3. Semesterprojekt - Goofy Candy Gun
Dokumentation - Gruppe 3

Rieder, Kasper
201310514

Jensen, Daniel V.
201500152

Nielsen, Mikkel
201402530

Kjeldgaard, Pernille L.
PK94398

Konstmann, Mia
201500157

Kloock, Michael
201370537

Rasmussen, Tenna
201406382

19. marts 2016

Indhold

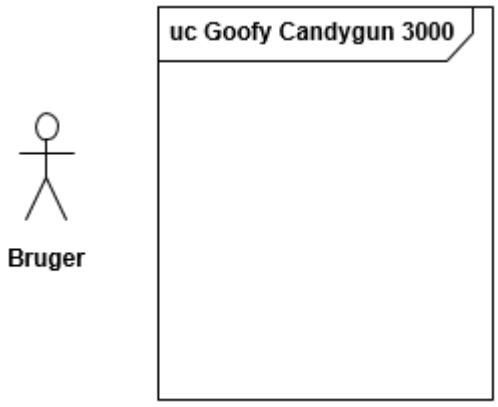
Figurer

1 Kravspecifikation

Det følgende afsnit udpensler projektet ved specifikation af aktører, use cases, samt ikke-funktionelle krav.

1.1 Aktør kontekst diagram

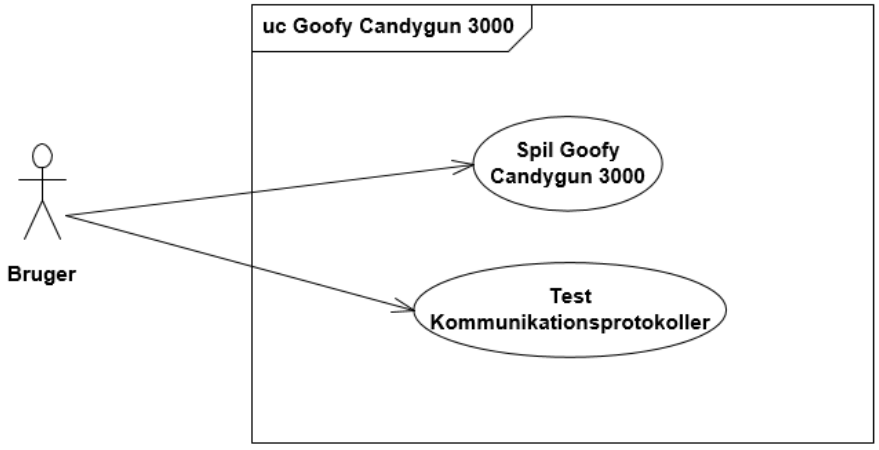
Figur 1 viser et kontekst diagram for Goofy Candygun 3000.



Figur 1: Kontekst diagram for slikkanonen

1.2 Use Case Diagram

Figur 2 viser et use case diagram for Goofy Candygun 3000.



Figur 2: Use case diagram for slikkanonen

1.3 Aktør beskrivelse

Det følgende afsnit beskriver de identificerede aktører for Goofy Candygun 3000.

1.3.1 Aktør - Bruger

Aktørens Navn:	Bruger
Alternativ Navn:	Spiller
Type:	Primær
Beskrivelse:	Brugeren initierer Goofy Candy Gun, ved at vælge spiltype på brugergrænsefladen. Derudover har brugeren mulighed for at stoppe spillet igennem brugergrænsefladen. Brugeren vil under spillet interagere med Goofy Candy Gun gennem Wii-Nunchucken. Brugeren starter også Goofy Candy Gun system-testen for at verificere om det er operationelt.

1.4 Fully Dressed Use Cases

Det følgende afsnit indeholder de *fully dressed use cases* for Goofy Candy Gun, som kan findes under afsnittet **Use Case Diagram**.

1.4.1 Use Case 1 - Spil Goofy Candy Gun 3000

Navn	Spil Goofy Candygun 3000
Mål	At spille spillet
Initiering	Bruger
Aktører	Bruger
Antal samtidige forekomster	Ingen
Prækondition	Spillet og kanonen er operationel. UC2 Test kommunikationsprotokoller er udført
Postkondition	Brugeren har færdiggjort spillet
Hovedscenarie	1. Bruger vælger spiltype på brugergrænseflade 2. Bruger vælger antal skud til runde 3. Bruger fylder magasin med slik tilsvarende antal skud 4. Bruger indstiller kanon med analogstick på Wii-nunchuck 5. Bruger udløser kanonen med Wii-nunchucks trigger 6. System lader et nyt skud 7. Brugergrænseflade opdateres med spillets statistikker 8. Punkt 4 til 7 gentages indtil skud er opbrugt [Extension 1: Bruger vælger 2 player mode] [Extension 2: Bruger afslutter det igangværende spil] 9. Brugergrænseflade viser afslutningsinfo for runden 10. Bruger afslutter runde 11. Brugergrænseflade vender tilbage til starttilstand
Udvidelser/ undtagelser	[**Extension 1: Brugeren vælger 2 player mode**] 1. Bruger overdrager Wii-nunchuck til den anden bruger 2. Punkt 4 til 7 gentages indtil skud er opbrugt 3. Use case genoptages fra punkt 8 [**Extension 2: Bruger afslutter det igangværende spil**] 1. Brugergrænseflade vender tilbage til starttilstand 2. Use case afsluttes

1.4.2 Use Case 2 - Test Kommunikationsprotokoller

Navn	Test kommunikationsprotokoller
Mål	At teste kommunikations protokoller
Initiering	Bruger
Aktører	Bruger
Antal samtidige forekomster	Ingen
Prækondition	Systemet er tændt
Postkondition	Systemet er gennemgået testen og resultaterne er vist

Hovedscenarie	
	1. Bruger vælger test system på brugergrænseflade 2. Devkit sender start SPI test til PSoC0 via SPI 3. PSoC0 sender acknowledge til Devkit via SPI [Exception 1: PSoC0 sender ikke acknowledge] 4. Brugergrænseflade meddeler om gennemført SPI test 5. Devkit sender start I2C test til PSoC0 via SPI 6. PSoC0 sender start I2C test til PSoC slaver via I2C 7. PSoC slaver sender acknowledge til PSoC0 via I2C [Exception 2: PSoC slaver sender ikke acknowledge] 8. PSoC0 meddeler om gennemført I2C test til Devkit via SPI 9. Brugergrænseflade meddeler om gennemført I2C test 10. Brugergrænseflade anmoder bruger om at trykke på knap 'Z' på Wii-nunchuck 11. Wii-nunchuck sender besked "Knap Z trykket"til PSoC2 via I2C [Exception 3: Wii-nunchuck sender ikke "Knap Z trykket"] 12. PSoC2 sender besked om "Knap Z trykket"til Psoc0 via I2C 13. PSoC0 videresender besked om "Knap Z trykket"til Devkit via SPI 14. Brugergrænseflade meddeler om gennemført Wii-nunchuck test 15. Brugergrænseflade meddeler at test af kommunikationsprotokoller er gennemført

Udvidelser/ undtagelser	[**Exception 1: PSoC0 sender ikke acknowledge**] 1. Brugergrænseflade meddeler fejl i SPI kommunikation 2. UC2 afsluttes [**Exception 2: PSoC slaver sender ikke acknowledge**] 1. PSoC0 sender fejlmeddelse til Devkit 2. Brugergrænseflade meddeler fejl i I2C kommunikation 3. UC2 afsluttes [**Exception 3: Wii-nunchuck sender ikke "Knap Z trykket"**] 1. PSoC2 sender fejlmeddelse til PSoC0 2. PSoC0 videresender fejlmeddelse til Devkit 3. Brugergrænseflade meddeler fejl i I2C kommunikation med Wii-nunchuck 4. UC2 afsluttes

1.5 Ikke funktionelle krav

1. Kanonen skal kunne drejes med en nøjagtighed på ± 5 °
   1.1. Vertikalt gælder dette for intervallet fra 0 til 70°
   1.2. Horizontalt gælder dette for intervallet fra -45 til 45°
2. Kanonen skal kunne affyre projektiler med en diameter på 1,25 cm ± 2 mm
3. Kanonen skal kunne affyre sit projektil minimum 1 meter
4. Kanonens størrelse må maksimalt være 40cm høj, bred og dyb
5. Fra aftryk på trigger til affyring må der maksimalt gå ti sekunder
6. Affyring af kanonen skal kunne afvikles minimum tre gange pr. minut
7. Figur 3 viser en skitse af hvordan den grafiskbrugergrænseflade kommer til at se ud

Figur 3: Skitse af brugergrænsefladen

2 Accepttestspecifikation

2.1 Use case 1 - Hovedscenarie

Step	Handling	Forventet observation/resultat	Faktisk observation/resultat	Vurdering (OK/FAIL)
1	Vælg one-player mode.	Brugergrænsefladen viser spilside for one-player mode og anmoder om valg af antal skud.		
2	Vælg ti skud.	Brugergrænseflade anmoder om, at der fyldes ti stykker slik i magasin.		
3	Fyld ti stykker slik i magasinet og tryk på knap for at starte spil.	Brugergrænseflade går til spilside og anmoder om, at kanon indstilles.		
4	Indstil kanon til affyring med Wii-nunchuck.	Kanon indstiller sig svarende til Wii-nunchucks placering.		
5	Udløs kanon med trigger på wii-nunchuck.	Kanon udløses.		
6	Gentag punkt 4 og 5 ti gange.	Punkt 4 og 5 gentages.		
7	Kig på brugergrænsefladen.	Brugergrænsefladen viser info om spillet.		
8	Tryk på knap for at vende tilbage til starttilstand.	Brugergrænseflade vender tilbage til startside.		

2.1.1 Use case 1 - Extension 1

Step	Handling	Forventet observation/resultat	Faktisk observation/resultat	Vurdering (OK/FAIL)
1	Vælg two-player mode.	Brugergrænsefladen viser spilside for two-player mode og anmoder om valg af antal skud.		
2	Vælg ti skud på brugergrænseflade.	Brugergrænseflade anmoder om, at der fyldes ti stykker slik i magasin.		
3	Fyld ti stykker slik i magasinet og tryk på knap for at starte spil.	Brugergrænseflade går til spilside og anmoder om, at kanon indstilles.		
4	Indstil kanon til affyring med Wii-nunchuck.	Kanon indstiller sig svarende til Wii-nunchucks placering.		
5	Udløs kanon med trigger på wii-nunchuck.	Kanon udløses.		
6	Giv Wii-nunchuck til den anden spiller.	Den anden spiller modtager Wii-nunchuck.		
7	Gentag punkt 4 til 6 indtil skud er opbrugt.	Punkt 4 til 6 gentages.		
8	Kig på brugergrænseflade.	Brugergrænseflade viser info om spil.		
9	Tryk på knap for at vende tilbage til starttilstand.	Brugergrænseflade vender tilbage til startside.		

2.1.2 Use case 1 - Extension 2

Step	Handling	Forventet observation/resultat	Faktisk observation/resultat	Vurdering (OK/FAIL)
1	Vælg one-player mode.	Brugergrænsefladen viser spilside for one-player mode og anmoder om valg af antal skud.		
2	Vælg ti skud på brugergrænseflade.	Brugergrænseflade anmoder om, at der fyldes ti stykker slik i magasin.		
3	Fyld ti stykker slik i magasinet og tryk på knap for at starte spil.	Brugergrænseflade går til spilside og anmoder om, at kanon indstilles.		
4	Tryk på knap for afslutning af spil.	Brugergrænseflade vender tilbage til startside.		

2.2 Use case 2 - Hovedscenarie

Step	Handling	Forventet observation/resultat	Faktisk observation/resultat	Vurdering (OK/FAIL)
1	Tryk start test på brugergrænseflade	Brugergrænsefladen udskriver at SPI og I2C testen er godkendt. Brugergrænsefladen anmoder bruger om tryk på Z på Wii-nunchuck		
2	Tryk Z på Wii- nunchuck	Brugergrænsefladen udskriver at Wii-testen er godkendt		

2.2.1 Use case 2 - Exception 1

Step	Handling	Forventet observation/resultat	Faktisk observation/resultat	Vurdering (OK/FAIL)
1	Fjern SPI-kablet fra DevKittet.			
2	Tryk på start test på brugergrænseflade	Brugergrænsefladen udskriver SPI forbindelses fejlmeddelelse.		

2.2.2 Use case 2 - Exception 2

Step	Handling	Forventet observation/resultat	Faktisk observation/resultat	Vurdering (OK/FAIL)
1	Fjern I2C-kabler fra alle I2C slaver.			
2	Tryk på start test på brugergrænseflade	Brugergrænsefladen udskriver I2C forbindelses fejlmeddelelse.		

2.2.3 Use case 2 - Exception 3

Step	Handling	Forventet observation/resultat	Faktisk observation/resultat	Vurdering (OK/FAIL)
1	Disconnect Wii nunchucken fra systemet.			
2	Tryk på start test på brugergrænseflade			
3	Vent på timeout.	Brugergrænsefladen udskriver Wii Nunchuck forbindelses fejlmeddelelse		

2.3 Ikke-funktionelle krav

Krav	Test	Forventet observation/resultat	Faktisk observation/resultat	Vurdering (OK/FAIL)
1.1	Bruger styrer kanon fra "top"position til "bund"posiion, og måler vinkelforskellen.	Den afmålte vinkelforskel må være 70 °± 5 °		
1.2	Bruger drejer kanonen fra længst til højre til længst til venstre og måler vinkelforskellen.	Den afmålte vinkelforskel ligger indenfor 70 °± 5 °		
2	Et projektil på 1.25 cm i diameter ± 5mm affyres fra kanonen.	Projektilet bliver affyret		
3	Et projektil affyres, og distancen mellem kanonen og stedet hvor projektilet lander måles.	Distancen er blevet målt til at være større end 1 meter.		
4	Mål kanonens dimensioner med en lineal.	Dimensionerne overstiger ikke 40cm x 40cm x 40cm.		
5	Tryk på "triggeren"på Wii Nunchuck, og mål med et stopur hvor lang tid der går fra tryk, til kanonen bliver affyret.	Den målte tid er mindre end 10 sekunder.		
6	Kanonen affyres 3 gange, og et stopur startes ved første skud, og stoppes ved det tredje skud.	Den målte tid er mindre end 60 sekunder.		

3 Systemarkitektur

3.1 Blokbeskrivelse

DevKit 8000
DevKit 8000 er en embedded Linux platform med touch-skærm der bruges til brugergrænsefladen for produktet. Det er her hvor brugeren interagerer med systemet og ser status for spillet.

Motorstyring
Motorstyring er blokken som består af Candy Gun 3000's motorerer - brugt til at styre den - samt *PSoC1*, som bruges til styring af disse motorer.

Wii-Nunchuck-Styring
Wii-Nunchuck-Styring er blokken som består af den fysiske Wii-Nunchuck controller der bruges af brugeren til at styre kanonen, samt *PSoC2*, som bruges til at videresende I2C dataen fra controlleren.

Wii-Nunchuck
Wii-Nunchuck er controlleren brugeren styrer kanonen med.

Motor
Motor blokken er Candy Gun 3000's motorerer der bruges til styring af kanonen i forskellige retninger.

PSoC0
PSoC0 er PSoC hardware der både er I2C master og SPI slave. Denne PSoC fungerer som bindeled mellem resten af systemets hardware, så kommunikation er muligt.

PSoC1
PSoC1 er PSoC hardware der bruges til softwarestyring af Candy Gun 3000's motorerer samt affyringsmekanisme.

PSoC2
PSoC2 er PSoC hardware der bruges til at videresende input data fra Wii-Nunchuck controlleren.

SPI (FlowSpecification)
SPI (FlowSpecification) beskriver signalerne der indgår i *SPI* kommunikation.

I2C (FlowSpecification)
I2C (FlowSpecification) beskriver signalerne der indgår i *I2C* kommunikation.

3.2 Signalbeskrivelse

Generelt for signalbeskrivelsen gælder, at når et signal beskrives som 'højt' menes der i et spændingsområde på 3.5V til 5 V, som er defineret for CMOS kredse (Kilde allaboutcircuits.com/textbook/digitall/chpt-3/logic-signal-voltage-levels/). På samme måde er signaler beskrevet som 'lav' defineret som spændinger indenfor 0 V til 1.5 V.

Blok-navn	**Funktionsbeskrivelse**	**Signaler**	**Signalbeskrivelse**

Devkit8000	Fungerer som grænseflade mellem bruger og systemet.	masterSPI	Type: SPI Spændingsniveau: 0-5V Hastighed: ??
		touch	Type: touch Tryk på DevKit8000 display.
PSoC0	Fungerer som I2C master for systemet samt SPI slave til DevKit8000.	slaveSPI	Type: SPI Spændingsniveau: 0-5V Hastighed: ??
		masterI2C	Type: I2C Spændingsniveau: 0-5V Hastighed: 100kbit/sekund
Motorstyring	Modtager input fra Wii-Nunchuck og omsætter det til PWM signaler.	motorSlave	Type: I2C Spændingsniveau: 0-5V Hastighed: 100kbit/sekund Beskrivelse: Indeholder Wii-Nunchuck data der skal bruges til motorstyring.
		power	Type: V_{CC} Spændingsniveau: 5V Beskrivelse: Strømforsyning til motorstyringen.
PSoC1	Modtager input fra Wii-Nunchuck og omsætter det til PWM signaler.	MotorI2C	Type: I2C Spændingsniveau: 0-5V Hastighed: 100kbit/sekund Beskrivelse: Indeholder formatteret Wii-Nunchuck data som skal bruges til styring af motorens PWM signal.

		PWM	Type: PWM Frekvens: 22kHz PWM %: 0-100% Spændingsniveau: 0-5V Beskrivelse: PWM signal til styring af motorens hastighed. Udregnet ud fra MotorI2C signalet.
Motor	Motorerne der skal styre kanonen	PWM	Type: PWM Frekvens: 22kHz PWM%: 0-100% Spædingsniveau: 0-5V Beskrivelse: PWM signal til styring af motorens hastighed.
		power	Type: V_{CC} Spændingsniveau: 12V Beskrivelse: Strømforsyning til motorstyringen
PSoC2	Modtager input data fra Wii-Nunchuk og videresender det i behandlet format.	wiiSlave	Type: I2C Spændingsniveau: 0-5V Hastighed: 100kbit/sekund Beskrivelse: Sender input data fra Wii-Nunchuck til PSoC2.
		WiiI2C	Type: I2C Spændingsniveau: 0-5V Hastighed: 100kbit/sekund Beskrivelse: Videresender behandlet Wii-Nunchuk data til andre dele af systemet.

Wii-nunchuck	Den fysiske controller som brugeren styrer kanonen med.	WiiSlave	Type: I2C Spændingsniveau: 0-5V Hastighed: 100kbit/sekund Beskrivelse: Denne I2C linje bruges til kommunikation mellem PSoC 2 og Wii-Nunchuck.
		buttonPress	Type: I2C Det fysiske tryk når brugeren trykker på Wii-Nunchuck knapper.
SPI	Denne blok beskriver den ikke-atomiske SPI forbindelse.	MOSI	Type: CMOS Spændingsniveau: 0-5V Hastighed: ?? Beskrivelse: Binært data som sendes fra master til slave.
		MISO	Type: CMOS Spændingsniveau: 0-5V Hastighed: ?? Beskrivelse: Binært data som sendes fra slave til master.
		SCLK	Type: CMOS Spændingsniveau: 0-5V Hastighed: ?? Beskrivelse: Clock signalet fra master til slave, som bruges til at synkronisere den serielle kommunikation.
		SS	Type: CMOS Spændingsniveau: 0-5V Hastighed: ?? Beskrivelse: Slave-Select, som bruges til at vælge slaven der skal modtage og sende data.

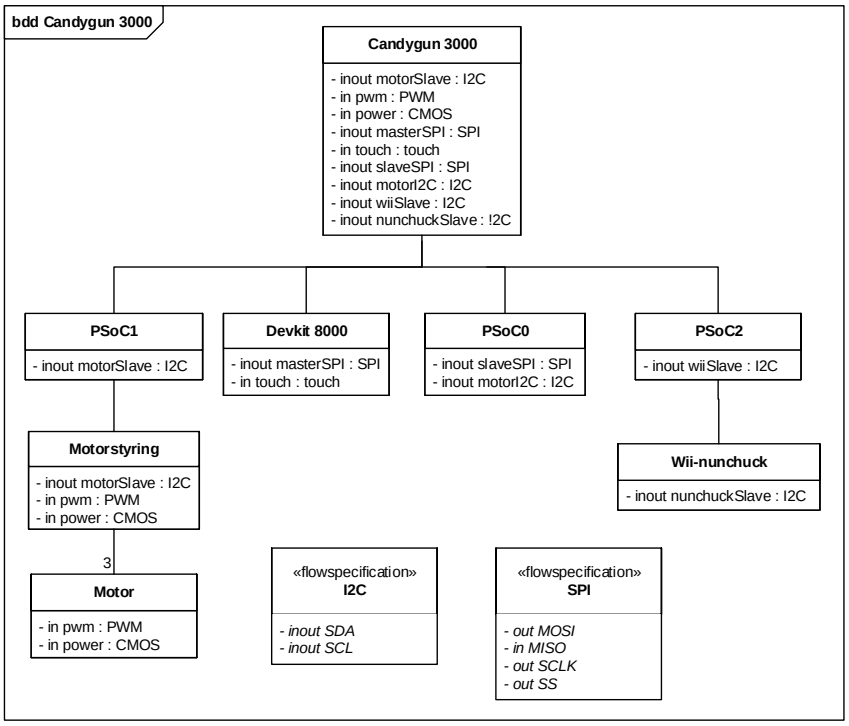
I2C	Denne blok beskriver den ikke-atomiske I2C forbindelse.	SDA	Type: CMOS Spændingsniveau: 0-5V Hastighed: ?? Beskrivelse: Databussen mellem I2C masteren og I2C slaver.
		SCL	Type: CMOS Spændingsniveau: 0-5V Hastighed: ?? Beskrivelse: Clock signalet fra master til lyttende I2C slaver, som bruges til at synkronisere den serielle kommunikation.

3.3 Hardware Arkitektur

I hardwarearkitekturen brydes systemet ned i dele, som senere gør det muligt at uddele arbejdsopgaver, og specificere grænseflader. Hardwarearkitekturen består at BDD og IBD for systemet.

3.3.1 BDD for Candygun 3000

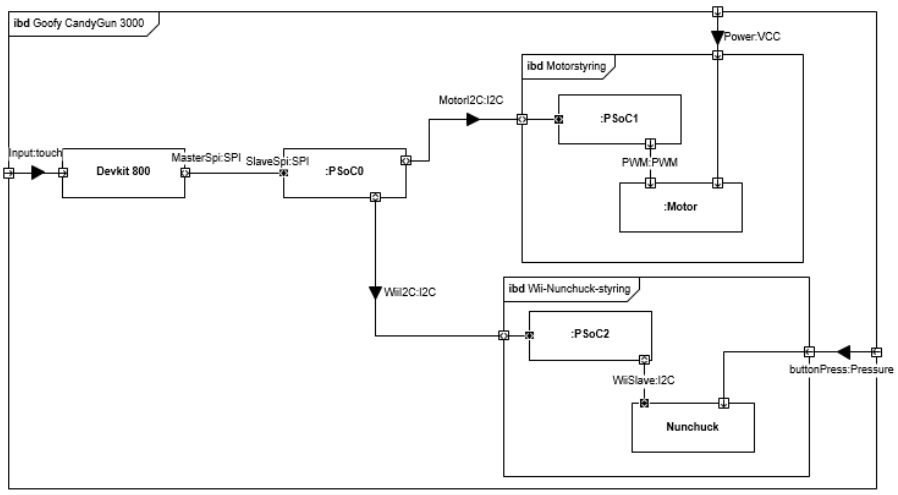
I BDD-diagrammet på figur 4 er Candygun 3000 brudt ned i blokkene PSoC0, PSoC1, PSoC2 og Devkit 8000. Devkit 8000 er brugergrænsefladen, som brugeren kan interagere med via touchskærmen. Den er forbundet via SPI til PSoC0, som er SPI-slave. PSoC0 er desuden også I2C-master. PSoC0 kommunikerer via I2C til PSoC1 og PSoC2. PSoC1 står for motorstyring, som via et PWM-signal styrer de 3 motorer. PSoC2 har til opgave at aflæse brugerinput fra Wii-nunchucken, som også kommunikerer via I2C. På figur 4 ses de forskellige blokke og deres porte. Desuden er der en flowspecification for I2C og SPI, hvor forbindelserne er beskrevet mere detaljeret (set fra master-synspunkt).



Figur 4: Overordnet BDD for Candygun 3000.

3.3.2 IBD for Candygun 3000

I IBD'et på figur 5 er forbindelserne mellem de forskellige blokke overskueliggjort. Det er dermed let at få et overblik over, hvilke grænseflader der skal tages højde for i den videre udvikling.



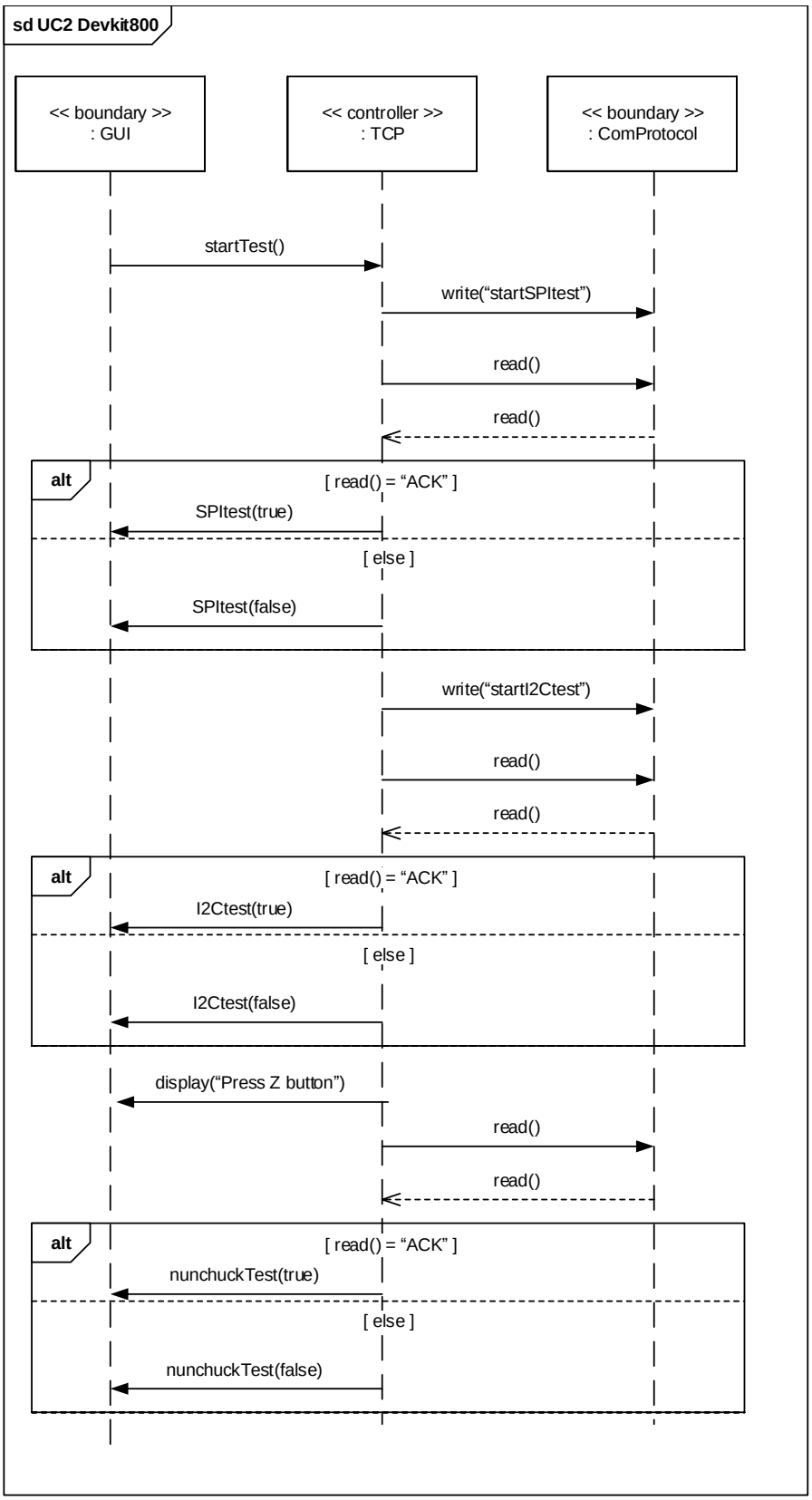
Figur 5: IBD for Candygun 3000.

3.4 Software Arkitektur

I softwarearkitekturen udarbejdes der applikationsmodeller bestående af sekvensdiagrammer og klassediagrammer for hvert delsystem med udgangspunkt i use case 2. Denne arkitektur overskueliggøre kravene til de boundaryklasser, der muliggør kommunikation mellem delsystemerne. Desuden bliver der gennem analyse af use case og sekvensdiagrammer udledt grundlæggende metoder i klasserne.

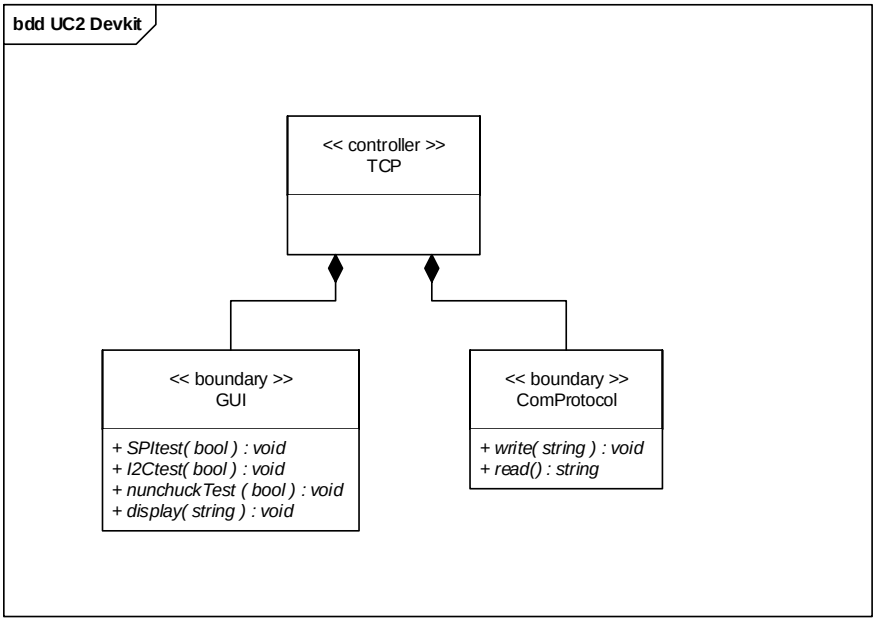
3.4.1 Applikationsmodel for Devkit 8000

Sekvensdiagrammet for Devkit 8000 ses på figur 6. Det tager udgangspunkt i use case 2. Der er to boundaryklasser, da brugergrænsefladen skal kommunikere med brugeren og PSoC0. Ud til brugeren er der en GUI. Boundaryklassen ComProtocol, skal kunne håndtere SPI-kommunikationen til PSoC0. Som det ses af diagrammet initieres testen af brugeren via GUI'en, og derfra er det controlklassen på Devkittet, der sørger for, at de forskellige tests bliver sat i gang, og melder resultatet ud til brugeren via GUI'en.



Figur 6: Sekvensdiagram for Devkit 8000.

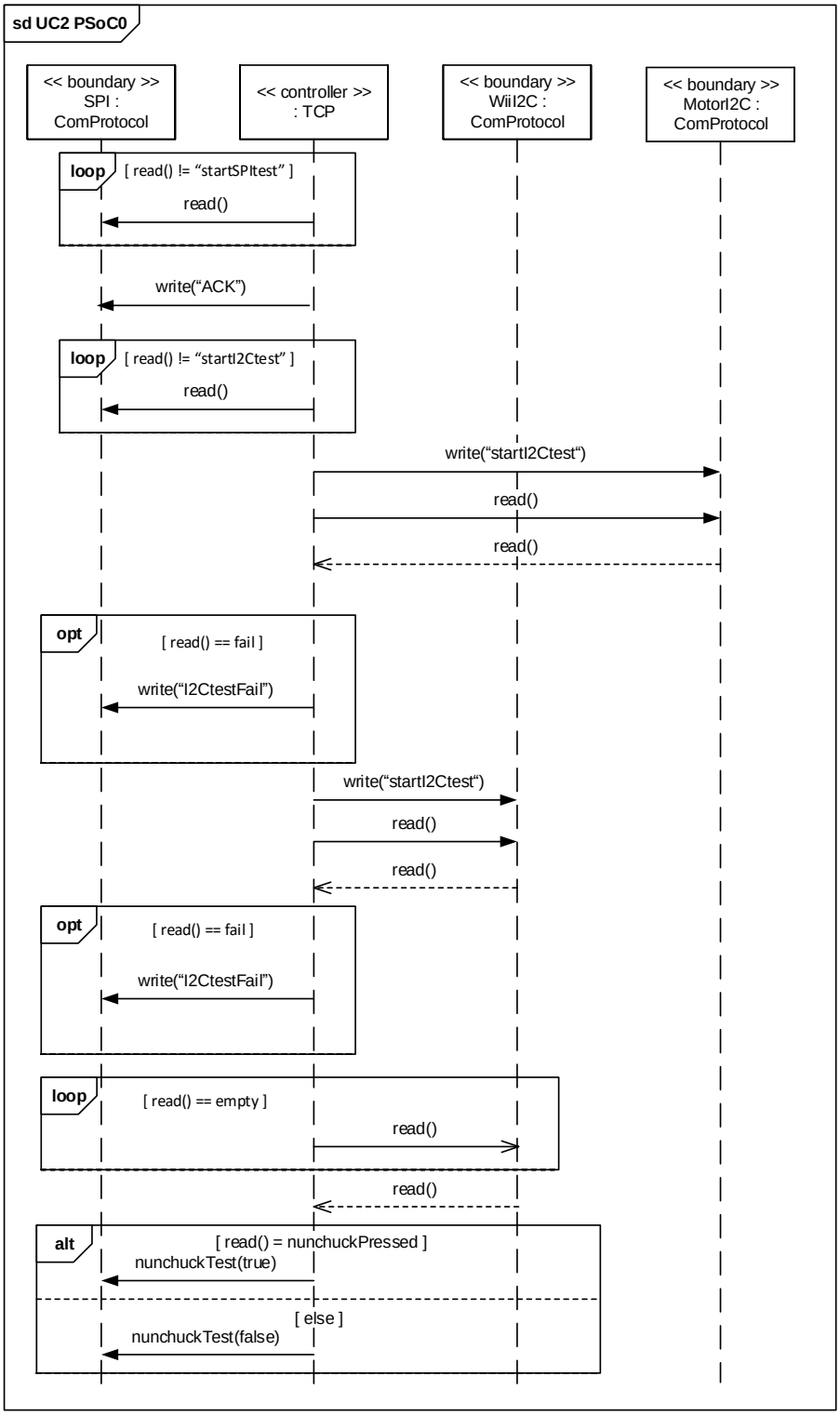
Ud fra sekvensdiagrammet for Devkit 8000 er der udledt foreløbige metoder til klasserne. De ses i klassediagrammet på figur 7.



Figur 7: Klassediagram for Devkit 8000

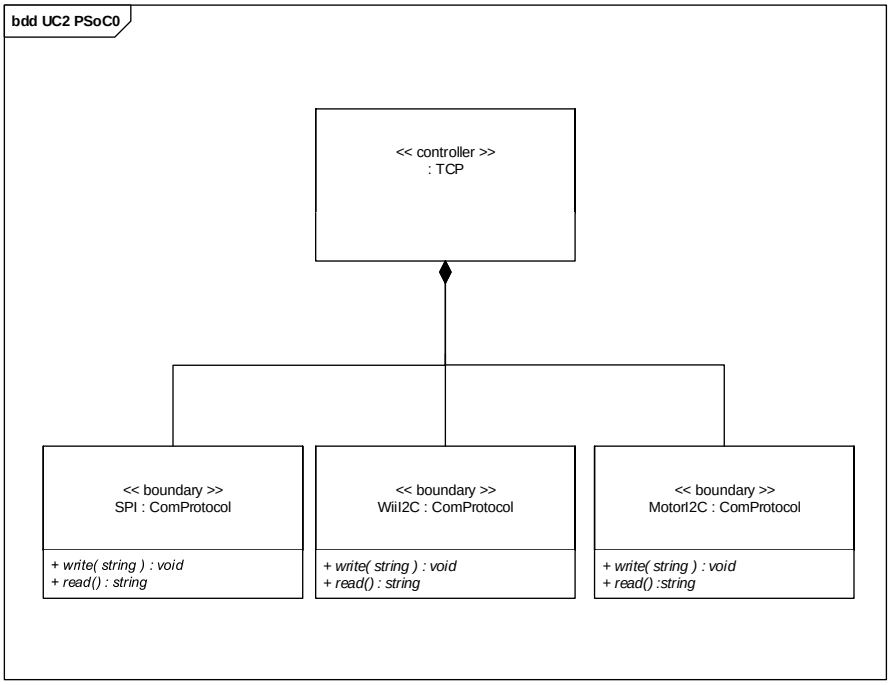
3.4.2 Applikationsmodel for PSoC0

På figur 8 ses et sekvensdiagram for PSoC0 med udgangspunkt i vores test use case - usecase 2. Controlklassen, som er opkaldt efter use case navnet, hedder Test of Protocols. Hvilket på figur 8 er forkortet til TOP. Desuden er der tre boundaryklasser, da PSoC0 skal kommunikere både med Devkit 8000 og de to andre PSoC'er. Som det ses på sekvensdiagrammet, står TOP-klassen og tjekker på boundaryklassen med SPI-forbindelse til Devkit 8000, for at holde øje med om en test bliver startet. Derudover sørger controlklassen for at kommunikere videre ud til de andre PSoC'er, der styrer henholdsvis motorerne og Wii-nunchucken.



Figur 8: Sekvensdiagram for PSoC0.

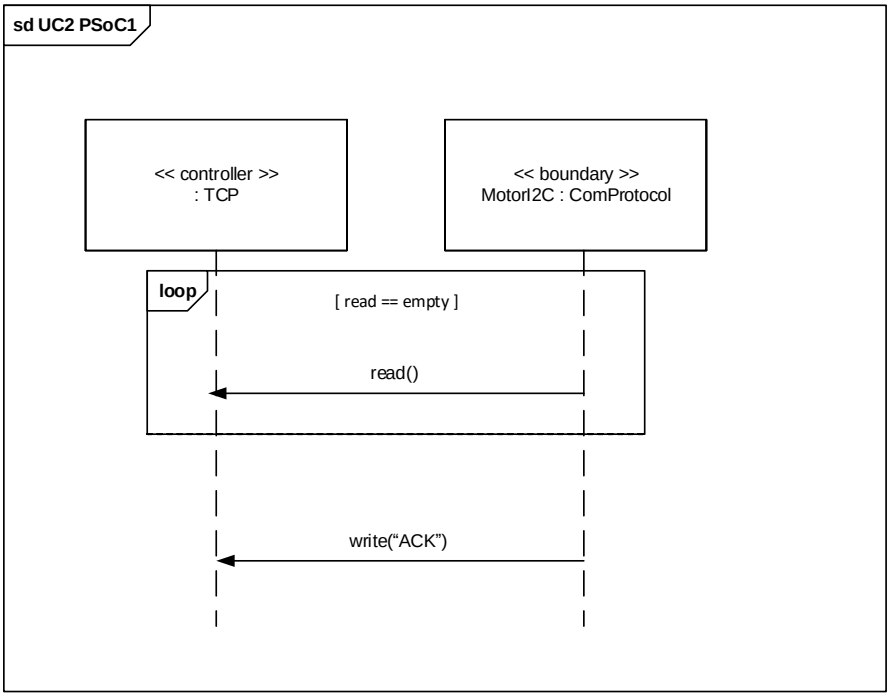
I klassediagrammet på figur 9 ses controlklassen og de tre boundaryklasser, som hører til PSoC0. I klasserne er der tilføjet metoder, som er udledt ud fra sekvensdiagrammet på figur 8. Det giver en god struktur at starte ud fra, når der skal designes og implementes.



Figur 9: Klassediagram for PSoC0.

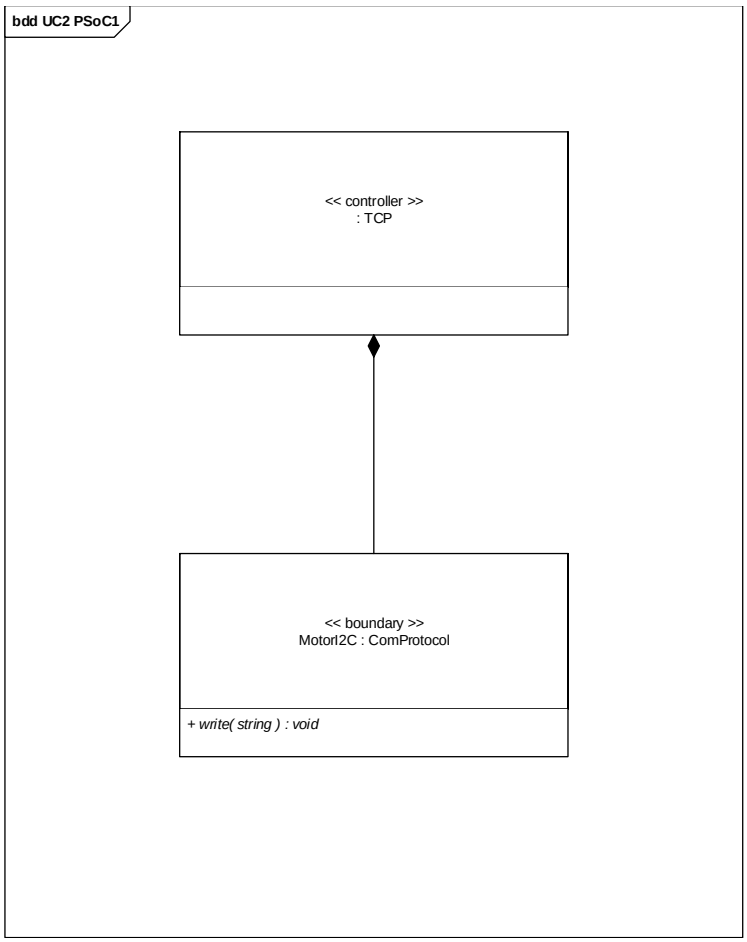
3.4.3 Applikationsmodel for PSoC1

Sekvensdiagrammet for PSoC1 med udgangspunkt i use case 2 ses på figur 10. Også her er controlklassen opkaldt efter use case navnet "Test of Protocols"og i diagrammet forkortet til TOP. Her er der kun én boundaryklasse, da PSoC1, som ellers står for motorstyring, i use case 2, kun anvendes til test af I2C. Dermed skal den kommunikere med PSoC, men har ikke behov for at have andre boundaryklasser. I sekvensdiagrammet ses det, at controlklassen tjekker boundary klassen, for at se om der bliver kommunikeret fra PSoC0.



Figur 10: Sekvensdiagram for PSoC1.

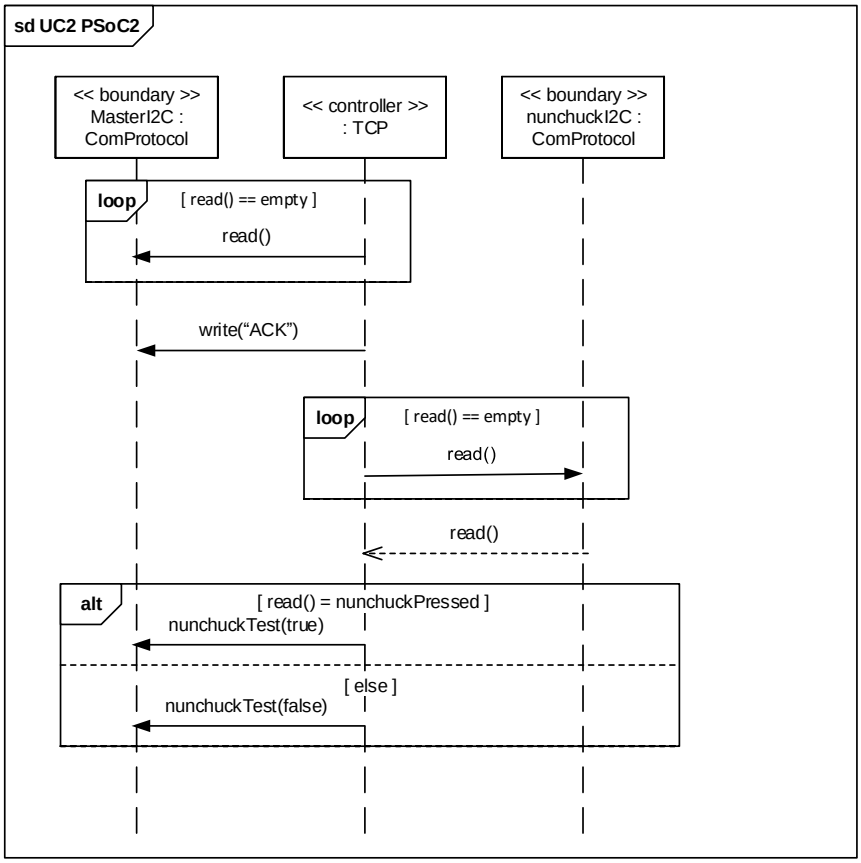
Fra det simple sekvensdiagram på figur 10, er der også udledt et simpelt klassediagram, som kan håndtere de få funktioner, som ses af det tilhørende sekvensdiagram.



Figur 11: Klassediagram for PSoC1.

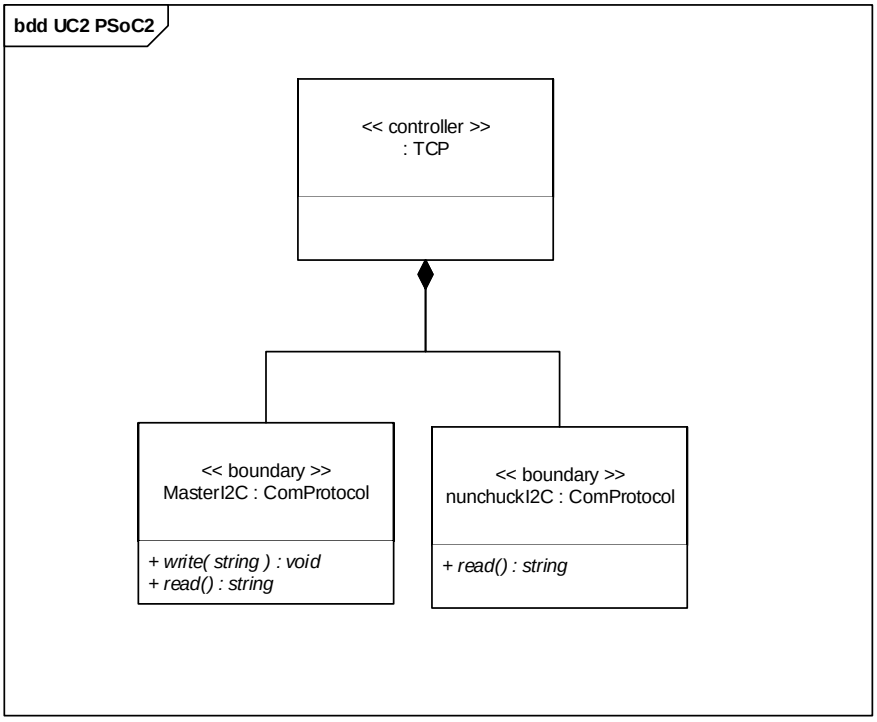
3.4.4 Applikationsmodel for PSoC2

Af sekvensdiagrammet på figur 12 ses kommunikationen mellem klasserne for PSoC2 - også her med udgangspunkt i use case 2. Controlklassen hedder, som i de andre diagrammer TOP, og delsystemet har to boundaryklasser. PSoC2 skal kommunikere med PSoC0, som fortæller, når der skal testes. Den kommunikation foregår gennem "MasterI2C: ComProtocol-klassen, og så skal PSoC2 også aflæse brugerinput fra Wii-nunchucken, hvilket også foregår via I2C, dog gennem den anden boundaryklasse.



Figur 12: Sekvensdiagram for PSoC2.

Ud fra sekvensdiagrammet på figur 12 er der udledt metoder til et klassediagram, som ses på figur 13. Her ses controlklassen og de to boundaryklasser med deres foreløbige metoder, der skal anvendes, som et udgangspunkt til design og implementering af softwaren på PSoC2.

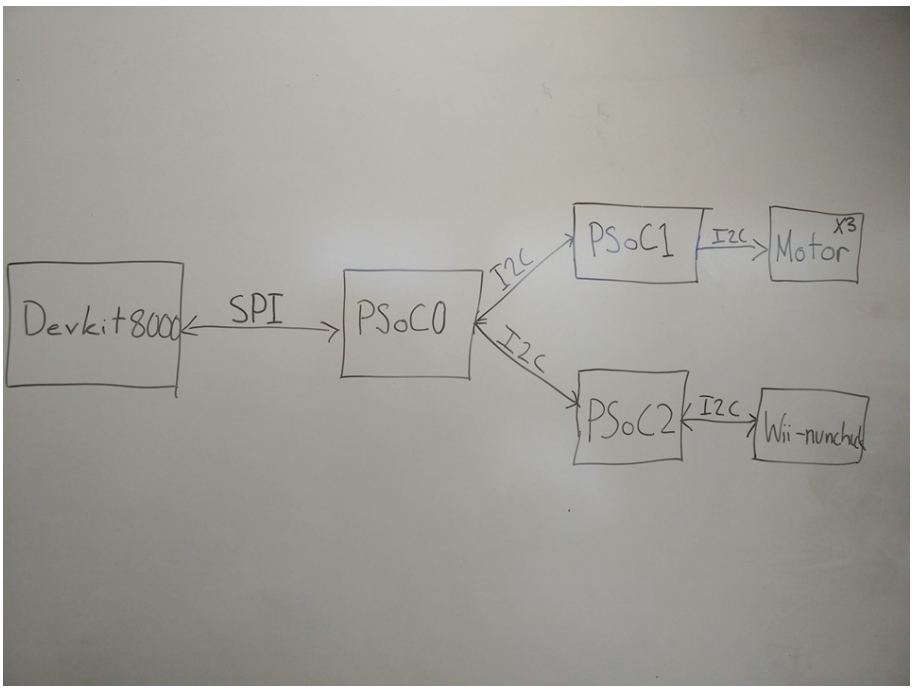


Figur 13: Klassediagram for PSoC2.

3.4.5 Kommunikationsprotokoller

Dette afsnit beskriver de kommunikationsprotokoller som bruges til at sende data mellem systemets komponenter på de brugte bustyper - I2C og SPI.

På figur 14 gives et overblik over systemets forbindelser mellem dets embedded linux platform og microcontrollers. For hver forbindelse ses typen af bus der bruges.



Figur 14: Forbindelser mellem systemets komponenter

SPI Protokol

I2C Protokol

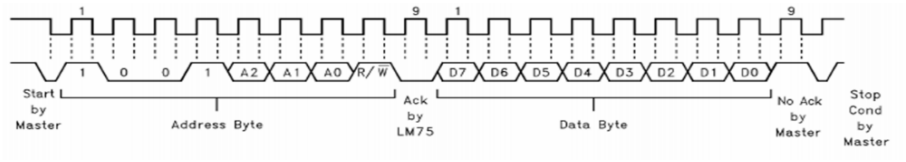
I2C[1] er en bus bestående af to ledninger. Den ene ledning bruges som databus og navngives *Serial Data Line* (SDA). Den anden ledning bruges til clock signalet, for at synkronisere kommunikationen, og navngives *Serial Clock Line* (SCL). Enheder på I2C bussen gør brug af et master-slave forhold til at sende og læse data. En fordel ved I2C bussen er at netværket kan bestå af multiple masters og slaver.

I2C gør brug af en indbygget protokol der anvender adressering af hardwareenheder for at identificere hvilken enhed der kommunikeres med. På tabel 3 ses addresserne tildelt systemets PSoCs.

I2C Adresse bits	7	6	5	4	3	2	1	LSB er read/write indikator
PSoC0	0	0	0	1	0	0	0	0/1
PSoC1	0	0	0	1	0	0	1	0/1
PSoC2	0	0	1	0	0	0	0	0/1

Tabel 3: Adresser brugt på systemets I2C bus

Den indbyggede I2C protokol sender data serielt i pakker af 8-bit (1 byte). På figur 15 ses et timing-diagram for aflæsning af 1 byte. Her ses at processen begynder med en addresse-byte, efterfulgt af en data-byte.



Figur 15: Timing Diagram af 1-byte I2C aflæsning

Da I2C data udveksling sker bytevist, er I2C protokollen opbygget ved at første modtagede byte indikerer typen af kommando, hvorefter N efterfølgende bytes er tilhørende data payload. På denne måde ved modtagere hvordan dataen skal fortolkes.

For at formatere den data der bliver sendt over I2C bussen, bliver der gjort brug af følgende protokol.

For at sende data: Først sendes en kommando type. For denne use case er der kun brug for en enkelt kommando type, nemlig Nunchuck_Data. Derefter afsendes dataen for kommandotypen, som i Nunchuck_Data vil bestå af 3 bytes: xAxis, yAxis og buttonsPressed. Når kommandotypens data er sendt, senderen færdig.

Modtageren skal følge følgende protokol: Først skal den modtagne kommando type aftolkes, og alt efter hvilken type der er modtaget, skal resten af dataen gemmes i bestemte buffere. Når alle dataen er modtaget, skal modtageren ud fra kommando typen bestemme hvilken kommando den skal udføre.

4 Referencer

Litteratur

[1] UM10204, *I2C Bus Specification and user manual*, 4 April 2014