

Rettelser

INGENIØRHØJSKOLEN AARHUS

3. SEMESTERPROJEKT PRJ3

MÅLMANDSTRÆNEREN "BLOW-BALL 3000"

Rapport

GRUPPE 14

Navn	Studienummer
Alexander Dahl Bennedsen	201310498
Cecilie Moriat	201405949
Jonas Møgelvang Hansen	201407199
Lasse Stenhøj	201407500
Sarah Overby	201407410
Simon Hansen	201403871
Thomas Nielsen	201405723
Thomas Skovgaard Rasmussen	201406754

VEJLEDER: MICHAEL ALRØE

DATO: 16/12/2015

Indholdsfortegnelse

Indholdsfortegnelse	3
Resumé	5
Abstract	5
Kapitel 1 Indledning	6
1.1 Begrebsliste	7
1.2 Læsevejledning	8
1.3 Ansvarsområder	9
1.4 Projektafgrænsning	11
Kapitel 2 Projektformulering	12
2.1 Systembeskrivelse	14
Kapitel 3 Krav	18
3.1 Use Case Beskrivelse	18
3.1.1 Scan mål	18
3.1.2 Affyr Bolde	18
3.1.3 Kontroller Blow-Ball 3000 over WiFi	18
Kapitel 4 Projektbeskrivelse	19
4.1 Specifikation og analyse	19
4.2 Udviklingsværktøjer	20
4.2.1 PSoC Creator	20
4.2.2 Qt Creator	20
4.2.3 Arduino IDE	20
4.2.4 Multisim	20
4.2.5 Ultiboard	20
4.2.6 Git	20
4.2.7 Redmine	21
4.2.8 L ^A T _E X	21
4.2.9 Microsoft Visio	21
4.2.10 Analog Discovery	21
4.3 Projektgennemførelse	22
Kapitel 5 Arkitektur	24
Kapitel 6 Hardware design	26
6.1 Servomotor	26
6.2 DC-motor	26
6.3 Sensor	27
6.4 LED-countdown	28

6.5	PSU	28
6.6	Level converter	28
6.6.1	WiFi-modul	28
6.7	Step-motor	28
Kapitel 7 Software Design		30
7.1	PSoC4	30
7.2	User interface	34
7.3	ESP8266	36
7.4	UART	36
Kapitel 8 Resultater og diskussion		39
8.1	Resultater	39
8.2	Tidsplan	39
Kapitel 9 Opnæde erfaringer		42
9.1	Generelle erfaringer	42
9.1.1	Individuelle erfaringer	42
9.2	Fremtidigt Arbejde	45
Kapitel 10 Konklusion		47
Litteratur		48

Resumé

Denne rapport er udarbejdet som en del af læringsmålene for kurset PRJ3 på Ingeniørhøjskolen Aarhus. Rapporten omfatter en gennemgående beskrivelse af projektforløbet, samt dokumentation af produktets software og hardware.

Kravende til dette semesterprojekt har bestået af følgende fire elementer:

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

Dette er udmundet i produktet **Blow-Ball 3000**; En automatiseret målmandstræner. Blow-Ball 3000 har implementeret en afstands-sensor til at opmåle målgrænser, således at brugeren kan vælge ønskede målområder der skal trænes indenfor på den implementerede linux-baserede brugergrænseflade. Herpå indstilles Blow-Ball 3000 til affyring ved hjælp af den implementerede PSoC platform.

Projektarbejdet er udført vha. en tilpasset udgave af SCRUM, efter en prioriteringsorden udarbejdet efter MoSCoW-modellen.

Produktet er skabt som en nedskaleret prototype der blot affyrer mindre hoppebolde, men skaleringen beror alene på størrelsen af motorer og forsyning.

Vedlagt er en zip-fil med komplet dokumentation, datablade etc.

Abstract

This report has been produced as a part of the educational goals of the course PRJ3 at Ingeniørhøjskolen Aarhus, the Aarhus Academy of Engineering. The report contains a thorough description of the working process, as well as documentation of the product's software and hardware.

The mandatory requirements of this semester-project have consisted of the following four elements:

- The system *must* interact to the surrounding world by the help of sensors/actuators.
- The system *must* have a user-interface.
- The system *must* contain academic elements from other courses of the semester.
- The system *must* make use of an embedded Linux platform and a PSoC platform.

This has concluded with the product **Blow-Ball 3000**; An automated goalkeeper trainer. Blow-Ball 3000 has an implemented distance-sensor, in order to measure the size of the goal, so that the user may choose the desired areas of training on the implemented, linux-based user-interface. Hereafter Blow-Ball 3000 is set for shooting, by the help of the implemented PSoC.

The product has been created at the process of an adapted version of SCRUM, after a priority made by the MoSCoW-model.

The product has been scaled down to a prototype that shoots smaller bouncing balls, but the scaling is solely based on the size of the engines and power-supply.

Enclosed is a zip-file containing complete documentation, datasheets, etc.

Indledning 1

Denne rapport og dens bilag er skrevet som dokumentation for udarbejdelsen af et produkt og dertilhørende arbejdsproces i forbindelse med 3. semester projektet på Ingeniørhøjskolen ved Aarhus Universitet (IHA).

Dette semesterprojekt har været valgfrit ift. hvilket produkt der skulle udvikles, men med enkelte krav til implementeringen af dets funktionalitet.

Produktet der udvikles, er døbt **Blow-Ball 3000**: Den Automatiserede Målmandstræner.

Produktet udvikles til en målmand, der på egen hånd kan træne redninger i målet uden behovet for en medspiller, som kan sparke til bolden. Ved at vælge områder af målet som bolden tilfældigt skal affyres imod, kan målmanden dermed øve de svære redninger.

Der ønskes i dette projekt at udvikle en prototype. Dette skyldes at størrelsen på en rigtig model, vil tage for lang tid rent konstruktionsmæssigt. Derudover vil motorstørrelsen, der skal anvendes til at affyre en fodbold være af en størrelse, der ikke er billig at fremskaffe.

1.1 Begrebsliste

Følgende liste giver en forklaring på de ord og begreber som anvendes i denne rapport.

PSoC

Anvendes som forkortelse for den PSoC4 chip som sidder på boardet PSoC 4 Pioneer kit - CY8CKIT-042 fra Cypress. Det er dette board som bruges til at styre funktionaliteten af motorer m.m. Se afsnit 7.1 på side 30.

Devkit

Angiver Devkit8000 boardet, lavet af IHA - Århus Universitet. Devkittet bruges til brugerinterface på den indbyggede touchskærm. Se afsnit 7.2 på side 34.

Kanon

Ordet kanon anvendes om de 2 DC-motorer der bruges til at sende bolde af sted. Se afsnit 6.2 på side 26.

Tårnet

Med tårnet, menes der selve konstruktionen som kan dreje i x-aksen og vippe i y-aksen. Kanonen sidder på tårnet. Tårnet er en samlet betegnelse af Sektion F - I på figur 2.2 på side 15.

PSU

Denne forkortelse bruges for den eksterne Power Supply Unit, der bruges til at forsyne hele enheden med. Se afsnit 6.5 på side 28.

ESP8266

Anvendes som forkortelse for ESP8266-01 WiFi modul som består af en ESP8266EX SoC[1]. Denne SoC understøtter WiFi, der gør det muligt at have et webinterface på Blow-Ball 3000. Se afsnit 6.6.1 på side 28 og 7.3 på side 36.

LIDAR

Forkortet betegnelse for Lidar Lite v2 Sensoren der bruges til at måle afstanden til målets stolper og dermed er med til at afgrænse målområdet. Se afsnit 6.3 på side 27.

1.2 Læsevejledning

Denne rapport redegører for den proces som projektet har fulgt og giver læseren et indblik i projektdokumentationen med vægt på projektets særegenheder. Denne rapport følger som udgangspunkt strukturen, der er anbefalet i det vejledende dokument udleveret af IHA. Da rapporten er en del af den samlede dokumentation for projektet vil der forekomme henvisninger til tekst og figurer i projektdokumentationen.

Rapporten kan overordnet inddeltes i 5 sektioner:

I første sektion gives der i kapitel 1 en indledende beskrivelse af det valgte projekt og dets afgrænsninger. Der kan i denne sektion også se ansvarsområderne for de forskellige gruppemedlemmer. I kapitel 2 findes der en beskrivelse af projektet og i kapitel 3 bliver kravene til systemet beskrevet.

I anden sektion beskrives der i kapitel 4 de arbejdsgange og metoder, som er blevet brugt under projektet.

I tredje sektion ses de modeller der er blevet udarbejdet og hvordan Blow-Ball 3000 er blevet designet, implementeret og testet. Sektionen dækker over kapitel 5,6 og 7.

I fjerde sektion af rapporten finder man resultater og diskussionen, samt hvilke overvejelser man har gjort for at projektet kan videreudvikles. Fjerde sektion indeholder ligeledes generelle og individuelle erfaringer. Dette sker i kapitel 8 og 9.

Femte sektion indbefatter kapitel 10, hvor der konkluderes på projektet som helhed, og der drages parallel til indledningen.

1.3 Ansvarsområder

	Ansvarsområde	Beskrivelse af arbejdsopgaver
Alexander Dahl 	PSoC4-programmering Mindre HW-opgaver Konstruktion	Hovedprogrammør på Blow-Ball 3000's PSoC implementering. Derudover tæt konsultation med HW og udskiftninger af HW undervejs, med dertil ny programmering.
Cecilie Moriat 	Devkit-Programmering	User interface ansvarlig Ansvarlig for udvikling af grafisk brugergrænseflade i Qt Creator.
Jonas Hansen 	Scrummaster WiFi-Programmering UART-Programmering	Scrummaster med ansvar for scrum-standup møder. Ansvarlig for WiFi-funktionalitet samt implementering af UART-Drivere på de forskellige enheder.
Lasse Stenhøj 	Projektleder HW-ansvarlig Konstruktion	Projektleder med ansvar for alle møder, deadlines og primærstruktur for projektet. Ansvarlig for HW-delen med design, test og implementering af servo-motorer og LED, samt tæt samarbejde med PSoC-programmørerne ift. fejlfinding og problemløsninger. Hovedansvarlig for opbygning og design af hele konstruktionen, samt påmontering med ledninger.

	PSoC4-Programmering Doxygen PSoC	Programmør på Blow-Ball 3000's PSoC implementering. Hovedansvarlig for at lave Doxygen over PSOC-koden.
	PSoC4-programmering	Programmør på Blow-Ball 3000's PSoC implementering, samt enhedstestning af Blow-Ball 3000's metoder.
	HW-Ansvarlig Konstruktion	Enhedstester af alle motorer og andet anvendt i projektet. Designer på HW-del, inklusiv test og implementering. Derudover opbygning af konstruktion.
	HW-Ansvarlig Konstruktion	Hardware-mand på Blow-Ball 3000. Har udført design, test og implementering på motorstyringsprint og spændingsforsyning.

1.4 Projektafgrænsning

Projektet har haft en naturlig begrænsning i form af den korte tid fra idé til produkt, som et sådant semesterprojekt altid vil have.

Derfor har hovedfokus for produktet Blow-Ball 3000 ligget på kernefunktionaliteten; Målmandstræning.

Derudover er produktet fremstillet i en nedskaleret udgave, som "proof-of-concept", da den nødvendige hardware for at afsende fuld-størrelse fodbold afsted i tilfredsstillende hastighed ikke har været tilgængelig.

Denne afgrænsning har været klarlagt fra projektets start, og har med sit realistiske udgangspunkt forebygget eventuelle forsinkelser fra omdesigns, uopfyldte krav og lignende.

Projektformulering 2

Formålet med dette projekt er at skabe en interaktiv maskine, som kan sikre den optimale målmandstræning. Produktet kaldes **Blow-Ball 3000**. Systemet vil ved hjælp af en afstandssensor kunne aflæse et målområde og inddele dette i firkantede felter, for at kunne afbilde dette på en berøringsfølsom skærm. Denne grafiske grænseflade benyttes af brugeren til at få en boldkanon til at sigte mod målet og affyre bolde. Brugeren vil have mulighed for selv at vælge det område, som Blow-Ball 3000 sigter efter. Indefor de valgte områder affyres der tilfældigt. MoSCoW-modellen er benyttet til prioriteringen:

1. Scan mål (Must)

Denne Use Case involverer afstands-sensoren, hvilket gør den til en vital del af projektet for at opfylde opgavekravene. Formålet her er at scanne målet, således at der altid affyres korrekt uanset placering af Blow-Ball 3000 eller mål.

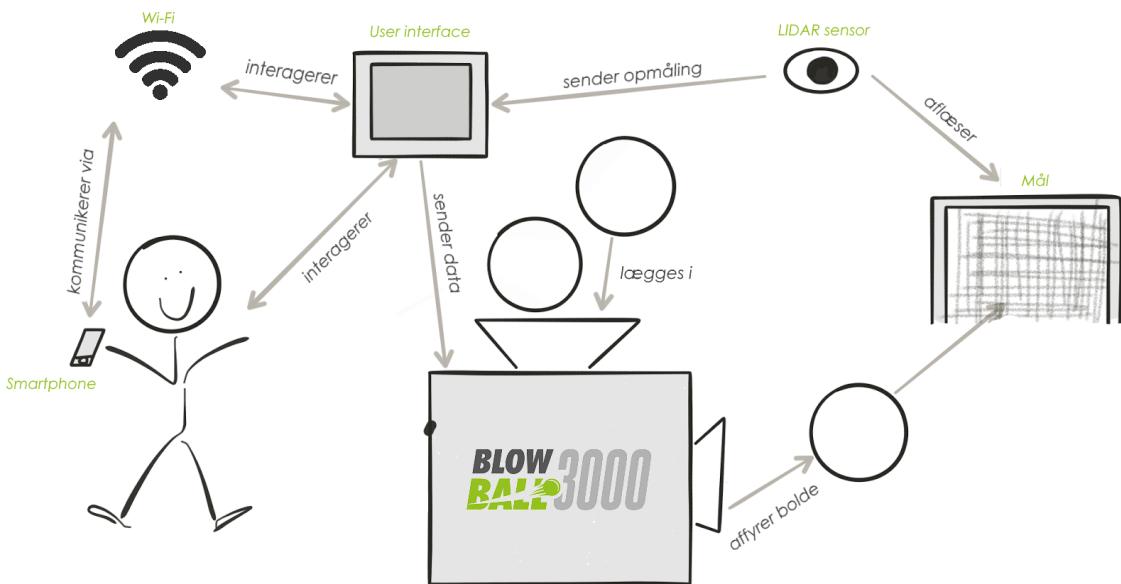
2. Affyr bold (Must)

Denne Use Case står for affyring af bolden, hvilket er en væsentlig del af maskinens funktion. Brugeren vælger hvilke områder boldene skal affyres i og med hvilken hastighed de skal affyres. Herpå affyres en sekvens af tre bolde i tilfældige dele af de markerede områder.

3. Kontroller Blow-Ball 3000 over WiFi (Should)

Denne Use Case gør det muligt for brugeren at vælge scan eller shoot på sin egen enhed f.eks. mobiltelefonen. Denne funktionalitet er ikke en vital del af projektet, men bidrager til en bedre brugeroplevelse.

Figur 2.1 nedenfor illustrerer anvendelsen af Blow-Ball 3000. Billedet beskriver hvordan systemet og brugeren interagerer med hinanden.



Figur 2.1. Rigt billede over systemet

2.1 Systembeskrivelse

På billede 2.2 - 2.4 ses Blow Ball 3000 realiseret.

Sektion A viser PSU'en som forsyner alle dele af Blow-Ball 3000 med strøm.

Sektion B viser Devkit8000 som er brugergrænsefladen, hvorfra brugeren kan styre systemet.

Sektion C er level converteren samt WiFi-modulet.

Sektion D viser PSoC'en som kommunikerer med Devkit'et og som benyttes til at styre motorerne, LIDAR sensoren og LED'en.

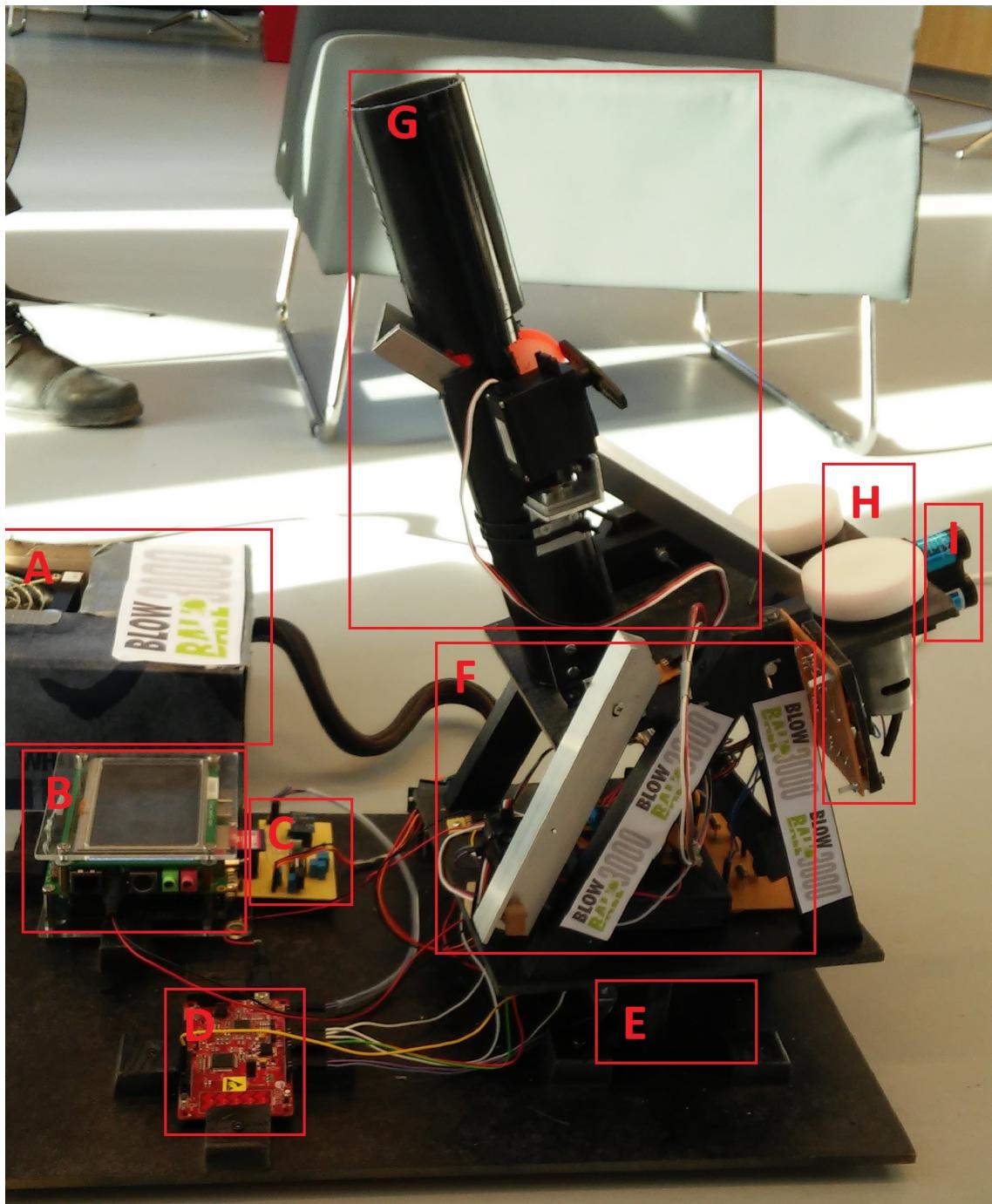
Sektion E viser styring af x-aksen, der består af en servo-motor som styres af PSoC'en.

Sektion F viser styringen af y-aksen, som ligeledes består af en servo-motor der styres ved hjælp af PSoC'en. Udover servo-motor, ses fordeler-printet.

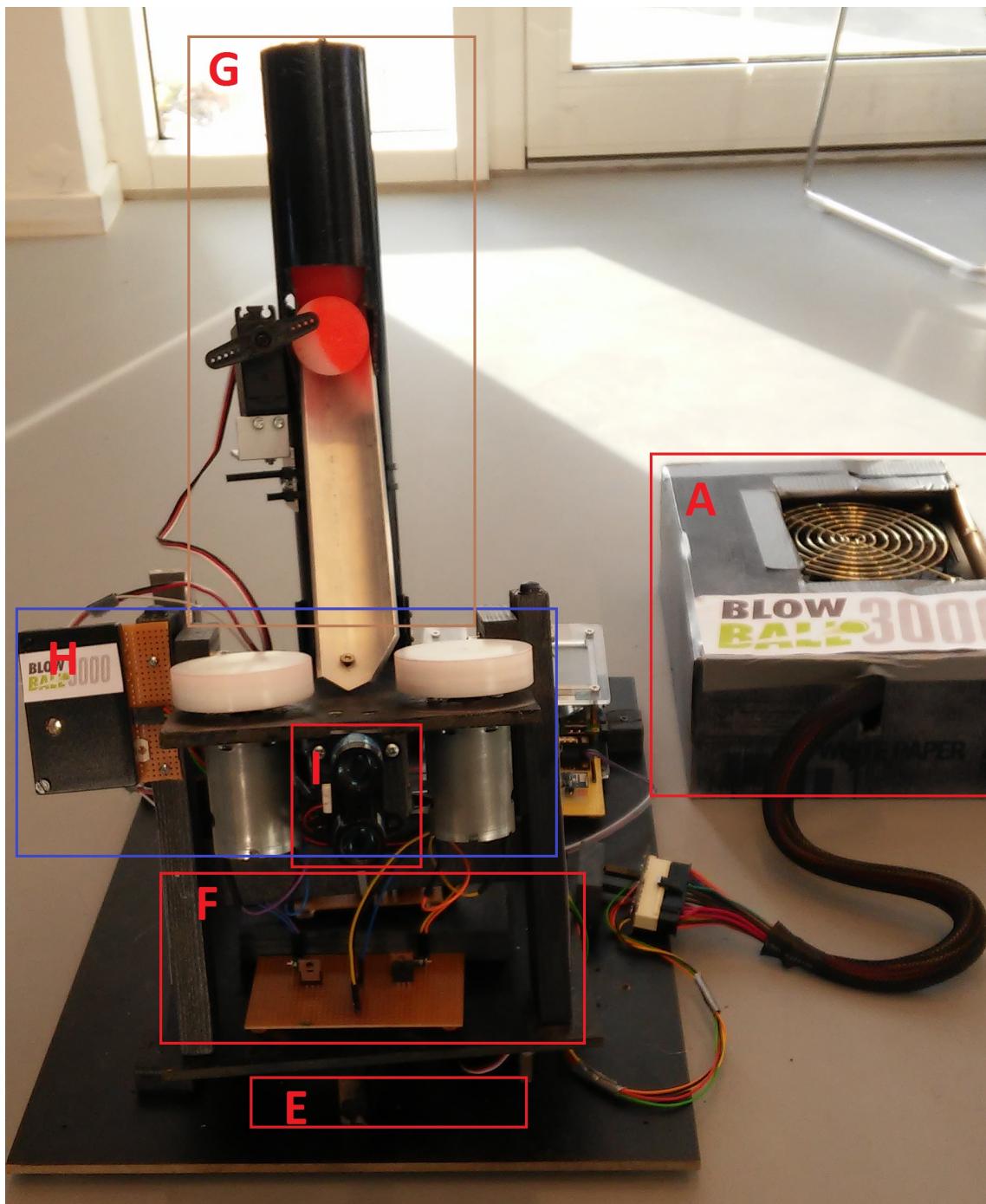
Sektion G er ball-loaderen, som består af en rampe der fører boldene ned til hjulene på DC-motorerne, samt en servo-motor der lukker boldene ud af opbevaringsrøret.

Sektion H er affyringsmekanismen shooting-engine, som består af to DC-motorer der styres af PSoC'en. I denne sektion kan man desuden også se count-down LED'en, som tæller ned således, at målmanden ved hvornår en affyringssekvens startes.

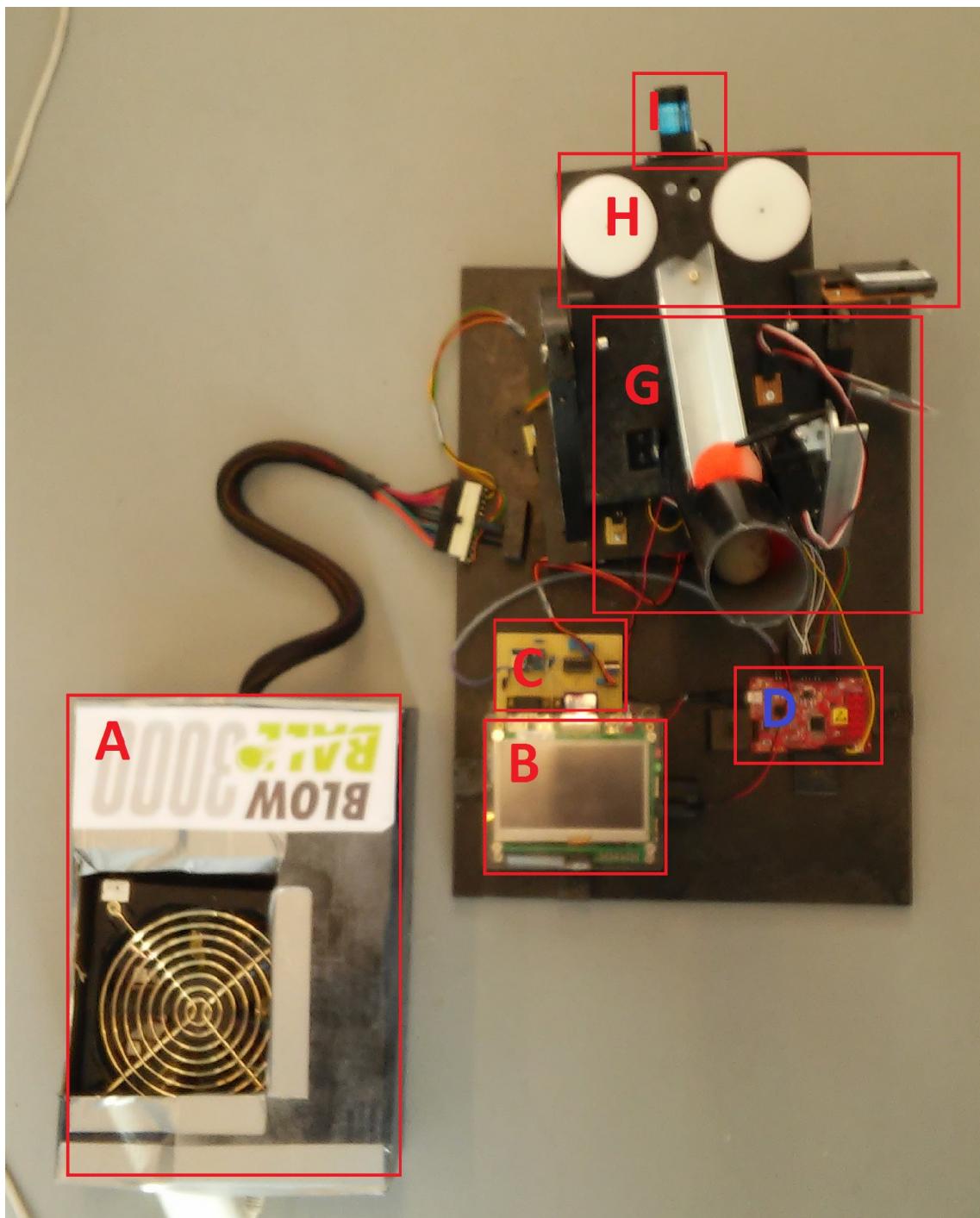
Sektion I viser LIDAR sensoren, som benyttes til at registrere målet.



Figur 2.2. Blow Ball 3000 set fra højre



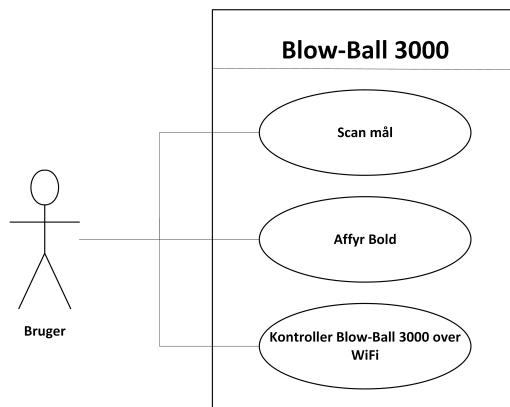
Figur 2.3. Blow Ball 3000 set forfra



Figur 2.4. Blow Ball 3000 set fra oven

Krav 3

Ud fra vores brugerscenarie (se afsnit 3.1 på side 11 i dokumentationen) har vi kunnet danne 3 Use Cases, som er afbilledet på figur 3.1. På figuren ses det at *bruger* interagerer med alle Use Cases, som primær aktør.



Figur 3.1. Use Case Diagram

3.1 Use Case Beskrivelse

I dette afsnit vil der være en beskrivelse af de forskellige Use Cases.

3.1.1 Scan mål

Her vælger *bruger* at scanne målet, så Blow-Ball 3000 ved hvor stort målet er. Målets størrelse benyttes til at definere 6 målområder, man kan skyde indenfor. En detaljeret Use Case-beskrivelse kan ses i afsnit 3.4.1 på side 13 i dokumentationen.

3.1.2 Affyr Bolde

Her trykker *bruger* på shoot knappen, hvorefter *bruger* kan vælge hastighed og hvor de 3 bolde skal skydes hen. Hvis *bruger* ikke vælger at ændre hastigheden skyder Blow-Ball 3000 med en standard hastighed. En detaljeret Use Case-beskrivelse kan ses i afsnit 3.4.2 på side 14 i dokumentationen.

3.1.3 Kontroller Blow-Ball 3000 over WiFi

Her vælger *bruger* at styre Blow-Ball 3000 over WiFi med en WiFi kompatibel enhed. *Bruger* forbinder til WiFi og indtaster IP i browseren og kan herefter enten vælge Scan eller Shoot. En detaljeret Use Case-beskrivelse kan ses i afsnit 3.4.3 på side 15 i dokumentationen.

Projektbeskrivelse 4

4.1 Specifikation og analyse

Som udgangspunkt blev der diskuteret hvilket projekt der skulle bygges. Her blev gruppen enige om, at bygge en boldkanon som kunne bruges til at træne en målmands redninger. Produktet blev i denne fase døbt Blow-Ball 3000: Den Automatiserede Målmandstræner. For simplicitetens skyld blev der valgt, at boldkanonen blot skulle affyre små bolde - det skulle altså være en prototype af slutproduktet.

Herefter blev der taget fat i MoSCoW-modellen hvori det færdige produkts egenskaber blev specificeret og prioriteret. Resultatet af dette arbejde blev systemets Use Cases. Ved videre arbejde med de færdige Use Cases blev der udarbejdet punkter til accepttest. Disse punkter skulle produktet kunne udføre, når produktet var færdigt.

Herfra blev der gået videre til, at analysere hvordan produktet eventuelt kunne opbygges. Dette blev gjort ud fra de specificerede Use Cases og de krav der var blevet stillet omkring projektgennemførelse.

Boldkanonen skulle kunne bruge en sensor til at opfange et vilkårligt mål og kalibrere derefter. Ud fra denne kalibrering skulle kanonen kunne sigte og ramme valgte felter i målet.

Kanonen skulle altså kunne dreje rundt om en horisontal akse og en vertikal akse. Hertil blev der taget udgangspunkt i forskellige typer af elektromotorer. Et krav til valget af motorer var, at de skulle kunne dreje konstruktionen præcist og hurtigt.

Til kommunikation mellem hardware og software blev der brugt en PSoC - idet den har en række funktioner (bl.a. PWM og UART) som er velegnede til, at kunne opfylde vores krav.

Til kommunikation mellem bruger og systemet bruges et Devkit. Dette er en simpel platform hvorpå der kan udvikles en brugergrænseflade; som var et krav til vores projekt. Devkittet understøtter desuden også UART og kunne derfor nemt kommunikere med PSoC og et WiFi-modul.

For at tillade brugeren, at kunne kontrollere systemet trådløst, skulle der bruges et WiFi-modul. Hertil skulle der bruges en simpel og billig komponent. Valget faldt her på et modul af typen ESP8266.

4.2 Udviklingsværktøjer

Herunder vil de udviklingsværktøjer der er brugt under udviklingen af Blow-Ball 3000 blive beskrevet. Versioner af de forskellige programmer m.m. kan ses i afsnit ?? på side ?? i dokumentationen.

4.2.1 PSoC Creator

PSoC Creator[2] er en IDE (Integrated Development Environment) der blev brugt til udviklingen af PSoC(Programmable System-on-Chip) softwaren. I værktøjet kan der opsættes port-forbindelser, PWM-moduler m.m. Udviklingen af koden bliver også gjort i PSoC Creator. Som udgangspunkt er det muligt at skrive i C og C++. Koden til dette projekt er skrevet i C.

4.2.2 Qt Creator

Qt er et event-drevet GUI (Graphical user interface) C++ bibliotek, hvor Qt Creator[3] er en række værktøjer brugt til at simplificere udviklingen af den brugergrænseflade der bliver vist på DevKit. Qt Creator består af en IDE hvori der er værktøjer til at designe brugergrænseflader samt udvikling af kode.

4.2.3 Arduino IDE

Arduino IDE'en[4] blev brugt til udviklingen af WiFi-funktionaliteten, som standard kan Arduino IDE'en ikke udvikle til WiFi modulet ESP8266[1] brugt til projektet, derfor blev der brugt et Community-created plugin[5] som gjorde det muligt.

4.2.4 Multisim

Multisim[6] er et værktøj brugt til design og simulering af kredsløb, som under udviklingen af Blow-Ball 3000 er blevet brugt til at designe kredsløbsdiagrammer.

4.2.5 Ultiboard

Ultiboard[7] er et værktøj til design og tjek af printplade. Det er muligt at importere designs fra multisim og efterfølgende tjekke om alle forbindelser er lavet, derved formindske chancen for fejl. Under udviklingen af Blow-Ball 3000 blev Ultiboard brugt til at designe printplader, men grundet problemer og tiden det tog at få produceret printplader er der ikke nogle færdige printplader på produktet.

4.2.6 Git

Git[8] er et versionsstyrings-værktøj der i dette projekt er blevet brugt til at holde styr på backup af samtlige filer. Git har under udviklingen af Blow-Ball 3000 været med til at tillade flere udviklere at arbejde på samme projekt/dokument uden alt for mange problemer.

4.2.7 Redmine

Redmine[9] er en service/hjemmeside der kan bruges til at holde styr på opgaver til et projekt, git-repositoriet, tid brugt m.m. Denne service stod tilgængelig til projektet fra IHA. Under udviklingen af Blow-Ball 3000 skulle Redmine have været brugt til at holde styr på alle opgaver, servicen er dog ikke specielt intuitiv/brugervenlig, så det blev ikke brugt i det omfang originalt tiltænkt.

4.2.8 L^AT_EX

L^AT_EX[10] er et opmærkningssprog brugt til tekstformatering af dokumenter, hvor alt bliver skrevet i plain-tekst. Under udviklingen af Blow-Ball 3000 er LaTeX blevet brugt til alle dokumenter som møde-referater, dokumentationen m.m. To af grundende til brug af LaTeX i projektet er at det fungerer sammen med Git og at det tillader flere udviklere at arbejde parallelt i samme dokument.

4.2.9 Microsoft Visio

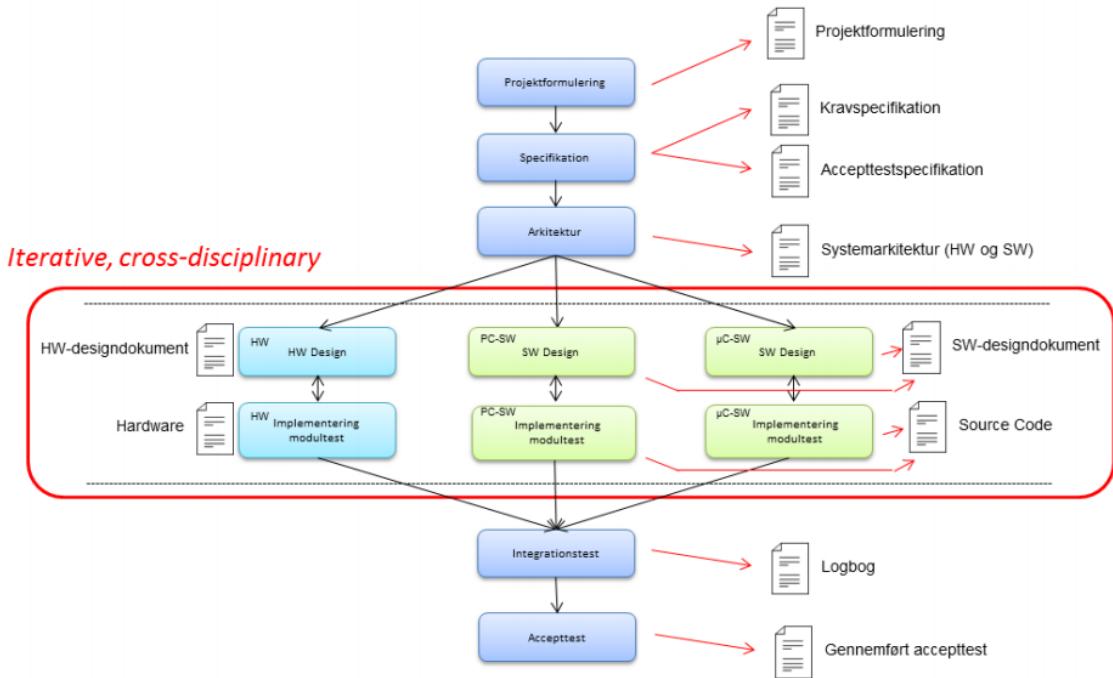
Microsoft Visio[11] er et værktøj til udvikling af diagrammer. Alle Visio dokumenter er vektor baseret, dvs. de aldrig bliver uklare. Derfor er alle diagrammer udviklet i forbindelse med Blow-Ball 3000 lavet i Visio.

4.2.10 Analog Discovery

Analog Discovery[12] er et bærbart oscilloskop og funktionsgenerator brugt til individuelt test af hardware enheder under udviklingen af Blow-Ball 3000.

4.3 Projektgennemførelse

Projektgennemførelsen tager udgangspunkt i ASE-modellen illustreret på figur 4.1



Figur 4.1. ASE udviklingsmodel[13]

ASE-modellen er en kombination af iterativ- og agil udvikling, som er meget anvendelig for et system bestående af både hardware og software og hvor der er et godt domænekendskab.

ASE-modellen er specielt anvendelig i projekter, hvori målene er veldefinerede og sikrer, at produktet, rapporten og dokumentationen udarbejdes med øje for de fastsatte læringsmål for 3. semesterprojekt. Dette semester har haft to fastlagte reviews af henholdsvis *kravspecifikation*, *testspezifikation* og *systemarkitektur* samt *implementation* og *proces*. Idet disse artefakter blev godkendt kunne den iterative fase påbegyndes. Heri arbejder man i iterationer med henblik på design, implementering og modultest. Disse samles i en accepttest og rapport samt dokumentation udarbejdes herefter.

I løbet af projektet har væsentlige dele fra SCRUM været anvendt til at styre processen og kommunikationen mellem gruppemedlemmerne. Én gang ugentligt blev der holdt vejledermøde, hvor vejleder blev informeret om projektets udvikling. To gange ugentligt har der været holdt SCRUM stand-up møde, hvor alle besvarede tre spørgsmål:

- Hvad lavede jeg i går?
- Hvad vil jeg lave i dag?
- Hvilke udfordringer har jeg?

Dette er valgt for at forstærke kommunikationen mellem de ansvarlige for de forskellige dele af produktet, for at sikre, at alle er med og ved hvad de skal give sig til. Det har vist sig at være meget gavnligt i forhold til erfaringer fra tidligere semestre, hvor for lidt

kommunikation resulterede i et adspredt og mislykket projekt. Alle gruppemedlemmer har derudover fulgt et SCRUM-kursus sideløbende med projektudviklingen.

Optimalt burde der være et dagligt stand-up møde, men dette har ikke været muligt eller nødvendigt, idet undervisning i andre fag også har krævet tid.

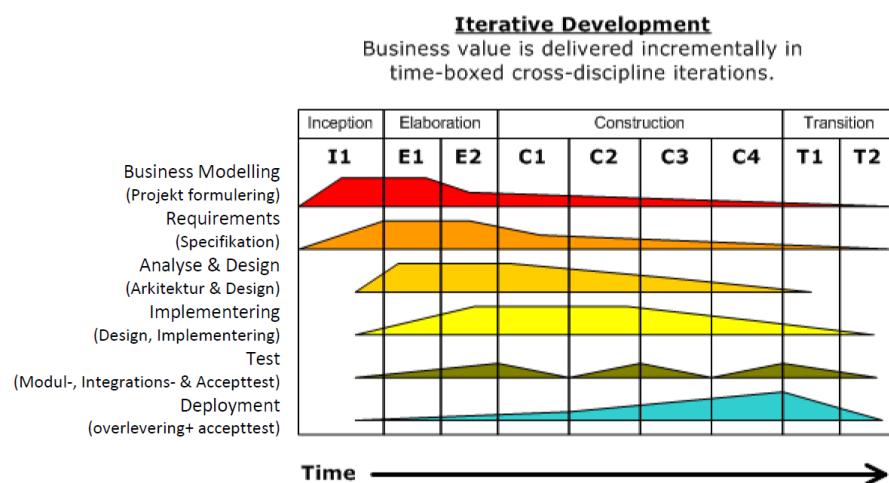
Ved rapport- og dokumentationsskrivning har alle elementer i rapporten været opdelt på et SCRUM-board i kategorierne "To-Do", "In progress" og "Done", hvilket har givet et godt overblik samt en struktureret arbejdsgang.

Projektet har været opdelt i syv sprints;

- Projektformulering
- Specifikation (+ review)
- 1. Construction sprint (+ review)
- 2. Construction sprint
- 3. Construction sprint
- 4. Construction sprint
- Rapport

Efter hvert sprints afslutning blev der holdt et opsamlingsmøde og næste sprint blev planlagt i forhold til arbejdsopgaver og tidsestimer.

Tidsplanen over projektgennemførselen kan ses i tabel 8.1 på side 41.



Figur 4.2. Udviklingsfaser og iterationer [13]

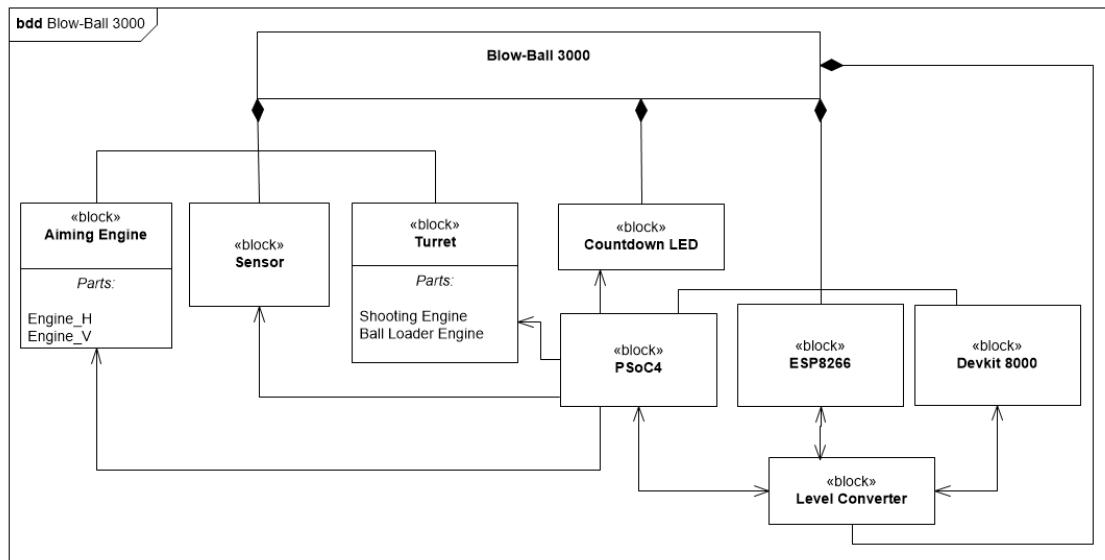
På figur 4.2 gives en visuel fremstilling af, hvordan iterationsfasen kan forløbe, hvor teksten i paranteserne angiver ASE-termer fra figur 4.1 på forrige side. Det ses hvordan de forskellige arbejdsopgaver ligger i løbet af projektudarbejdelsen.

Arkitektur 5

For at dokumentere arkitektur-delen af projektet, anvendes SysML og UML. Her i rapporten ses de overordnede diagrammer. For fuld arkitektur henvises der til kapitel 4 på side 17 i dokumentationen.

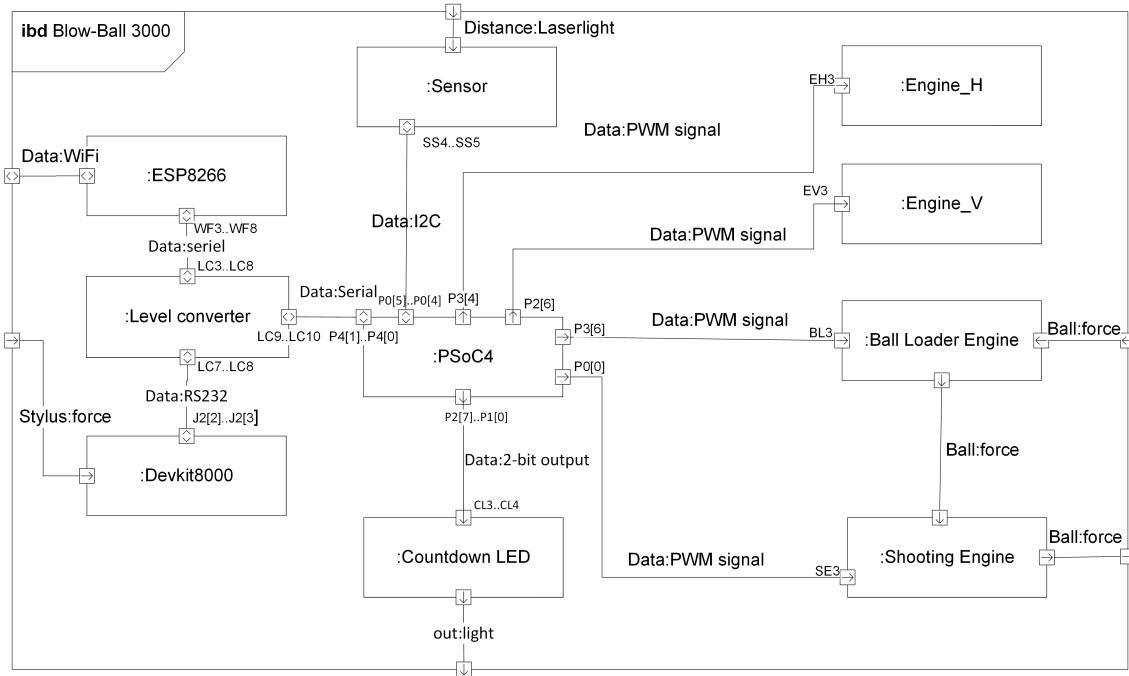
SysML og UML anvendes for at lette arbejdet i den senere design-fase. Hvor BDD og IBD anvendes primært under hardware og domænemodel, sekvensdiagram og state machine anvendes under udvikling af software.

Figur 5.1 viser et samlet Block definition diagram(BDD) for Blow-Ball 3000. BDD diagrammet bruges til at skabe oversigt over de hardware-dele produktet består af. Yderligere kan det også anvendes til at vise de indbyrdes afhængigheder, samt at udvikle et IBD-diagram ud fra.



Figur 5.1. BDD af Blow-Ball 3000

På figur 5.2 ses internal block diagram(IBD). Her ses de indbyrdes forbindelser, der er imellem enhederne. Derudover bruges det også til at definere hvilken type kommunikation der bruges mellem enheder. Dette er en stor fordel i det videre forløb, da alt kommunikation er synlig her, hvilket anvendes ti design af systemer.



Figur 5.2. IBD af Blow Ball 3000

Hardware design 6

I dette afsnit vil der blive gennemgået hvordan der i hardwaren er blevet arbejdet mod et funktionelt system. Ikke alt hardware der er implementeret vil blive kommenteret. Generelt for udregninger, kredsløbsdiagrammer og dybere forklaring henvises til kapitel 5 på side 35 i dokumentationen.

6.1 Servomotor

Der blev taget udgangspunkt i, at en stepermotor skulle styre den horisontale og vertikale position. Dette kunne ikke lade sig gøre og der måtte overvejes nye løsninger. Blandt disse var servomotoren, som blev implementeret på tårnet ved hjælp af en snor som kunne vippe kanonen når den roterede. Senere blev snoren erstattet af en plejlstang for at styrke y-aksens funktionalitet.

Da servomotorens position styres ved hjælp af elektriske impulsers tidsafstand, blev motoren testet ved at forsyne den med den specificerede forsyningsspænding og derefter regulere et PWM-signals duty-cycle ved hjælp af en funktionsgenerator. Forventningen var, at motorens position kunne bestemmes indenfor en 180 vinkel, ved en dutycycle mellem 5% og 10%. Testen viste dog, at denne type motor kan justeres indenfor ca 200 med en dutycycle mellem 3% og 12%.

En identisk servomotor er desuden blevet implementeret, som erstatter den oprindelige step-motor under tårnet. Her styres den horisontale akse af kanonen. Det største problem ved denne løsning, var stabiliteten af kanonen. Servomotorens rotor bestod af et gummihjul hvor på en servo-arm kunne fastmonteres. Tandhjulet under tårnet kunne derfor ikke fastgøres stabilt, eftersom pinolskruen i tandhjulet ikke kunne stramme på gummihjulet. Der er derfor monteret stabiliseringsspinde under tårnet. Efter test kan det konkluderes, at servomotoren kan rotere stabilt og præcist.

En tredje servomotor af samme type blev også implementeret til, at udføre boldindfødningsmekanismen. Her blev der testet hvilken position servomotoren kunne sættes i således, at den skulle foretage mindst mulig sving for, at der kunne åbnes og lukkes for bold-slisken hurtigst muligt. Dette forhindrer, at to eller flere bolde bliver skudt afsted samtidig.

6.2 DC-motor

Der anvendes DC-motorer til affyring af bolde. På disse er der monteret plasthjul, der tilfører boldene hastigheden. Der anvendes 90W's motorer, hvilket gør dem kraftige nok til at overholde de krav der er opstillet for projektet, når der affyres hoppebolde. Valget af DC-motorer er blevet taget idet de forholdsvis nemt kan implementeres og styres, eftersom de direkte omdanner elektrisk energi til rotationsenergi. De valgte DC-motorer skulle

samtidig være kraftige nok til, at skyde hoppebolde og være lette nok til, at de kunne løftes af en motor på den vertikale akse.

Til design af DC-motorstyringen blev der taget udgangspunkt i et udleveret print der kunne styre motorerne. Dette print indeholdt fire MOSFETs som kunne åbne og lukke, når de blev tilført et PWM-signal. Ved at regulere på dette signals duty-cycle, så kunne man ændre hastigheden af motorerne ved, at give motorerne en spænding på 12V.

Det udleverede print indeholdt dog fire MOSFETs hvilket var flere end der var brug for. Der blev derfor lavet et nyt styringsprint som kun indeholdt to MOSFETs - en til hver DC-motor. Derudover blev der tilføjet nye dioder på printet, idet den anvendte PSU tilførte flere ampere end de strømforsyninger der hidtil var brugt i laboratoriet.

De nye dioder kan klare en strøm på 3.0A.

Under test af styringsprintet opstod der dog problemer når der blev tilført strøm fra PSU. Et par gange lukkede de anvendte MOSFETs ned og måtte udskiftes. Det har ikke været muligt at teste, men grundten til dette menes at skyldes, at de brugte MOSFETs nåede deres thermal-shutdown temperatur. Dette skete hvis motorerne kørte for længe ved høje duty-cycles. Herefter slukkede MOSFET'erne og motorerne kørte uafhængigt af PWM-signalet.

Af samme grund kunne det heller ikke lade sig gøre, at køre motorerne ved 100% duty-cycle.

Udover brændte MOSFETs, så blev de brugte DC-motorer varme og ved længere tids kørsel lavede de skrabelyde. Dette skyldtes, at det signal der skulle drive motorerne var på 12V - selvom de kun var specificeret til 9.6V. Der blev derfor tilføjet effektmodstand med en størrelse på 1.5Ω i serie med motorerne. Dette nedsatte spændingen over motorerne og begrænsede den strøm der løb gennem dem. Der blev desuden tilføjet køleplader til de to MOSFETs.

Efter stress-test blev fundet, at motorerne kørte mere stabilt og de anvendte komponenter ikke bliver nær så varme. Motorerne kunne derfor også holde til, at PWM-signalet har en duty-cycle på 100%.

6.3 Sensor

Der blev i starten af projektet valgt en sonar-sensor til at måle afstanden fra kanonen til målet. Denne sensor blev dog senere droppet, da sonarsensorens virkemåde ikke stemte overens med de krav der var opsat i kravspecifikationen. Dette skyldes, at en sonarsensor udsender en kegleformet lydbølge. Keglen blev bredere med afstanden og opfangede derfor ikke præcist nok, når målet stod ud over en længere afstand.

Istedet for sonarsensoren blev der valgt en løsning der involverer en LIDAR-sensor. Ved brug af denne sensor, blev der udryddet alle problemer, der opstod omkring brugen af sonar-sensor. LIDAR-sensoren bruger en laserstråle til afstandsmålingen og vil derfor måle i den lige linje som strålen bevæger sig i. Den burde altså ikke opfange ting, udover det som laserstrålen rammer. LIDAR-sensoren kan desuden måle over længere afstande end sonar-sensore. Dog skete kommunikationen ved LIDAR-sensor over I2C protokolen, kontra analogt med sonar-sensor, hvilket betød at der nu var en væsentlig større kompleksitet ved afstandsmåling. Da afstandsmålingen bliver returneret vha. I2C protokol, vil støj ikke længere være et problem i samme forstand, som det var tilfældet for sonar.

6.4 LED-countdown

For at kunne signalere til brugeren, at systemet talte ned til affyring, blev der implementeret en LED. Hertil blev der valgt en RGBW-LED(Red-Green-Blue-White) som skulle blinke rødt i 8 sekunder indtil affyringen begyndte. Herefter skulle den lyse grøn. De brugte LED'er blev tændt og slukket ved hjælp af mosfets som kunne styres direkte fra PSoC. Derudover anvendes dette modul også som visuel fejlmeddelse på fejlet scanning.

6.5 PSU

Der er under projektet brugt en PSU (Power Supply Unit) af mærket "Energon"[14], til at forsyne systemet med strøm. Denne komponent er ikke implementeret for at opfylde en Use Case, men derimod for, at mindske antallet af ledninger til omverdenen og derved gøre systemet mere simpelt og brugervenligt. Den brugte PSU er koblet til systemet via et 24-pins atx-stik. Det er dog kun 4 af disse pins der ledes til systemet. Strømforsyningen er i stand til, at omforme 230V fra elnettet til 12V og 5V - samt deres tilhørende GND. Derudover bruges endnu et signal fra PSU kaldet "Power On". Dette signal skal kortsluttes med GND for, at PSU'en kan tænde. Denne kortslutning udføres med en switch på fordelerprintet (se afsnit 5.6 på side 45 i dokumentationen) og fungerer derfor som en on/off-switch for hele systemet. De resterende 20 signaler bruges ikke til noget og er derfor ikke forbundet til fordelerprintet.

Oprindeligt blev der også hentet 3.3V fra forsyningen. Efter test med denne udgang viste det sig dog, at den var for ustabil til, at kunne forsyne WiFi-modulet. Det brugte WiFi-modul, som krævede en forsyningsspænding på 3.3V, måtte derfor forsynes ved, at implementere en spændings-regulator som regulerede 5V ned til de nødvendige 3.3V. Den brugte 3.3V-udgang blev herefter fjernet fra fordelerprintet.

6.6 Level converter

Der er under projektet anvendt en level converter der skal sørge for at de spændingsniveauer, de forskellige kommunikations-enheder kan håndtere bliver overholdt. Til dette anvendes en MAX232 IC[15]. Denne er specielt designet til at konvertere fra spændingsniveauer fra Devkit til de niveauer PSoC maksimalt kan håndtere på indgangene. Netop på grund af PSoC er begrænset på acceptabel indgangsspænding, anvendes der en level converter.

6.6.1 WiFi-modul

For at kommunikere til systemet fra en online enhed, benyttes et WiFi-modul[1] af typen ESP8266, til Blow-Ball 3000. Processoren i modulet har mulighed for, at kommunikere over en række forskellige protokoller. I dette produkt er den sat op til at kommunikere over UART, således den kan kommunikere med PSoC og Devkit. WiFi-modulet er et færdigt produkt der anvendes.

6.7 Step-motor

Den oprindelige plan til styring af den horisontale position af tårnet var, at bruge en step-motor. Dette blev valgt idet en step-motor har mulighed for, at styre position forholdsvis præcist - givet at motoren kan rotere nogenlunde uhindret. Der blev testet forskellige step-motorer ved hjælp af et simpelt styringsprogram på en Mega32 µC fra Atmel[16] og et

styringsprint. Efter test af de forskellige stepmotorer, blev det konkluderet, at en bipolær step-motor af typen Wantai Mini Stepper var den sterkeste og hurtigste step-motor til rådighed. Den blev derfor implementeret under tårnet.

Kort tid før acceptttest blev systemet testet med alle hardware dele påmonteret. Det blev her konkluderet, at ledningerne fra fordelerprintet som er placeret på tårnet og ned til komponenter på bundpladen var nok til, at step-motoren ikke havde kræfter nok til, at dreje tårnet rundt. Step-motoren og step-motorstyringsprintet blev derfor afmonteret og fjernet som del af projektet.

Til styring af kanonens vertikale position var den originale plan, at bruge endnu en step-motor. Dette blev valgt af samme grund som styringen af den horisontale position - præcision. Den største udfordring her var, at denne motor skulle kunne holde igen, idet vægten af kanonen ville vippe tårnets position tilbage. Der blev ikke fundet nogen step-motor som var stærk nok til, at løfte kanonen og denne ide måtte opgives.

Det virker til at det generelle problem med step-motorer, ligger omkring det faktum at der ikke er trækkraft nok til at modstå påvirkninger på tværs af motoren. Under alle test af step-motor, blev belastningen placeret ned ovenpå den. Her havde stepmotoren ingen problemer med at håndtere en stor vægt. Derfor blev det antaget at step-motor løsningen ville være tilstrækkeligt. Dette blev senere afkræftet, efterhånden som vægten og ydre påvirkninger - såsom ledninger - blev forøget.

Software Design

7

I dette afsnit vil der blive beskrevet hvordan softwaren for brugergrænseflade, PSoC og Wifi er designet, implementeret og testet.

7.1 PSoC4

I projektet er PSoC4 blevet brugt til at styre følgende elementer:

- 3 Servomotorer; Engine_H, Engine_V og Ball Loader Engine
- 2 DC motorer; Shooting Engine
- LIDAR Sensor; Sensor
- Multicolor LED; Countdown LED
- Kommunikation med Devkit gennem UART.

Servomotorerne benyttes til at styre den horisontal akse, den vertikale akse og ball-loader mekanismen. Alle servomotorne styres af et PWM signal med en dutycycle mellem 3 % og 12 %, som beskrevet i afsnit 6.1 på side 26.

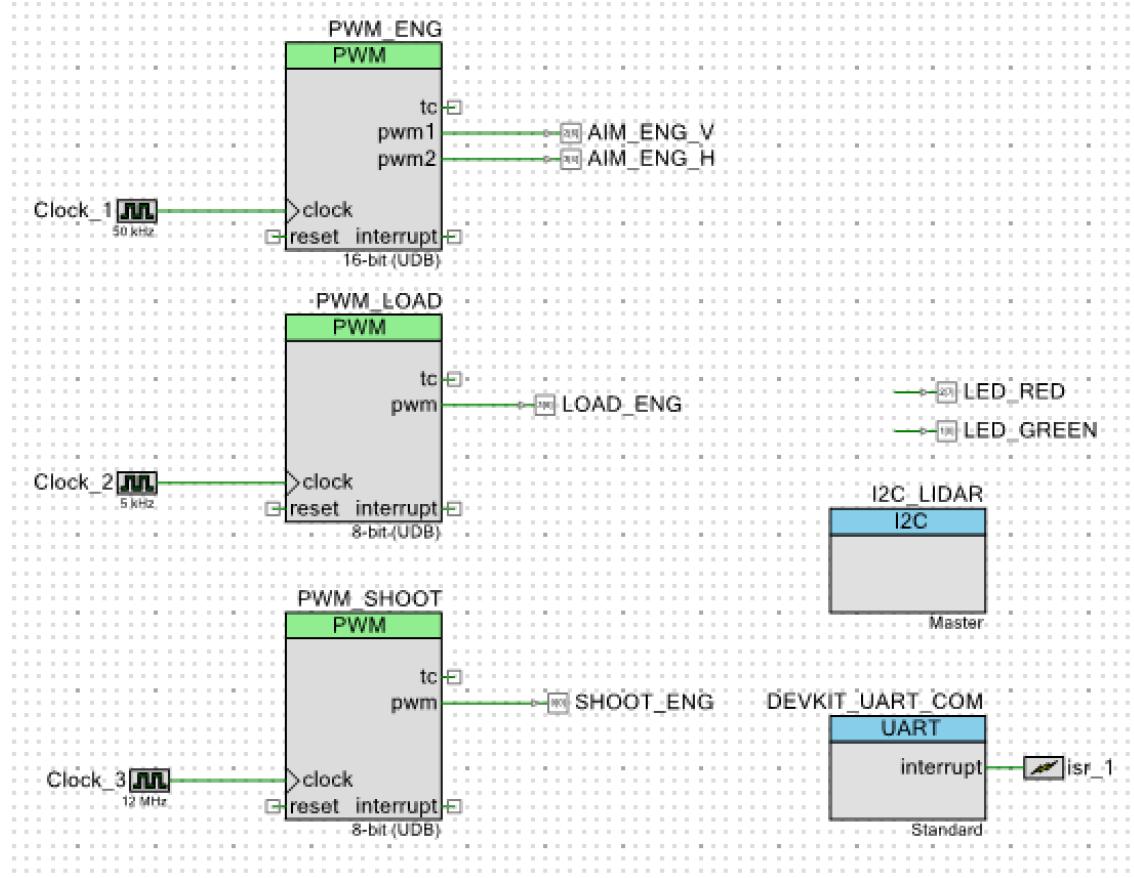
DC motorerne bruges til affyringsmekanismen. DC motorernes hastighed styres med et PWM signal med en dutycycle fra 0 % til 100 %.

Sensoren bruges til at aflæse afstanden til målet og kommunikerer med PSoC4 via I2C.

Countdown LED benyttes til nedtælling, så bruger ved hvornår bolden bliver affyret. LED'en styres med to IO pins, én for rød og én for grøn, som enten sættes lav(0) eller høj(1).

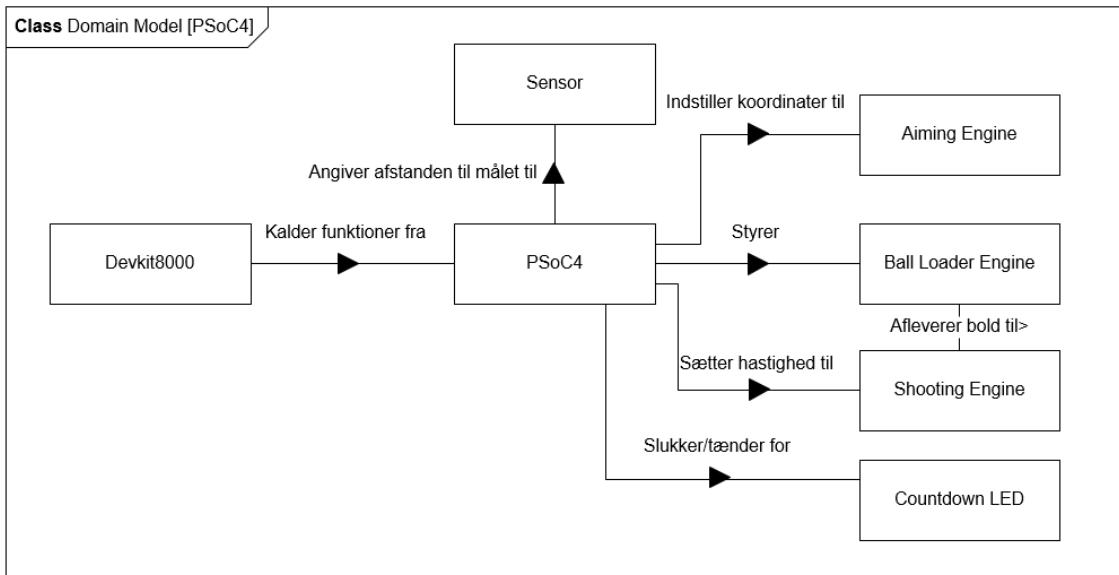
Kommunikation bliver uddybet i UART afsnittet, 7.4 på side 36.

Figur 7.1 viser topdesignet i PSoC Creator. Som beskrevet ovenfor, styres den horisontale og vertikale akse af PWM komponenten "PWM_ENG". Ball-loader mekanismen styres af "PWM_LOAD" og DC motorerne bliver styret af "PWM_SHOOT". Sensoren bliver styret af en I2C komponent i PSoC4 og UART bliver styret via en UART komponent.



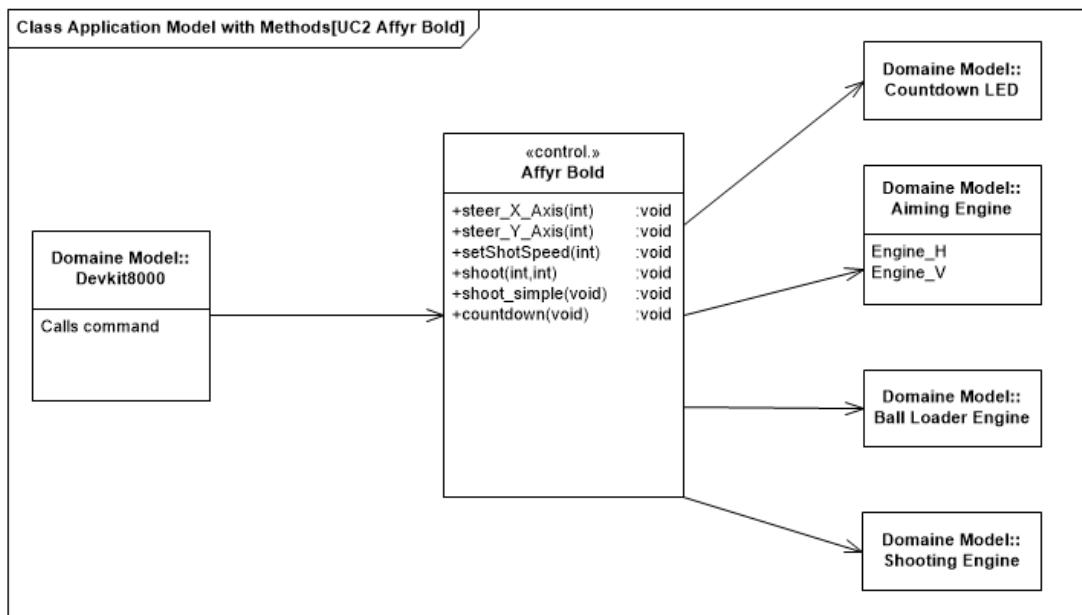
Figur 7.1. Topdesign for det færdige program til PSoC4

Domæne modellen er udarbejdet for at give et overblik over hvordan PSoC'en interagerer med de forskellige dele af systemet. 7.2



Figur 7.2. Domæne model for PSoC4

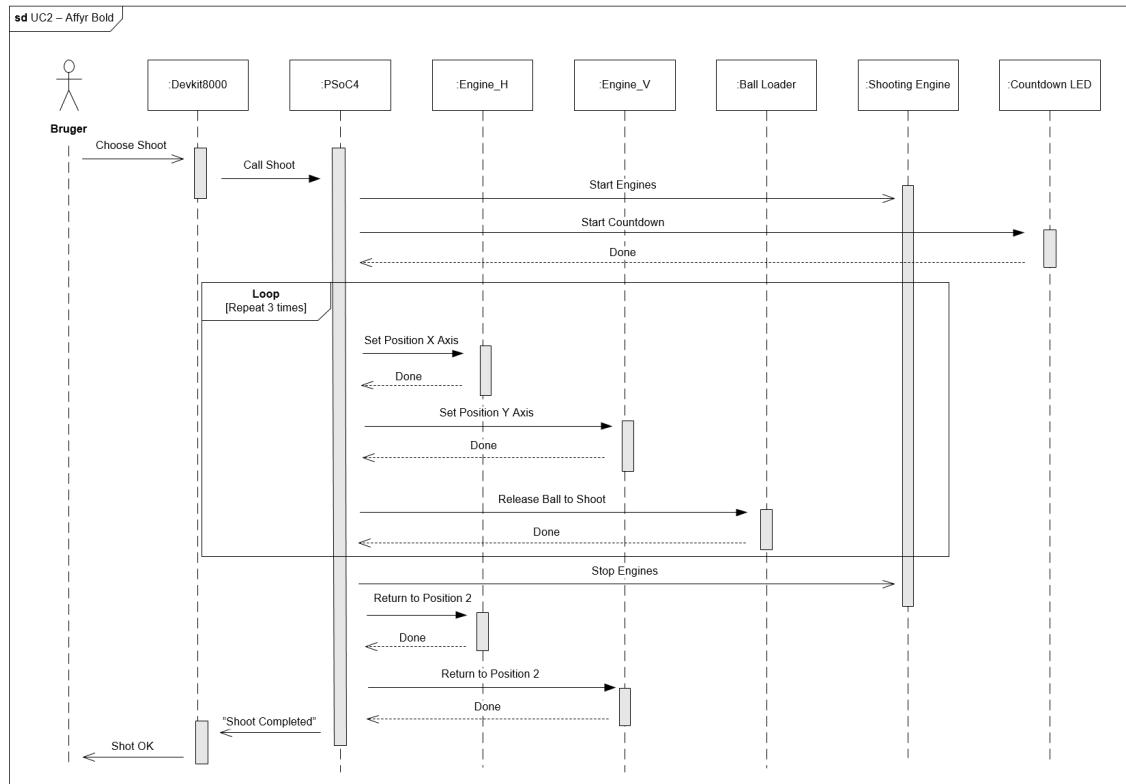
På baggrund af domæne-modellen er der blevet udarbejdet 2 applikationsmodeller med metoder for henholdsvis *Use Case 1: Scan mål* og *Use Case 2: Affyr Bold*. Applikationsmodellen bruges som bro mellem hvad PSoC'en skal gøre og hvordan det skal gøres. Applikationsmodellen for Use Case 2 kan ses på figur 7.3. Den anden applikationsmodel kan ses på figur 4.10 på side 30 i dokumentationen.



Figur 7.3. Applikations model for Use Case 2: Affyr Bold

Der er desuden også blevet udarbejdet et sekvensdiagram for Use Case 1 og 2. Sekvensdiagrammerne beskriver processerne i Use Case'ene for PSoC'ens funktionalitet.

På figur 7.4 ses sekvensdiagrammet for *Use Case 2: Affyr Bold*. Sekvensdiagrammet for Use Case 1 kan ses på figur 4.11 på side 31 i dokumentationen.



Figur 7.4. Sekvensdiagram for Use Case 2: Affyr Bold

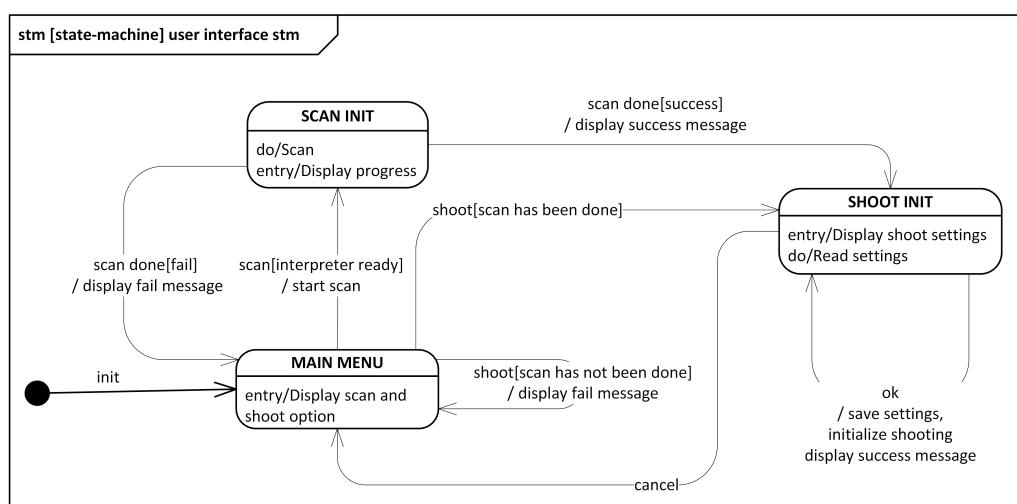
PSoC4 koden er udarbejdet i C og der er derfor ikke benyttet klasser, men de forskellige metoder er opdelt i CentralFunctions.h, LED.h, Sensor.h og EngineControl.h og deres tilhørende C-filer, for overskuelighed og bedre Doxygen-output[17].

7.2 User interface

User interfacet er brugerens grænseflade til styring af Blow-Ball 3000, og der er derfor lagt meget vægt på et enkelt og brugervenligt setup. Det er udviklet i Qt og derfor hovedsageligt skrevet i C++. Applikationen består af fem klasser og en main() funktion. De tre af klasserne står hovedsageligt for opsætning af user interfacet for hovedmenuen (MainMenu), scan-initialiseringen (ScanInit) og shoot-indstillinger (ShootInit). Herudover styrer klasserne Scan og Shoot henholdsvis scan- og affyringsfunktionerne. Der har blandt andet været brug for en intuitiv randomiseringsfunktion, som vil tage brugerinput fra ShootInit-klassens interface og ud fra disse beregne tre tilfældige felter der skal skydes i.

Hele applikationen er event-drevet og vil fungere på Devkit ved opstart af dette, så det er nemt at gå til.

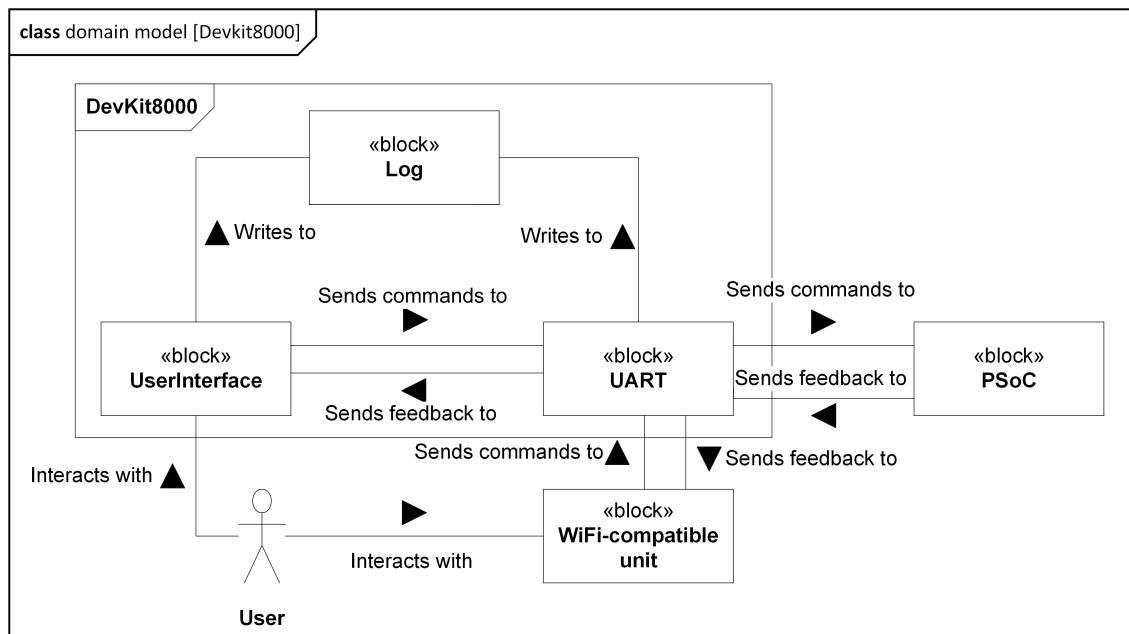
Et state machine diagram for DevKit applikationen kan ses på figur 7.5



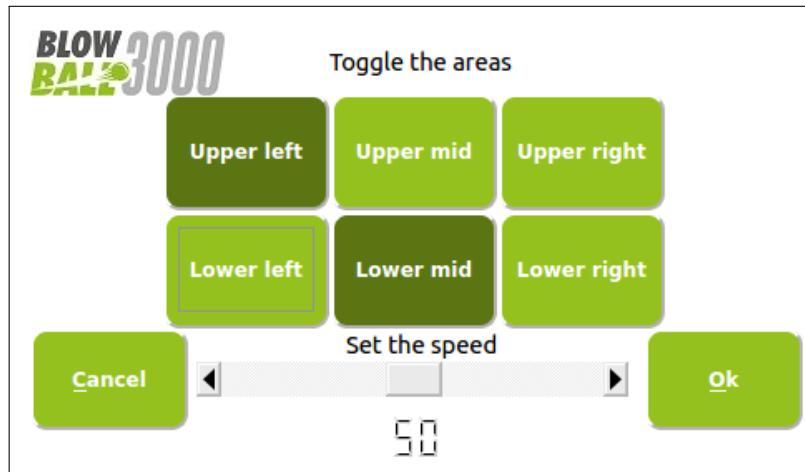
Figur 7.5. State Machine Diagram for den grafiske brugergrænseflade

Heri er det illustreret, hvordan brugeren kan bevæge sig rundt i de forskellige states. Der er valgt tre states for at simplificere interfacet, og brugeren vil derfor få feedback i form af pop-up bokse på devkittet. Disse er alle tilknyttet en timerfunktion, som vil lukke dem efter et givent interval, så de ikke blokerer applikationen, hvis brugeren eksempelvis kontrollerer systemet over WiFi.

Under udviklingen har det været muligt at skrive til en log fil, hvilket har simplificeret test af applikationen, og det vil derfor være muligt at tjekke op på, hvilke områder Blow-Ball 3000 vælger at skyde i, samt hvor i applikationen den eventuelt fejler. Dette har videre været anvendt i kommunikationsklasserne, hvilket domænemodellen, som kan ses på figur 7.6 også illustrerer. Det er her også muligt at få et enkelt overblik over, hvordan applikationen er relateret til de andre elementer samt *bruger*.

**Figur 7.6.** Domænemodel over PSC4

Et eksempel på den grafiske brugergrænseflade ses på figur 7.7, hvor logo er implementeret samt store knapper og let forståelige kommandoer.

**Figur 7.7.** Shoot-indstillinger som vist på Devkit. Det ses at knapperne Upper Left og Lower Mid er toggled, altså valgt aktive af brugeren, og farten på affyring er sat til 50 %.

7.3 ESP8266

ESP8266 er et WiFi modul der tillader brugeren, at kontrollere Blow-Ball 3000 gennem en hjemmeside, over WiFi som beskrevet i Use Case 3. For at gøre dette muligt skal programmet have følgende funktioner:

- Accesspoint
- DHCP-Server
- Webserver

Accesspoint tillader brugeren at forbinde til et WiFi netværk der kommer op ved navn "Blow-Ball 3000".

DHCP-Server uddeler ip-addresser til de WiFi-kompatible enheder der forbinder til Accesspointet.

Webserver opsætter en hjemmeside som brugeren har mulighed for at gå ind på efter der er forbundet til accesspointet. På hjemmesiden har brugeren mulighed for at kontrollere Blow-Ball 3000.

På hjemmesiden har brugeren mulighed for at trykke "Scan"eller "Shoot"som henholdsvis sætter kalibrering af mål og skydning i gang. Undervejs får brugeren feedback om der er sket en fejl under udførelsen af disse kommandoer. Designet som tillader disse funktioner kan ses på figur 7.8 på den følgende side.

For at kommunikere med Devkit bruges der UART som beskrevet i kapitel 7.4.

7.4 UART

UART er den protokol der er blevet brugt til at kommunikere mellem følgende enheder:

- ESP8266
- PSoC
- DevKit

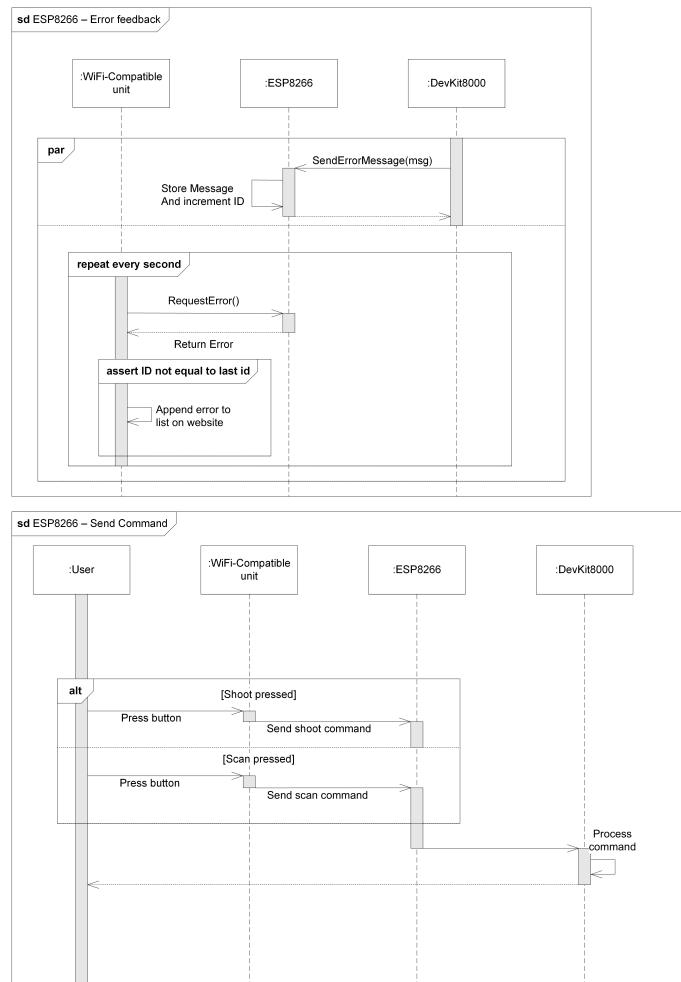
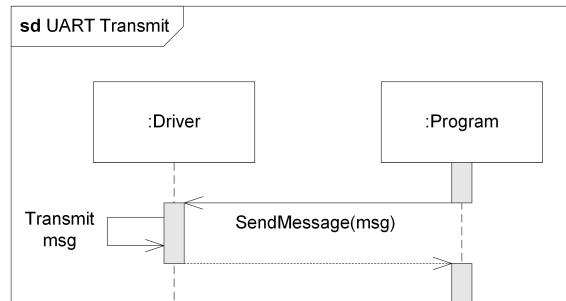
UART konfigurationen brugt til dette projekt kan ses i kapitel 4.8 på side 32 i dokumentationen. På figur 7.9 på næste side kan der ses hvordan UART kommunikationen er forbundet på Blow-Ball 3000. Hvor at DevKit læser og processere inputtet fra ESP8266 og sender det videre til PSoC'en efter at have tilføjet indstillingerne brugeren har sat i brugergrænsefladen på DevKit.

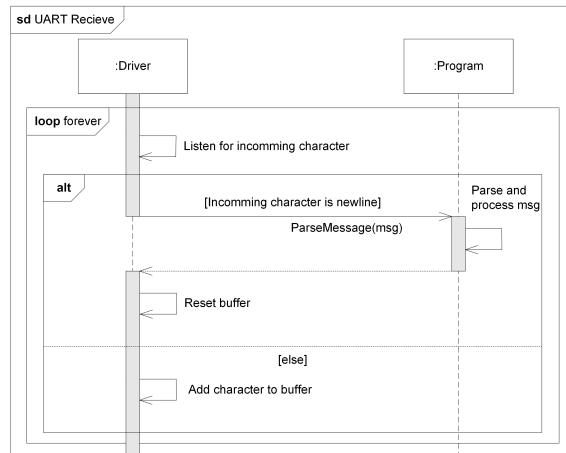
De forskellige enheder har alle en implementering der er baseret på designet der kan ses på figur 7.11 og 7.10 på den følgende side.

ESP8266:

Implementeringen af UART på ESP8266'en følger designet som der kan ses på figur 7.10 og 7.11 på side 38, Hvor at den skiftevis står og tjekker om der er blevet modtaget noget på UART'en eller om der er en HTTP-request der skal håndteres.

PSoC:

**Figur 7.8.** ESP8266 Sekvensdiagram**Figur 7.9.** UART-Kommunikation på Blow-Ball 3000**Figur 7.10.** UART send og modtag design



Figur 7.11. UART send og modtag design

UART implementeringen til PSoC blev, efter at have indset en potentiel fejl, ændret til at inkludere en cirkel-buffer. Grunden til dette er at nogle af de kommandoer PSoC'en skal udføre kan tage lang tid som f.eks. scan og shoot. Så i tilfældet at der bliver sendt flere kommandoer i træk uden PSoC når at proccessere dem, bliver de ældste overskrevet inden de er blevet håndteret af PSoC'en. Det blev løst ved at indsætte en cirkel-buffer som let kan udvides efter behov. For at spare på processor-kraft blev der i stedet for et loop, brugt et interrupt til at læse fra UART.

DevKit:

På DevKittet, som kører en linux-distribution, blev der gjort brug af en char-driver som allerede var lavet til at kunne læse/skrive bits over UART. Til at tjekke om der er modtaget noget fra UART er der startet en separat tråd der læser fra UART-driver filen. Designet brugt til dette er det samme som der kan ses på figur 7.10 og 7.11.

Resultater og diskussion

8

8.1 Resultater

Blow-Ball 3000 opnåede et tilfredsstillende resultat og er klar til brug, som det er skrevet i systembeskrivelsen. Alle elementer i accepttesten blev godkendt, den fulde accepttest kan ses i afsnit 7 på side 72 i dokumentationen.

For at optimere produktet kan følgende ting nævnes. Kalibrering af mål kunne gøres mere flydende dvs. akserne skal bevæge sig i mindre steps. Dette ville resultere i Blow-Ball 3000 vil kunne opfange målet på en større distance. Alt i alt er der opnået et tilfredsstillende resultat med henblik på kalibrering af mål. Affyring af bold kan forbedres ved at opgradere de nuværende motorer, således at Blow-Ball 3000 kan skyde på mål over større distancer. Affyring af bold virker tilfredsstillende med henblik på vores nuværende fastsatte krav. Kontrol af Blow-Ball 3000 over WiFi lever op til vores krav, dog kunne brugergrænsefladen gøres mere intuitiv som brugergrænsefladen på Devkit 8000. Denne grafiske brugergrænseflade giver en god brugeroplevelse, da den er let at finde rundt i og har et overskueligt design. Den færdige konstruktion kunne med fordel gøres mere robust med henblik på, at den ikke kommer til at stå i et beskyttet miljø. Konstruktionen er tilfredsstillende, da det er en prototype.

8.2 Tidsplan

Tidsplanen blev i grove træk overholdt, dog med små forsinkelser på grund af integrering af de forskellige delelementer ikke forløb som forventet. Det var svært at danne et overblik over tidsplanen, da værktøjet redmine ikke levede op til forventningerne. Sprintene gjorde det let at overholde tidsplanen, da der hele tiden blev sat nye deadlines og tilhørende arbejdsopgaver. I tabel 8.1 ses den overordnede tidsplan for projektet og i tabel 8.2 ses sprintene.

Subject	Re-spons-ible	EST. time /hours	Spent time /hours	% Do-ne	Start date	Due date	Closed	Release
Test af motorer	TS	5	4	100	07-12-2015	07-12-2015	07-12-2015	3. Construction sprint
HW level converter	TN	6		100	07-12-2015			
HW DC-styring	TN	3		100	07-12-2015			
HW Count-down_LED	LS		3	100	26-11-2015	04-12-2015	08-12-2015	3. Construction sprint

PSoC Enhedstest af kalibre-ring/sensor	SO	2	5	100	11-09-2015	20-11-2015	07-12-2015	3. Construction sprint
PSoC Enhedstest af ball-loader	SO	2	0.5	100	09-11-2015	20-11-2015	07-12-2015	3. Construction sprint
PSoC Enhedstest af vertikal motor	SO	2	1	100	09-11-2015	20-11-2015	12-07-2015	3. Construction sprint
PSoC Enhedstest af horisontal motor	SO	2	1	100	06-11-2015	13-11-2015	07-12-2015	2. Construction sprint
SW - UART (DevKit/P-SoC5)	JH	8	15	100	30-10-2015	11-11-2015	07-12-2015	2. Construction sprint
SW - ESP8266	JH	10	20	100	11-11-2015	25-11-2015	07-12-2015	3. Construction sprint
ME - Y Axis	LS		4	100	01-10-2015		09-11-2015	2. Construction sprint
ME - X axis	LS	10	4	100	01-10-2015		18-11-2015	2. Construction sprint
ME - Shoo-ter	LS		6	100	01-10-2015		26-11-2015	2. Construction sprint
HW - Ball loader	LS		5	100	01-10-2015		26-11-2015	2. Construction sprint
HW - Step- per motor driver	LS		12	100	01-10-2015		26-11-2015	2. Construction sprint
HW - DC Motor Driver	LS		4	100	01-10-2015		19-11-2015	2. Construction sprint
SW - Dev- Kit	CM		60	100	01-10-2015	08-12-2015	08-12-2015	
SW - PSoC	SO	20	16	100	01-10-2015		07-12-2015	
Accept test specifika- tion		3	3	100	07-09-2015		04-11-2015	Specifikation
Krav speci- fikation	CM	3	3	100	07-09-2015	10-09-2015	04-11-2015	Specifikation

Grænseflader	JH	4	2	100	07-09-2015		23-09-2015	Specifikation
IBD	LS	3	1	100	07-09-2015		05-10-2015	Specifikation
BDD	LS	1.5	0.5	100	07-09-2015		05-10-2015	Specifikation
Projekt-formulering	CM			100	01-09-2015	09-09-2015	09-09-2015	Projektformulering
Rige-billeder	CM			100	01-09-2015	09-09-2015	09-09-2015	Projektformulering
Fully dressed usecases	AB	1		100	01-09-2015		21-10-2015	Specifikation
MoSCoW	AB			100	31-08-2015	09-09-2015	09-09-2015	Projektformulering

Tabel 8.1. Tidsplanen for projektets emner

Releases:	Fra:	Til:
Projektformulering	31-08-2015	09-09-2015
Specifikation	09-09-2015	09-10-2015
1. Construction sprint + review	09-10-2015	30-10-2015
2. Construction sprint	30-10-2015	11-11-2015
3. Construction sprint	11-11-2015	25-11-2015
4. Construction sprint + acctest	25-11-2015	07-12-2015
Rapport	07-12-2015	16-12-2015

Tabel 8.2. Sprints

Opnåede erfaringer

9

9.1 Generelle erfaringer

Der er i dette projekt-forløb gjort mange erfaringer indenfor hardware, software og projektstyring. Et vigtigt redskab til dette har været SCRUM. Redskabet har ikke været anvendt i sin fulde udstrækning, men der er anvendt dele som blev fundet relevante for dette projekt. Der var ingen gruppemedlemmer der før dette projekt havde arbejdet med scrum. Derfor sidder gruppen tilbage med en følelse af generelt at have større viden indenfor SCRUM, herunder specielt hvordan det implementeres i gruppearbejde.

Generelt er den store udvikling sket indenfor gruppearbejdsprocesser, hvor gruppen har haft stort udbytte af at arbejde struktureret og under konstante deadlines.

9.1.1 Individuelle erfaringer

Alexander

Effektivisering af arbejde og minimering af spildtid har været et stort fokuspunkt i dette projekt, hvilket særligt er lykkedes takket være implementeringen af SCRUM i vores projekt, som jeg personligt har opnået stor erfaring med.

Derudover har jeg opnået stor viden indenfor PSoC og elementerne vi har benyttet i projektet, såsom I2C kommunikation, opsætning af PWM-signaler og generel kontrol af HW, som Servomotorer, stepmotorer og DC-motorer. En vigtig erfaring der er gjort, er at holde sig opdateret på alle dele af projektet, således at man har en grundlæggende forståelse for "hvad gør deres, hvad får jeg fra dem og hvad forventer de at få fra mig".

Cecilie

I dette projekt har jeg arbejdet mig hen imod en større forståelse for programmering af en eventdrevet grafisk brugergrænseflade. På den grafiske del har jeg oplevet at det har været nødvendigt selv at undersøge mange af de løsninger, som jeg har implementeret, men har også kunnet drage nytte af faget I3ISU, som er afviklet sideløbende med projektet.

En anden erfaring er vigtigheden af fortløbende integrationstest. Inden hardwaren blev helt færdig, har det været svært at teste, om programmet gjorde præcis det, den skulle, hvilket har skubbet tests og optimering af GUI hen i slutningen af projektfasen, hvilket kunne være afhjulpet ved flere tidlige tests af mindre funktionaliteter på produktet. I forhold til udvikling af produktet og samarbejde på tværs af projektgruppen er jeg blevet kendt med SCRUM, som jeg føler fremover kan anvendes til en struktureret arbejdsgang rettet imod produktet.

Jonas

En af de essentielle erfaringer jeg tager med videre, er at scrum-møder flere gange om ugen giver et godt indblik i projektets fremskridt, samt er det let at få opklaret om der nogle problemer man enten har brug for hjælp til, eller kan hjælpe med at løse dem. Undervejs brugte vi Git[8] til versions styring af filer, hvor jeg har fået en bedre forståelse for værktøjet og kan nu bruge værktøjet fra commandline, som giver mig en stor fordel til at rette konflikter i filer m.m. da et GUI ikke altid giver det ønskede resultat. Undervejs har vi gjort brug af mange forskellige nye værktøjer som Doxygen og Redmine, hvor at jeg har fået indsigt i hvordan man opsætter disse værktøjer til at hjælpe en i produktudviklingen. Jeg har undervejs åbnet mine øjne for essentielt det er at enheds- og intergrations-teste alle enheder når det muligt, da det kan mindste chancen for fejl hen mod projektets afslutning.

Lasse

I dette semesterprojekt har jeg opnået stor erfaring indenfor forskellige elektriske motorer og hhv software- og hardware-styring til disse. Jeg føler selv, at jeg nu kan bruge og indlejre step-, servo- og DC-motorer i fremtidigt arbejde. Jeg har meget konsekvent lavet enhedstests af de dele jeg har lavet til projektet, netop for at være sikker på at tingene virker enkeltvis, inden det bliver sat sammen med andre dele. Det har til dels virket. En vigtig læring for mig, har været, at selvom man kun er sat på nogle få delopgaver, er det vigtigt at have kendskab til resten af projektet for at være sikker på at alt er kompatibelt med hinanden. En anden vigtig erfaring fra dette projekt, har været at man i sin enhedstest skal have fuldstændig styr på enhedens spændings- og strømoptag, så beregningerne og funktionaliteten bliver præcis.

Vi har også benyttet os af begrebet "Kill your darlings", i forbindelse med arbejdet på stepermotorerne, da vi i lang tid var overbeviste om, at dette var den rigtige måde at styre den horisontale- og vertikale akse for tårnet. Vi måtte dog indse, at det ikke kunne lade sig gøre i praksis, da motoren ikke var stærk nok til at trække det komplette tårn, og istedet blev udskiftet med servomotorer.

Sarah

Ved dette projekt har jeg opnået stor viden indenfor programmering af PSoC, herunder de anvendte komponenter fra PSoC såsom PWM og ikke mindst anvendelse af dette til motorstyrring. Generelt er motorstyrring en af de ting, jeg har opnået meget stor viden om, da vi både har benyttet en step-, servo og DC-motor. Jeg har desuden også opnået viden om I2C kommunikation, hvilket blev benyttet til sensoren.

Med henblik på samarbejdet i projektgruppen samt selve udviklingsforløbet af projektet, har jeg lært at benytte SCRUM. Et værktøj jeg føler har bidraget positivt til projektet, ved at fremme kommunikationen mellem gruppemedlemmerne, samt til at fastlægge deadlines ved hjælp af sprints, således at man opnår en strukturet arbejdsproces.

Simon

Jeg har i dette projekt fået en større forståelse for PSoC4 og PSoC Creator, samt de PSoC4 komponenter vi har brugt. Min viden indenfor stepermotor, servomotor og DC-motorer,

samt I2C kommunikation er rykket sig en del, da disse dele har været styret af PSoC4 programmet. Jeg er blevet klogere på SCRUM, hvilket har gjort projektforløbet en del nemmere. De daglige stå-op-møder har gjort at man har haft større forståelse for hvad de andre lavede, samt hvor langt de var. Jeg har lært at man skal være mere konsekvent med at uddeletere tingene, da det opstod tvivl om det var PSoC4 eller Devkit8000 der stod for scanningsmetoden.

Thomas N

Jeg tager 2 ting med fra dette projekt.

Jeg har arbejdet med hardware-delen og dermed test af de forskellige motorer anvendt i projektet. Derudover også de styringer, der skal til disse. Når man arbejder med motorer har jeg gjort den erfaring, at det er vigtigt at stress-teste motoren udover de grænser, der indledningsvis opstilles. Det er vigtigt at teste motorene til en grænse, hvor man er sikker på at den uden problem kan magte opgaven. Derudover har jeg opdaget vigtigheden af at stress teste udviklede styringer, så man kan garantere en oppetid på længere end 30 sekunder. Den tid det tager at teste og opfatte en styring som værende færdig.

Derudover har jeg opdaget vigtigheden af en ordentligt styret projekt-gruppe. Dette være sig anvendelse af en projektleder, men i lige så høj grad en arbejdsproces, der skaber overblik og overskuelighed for gruppens medlemmer. Vi har anvendt dele af SCRUM, som arbejdsproces, hvilket har fungeret godt. Under SCRUM, har specielt stå-op møder og anvendelse af sprint været til stor glæde for mig.

Thomas R

Jeg har opnået erfaringer indenfor projektstyrelse i dette semesterprojekt. Dette er især sket gennem brugen af SCRUM i projektgruppen. En af de store hjælpemidler indenfor denne projektstyring er det såkaldte "stå-op-møde". Her fik man hurtigt et overblik over hvad de andre i gruppen lavede og hvorhenne vi var i projektarbejdet.

Jeg har i projektet arbejdet med motorerne. Herunder fandt vi ret sent i forløbet ud af, at den valgte step-motor ikke var stærk nok til, at dreje tårnet. Der blev lavet nogle desperate forsøg på, at få step-motoren til, at virke optimalt igen. Dette kunne dog ikke lade sig gøre og en ret grundlæggende del af systemet måtte skrottes. Dette gav også noget erfaring idet vi måtte være omstillingsparate.

Jeg har i undervisning i GFV lært om principperne bag DC-motorer og step-motorer. Viiden om disse motorer er gennem praktisk arbejde blevet udvidet gennem dette projektforløb.

9.2 Fremtidigt Arbejde

Under udarbejdelse af dette produkt, er alle krav opfyldt. Derfor vil videre arbejde med produktet forbedre funktionalitet med løsninger og enheder ikke beskrevet andre steder i rapporten her.

Den udviklede model er en prototype, så det første skridt på vejen ved videreudvikling, ville være at fremstille en færdig model i rigtig størrelse. Det vigtigste ved denne model, ville være at fremstille den med de krævede dimensioneringer for en fodbold. Dernæst at anvende motorer der kan yde en tilstrækkelig effekt til at affyre fodbolde. Det vil være af stor prioritet, ved fremstilling af færdig model, at selve konstruktionen foregår i materialer der udstråler professionalisme. Ydermere kan der drages nytte af professionelle håndværkere til opbygning af færdig produkt, hvilket vil give et mere robust produkt. Rent konstruktionsmæssigt, ville et mere minimalistisk design også gøre produktet kompakt og dermed lettere for både privatpersoner og klubber at opbevare.

Til det færdige produkt ville det være en stor fordel at anvende en berøringsskærm, der er mere sensitiv overfor input end den der er anvendt her. En ny berøringsskærm vil fjerne en stor del af brugeren og dermed give brugeren en bedre oplevelse. Dette skyldes at den gamle touchskærm var ekstrem udførlig, selv ved brug af pen. Derfor har det stor prioritet at anvende en anden touchskærm. Gerne en kapacativ type, kontra en resistiv, som der er anvendt i produktet. Yderligere vil det være smart at implementere alt software på blot én platform, i stedet for de 2 anvendte. Dette vil fjerne nogle problemer der kunne opstå ved kommunikation under støjfylde forhold.

For at optimere produktet ville det være optimalt at fremstille en dedikeret strømforsyning til formålet. Den nuværende er overdimensioneret både med hensyn til fysisk størrelse og tilslutningsmuligheder. Ved optimering af PSU, vil der spares en stor del af vægten. Ydermere vil produktet blive mere pladsbesparende. Eventuelt bruge et batteri, der kan levere nok effekt til at drive systemet. Dette vil gøre Blow-Ball 3000 flytbar.

Til videreudvikling af softwaren, vil det være muligt at tilføje mere funktionalitet til brugergrænsefladen. Her kunne man give brugeren mulighed for at tilpasse produktet i endnu større grad. Brugeren kunne få mulighed for selv at bestemme nedtællingstiden og tiden mellem hvert skud. Derudover vil det være muligt at inddæle målet i flere felter, således at man mere præcist kan bestemme hvor bolden lander hen. For at dette skulle lykkedes, ville det dog kræve præcise affyringsmotorer og positionsmotorer.

Produktet kunne udvides med præcise afstandsmålinger. Det vil gøre det muligt for Blow-Ball 3000 at blive uafhængig af afstanden, ved at foretage beregninger på afstanden og derved mere præcist ramme indenfor de rigtige felter. Her er det nødvendigt med bedre kontrol over motoren, da det kræver mere præcis affyringshastighed.

For at udvide funktionaliteten af Blow-Ball 3000 kunne et samarbejde med en målmandstræner være givet. Her kunne der indsættes forskellige moduler, der træner forskellige målmandsfærdigheder, det være sig høje indlæg, flade indlæg eller en helt tredje ting. Disse kunne vælges på brugergrænsefladen og dermed gøre trænerens job lettere. Under dette kunne der implementeres en højtaler, der giver præcise instruktioner til brugeren. Her kunne der både gives instruktioner om hvordan øvelsen udføres, hvad fokuspunkter for

målmanden skal være og opmunrende tilråb.

I forbindelse med førnævnte udvidelse, kan der også inddrages fagpersoner for markspillerområdet, hvor Blow-Ball 3000 kan bruges som oplægger til afslutningsøvelser. Hvor den kan anvendes som oplægger i forskellige scoringssituationer. Her ville der være mulighed for at vælge hvilken type afslutning der ønskes trænet. Dette valg kunne ligeledes foretages på GUI. Ved denne udvidelse vil produktet ramme en væsentlig større målgruppe, hvilket gør at produktet vil have større salgschancer. Dette skyldtes klubber og enkeltpersoner vil have større incitament for at erhverve Blow-Ball 3000, da anvendelsesmulighederne vil være store.

Konklusion 10

Blow-Ball 3000 var en succes, da samtlige mål og krav blev opfyldt for produktet ved test.

Grundet et effektivt team, med et klart mål for øje, samt en delvis anvendelse af SCRUM, blev spildtiden begrænset til et minimum, mens effektiviseringen var skruet i vejret.

Produktet var håndgribeligt fra start til slut for alle parter, og netop dette faciliterede yderligere overblikket over de nødvendige arbejdsopgaver.

Dermed kunne gruppen opdeles i mindre hold, som hver især specialiserede sig i deres element med en daglig kommunikation imellem holdene, således at signaler og koblinger imellem elementerne hele tiden var korrekte og opdaterede, selvom elementer undervejs måtte udskiftes og opbygges på ny.

Det kan konkluderes, at det er lykkes at fremstille en prototype, hvorfra der kan affyres bolde, der kan anvendes som målmandstræner. Denne prototype kan scanne et mål og affyre tre bolde imod målet.

Blow-Ball 3000 vil uden problemer kunne skaleres op med det nuværende design, samt udvide sin funktionalitet, såfremt dette skulle være ønskeligt. Dermed er det ligeledes lykkedes at lave et simpelt systemet, med høj mulighed for videre-udvikling.

Litteratur

- [1] Espressif systems. *ESP8266*, 2013. Bilag: ESP8266-01.pdf.
- [2] Cypress Semiconductor Corporation. Psoc creator, 2015. URL <http://www.cypress.com/products/psoc-creator-integrated-design-environment-ide>. Last Visited d. 8/12-2015.
- [3] The Qt Company Ltd. Qt creator manual 3.5.0, 2015. Last Visited 04/12-2015.
- [4] Arduino. Arduino 1.6.6 IDE, 2015. URL <https://www.arduino.cc/en/Main/Software>. Last Visited d. 7/12-2015.
- [5] ESP8266 Community Forum. ESP8266 Core for Arduino, 2015. URL <https://github.com/esp8266/Arduino>. Last Visited d. 7/12-2015.
- [6] National Instruments. Multisim, 2015. URL <http://www.ni.com/multisim/>. Last Visited d. 10/12-2015.
- [7] National Instruments. Ultiboard, 2015. URL <http://www.ni.com/ultiboard/>. Last Visited d. 10/12-2015.
- [8] Linus Torvalds. Git, 2015. URL <https://git-scm.com/>. Last Visited d. 8/12-2015.
- [9] Jean-Philippe Lang. Redmine, 2015. URL <https://www.redmine.org/>. Last Visited d. 8/12-2015.
- [10] Leslie Lamport. Latex, 2015. URL <https://www.latex-project.org/>. Last Visited d. 10/12-2015.
- [11] Microsoft. Visio, 2015. URL <https://products.office.com/en/visio-flowchart-software>. Last Visited d. 10/12-2015.
- [12] Digilent. Analog Discovery, 2015. URL www.digilentinc.com/analogdiscovery/. Last Visited d. 10/12-2015.
- [13] Peter Høgh Mikkelsen. *Vejledning til udviklingsprocessen for projekt 3 Ver. 1.10*. 2015.
- [14] Inter tech Elektronik Handels GmbH. *Energon EPS-750W*, 2006. Bilag: EPS-750-e.pdf.
- [15] Texas Instrument. *MAX232x datasheet*, 2014. Bilag: max232.pdf.
- [16] Atmel. *STK500 user guide*, —. Bilag: STK500.pdf.
- [17] *Kode dokumentation*, 2015. Bilag: CodeDocumentation.pdf.

Rettelser