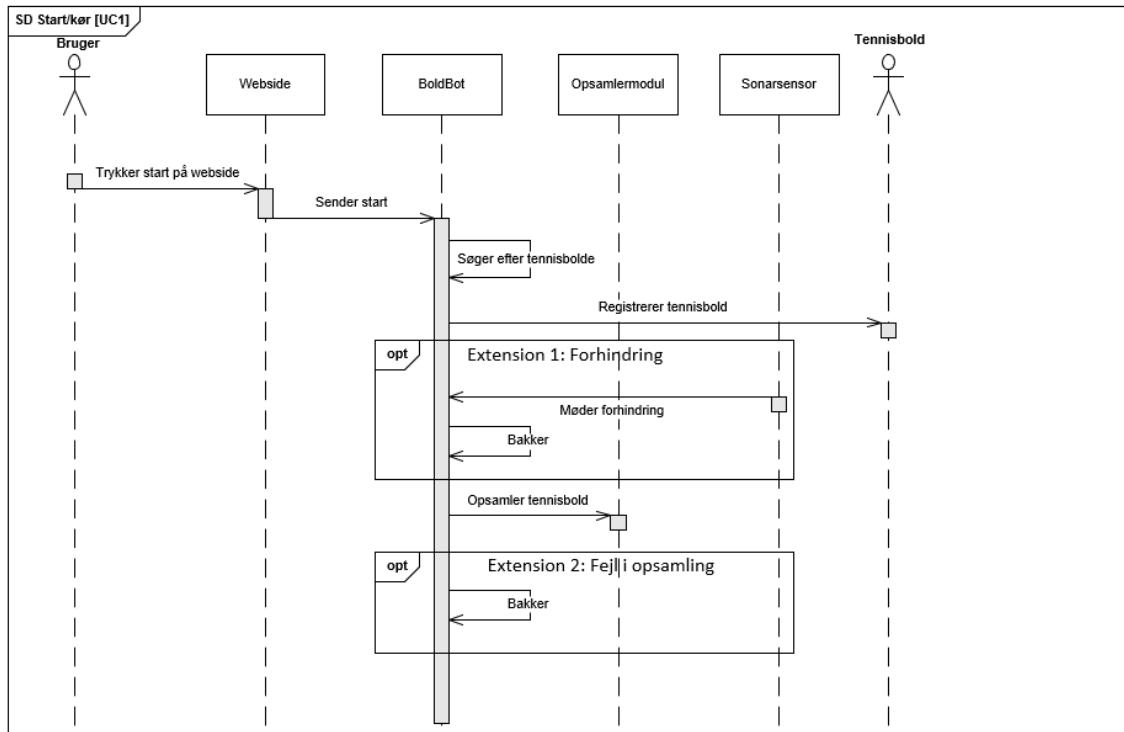
Figur 3.1: BDD af BoldBot

3.1.2 Software systemarkitektur

Denne sektion beskriver den overordnede systemarkitektur for dette projekts software. Dette indebærerer sekvensdiagrammer for begge Use Cases, domænemodel for systemet og et overordnet state machine diagram.

Sekvensdiagram - UC1

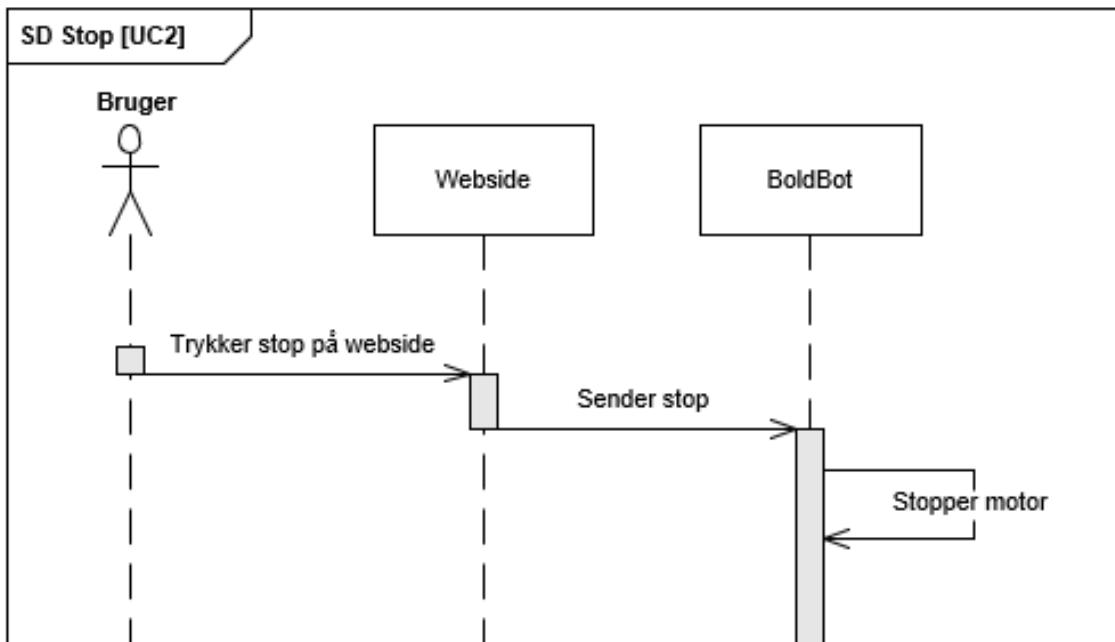
På 3.2 ses et overordnet sekvensdiagram for use case 1 (start/kør). Diagrammet bruges til at forklarer hvordan systemet overordnet set opfører sig.



Figur 3.2: Sekvensdiagram for Start/kør

Sekvensdiagram - UC2

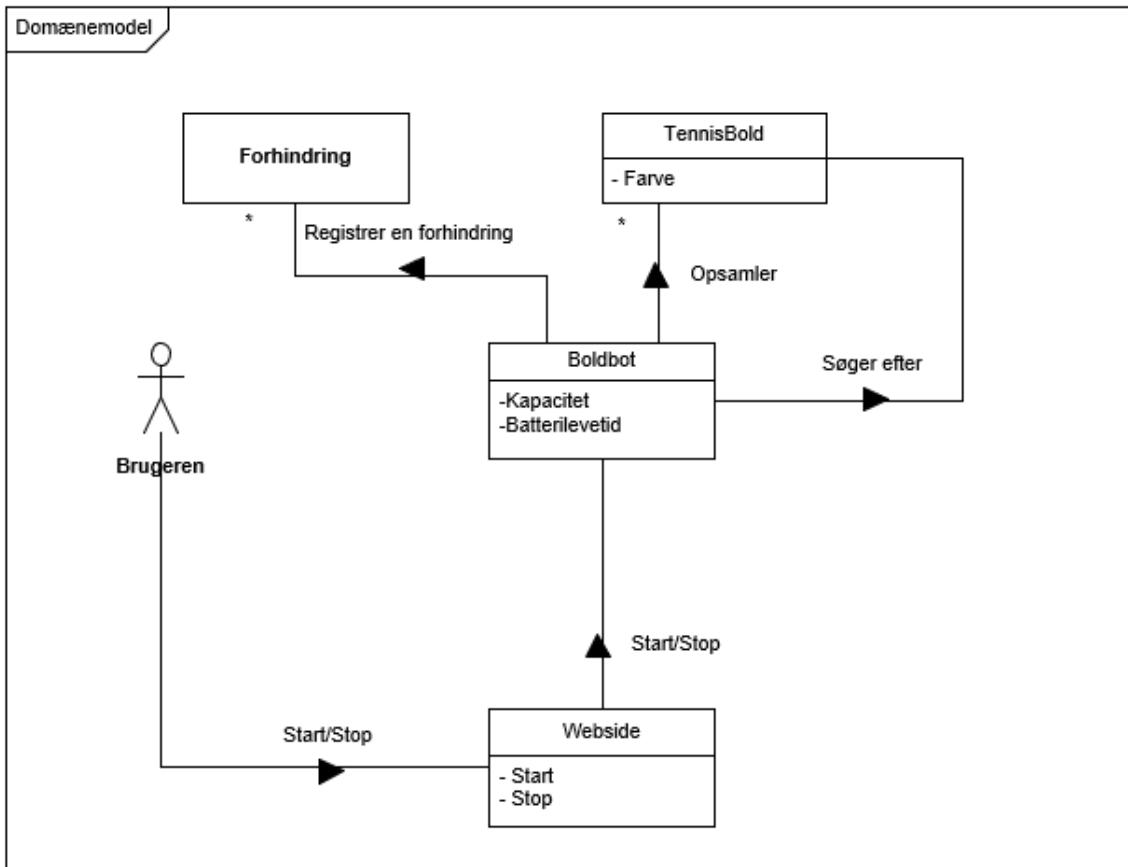
På 3.3 ses et overordnet sekvensdiagram for use case 2 (Stop). Diagrammet bruges til at forklarer hvordan systemet overordnet set opfører sig.



Figur 3.3: Sekvensdiagram for UC2

Domænemodel

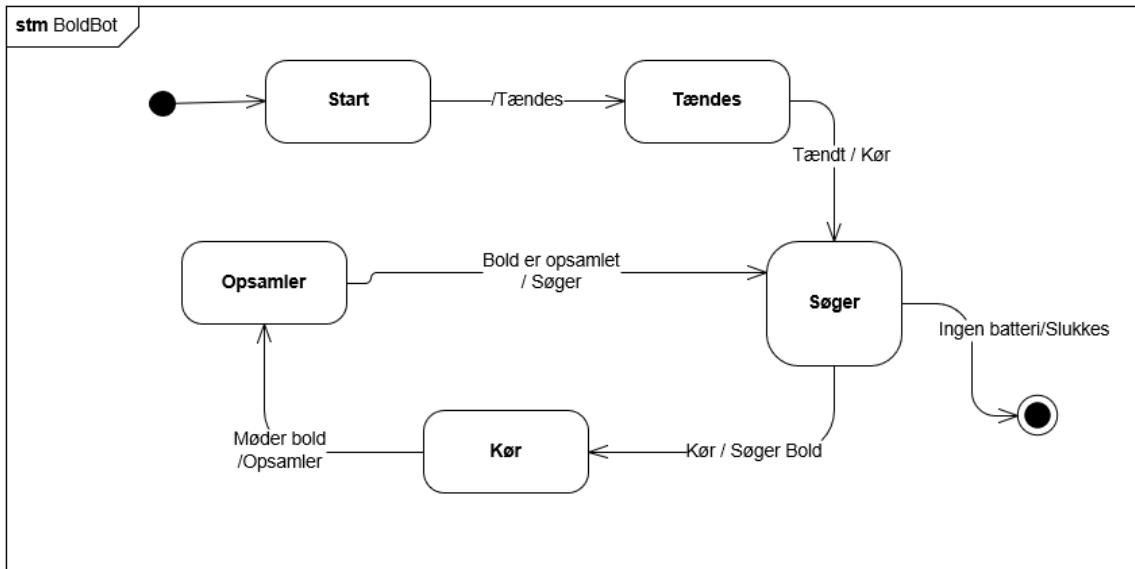
På 3.4 ses en overordnet domænemodel for BoldBot. Diagrammet forklarer use cases, deres opførelse og hvordan brugeren (primær aktør) interagerer med systemet.



Figur 3.4: Domænemodel for BoldBot.

State Machine Diagram

På 3.5 ses et overordnet state machine diagram for BoldBot. Dette diagram beskriver de forskellige states som BoldBot kan befinde sig i, hvilke events der skal til for at der skiftes imellem de forskellige states, samt hvilken aktion der sker for at den kommer i det givne state.



Figur 3.5: State Machine Diagram for BoldBot.

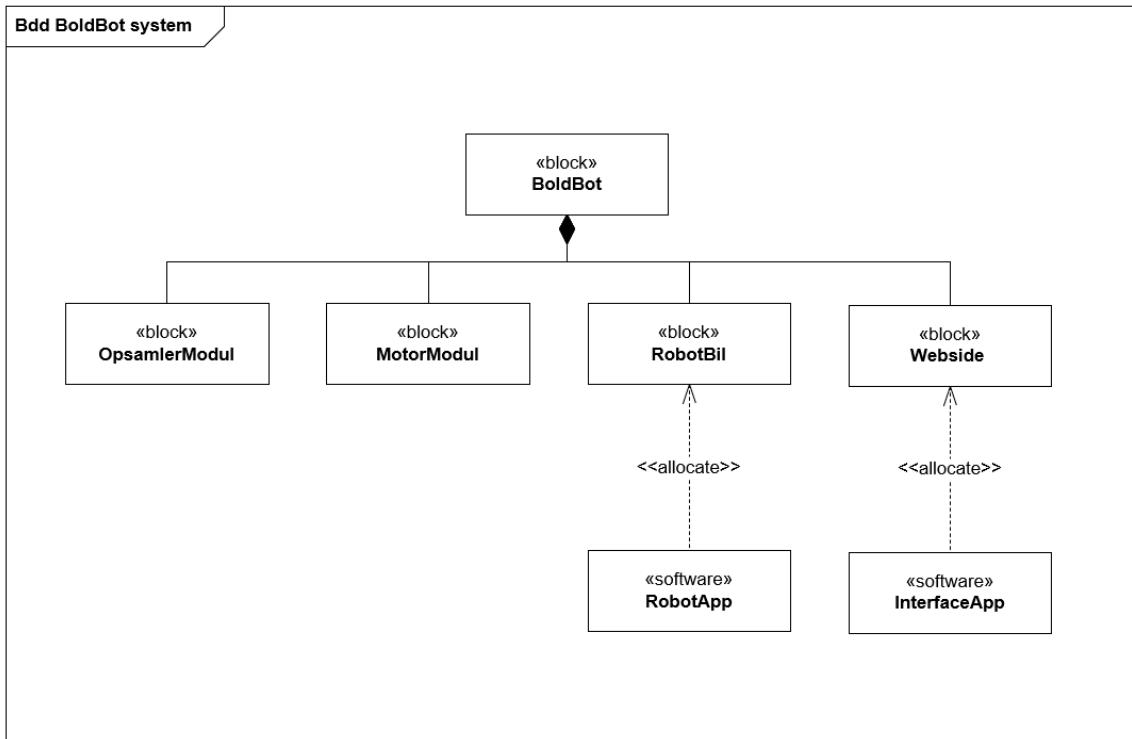
3.2 Hardware arkitektur

Hardware arkitekturen er opbygget af to typer diagrammer. Der er BDD'er, som er block definition diagrams, som beskriver de fysiske dele som systemet er opdelt i. Det andet er IDB'er, som er internal block diagrams, der beskriver de interne forbindelser imellem de hardwaremæssige dele.

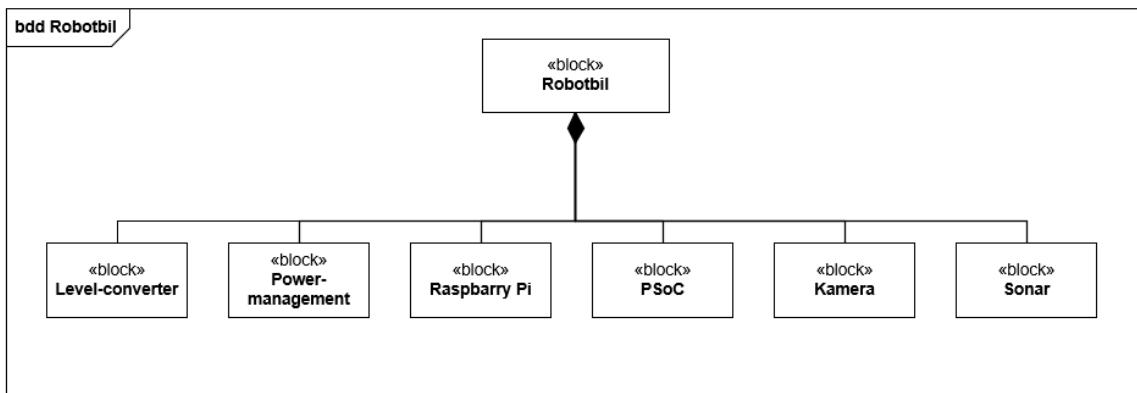
3.2.1 Block Definition Diagram

På Figur 3.6 ses et overordnet BDD af systemet. Systemet består af en Boldbot som har et opsamlingsmodul, robotbil og en GUI. Opsamlingsmodulet består af en servomotor som skal styre den robotarm som skal samle boldene op. RobotBil består af en PSoC til at styre sensorer og motorne, for at kunne styre boldbot to motorer skal der en motorstyring til som også kan ses i blokken RobotBil. Blokken GUI består af en Raspberry Pi og en Pc Lenovo for at få adgang til systemet.

Bloknavn	Funktion
OpsamlerModul	OpsamlerModulet her til formål at opsamle en tennisbold nå BoldBot har lokaliseret en
MotorModul	MotorModulet h
RobotBil	Blokken robotBil er her alt softwaren til bilen er lokaliseret, denne blok her til opgave at styre bilens funktioner.
Webside	Webside er herfra bilen bliver tændt og slukket

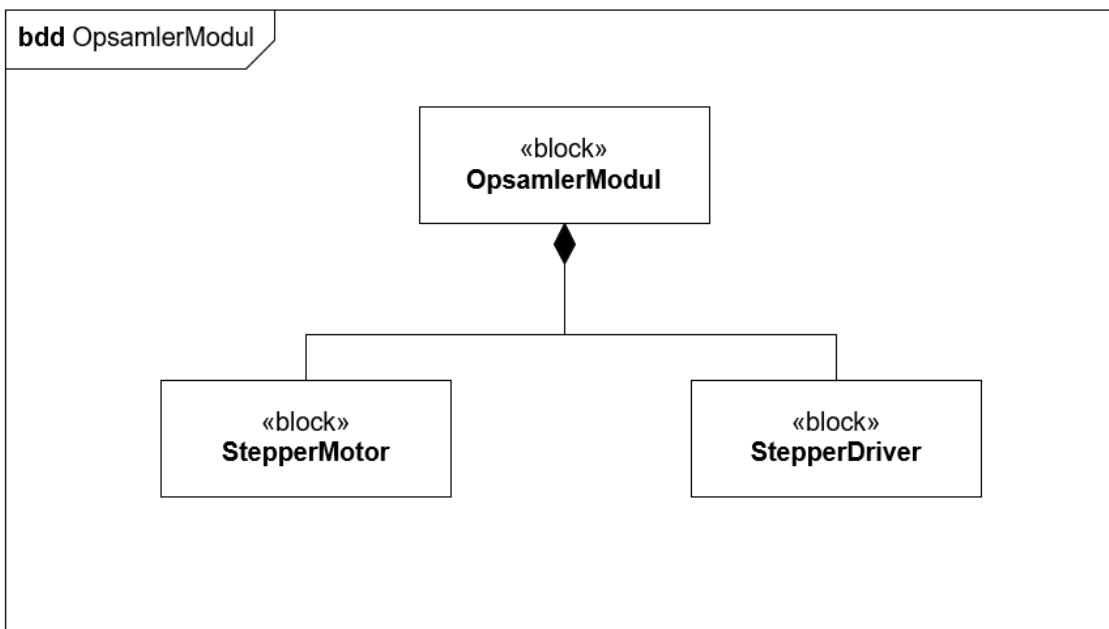


Figur 3.6: BDD af BoldBot



Figur 3.7: BDD af robotbil.

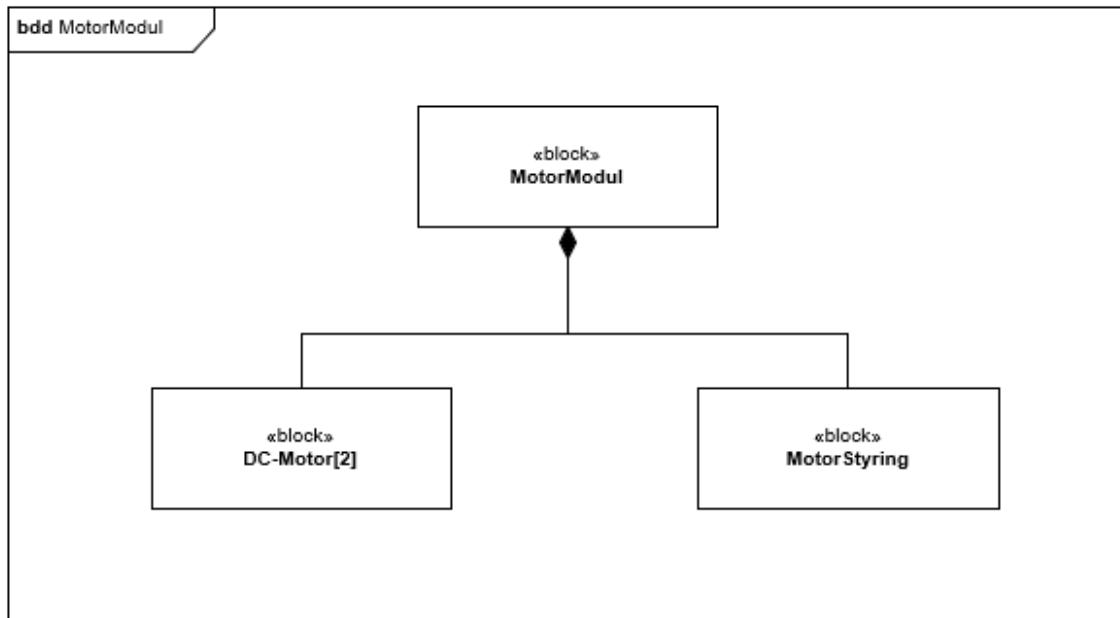
Bloknavn	Funktion
Level-converter	Level-converteren her til opgave at konvertere de to logiske niveauer fra PSoC og Raspberry Pi, så de kan kommunikere med hinanden
Power-management	Power-management blokken har til opgave at regulere batteri spærvåge batteri niveau,
Raspberry Pi	Raspberry Pien har til opgave at styre BoldBot'en ud fra de data den modtag fra Kamraet
PSoC	PSoCen her til opgave at styre opsamlermodul og motormodul
Kamera	Kameraet har til opgave at lokalisere en tennisbold og vise denne dens position
Sonar	Sonar her til opgave at stoppe BoldBot fra at køre ind i objekter



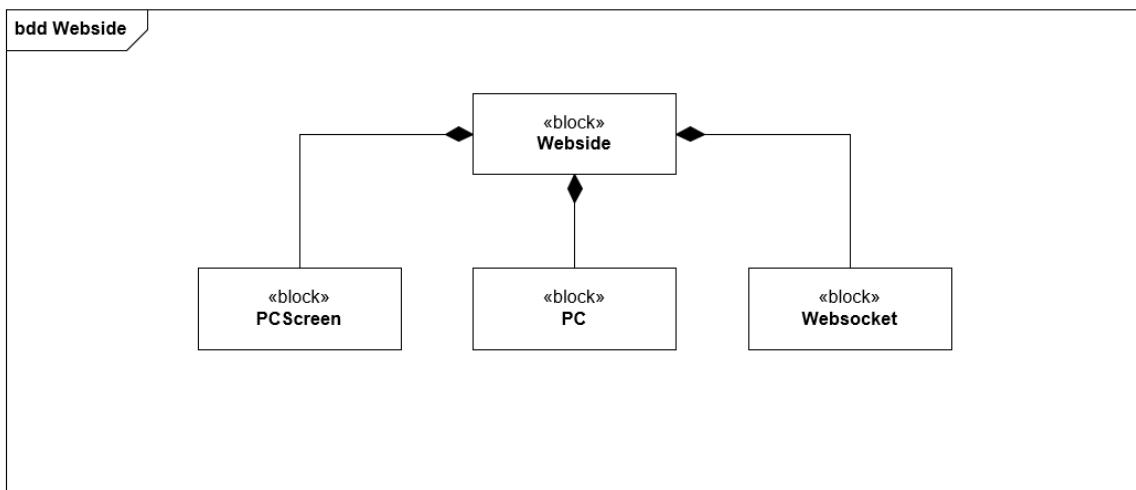
Figur 3.8: BDD af hele opsamler modul.

Bloknavn	Funktion
StepperMotor	
StepperDriver	

Bloknavn	Funktion
DC-Motor	
MotorStyring	

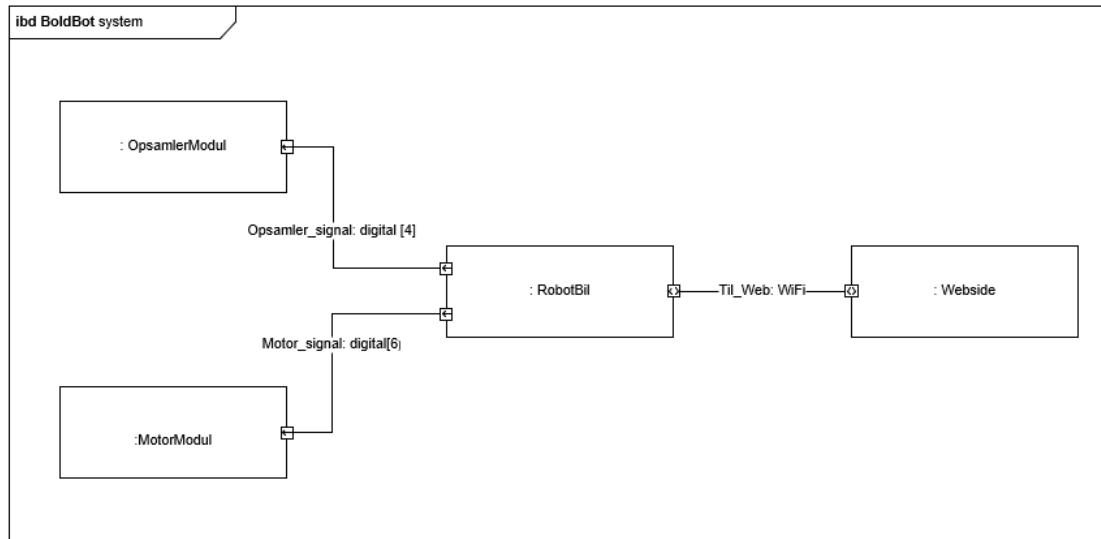


Figur 3.9: BDD af hele Motor modul.

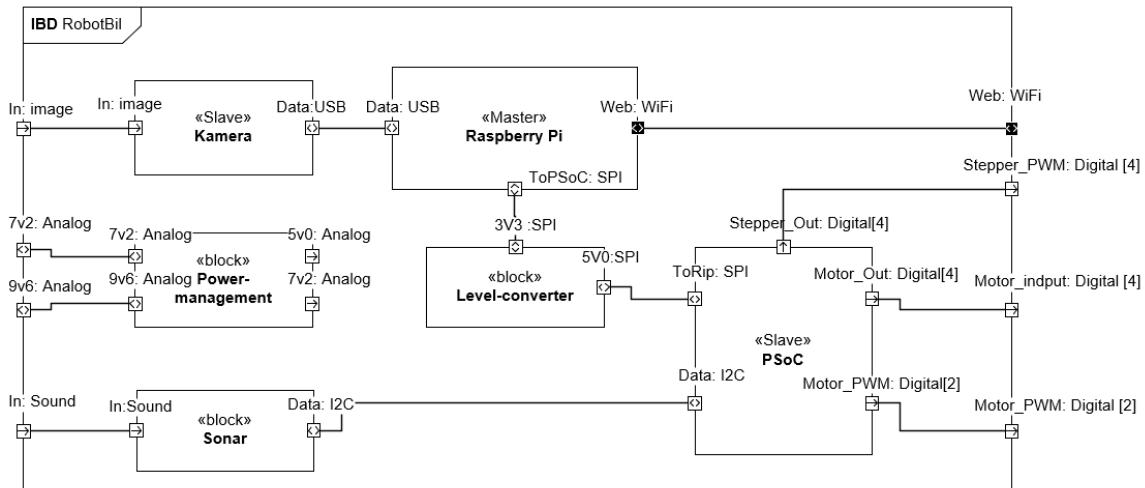


Figur 3.10: BDD af Webside.

3.2.2 Internal Block Diagram

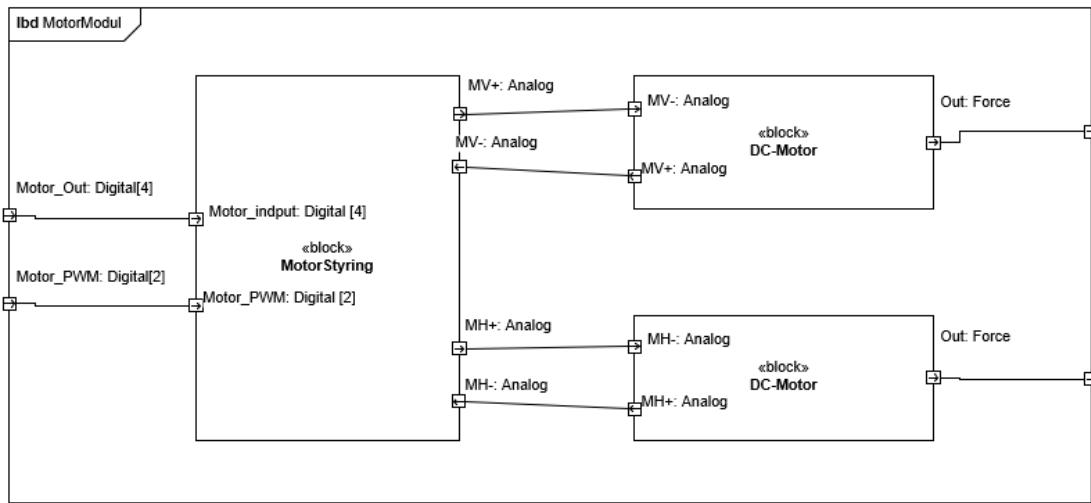


Figur 3.11: IBD af BoldBot.



Figur 3.12: IBD af robotbil

Bloknavn	Port funktion	Portnavn	Type	Port specifikationer
Kamera	USB signal til Raspberry Pien, med data om ting på billede	Data	Serial	USB
Sonar	I2C signal mellem Sonar og PSoC	Data	Serial	I2C TTL Logic, lavt < 1.5 V og højt > 3.5 V
Raspberry Pi	USB data signal fra kameraet	Data	Serial	USB
	Serial SPI signal til PSoC. Signalet bliver converteret af Level converterne	ToPSoC	Serial	SPI, 3.3 V logic, lav < 0.8 V og højt < 2.4 V, max 50 mA
	WiFi forbindelse mellem Raspberry og internettet	Web	Serial	WiFi
	5 V Forsyning til fra Power-management	5V0	DC	5.0 V +- 0.5, max 1 mA
PSoC	Serial I2C Signal mellem PSoC og Sonar	Data	Serial	I2C, TTL Logic, lavt < 1.5 V og højt > 3.5 V
	Serial SPI Signaler mellem PSoC Og Raspberry pi. Signalet bliver converteret af Level converterne.	ToRip	Serial	SPI, TTL Logic, lavt < 1.5 V og højt > 3.5 V, max 28 mA
	4 PWM signaler fra PSoC til StepperStyring.	Stepper_Out[4]	Digital	5 V PWM signal, max 40 kHz, and max 41 mA
	4 Digitale signal fra PSoC til Motorstyring, til at syre motor rating.	Motor_Out[4]	Digital	5 V trigger, input Lav mellem -0.3 V og 1.5 V og Høj mellem 2.3 V og 5.0 V, Max 41 mA
	2 PWM signaler til motorsyring, der bruges til at styre hastigheden	Motor_PWM[2]	Digital	5 V PWM signal, max 40 kHz, and max 41 mA
	Forsyning til PSoC	VDD	DC	5.0 V +- 0.5 V, max 500 mA
Level-converter	SPI til Raspberry Pi	3V3	Serial	SPI, 3.3 V logic, lav < 0.8 V og højt < 2.4 V, max 50 mA
	SPI til PSoC	5V0	Serial	SPI, TTL Logic, lavt < 1.5 V og højt > 3.5 V max 50 mA
Power-management 29.05.2019	Input fra 7.2 V Batteri	7v2	DC	7.2 V +-2.2 V DC
	Constant 5 V spændings output til forsyning af andre blok-	5v0	Analog	5.0 V +- 0.5 V analog output, max 1500 mA

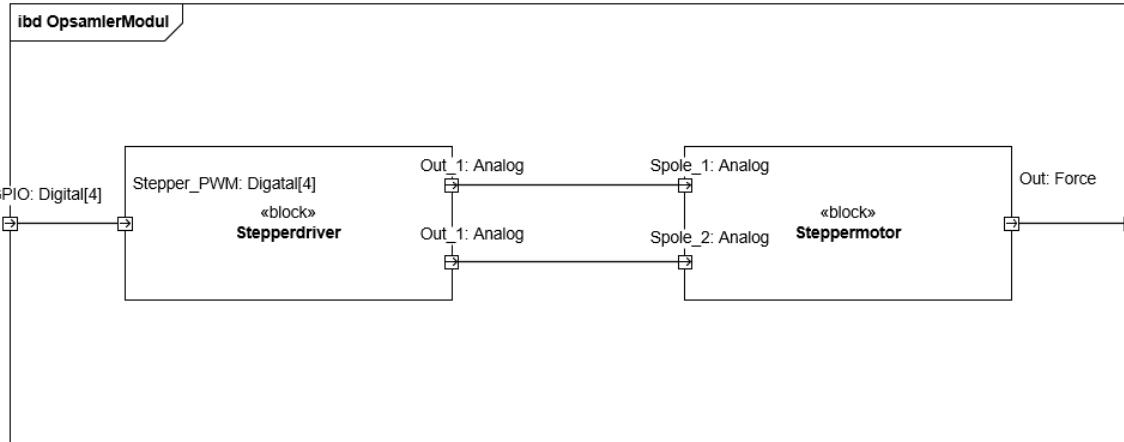


Figur 3.13: IBD af Motor modul.

MotorModul

Bloknavn	Funktion	Portnavn	Type	Port specifikationer
Motor-Styring	2 PWM input signaler fra PSoC	Motor_PWM[2]	Digital	5 V PWM signal, max 40 kHz, and max 100 mA
	4 Digitale input til styre retningen på hvert motor	Motor_input[4]	Digital	5 V trigger, input Lav < 1.5 V og Høj > 2.3 V max 5.0 V, max 100 mA
	Forsyning til H-Broen	H-bro_Forsyning	DC	5.0 V +- 0.5 V, 100 mA
	Forsyning til motorenne fra 9.6 V batteriet	Motor_Forsyning	DC	9,6 V +- 2.2 V DC, Max 2500 mA over 10 ms
	Forbindelse mellem h-broen og + på venstre motor	MV+	Analog	9,6 V +- 2.2 V DC, Max 2500 mA over 10 ms
	Forbindelse mellem h-broen og - på venstre motor	MV-	Analog	
	Forbindelse mellem h-broen og + på højre motor	MH+	Analog	9,6 V +- 2.2 V DC, Max 2500 mA over 10 ms
	Forbindelse mellem h-broen og - på højre motor	MH-	Analog	
DC-Motor	Forbindelse mellem h-broen og + på venstre motor	MV+	Analog	9,6 V +- 2.2 V DC, Max 2500 mA over 10 ms
	Forbindelse mellem h-broen og - på venstre motor	MV-	Analog	
DC-Moter	Forbindelse mellem h-broen og + på højre motor	MH+	Analog	9,6 V +- 2.2 V DC, Max 2500 mA over 10 ms
	Forbindelse mellem h-broen og - på højre motor	MH-	Analog	

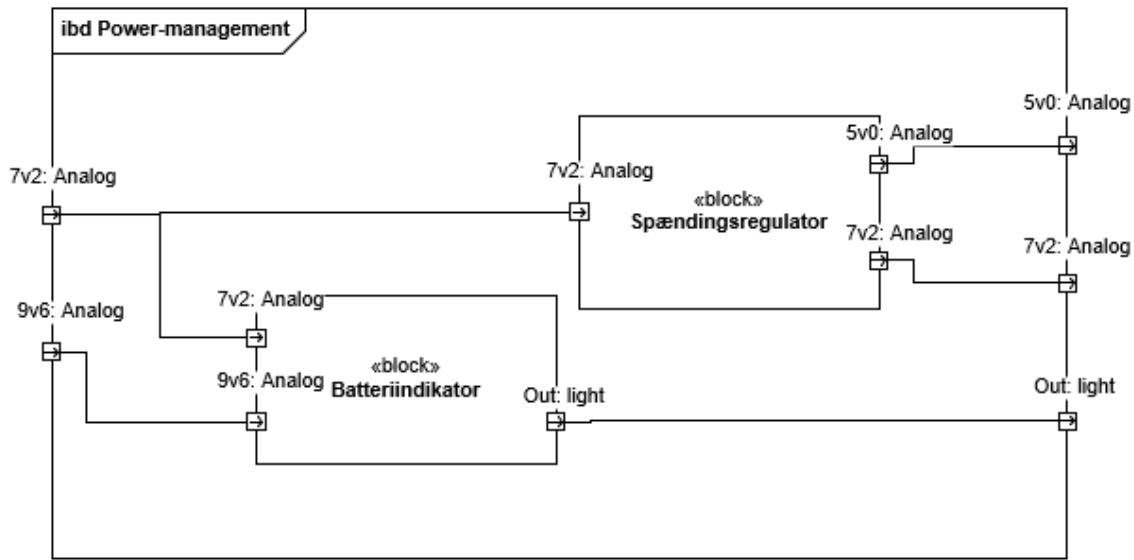
OpsamlerModul



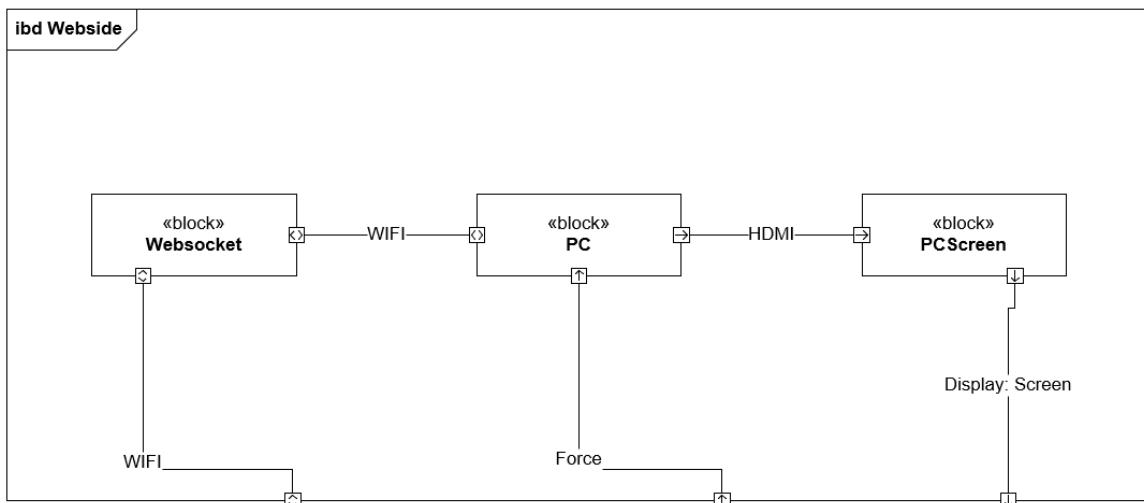
Figur 3.14: IBD af opsamler modul.

Bloknavn	Funktion	Portnavn	Type	Port specifikationer
Stepper-Styring	4 PWM indputs fra PSoC, til at styre stepperne	Stepper_PWM[4]	Digital	5 V PWM signal, max 40 kHz, and max 25 mA
	Output til stepper-mortorens ene spole	Out_1	Analog	7.2V +-2.2 V, max 2000 mA
	Output til stepper-mortorens anden spole	Out_2	Analog	7.2V +-2.2 V, max 2000 mA,
	Forsyning til Stepper motoren fra 7.2 V batteri	Forsyning	DC	7.2 V +-2.2 V DC, Max 2000 mA
Stepper-Motor	Den ene spole på steppermotoren	Spole_1	Analog	7.2V +-2.2 V, max 2000 mA
	Den anden spole på steppermotoren	Spole_2	Analog	7.2V +-2.2 V, max 2000 mA

Bloknavn	Funktion	Portnavn	Type	Port specifikationer
Spændingsregulator	Indgang fra 7,2 V batteri	7v2	Analog	7.2V +-2.2 V, max 2.5
	7.2 V Udgang fra spændingsregulatoren til Stepperdriver	7v2	Analog	7.2 V +-2.2 V DC, Max 2000 mA
	5 V udgange til forsyning af forskellig komponinter i systemet	5v0	Analog	5.0 V +- 0.5 V analog output, max 1500 mA
Batteri-indikator	Indgangen fra 7,2 v batteriet	7v2	Analog	7.2V +-2.2 V,
	Indgangen fra 9,6 v batteriet	9v6	Analog	9.6 V +-2.2 V,



Figur 3.15: IBD af Power-management.

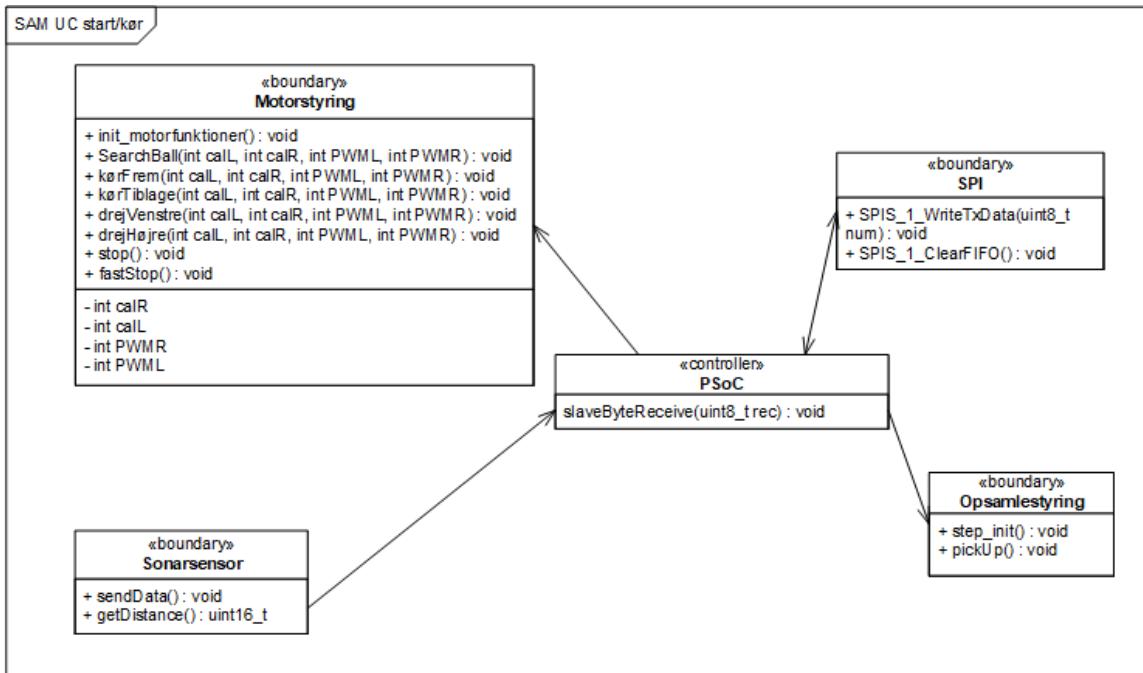


Figur 3.16: IBD af webside.

3.3 Software arkitektur

3.3.1 Klassediagram PSoC

Figur 3.17 viser et klassediagram af PSoC lavet ud fra UC1-Start/kør. Klassediagrammet består af fire klasser samt en main (controller).



Figur 3.17: Klassediagram for PSoC kode

Metode forklarelse - PSOC(main)

`void slaveByteRecieved(rec)`

Parameter: rec: Læser hvad der bliver modtaget fra SPI.

Returværdi: Ingen.

Beskrivelse: Står i en Switch case. .

Metode forklarelse - Motorstyring

`void init_motorFunktioner()`

Parameter: Ingen.

Returværdi: Ingen.

Beskrivelse: Initierer motorstyring.

`void fastStop()`

Parameter: Ingen.

Returværdi: Ingen.

Beskrivelse: Stopper begge motorer.

`void stop()`

Parameter: Ingen.

Returværdi: Ingen.

Beskrivelse: Stopper begge motorer.

```
void searchBall(int callL, int calR, int PWML, int PWMR)
```

Parameter: callL: Kalibrerer venstre motor.
 calR: Kalibrerer højre motor.
 PWML: Sætter duty cycle på venstre motor.
 PWMR: Sætter duty cycle på højre motor.

Returværdi: Ingen.
Beskrivelse: BoldBot roterer rundt om sin egen akse..

```
void korFrem(int callL, int calR, int PWML, int PWMR)
```

Parameter: callL: Kalibrerer venstre motor.
 calR: Kalibrerer højre motor.
 PWML: Sætter duty cycle på venstre motor.
 PWMR: Sætter duty cycle på højre motor.

Returværdi: Ingen.
Beskrivelse: BoldBot kører fremad.

```
void korTilbage(int callL, int calR, int PWML, int PWMR)
```

Parameter: callL: Kalibrerer venstre motor.
 calR: Kalibrerer højre motor.
 PWML: Sætter duty cycle på venstre motor.
 PWMR: Sætter duty cycle på højre motor.

Returværdi: Ingen.
Beskrivelse: BoldBot bakker.

```
void drejHøjre(int callL, int calR, int PWML, int PWMR)
```

Parameter: callL: Kalibrerer venstre motor.
 calR: Kalibrerer højre motor.
 PWML: Sætter duty cycle på venstre motor.
 PWMR: Sætter duty cycle på højre motor.

Returværdi: Ingen.
Beskrivelse: BoldBot drejer til højre.

```
void drejVenstre(int callL, int calR, int PWML, int PWMR)
```

Parameter: callL: Kalibrerer venstre motor.
 calR: Kalibrerer højre motor.
 PWML: Sætter duty cycle på venstre motor.
 PWMR: Sætter duty cycle på højre motor.

Returværdi: Ingen.
Beskrivelse: BoldBot drejer til venstre.

Metode forklarelse - opsamlerStyring

```
void stepper_init()
```

Parameter: Ingen.
Returværdi: Ingen.
Beskrivelse: Initierer steppermotor til styring af opsamlermodul.

```
void stepper_up(uint8_t steps)
```

Parameter: steps: Bestemmer hvor mange steps steppermotor skal tage.

Returværdi: Ingen.

Beskrivelse: Hæver opsamlermodulet fra jorden op i udgangspunkt.

```
void stepper_down(uint8_t steps)
```

Parameter: steps: Bestemmer hvor mange steps steppermotor skal tage.

Returværdi: Ingen.

Beskrivelse: Sænker opsamlermodulet fra udgangspunktet ned til jorden.

```
void stepper_top()
```

Parameter: Ingen.

Returværdi: Ingen.

Beskrivelse: Stopper steppermotoren.

Metode forklarelse - sonarsensor

```
uint16_t getDistance()
```

Parameter: Ingen.

Returværdi: uint16_t: Retunerer distancen til et registreret objekt.

Beskrivelse: Registrerer objekter og returnerer distancen til dem.

```
void sendData()
```

Parameter: Ingen.

Returværdi: Ingen.

Beskrivelse: Sætter en gpi0 højt på RPi hvis distancen til et objekt fra sonarsensor er under 40cm.

Metode forklarelse - SPI

```
void SPIS_1_WriteTxData(uint8_t)
```

Parameter: uint8_t: Modtager 8-bit data fra Master.

Returværdi: Ingen.

Beskrivelse: Modtager data fra Master over SPI.

```
void SPIS_1_ClearFIFI()
```

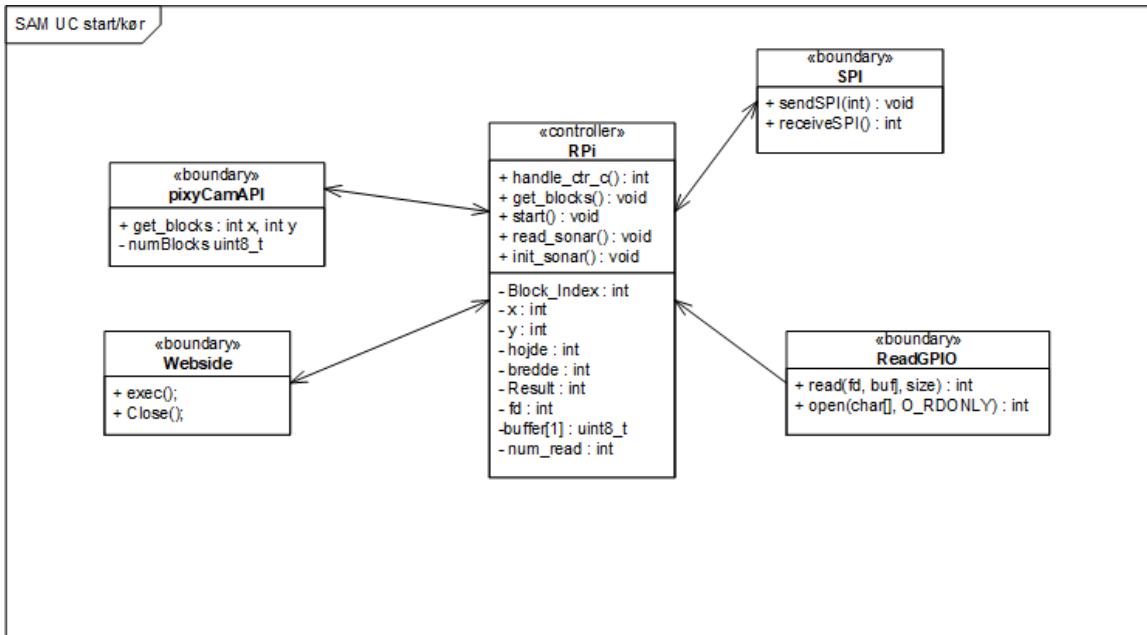
Parameter: Ingen.

Returværdi: Ingen.

Beskrivelse: Rydder køen efter data er modtaget.

3.3.2 RPi klassediagram

På 3.18 ses klassediagrammet for RPi lavet ud fra Sue case 1, Start/Kør. Diagrammet består af 4 klasser og en controller. Forklaringen af metoderne som ses på diagrammet kan findes i sektionerne 3.3.2 på side 30 ', 3.3.2 på side 30 ' samt 3.3.2 på side 31 '.



Figur 3.18: SAM UC start/kør

Metodeforklarelse - RPi

`int handle_ctrl_c()`

Parameter: Ingen.

Returværdi: Ingen.

Beskrivelse: Håndtere afslutning af program via ctrl_c.

`void start()`

Parameter: Ingen.

Returværdi: Ingen.

Beskrivelse: Håndtere afslutning af program via ctrl_c.

`void get_blocks(int sig)`

Parameter: Sig: sættes til 1 og bruges som parameter til funktionskald på pixycam.

Returværdi: Ingen.

Beskrivelse: Håndtere den sendte information fra pixycam, hvorefter der reageres på inputtet fra pixycam og den tilsvarende funktion sendes over SPI kommunikationen.

`void init_sonar()`

Parameter: Ingen.

Returværdi: Ingen.

Beskrivelse: Initiere sonarsensoren.

SPI

```
void start()
```

Parameter: Ingen.
Returværdi: Ingen.
Beskrivelse: Håndtere data fra sonarsensoren.

```
int receiveSPI()
```

Parameter: Ingen.
Returværdi: Ingen.
Beskrivelse: Modtager data over SPI kommunikationen.

```
void sendSPI(int sendbyte)
```

Parameter: sendbyte: den integer som sendes over SPI kommunikationen
Returværdi: Ingen.
Beskrivelse: Sender data via SPI kommunikationen.

ReadGPIO

```
int read(int fd, char[] buf,int size)
```

Parameter: fd: file desctiptor.
buf: buffer for resultater
size: read size
Returværdi: int.
Beskrivelse: Læser fra GPIOdriveren.

```
int open(char[] filename, O_RDONLY)
```

Parameter: filename: fil navnet på stien.
O_RDONLY: tilladelse til kun at læse
Returværdi: int.
Beskrivelse: Retunere file descriptoren til filen.

Webseite

```
var exec = require('child_process').execFile
```

Parameter: child_process
Returværdi: Ingen.
Beskrivelse: Eksekvere output filen til BoldBot programmet.

```
Ps.kill(process.pid)
```

Parameter: process.pid
Returværdi: Ingen.
Beskrivelse: Terminere processen med et defineret PID.

PixyCam

int8_t getBlocks(bool wait[optional], uint8_t sigmap[optional], uint8_t maxBlocks[optional])

Parameter: Ingen.

Returværdi: int8_t: returnere enten error værdien < 0, eller succes værdien >= 0

Beskrivelse: Gemmer de detekterede blocks fra den sidst nye frame i member variablen blocks.

int8_t init(uint32_t arg[optional]

Parameter: Ingen.

Returværdi: int8_t: returnere enten error værdien < 0, eller succes værdien 0

Beskrivelse: Tjekker om kommunikationen med modulet pixycam er muligt.

int8_t getVersion()

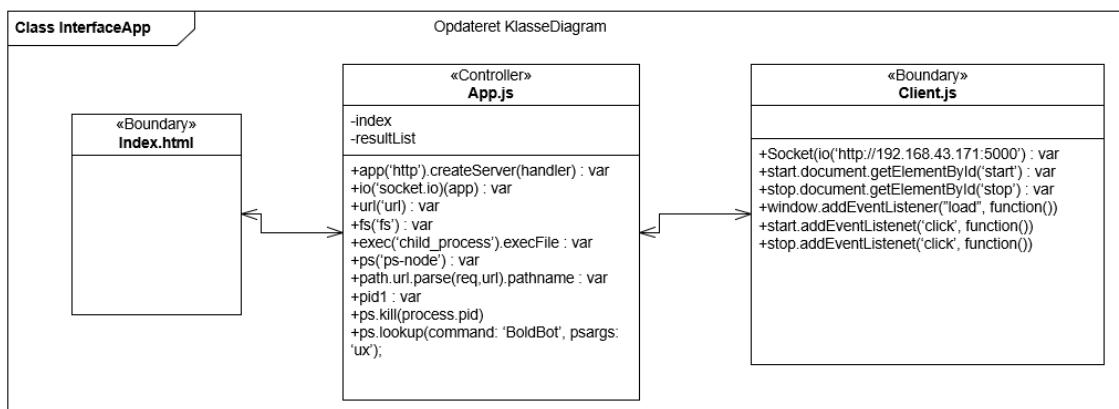
Parameter: Ingen.

Returværdi: int8_t: returnere succes værdien 0, ellers returneres error værdien < 0

Beskrivelse: Gemmer firmware og hardware versionen for pixycam i member variablen version.

3.3.3 Klassediagram Webside

Figur 3.19 viser et klassediagram for webserver samt webclient lavet ud fra UC1 Start/Kør samt UC2 stop. I sektionerne 3.3.3 på side 32 ” og 3.3.3 på side 33 ”.



Figur 3.19: Klassediagram for webside

Metode forklarelse - App.js

Var app = require('http').createServer(handler)

Parameter: 'http' indeholder http serveren.

Returværdi: Ingen.

Beskrivelse: Håndtere http serveren, og opretter serveren med funktionen handler.

Var io = require('socket.io')(app)

Parameter: 'socket.io' og app

Returværdi: Ingen.

Beskrivelse: Anmoder om brug af socket.io modulet og sender app objektet til modulet.

```
Var url = require('url')
```

Parameter: 'url' indeholder url for webside

Returværdi: Ingen.

Beskrivelse: Bruges som hjælpe funktion til at håndtere det oprettede url.

```
Var url = require('url')
```

Parameter: 'url' indeholder url for webside

Returværdi: Ingen.

Beskrivelse: Bruges som hjælpe funktion til at håndtere det oprettede url.

```
Var fs = require('fs')
```

Parameter: 'fs' eller filesystem, indeholder stien, til de filer der anvendes

Returværdi: Ingen.

Beskrivelse: Hjælpe funktion til at holde styr på fil stien, til de anvendte filer som index.html og client.js

```
Var exec = require('child_process').execFile
```

Parameter: 'child_process' er den anden process der skal køre uddover selve websiden.

Returværdi: Ingen.

Beskrivelse: Eksekvere boldbot programmet, som child_process, til selve websiden, således at boldbot programmet kan styres fra websiden mht. start og stop.

```
Var ps = require('ps-node')
```

Parameter: 'ps-node' er et modul der anvendes på websiden.

Returværdi: Ingen.

Beskrivelse: Loader ps-node modulet således det kan anvendes med websiden.

```
Var path = url.parse(req.url).pathname
```

Parameter: Ingen.

Returværdi: Ingen.

Beskrivelse: Path analyserer url anmodningen, og hjælper med at finde stien til index.html og client.js

```
Var pid1 = process.pid
```

Parameter: Ingen.

Returværdi: Ingen.

Beskrivelse: Finder og returnere PID for child_process(boldbot).

```
Ps.kill(process.pid);
```

Parameter: process.pid, indeholder PID for det program der skal stoppes.

Returværdi: Ingen.

Beskrivelse: Terminere programmet der har process.pid. I vores tilfælde vil det være boldbot programmet.

```
Ps.lookup(command: 'BoldBot', psargs: 'ux')
```

Parameter: Navnet på det program der skal findes ID for.

Returværdi: Ingen.

Beskrivelse: Finder PID for programmet defineret i parameter.

Metode forklarelse - Client.js

```
Var socket = io('http://192.168.43.171:5000')
```

Parameter: Parameteren, indeholder den IP og port man skal bruge for at oprette forbindelse til websiden.

Returværdi: Ingen.

Beskrivelse: Opretter socket objektet der bruges til at sende og modtage data fra serveren.

```
Var start = document.getElementById('start')
```

Parameter: 'Start'

Returværdi: Ingen.

Beskrivelse: Start indeholder funktionen getElementById, der opretter forbindelse med elementet 'start' som er startknappen på websiden.

```
Var stop = document.getElementById('stop')
```

Parameter: 'Stop'

Returværdi: Ingen.

Beskrivelse: Stop indeholder funktionen getElementById, der opretter forbindelse med elementet 'stop' som er stopknappen på websiden.

```
Window.addEventListener("load", function()
```

Parameter: "load", function()

Returværdi: Ingen.

Beskrivelse: Funktionen "lytter" på websidens interaktive elementer når den loades.

```
Start.addEventListener('click', function())
```

Parameter: 'Click'

Returværdi: Ingen.

Beskrivelse: Funktionen tillader clienten at lytte på start knappen i index.html. Når der bliver klikket på start knappen, vil clienten så sende information til app.js om at programmet boldbot skal starte.

```
Stop.addEventListener('click', function())
```

Parameter: 'Click'

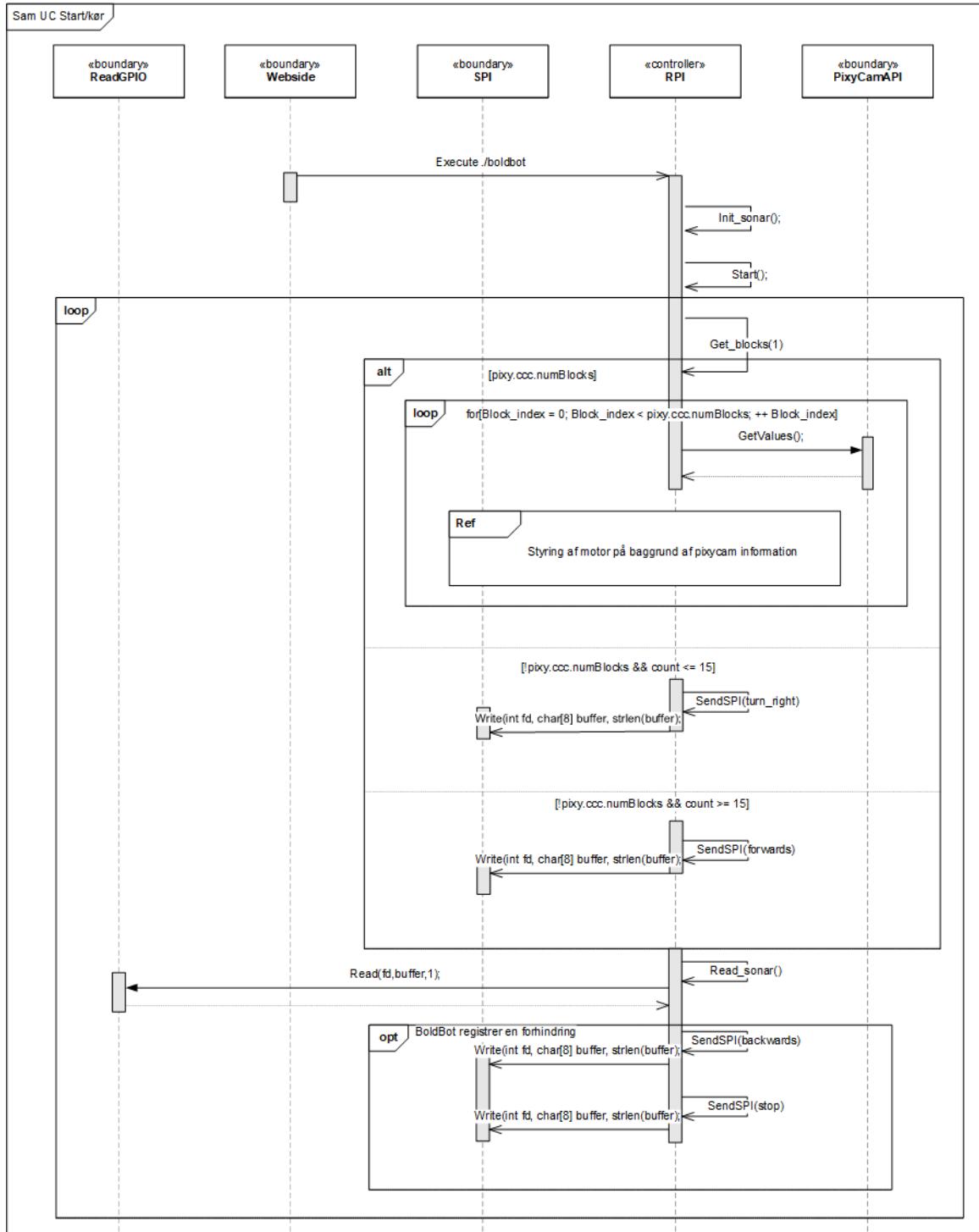
Returværdi: Ingen.

Beskrivelse: Funktionen tillader clienten at lytte på stop knappen i index.html. Når er bliver klikket på stop knappen, vil clienten så sende information til app.js om at programmet boldbot skal termineres.

3.3.4 Sekvensdiagrammer

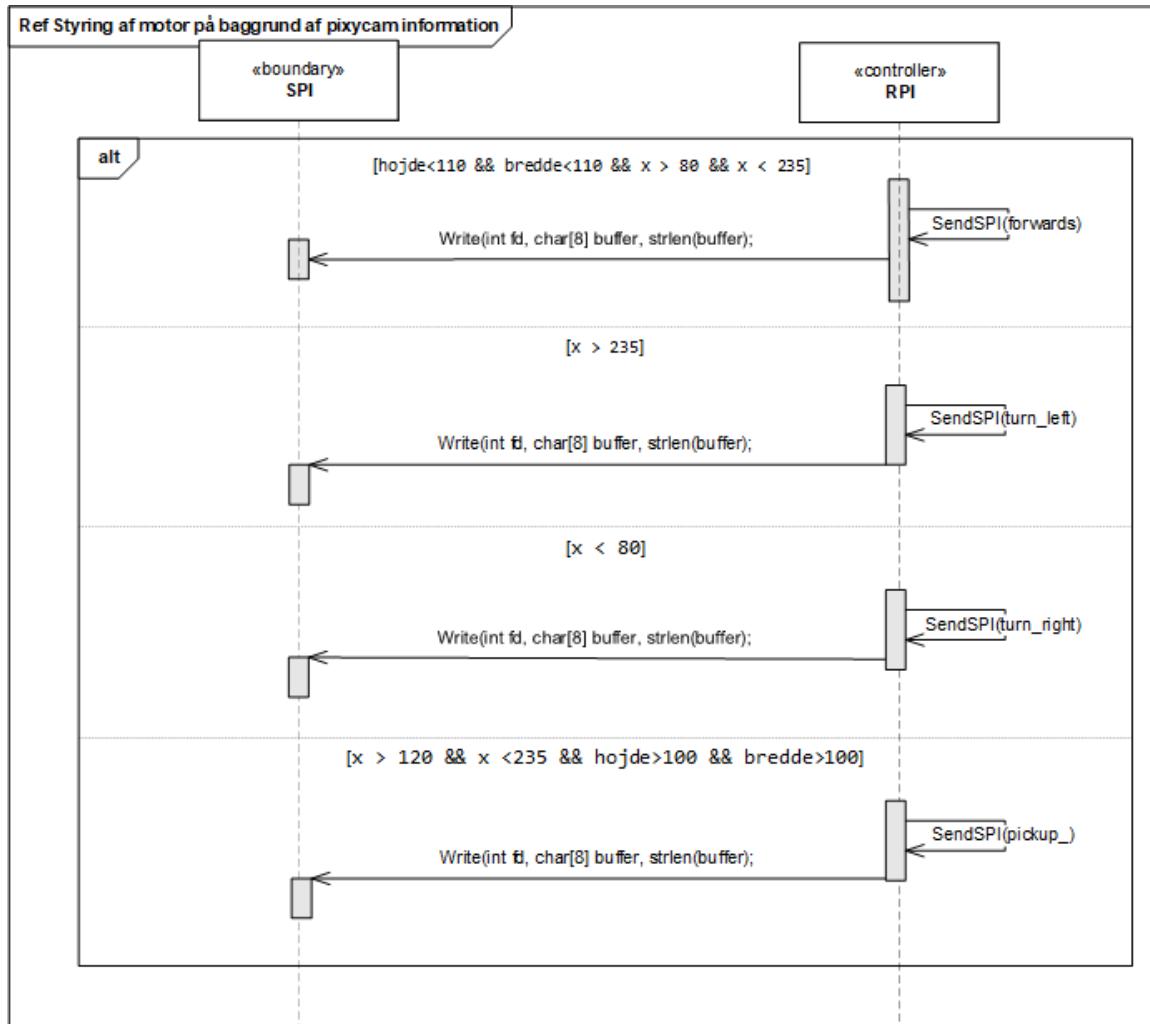
RPi sekvensdiagrammer

På figur 3.20 kan sekvensdiagrammet for RPi ses for UC1 (Start/kør). Sekvensdiagrammet tager brug af en referenceblok, som bliver beskrevet nærmere i figur 3.21. Af system applikations model sekvensdiagrammet fremgår subsystemet med RPi'en som controller. Dette subsystem håndtere hvad der skal sendes over SPI kommunikationen på baggrund af dataen fra pixycam. Hvordan subsystemet håndtere dette fremgår af sekvensdiagrammet på figur 3.20 og på figur 3.21, hvor det autonome system starter fra den tilknyttet webside.



Figur 3.20: Sekvensdiagram for RPi kode

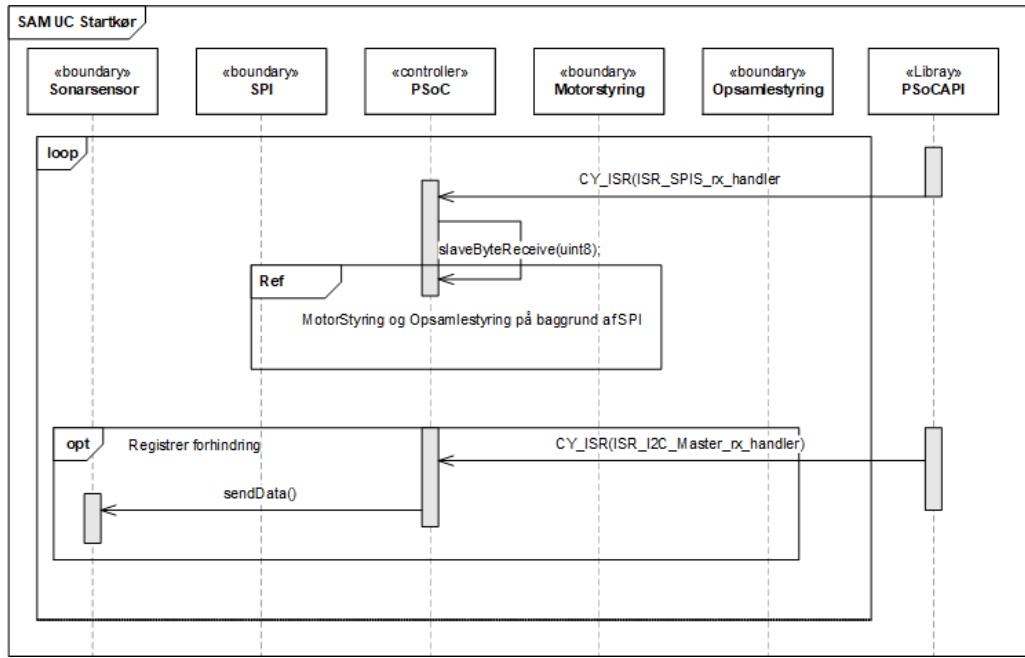
På figur 3.21 fremgår det reference sekvensdiagram, hvoraf styring af hvad der sendes over den serielle kommunikation SPI ses. Dette sker på baggrund af input størrelsen fra pixycam, hvor det bestemmes hvad der sendes over SPI kommunikationen.



Figur 3.21: Reference til sekvensdiagram for RPi kode

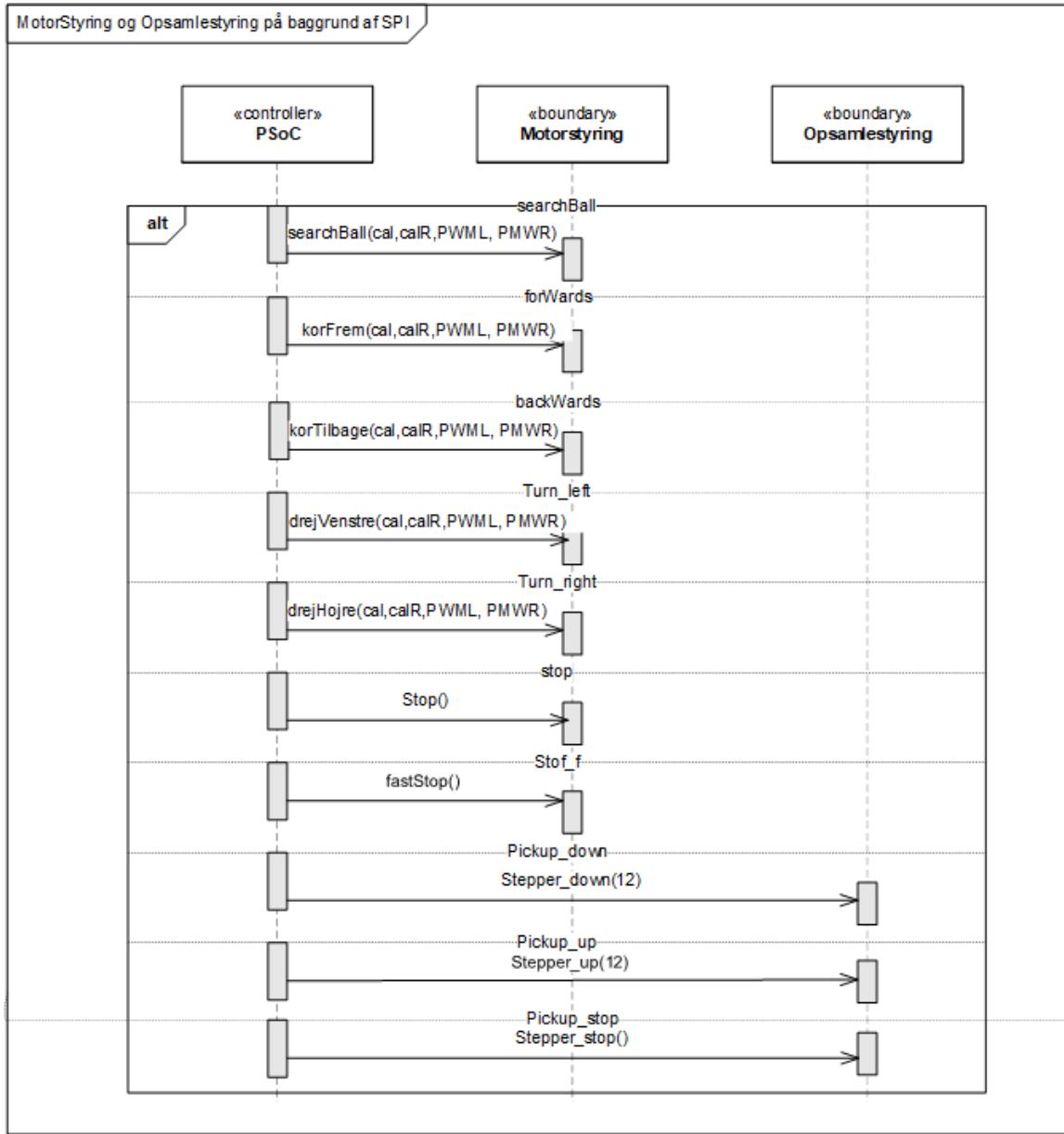
PSoC sekvensdiagram

På figur 3.22 kan sekvensdiagrammet for PSoC koden ses. Der er blevet brugt referencer til at gøre læsningen af diagrammet mere overskueligt. Den første reference, refererer til figur 3.23, hvori det fulde diagram kan ses.



Figur 3.22: Sekvensdiagram for PSoC kode

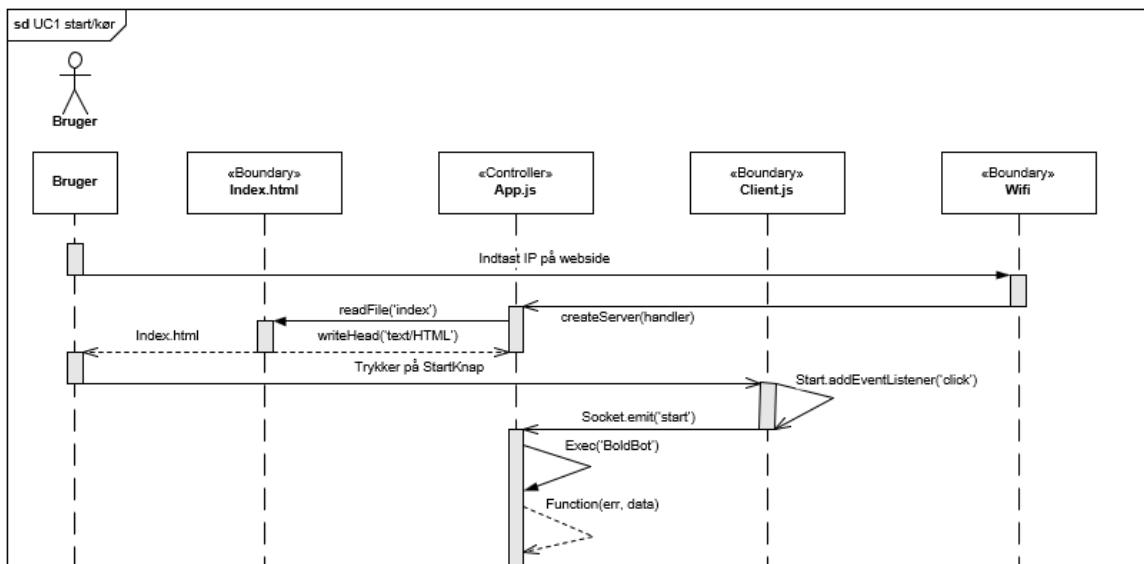
På figur 3.23 kan det ses, hvordan de forskellige metoder for motorstyring samt opsamlerstyringen ses.



Figur 3.23: Reference sekvensdiagram for PSoC kode

Webside sekvensdiagram

på figur 3.24 kan sekvensdiagrammet for Websidemodulet ses for UC1 (start/kør). Det ses hvordan brugeren anvender websiden til at sende et signal fra webside over WIFI til BoldBot.



Figur 3.24: Sekvensdiagram for webside kode

4. Hardware design og implementering

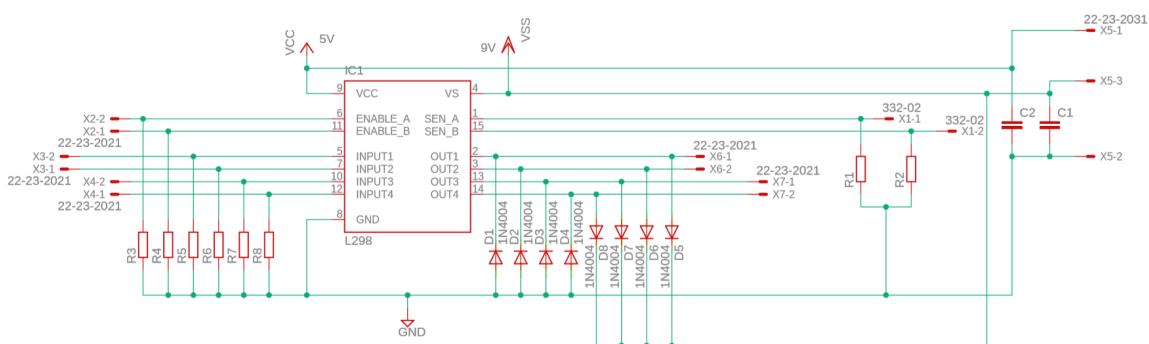
4.1 Motormodul

4.1.1 Indledning

I dette afsnit beskrives det hvordan hardwaren for motorstyringen på BoldBot er designet. Der bliver redegjort for valg af komponenter. Funktionsbeskrivelsen for modulet kan findes i tabellen for blokbeskrivelse i afsnit 3.2.1 på side 17 'Block Definition Diagram'. Til motorstyringsprintet bliver der brugt en PSoC som står for styringssignalene til motorprintet. I denne del er der brugt Eagle til tegning af kredsløbsdiagrammer og Veecad til layout af print.

4.1.2 Design af motorstyring

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 der har 2 H-bro'er indbygget og er både billigere og fylder mindre størrelsesmæssigt. Frem for selv at fabrikere en H-bro.



Figur 4.1: Eagle diagram over motorstyringsprintet

Som det kan ses på Figur 4.1 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 moterne.

Modstandende R1 og R2 er begge effektmodstande på hhv. 1 ohm og 6W og som hver

især sidder i serieforbindelse med en af moterne. 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.

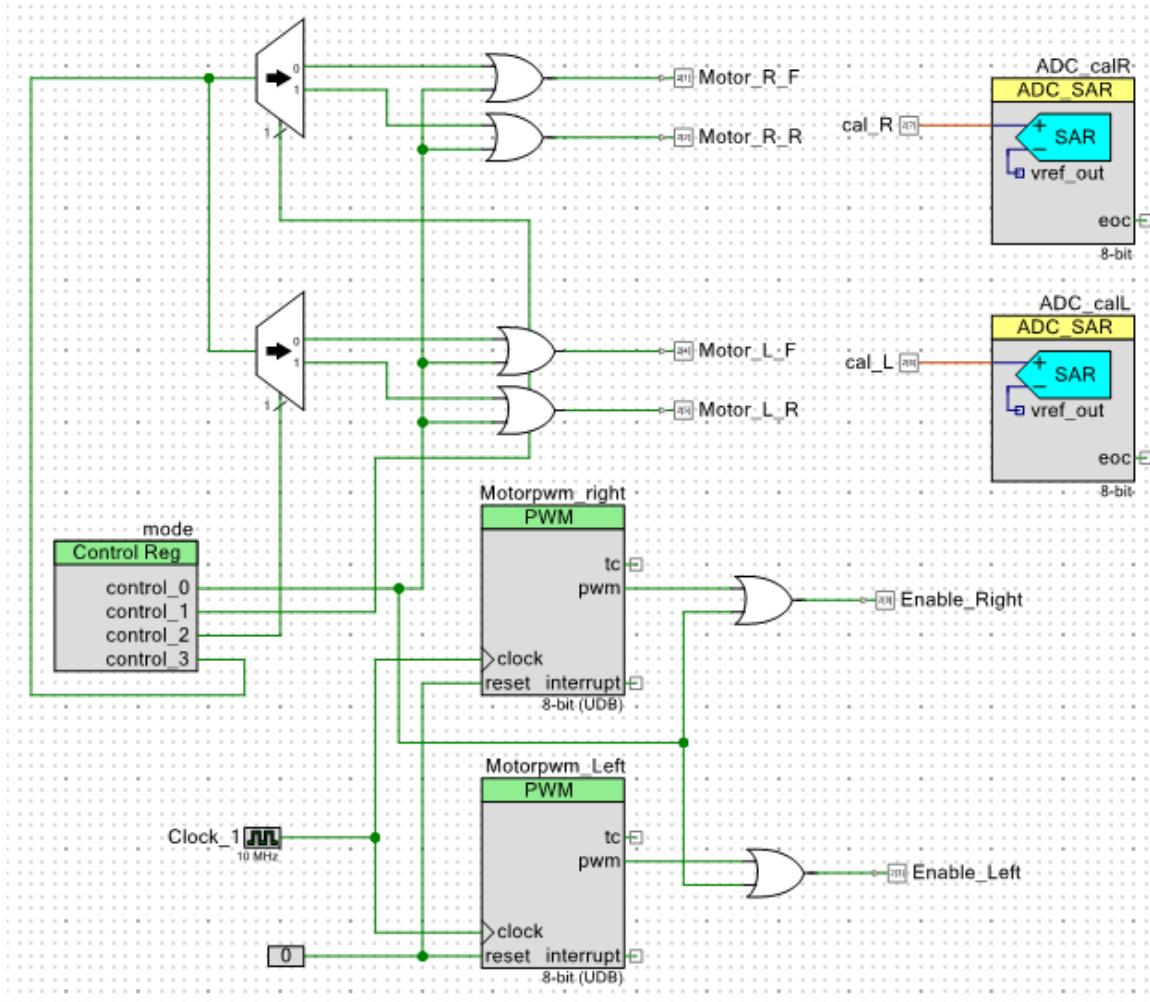
Stikket X5 er forsyningen til både den logiske del i L298 og forsyningspændingen for motorerne. Her er der også sat nogle kondensatorer i forbindelse med de to forsyninger som hhv. er C1 for 9V forsyningen og C2 for 5V forsyningen som er den logiske del i L298.

4.1.3 Stykliste

Stykliste Motorstyring			
Antal	Beskrivelse	Package	RefDes
2	1 Ohm, 6 Watt	-	R1, R2
6	1K ohm, 0.250 Watt	-	R3, R4, R5, R6, R7, R8
6	1N5819	DO-41	D1, D2, D3, D4, D5, D6
1	L298, Dual full-brigde	Multiwatt15	IC1
1	3 pin Euroblok, han	-	X5
3	3 pin Harwin, han	-	X2, X3, X4
2	1 pin Harwin, han	-	X1

4.1.4 Hardware design i PSoC

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 4.2: På denne figur kan diagrammet for motorprintet ses.

Topdesignet på Figur 4.2 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 aktivt bremse.

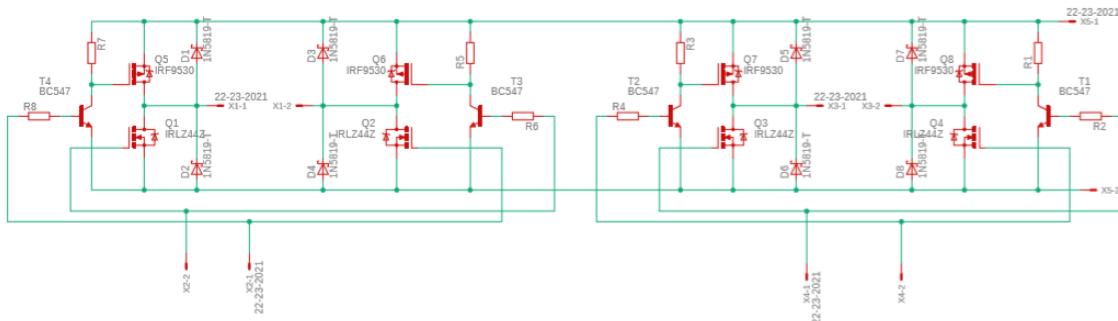
4.2 Opsamlingsmodul

4.2.1 Indledning

Opsamlingsmodulet på BoldBot består af en steppermotor og et driverprint til at drive steppermotoren fra den PSoC som også bruges til motorstyringen.

4.2.2 Design af opsamlingsmodul

For at styre en bipolær steppermotor som der bruges i dette modul, kan denne steppermotor kun kontrolleres ved hjælp af 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. Grunden til spørgsmålet er at leveringen af komponenter, blev denne dobbelte H-bro lavet ud af transistorer og mosfets. Dette kan ses på Figur 4.3.

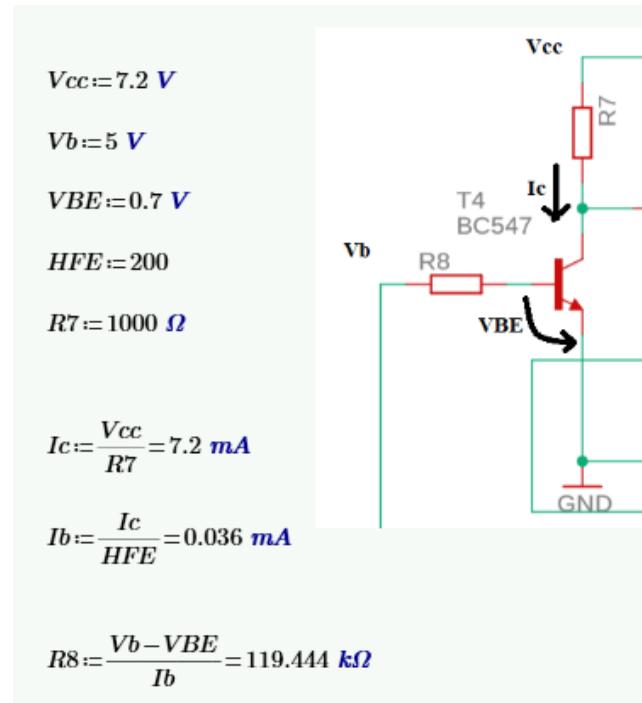


Figur 4.3: På denne figur kan diagram for opsamlingsmodul ses.

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 udgangsterminerne 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 4.3.

IRFZ44Z er valgt da det var den billigste komponent som kunne skaffes inden for vores tidsramme og fordi det var den eneste MOSFET tilgængelig som kunne håndtere den strøm som steppermotoren bruger. Oven i købet kan den drives med logiske spændinger direkte fra en mikrocontroller eller lignende.

Der bliver brugt P-channel MOSFETS til at afbryde og tilslutte VCC til udgangsterminerne X1-1 og X1-2 samt X3-1 og X3-2. Her er der valgt en IRF9Z34 af samme grunde som den type N-channel MOSFET som er valgt til at tilslutte og afbryde ground til udgangsterminerne. Dog kan IRF9Z34 ikke styres ved logiske niveauer, derfor er det nødvendigt at lave et lille transistorkredsløb til at gøre dette muligt. Dette er beskrevet i Figur 4.4.



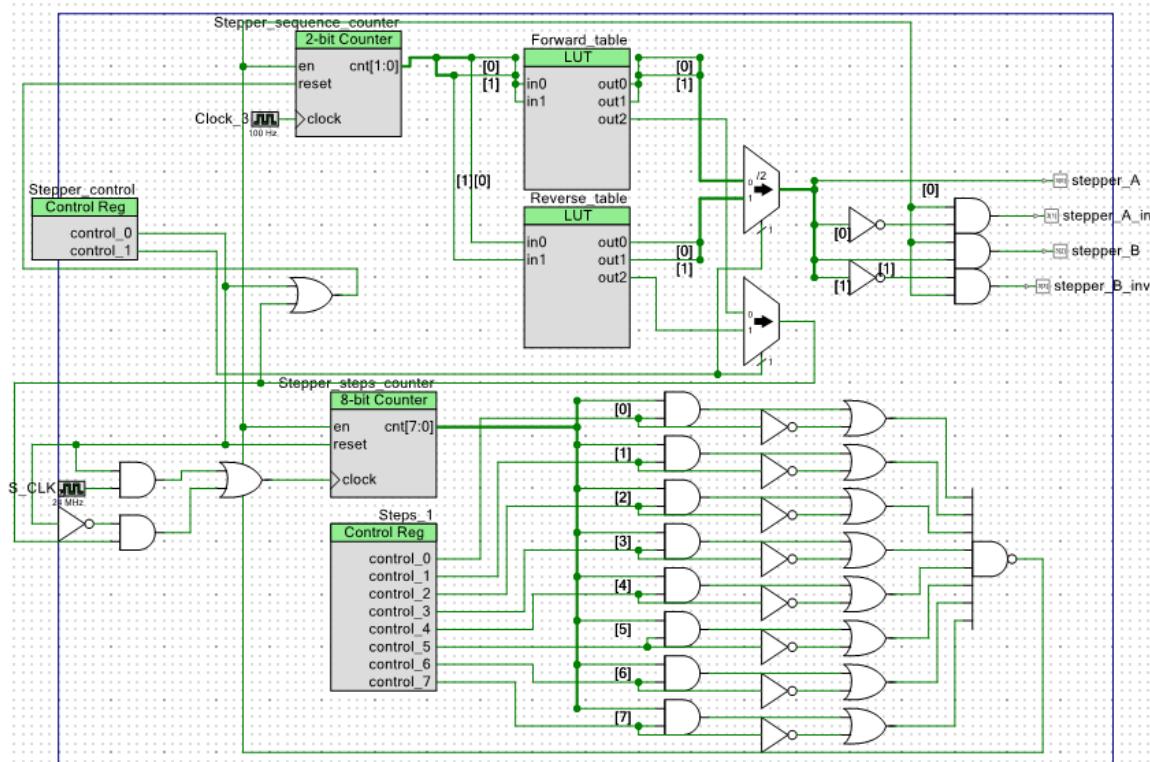
Figur 4.4: På denne figur kan en af transistorene på opsamlingsmodul printet ses.

På Figur 4.4 kan beregningen for transistorene ses i opsamlingsmodulprintet. De er beregnet som switch.

Stykliste opsamlingsmodul

Antal	Beskrivelse	Package	RefDes
2	180K ohm modstand 1%	-	R8, R6, R4, R2
4	IRLZ44Z	-	Q1, Q2, Q3, Q4
4	IRF9Z34	-	Q5, Q6, Q7, Q8
1	1K ohm modstand 1%	-	R1, R3, R5, R7
8	1N5819 Schottky diode	-	D1, D2, D3, D4, D5, D6, D7, D8
3	2 pin Euroblock, han	-	X1, X3, X5
2	2 pin Harwin, han	-	X2, X4

4.2.3 Hardware design i PSoC



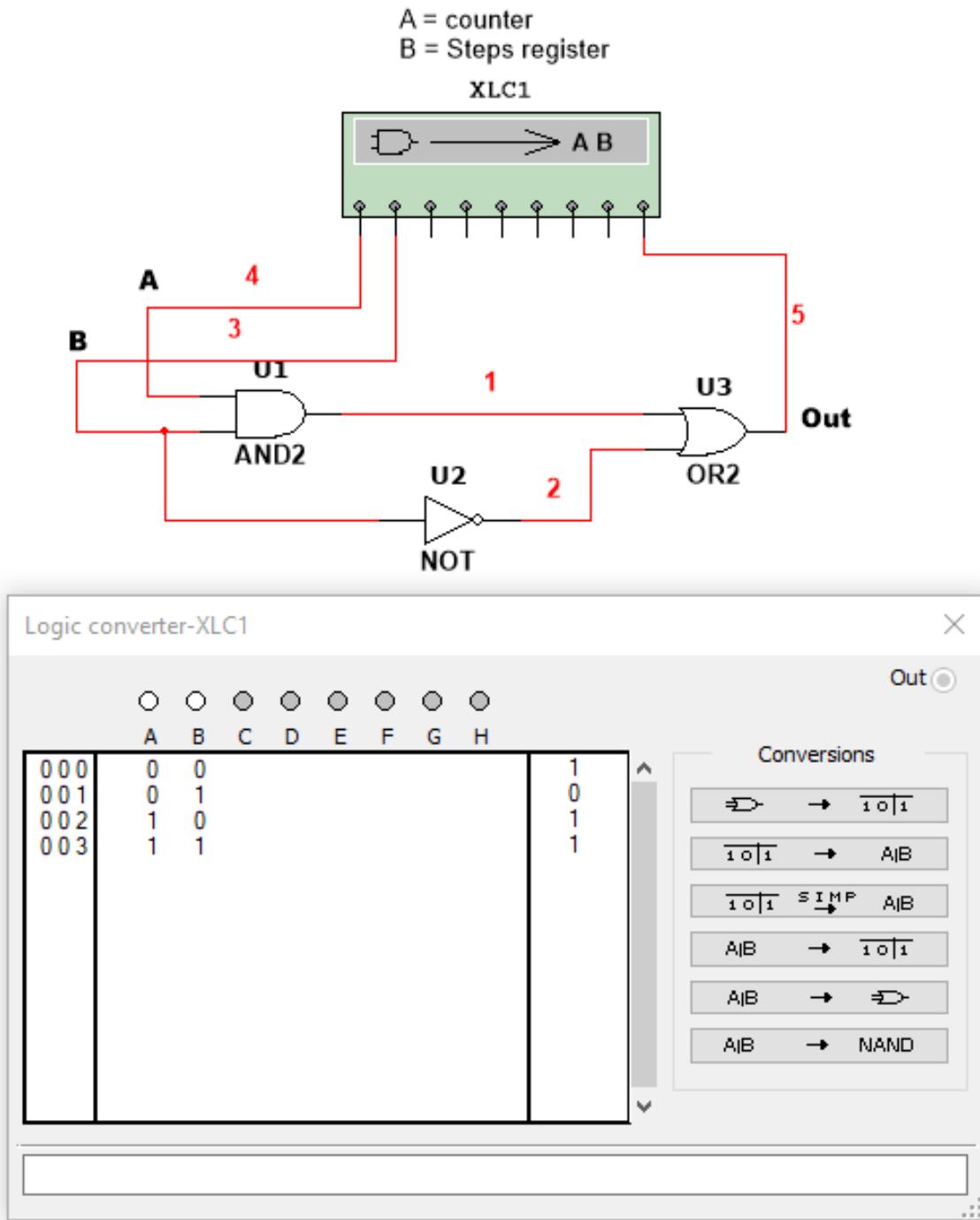
Figur 4.5: På denne figur kan topdesign i PSoC for opsamlingsmodul ses.

Figur 4.5 kan det ses at systemet 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 sekundet. Counteren er koblet til henholdsvis to separate LUT'er, hvor hver LUT er en sandhedstabell som beskriver tændingsrækkefølgen af steppermotoren.

Den LUT som bliver brugt afhænger af, om 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 separate multiplexere hvoraf den øverste vælger den valgte tændingssekvens (rotationsretning) mens den nederste vælger det rette reset signal for den valgte LUT sandhedstabell.

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 inputtet. 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 4.6 som er en sandhedstabel over kredsløbet.



Figur 4.6: På denne figur kan sandhedstabel for det lille gate kredsløb.

Her kan det 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.

4.3 Spændingsregulator

4.3.1 Indledning

I dette afsnit beskrives det hvordan hardwaren for spændingsregulatoren på BoldBot er designet. Der bliver redegjort for valg af komponenter samt en funktionsbeskrivelse af spændingsregulatorprintet.

4.3.2 Design af spændingsregulator

For at regulere spæningen, er der brugt en LM7805 spændingsregulator. Denne leverer et spændingsoutput på typisk 5 V og samt maks og min på 5,2 og 4,8 V. Hvilket er inden for de defineret grænser for 5 V forsyning, på 5 V +- 0,5 V. Output strømmen er i databladet giver ved 1,5 A. (**Im7800**)

Inputspændingen for at sikre lineær regulering er opgivet til at være 7,5 V ved 1 A strømtræk. Ved et strømtræk på 500 mA skal inputspænding ligge på 7 V. Dette kan give problemer, hvis strømrækket bliver større end 1 A, for herved skal inputspændingen være højere en 7,5 V. Da det samelede strømtræk på spændingsregulatoren ikke overskrides 750 mA giver dette dog ikke anledning til problemer. Herved kan det antages at spænding for at opretholde en lineær regulering må ligge mellem 7 og 7,5 V. Dette passer fint da spændingen der skal reguleres er 7,2 V. Selv hvis strømtrækket skulle komme op på 1 A, vil det ikke blive et problem, da den reelle spændingen på det brugte 7,2 V batteri overskrides 7,5 V, og altså ligger på en spænding på ca. 8 V i fuldt opladet tilstand. Regulatoren kan altså reguler ned til 5 V, også selvom der er et højere strømtræk i en kort periode. Tabel over strømtræk fra de forskellige moduler på BoldBot:

Stømforbrug

5 V

Konponent	Strøm [mA]
kamara	140
PSoC	30
Psoc til sonar	5
psoc to IIC	0.006
Psos to I2C	56
Psoc TO H-bro	6
Psoc til spi	41
psoc til stepper	20
Psoc ialt	160
Rpi	200
motorsyng	40

Beregningen af det samlede strømtræk:

$$160 + 140 + 40 + 200 = 540$$

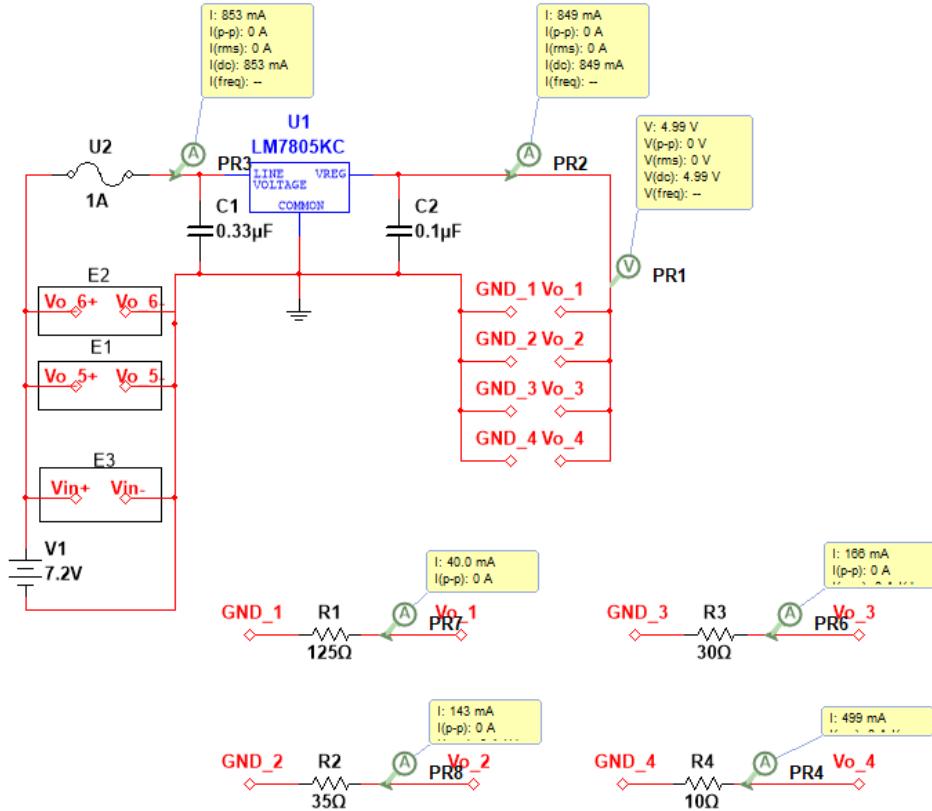
På figur 4.7 ses en simulering af spændingsregulatoren. For at kunne lave en præcis simulering er der udregnet en samlet modstand for hver modul der skal tilsluttes spændingsregulatoren. Disse modsande er fundet ud fra strømtrækket fra de forskellige moduler. Da spændingen over dem, samt strømtrækket er kendt kan modstanden regnes ud fra Ohm's lov.

$$\frac{5.0V}{0.04A} = 125\Omega$$

$$\frac{5.0V}{0.14A} = 35.7\Omega$$

$$\frac{5.0V}{0.16A} = 31.3\Omega$$

$$\frac{5.0V}{0.5A} = 10\Omega$$



Figur 4.7: Multisim simulering af spændingsregulatoren.

For at beskytte kredsløbet bruges der en 1 A sikring. Herefter er det også sat kondensator på indgangen såvel som udgangen af IC'en kondensatoren på output er til for at opnå det optimal stabilitet og transientrespons. Indgangskondensatoren er til stede for at sikre imod støj fra omgivelserne.

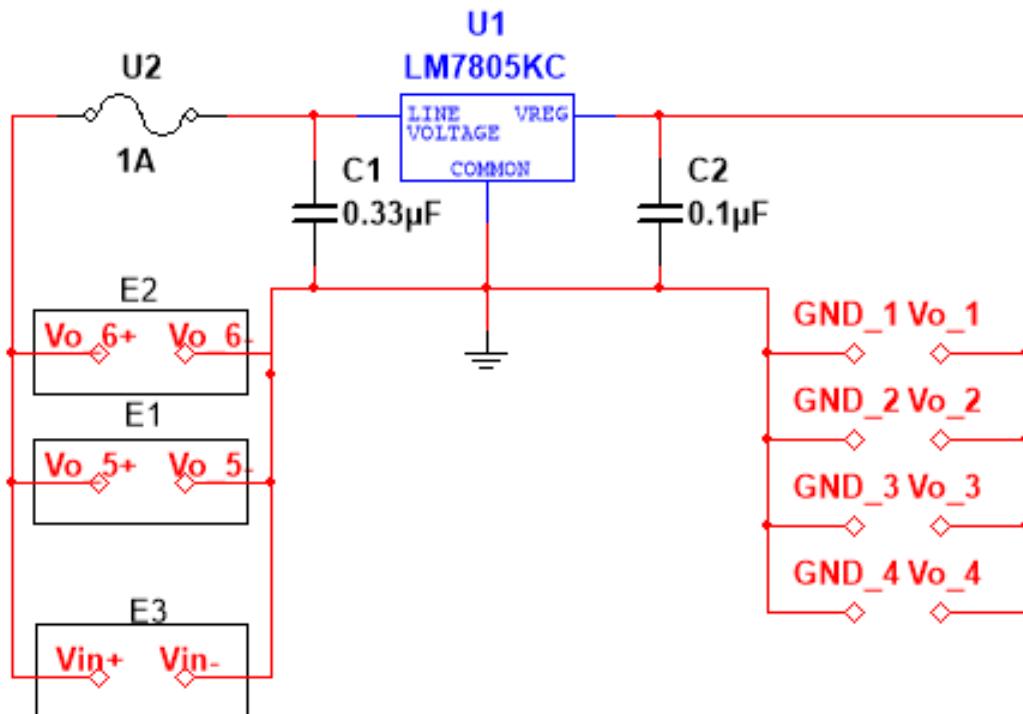
4.3.3 Implementering

Stykliste Spændingsregulator

Antal	Beskrivelse	Package	RefDes
1	LM7805, 1,5 A Fixed Voltage Regulator	TO220	U1
1	5*20mm, 1A Sikring,	-	U2
1	0.33 uF MKT film Kondensatior	-	C1
1	0.10 uF MKT film Kondensatior	-	C2
3	3 pin Euroblok, han	-	E1, E2, E3
2	4 pin Harwin, han	-	-

Implementeringen af modulet er foregået på veroboard. På figur 4.8 Kan der ses et diagram over hvordan de forskellige komponenter af modulet er sat sammen. De tre komponenter E1, E2 og E3 er euroblok stik, E1 er til at tilslutte batteriet til. E2 og E3

er henholdsvis til at tilslutte andet elektronik direkte til batteriet. Vo_1 til Vo_4 er han Harwin stik. Det er her, de moduler der skal forsynes med 5 V, er tilsluttet. GND_1 til GND_4 er også han Harwin stik. Dette er stik til at tilslutte stel til modulerne, herved kan fælles stel for alle moduler opnås, da disse pins alle er loddet direkte til batteriets negative pol.



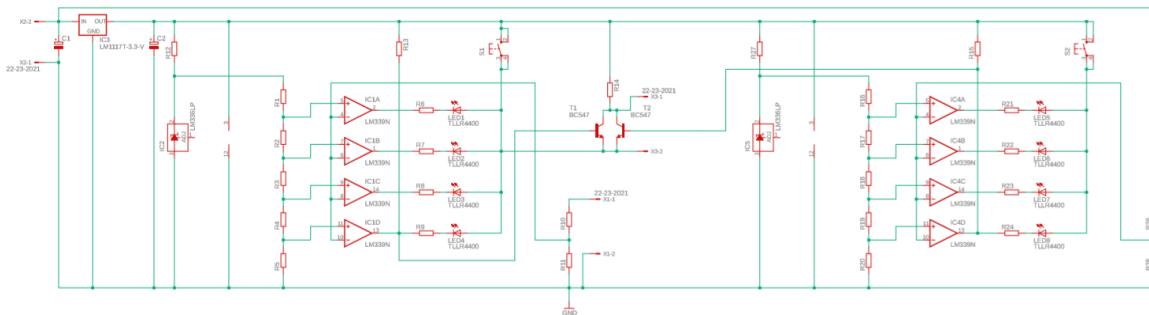
Figur 4.8: Multisim opstilling af spændingsregulatoren.

4.4 Batteriindikator

4.4.1 Indledning

I dette afsnit beskrives det hvordan hardwaren for batteriindikatoren på BoldBot er designet. Der bliver redegjort for valg af komponenter samt en funktionsbeskrivelse af motorstyringsprintet. Til motorstyringsprintet bliver der brugt en PSoC som står for styringssignalene til motorprintet. I denne del er der brugt Eagle til tegning af kredsløbsdiagrammer og Veecad til layout af print.

4.4.2 Design af Batteriindikator



Figur 4.9: På denne figur kan kredsløbet for batteriindikatoren ses.

På Figur 4.9 kan diagrammet for batteri-indikatoren ses. Batteri-indikatoren er opbygget med en spændingsregulator på 3.3 volt til at forsyne de otte komperatorer af typen LM339, som bruges til at måle de forskellige spændingsniveauer på de to batterier. Denne type komparator kan køre på 3.3 volt og så har den en lav offsetspænding på indgangene hvilket gør den ideel til vores brug. For at komperatorne kan vurderer 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 4.9, en til hvert batteri. Foran de to zenerdioder sidder der hver en modstand for at begrænse strømmen som løber igennem den individuelle diode. På Figur 4.10 kan udregningen for begge modstande R12 og R27 ses.

Simulering:

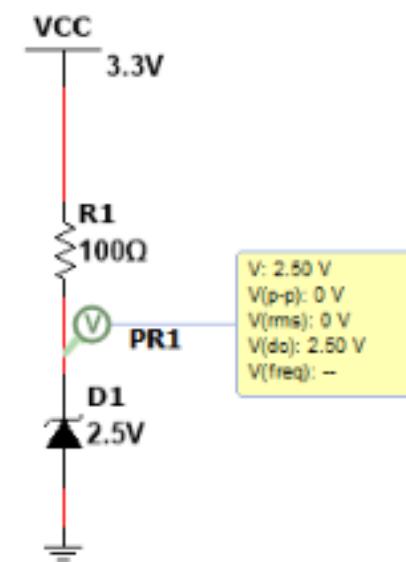
Suppliespænding:

$$V_{CC} = 3.3 \text{ V}$$

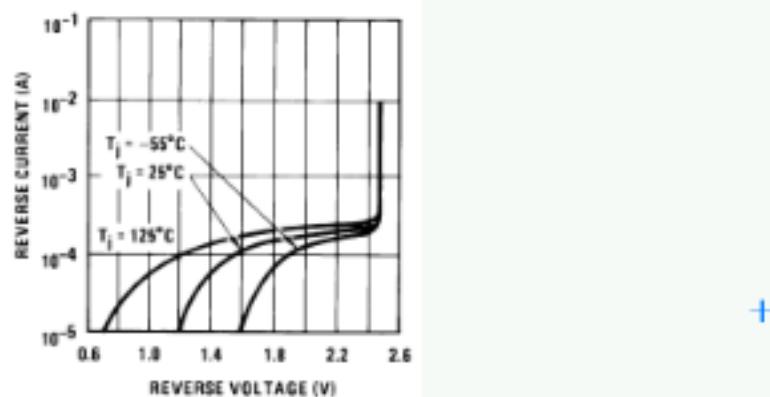
Spændingsfald over diode:

$$U_{Diode} = 2.5 \text{ V}$$

Dioden kan operere mellem 400 μA til 10mA.
Der er valgt en strøm på 8 mA. Dette kan også ses på grafen under, her gælder det om at have en reverse voltage på 2.5 volt.
Da det er vores ønskede reference spænding.
Men strømmen må ikke overstige 10^{-2} som er 10 mA, da det er grænsen for denne diode.



Reverse Characteristics



$$I_{Diode} = 8 \text{ mA}$$

Ud fra værdierne kan formodstanden for reference/ zenerdioden bestemmes, ved Ohm's lov.

$$R_1 := \frac{V_{CC} - U_{Diode}}{I_{Diode}} = 100 \Omega$$

Derfor skal både R12 og R27 være 100 Ohm.

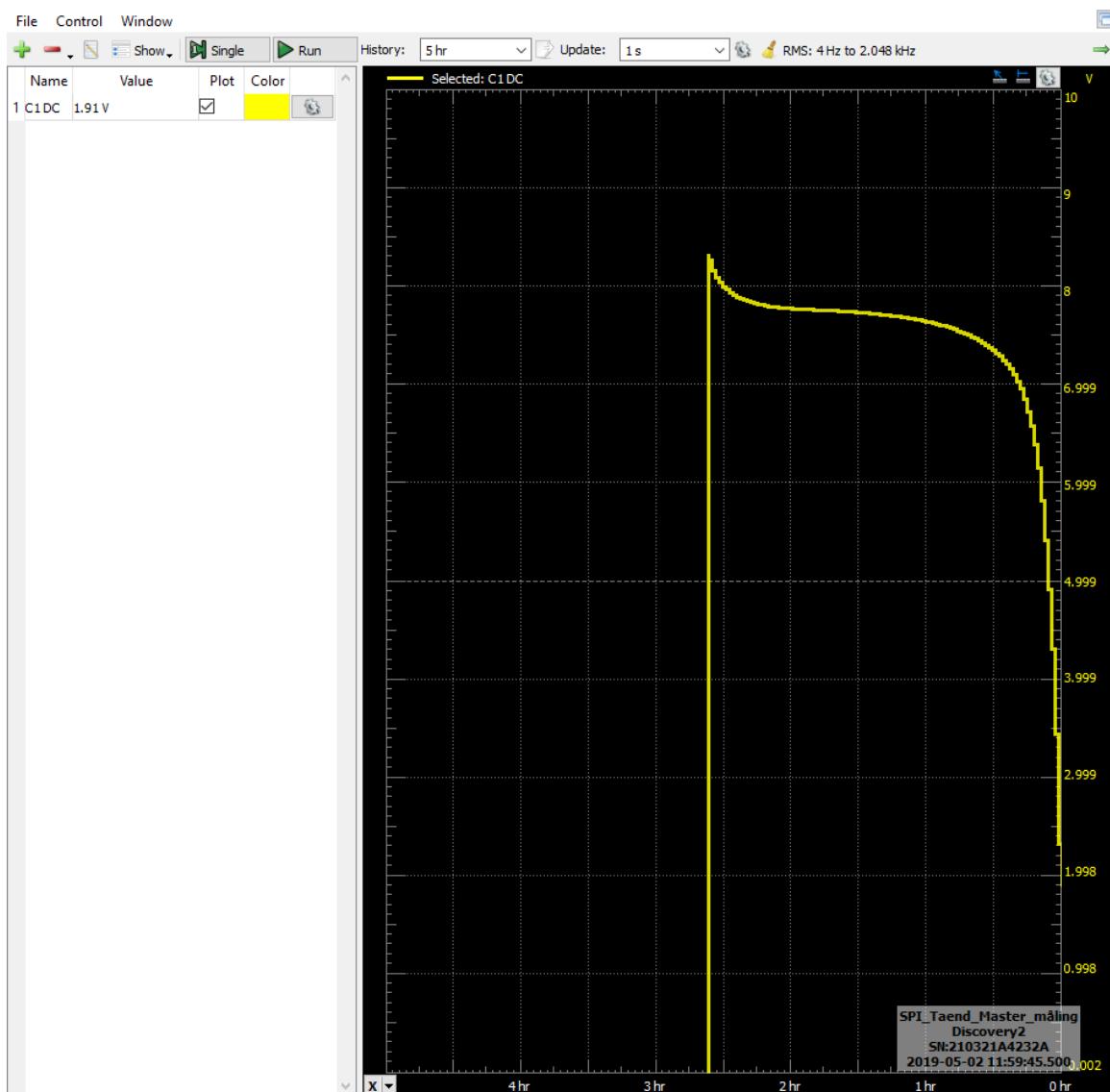
Figur 4.10: På denne figur kan kredsløbet for batteriindikatoren ses.

Efter spændingsreferencen bliver spændingen delt ned ved hjælp af en stor spæn-

dingsdeler som består af R₁, R₂, R₃, R₄, R₅ for det ene batteri på 9.6 volt og R₁₆, R₁₇, R₁₈, R₁₉, R₂₀ for det andet 7.2 volt batteri. Værdierne for disse modstande findes senere i dette afsnit ved hjælp af en afladningskurve.

For at kunne lave en batteri-indikator til Boldbot er man først nødt til at lave en afladningskurve for begge batterier. Dette skal gøres fordi at et batteri ikke aflader lineært, hvilket gør at batteriet pludselig kan aflade meget hurtigt. Derfor er det vigtigt at lave en afladningskurve på begge batterier, så man lige nøjagtig kan se kapaciteten af ved hvilke spændingsniveauer på batteriet.

Dette har 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 på batteri-indikatoren at den skal vise følgende batteri niveauer på henholdsvis 20,40,60 og 80 procent.



Figur 4.11: På denne figur kan afladningskurven for 7.2 volt batteriet ses.

På figur 4.11 kan afladningskurven for 7.2 Volt batteriet ses. Ud fra kurven kan man beregne antal procent kapacitet på batteriet og se fra hvilken spænding falder energien i batteriet hurtigst inden det er helt afladt. På Figur 4.12 kan beregningen af de forskellige procent niveauer ses og dertil også beregning af spændingsdeler til brug som referencespænding senere i dette afsnit.

7.2V batteri

$$Enprocent := \frac{9377}{100} = 93.77 \quad \text{En procent af den totale afladningstid.}$$

$$80 \cdot Enprocent = 7.502 \cdot 10^3 \quad \text{Aflæst til: } V_{Bat80} := 7.76551$$

$$60 \cdot Enprocent = 5.626 \cdot 10^3 \quad \text{Aflæst til: } V_{Bat60} := 7.73806$$

$$40 \cdot Enprocent = 3.751 \cdot 10^3 \quad \text{Aflæst til: } V_{Bat40} := 7.6516$$

$$20 \cdot Enprocent = 1.875 \cdot 10^3 \quad \text{Aflæst til: } V_{Bat20} := 7.37023$$

$$VCC := 2.5$$

$$V_1 := V_{Bat80} \cdot \frac{16000}{16000 + 43000} = 2.106$$

$$V_2 := V_{Bat60} \cdot \frac{16000}{16000 + 43000} = 2.098$$

$$V_3 := V_{Bat40} \cdot \frac{16000}{16000 + 43000} = 2.075$$

$$V_4 := V_{Bat20} \cdot \frac{16000}{16000 + 43000} = 1.999$$

$$R_{16} := 820$$

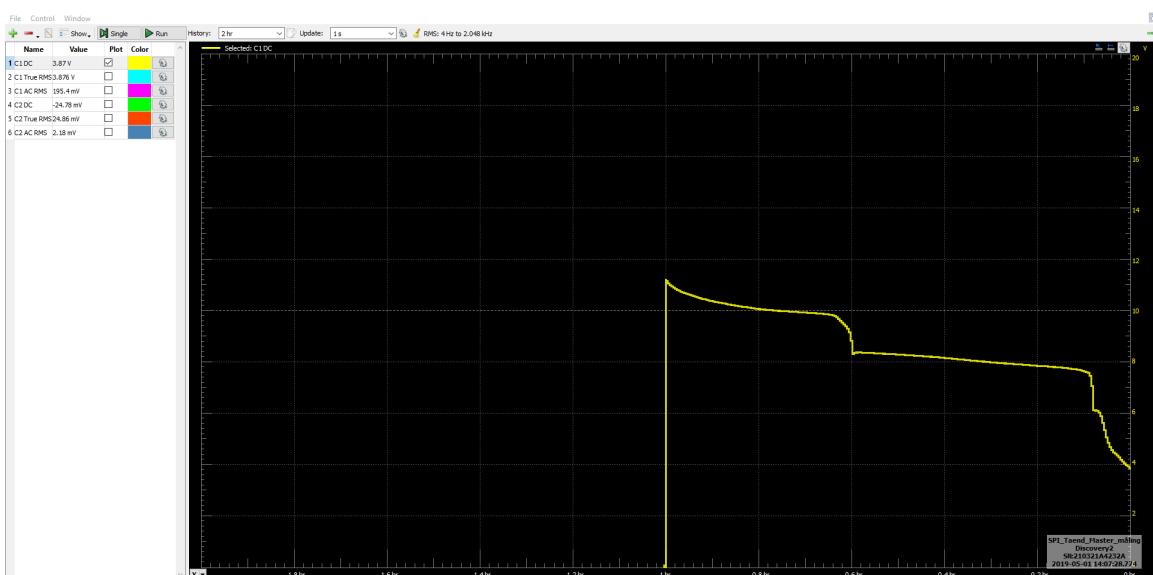
$$\begin{bmatrix} R_{17} \\ R_{18} \\ R_{19} \\ R_{20} \end{bmatrix} := \left[\begin{array}{l} V_1 = VCC \cdot \frac{R_{17} + R_{18} + R_{19} + R_{20}}{R_{16} + R_{17} + R_{18} + R_{19} + R_{20}} \\ V_2 = VCC \cdot \frac{R_{18} + R_{19} + R_{20}}{R_{16} + R_{17} + R_{18} + R_{19} + R_{20}} \\ V_3 = VCC \cdot \frac{R_{19} + R_{20}}{R_{16} + R_{17} + R_{18} + R_{19} + R_{20}} \\ V_4 = VCC \cdot \frac{R_{20}}{R_{16} + R_{17} + R_{18} + R_{19} + R_{20}} \end{array} \right] \xrightarrow{\text{solve, } R_{17}, R_{18}, R_{19}, R_{20}} [15.488 \Omega, 48.78 \Omega, 158.76 \Omega, 4158 \Omega]$$

$$R_{17} := 15.488 \Omega \quad R_{18} := 48.78 \Omega \quad R_{19} := 158.76 \Omega \quad R_{20} := 4158 \Omega$$

Figur 4.12: På denne figur kan beregning for 7.2 volt batteriet ses.

På Figur 4.12 kan udregningen af de forskellige batteri-niveauer ses. Dette gøres først ved at tage den tid som batteriet har afladet i og dividere det med 100 som er gjort øverst. Dette er 1% af batteriet, derfor ganges det med henholdsvis 80,60,40 og 20 procent, hvor man efter kan gå ind i tabellen for afladningskurven og finde den præcise spænding for et givet niveau. Det kan ses at for en kapacitet på 80% har batteriet en spænding på 7.76551 volt, dette er gentaget nedad.

Den standard spændingsreference som skal bruges i dette batteri indikator kredsløb kan kun levere 2.5 volt derfor er VCC sat til 2.5 volt. Dette gør også at vi ikke kan koble batteriet direkte på batteri-indikatoren uden af bruge en spændingsdeler på batteri siden også. Hvilket er derfor de forskellige batterispændinger ganges med $16000/16000+43000$ ohm. Her fås en række spændinger som ikke ligger højere end spændingsreferencen på 2.5 volt. Herefter laves en ny afladningskurve for 9.2 volt batteriet som ses på Figur 4.13.



Figur 4.13: På denne figur kan afladningskurven for 9.6 volt batteriet ses.

Ud fra afladningskurven på Figur 4.13 kan spændingsniveauerne for batteriets procentvise kapacitet findes. Dette udregnes på Figur 4.14.

9.6 volt batteri:

$$Enprocent := \frac{2723}{100} = 27.23$$

$$VCC := 2.5$$

$$80 \cdot Enprocent = 2.178 \cdot 10^3 \quad \text{Aflæst til: } V_{Bat80} := 9.1699$$

$$60 \cdot Enprocent = 1.634 \cdot 10^3 \quad \text{Aflæst til: } V_{Bat60} := 8.2504$$

$$40 \cdot Enprocent = 1.089 \cdot 10^3 \quad \text{Aflæst til: } V_{Bat40} := 8.0056$$

$$20 \cdot Enprocent = 544.6 \quad \text{Aflæst til: } V_{Bat20} := 7.7962$$

$$V_1 := V_{Bat80} \cdot \frac{100000}{300000 + 100000} = 2.292$$

$$V_2 := V_{Bat60} \cdot \frac{100000}{300000 + 100000} = 2.063$$

$$V_3 := V_{Bat40} \cdot \frac{100000}{300000 + 100000} = 2.001$$

$$V_4 := V_{Bat20} \cdot \frac{100000}{300000 + 100000} = 1.949$$

$$R_1 := 820$$

$$\begin{bmatrix} R_2 \\ R_3 \\ R_4 \\ R_5 \end{bmatrix} = \left[\begin{array}{l} V_1 = VCC \cdot \frac{R_2 + R_3 + R_4 + R_5}{R_1 + R_2 + R_3 + R_4 + R_5} \\ V_2 = VCC \cdot \frac{R_3 + R_4 + R_5}{R_1 + R_2 + R_3 + R_4 + R_5} \\ V_3 = VCC \cdot \frac{R_4 + R_5}{R_1 + R_2 + R_3 + R_4 + R_5} \\ V_4 = VCC \cdot \frac{R_5}{R_1 + R_2 + R_3 + R_4 + R_5} \end{array} \right] \xrightarrow{\text{solve, } R_2, R_3, R_4, R_5} [908.312251;]$$

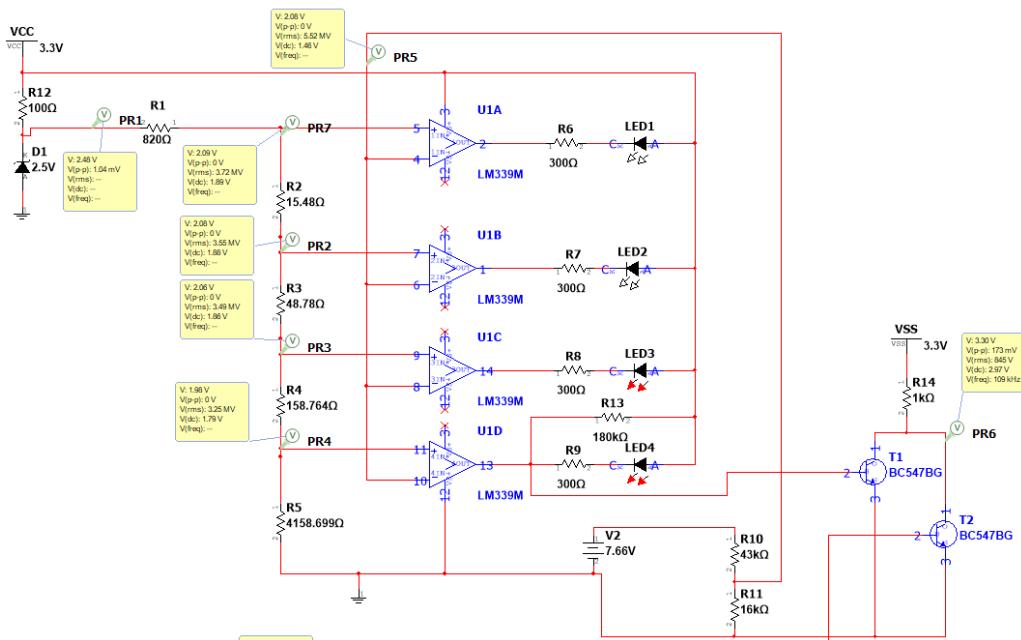
$$R_2 := 908.31 \quad R_3 := 241.82 \quad R_4 := 206.85 \quad R_5 := 7701.34$$

Figur 4.14: På denne figur kan udregningen af procent og modstande for 9.6 volt batteriet ses.

På Figur 4.14 findes først 1 procent af den tid batteriet er afladt i, hvilket er 27.23 sekunder. Dette tal ganges med henholdsvis 80, 60, 40 og 20 procent for at finde tids punkterne for disse niveauer. Herefter findes disse tidspunkter inde i vores tabel for af-

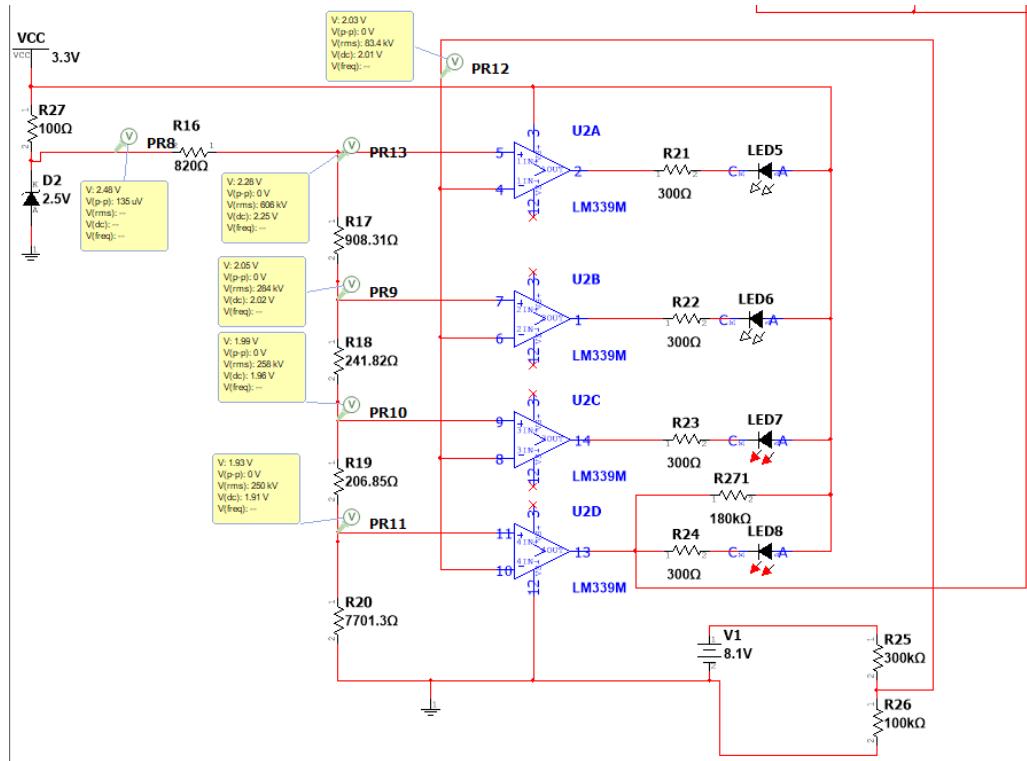
ladningskurven hvor de fire spændingsværdier findes, som ses som "aflæst til:" for hver udregning. Da 9.6 volt batteriet er for høj en spænding i forhold til spændingsreferencen på 2.5 volt, bliver de 9.6 volt først kørt igennem en spændingsdeler. Hvilket er derfor batterispændingerne ganges med $100000/300000+100000$. Nederst på Figur 4.14 kan resultatet for modstandsværdierne ses.

Figur 4.15 viser det den ene del af det færdige kredsløb for batteriindikatoren som indikerer batteriniveauet for 7.2 volt batteriet.



Figur 4.15: På denne figur kan udregningen af procent og modstande for 9.6 volt batteriet ses.

Figur 4.15 viser samtidigt også en simulering over kredsløbet, der bliver tilført en batterispænding på 7.66 volt. Ifølge beregningen på Figur 4.12 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 det kan ses på Figur 4.15 at de gør.



Figur 4.16: På denne figur kan udregningen af procent og modstande for 9.6 volt batteriet ses.

Den anden del af kredsløbet er identisk med det første, bortset fra nogle af modstandsverdierne. Da dette kredsløb måler batteriniveauet på 9.6 volt batteriet i stedet for 7.2 volt batteriet. Kredsløbet kan ses på Figur 4.16 hvor det er simuleret med en batterispænding på 8.1 volt. På Figur 4.14 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 4.16.

I dette projekt var det meningen at batterimåleren skulle kobles til den Raspberry Pi som bruges på Boldbot. Derfor er der lavet et lille transistorkredsløb som leverer et højt signal på 3.3 volt når batteriniveauet er over 20 procent på begge batterier og et lav signal når batteriniveauet på et af de to batterier er under 20 procent. Grunden til at det blev gjort med to transistorer istedet for en gate er at det er væsenlig billigere og at man ikke kan få en enkelt NOR i en fysisk pakke, derfor ville der være overflødige komponenter også. Beregning af transistorernes formodstand R13 og R15 er den samme som beregningen på Figur 4.4 i Design af opsamlingsmodul.

Stykliste
Batterimåler

Antal	Beskrivelse	Package	RefDes
2	LM339 Komperator	DIP	U1, U4
2	BC547B bipolar NPN transistor	-	T1, T2
2	100 Ohm modstand 1%	-	R12, R27
8	300 Ohm modstand 1%	-	R6, R7, R8, R9, R21, R22, R23, R24
2	Rød LED	-	LED4, LED8
4	Gul LED	-	LED2, LED3, LED6, LED7
2	Grøn LED	-	LED5, LED1
2	180K ohm modstand 1%	-	R13, R15
1	43K ohm modstand 1%	-	R10
1	16K ohm modstand 1%	-	R11
2	820 ohm modstand 1%	-	R1, R16
1	15 ohm modstand 1%	-	R2
1	50 ohm modstand 1%	-	R3
1	158 ohm modstand 1%	-	R4
1	4.158K ohm modstand 1%	-	R5
1	908,31 ohm modstand 1%	-	R17
1	241.8 ohm modstand 1%	-	R18
1	206.85 ohm modstand 1%	-	R19
1	7.70K ohm modstand 1%	-	R20
1	300K ohm modstand 1%	-	R25
1	100K ohm modstand 1%	-	R26
2	2.5 volt reference diode LM336 1%	-	IC2, IC5
1	LM1117T	-	IC3
2	100 nF MKT metalfilm kondensator	-	C1, C2
2	2 pin Euroblock connector	-	X1, X2
1	2 pin Harwin	-	X3
2	tactile switch	-	S1, S2

4.5 Sonarmodul

4.5.1 Indledning

4.5.2 Design af Sonarmodul

4.5.3 Implementering

5. Software design og implementering

5.1 RobotApp

5.1.1 Indledning

I dette afsnit beskrives det, hvordan softwaren på RPI er designet. Der bliver redegjort for valg af metoder, kommunikation og kode. Derudover vil overvejelser i løbet af processen blive beskrevet, samt dybdegående forklarelser af kode gennemgået. Der vil både blive gennemgået implementering af kernemoduler, overlays, PixyCam, PSoC og metoder til aflæsning på kernemoduler.

5.1.2 Design af SPI-kernemodul og device tree overlay

Et kort resumé er givet i design afsnittet i rapporten, og den fulde beskrivelse ses herunder.

Implementeringen af kernemodulet, er implementeret som et Hot-pluggable dynamisk kernemodul, der automatisk sætter major og minor nummer. Til implementeringen er der lavet følgende metoder: `spiDriver_init()`, `spiDriver_exit()`, `spiDriver_read()`, `spiDriver_write()`, `spiDriver_probe()`, `spiDriver_remove()`.

`spiDriveInit()`: Initierer driveren med major og minor nummer, og registrerer driveren i kernen.

`spiDriver_exit()`: Denne funktion bruges til at rydde op i al den allokerede hukommelse, som driveren har taget i brug.

`spiDriver_probe()`: Henter chipselect samt baud rate fra device tree, og sætter hvor mange bits der skal sende pr ord. Derudover bliver devicet oprettet i kernen, og dermed bliver device nummer tildelt.

`spiDriver_remove()`: Frigiver device nummereet og deallokerer device klassen.

`spiDriver_read()`: Læser det som er kommet ind i tx bufferen for SPI devicet, og returnerer dette i en buffer.

`spiDriver_write()`: Tager en buffer givet fra brugerden, laver det om til et integer og ligger det over i rx bufferen for SPI devicet, så det bliver sendt til slaven som det næste.

Udover kernemodulet er der i forbindelse med driveneren også lavet et device tree overlay. Det indsættes sammen med modulet, og er det som gør det muligt at lave hotplugging. I device tree overlayet beskrives der for det specifikke device, hvilke CPU'er og CPU devices som er kompatibelt med det. Devicet SPI specifikke detaljer sættes også hvilken chip select der bruges, om CPHA og CPOL er aktive og max frekvens.

5.1.3 Design af GPIO driver-kernemodul og device tree overlay

Dette afsnit beskriver designet af kernemodulet og device tree overlay, der gør det muligt at læse og skrive fra GPIO 17. Kernemodulet er som ved SPI-kernemodulet implementeret som et Hot-pluggable dynamisk kernemodul, hvilket betyder at major og minor numre er automatisk sat, major ved indsætelse af modulet og minor ved registrering af nyt modul. Der er implementeret følgende metoder i kernemodulet:

`platDriver_init()`: Initierer driveren med major og minor nummer, og registrerer driveren i kernen.

`platDriver_exit()`: Denne funktion bruges til at rydde op i al den allokerede hukommelse, som driveren har taget i brug.

`platDriver_probe()`: Som henter GPIO nr samt GPIO direction fra device tree, og sætter disse. Derudover bliver en device nummeret tildelt.

`platDriver_release()`: Frigiver de GPIO pins som probe har anmodet om at få tildelt, og frigiver også det device nummer der er tildelt.

`platDriver_open()`: Åbner det givne device, så det er muligt at skrive til eller fra det.

`platDriver_close()`: Lukker det givne device, så det ikke længere er muligt at skrive til eller fra det.

`platDriver_read()`: Denne funktion gør det muligt at læse fra den valgte GPIO pin, der kan kun læses højt eller lavt, altså 1 eller 0. Funktionen kan kun bruges, hvis direction er sat til input.

`platDriver_write()`: Kan skrive ud fra den givne GPIO pin. Ligesom med read kan man kun sende højt eller lavt ud, 1 eller 0. Denne funktion kan kun bruges hvis at direction er sat til output.

I device treeoverlay sættes hvilke GPIO pins som skal initieres samt om det skal være en input eller output pin. Udover dette, bestemmes hvilke CPU'er og CPU devices som devicet er kompatiblet med, og hvilke ikke CPU devices som den er komaptibel med.

5.1.4 SPI

Efter der er blevet lavet et kernemodul til kommunikation mellem PSoC og RPi, skal der implementeres to metoder, som kan henholdsvis læse fra og skrive til det oprettede device (`spi_drv0`). For at kunne sende informationer fra Master (RPi) til Slave (PSoC) bliver `'void sendSPI(int)` implementeret. Metoden åbner devicet, for herefter give en buffer en værdi, som er bestemt i parametreren. Herefter vil bufferen blive skrevet til dette device, som herefter kan sende det over SPI protokollen til PSoC. Samme procedure foregår, når der skal modtages data fra Slave til Master. Her bliver der bare gjort brug af `'void receiveSPI()'`. Vi har anvendt samme fremgangsmåde i denne funktion, men i stedet for at skrive til det oprettede device, bliver der læst herfra.

5.1.5 Sonar

Efter der er blevet lavet et kernemodul, der kan læse når en GPIO indgang er høj, skal der laves en metode, som kan åbne denne driver. Til dette bliver der implementeret metoden 'void init_sonar()'. Denne metode har til formål at åbne det oprettede device (gpio17), som herefter gør det muligt for 'void read_sonar()' at kunne læse på det oprettede device. Der bliver læst på devicet vha. funktionen 'read()', som tager tre parametrer, og som fylder en buffer med enten et højt signal (1) eller et lavt signal (0). Hvis bufferen er høj, vil BoldBot undvige det registrerede objekt.

5.2 PSoC

5.2.1 Indledning

I dette afsnit beskrives alt koden der er blevet implementeret i PSoC Creator. Koden er overordnet set blevet inddelt i 4 forskellige moduler - motorstyring, sonarsensor, opsamlerstyring samt alt kommunikationen. Alle disse moduler er beskrevet nærmere i arkitektur afsnittet, hvor strukturen nærmerer kan gennemlæses.

5.2.2 motorStyring

Motorstyrings programmet på PSoC håndtere retningen hvoraf de to anvendte DC motorere rotere. Programmet består af 8 funktioner som tilsammen styrer hvordan DC motorerne skal rotere således BoldBot drejer, bakker, kører frem, stopper og leder efter en bold. Koden er implementeret ved at skrive til motorstyringens kontrol registre i topdesignet, hvoraf pinsene på PSoC sættes til logisk 1 eller 0 alt efter hvilke funktioner der kaldes. De forskellige funktionskald for motorstyring sker i en switch case i main programmet på PSoC.

5.2.3 OpsamlerStyring

Opsamlestyrings programmet allokeret på PSoC håndtere styring af den anvendte stepper motor, som opsamlemodulet består af. Programmet består af fire funktioner, som består af en init, en løfte op funktion, en løfte ned funktion og en stop funktion. Tilsammen bruges disse funktionen til at håndter opsamlingen af en registeret tennisbold i main programmet på PSoC. Dette er implementeret ved at skrive til de kontrol registrene for stepper motoren i topdesignet på PSoC.

5.2.4 SonarSensor

Der er blevet udviklet et program på PSoC til at læse fra sonaren. Da sonaren bruger I2C kommunikation, er det udviklet ved brug af I2C. Koden består af 2 funktioner. Den ene funktion returnerer distancen til et registeret objekt (getDistance()), hvor den anden

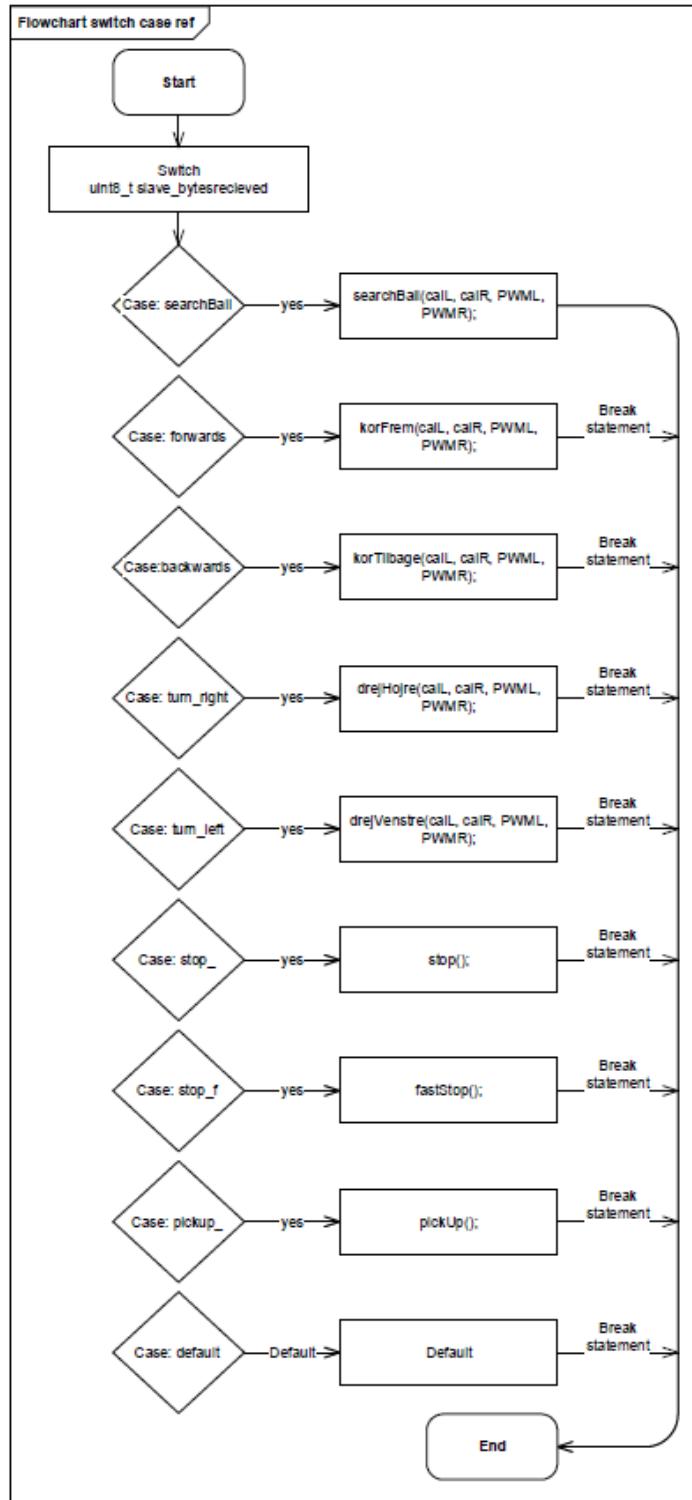
læser på dennes returværdi, og sender et højt signal til en GPIO på RPi, hvis distancen er under 40 cm.

5.2.5 Main

Main funktionen for PSoC kodden initierer først alle komponenterne i projektet, I2C, SPI-slaven, motor komponenterne og stepper motor komponenterne. Derefter går den ind i et evigt for loop, hvor der den først henter data fra sonar sensoren. PSoC slår interrupt hver gang den modtager noget på SPI. Den variabel der modtages bruges i funktionen slaveBytesReceive(uint8), som bruger parameteren i et switch case. Afhængigt af værdien modtaget manøvrer bilen efter det. På figur 5.1 kan det ses hvilke funktioner der kaldes, alt efter hvilken værdi modtaget fra RPi. Alle værdier er defineret med variabelnavne, som der er repræsenteret med på diagrammet.

Flowchart for main

På figur 5.1 kan en flowchart model ses for hvordan de implementerede metoder bliver kaldt på PSoC over SPI via RPi.



Figur 5.1: Flowchart over Switch case for main.c på PSoC

5.3 PixyCam

Til anvendelse af PixyCam anvendes den implementerede **pixycam API**. Det er med API'en hvoraf dataen fra pixycam fås. Dataen fra pixycam sendes med USB protokollen på pixycam. Hvor denne data anvendes på RPi til at bestemme hvordan BoldBot styres.

Ved at oprette et objekt af klassen Pixy2 kan de forskellige funktioner i API'en anvendes. Her anvendes funktionerne init() og getVersion() direkte på Pixy2 objektet, hvor init funktionen tjekker om kommunikationen med modulet pixycam kan oprettes. Herudover anvendes funktionen pixy.ccc.getBlocks(). Dette kræver dog at man anvender ccc API, som kan kaldes da ccc er en member variabel til klassen Pixy2. ccc (color connected components) anvendes til at kunne kalde member funktionen getBlocks(). Det er denne funktion hvoraf pixycam detektere alle blokke i den sidst nye frame. de detekterede blokke gemmes i arrayet blocks, som er en member variabel i klassen ccc. Med ccc algorithmen gemmes x, y, højde og brede koordinanterne for den lærte signatur, hvilket i dette tilfælde er en tennisbold. Det er i member variablen blocks hvoraf den data, som anvendes på RPi ligger. Ved at bruge member variablen numBlocks i ccc, retunes antallet af elementer i member variablen blocks, hvorefter det er muligt iterativt at søge igennem blocks arrayet, og anvende de detekterede blocks der er gemt fra den sidst nye frame på modulet pixycam.

5.4 RPI

RPI klassen er selve styresystemet til BoldBot. Den sørger for at håndtere inputs fra PixyCam, og sende dem videre til motorstyringen allokeret på PSoC'en. Variablerne x, y, højde og bredde, indeholder data'en som klassen bruger til at styre BoldBotten mod tennisbolden. Det gøres ved at løbe igennem et for-loop, der tjekker på alle blocks i pixy.ccc.numBlocks, hvorefter x, y, højde og bredde bliver opdateret, med de korrekte værdier, for hver modtaget block. Inde i for-loopet ligger der seks if-sætninger, hver if-sætning sammenligner værdierne på de private variabler med et statisk defineret interger værdi. Alt efter hvilken værdi variabler indeholder, vil de gå ind i en af de seks if-sætninger, som så vil sende signal til motorstyringen om hvad der skal ske. Indeholder de private variabler x, y, højde og bredde derimod ikke nogle gyldige værdier, vil en if-loop der står for søgefunktionen gå i gang, indtil der igen modtages blocks fra PixyCam. Alt dette er implementeret i funktionen get_blocks(), som er en del af klassen BoldBot.

5.4.1 Main

Inde i selve mainen oprettes et objekt af klassen boldbot, hvorefter start() og init_sonar() kaldes. Dette initiere kommunikationen til PixyCam og sonarsensoren. Efterfølgende går main ind i en while(1) løkke, der køre så længe der er forbindelse til PixyCam. Inde i while-loopen, kaldes funktionerne get_blocks() og read_sonar() af det oprettede boldbot objekt. Dette styre BoldBotten mod det defineret mål, og udenom eventuelle forhindringer, så længe forbindelsen til PixyCam opretholdes.

5.5 InterfaceAPP

5.5.1 Indledning

I dette afsnit beskrives det, hvordan softwaren til websiden er skrevet og implementeret. Der vil kort blive redegjort for websidens overordnet interne funktionalitet, i de moduler den er delt op i. Disse moduler er også kortere beskrevet i arkitekturen, for websiden. Den vanskelige del af den interne funktionalitet for websiden, vil blive nærmere beskrevet i dette afsnit.

Selve websiden er implementeret ifht. Websocket protokollen, der tillader at lave en server og en client. Clienten opretter forbindelse til serveren, og sender og modtager beskeder til og fra serveren. Det gør det nemt at sende simpelt data, som en string eller interger. Websidens server og client er udviklet i JavaScript, og selve websocket forbindelsen er lavet med et open-source JavaScript run-time environment kaldet Node.js, samt JavaScript bilioteket Socket.IO. Node.js anvendes fordi det tillader websiden at køre scripts ved siden af serveren, dette tillader websiden at blive en mere dynamisk og interaktiv website, således at kan vi styre vores BoldBot derfra. Socket.IO bruges til kommunikationen imellem server og client. Den opnåede website kan ses i afsnit 6.12 på side 84 'Webside'

5.5.2 App.js

App.js er serveren bag websiden, den opretter en server via funktionen createServer(handler), der ligger i variablen app. App variablen sættes efterfølgende til at "lytte" på en bestemt port, i vores tilfælde port 5000. Http handler funktionen køres efterfølgende, og via variablen path, fs og url, er den i stand til at læse index.html og client.js filerne, og gøre dem en interaktiv del af serveren. Til kommunikationen imellem server og client, er der en funktion io.sockets.on(), der opretter forbindelse til websocket. Den lytter på clienten, via socket.on(), og kommer der så en besked fra clienten, som er enten stop eller start, vil serveren enten eksekvere BoldBot programmet, eller terminere det. Start af BoldBot programmet, sker ved at eksekvere output filen til BoldBot, som en sub process til selve serveren. Dette gøres med exec variablen. Stop af BoldBot programmet, sker når serveren modtager 'stop' fra klienten. Serveren finder frem til BoldBot programmets PID via funktionen ps.lookup(), hvor der efterfølgende bliver kaldt ps.kill(), der terminere processen, for det givne PID.

5.5.3 Client.js

Client.js fungere som client, til serveren. Den sørger for at behandle de inputs der kommer på selve websidens interface, og sende dem videre til serveren, der så agere ud fra det clienten har sendt. Clienten opretter forbindelse til den anvendte IP adresse, via. socket funktion io(), hvor den statiske IP adresse er defineret. Efter forbindelsen er sikret, oprettes to variabler, henholdsvis start og stop. Start bliver knyttet til elementet 'start', som er defineret i HTML filen, og agere som vores startknap. Stop bliver ligeledes tilknyttet 'stop', som er stop knappen. Efterfølgende bliver en EventListener tilføjet til start og stop variablerne, som gør det muligt for clienten at vide hvornår start knappen bliver trykket på, og hvornår stop knappen bliver trykket på. Eventlistener funktionen for start, og stop, indeholder hver især en besked, som sendes til serveren, når der klikkes på knappen, på websidens interface. Denne besked sendes via. socket.emit(), og serveren er så i stand til at reagere på den sendte besked.

5.5.4 Index.html

Index.html indeholder interfacet til websiden, der gør det muligt for brugeren at navigere rundt på en webbrowser, og kunne interagere med start og stop knappen på websiden. Der er ingen funktioner der aktivt gør noget i dette kode, kun to knapper der hver især er tildelt et id. Start er tildelt "start" som id, og stop er tildelt "stop". Index.html køre et enkelt script, som sørger for at opretholde forbindelse til clienten. Som stylesheet til HTML koden, er der anvendt **w3schools stylesheet** der gør det muligt at lave en simpel og pæn hjemmeside, på en meget enkelt måde.

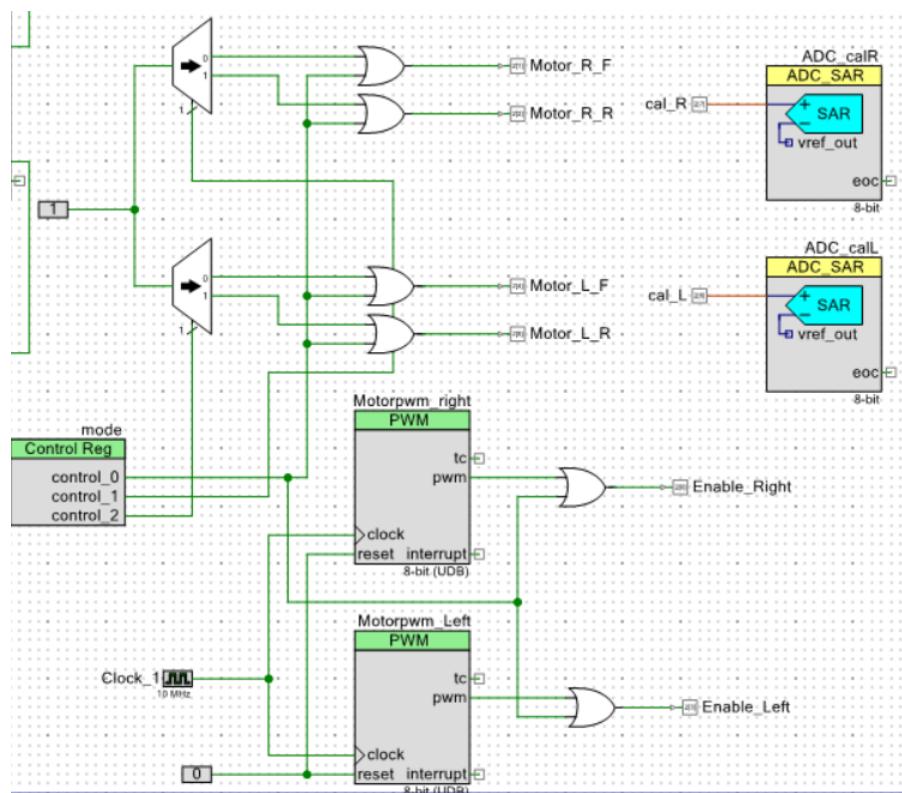
6. Modultest

6.0.1 Indledning

Denne sektion gennemgår modultest for de individuelle moduler i systemet. Der forklares hvad formålet med den specifikke test er, hvordan det er testet og til sidst hvad resultatet af testen er.

6.1 PSoC motorstyring

For at vælge hvorvidt motorerne skal roterer forlæns eller baglæns bruges topdesignet på figur 6.1.



Figur 6.1: Topdesign af motorstyring fra PSoC projekt.

Dette giver os følgende sandhedstabell til at vælge mellem hvilken retning motorerne rotere.

Resultater: (Osiloskop billede af diverse funktionerne, som bekræfter sandhedstabellen)

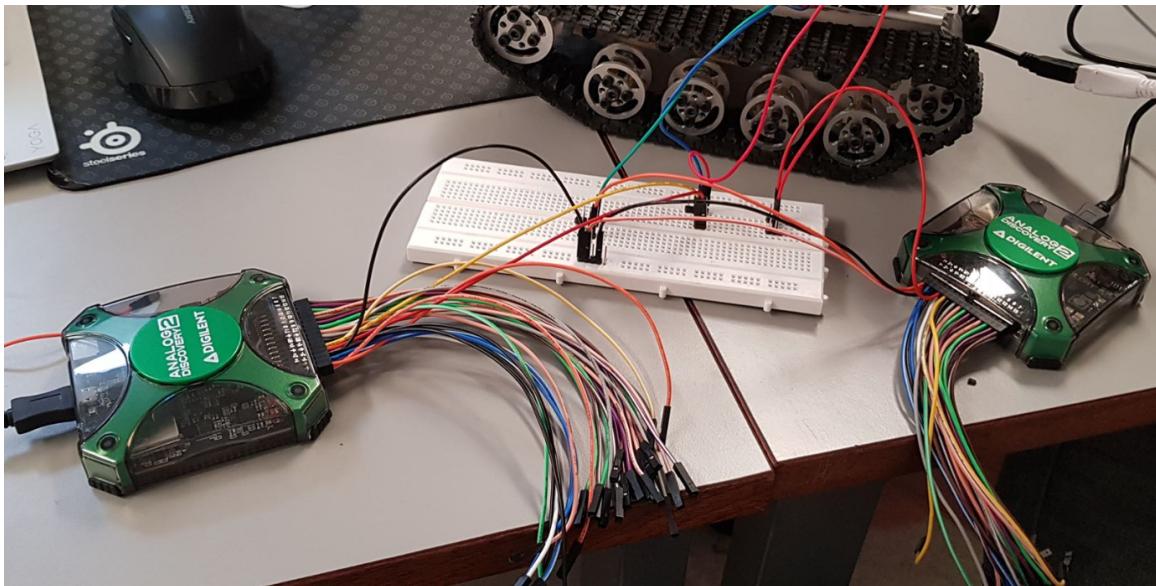
6.2 Levelconverter

Levelconverteren er testet, ved at tage en Analog Discovery på hver side af levelconverteren. Der først sendes et signal fra side B, der har en referencespænding på 3.3V.

Et oscilloskop er sat på A siden, der har en referencespænding på 5V. På figur 6.2 kan det verificeres at der kommer et 5V signal på oscilloskopet. Forsøgsopstillingen af dette forsøg kan ses på figur 6.3.



Figur 6.2: Screenshot af Analog Scope ved test af levelconverter



Figur 6.3: Forsøgsopstillingen ved test af levelconverter 3.3V og 5V

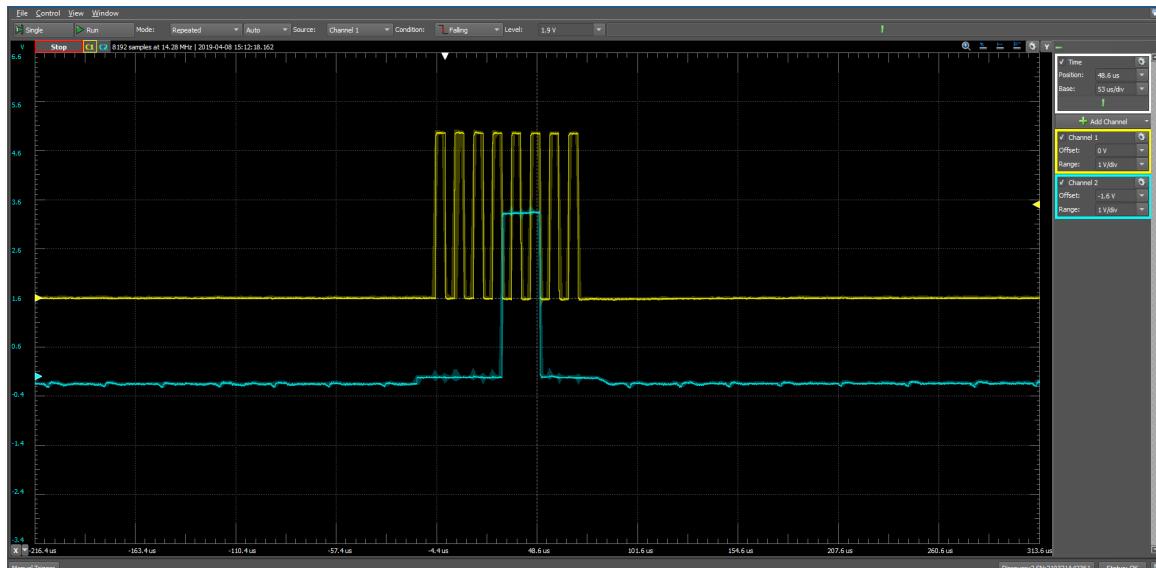
For at verificere at levelconverteren også virker fra 5V til 3.3V, er der også opsat en test som har funktionsgenerator på A siden og et oscilloskop på B. På figur 6.4 ses et oscilloskop billede, som verificerer at når der sendes et 5V signal ind på A siden, modtages et 3.3V signal på B siden.



Figur 6.4: Screenshot af Analog Scope ved test af levelconverter

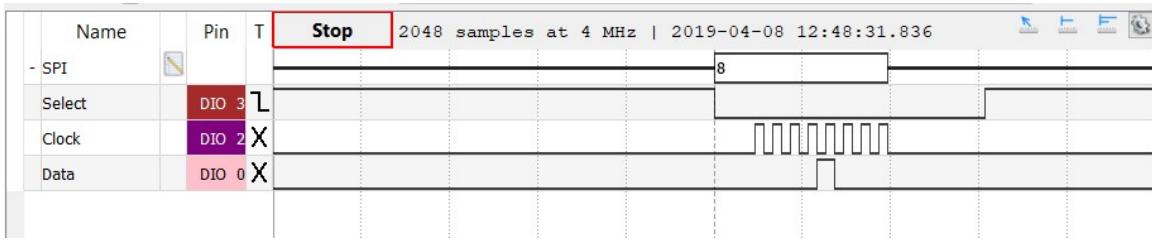
6.3 SPI Driver

Denne modultest har har til formål at teste, Rpi kan kommunikere med PSoC med SPI. Derfor er der blevet målt, om der bliver sendt det rigtige signal. For at teste dette er der blevet målt med Analog Discovery med både scope og Logic metoden. Billede 6.5 viser test af hvad Rpi sender over SPI.



Figur 6.5: Screenshot af Analog Scope

På de to figurer 6.6 og 6.7 under ses det at 8 modtages i Analog Discovery's logic analyzer, og at det også er 8 som er blevet sendt fra RPI.

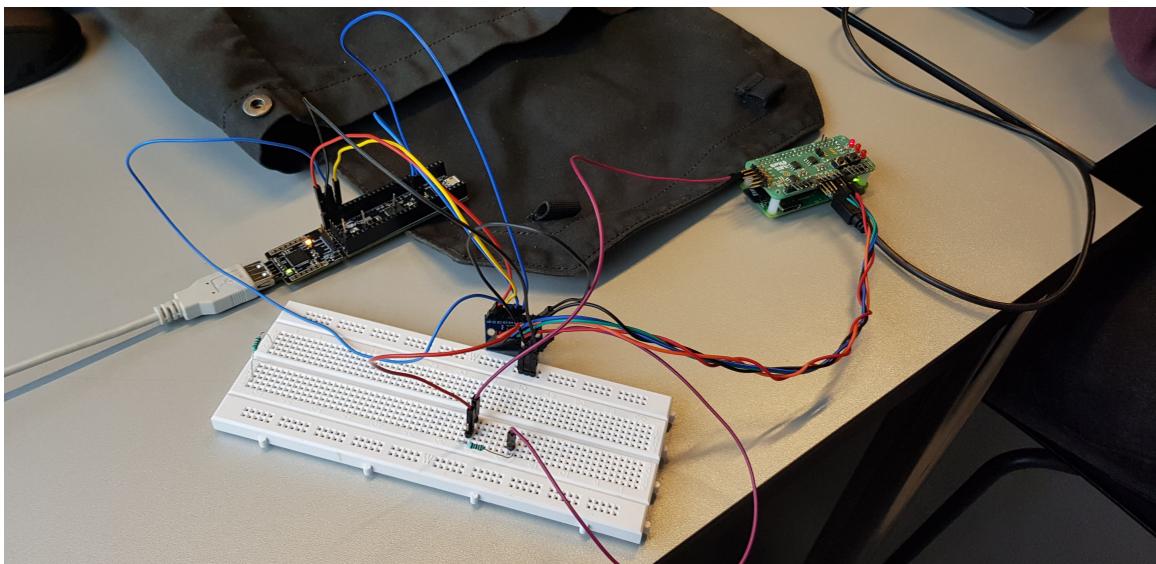


Figur 6.6: Logic analyzer af Analog Discovery

```
root@raspberrypi0-wifi:/dev# echo '8' > spi_drv0
```

Figur 6.7: Terminalvindue af RPI der, hvor der sendes '8'

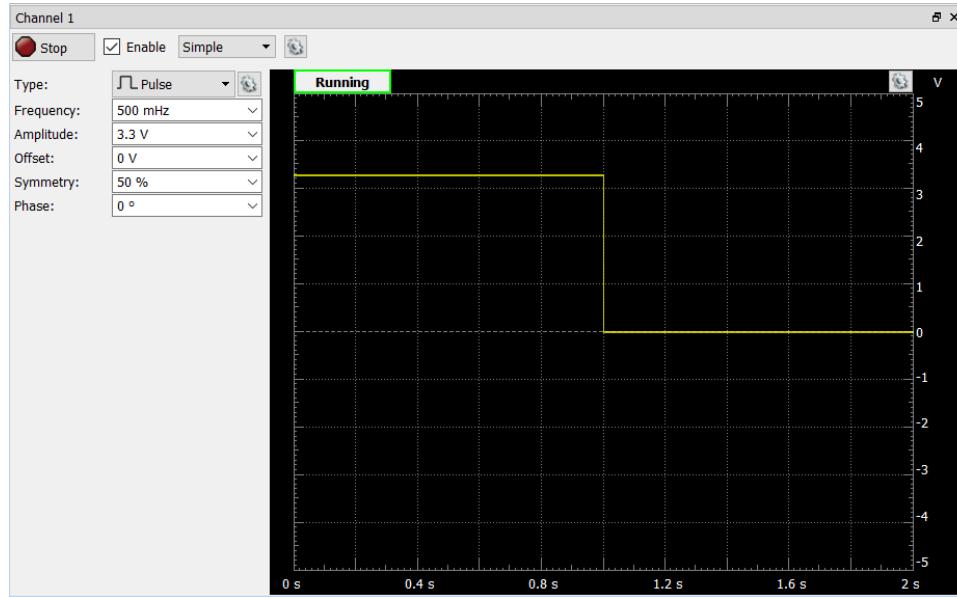
Billede 6.8 viser forsøgsopstillingen af modultesten. Der er under testen blevet gjort brug af et fumlebræt, levelconverter og Analog discovery.



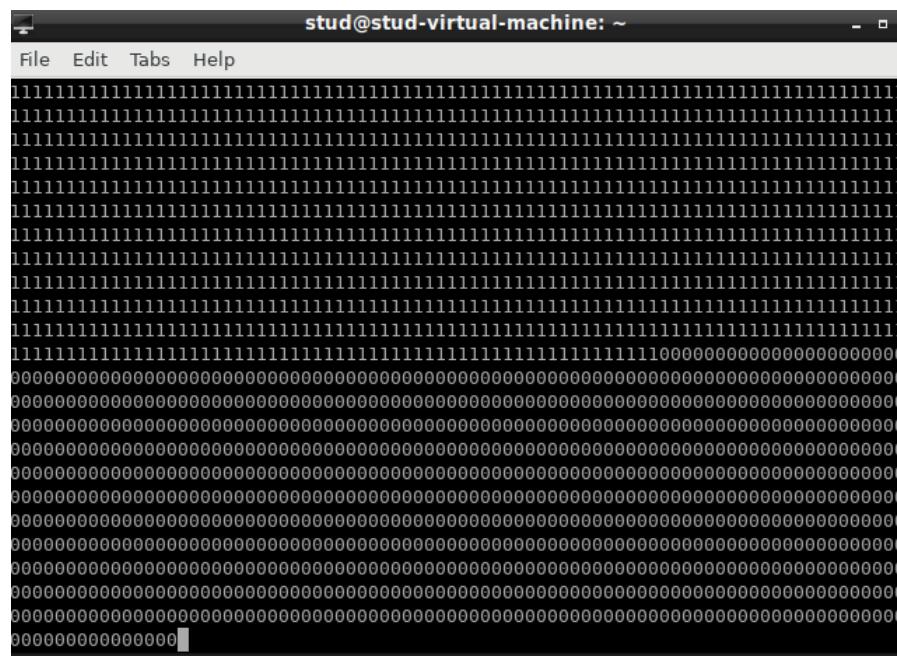
Figur 6.8: Billede af forsøgsopstilling til modultest af Rpi SPI kommunikation

6.4 GPIO driver

Denne test har haft til formål at teste hvorvidt GPIO driveren, gør det muligt for RPi og modtage højt eller lavt signal på GPIO pin 17. Testen er udført ved at sætte en Analog Discovery til GPIO 17, hvorefter der læses fra pinnen i terminalen. Der sættes så skiftesvis højt eller lavt signal ind på GPIO 17, og det som læses på RPi verificeres. Der sendes fra Analog på Discovery et puls signal ud op 0,5 Hz, dette kan ses på figur 6.9, og på figur 6.10 ses det hvad der læses på RPi i terminalvinduet. Som det vises på figurerne, bliver det som sendes ud fra Analog Discovery også modtaget korrekt på RPi.

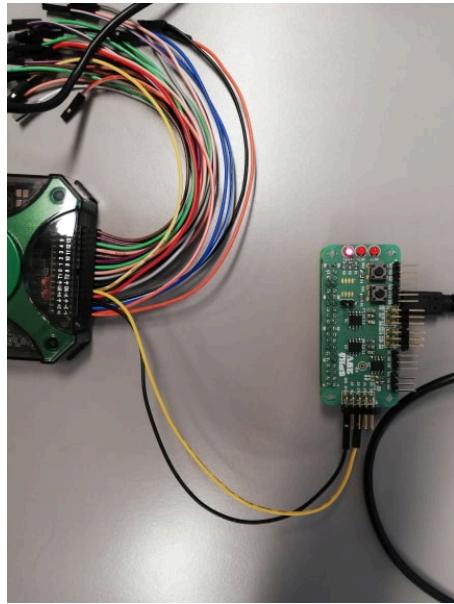


Figur 6.9: Screenshot af hvad der sendes ud fra Analog Discovery.



Figur 6.10: Terminalvindue af RPi, hvor der modtages et højt signal og lavt signal fra RPi

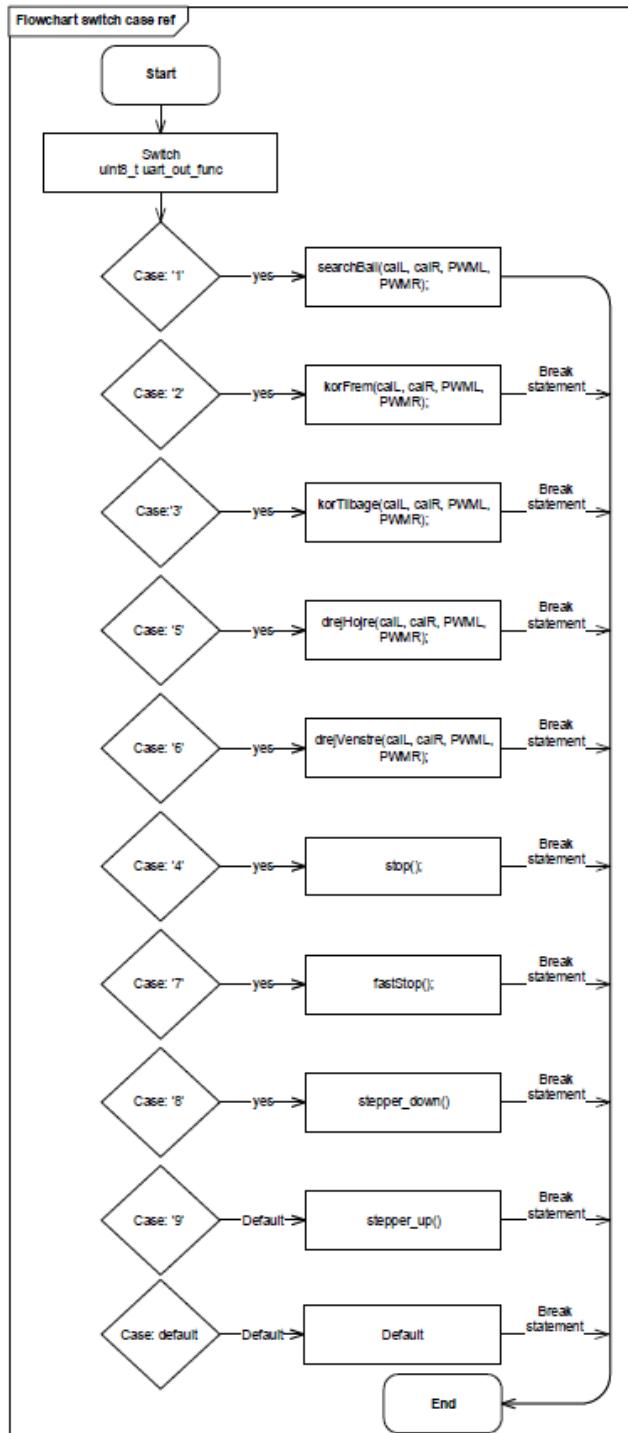
På figur 6.11 ses forsøgsopstillingen, hvor Analog Discovery opkoblet til RPi GPIO 17. Det gule kabel er funktionsgenerator, og det sorte kabel er ground.



Figur 6.11: Forsøgsopstilling til modultest af RPi GPIO driver.

6.5 PSoC software

Til denne modultest er formålet at teste om PSoC går i det rigtige switch case, afhængigt af det modtagede signal. Dette er testet ved at opsætte UART i PSoC koden, så i stedet for at når den modtager en værdi på SPI, så er switch casene afhængige af det som modtages over UART. Sammenhængen i dette switch case er opstillet i figur 6.12.



Figur 6.12: Flowchart der beskriver sammenhængen i UART switch case

6.6 Sonarsensor

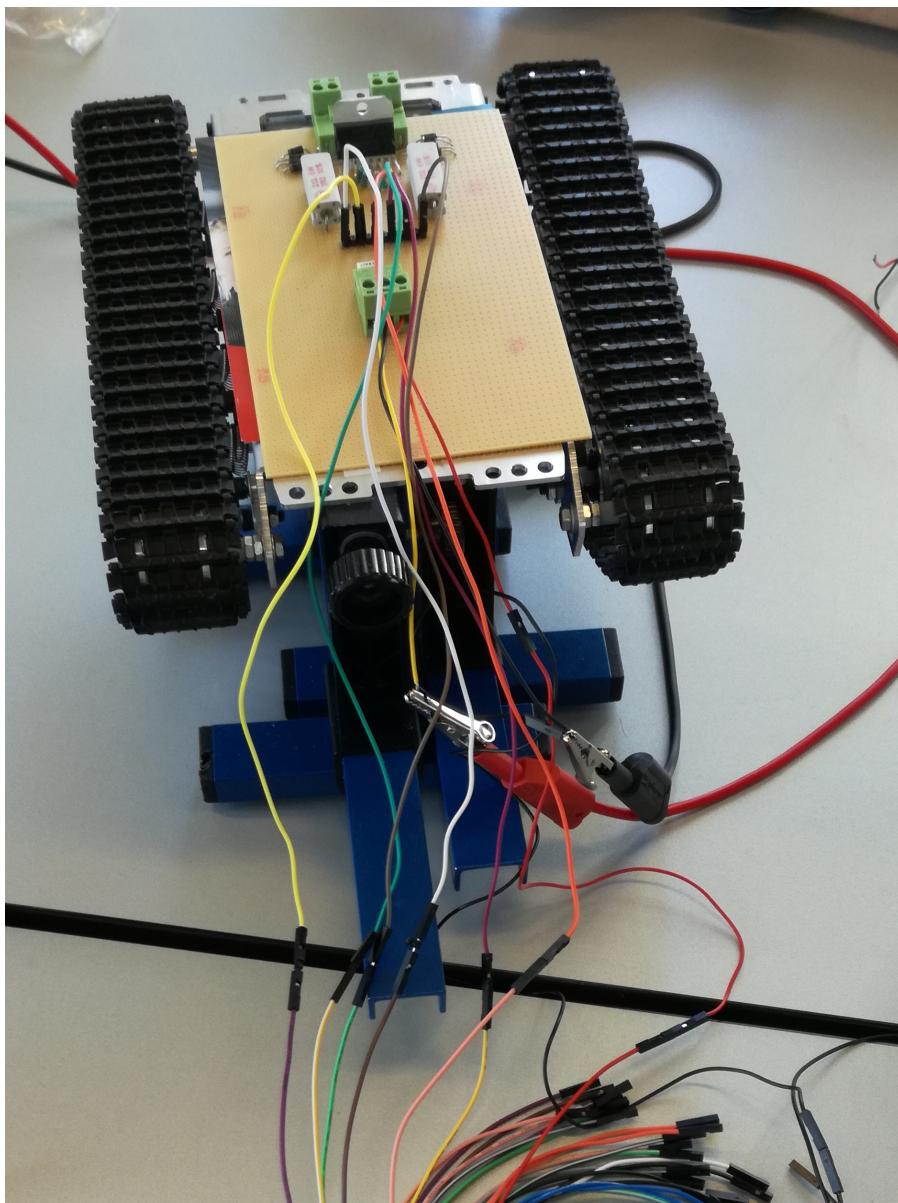
6.7 Motormodul

6.7.1 Opstilling

For at teste funktionaliteten af Motormodul bruges en testopstilling, hvor følgende indgår:

- Motormodul
- Digilent Analog Discovery 2
- Strømforsyning der kan levere min. 12 V

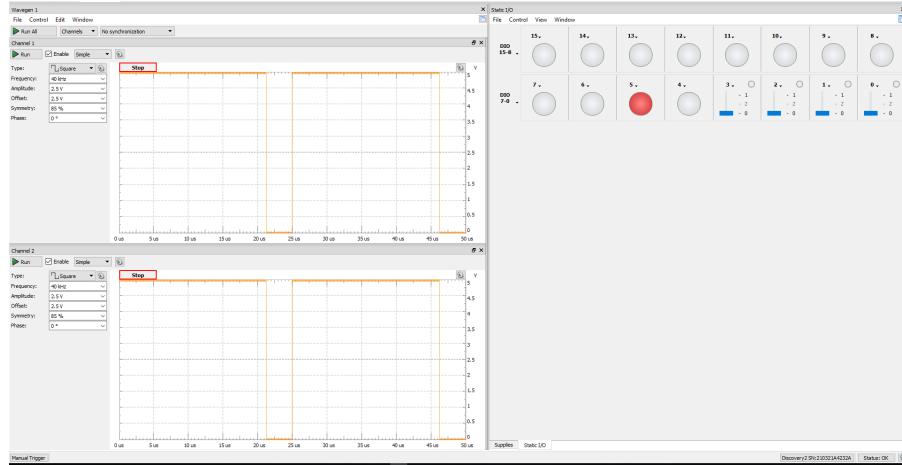
Selve opstillingen for testen ses på Figur 6.13. Motormodulet bliver tilsluttet Analog Discovery. I alt 6 ledninger bliver tilsluttede de 6 harwin pins på modulet. 2 af ledningen bliver tilsluttet Motor_PWM indgangen, og de 4 sidste bliver tilsluttet Motor_input. Derudover bliver der også tilsluttede en strømforsyning på 9.6 V for at simulere et batteri. Herudover bliver motormodulet også forsynet med 5 V fra Analogen.



Figur 6.13: Billede af Motor test opsætning .

Opsætningen af Wavegen til at styre Analog Discovery kan ses på Figur 6.14. Her kan det ses at sendes to PWM signaler ind på printet, disse PWM signaler er til styre

hastigheden at de to motorer. Der kan også ses 4 static IO's, disse 4 bliver brugt til at kontrollere motorens retning, samt om de køre eller aktivt bremser.



Figur 6.14: Billede af Motor test opsætning .

Testen fortages ved at optrykke modulet med to PWM signaler, og herefter toggle de fire motor_input på skift Herefter justeres dutycykelen med motorene tænt. Figur 6.15 viser de forskellige inputs på motorstyringen. Dette er kun den ene motor, og der er derfor to af hver input. V_en er PWM inputet, C og D er Motor_input.

Inputs		Function
$V_{en} = H$	$C = H ; D = L$	Forward
	$C = L ; D = H$	Reverse
	$C = D$	Fast Motor Stop
$V_{en} = L$	$C = X ; D = X$	Free Running Motor Stop

L = Low

H = High

X = Don't care

Figur 6.15: Tabel over inputs og funktion. (**L298_H_Bridge**)

6.7.2 Forventet resultat

Det forventes at motorstyringen kan kontrollere motorens fart og retning.

6.7.3 Faktiske resultat

Det er muligt at kontrollere farten og retningen på begge motorer

6.7.4 konklusion

Det kan konkluderes at der er muligt at styre motorenes fart og retning.

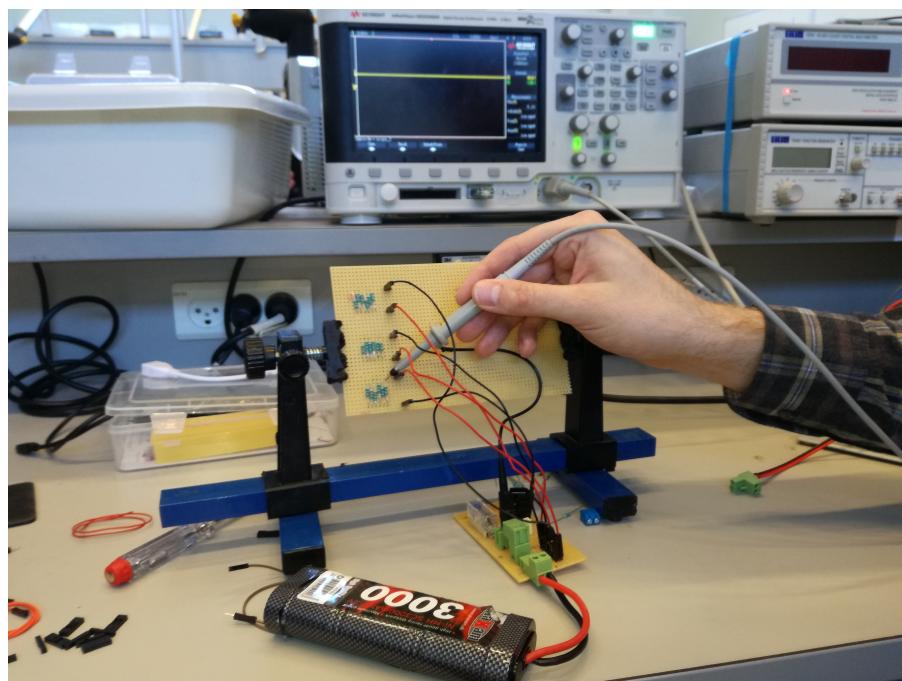
6.8 Spændingsregulator

6.8.1 Opstilling

For at teste funktionaliteten af spændingsregulatoren bruges en testopstilling, hvor følgende indgår:

- spændingsregulatoren
- Strømforsyning der kan levere min. 10 V
- Voltmeter der kan måle min. op til 5 V
- 3 effektmodstande der kan holde til min. 1.5 Watt

Selve opstillingen for testen ses på Figur 6.16. Her kan der ses at printet tilsluttes et 7.2 V batteri samt 3 effektmodstande. Disse effektmodstande er til for at emulere 3 komponenter der hver især har et strømtræk på 300 mA. Derved ville det samlede strømtræk være på 900 mA.



Figur 6.16: Måleopstilling for test af spændingsregulator

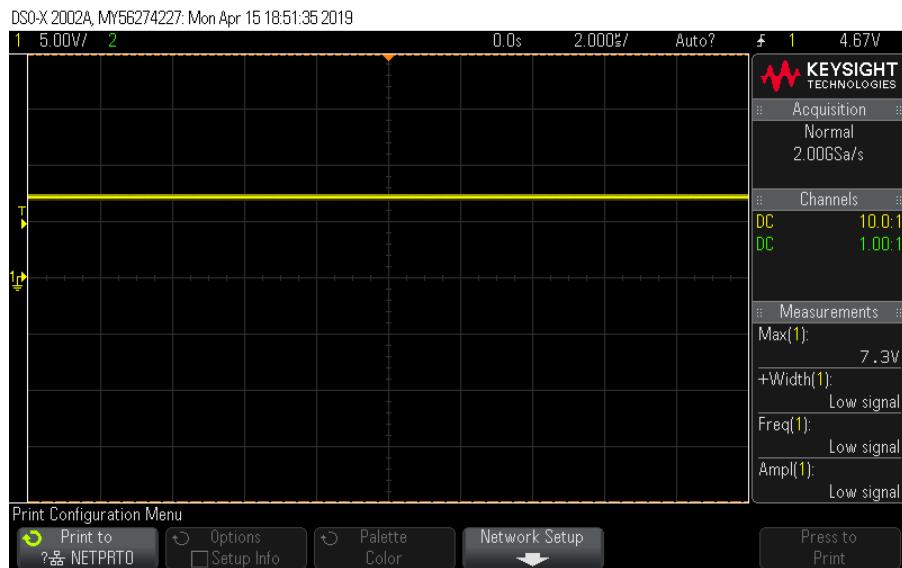
Testen fortages ved at måle spænding på indgangen og på udgangen med et Voltmeter. Herefter sammenlignes det med det forventede resultat.

6.8.2 Forventet resultat

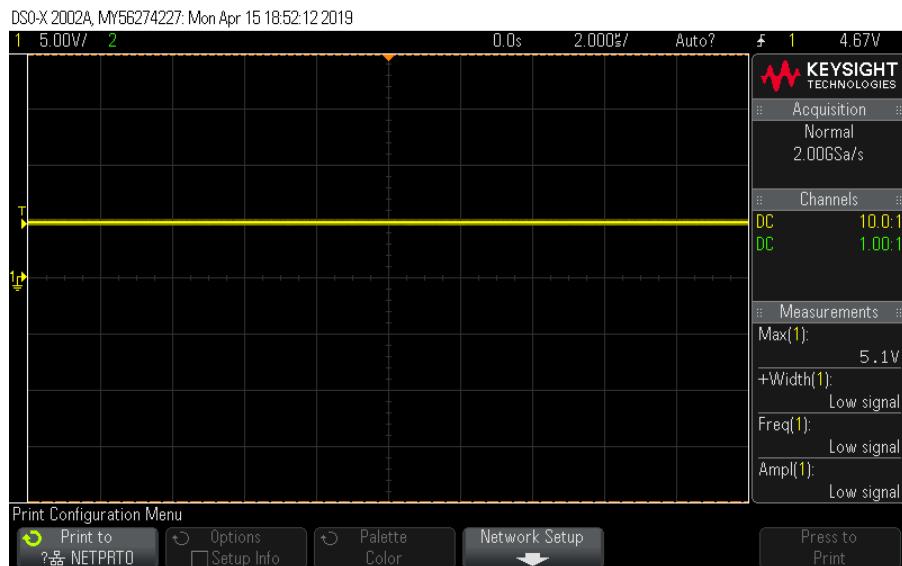
Spændingsregulatoren skal, ved en indgangsspænding på ca. 7,2 V, kunne levere en udgangsspænding på ca. 5 V.

6.8.3 Faktiske resultater

En spænding på 7,3 V som på Figur 6.17 sendes ind på indgangen af spændingsregulatoren og den resulterende udgangssspænding ses på Figur 6.18.



Figur 6.17: Måling af spændingen på indgangen af spændingsregulatormodulet



Figur 6.18: Måling af spændingen på udgangen af spændingsregulatormodulet

6.8.4 Konklusion

Spændingsregulatoren er altså lykkedes med at regulere 7,3 V ned til 5,1 V og virker som forventet.

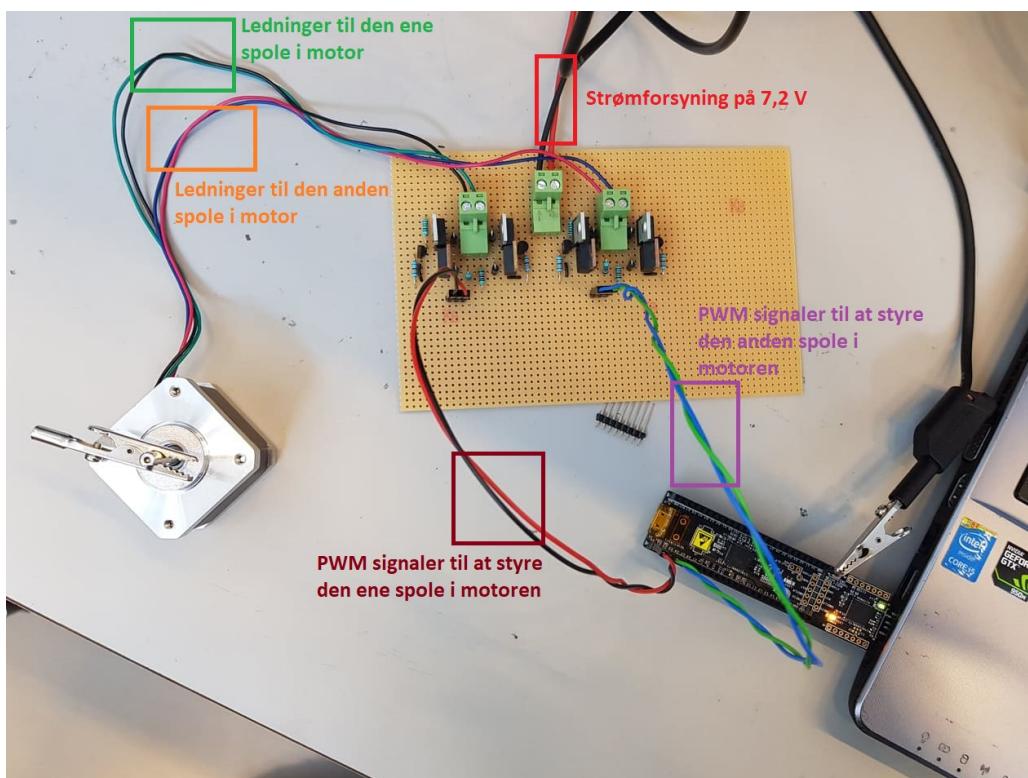
6.9 Opsamlingsmodul

6.9.1 Opstilling

For at teste funktionaliteten af opsamlingsmodulet bruges en testopstilling, hvor følgende indgår:

- Opsamlingsmodul
- PSoC med tilhørende testkode
- Strømforsyning der kan levere min. 10 V og min. 2 A

Selve opstillingen for testen ses på Figur 6.19. Testkoden på PSoC'en står for at skifte mellem at sende strøm gennem spolerne, men også skifte polaritet på spolerne i steppermotoren. Dette gøres ved at switche strøm fra strømforsyningen gennem den dobbelte H-bro.



Figur 6.19: Opstilling til modultest af opsamlingsmodul

6.9.2 Forventet resultat

Det forventes af testen, at steppermotoren tager et antal step der svarer til ca. 90° , dette observeres visuelt.

6.9.3 Faktiske resultater

Steppermotoren formår at tage et antal step der svarer til at den drejer 90° .

6.9.4 Konklusion

Det er lykkedes vha. et stepperdriver print at styre steppermotoren som ønsket til formålet.

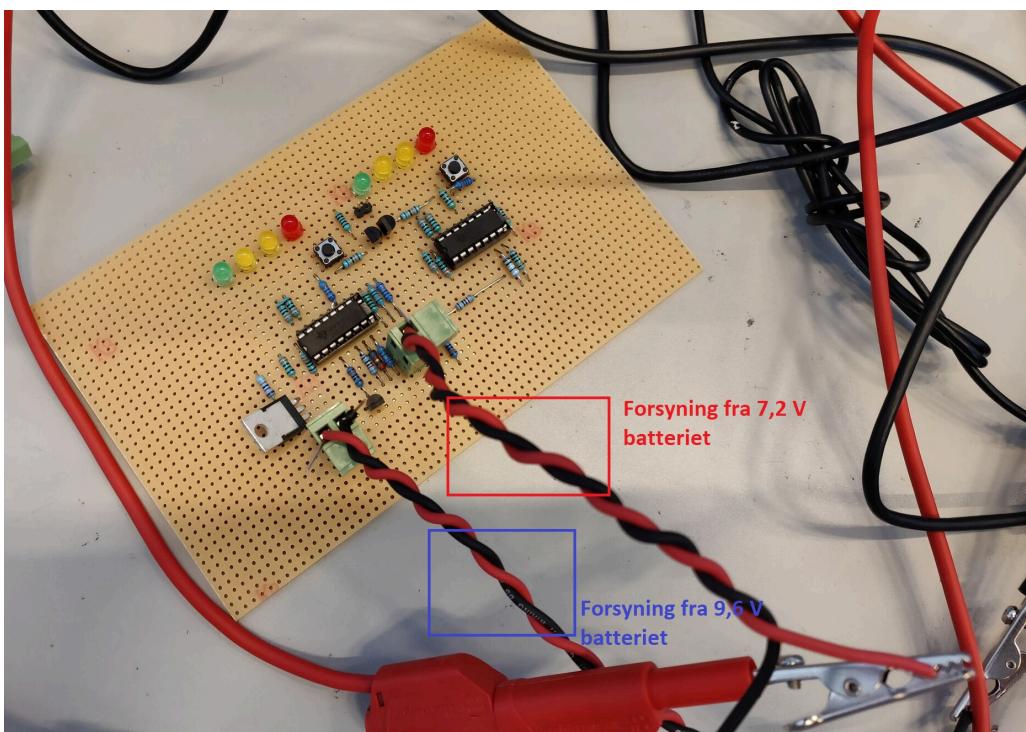
6.10 Batteriindikator

6.10.1 Opstilling

For at teste funktionaliteten af batteriindikatoren bruges en testopstilling, hvor følgende indgår:

- Strømforsyning der kan levere min. 12 V og min. 100 mA
- Batteriindikatorprint
- Voltmeter der kan måle min. op til 12 V

Selve opstillingen for testen ses på Figur 6.20.



Figur 6.20: Opstilling til modultest af batteriindikator

7.2 V batteriet testes først ved at sende en spænding på over 7.765 V på indgangen af batteriindikatoren, hvor 7.2 V batteriet skal tilkobles. Denne spænding sænkes så langsomt og tact-switchen der tilhører LED'erne til 7.2 V indikatoren holdes inde. Dernæst

observeres hvad der sker med LED'erne.

På samme måde testes 9.6 V batteriindikatoren, med den anden indgang - men her startes ved en spænding på 9.16 V hvorefter den sænkes på samme måde.

6.10.2 Forventet resultat

De to nedenstående tabeller viser de forventede resultater ved givne spændingsniveauer for hhv. 7.2 V batteriet og 9.6 V batteriet.

Tabel 6.1: Udregnede spændingsniveauer for 7.2 V batteri

7.2 volt batteri	
7.76551 V	4 LED'er lyser
7.73806 V	3 LED'er lyser
7.6516 V	2 LED'er lyser
7.37023 V	1 LED'er lyser
<7.37023 V	Ingen LED'er lyser

Tabel 6.2: Udregnede spændingsniveauer for 9.6 V batteri

9.6 volt batteri	
>9.1699 V	4 LED'er lyser
8.2504 V	3 LED'er lyser
8.0056 V	2 LED'er lyser
7.7962 V	1 LED'er lyser
<7.7962 V	Ingen LED'er lyser

6.10.3 Faktiske resultat

Nedenstående to tabeller for 7.2 V batteriet viser testobservationer for modultest af batteriindikatoren. Øverste tabel viser hvad der observeres når de udregnede spændingsniveauer for 7.2 V batteriet sendes ind på 7.2 V indgangen på batteriet. Nederste tabel viser hvad de faktiske spændingsniveauer ligger på. Det samme gør sig gældende for tabellerne for 9.6 V batteriet.

Tabel 6.3: Testobservationer for udregnede niveauer test af 7.2 V batteriindikator

7.2 volt batteri	
7.77 V	Ingen LED'er lyser
7.73 V	Ingen LED'er lyser
7.65 V	Ingen LED'er lyser
7.37 V	Ingen LED'er lyser
<7.37023 V	Ingen LED'er lyser

Tabel 6.4: Testobservationer for test af 7.2 V batteriindikator

7.2 volt batteri	
>8.432 V	4 LED'er lyser
Ikke testbar	3 LED'er lyser
8.418 V	2 LED'er lyser
8.307 V	1 LED'er lyser
7.99 V	Ingen LED lyser

Tabel 6.5: Testobservationer for test af 9.6 V batteriindikator

7.2 volt batteri	
>9.166 V	4 LED'er lyser
8.248 V	3 LED'er lyser
8.00 V	3 LED'er lyser
7.79 V	3 LED'er lyser
<7.79 V	Ikke testet

Tabel 6.6: Testobservationer for test af 9.6 V batteriindikator

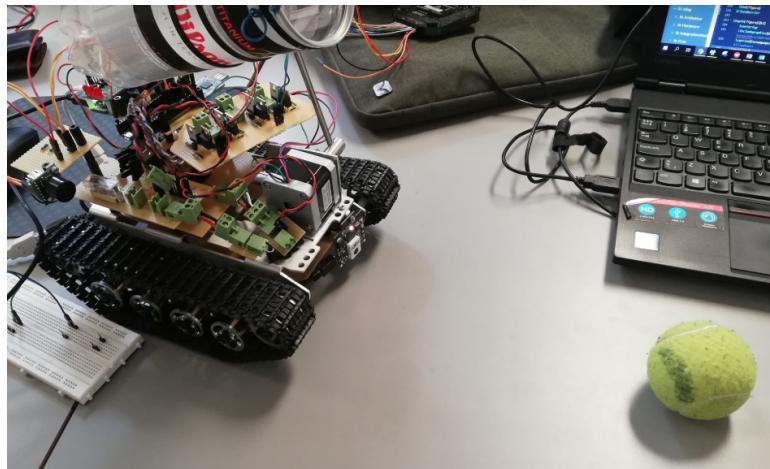
9.6 volt batteri	
>8.3 V	4 LED'er lyser
8.209 V	3 LED'er lyser
7.518 V	2 LED'er lyser
7.291 V	1 LED'er lyser
4.184 V	Ingen LED'er lyser

6.10.4 Konklusion

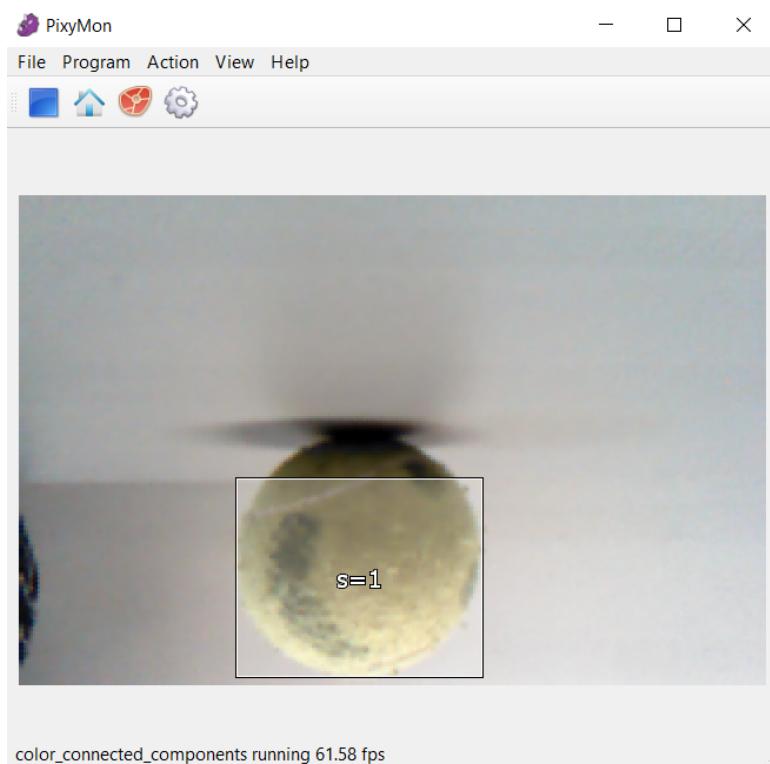
Det er ikke lykkedes at lave en fungerende batteriindikator, som kan vise batteriniveauerne for batterierne med en nogenlunde nøjagtighed. Multisim simuleringen fungerer efter de udregnede værdier, men der har været en del fejlfinding på batteriindikatorprintet, så der er blevet rettet en del fejl. Men pga. tidspres bliver der ikke testet mere på batteriindikatoren.

6.11 PixyCam

Formålet med denne test er at teste om der kan oprettes forbindelse til PixyCam, via en USB forbindelse, samt teste om PixyCam kan registrere en tennisbold, ud fra den indstillet signatur på PixyCam programmet PixyMon. Testen udføres ved at sætte et USB kabel fra PixyCam til en PC, og placere en tennisbold foran PixyCam. PixyMon åbnes på PC'en der er forbundet til PixyCam, og der observeres om der er oprettet forbindelse til PixyCam, og om tennisbolden vises som signatur. På figur 6.21 ses forsøgsopstillingen, hvor en bold er placeret foran BoldBot. Figur 6.22 viser PixyMon softwaren, der har oprettet forbindelse til PixyCam og viser tennisbolden som en signatur.



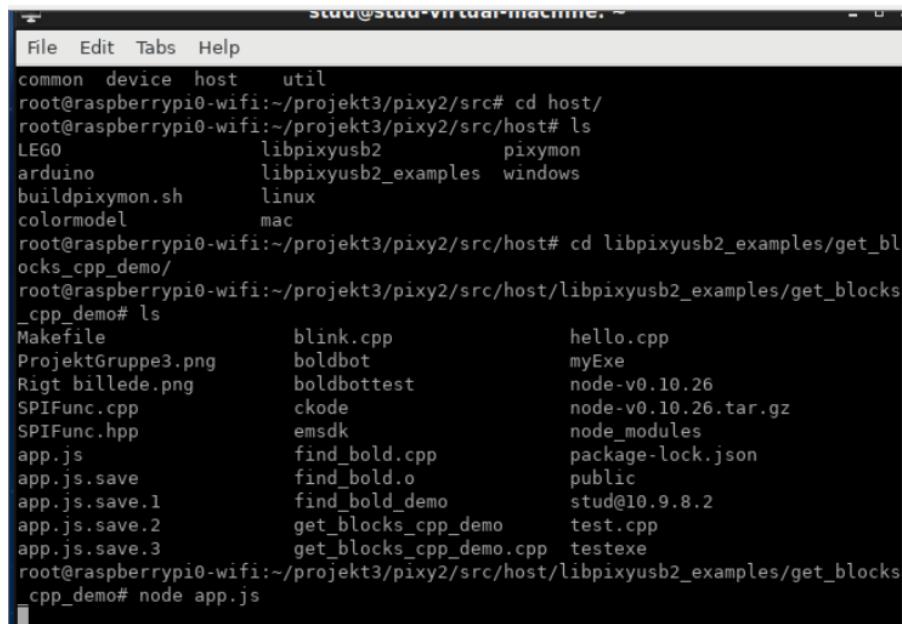
Figur 6.21: Forsøgsopstilling til modultest af PixyCam.



Figur 6.22: PixyCam synsfelt på PixyMon programmet. Med registreret objekt.

6.12 Webside

Formålet med denne test er, at dokumentere om websiden virker efter hensigten. Først bliver der testet, om 'app.js' kan blive startet korrekt (se figur 6.23).



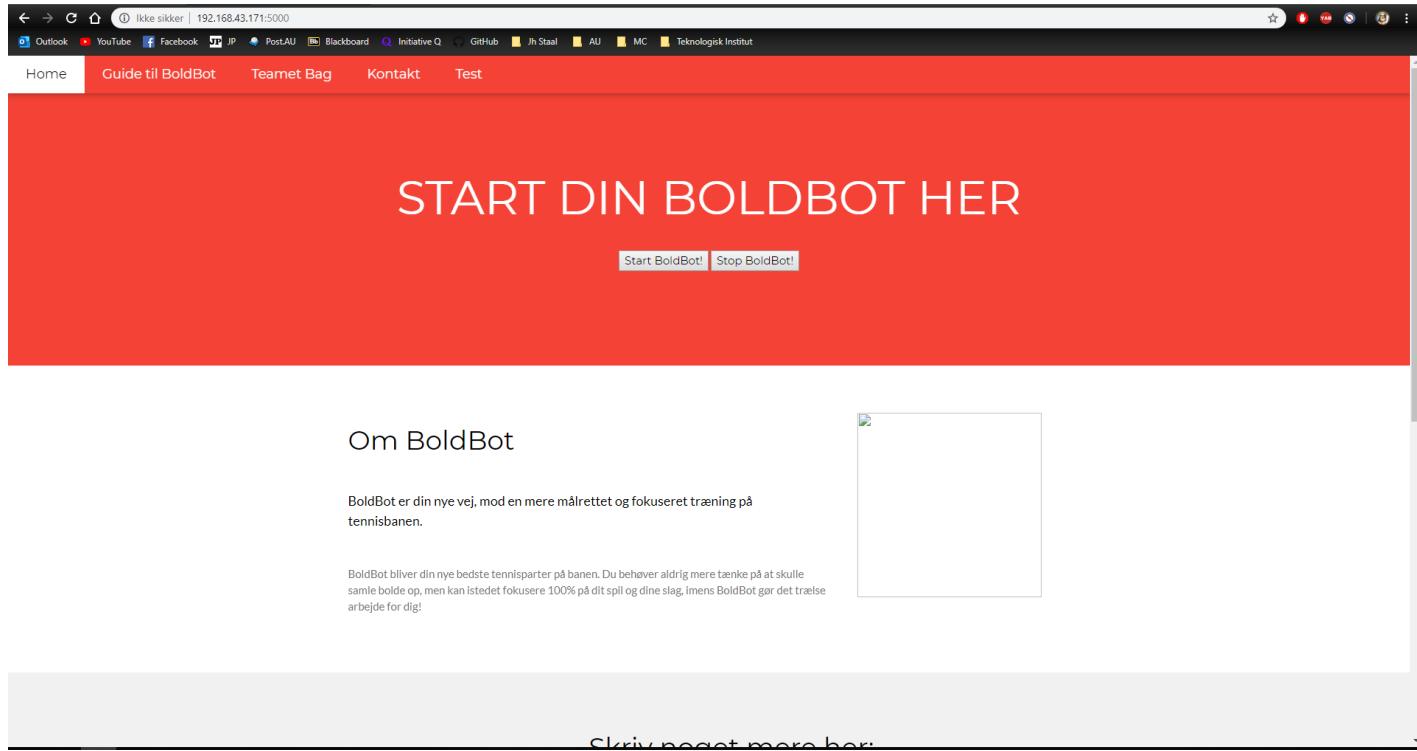
```

File Edit Tabs Help
common device host util
root@raspberrypi0-wifi:~/projekt3/pixy2/src# cd host/
root@raspberrypi0-wifi:~/projekt3/pixy2/src/host# ls
LEGO           libpixyusb2      pixymon
arduino        libpixyusb2_examples windows
buildpixymon.sh    linux
colormodel      mac
root@raspberrypi0-wifi:~/projekt3/pixy2/src/host# cd libpixyusb2_examples/get_blocks
blocks_cpp_demo/
root@raspberrypi0-wifi:~/projekt3/pixy2/src/host/libpixyusb2_examples/get_blocks
_blocks_cpp_demo# ls
Makefile          blink.cpp      hello.cpp
ProjektGruppe3.png boldbot       myExe
Rigt billede.png boldbottest   node-v0.10.26
SPIFunc.cpp       ckode         node-v0.10.26.tar.gz
SPIFunc.hpp      emsdk         node_modules
app.js           find_bold.cpp package-lock.json
app.js.save      find_bold.o   public
app.js.save.1    find_bold_demo stud@10.9.8.2
app.js.save.2    get_blocks_cpp_demo test.cpp
app.js.save.3    get_blocks_cpp_demo.cpp testexe
root@raspberrypi0-wifi:~/projekt3/pixy2/src/host/libpixyusb2_examples/get_blocks
_blocks_cpp_demo# node app.js

```

Figur 6.23: Billede af terminalvindue, hvor 'app.js' bliver kørt.

Når clienten køres, bliver serveren oprettet, som herefter kan tilgåes vha. en webbrowser. Vi har fået adgang til den oprettet client vha. Google Chrome, heri bliver ip-adressen skrevet (<http://192.168.43.171:5000/>). Et billede af den åbne webside kan ses på figur 6.24.



Figur 6.24: Billede af webside, med start og stop knapper

Efter websiden er blevet åbnet, er de to funktioner på websiden blevet testet. Først

bliver 'Start BoldBot!' funktionen testet, hvorefter 'Stop BoldBot!' bliver testet. Dokumentation for testen kan ses på figur 6.25. Her kan det ses, at forbindelsen mellem clienen og serveren har forbindelse.



Figur 6.25: Billede af test for 'Start BoldBot!' og 'Stop BoldBot!' funktionerne.

På figur 6.26 kan dokumentationen for handlingerne fra figur 6.25 ses. 'app.js' kører som det skal, hvorefter det kan ses, at først bliver der printet 'det virker' ud på terminalen, når brugeren trykker på 'Start BoldBot!'. Herefter bliver der trykket på 'Stop BoldBot!', hvorefter programmet terminerer succesfuldt.

```
root@raspberrypi0-wifi:~/projekt3/pixy2/src/host/libpixyusb2_examples/get_blocks_cpp_demo# node app.js
det virker
Program stopper
PID: 251, COMMAND: /home/root/projekt3/pixy2/src/host/libpixyusb2_examples/get_blocks_cpp_demo/boldbot, ARGUMENTS:
Dette er PID for programmet der er igang: 251
{ Error: Command failed: /home/root/projekt3/pixy2/src/host/libpixyusb2_examples/get_blocks_cpp_demo/boldbot

    at ChildProcess.exithandler (child_process.js:276:12)
    at emitTwo (events.js:126:13)
    at ChildProcess.emit (events.js:214:7)
    at maybeClose (internal/child_process.js:915:16)
    at Socket.stream.socket.on (internal/child_process.js:336:11)
    at emitOne (events.js:116:13)
    at Socket.emit (events.js:211:7)
    at Pipe._handle.close [as _onclose] (net.js:561:12)
  killed: false,
  code: null,
  signal: 'SIGKILL',
  cmd: '/home/root/projekt3/pixy2/src/host/libpixyusb2_examples/get_blocks_cpp_d
```

Figur 6.26: Billede af terminalvinduet på RPi, som viser at programmet først starter og herefter terminerer.

6.13 RPi software

Nedenstående afsnit vil gennemgå test af softwaren allokeret på Rpi. Dette betyder en test af klassen Boldbot, og hvordan denne håndtere data sendt fra PixyCam modulet. Testen udføres ved Pixy kobles til RPi data stik, med USB tilslutning. Testprogram opstartes på RPi. En tennisbold placeres foran PixyCam og rykkes på, for at se om koordinaterne ændrer sig korrekt.

Test af funktionen start()

Afsnittet gennemgår en test af funktionen start(). Her tjekkes det om PixyCam initieres korrekt.

Precondition: Programmet eksekveres.

Postcondition: Programmet udskriver "BoldBot starter." Ellers udskrives en fejl besked.

Nedenstående figur ?? viser resultatet testet for funktionen start().

```
stud@stud-virtual-machine:~/pixy2/src/host/libpixyusb2_examples/get_blocks_cpp_demo$ ./boldbot
Test af start funktion og init funktionen til pixy!
=====
= BoldBot Start!                               =
=====
Connecting to Pixy2...Success
hardware ver: 0x0 firmware ver: 0.0.0 general
```

Figur 6.27: Resultater af Start() funktion

Test af funktionen getBlocks()

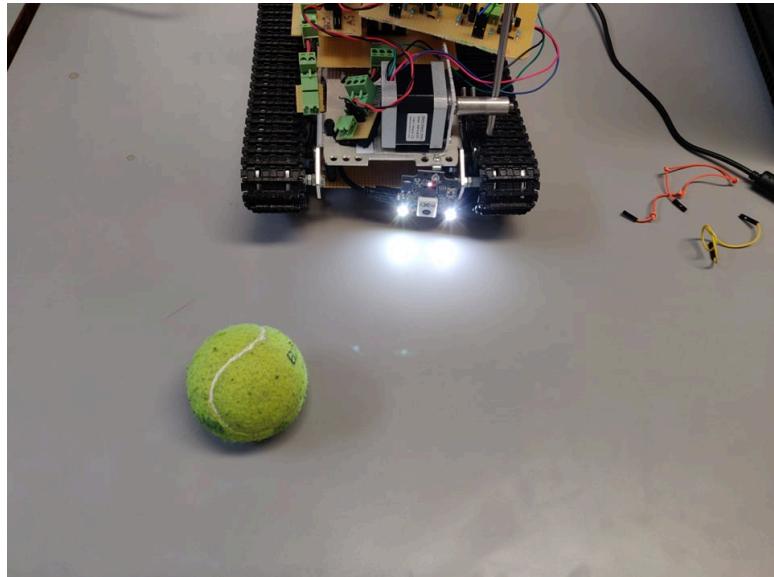
Afsnittet gennemgår en test af funktionen getBlocks() på baggrund af den sendte data fra PixyCam modulet. Her tjekkes der på de forskellige cases for de forventet resultat for den sendte data fra PixyCam.

Case DrejHøjre

Precondition: PixyCam er initieret, og tennisbolden befinner sig indenfor værdierne: ($x < 115$).

Postcondition: Rpi programmet udskriver "drej til højre."

Nedenstående figur 6.28 viser opstilling for testen af case DrejHøjre.



Figur 6.28: Opstilling af test DrejHøjre med tennisboldens position i forhold til PixyCam

Nedenstående figur 6.29 viser resultatet for case DrejHøjre.

```
X er mindre end 70, kør mod højre!
X: 23, Højde: 64, Bredde 46
X er mindre end 70, kør mod højre!
X: 23, Højde: 41, Bredde 46
X er mindre end 70, kør mod højre!
X: 23, Højde: 62, Bredde 46
X er mindre end 70, kør mod højre!
X: 23, Højde: 59, Bredde 46
X er mindre end 70, kør mod højre!
X: 23, Højde: 65, Bredde 46
X er mindre end 70, kør mod højre!
X: 23, Højde: 66, Bredde 46
X er mindre end 70, kør mod højre!
X: 23, Højde: 70, Bredde 46
X er mindre end 70, kør mod højre!
```

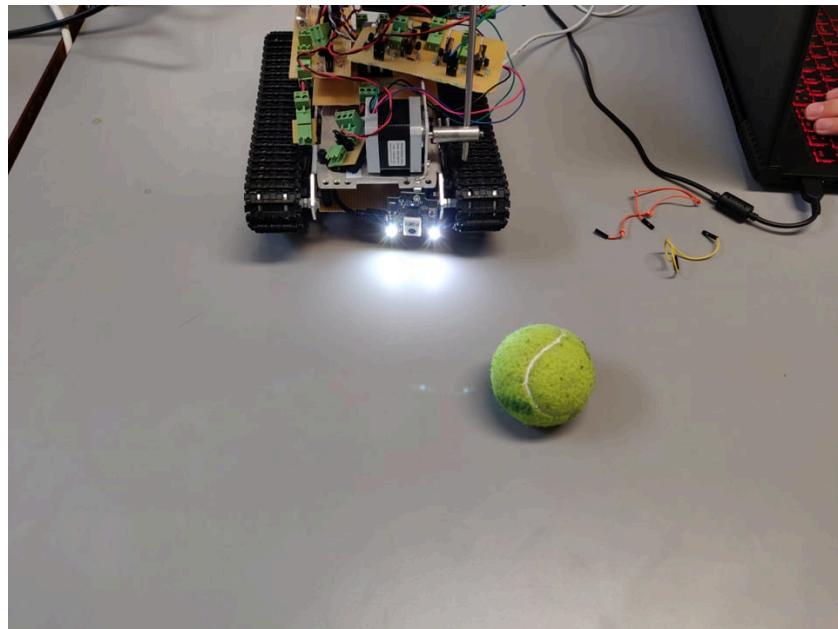
Figur 6.29: Resultater for DrejHøjre test

Case DrejVenstre

Precondition: PixyCam er initieret, og tennisbolden befinner sig indenfor værdierne: ($x > 185$).

Postcondition: Rpi programmet udskriver "drej til venstre."

Nedenstående figur 6.30 viser opstilling for testen af case DrejVenstre.



Figur 6.30: Opstilling af test DrejVenstre med tennisboldens position i forhold til PixyCam

Nedenstående figur 6.31 viser resultatet for case DrejVenstre.

```
X er større 250, kør mod vestre!
X: 304, Højde: 5, Bredde 20
X er større 250, kør mod vestre!
X: 303, Højde: 2, Bredde 26
X er større 250, kør mod vestre!
X: 309, Højde: 3, Bredde 14
X er større 250, kør mod vestre!
X: 308, Højde: 4, Bredde 16
X er større 250, kør mod vestre!
X: 299, Højde: 3, Bredde 14
X er større 250, kør mod vestre!
```

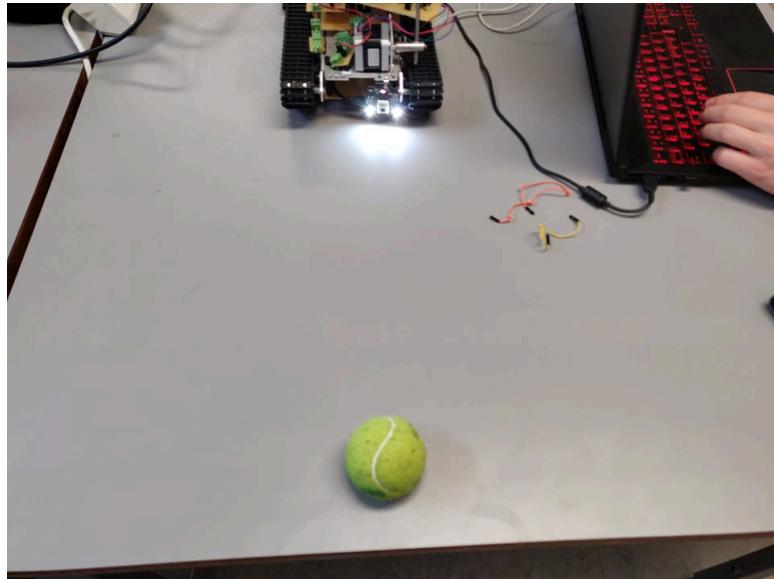
Figur 6.31: Resultater for DrejVenstre test

Case KørFrem

Precondition: PixyCam er initieret, og tennisbolden befinner sig indenfor værdierne: ($højde < 125 \&\& bredde < 135 \&\& x > 120 \&\& x < 180$).

Postcondition: Rpi programmet udskriver "Bolden er for langt væk kør tættere på."

Nedenstående figur 6.32 viser opstilling for case KørFrem.



Figur 6.32: Opstilling af test KørFrem med tennisboldens position i forhold til PixyCam

Nedenstående figur 6.33 viser resultatet case KørFrem.

```
Bolden er for langt væk kør tættere på
X: 131, Højde: 28, Bredde 58
Bolden er for langt væk kør tættere på
X: 132, Højde: 50, Bredde 40
Bolden er for langt væk kør tættere på
X: 132, Højde: 50, Bredde 40
Bolden er for langt væk kør tættere på
X: 140, Højde: 56, Bredde 56
Bolden er for langt væk kør tættere på
X: 140, Højde: 56, Bredde 56
Bolden er for langt væk kør tættere på
```

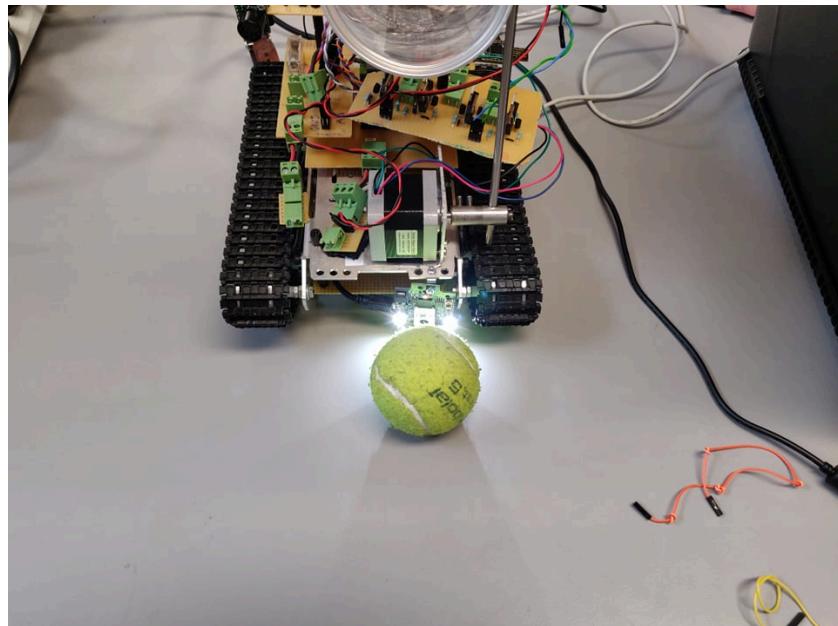
Figur 6.33: Resultater for Kørfrem test

Case KørTilbage

Precondition: PixyCam er initieret, og tennisbolden befinner sig indenfor værdierne: ($højde < 80 \&& bredde < 80 \&& x > 116 \&& x < 184$).

Postcondition Rpi programmet udskriver "Bolden er for tæt på bag tilbage."

Nedenstående figur 6.34 viser opstilling for testen af case KørTilbage.



Figur 6.34: Opstilling af test KørTilbage med tennisboldens position i forhold til PixyCam

Nedenstående figur 6.35 viser resultatet for case KørTilbage.

```
Bolden er for langt væk kør tættere på
X: 131, Højde: 28, Bredde 58
Bolden er for langt væk kør tættere på
X: 132, Højde: 50, Bredde 40
Bolden er for langt væk kør tættere på
X: 132, Højde: 50, Bredde 40
Bolden er for langt væk kør tættere på
X: 140, Højde: 56, Bredde 56
Bolden er for langt væk kør tættere på
X: 140, Højde: 56, Bredde 56
Bolden er for langt væk kør tættere på
```

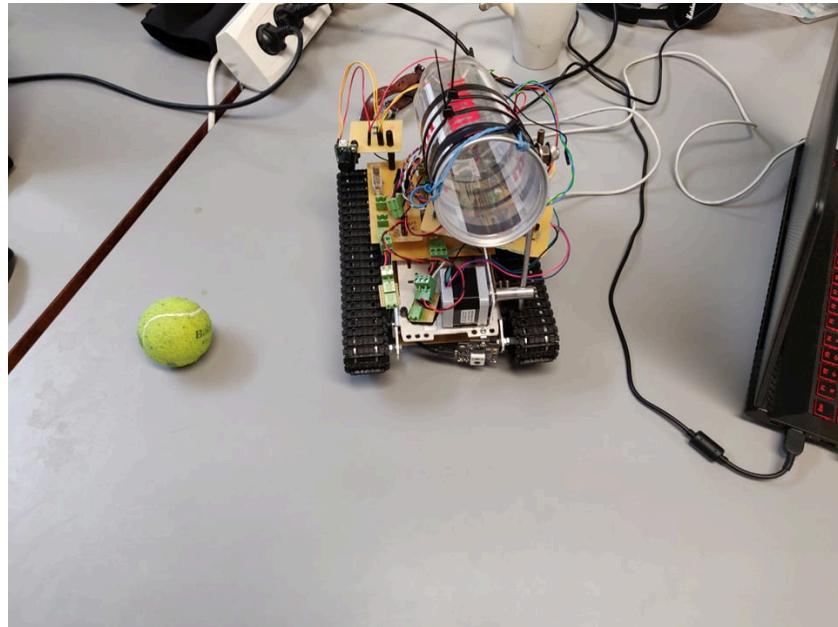
Figur 6.35: Resultater for KørTilbage test

Case SearchBall

Precondition: PixyCam er initieret, og tennisbolden registreres ikke af PixyCam.

Postcondition: Rpi programmet udskriver "Seach ball."

Nedenstående figur 6.36 viser opstilling for testen af case SearchBall.



Figur 6.36: Opstilling af test SearchBall med tennisboldens position i forhold til PixyCam

Nedenstående figur 6.37 viser resultatet for case SearchBall.

```
X: 154, Højde: 126, Bredde 188
Search ball, count er 0
X: 154, Højde: 126, Bredde 188
Search ball, count er 1
X: 154, Højde: 126, Bredde 188
Search ball, count er 2
X: 154, Højde: 126, Bredde 188
Search ball, count er 3
X: 154, Højde: 126, Bredde 188
Search ball, count er 4
X: 154, Højde: 126, Bredde 188
Search ball, count er 5
X: 154, Højde: 126, Bredde 188
Search ball, count er 6
X: 154, Højde: 126, Bredde 188
Search ball, count er 7
X: 154, Højde: 126, Bredde 188
Search ball, count er 8
```

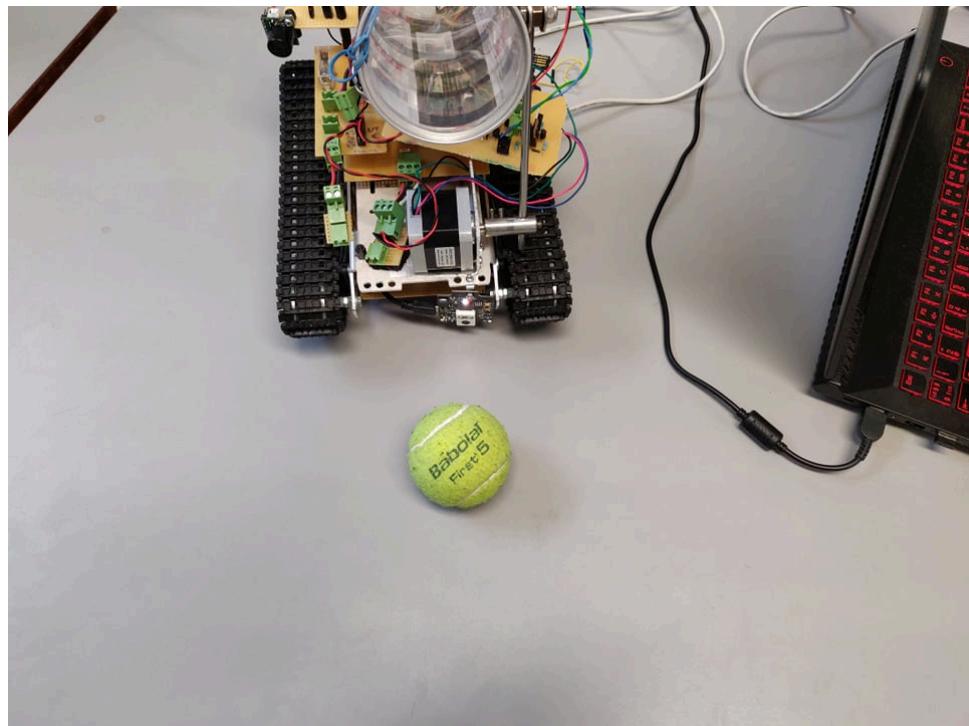
Figur 6.37: Resultater for SeachBall test

Case Samle Op

Precondition: PixyCam er initieret, og tennisbolden befinner sig indenfor værdierne: ($højde > 125 \&& bredde > 135 \&& x > 120 \&& x < 180$).

Postcondition: Rpi programmet udskriver "Samle bold op."

Nedenstående figur 6.38 viser opstilling for testen af case Samle Op.



Figur 6.38: Opstilling af test Samle Op med tennisboldens position i forhold til PixyCam

Figur 6.39 viser resultatet for case Samle Op.

```
Search date, count 0, 0
X: 174, Højde: 129, Bredde 192
Samler bold op
X: 153, Højde: 128, Bredde 186
Samler bold op
X: 153, Højde: 128, Bredde 190
Samler bold op
X: 153, Højde: 127, Bredde 186
Samler bold op
```

Figur 6.39: Resultater for Samle Op test

7. Integrationstest

Indledning

I dette afsnit laves integrationstest for modulerne. Integrationtesten er lavet systematisk, så det starter med få moduler, og løbende kobles flere moduler til for at arbejde mod det fulde system. Der vil desuden blive refereret til tidligere afsnit i henholdsvis modultest og tidligere scenarier i integrationstesten.

7.1 Scenarie 1

Enheder: Motormodul, spændingsregulator og motorstyring software.

Beskrivelse af test

Vi styrer diverse funktioner for motorstyringen igennem en UART terminal hvor der bekræftes hvorvidt bilen reagerer som forventet når de forskellige motorstyringsfunktioner kaldes gennem terminalen. Vi har til denne test lavet en testkode, som ikke kommunikerer over SPI, men UART.

Visuel test

Motorstyringsfunktioner	Forventet resultater	Visuel test
korFrem()	Enable_Right og Enable_Left bliver begge sat høje. Derudover forventes det at motor_R_F er aktiveret og at motor_L_F er aktiveret, samt at motor_L_R og motor_R_R er deaktiveret.	Begge motorer rotere fremad med samme PWM signal, så bilen kører fremad i en lige linje.
korTilbage()	Enable_Right og Enable_Left bliver begge sat høje. Derudover forventes det at motor_R_R er aktiveret og at motor_L_R er aktiveret, samt at motor_L_F og motor_R_F er deaktiveret.	Begge motorer rotere baglæns med samme PWM signal, så bilen kører baglæns i en lige linje.
drejHøjre()	Det forventes at Enable_Right er aktiveret og Enable_Left er deaktiveret. Derudover er motor_R_R aktiveret, og motor_R_F, motor_L_F og motor_L_R deaktiveret.	Højre motor rotere baglæns mens venstre motor er deaktiveret, så bilen drejer højre om egen akse.

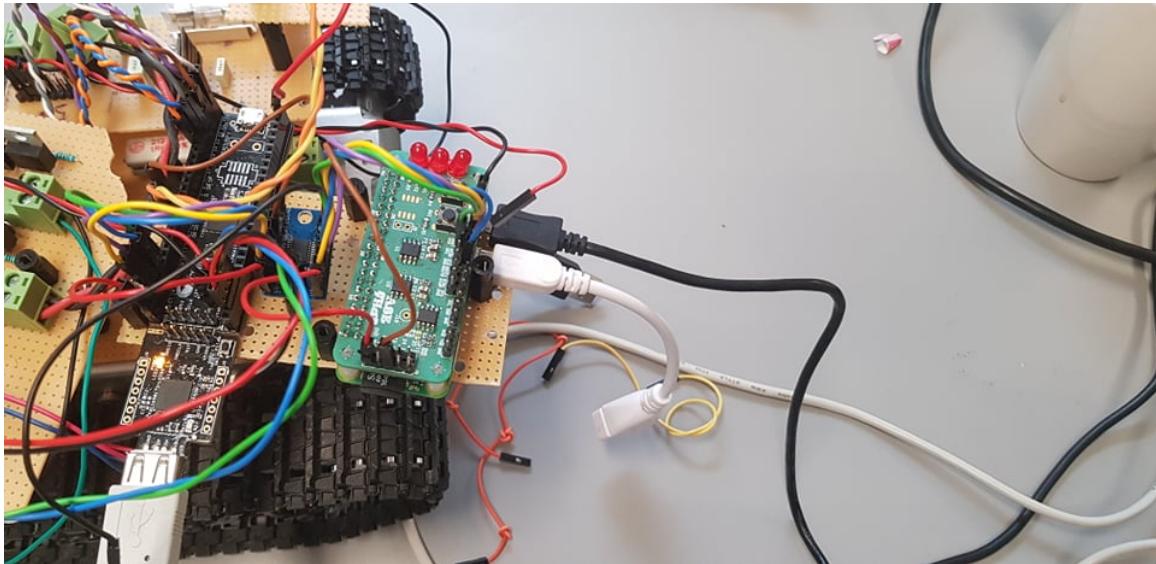
Motorstyringsfunktioner	Forventet resultater	Visuel test
drejVenstre()	Det forventes at Enable_Left er aktiveret og Enable_Right er deaktiveret. Derudover er motor_L_R aktiveret, og motor_R_F, motor_L_F og motor_R_R deaktiveret.	Venstre motor rotere baglæns og højre motor er deaktiveret så bilen drejer venstre om egen akse.
stop()	Både Enable_Right og Enable_Left er deaktiveret. Derfor er tilstanden for alle andre irrelevante.	Bilen stopper ved kald af funktionen stop().
fastStop()	Både Enable_Right og Enable_Left er aktive, men alle motor indstillinger er deaktiverede.	Bilen stoppede ved kald af funktionen fastStop(), hvilket resulterede i at begge motorer stoppede med at rotere.

7.2 Scenarie 2

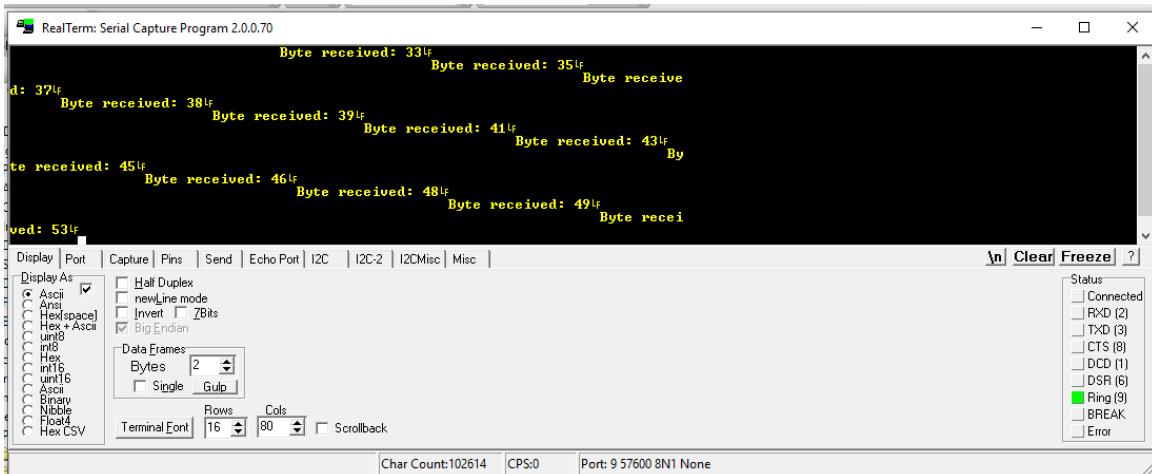
Enheder: PSoC, RPI og levelconverter.

Beskrivelse af test

I denne test, bliver der sendt data vha. SPI fra RPI til PSoC. Formålet med testen er at sende data fra RPi, og se den samme mængde data i Realterm på computeren. Figur 7.2 viser dokumentation for testen. Master sender i dette tilfælde et integer der tælles op hver gang der sendes, og dette kan også ses i terminalvinduet i Realterm. På figur 7.1 kan forsøgsopstillingen ses for integrationstesten mellem PSoC og RPi.



Figur 7.1: Forsøgsopstilling af test til SPI mellem PSoC og RPi.



Figur 7.2: Billede af Realterm hvor det kan ses at PSoC modtager det korrekte signal sendt fra Rpi

I tabel 7.1 kan forbindelser mellem PSoC og RPi ses.

Indgang (Levelconverter)	Forbindelse
VB	5 V
B1	PSoC (MOSI)
B2	PSoC (MISO)
B3	PSoC (CLK)
B4	PSoC (SS)
B5	PSoC (GPIO17)
VA	RPi 3.3 V
A1	RPi (MOSI)
A2	RPi (MISO)
A3	RPi (CLK)
A4	RPi (SS)
A5	RPi (GPIO17)
Ground	PSOC (Ground)

Tabel 7.1: Signalbeskrivelse af forbindelser mellem PSoC og RPi over levelconverter

7.3 Scenarie 3

Enheder: Motorstyring software, motormodul, spændingsregulator, Rpi og Pixy.

Beskrivelse af test

Som i scenarie 2, sektion 7.2 på side 96 'Scenarie 2', opsættes alle komponenter på bilen, med forsyning fra batterierne. Derudover er PixyCam også tilkoblet i denne test. RPi tilkobles over WiFi, og programmet startes, via terminal vindue på VM. En tennisbold placeres i forskellige positioner i forhold til BoldBot, og resultat observeres.

Visuel test

Placing af tennisbold	Sendt data fra RPi	Forventet resultater	Visuel test
Placing: 20cm lige for BoldBot	Data Sendt: 0b00001000	BoldBot kører fremad.	BoldBot kører fremad i en lige linje.
Placing: 20cm lige for BoldBot og 10cm til venstre for BoldBot	0b00001100	BoldBot drejer til venstre.	BoldBot aktiverer venstre motor og drejer til venstre.
Placing: 20cm lige for BoldBot og 10cm til højre for BoldBot	0b00001010	BoldBot kører mod højre.	BoldBot aktiverer kun højre motor og drejer til højre.
Placing: 1cm lige for BoldBot	0b00011110	BoldBot kører tilbage.	BoldBot kører tilbage.
Placing: udenfor BoldBots frame	0b00001010	BoldBot søger efter blod.	BoldBot drejer rundt om egen akse, hvorefter BoldBot kører fremad. Dette gentages indtil BoldBot registrerer en tennisbold.

7.4 Scenarie 4

Enheder: Motorstyring software, motormodul, spændingsregulator, Rpi, Pixy og Webside.

Beskrivelse af test

Formålet med denne test, er at bygge ovenpå testscenarie 3, sektion 7.3 på side 98 'Scenarie 3'. I denne test skal BoldBot programmet startes og stoppes fra Websiden, for

at at vise at BoldBot kan opstartes og slukkes fra brugergrænsefladen. BoldBot opsættes præcist som i scenarie 3, sektion 7.3 på side 98 'Scenarie 3', og der forbindes til BoldBots webside via. den fast defineret IP adresse 192.168.43.171:5000. Der trykkes først på start knappen på websiden, og der observeres om BoldBot startes. Efterfølgende trykkes der på stop, og der observeres om BoldBot stoppes.

Visuel test

Test	Forventet resultater	Visuel test
Start fra Webside	Webside starter BoldBot program på RPi, og søgning påbegyndes.	BoldBot starter op, og begynder at søge efter en bold.
Stop fra Webside	Webside stopper BoldBot program på RPi.	BoldBot stopper, og søger ikke længere efter en bold.

7.5 Scenarie 5

Enheder: Motorstyring software, motormodul, spændingsregulator, RPi, Pixy, Webside og opsamlermodul.

Beskrivelse af test

Formålet med denne test er at bygge ovenpå testscenarie 4, sektion 7.4 på side 99 'Scenarie 4'. I denne test bliver opsamlemodulet testet, sammen med resten af modulerne. BoldBot opsættes op med samme fremgangsmåde som i scenarie 5, sektion 7.5 på side 100 'Scenarie 5', hvorefter en tennisbold placeres i korrekt position foran BoldBot. Denne test udgøre den samlet integration af alle modulerne i systemet med udgangspunkt i at vise opsamlemodulets funktionalitet med resten af modulerne.

Visuel test

Placing af tennisbold	Sendt data	Forventet resultater	Visuel test
Placing: Korrekt position foran BoldBot $120 < x < 180 \&& højde > 125 \&& bredde > 125 \&& bredde < 200 \&& højde < 200$	0b00001001	Opsamlermodulet kører fra udgangspunkt ned til jorden	Opsamlermodulet kører ned til jorden.
Placing: Tennisbold er i opsamlermodulet	0b00001000	Opsamlermodulet bliver kørt op i startposition	Opsamlermodulet bliver aktiveret og kører op i startposition.

7.6 Scenarie 6

Enheder: Motorstyring software, motormodul, spændingsregulator, RPi, Pixy, webisde, opsamlermodul og sonarsensor.

Beskrivelse af test

Formålet med denne test er at bygge ovenpå testscenarie 5, sektion 7.5 på side 100 'Scenarie 5'. I denne test skal sonarsensoren testes, sammen med de andre moduler. BoldBot opsættes præcist som i scenarie 4, sektion 7.4 på side 99 'Scenarie 4', hvorefter BoldBot placeres med front mod en forhindring, og startes fra webside. Denne test laves for at vise at BoldBot kan styre udenom forhindringer.

Visuel test

Test	Forventet resultater	Visuel test
BoldBot placeres placeres med front mod en væg, højst 40 cm fra væggen.	BoldBots sonar sensor registrerer en væg, og bakker tilbage, og begynder søgning.	BoldBot bakker tilbage.

8. Accepttest

8.1 Accepttest 1.1

Accepttest for funktionelle krav

Vejleder har overværet denne accepttest.

Version 1.1. 10/05/2019 12:40

Software historik:

PSoC version: 1.3

RPi version: 1.4

Hovedscenarie

Use case under test	Start/Kør			
Scenarie	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 60cm foran BoldBot.	BoldBot kører mod bolden og opsamler i beholder.	BoldBot kører hen til tennisbolden og opsamler bolden.	OK
4	Bruger placerer tennisbold 150 cm foran BoldBot.	BoldBot kører mod bolden og opsamler i beholder.	BoldBot fejler i opsamling af tennisbolden, opsamlingsmodulet hæves ikke efter opsamling	Ikke gennemført korrekt, opsamlingsmodul hæves ikke efter opsamling.

Tabel 8.1: Accepttest af Use Case 1 - Start/Kør

Extension 1

Use case under test	Start/Kør			
Scenarie	Extension 1 - BoldBot møder forhindring			
Prækondition	Boldbot søger efter en bold.			
No.	Handling	Forventet resultat	Faktisk resultat	Vurdering (+/-)
1	Bruger placerer en fod 40 cm foran sonarsensoren imens den køre	BoldBot stopper 40 cm før forhindringen, bakker, og starter en ny søgning efter tennisbold	Ikke udført, pga koden ikke virker med sonar sensor.	Ikke OK

Tabel 8.2: Accepttest af Use Case 1 - Extension 1

Extension 2

Use case under test	Start/Kør - Extension 2			
Scenarie	Extension 2 - Fejl i opsamling			
Prækondition	Boldbot er aktiv.			
No.	Handling	Forventet resultat	Faktisk resultat	Vurdering (+/-)
1	Bruger rykker på tennisbold, under igangværende opsamling.	BoldBot bakker og starter søgning efter tennisbold.	Bruger fjerner bold korrekt, men BoldBot hæver ikke opsamlingsmodul efter forsøg på opsamling.	Not OK

Tabel 8.3: Accepttest af Use Case 1 - Extension 2

Use Case 2 - Stop BoldBot

Use case under test	Start/Kør			
Scenarie	Hovedscenarie			
Prækondition	Boldbot er aktiv.			
No.	Handling	Forventet resultat	Faktisk resultat	Vurdering (+/-)
1	Bruger indtaster 192.168.43.171:5000 på valgfri webbrowser.	Webside vises på skærmen.	Webside vises korrekt.	
2	Bruger trykker på 'Stop Boldbot' knappen på hjemmesiden.	BoldBot stopper sin kørsel.	BoldBot stopper korrekt, men først efter 3 klik på "Stop BoldBot".	OK

Tabel 8.4: Accepttest af Use Case 2 - Stop

Accepttest for ikke funktionelle krav

Ikke funktionelle krav		GUI			
No.	Krav	Test/Udførelse	Forventet resultat	Faktisk resultat	Vurdering (+/-)
1	BoldBot må ikke overstige 40x40x4 cm	BoldBot måles med centimetermål på x, y og z akse.	BoldBot overskridt ikke målene i kravene.	BoldBot måles med et målebånd på +/- 2 cm, til at være 38x32x19 HxLxB	
2	BoldBot skal kunne samle 1 bold op på mindst 4 minutter og 55 sekunder.	Der placeres en bold indenfor BoldBots rækkevidde og BoldBot startes. Målt med et stopur med en nøjagtig på +/- 5 sekunder	BoldBot har opsamlet bolden, indenfor 5 min.	BoldBot finder bold og opsamler den, på 3 minutter og 40 sekunder.	OK
3	BoldBot skal kunne opbevare mindst 1 bolde før den skal tømmes.	Der placeres 1 bold i opsamlingsrøret af BoldBot.	De 1 bolde forbliver i opsamlingsrøret.	BoldBot holder den ene bold imens den søger.	OK
4	BoldBot skal kunne testes på andet end en tennisbane for at se om den kan detekttere bolde og samle dem op.	BoldBot placeres på andet sted end en tennibane. Tennisbold placeres +/- 5 meter fra BoldBot.	BoldBot registrerer tennisbolden hvorefter den kører mod tennisbolden og samler den op.	BoldBot testes på gulvet af lokale 145 i Kanh.	OK
5	Batteriindikatoren skal kunne vise 5 forskellige batteriniveauer.	Alle 8 LED'er er slukkede.	Batteriindikatorne viser hvor hvor høj en spænding batteriet har, ud fra de 5 indikatorer.		
6	Når ikke BoldBot kan se en tennisbold skal den begynde at rotere omkring sig selv.	BoldBot placeres på en testbane uden tennisbolde.	BoldBot roterer en gang omkring sig selv og kører en bestemt afstand frem.		
7	Boldbot skal kunne køre mindst 15 minutter på en fuld opladning.	BoldBot placeres på en testbane uden tennisbolde.	BoldBot sættes til at køre på en bane med et bestemt antal bolde	BoldBot har kørt i 16 min fra start af test til slut. Kravet er opfyldt.	

Tabel 8.5: Accepttest af ikke funktionelle krav

8.2 Accepttest 1.2

Accepttest for funktionelle krav

Version 1.2 24/05/2019 15:50

Software historik:

PSoC version: 1.4

RPi version: 1.6 Vejleder var ikke tilstede.

Hovedscenarie

Use case under test		Start/Kør		
Scenarie		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 100 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 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 8.6: Accepttest af Use Case 1 - Start/Kør

Extension 1

Use case under test		Start/Kør		
Scenarie		Extension 1 - BoldBot møder forhindring		
Prækondition		Boldbot søger efter en bold.		
No.	Handling	Forventet resultat	Faktisk resultat	Vurdering (+/-)
1	Bruger placerer en fod 40 cm $\pm 2cm$ foran sonarsensoren imens den køre	BoldBot stopper 40 cm $\pm 2cm$ før forhindringen, bakker, og starter en ny søgning efter tennisbold	Ikke udført, pga fejlet integrationstest.	Ikke OK

Tabel 8.7: Accepttest af Use Case 1 - Extension 1

Extension 2

Use case under test		Start/Kør - Extension 2		
Scenarie		Extension 2 - Fejl i opsamling		
Prækondition		Boldbot er aktiv.		
No.	Handling	Forventet resultat	Faktisk resultat	Vurdering (+/-)
1	Bruger rykker på tennisbold, under igangværende opsamling.	BoldBot bakker og starter søgning efter tennisbold.	BoldBot bakker, og efterfølgende søges efter bold igen.	OK

Tabel 8.8: Accepttest af Use Case 1 - Extension 2

Use Case 2 - Stop BoldBot

Use case under test	Start/Kør			
Scenarie	Hovedscenarie			
Prækondition	Boldbot er aktiv.			
No.	Handling	Forventet resultat	Faktisk resultat	Vurdering (+/-)
1	Bruger indtaster 192.168.43.171:5000 på valgfri webbrowser.	Webside vises på skærmen.	Webside vises korrekt.	OK
2	Bruger trykker på 'Stop Boldbot' knappen på hjemmesiden.	BoldBot stopper sin kørsel.	BoldBot stopper korrekt, efter tryk på 'Stop' på webside.	OK

Tabel 8.9: Accepttest af Use Case 2 - Stop

Accepttest for ikke funktionelle krav

Ikke funktionelle krav		GUI			
No.	Krav	Test/Udførelse	Forventet resultat	Faktisk resultat	Vurdering (+/-)
1	BoldBot må ikke overstige 40x40x4 cm	BoldBot måles med centimetermål på x, y og z akse.	BoldBot overskrider ikke målene i kravene.	BoldBot målet med et målebånd på +/- 2cm til at være 38x32x19 HxLxB	OK
2	BoldBot skal kunne samle 1 bold op på mindst 4 minutter og 55 sekunder	Der placeres en bold indenfor BoldBots rækkevide og BoldBot startes. Målt med et stopur med en nøjagtig på +/- 5 sekunder	BoldBot har opsamlet bolden, indenfor 5 min.	BoldBot finder bold og opsamler den, på 2 minutter og 59 sekunder	OK
3	BoldBot skal kunne opbevare mindst 1 bold før den skal tømmes.	BoldBot opsamler en enkelt bold.	Bolden forbliver i røret efter opsamling	BoldBot holder den ene bold imens den søger.	OK
4	BoldBot skal kunne testes på andet end en tennisbane for at se om den kan detektere bolde og samle dem op.	BoldBot placeres på andet sted end en tennibane. Tennisbold placeres +/- 5 meter fra BoldBot.	BoldBot registrerer tennisbolden hvorefter den kører mod tennisbolden og samler den op.	BoldBot testes på gulvet af lokale 245 i Kanh.	OK
5	Batteriindikatorer skal kunne vise 5 forskellige batteriniveauer.	Alle 8 LED'er er slukkede.	Batteriindikatorne viser hvor hvor høj en spænding batteriet har, ud fra de 5 indikatorer.	Ikke testet pga. manglende implementering.	Ikke OK
6	Når ikke BoldBot kan se en tennisbold skal den begynde at rotere omkring sig selv.	BoldBot placeres på en testbane uden tennisbolde.	BoldBot roterer en gang omkring sig selv og kører 30 cm +/- 5 cm frem.	BoldBot begynder at søge, og roterer omkring sig selv, og kører 30 cm frem +/- 2 cm frem.	OK
7	. Boldbot skal kunne køre mindst 15 minutter på en fuld opladning.	BoldBot placeres på en testbane uden tennisbolde.	BoldBot sættes til at køre på en bane med 1 bold.	BoldBot opsamler bolden, og søger resten af tiden på banen.	OK

Tabel 8.10: Accepttest af ikke funktionelle krav

Bibliografi

- [1] Ashen Gamage, T.A.G, (2017) *HTTP and Websockets: Understanding the capabilities of today's web communication technologies.* Tilgængelig på: <https://medium.com/platform-engineer/web-api-design-35df8167460>
- [2] jfrench, 2018 How to talk to Pixy2. Tilgængelig på: https://docs.pixycam.com/wiki/doku.php?id=wiki:v2:portin_guide
- [3] PixyCam, 2018 Pixy2 Full API. Tilgængelig på: https://docs.pixycam.com/wiki/doku.php?id=wiki:v2:full_apiplugin_include_wiki_v2_general
- [4] W3Schools, W3.CSS Reference. Tilgængelig på: https://www.w3schools.com/w3css/w3css_references.asp
- [5] Datablad: lm7800, Texas Instruments, 2016
- [6] Datablad, L298_HBridge, STMicroelectronics, 2000