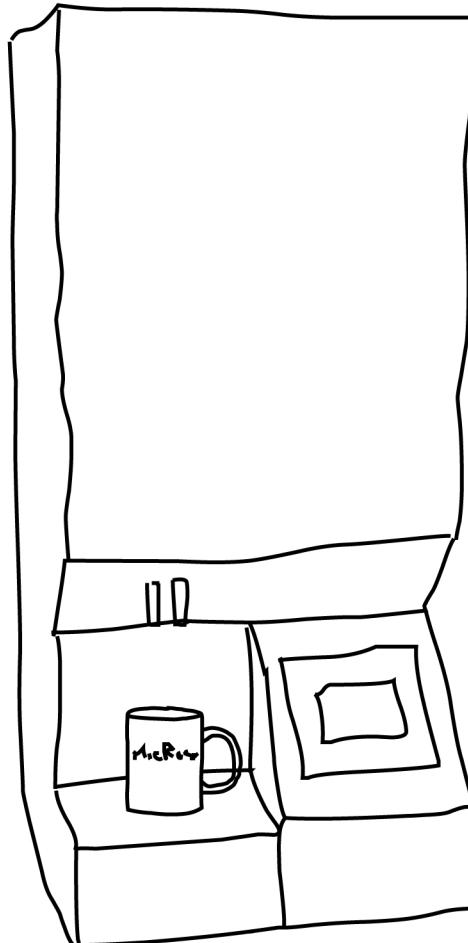


MoccaMonster3000

Dokumentation

3. semesterprojekt

Vejleder: Søren Hansen
31-05-2013



Kenni Reinholt Nielsen

Rasmus Skov Nielsen

Jesper Hede Christensen

Christian Oxholm

Thomas Kjær Christensen

Kristian Møllebjerg Matzen

Martin Munk Hansen

Nikolaj Riishøj Grarup

Andreas Pehn Sloth

Forord

Dokumentationen er udarbejdet for at dokumentere semesterprojektet på 3. semester EIT.

Denne rapport er udarbejdet med interne referencer implementeret som fodnoter. Alle disse referencer, bilag, kode, samt andet væsentlig dokumentation er at finde under mappen bilag på vedlagte CD-ROM.
Denne mappe indeholder følgende:

- Source kode
- PCB
- Test applikationer
- Datablade

Undermapperne i mappen Bilag bærer samme navn som ovenstående punkter.

Indhold

Forord.....	1
Indhold	2
1. Kravspecifikation	5
1.1. Aktører	5
1.2.....	5
1.2.1. Bruger.....	5
1.2.2. AD databasen.....	6
1.2.3. Operatør.....	6
1.2.4. PayPal	6
1.3. Termliste	7
1.3.1. Systemet klar til brug.....	7
1.3.2. Kaffeautomat.....	7
1.3.3. Konteringsdatabasen.....	7
1.3.4. Webportal.....	7
1.3.5. Adgangskort.....	7
1.4. Use Cases.....	8
1.4.1. Use case 1	9
1.4.2. Use case 2	10
1.4.3. Use case 3	11
1.4.4.	12
1.4.5. Use Case 5.....	12
1.5. Ikke funktionelle krav.....	13
1.5.1. Egenkontrol af kaffeautomaten	13
1.5.2. Kabinet.....	13
1.5.3. Brugergrænsefladen.....	14
1.5.4. Webportal.....	15
1.5.5. Yderligere ikke funktionelle krav.....	18
1.5.6. Vedligeholdelsesplan.....	18
1.5.7. Rammer for udvikling.....	18
2. Systemarkitektur	20
2.1. Overordnet arkitektur.....	20
2.1.1. Overordnet allokeringsdiagram for systemet	21
2.2. Arkitektur for Kaffeautomat	22
2.2.1. Blokbeskrivelse for kaffeautomat	22
2.3. Allokering af logisk funktionalitet i kaffeautomaten.....	27
2.3.1. Krav til microcontroller, arm-platform samt IC	28
2.3.2. Signalbeskrivelse.....	29
2.4. Domæneanalyse.....	30
3. Softwarearkitektur	31
3.1. Blok- og aktørbeskrivelse	31
3.2. Design af brugergrænseflade.....	32
3.2.2. Problemidentifikation	35
3.2.3. Klasseidentifikation.....	37
3.2.4. Metodeidentifikation.....	42
3.2.5. Uddybende kontrolklasse beskrivelse	45

3.3. Implementering af Brugergrænseflade	48
3.3.1. Qt	48
3.3.2. Tråde	48
3.3.3. Pakkediagram	49
3.3.4. Klassediagrammer	50
4. Design og implementering af databaser og webfront	60
4.1.1. Konteringsdatabase	60
4.2. Webfront	61
4.3. AD Database	61
4.3.2. Database	63
4.4. Design og implementering af Kommunikation mellem DevKit og PSOC	64
4.4.2. SPI på PSoC	66
4.4.3. DevKit spi driver	67
4.4.4. Pin konfiguration	68
4.5. Booting DevKit	69
4.6. Softwaretest	70
4.6.1. Testbeskrivelser	71
5. Hardware dokumentation	78
5.1. PSU	79
5.2. Pulverdosering	82
5.2.1. Pulverniveausensor	82
5.2.2. Sensor	82
5.2.3. Forstærkerkredsløb	83
5.3. Doseringsmekanisme	84
5.3.1. Motordriver og stepper-motor	84
5.4. Vandstyring	88
5.4.1. Vanddosering	89
5.4.2. Temperatursensor	90
5.4.3. Varmelegeme	92
5.4.4. Vandniveausensor	93
6. Spildbakke	94
6.1. Kopsensor	95
6.1.1. Schmitt-trigger	95
6.2. Kortlæser	96
6.3. Enhedstest	98
6.3.1. Temperatursensor	98
6.3.2. Pulverniveausensor	100
6.3.3. Kopsensor	104
6.3.4. Vanddosering	107
6.3.5. PSU	108
6.3.6. Motordriver	110
6.3.7. Vandniveausensor	111
6.3.8. Varmelegeme	113
6.3.9. Kortlæser	114
6.4. HW integrationstest	115
6.4.1. Test 1	115
6.4.2. Test case 2	115

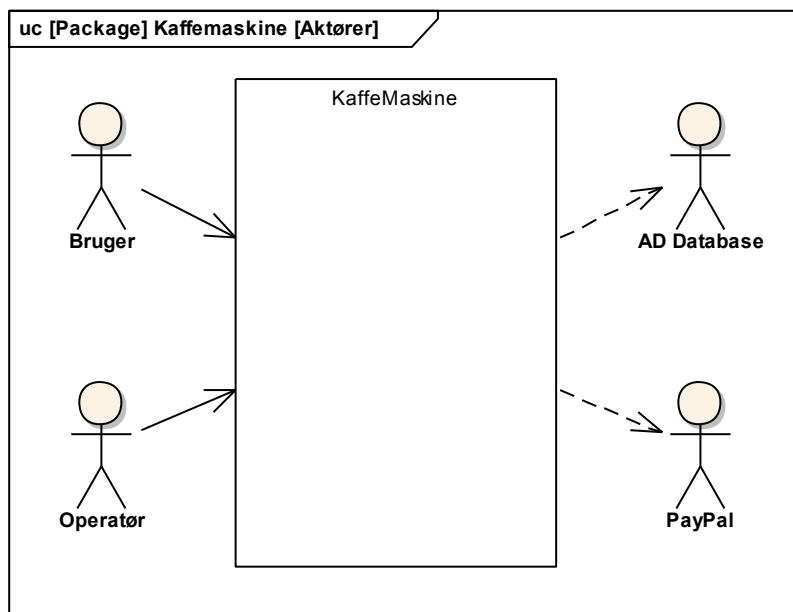
6.4.3. Test case 3	116
6.4.4. Test case 4	116
7. Pulverdispenser	117
8. Bilagsliste	121

1. Kravspecifikation

1.1. Aktører

I dette afsnit beskrives aktører og deres rolle i systemet.

Til højre ses et aktordiagram, som beskriver alle aktører og res forhold til systemet



Figur 1 - Aktordiagram

1.2.1. Bruger

Aktørnavn	Bruger
Type	Primær
Beskrivelse	Brugeren er den aktør der ønsker at benytte systemet. Hver bruger har tilknyttet en personlig konto, med en saldo som brugeren kan købe kaffe for. Brugeren kan ved hjælp af en webportal administrere sin konto.

1.2.2. AD databasen

Aktørnavn	Active Directory databasen
Type	Off-Stage
Beskrivelse	<p>Active Directory (AD) databasen indeholder oplysninger om en institutions brugere, og benyttes til at validere brugeren i forbindelse med oprettelse af konti, køb af kaffe, og log-in på webportalen. De oplysninger systemet skal kunne tilgå er blandt andet de brugernavne og adgangskoder, brugerne har til institutionen.</p>

1.2.3. Operatør

Aktørnavn	Operator
Type	Primær
Beskrivelse	<p>Operatørens opgave er at vedligeholde systemet. Operatøren modtager en e-mail fra systemet (1.5.1), når der er brug for vedligeholdelse. Operatøren har ansvar for at udbedre fejl, opfylde forbrugsstoffer, tømme spildbakke, rengøre kaffeautomaten og lignende. Operatørens opgaver beskrives i afsnittet om vedligeholdelse af kaffeautomaten (1.5.6)</p>

1.2.4. PayPal

Aktørnavn	PayPal
Type	Off-stage
Beskrivelse	<p>PayPal er en off-stage aktør der varetager pengetransaktioner mellem brugeren og systemet.</p>

1.3. Termliste

1.3.1. Systemet klar til brug

I flere af de følgende Use Cases er det en forudsætning at systemet er klar til brug. Dette indebærer at følgende er gældende

- Systemet har vand til rådighed
- Spildbakken er ikke fuld
- Der er kaffe i minimum én af kaffebeholderene.
- Systemet er rengjort inden for den sidste måned
- Systemet er tilsluttet internettet

1.3.2. Kaffeautomat

Det er i kaffeautomaterne at brugerne kan købe kaffe. Et opsat system kan have flere kaffeautomater, placeret rundt på den pågældende institution.

1.3.3. Konteringsdatabasen

Konteringsdatabasen er en database tilknyttet systemet. Det er i denne database brugerens kontopløsninger gemmes. Ved oprettelse af konti, importeres persondata fra AD databasen og sammenfattes med kortnummeret på brugerens adgangskort. Konteringsdatabasen indeholder oplysninger om brugerens saldo.

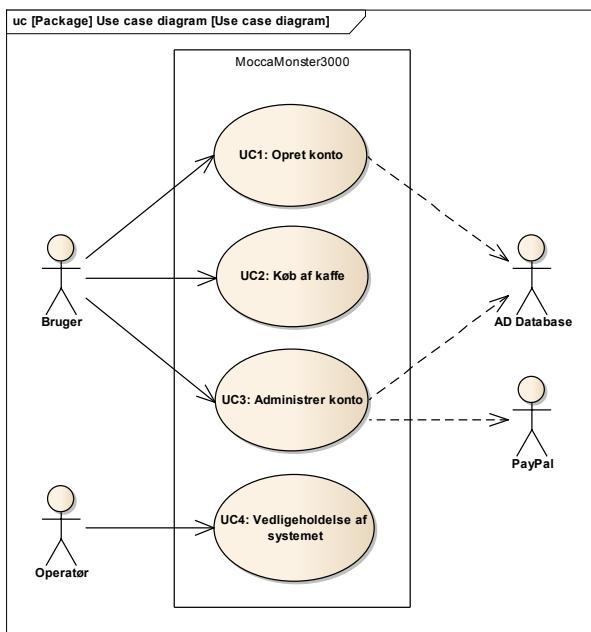
1.3.4. Webportal

Webportal er en hjemmeside tilknyttet systemet. Det er her at brugeren kan administrere sin konto. Webportalen er yderligere beskrevet i ikke funktionelle krav (1.5.4)

1.3.5. Adgangskort

Adgangskortet kan for eksempel være et studiekort på en uddannelsesinstitution. Det eneste krav til dette kort er at det skal indeholde et unikt ID som for systemet er læseligt. For eksempel med RFID eller magnetstribes.

1.4. Use Cases



Figur 2 - Use case diagram

1.4.1. Use case 1

Navn	Opret konto
ID	1
Samtidige forekomster	1
Primær aktør	Brugeren – Ønsker at oprette sig i systemet
Sekundær aktør	Konteringsdatabase
Off-stage aktør	Active Directory database
Initiator	Brugeren
Forudsætninger	Systemet er tændt og er klar til brug (1.3.1) Brugeren har et gyldigt adgangskort (1.3.5) og er ikke oprettet i systemet
Resultat	Brugeren er blevet oprettet i systemet

1.4.1.1. Hovedscenarie

Brugeren kører sit adgangskort forbi en af kaffeautomaternes kortlæser
 Systemet giver meddelelse om at kortnummeret ikke er kendt, og beder brugeren taste sit brugernavn.
 Brugeren indtaster brugernavn på kaffeautomatens brugergrænseflade
 Systemet henter oplysninger fra AD Databasen
 Systemet opretter brugeren i konteringsdatabasen
 Systemet bekræfter over for brugeren, at han er oprettet

1.4.1.2. Undtagelser

4. [Systemet henter oplysninger fra AD Databasen]
 - 4.1. Systemet kan ikke få forbindelse til AD databasen
 - 4.1.1. En fejlmeddeelse vises
 - 4.1.2. Use casen afbrydes
 - 4.2. Brugernavnet findes ikke i AD Databasen
 - 4.2.1. En fejlmeddeelse vises
 - 4.2.2. Punkt 3 gentages
 -
5. [Systemet opretter brugeren i konteringsdatabase]
 - 5.1. Systemet kan ikke få forbindelse til konteringsdatabasen
 - 5.1.1. En fejlmeddeelse vises
 - 5.1.2. Use casen afbrydes

*Brugeren kan til hver en tid annullere use casen

1.4.2. Use case 2

Navn	Køb af kaffe
ID	2
Samtidige forekomster	1
Primær aktør	Bruger – Ønsker at købe kaffe
Initiator	Bruger
Forudsætninger	Systemet er tændt og er klar til brug (1.3.1) Bruger har et gyldigt adgangskort (1.3.5) og er oprettet i systemet (1.4.1) Brugerens konto er ikke spærret Bruger har en beholder til kaffe
Resultat	Bruger har fået kaffe og saldoen er opdateret

1.4.2.1. Hovedscenarie

1. Bruger kører sit adgangskort forbi en af kaffeautomaternes kortlæser
2. Brugerens kontooplysninger vises på kaffeautomatens brugergrænseflade, samt valgmuligheder for type, styrke og mængde af kaffe
3. Bruger vælger ønsket type, styrke og mængde.
4. Systemet oplyser pris for ønsket køb
5. Bruger accepterer købet
6. Systemet validerer at bruger har placeret sin beholder under dysen
7. Systemet brygger den ønskede kaffe
8. Bruger tager sin beholder

1.4.2.2. Undtagelser

5. [Bruger accepterer købet]
 - 5.1. Saldoen på brugerens konto er lavere end prisen på købet
 - 5.1.1. Der vises en fejlmeddeelse
 - 5.1.2. Der springes til punkt 3 i hovedscenariet
6. [Systemet validerer at bruger har placeret sin beholder under dysen]
 - 6.1. Der er ikke placeret en beholder under dysen
 - 6.1.1. Bruger bedes indsætte en beholder
 - 6.1.1.1. Bruger indsætter en beholder
 - 6.1.1.2. Use case fortættes efter brugerens accept

*Bruger skal fra punkt 2 til og med punkt 4 have mulighed for at annullere købet.

1.4.3. Use case 3

Navn	Administrer konto
ID	3
Samtidige forekomster	1..*
Primær aktør	Brugeren – Ønsker at administrere sin konto
Initiator	Brugeren
Forudsætninger	Brugeren er oprettet i systemet (1.4.1)
Resultat	Brugeren har administreret sin konto

1.4.3.1. Hovedscenarie

1. Bruger logger ind på webportalen med sit brugernavn og adgangskode
2. Der vises kontooplysninger samt mulighed for at optanke, spærre, genåbne eller lukke sin konto
 - 2.1. Brugeren vælger at optanke konto
 - 2.1.1. Bruger vælger beløb
 - 2.1.2. Bruger betaler via PayPal
 - 2.1.3. Systemet informerer at konto er optanket
 - 2.1.4. Der springes tilbage til Punkt 2
 - 2.2. Brugeren vælger at spærre sin konto
 - 2.2.1. Brugeren bedes bekräfte at kontoen skal spærres
 - 2.2.2. Systemet informerer at kontoen er spærret
 - 2.2.3. Der springes til Punkt 2
 - 2.3. Brugeren vælger at genåbne sin konto
 - 2.3.1. Brugeren bedes bekräfte at kontoen skal genåbnes
 - 2.3.2. Systemet informerer at kontoen er genåbnet
 - 2.3.3. Der springes til Punkt 2
 - 2.4. Brugeren vælger at lukke sin konto
 - 2.4.1. Brugeren bedes bekräfte at kontoen skal lukkes
 - 2.4.2. Systemet informerer at kontoen er lukket
3. Bruger logger ud

1.4.3.2. Undtagelser

- 2.2. [Brugeren vælger at spærre sin konto]
- 2.2.1. Brugerens konto er allerede spærret
- 2.2.1.1. Der vises en fejmeddelelsen
- 2.2.1.2. Der springes til step 2.

2.3. [Brugerens vælger at genåbne sin konto]

2.3.1. Brugerens konto er allerede åben

2.3.1.1. Der vises en fejlmeddelelsen

2.3.1.2. Der springes til step 2.

1.4.4.

1.4.5. Use Case 5

Navn	Vedligeholdelse af systemet
ID	5
Samtidige forekomster	1
Primær aktør	Operatøren – Ønsker at vedligeholde systemet
Initiator	Operatøren
Forudsætninger	Operatøren har modtaget en mail om at en af kaffeautomaterne har brug for vedligeholdelse (1.5.1)
Resultat	Kaffeautomaten er blevet vedligeholdt

1.4.5.1. Hovedscenarie

1. Operatøren læser mailen om at en af kaffeautomaterne skal vedligeholdes
2. Operatøren åbner automaten og tømmer spilbakken, genopfylder kaffebeholderen og tjekker vandforsyningen
3. Kaffeautomaten tjekker at spilbakken er tom, at der er kaffe i kaffebeholderene samt at der er vand til rådighed
4. Kaffeautomaten låser op og er klar til brug

1.5. Ikke funktionelle krav

I dette afsnit beskrives de ikke funktionelle krav til systemet

1.5.1. Egenkontrol af kaffeautomaten

Efter hver gennemgang af use case 1 udføres en egenkontrol af kaffeautomaterne, hvor følgende kontrolleres

- At automaten har tilgang til vand
- At automaten har kaffe i kaffebeholderne
- At spildbakken ikke er fuld

Hvis en af disse punkter ikke er gældende, skal automaten gå i en låst tilstand og der skal vises en meddelelse på brugergrænsefladen, hvor det oplyses at automaten skal vedligeholdes. I den låste tilstand skal kaffeautomaten ikke kunne bruges. Når automaten detekterer at alle tre punkter igen er opfyldt, skal den forlade den låste tilstand og igen være klar til brug.

Hvis en eller flere af de angivne ikke er gældende skal der sendes en mail til operatøren, hvor der står hvilken kaffeautomat, der skal vedligeholdes.

Når denne mail er sendt startes use case 4.

1.5.2. Kabinet

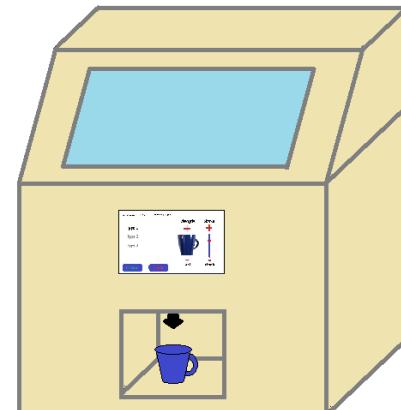
Kabinetet til kaffeautomaten kan som udgangspunkt ligne denne skitse.

Kabinetet laves af MDF-plader. Foran sættes et "vindue" i plexiglas, så det er muligt for brugeren at se hvad der sker inde i automaten. Brugergrænsefladen er monteret på forsiden af kabinetet.

Nederst i kabinetet er der en uddybning hvori kaffebeholderen kan placeres. I bunden af denne uddybning placeres spildbunken.

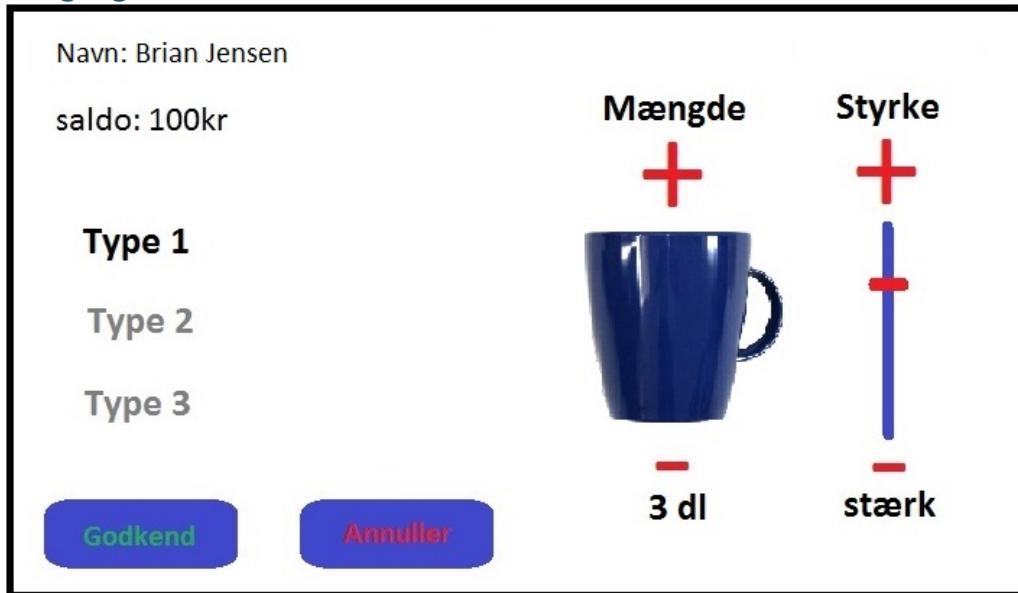
Kabinetet må max måle, HxBxD 85x40x50 cm +/- 5 cm.

Maskinen må max veje 45 kg



Figur 3 - Skitse af kabinet

1.5.3. Brugergrænsefladen



Figur 4 - Skitse for brugergrænseflade på maskinen

Brugergrænsefladen, som befinner sig på kaffeautomaten kan som udgangspunkt ligne denne skitse. For at give brugeren mest muligt overblik, skal alle valg foretages på samme skærbillede. Brugergrænsefladen skal befinde sig på en touchskærm.

På skærmen kan brugeren se sine kontooplysninger i form af navn og tilhørende saldo. Valg af type, mængde og styrke af kaffe vælges på brugergrænsefladen. Købet godkendes ved at trykke på knappen godkend og brugeren kan til enhver tid afbryde interaktionen ved et tryk på knappen annuler.

En type kaffe vælges ved at trykke på dennes ikon/tekst på touchskærmen. De kaffetyper, der ikke er valgt fades herefter til gråt. Mængden af kaffe vælges ved hjælp af plus- og minusknapper. Mængden vises sammen med et eksempel på en tilsvarende beholder.

Styrken på kaffen vælges ligeledes ved hjælp af plus- og minusknapper. Styrken vises på en akse sammen med den valgte styrke.

1.5.4. Webportal

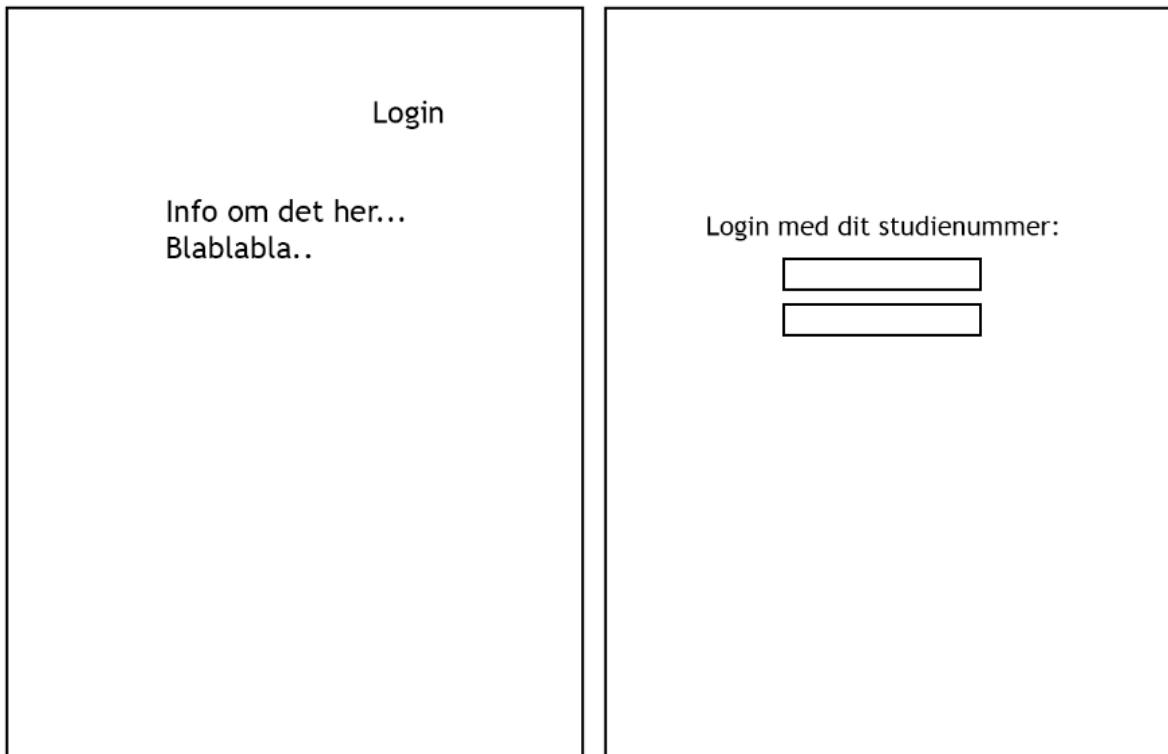
Til systemet skal der bruges en webportal hvor brugeren kan administrere sin konto.

Ud over den funktionalitet der er beskrevet i de funktionelle krav kan der fx implementeres

- Graf for kaffeforbrug
- Graf for kontobevægelse
- Log over tidligere køb

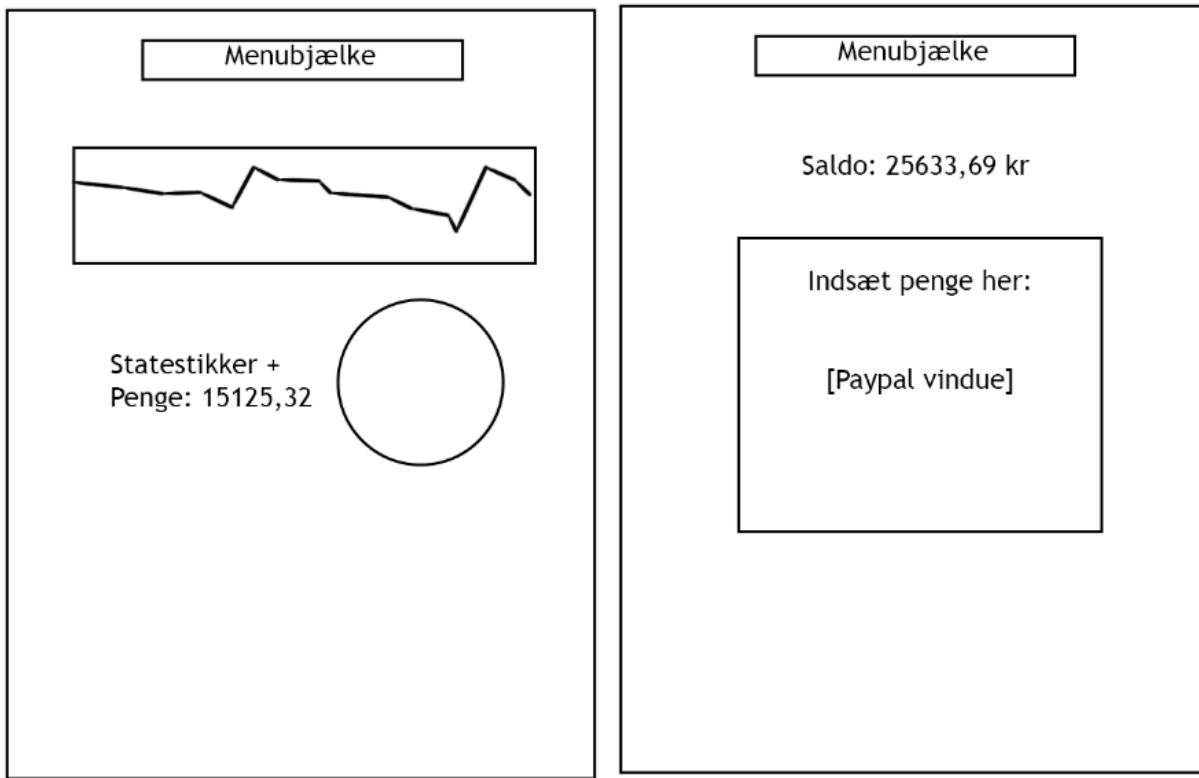
Et eksempel på udseendet på webportalen kan være som følgende

Forside

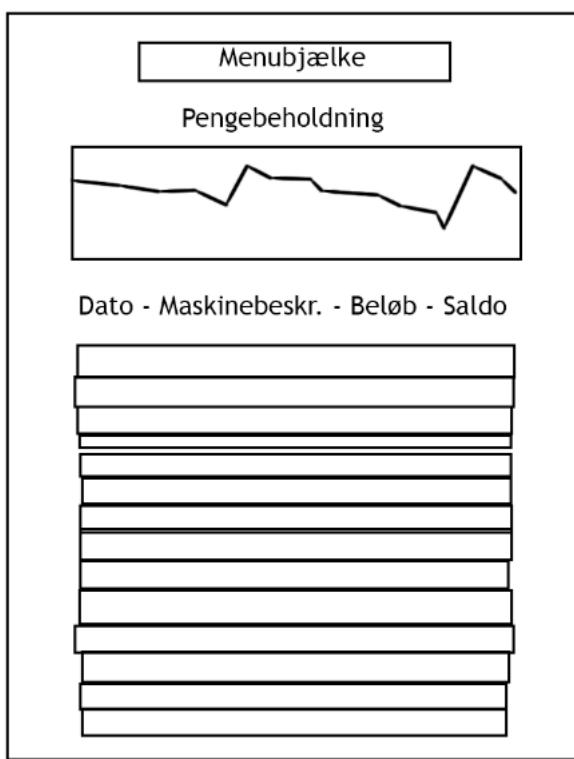


The wireframe illustrates a two-page web portal interface. The left page, labeled 'Forside' (Homepage), contains a 'Login' button at the top and the text 'Info om det her... Blablabla...' below it. The right page, labeled 'Login', contains the text 'Login med dit studienummer:' above two empty input fields.

Velkomstside – Tank konto:



Kontobevægelser:



1.5.5. Yderligere ikke funktionelle krav

Yderligere ikke funktionelle krav systemet skal overholde er beskrevet nedenfor

- Kaffeautomaterne skal ved opstart udføre en software baseret hardware test.
- Automaterne skal levere en drikkeklar kop kaffe på 92°C +/- 2°C
- Automaterne skal levere en drikkeklar kaffe på maks. 5 minutter.
- Automaternes styrekredsløb skal være isoleret fra mekaniske og våddele i en maskine kasse med kapslingsklasse på IP4x
- Firmwareen på automaterne skal kunne fjernopdateres efter installation
- Automaterne skal have en nødstopknap, så brugeren kan afbryde kaffebrygningen

1.5.6. Vedligeholdelsesplan

Da fødevarestyrelsen ikke har fastlagt specifikke krav til vedligeholdelsesplaner for kaffeautomater af denne type, er det valgt at operatøren mindst én gang hver 30. dag skal vedligeholde automaterne. Følgende udføres af operatøren under vedligeholdelse

- Automaten inspiceres visuelt
- Alle indvendige flader aftørres og desinficeres
- Vandreservoir tømmes og desinficeres
- Temperaturføler aftørres og desinficeres
- Alle niveauer på forbrugsstoffer tjekkes – er der nogen reserver der er under 50% af fuld kapacitet, fyldes disse

1.5.7. Rammer for udvikling

Mellem brugeren og den kaffebryggende del af produktet er DevKit8000 og PSoC3 kit.

DevKittet skal fungere som led mellem bruger, bruger databaser og resten af systemet. Som grænseflade til brugeren har DevKittet en touchskærm, hvor brugeren kan vælge sin egen variation af kaffe. Derudover har det forbindelse til internettet, hvor den kan se informationer om brugeren, f.eks. om kortet er i brug og om der er penge på. DevKittets forbindelse til systemet sker gennem PSoCen. Softwaren skrives i C og C++, og er krydskompileret til DevKittets processer.

PSoCen er ledet mellem DevKittet og den kaffebryggende del af systemet. PSoCen modtagerinformationerne om type, mængde og styrke af kaffe fra DevKittet. De informationer skal PSoCen bruge til at be-

regne hvor længe vandet skal koge, samt hvor længe de forskellige slanger der tilfører bønner og vand skal være åbne. Til at hjælpe med dette er PSoCen forbundet til et antal sensorer.

Vandet varmes op imens doseringen af kaffe udmåles. Temperaturen på vandet måles med en temperaturmåler, og PSoCen ved derfor løbende hvor varmt vandet er. Mængden af kaffe afmåles med en vægt, hvor PSoCen lukker for tilførslen når den korrekte mængde er nået. Når opvarmning og mængde har nået de rette værdier tilføres vandet til kaffen som løber igennem til en kop. Softwaren på PSoCen skrives i programmet PSoC creater, i sproget C.

2. Systemarkitektur

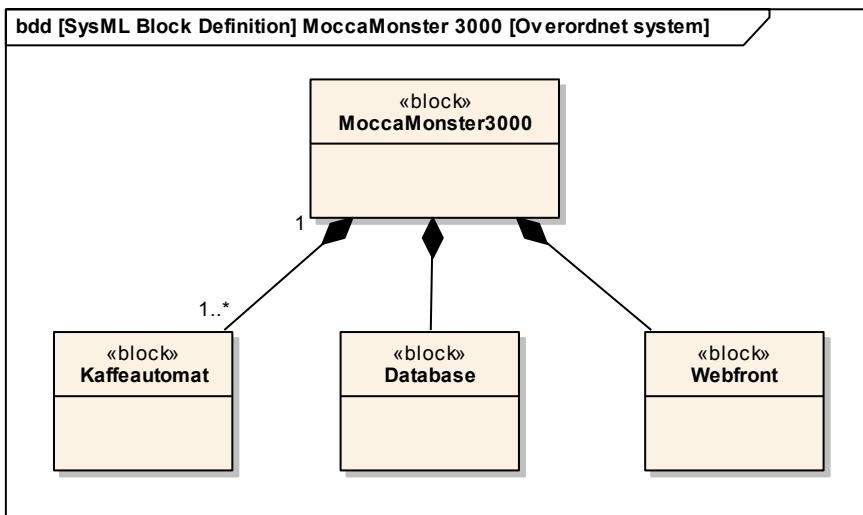
Dette afsnit beskriver systemarkitekturen for systemet "MoccaMonster3000", som formuleret i projektformuleringen og specificeret i kravspecifikationen. For en beskrivelse af det overordnede system henvises til systembeskrivelsen¹

Formålet er:

- At identificere overordnede komponenter samt at fastlægge deres grænseflader
- At identificere og beskrive de eksterne komponenter, der skal anvendes i projektet
- At identificere arbejdsopgaver for projektets design- og implementeringsfase

2.1. Overordnet arkitektur

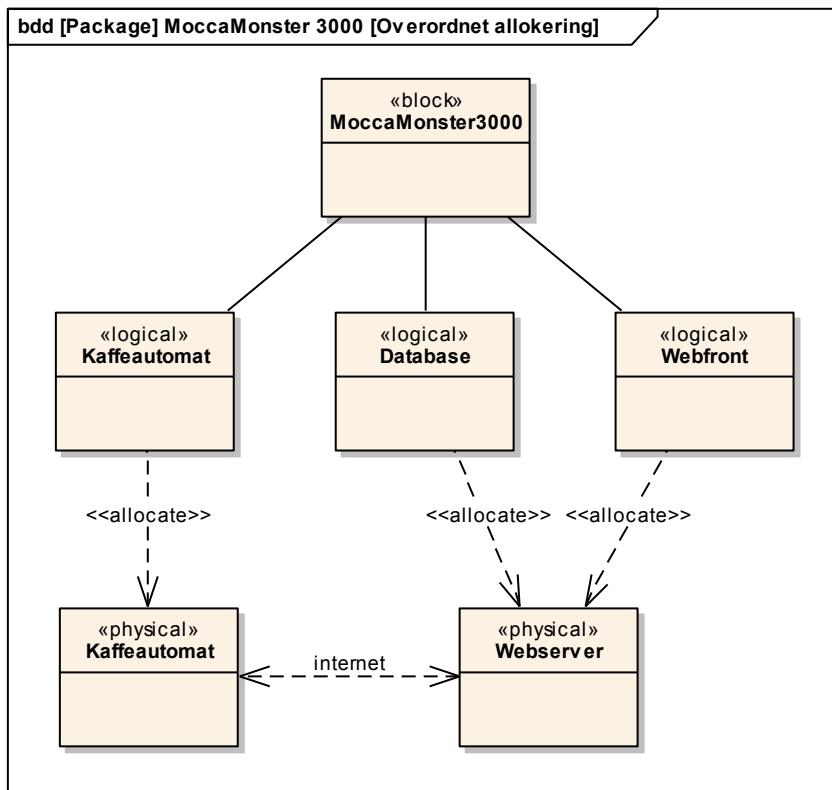
Det overordnede system er beskrevet i det følgende diagram.



¹ Se Rapport, afsnit 7.

2.1.1. Overordnet allokeringsdiagram for systemet

Nedenfor ses et overordnet allokeringsdiagram for systemet. Det ses at den logiske funktionelitet for databasen og webfronten er allokeret på samme webserver. Kaffeautomatens logiske funktionalitet er allokeret på en række interne komponenter i automaten. Denne interne allokering er beskrevet senere i systemarkitekturen. Der er en kommunikationen mellem kaffeautomaten og databasen, der er allokeret på en internetforbindelse. Da webfronten og databasen er allokeret på samme server, kommunikerer de internt.

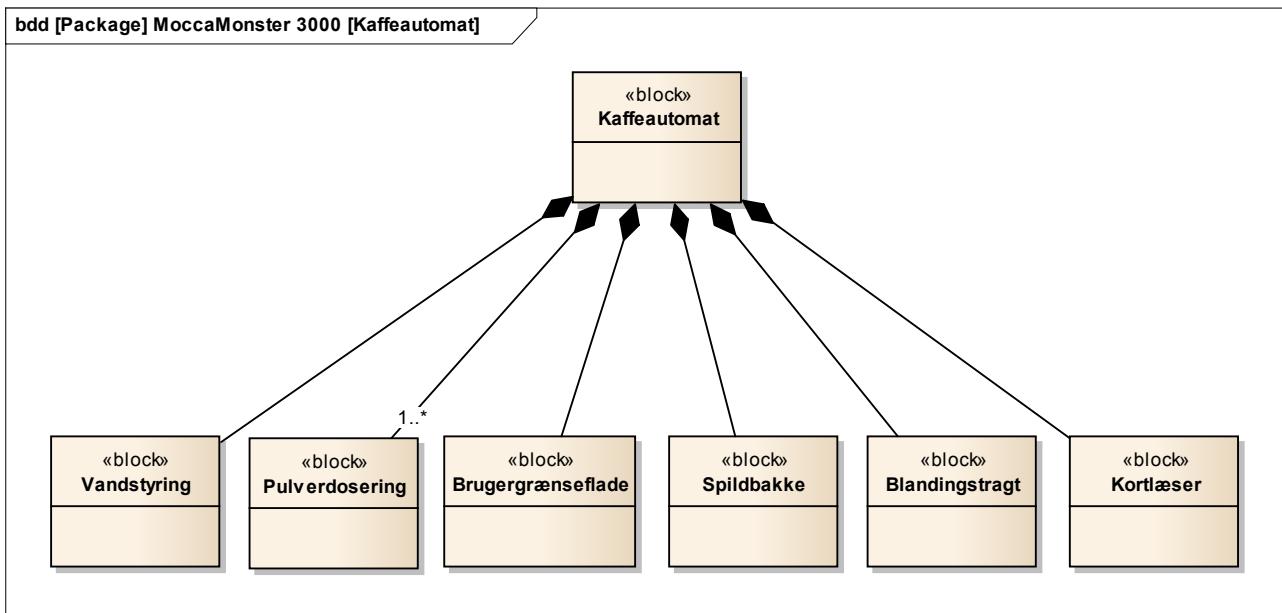


For yderligere beskrivelse af det overordnede system henvises til systembeskrivelsen².

² Se Rapport, afsnit 7.

2.2. Arkitektur for Kaffeautomat

På baggrund af en analyse af det overordnede system, er det besluttet at en kaffeautomat skal bestå af en **vandstyring**, en **pulverdosering**, en **spildbakke**, en **blandingstragt**, en **kortlæser** og en **brugergrænseflade**. Alle disse overordnede komponenter skal alle bygges ind i et kabinet, der overholder kravene specificeret i de ikke funktionelle krav³. Disse delsystemer ses beskrevet i de følgende afsnit.



2.2.1. Blokbeskrivelse for kaffeautomat

De definerede blokke i en kaffeautomat beskrives i det følgende:

Kortlæseren skal kunne aflæse brugerens adgangskort og sende kortnummeret til brugergrænsefladen.

Brugeren skal kunne interagere med systemet gennem **brugergrænsefladen**. Det er her at brugeren kan bestille kaffe. Derfor skal brugergrænsefladen kunne kommunikere med de andre delsystemer i kaffeautomaten, for at sende bestillingen videre til de blokke der skal brygge kaffen. Brugergrænsefladen skal kunne modtage et kortnummer fra kortlæseren, samt sammenholde og sammenkæde dette med data fra konteringsdatabasen. Dette medfører at brugergrænsefladen skal kunne kommunikere med konteringsdatabasen over en internetforbindelse.

Vandstyringen har ansvaret for at opvarme og dosere den korrekte mængde vand til den kaffe brugeren har bestilt og føre det ned i blandningstragten.

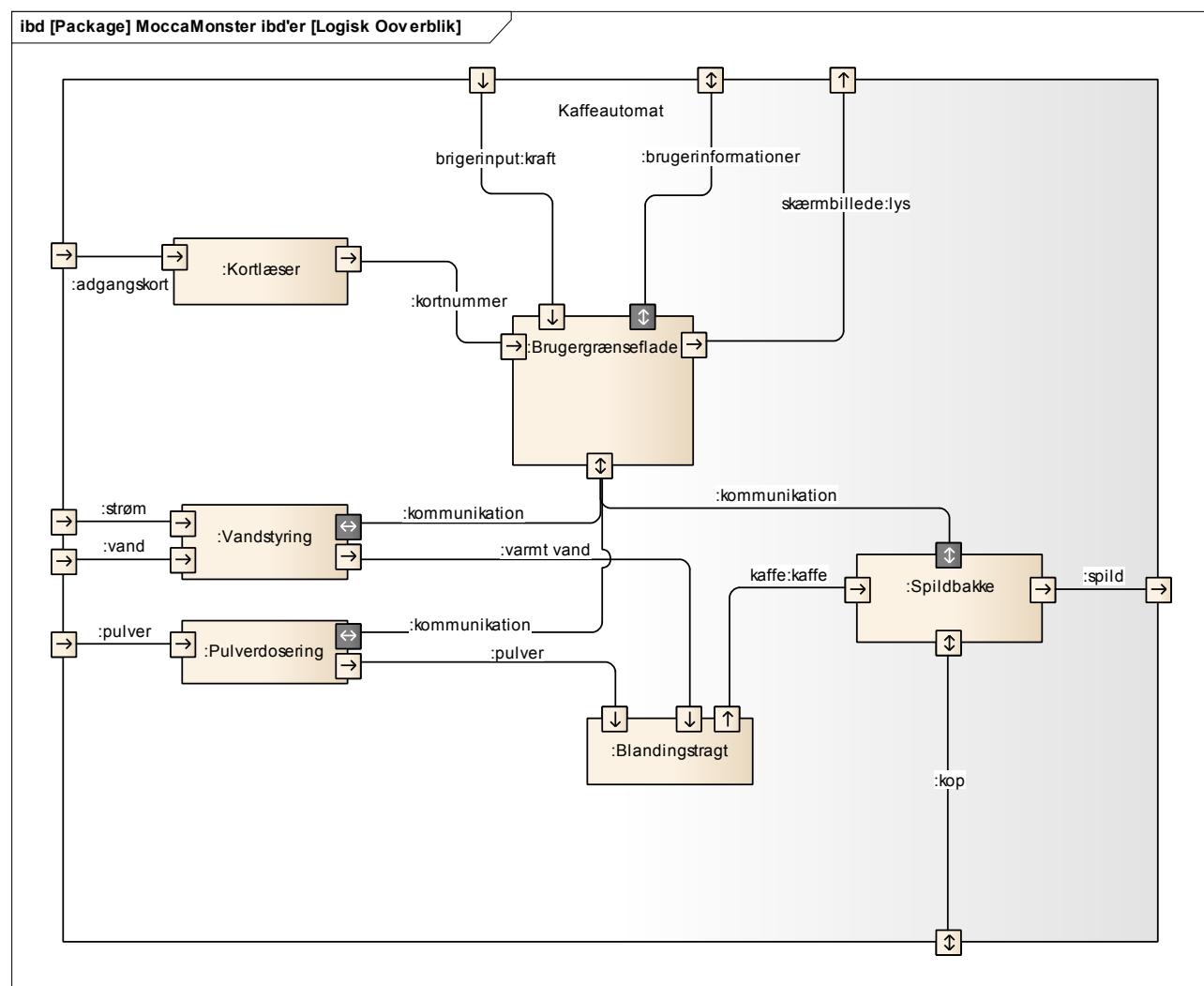
Pulverdosering har ansvaret for at dosere den korrekte mængde kaffepulver ud fra brugers bestilling og føre det ned i blandningstragten. Der skal kunne være en række pulverdoseringenheder, så brugeren får mulighed for at vælge forskellige typer kaffe.

³ Se afsnit 1.4

Blandingstragten er en tragt hvor pulveret og det opvarmede vand føres ned i brugerens kop.

Brugeren placerer sin kop på en rist der agerer som låg på **spilbakken**. Over risten skal udmundingen på blandingstragten placeres. Spilbakken skal både kunne opsamle eventuel spild fra brygningen, men også detektere hvorvidt brugeren har placeret sin kop på risten, så automaten ikke doserer vand og pulver, hvis der ikke er en kop tilstede. Spilbakken skal kunne tømmes manuelt af operatøren.

Det logiske flow mellem disse blokke er beskrevet i det følgende ibd. Kommunikationen mellem brugergrænsefladen, vandstyringen, pulverdoseringen of spilbakken er en udveksling af informationer, vedrørende bestillinger status på brygningen samt eventuelle fejlmeldelser.



*Brugergrænsefladen henter brugerinformationer fra konteringsdatabasen over en ethernet-forbindelse. Protokollen for kommunikationen er beskrevet i protokolbeskrivelsen⁴.

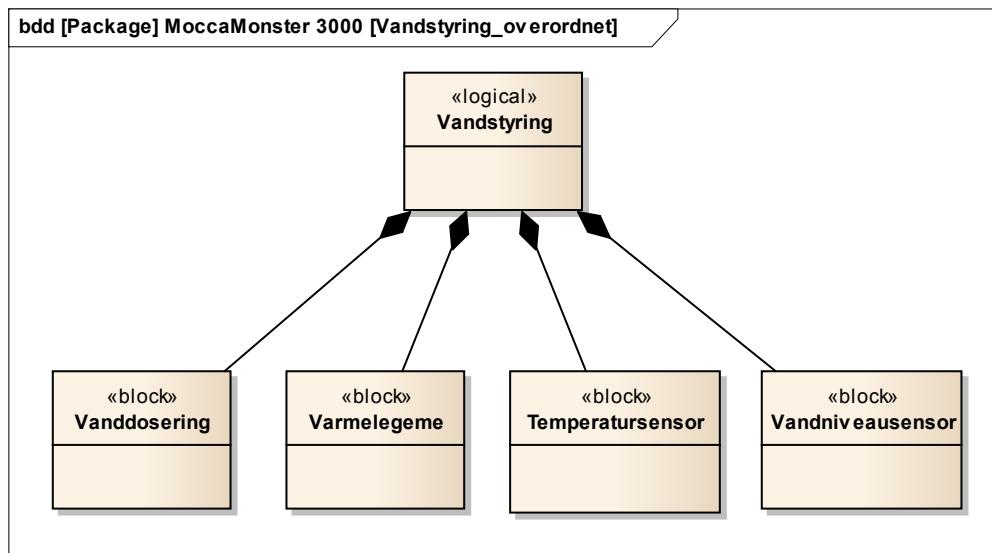
*For at vandstyring skal opvarme vand skal denne have strøm fra en ekstern strømkilde.

⁴ Se afsnit 4.7

Yderligere nedbrydning af systemet

For at de definerede blokke kan udføre deres funktionalitet skal de indeholde en række delsystemer. De blokke der er relevante at nedbryde er beskrevet i de nedenstående tre diagrammer.

Nedenfor ses et blokdiagram for **vandstyring**.



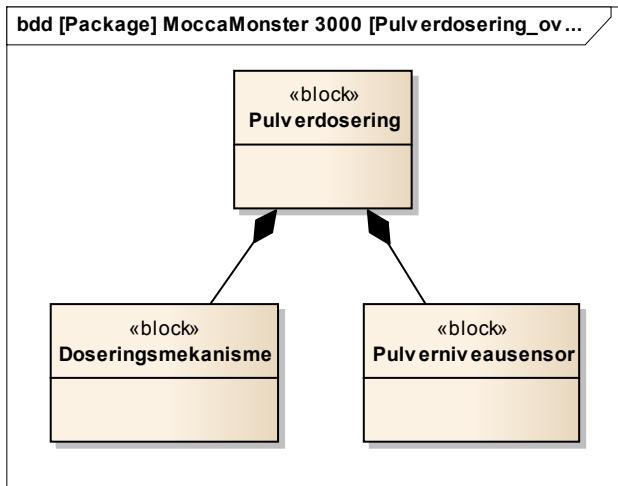
Vandreservoiret er et reservoir af vand der bruges til at brygge kaffe. På reservoiret sidder en **vanddosering**, der står for at dosere den rigtige mængde vand, ud fra hvad brugeren har bestilt. Dertil sidder der en **vandniveausensor**, der detekterer hvorvidt der er tilstrækkeligt med vand i reservoiret. Hvis der ikke er vand til rådighed skal brugergrænsefladen underrettes.

Varmestyring er det delsystem af vandreservoiret, der varetager at opvarme vandet i beholderen. For at kunne dette, skal der være et **varmelegeme** til at opvarme vandet på den fastsatte tid, og en **temperatursensor** så varmestyring kan opvarme vandet til den rigtige temperatur⁵. Varmelegemet skal have strøm fra en ekstern strømkilde for at opvarme vandet i reservoiret.

Temperatursensoren, **vandniveausensoren** og **varmelegemet** befinner sig inde i **vandreservoiret**. Når vandet er varmet tilstrækkeligt op skal den rigtige mængde føres ned i blandingstragten, ved at **vanddoseringen** åbner og lukker for bunden af **vandreservoiret**.

⁵ Se afsnit 1.4

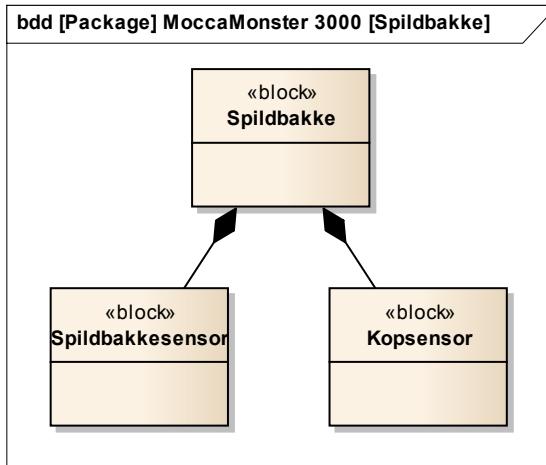
Nedenfor ses et blokdiagram for **pulverdosering**.



Pulverdosering består af ét **pulverreservoir** pr. type kaffe/kakao automaten tilbyder. Doseringen af kaffe/kakao ud af **pulverreservoiret** foregår via en **Doseringsmekanisme**. **Pulverniveausensoren** har til opgave at detektere hvis **pulverreservoiret** er tomt. Hvis **pulverreservoiret** er tomt skal brugergrænsefladen underrettes.

Doseringsmekanismen findes på bunden af pulverreservoiret og pulverniveausensoren er placeret i toppen, inden i pulverreservoiret.

Nedenfor ses et blokdiagram for **spildbakken**.

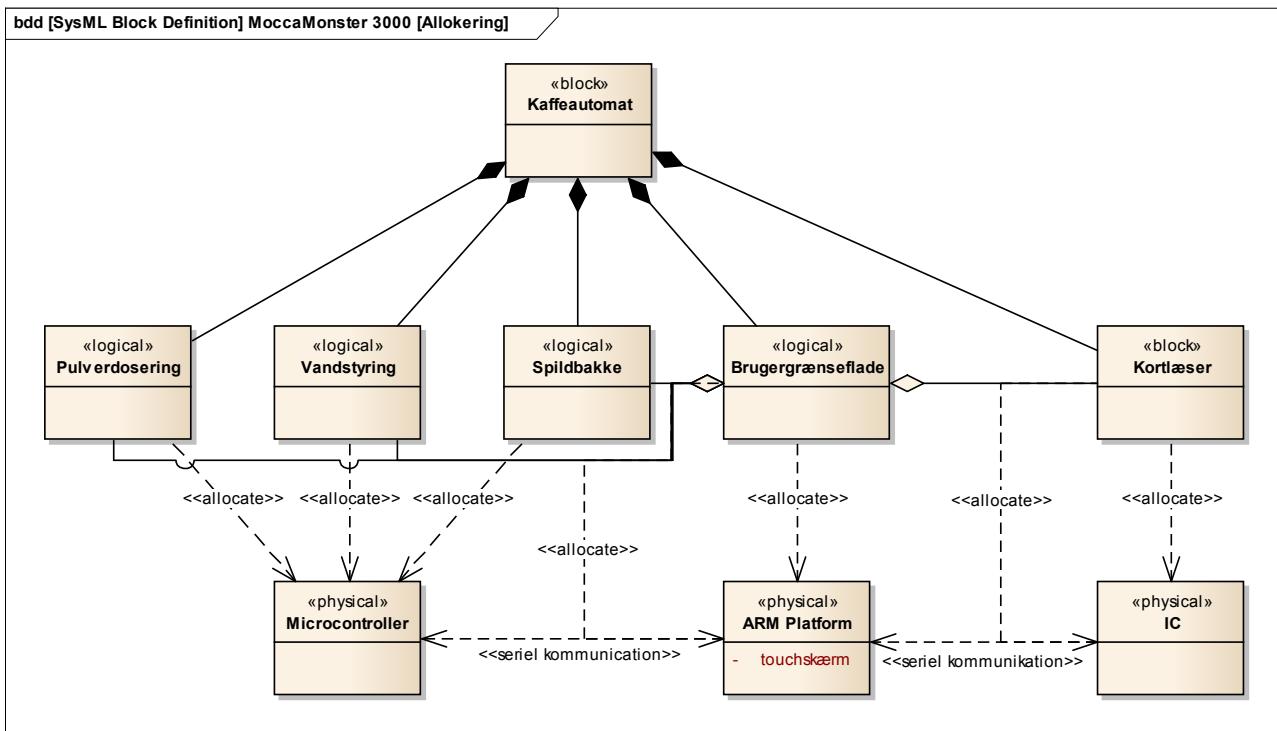


Spildbakken består af et **spildreservoir** med en **kopsensor**. Kopsensoren skal kunne detekterer om der er en kop placeret i spildbakken. Hvis der ikke er placeret en kop i automaten, underbrygningen af kaffe, skal brugergrænsefladen underrettes.

Spildbakken skal designes så spildreservoiret kan tømmes manuelt af operatøren.

2.3. Allokering af logisk funktionalitet i kaffeautomaten

Kaffeautomaten indeholder både en fysisk- og logisk funktionalitet. På følgende diagram vises allokeringen af systemets logiske funktionalitet. I vandstyring, pulverdosering og spildbakken er der en logisk funktionalitet, der kan styre diverse aktuatorer i henhold til målinger på sensorerne. Det ses på allokeringsdiagrammet at pulverdoseringen, vandstyringen og spildbakken logiske funktionalitet er allokeret på samme microcontroller. Brugergrænsefladens logiske funktionalitet er allokeret på en ARM platform med touchskærm og kortlæserens på en passende IC



På det interne blokdiagram for kaffeautomaten er det beskrevet at spildbakken, vandstyringen og pulverdoseringen alle kommunikerer med brugergrænsefladen. Denne kommunikation er allokeret som en seriell kommunikation mellem microcontrolleren og ARM platformen. Ligeledes skal kortlæseren sende et kortnummer til brugergrænsefladen over en seriell kommunikation. Denne serielle kommunikation yderligere beskrevet i signalbeskrivelsen⁶.

⁶ Se afsnit 3.3.2

2.3.1. Krav til microcontroller, arm-platform samt IC

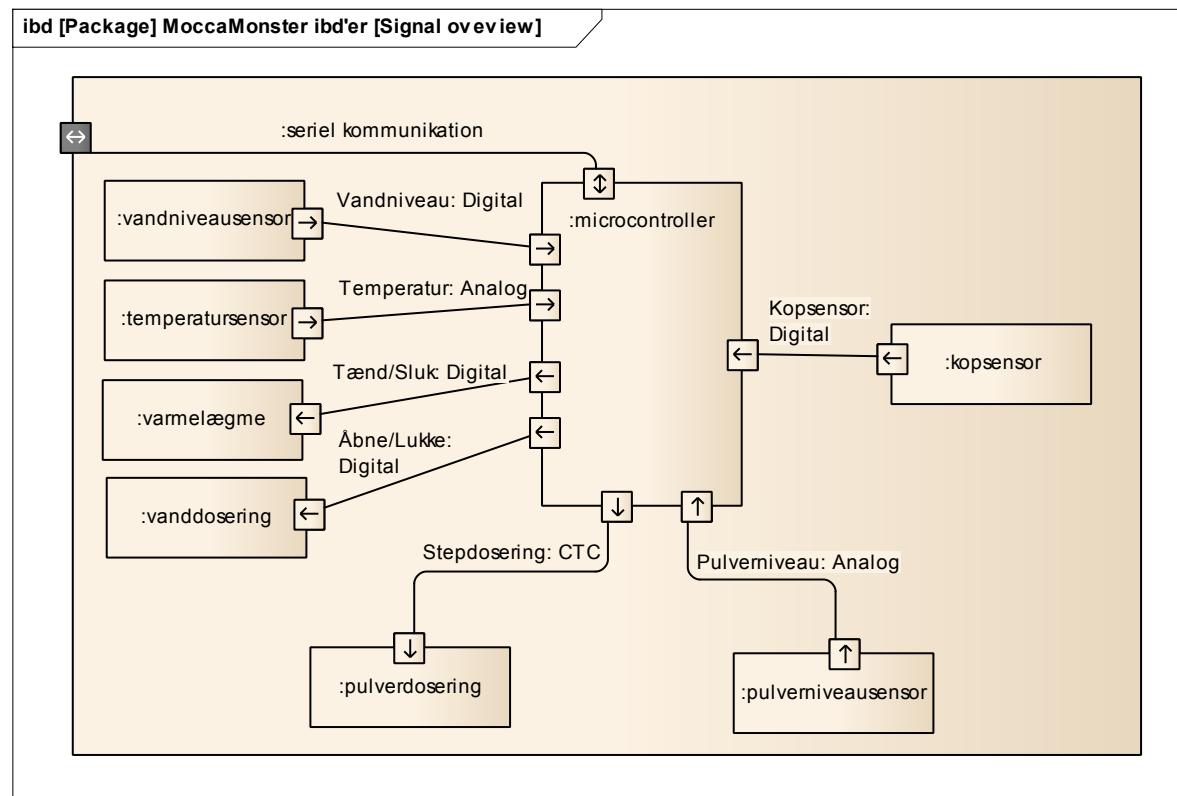
Det er et krav til dette projekt at microcontrolleren, der skal aflæse sensorer og styre aktuatorer er et PSoC3 Starter Kit, og at ARM platformen er et DevKit8000.

Kortlæserens logiske funktionalitet skal allokeres på en **IC**. Der er følgende krav til IC'en

- Kan læse 13,56 MHz RFID tags
- Have seriel kommunikation

Grænseflader for PSoC

I vandstyring, pulverdosering samt spildbakken findes en række blokke der enten agerer som sensor eller aktuator for kaffeautomaten. Alle sensorer sender signaler ind til microcontrolleren, der derefter kan styre diverse aktuatorer. I følgende interne blokdiagram gives et overblik over signalerne mellem microcontrolleren og diverse sensorer og aktuatorer. Alle disse signaler er yderligere beskrevet i signalbeskrivelsen⁷.



⁷ Se afsnit 3.3.2

2.3.2. Signalbeskrivelse

I grænseflader for kaffeautomat er der defineret en række signaler. Disse beskrives i nedenstående tabel

Signaltypen	Definition	Beskrivelse
Analog	Et analogt 0-5V signal.	Et analogt signal til beskrivelse af en fysisk størrelse.
Digital	Et digitalt signal.	Defineret som CMOS standarden
CTC	Et 100Hz digitalt CTC signal.	Defineret som CMOS standarden
Seriell Kommunikation	Seriell kommunikation kører over SPI standarden	Protokollen udvikles specifikt til dette projekt ⁸ .

Vandniveausensoren skal sende et digitalt signal til microcontrolleren. Signalet skal være et digitalt Low, hvis der er tilstrækkeligt med vand i vandreservovert, og et digitalt High, hvis vandreservovert skal opfyldes.

Temperatursensoren skal sende et analogt signal til microcontrolleren. Temperatursensorens skal designes så der måles $0V \pm 0.2V$ ved $0^{\circ}C$ og $5V \pm 0.2V$ ved $100^{\circ}C$

Varmelegemet skal styres med et digitalt signal. For at tænde varmelegemet skal microcontrolleren sende et digitalt High og for at slukke det skal microcontrolleren sende et digitalt Low.

Vanddoseringen skal ligeledes styres med et digitalt signal. For at åbne for vanddoseringen skal microcontrolleren sende et digitalt High og for at lukke den et digitalt Low.

Pulverdoseringen skal bygges således, at microcontrolleren kan styre doseringen i steps med et CTC signal.

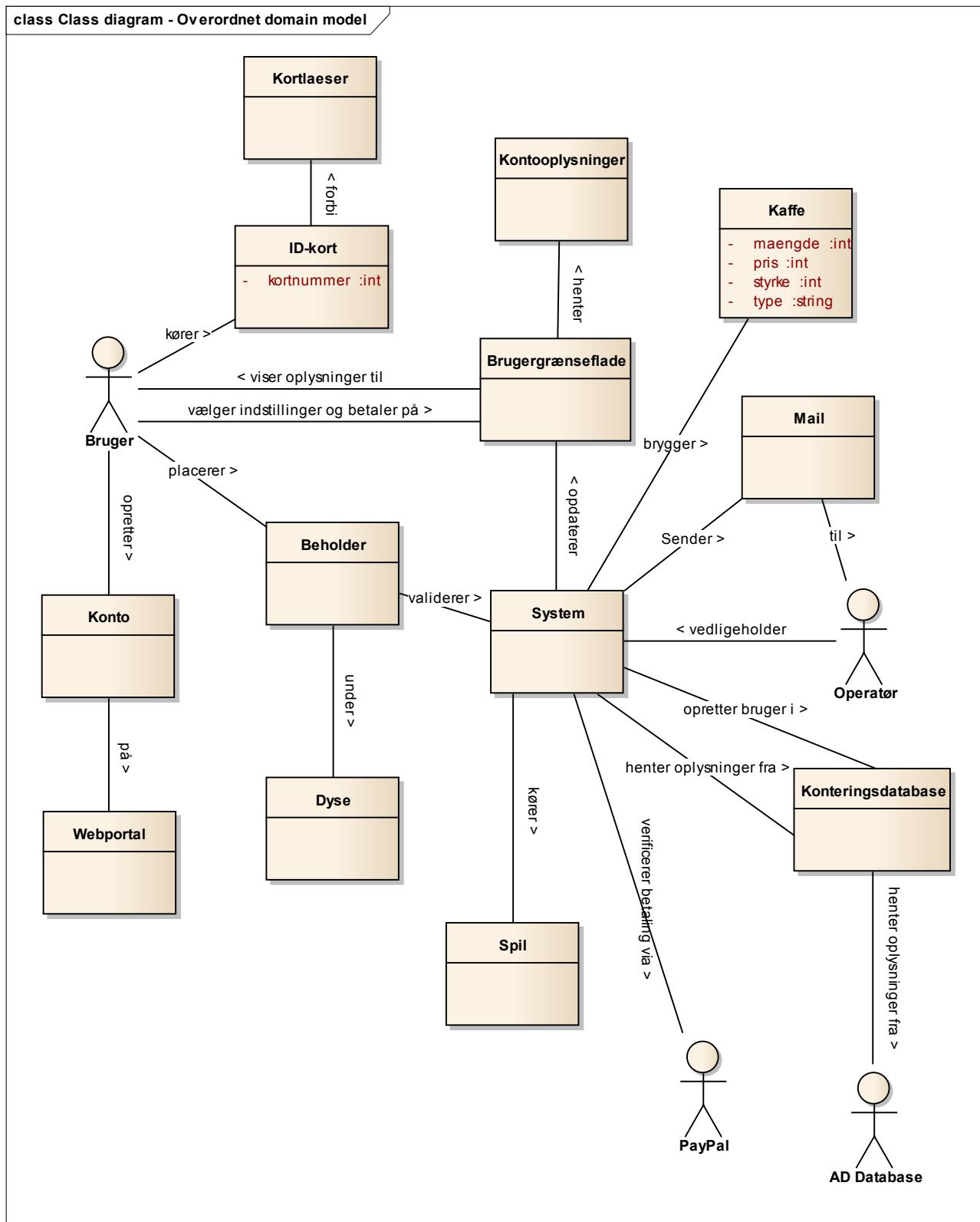
Pulverniveausensoren skal sende et analogt signal til microcontrolleren. Sensoren skal designes så der måles $5V \pm 0.2V$ ved 4cm afstand til pulveret og $0V \pm 0.2V$ ved 20cm afstand.

Kopsensoren skal sende et digitalt signal til microcontrolleren. Hvis der er placeret en kop foran microcontrolleren skal der måles et digitalt High, ellers skal der måles et digitalt Low.

⁸ Se afsnit 4.7

2.4. Domæneanalyse

Det overordnede domæne for interaktionen mellem de forskellige delelementer i systemet er beskrevet i følgende domænemodel. Den er udviklet på baggrund af en analyse af de udarbejdede Use Cases samt de definerede blokke i systemarkitekturen.



3. Softwarearkitektur

Dette afsnit beskriver softwarearkitekturen for systemet "MoccaMonster 3000", som formuleret i projekt-formuleringen og specificeret i kravspecifikationen. Afsnittet giver udviklerne mulighed for at udarbejde specificeret software indenfor fastlagte rammer. Afsnittet indeholder henholdsvis dokumentation af design, implementering og test af softwaren.

På baggrund af systemarkitekturen er softwaren bestemt til at bestå af tre blokke⁹, henholdsvis brugergrenseflade, webfront og konteringsdatabase, som beskrives herunder.

3.1. Blok- og aktørbeskrivelse

Her beskrives kort softwarens tre overordnede blokke. Derudover beskrives AD databasen, som institutionen skal have implementeret i forvejen.

3.1.1. Brugergrænseflade

Brugergrænsefladen er allokeret på en ARM platform. Brugergrænsefladens overordnede funktionalitet indebærer:

- At fungere som brugergrænseflade mellem system og bruger
- At kommunikere med konteringsdatabasen og AD databasen
- At foretage udregninger på baggrund af brugerens valg og videregive disse til PSoC'en
- At modtage data fra PSoC'en og bearbejde disse

3.1.2. Webfront

Webfronten er allokeret på en webserver og har til formål at

- Lade brugeren se saldo på sin konto
- Lade brugeren optanke penge på sin konto

3.1.3. Konteringsdatabase

Konteringsdatabasen er allokeret på en webserver og har til formål at holde informationer om brugerne. Den har desuden til formål at:

- Indeholde brugerinformation i form af ID på brugerens adgangskort, og brugernavn til systemet
- Indeholde information om transaktioner for hver enkelt bruger
- Indeholde information om forskellige kaffetyper der findes i maskinen, og dertilhørende data såsom pris, blandingsforhold og mængde af kaffe tilbage i beholderne
- Indeholde en log for hændelser på systemet

3.1.4. AD Database

Active Directory er en database udviklet til at håndtere brugere i store organisationer. Det er en fast implementeret del af Microsoft Windows Server suiten([henvisning?](#)) og er dermed inden for alle brancher blevet de facto standard til brugerhåndtering. Databasen tillader LDAP([henvisning?](#)) opslag, og er der-

⁹ Se afsnit 3.1.1

med nem at have med at gøre i forbindelse med udvikling af systemer hvor udlæsning af brugeroplysninger har relevans.

I MoccaMonster3000 benyttes LDAP opslag til at verificere brugere, og logge ind på webfronten. Desuden bruges LDAP til at udlæse navne på den pågældende bruger til brug på brugergrænsefladen.

3.2. Design af brugergrænseflade

Afsnittet indeholder

- En overordnet gennemgang af use cases via sekvensdiagrammer
- Definition af grænseflader og problemdomæner ud fra domænemodellen
- Oprettelse af grænseflade- og domæneklasser
- Detaljerede applikationsmodeller, pakkediagrammer og sekvensdiagrammer

Formålet med afsnittet er:

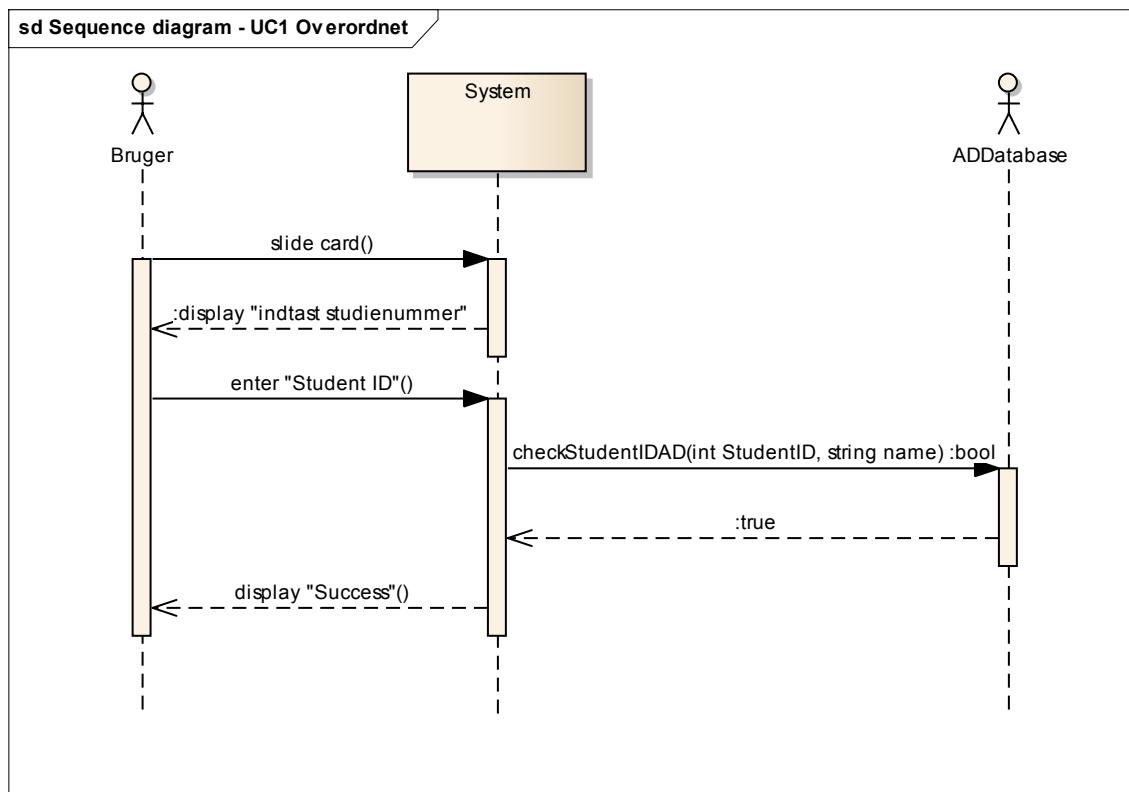
- At give overblik over software designets struktur
- At fastlægge systemets detaljerede softwarestruktur, drevet af kravspecifikationen og systemarkitekturen
- At fastlægge systemets softwareklasser og deres indbyrdes interaktioner
- At definere grænsefladerne imellem blokkene

Overordnede sekvensdiagrammer

Før der går i gang med applikationsmodellerne vises overordnede sekvensdiagrammer for hver use case. Formålet med disse diagrammer er at give et overblik over systemets funktionalitet i henhold til kravspecifikationen, og at gøre det lettere at forstå systemet i forhold til den ydre verden. Systemets interaktion med aktørerne vises og de første metoder identificeres.

3.2.1.1. Overordnet sekvensdiagram for use case 1 "Opret konto"

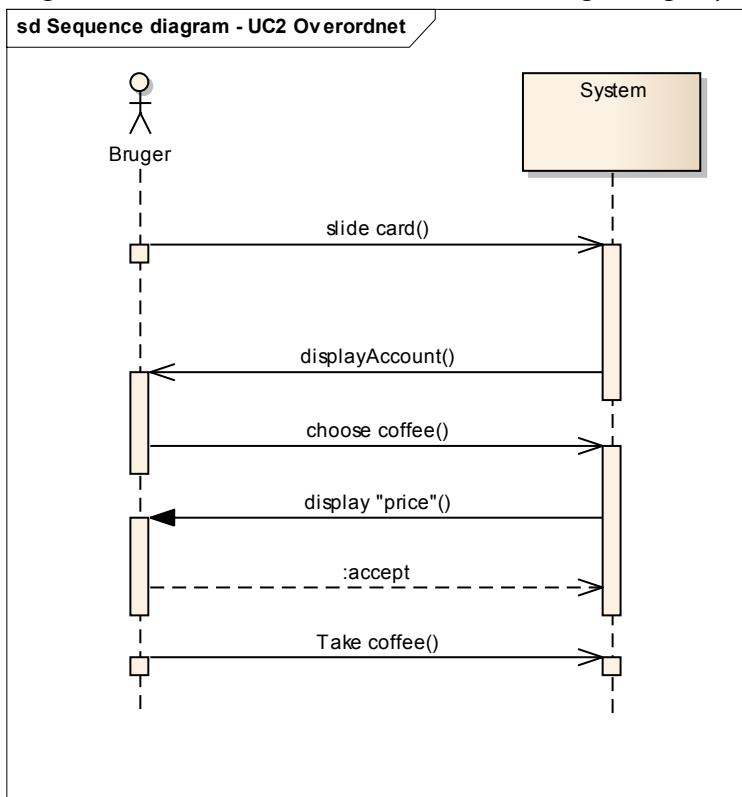
Diagrammet viser interaktion mellem bruger, system og AD database for use case 1, "Opret konto".



Figur 5 - Overordnet sekvensdiagram for UC1

3.2.1.2. Overordnet sekvensdiagram for use case 2 "Køb af kaffe"

Diagrammet viser interaktion mellem bruger og system for use case 2, "Køb af kaffe".

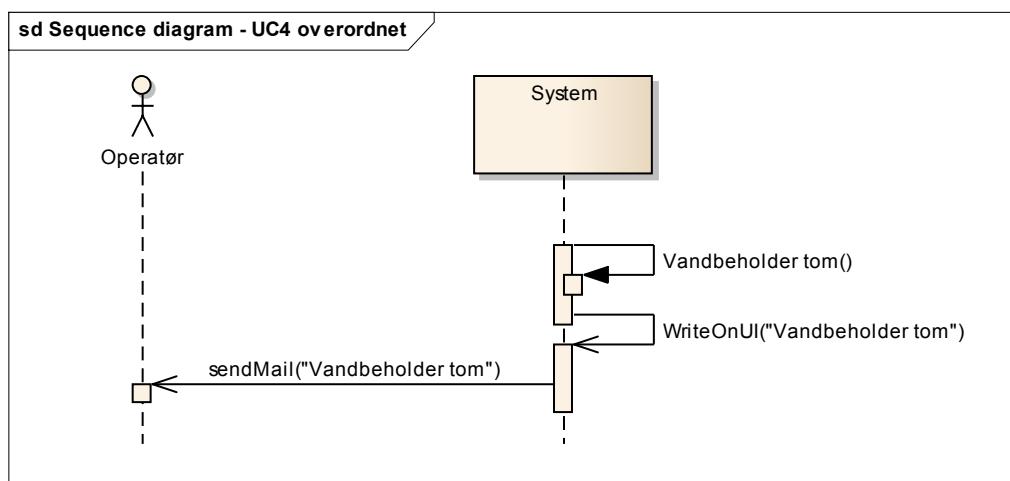


Figur 6 - Overordnet sekvensdiagram UC2

3.2.1.3. Overordnet sekvensdiagram for use case 4 "Vedligeholdelse af systemet"

Til at beskrive funktionaliteten ved fejl eller andre mangler i systemet, som kræver operatørens opmærksomhed, gennemgås både use case 4 og afsnittet "Egenkontrol af systemet". Første del af designet omhandler systemtjekket, som foretages efter hver bestilling hvorefter use case 4 køres. Dermed illustreres hele fejlhåndteringsprocessen.

Desuden viser diagrammet interaktionen mellem operatør og system.

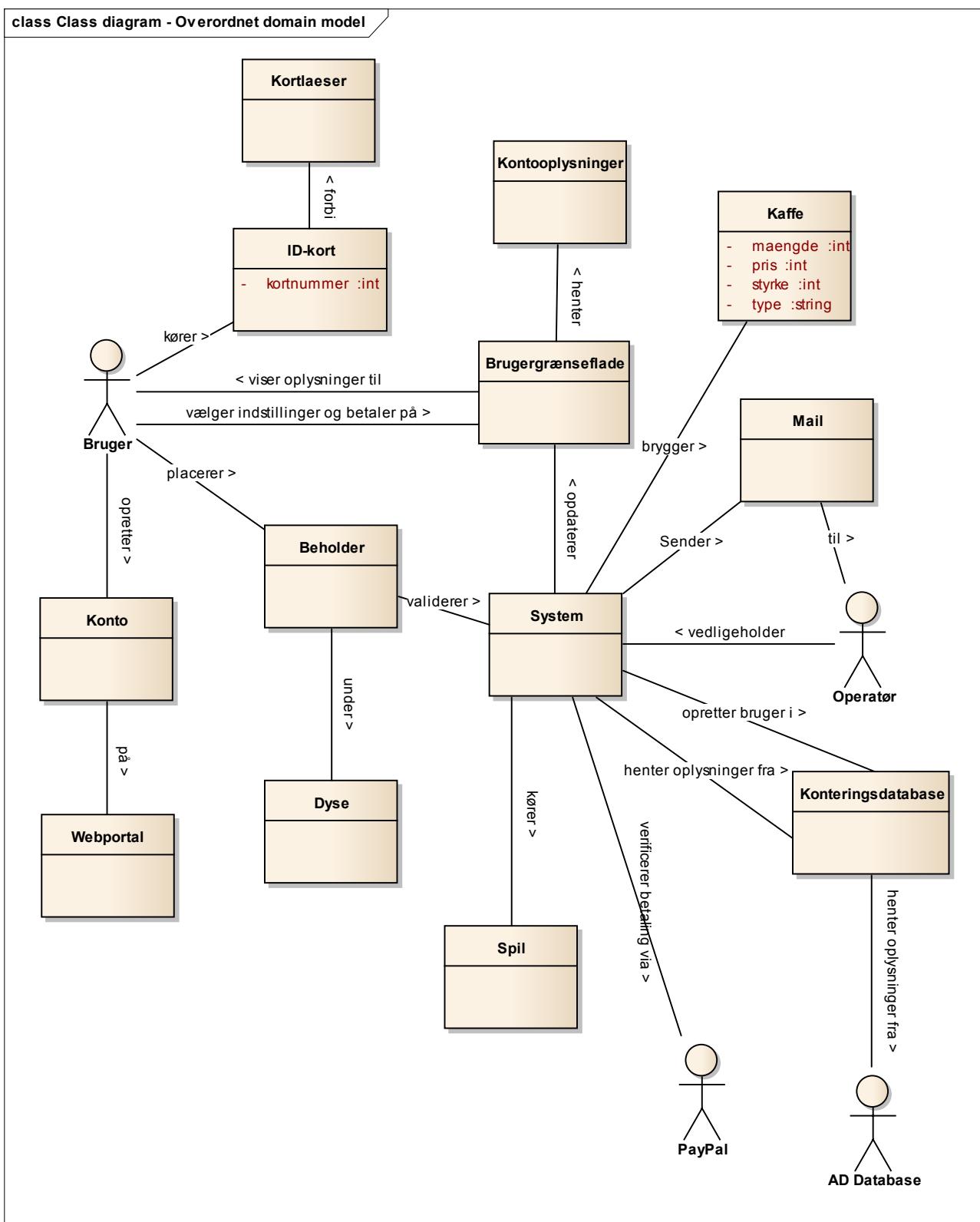


Figur 7 - Overordnet sekvensdiagram UC4

3.2.2. Problemidentifikation

I forlængelse af de overordnede sekvensdiagrammer påbegyndes nu den indledende designproces. Dette gøres ved at udarbejde en domænemodel. I domænemodellen er der identificeret konceptuelle klasser ud fra use casene¹⁰, hvor hver klasse har til ansvar at løse et specifikt problem. I det følgende bliver der, på baggrund af modellen, defineret problemer som systemet skal løse.

¹⁰ Se afsnit 1.3



Figur 8 - domænemodel

Softwareen er designet så klasserne har så specifikt og indskrænket ansvar som muligt. Formålet med dette er at give hver klasse en højere selvstændighed, og mindske koblingen mellem dem. Der defineres klasser ud fra to typer problemer, grænseflade og domæne. Grænsefladeproblemerne består af de problemer der opstår når to dele af systemet skal kommunikere. Domæneproblemerne er de resterende problemer systemet skal løse. Ud fra domænemodellen er følgende problemer defineret:

- Grænseflader
 - Brugergrænseflade til bruger
 - Grænseflade til kortlæser
 - Grænseflade til den kaffebryggende del af systemet
 - Grænseflade til databaserne
- Domæner
 - Håndtering af den aktuelle brugers informationer
 - Håndtering af de specifikke parametre for valgt kop
 - Efter hvilken protokol der skal kommunikeres mellem PsoC og DevKit

3.2.3. Klasseidentifikation

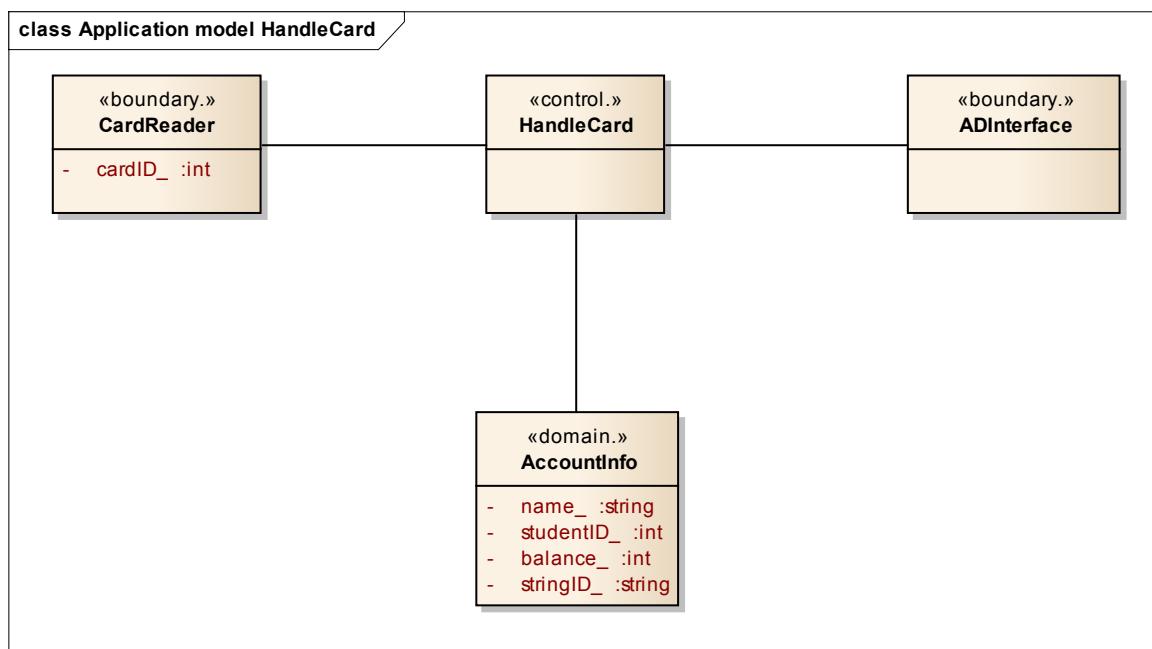
Ud fra domænemodellen udarbejdes applikationsmodeller. Dette gøres for at identificere systemets klasser og hver klassens individuelle formål. Hver applikationsmodel har til opgave at vise klasserne, som er involveret i den pågældende use cases funktionalitet. Dette håndteres af kontrol klassen. Grænseflade- og domæneklasserne er bestemt ud fra henholdsvis grænseflade- og domæneproblemerne. Der er identificeret følgende klasser:

- Boundaryklasser
 - **UserInterface** - Brugergrænseflade til bruger
 - **cardReader** - Grænseflade til kortlæser
 - **Sensors** - Grænseflade til den kaffebryggende del af systemet
 - **AccountDatabase** – Grænseflade til konteringsdatabasen
 - **ADinterface** - Grænseflade til AD databasen
 - **EmailSender** – Grænseflade til operatør og til bruger
- Domainklasser
 - **AccountInfo** – Håndtering af den aktuelle brugers informationer
 - **Coffee** – Håndtering af de specifikke parametre for valgt kop
 - **Protocol** – Hvilk protokol der skal kommunikeres mellem PsoC og DevKit

Ud fra use casene og de definerede klasser, kan der nu oprettes applikationsmodeller. Applikationsmodellerne består af henholdsvis en kontrolklasse der varetager use casens forløb, samt de definerede domain- og boundaryklasser, som kontrolklassen skal bruge for at opfylde dette.

3.2.3.1. Klassediagram for "handle card"

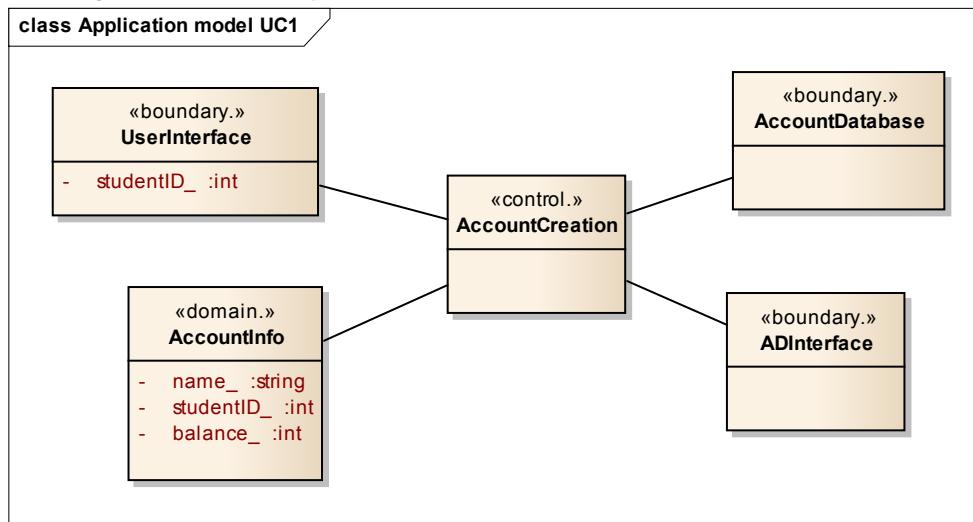
Da systemet altid skal opføre sig ens i starten af use case 1 og use case 2, oprettes der en sekvens til dette. Dermed slippes der for gentagelser af denne sekvens i modellerne for disse use cases.



Figur 9 - Klassediagram HandleCard

HandleCard er kontrolklassen, som håndterer den sekventielle gennemgang fra kortet køres forbi kortlæseren til en af de to use cases udspiller sig.

3.2.3.2. Klassediagram for use case 1 "Opret konto"



Figur 10 - Klassediagram UC1

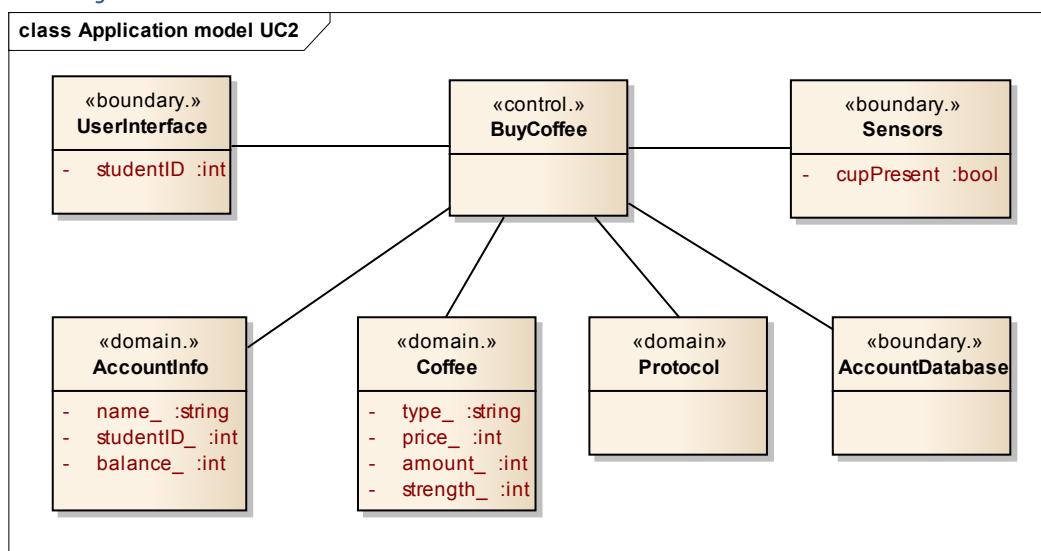
Dette klassediagram er benyttet til at identificere nødvendige klasser for at udføre use case 1. Nedenfor er beskrives kort klassernes ansvar for use case 1.

AccountCreation varetager og koordinerer interaktionen mellem de interne klasser i henhold til use casen.

AccountInfo indeholder den aktuelle brugers informationer. Informationerne indebærer navn, studienummer og saldo på kontoen.

UserInterface er grænseflade klassen mellem system og bruger.

3.2.3.3. Klassediagram use case 2 "Køb af kaffe"



Figur 11 - Klassediagram UC2

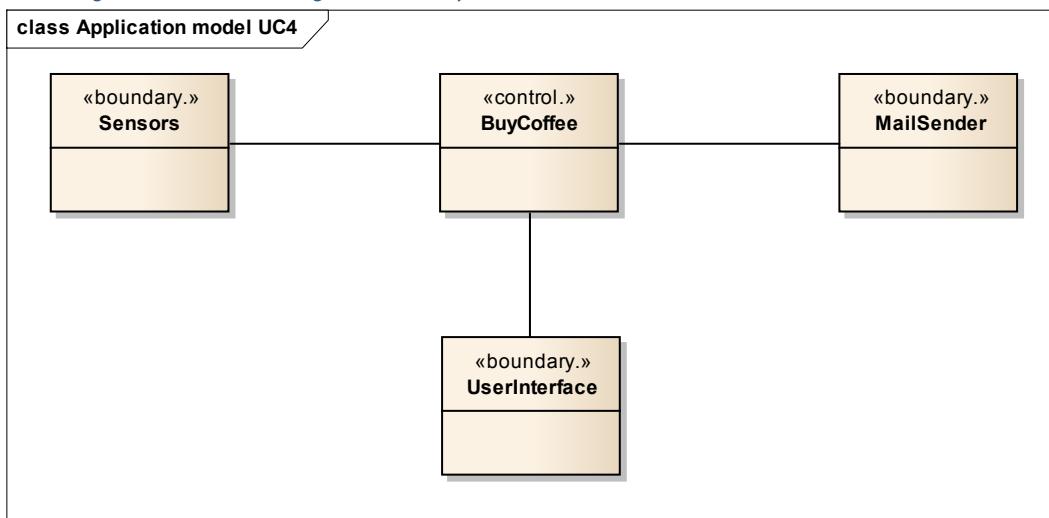
BuyCoffee varetager og koordinerer interaktionen mellem de interne klasser i henhold til use casen.

Sensors håndterer kommunikationen med PSoC'en gennem en seriel protokol.

Protocol indeholder den protokol som Sensors skal bruge, når der kommunikeres med PSoC'en.

Coffee gemmer brugerens valg om den aktuelle brygning, som skal videresendes til PSoC'en. Klassen indeholder ligeledes information om den kaffe der er valgt af brugeren. Dette indebærer pris, type og blandingsforhold.

3.2.3.4. Klassediagram use case 4 "Vedligeholdelse af systemet"



Figur 12 - Klassediagram UC4

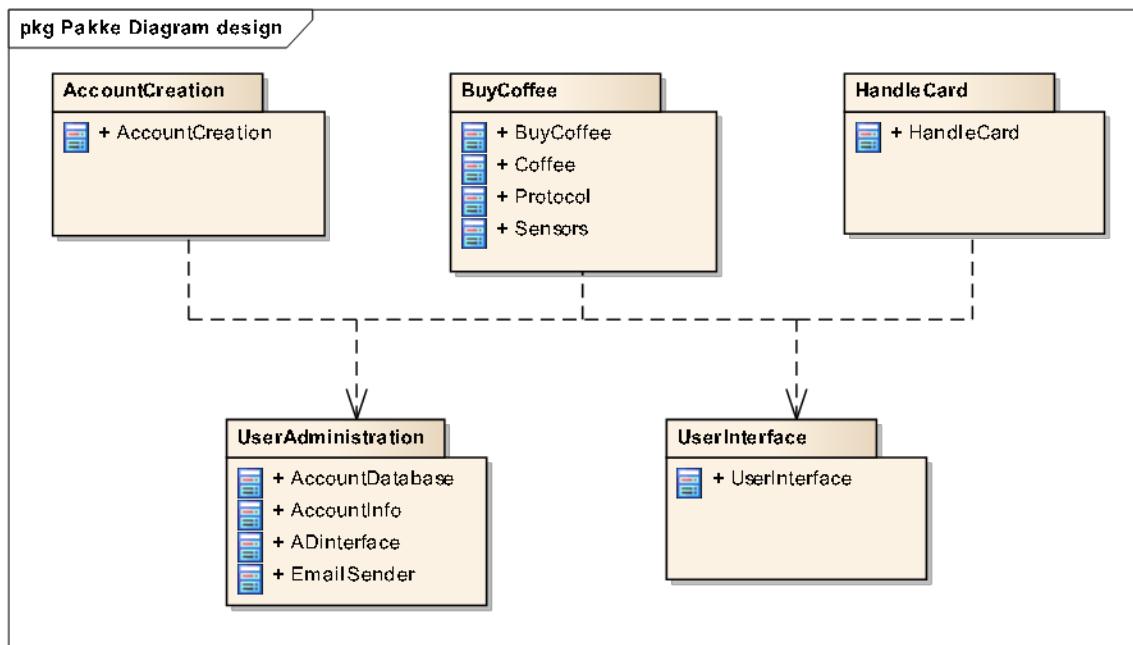
BuyCoffee er kontrol klassen, men i denne use case skal den i stedet fortolke og behandle fejl, som bliver sendt via klassen **Sensors**.

Sensors er grænsefladen mellem ARM-platformen og microprocessoren. Denne modtager fejl og videresender dem til kontrol klassen.

MailSender er klassen, som sender mails til operatøren hvis en fejl forekommer/opstår.

Pakkediagram

Dette afsnit indeholder et samlet pakkediagram for brugergrænsefladen. På pakkediagrammet kan det ses hvordan systemet er delt op. Systemet er designet så pakkerne er inddelt efter hvilke use cases de forskellige klasser tilhører. De klasser som bruges af flere use cases er delt op i pakker efter funktionalitet. I disse pakke er der ingen intern kommunikation, da de kun kan tilgås af kontrolklasserne, henholdsvis AccountCreation, BuyCoffee og HandleCard. På den måde sikres en lavere kobling i systemet.



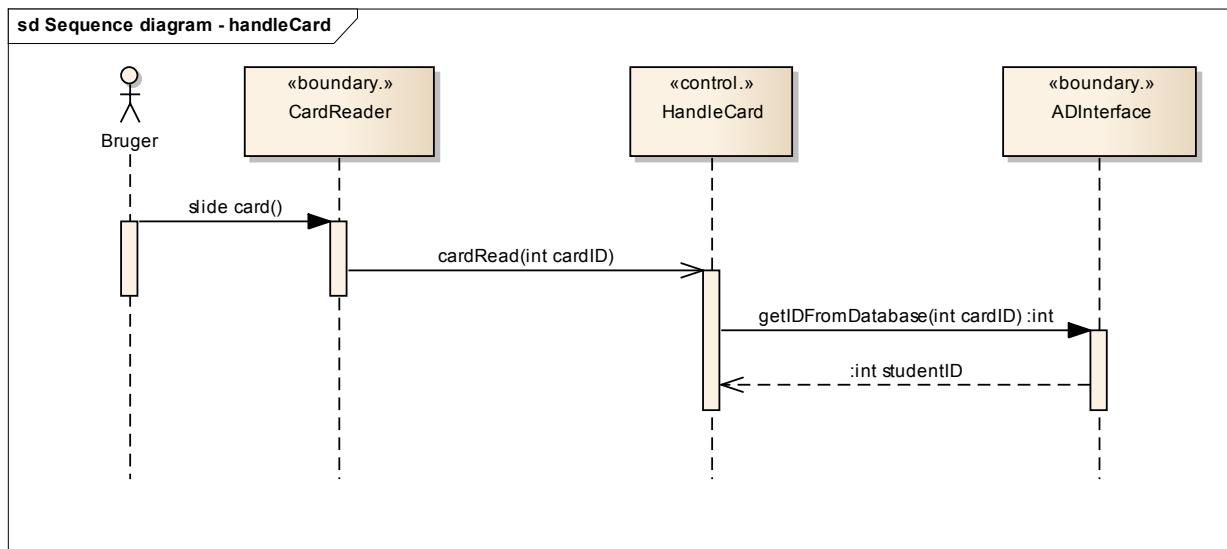
Figur 13 - Pakkeinddeling designfase

3.2.4. Metodeidentifikation

Der kan laves sekvensdiagrammer på baggrund af de nyligt fundne klasser og use cases. Ud fra diagrammerne kan de grundlæggende metoder, som fuldfører use casens funktionalitet, defineres. I sekvensdiagrammerne er det kun hovedscenariet fra use casene, som er beskrevet. Undtagelserne er beskrevet ved hjælp af state machines.

3.2.4.1. Sekvensdiagram "Handle card"

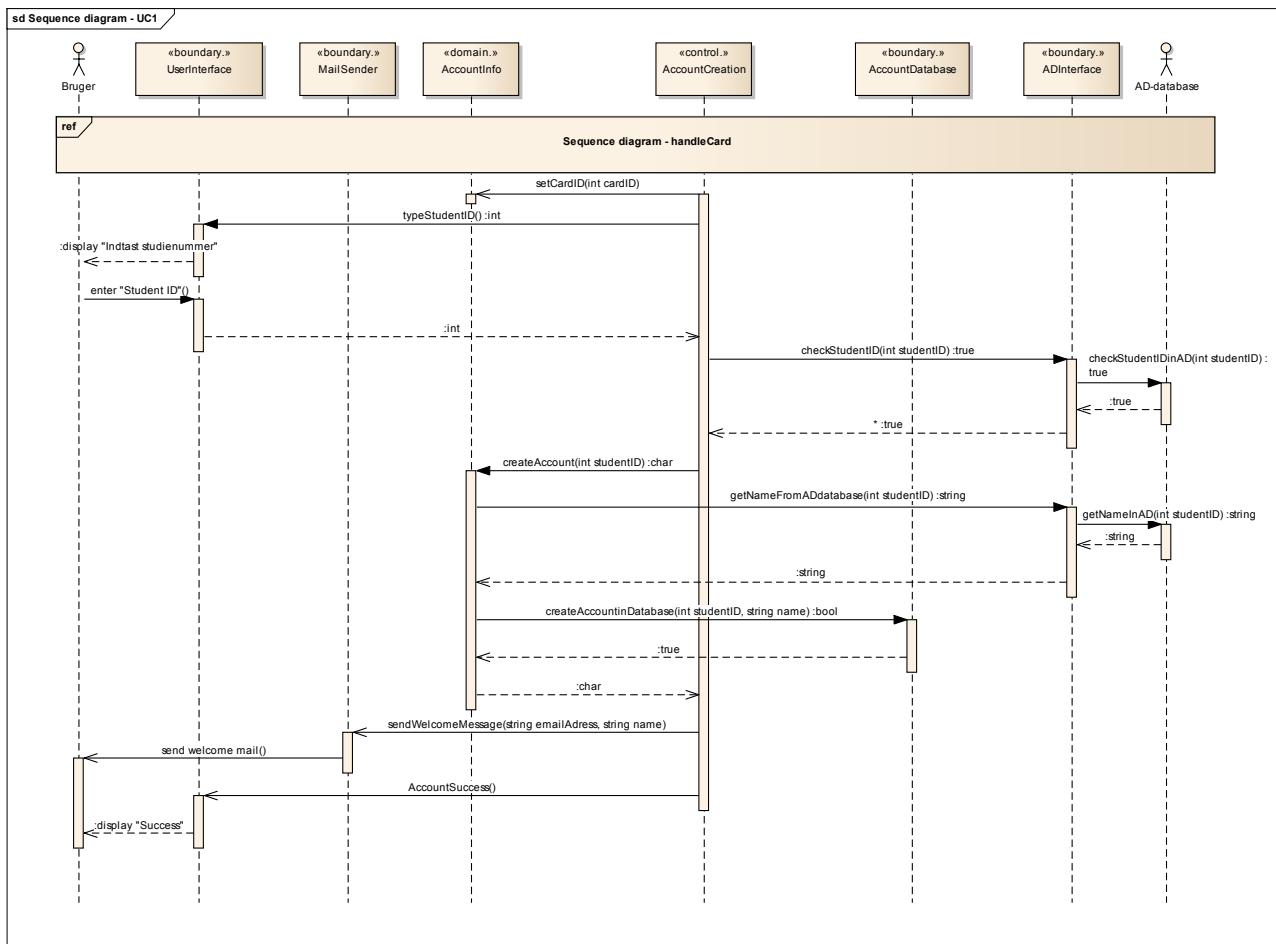
I forlængelse af applikationsmodellerne, laves der et sekvensdiagram for HandleCard. Sekvensen viser starten af use case 1 og 2, da de er ens. Diagrammet stopper når HandleCard definerer hvorvidt brugeren er oprettet i systemet i forvejen eller ej. Hvis brugeren ikke er oprettet køres use case 1 og hvis brugeren ikke er oprettet startes use case 2..



Figur 14 - Sekvensdiagram HandleCard

3.2.4.2. Sekvensdiagram use case 1 "Opret konto"

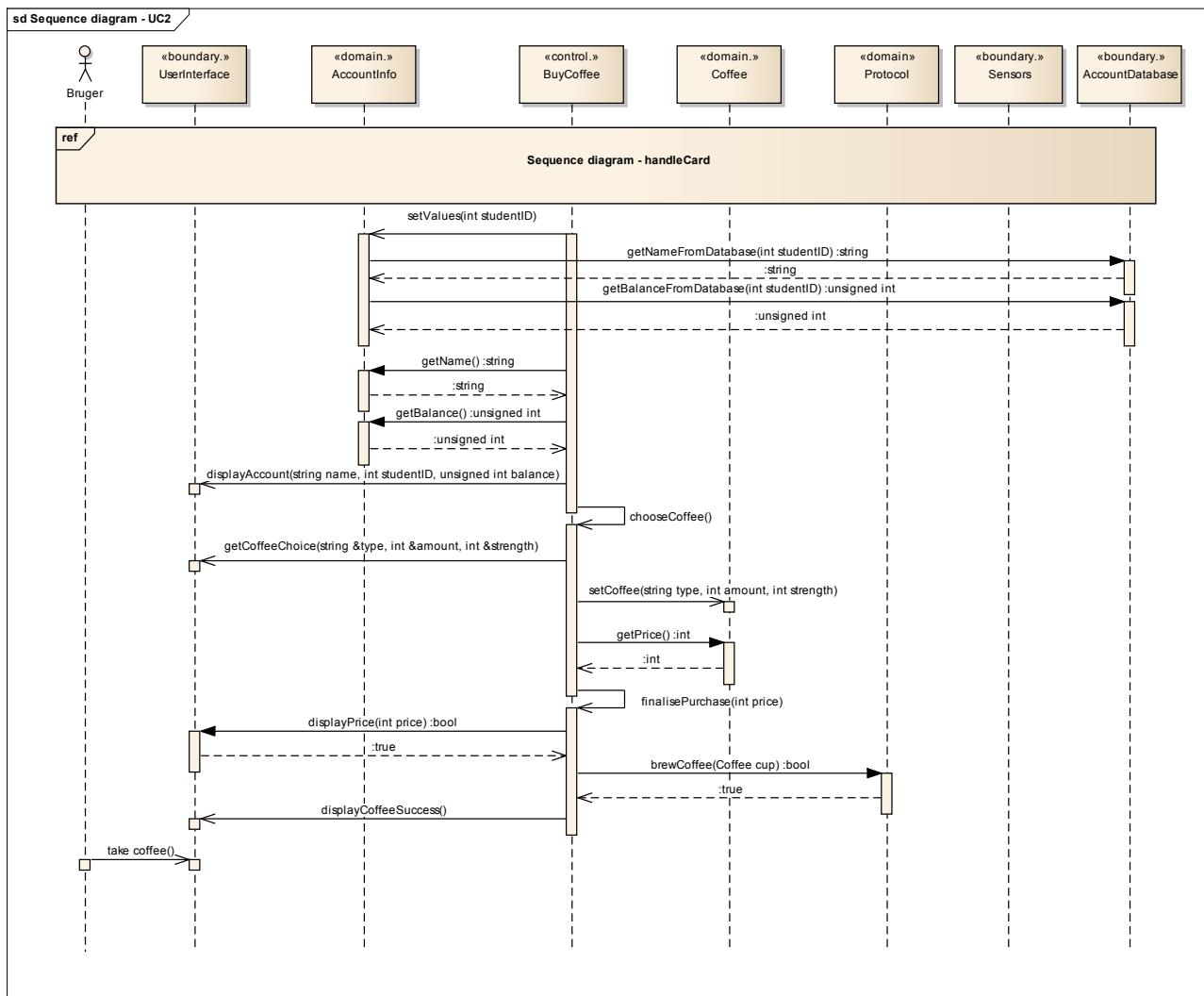
Den sekventielle gennemgang for use case 1 beskrives her ved hjælp af et sekvensdiagram.



Figur 15 - Sekvensdiagram UC1

3.2.4.3. Sekvensdiagram use case 2 "Køb af kaffe"

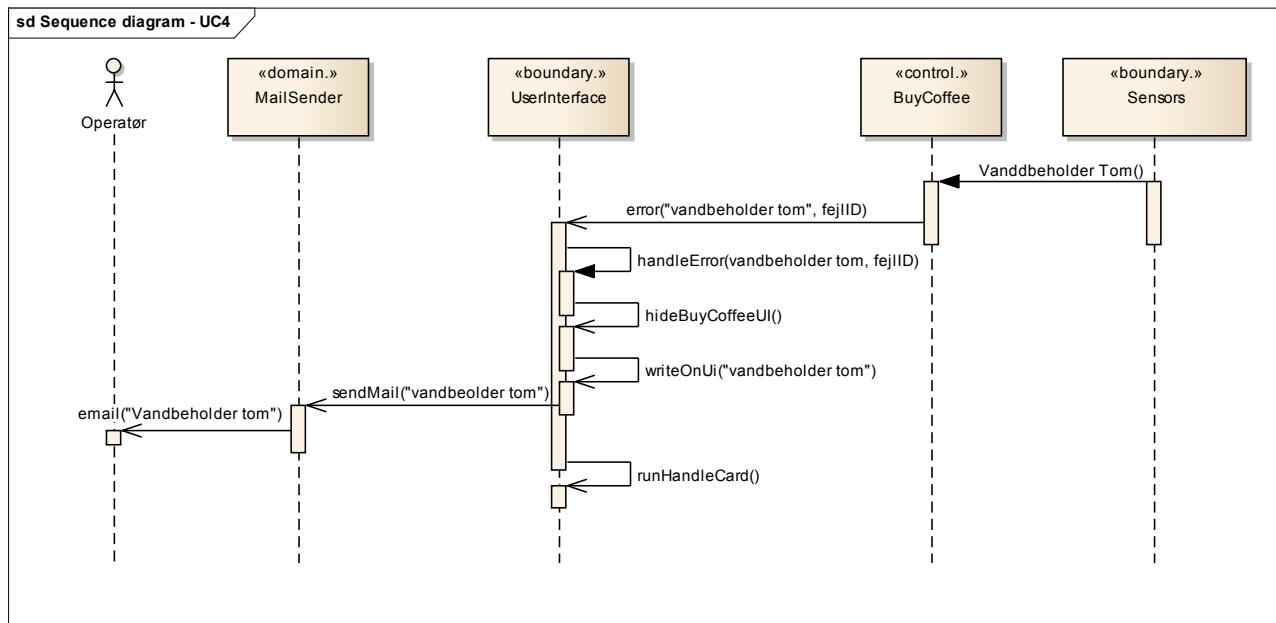
Den sekventielle gennemgang for use case 2 beskrives her ved hjælp af et sekvensdiagram.



Figur 16 - Sekvensdiagram UC2

3.2.4.4. Sekvensdiagram use case 4 "Vedligeholdelse af systemet"

Den sekventielle gennemgang for use case 4 beskrives her ved hjælp af et sekvensdiagram.



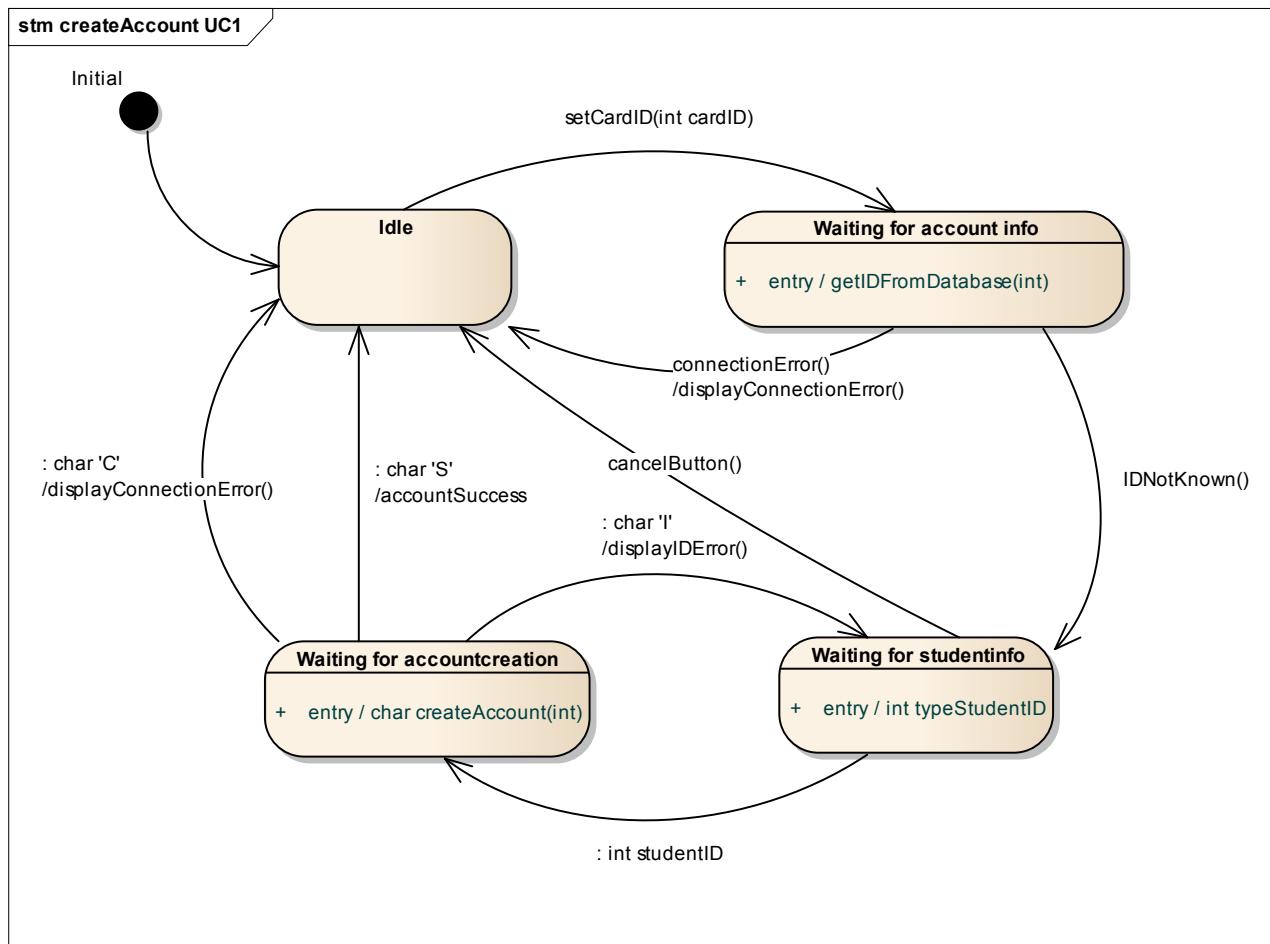
Figur 17 - Sekvensdiagram UC4

3.2.5. Uddybende kontrolklasse beskrivelse

For at give yderligere indblik i hvordan kontrol klassen varetager sine opgaver, beskrives softwaren som state machines. Formålet med state machines er at vise systemets tilstande igennem use casen. Her beskrives også hvordan de specificerede undtagelser fra use casen håndteres af kontrolklassen.

3.2.5.1. State machine use case 1 "Opret konto"

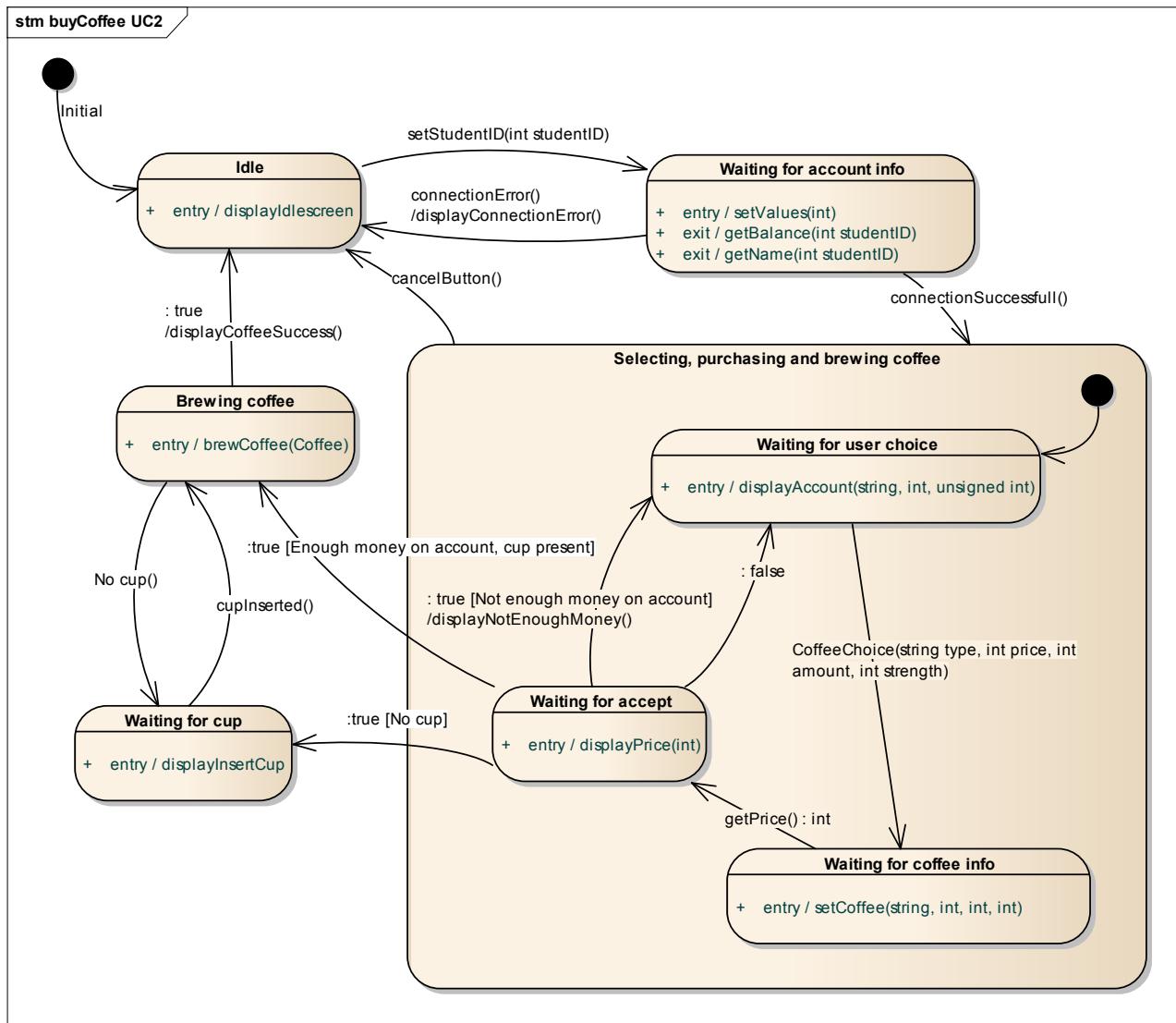
Det ses at systemet altid vil returnere til et idle state, efter at have håndteret hver brugers anmodning. Desuden ses det at der er identificeret nye metoder til at håndtere de specificerede undtagelser.



Figur 18 - Statemachine UC1

3.2.5.2. State machine use case 2 “Køb af kaffe”

Her identificeres yderligere metoder til undtagelseshåndtering når brugeren vil købe kaffe. Desuden ses det at det skal være muligt at annullere kaffekøbet.



Figur 19 - Statemachine UC2

3.3. Implementering af Brugergrænseflade

Afsnittet indeholder følgende:

- Beskrivelse af Qt frameworket
- Beskrivelse af tråde og hvordan de er brugt i dette system

3.3.1. Qt

Qt er et framework til softwareudvikling i C++, hvis brug kan spredes over mange platforme. Som udgangspunkt bruges Qt frameworket til at udvikle grafiske brugergrænseflader i alle størrelser. Qt Project har under udviklingen taget udgangspunkt i biblioteker der bruges i C++ og videreudviklet dem til QT Frameworket. Det resulterer i et framework der er yderst fleksibelt og råder over store udviklingsmuligheder. Med Qt 5 er der blandet andet lavet ikke-GUI relaterede features som SQL databaser, XML parsing, tråd-programmering, netværksunderstøttelse og API¹¹.

3.3.2. Tråde

Da der er mere funktionalitet, der skal køre sideløbende, bruges der tråde. Resten af programmet skrives i Qt frameworket, så trådsystemet derfra bruges også her. Til formålet nedarves alle aktive klasser fra klassen QThread. Alle Funktionaliteterne deles op i disse tråde:

1. **HandleCard tråd**, som håndterer den sekventielle funktionalitet, der udspiller sig fra brugeren kører sig kort forbi kortlæseren til en use case forekommer.
2. **AccountCreation tråd** som håndterer den overordnede sekventielle funktionalitet, som er beskrevet i sekvensdiagrammerne. Det bliver derfor kontrolklasserne, som bliver kørt i den.
3. **BuyCoffee tråd** som håndterer den overordnede sekventielle funktionalitet, som er beskrevet i sekvensdiagrammerne. Det bliver derfor kontrolklasserne, som bliver kørt i den.
4. **Brugergrænseflade tråd**, som håndterer UserInterface klassen og dens funktionalitet. Denne tråd er den eneste, som kan administrere den grafiske brugergrænseflade
5. **Kortlæser tråd**, som håndterer CardReader klassen og dens funktionalitet. Denne tråd laver blocking read på UART'en.
6. **Sensors tråd**, som håndterer Sensors klassen og dens funktionalitet. Tråden tjekker om der er kommet noget fra PSoC'en en gang i sekundet.

Da ovennævnte tråde skal kommunikere med hinanden, er det nødvendigt at bruge et beskedsystem. Qt har sit eget beskedsystem i form af signal & slots. Når en besked skal sendes, sendes et signal som underretter de tråde, som er forbundet til signalet. Dermed er beskedsystemet et publisher/subscriber-system.

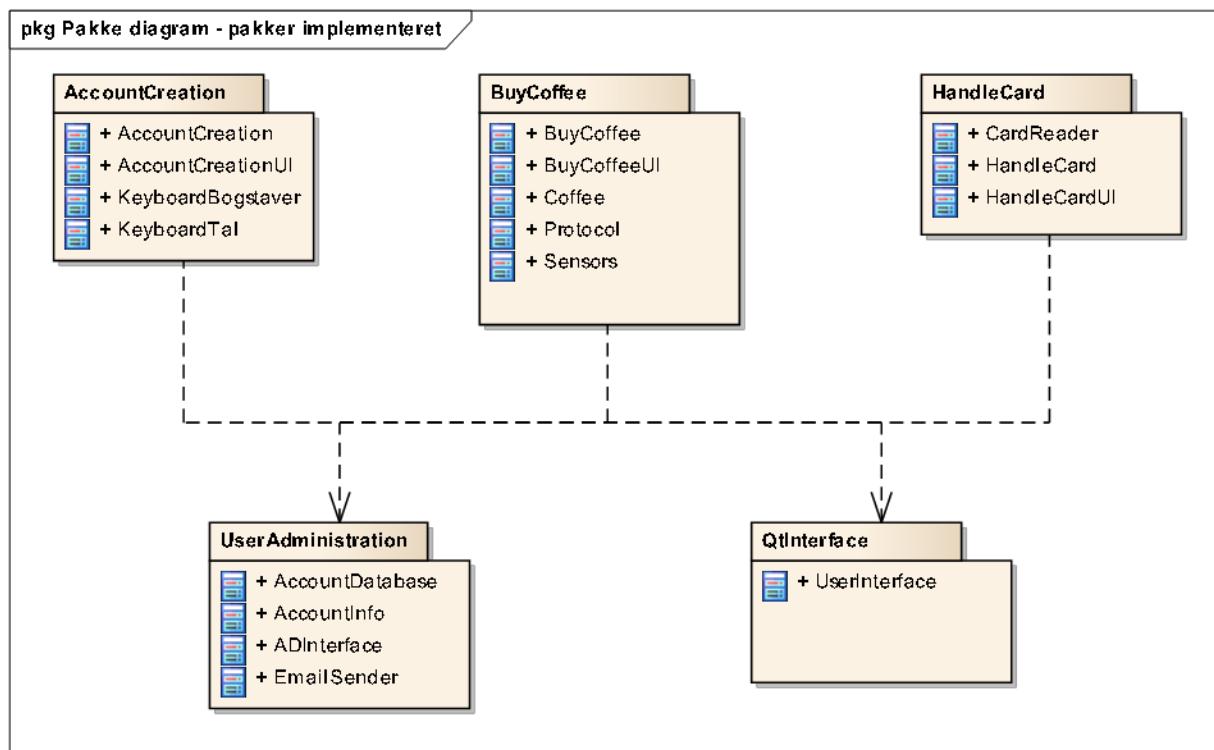
Når man forbinder tråde, forbinder man samtidig signalet med et slot. Slottet er den funktion, der bliver kaldt når signalet er emittet.

¹¹ Se <http://qt-project.org/wiki/>

3.3.3. Pakkediagram

Efter endt implementering er der lavet et nyt pakkediagram. Formålet med det er at kunne sammenligne med pakkediagrammet i designafsnittet, samt vise den endelige klasseinddeling i systemet. Antallet af pakker er ikke ændret, da der ikke er tilføjet use cases eller funktionalitet til systemet. Da systemet er implementeret i Qt har UserInterface derimod ændret navn til QtInterface. De nye klasser er tilført da Qt kræver en klasse per kontrol klasse til at varetage GUI. Klasserne tilføjes use case pakkerne, da de kun bruges i de tilhørende use cases.

UserInterface klassen bruges som hovedvinduet for systemet. Dvs. at når systemet starter er det UserInterface der bestemmer GUI. Når use casene begynder bliver den herefter erstattet af den korrekte UI klasse.



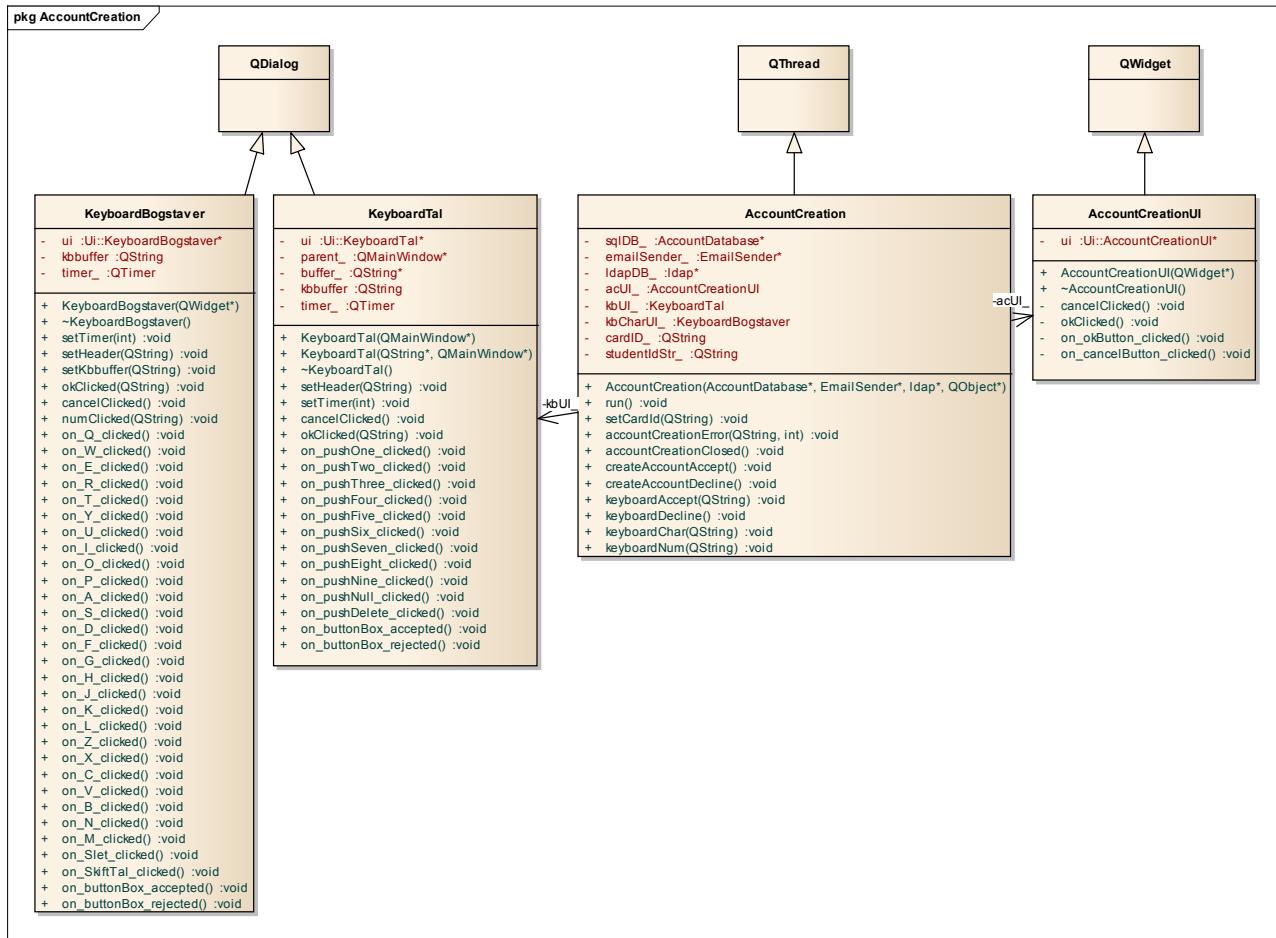
Figur 20 - Implementeret pakkediagram

3.3.4. Klassediagrammer

Ud fra implementeringen er der genereret illustrerende klassediagrammer. Diagrammerne er inddelt efter pakkerne i pakkediagrammet. Her vil hver implementeret klasse og diverse GUI skærme blive overordnet beskrevet. Hvis der ønskes yderligere indsigt i hvordan specifik kode er skrevet henvises der til sourcekoden¹².

Overordnet er brugergrænsefladen bygget op af 7 forskellige vinduer. Hvert vindue dækker over sin egen funktionalitet i sammenspil med resten af softwaren. De dokumenteres her hver for sig med nummererede inddelinger, for at få større overblik i beskrivelserne.

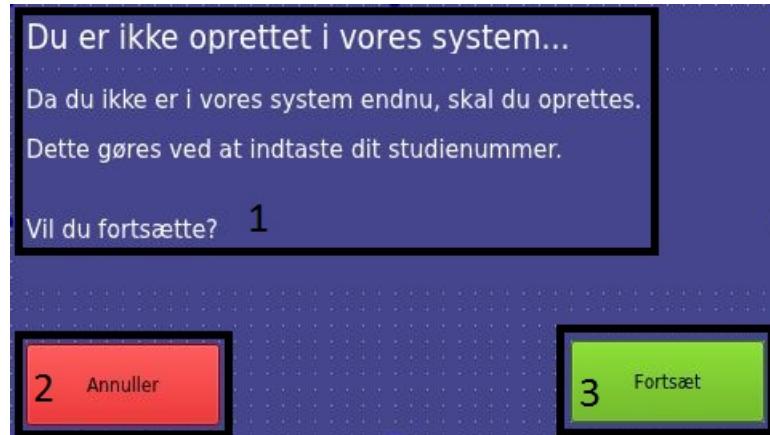
3.3.4.1. Klassediagram for AccountCreation:



AccountCreation er kontrol klassen for use case 1. Klassens primære formål er at håndtere den sekventielle gennemgang, som forekommer i use casen. Klassen er sin egen tråd og nedarver derfor fra QThread. Klassen fortolker input fra brugeren, som kommer via UserInterface. Dette gøres blandt andet ved hjælp af *keyboard*-metoderne, som findes under QDialog.

¹² Se Bilag 1.3

AccountCreationUI



Figur 21 - AccountCreationUI.ui

AccountCreationUI er et ganske simpelt vindue, hvis eneste funktionalitet er at underrette brugeren om at vedkommende ikke er oprettet i den interne database.

1: Besked til bruger.

2: Annuler-knappen emitter et signal, der hedder cancelClicked().

3: Fortsæt-knappen emitter et signal, der hedder okClicked().

KeyboardBogstaverUI



Figur 22 - KeyboardBogstaver.ui

Dette vindue bruges i forbindelse med oprettelse af brugere. Dog vises dette vindue kun, hvis en bruger har brug for at indtaste chars i sit brugernavn.

1: Header til vinduet.

2: Tekstfelt der opdateres løbende, mens brugeren indtaster. Tekstfeltet ændres ikke ved skift til tal. Der er desuden opsat OK/Cancel knapper til at fortsætte eller fortryde.

3: Diverse knapper til bogstaver.

4: Sletter sidstskrevne char/tal i tekstfeltet.

5: Skifter til KeyboardTal, så brugeren kan indtaste tal.

KeyboardTalUI



Figur 23 - KeyboardTal.ui

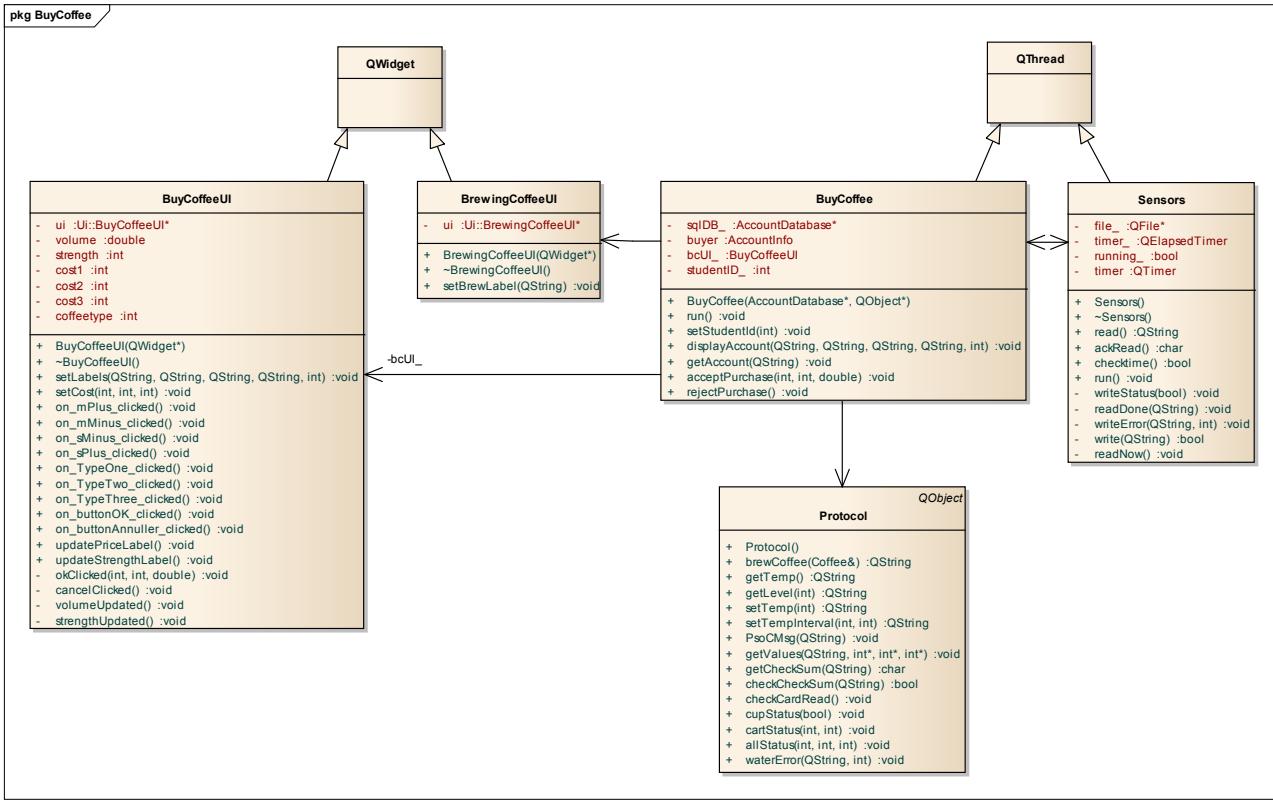
1: Header til vinduet.

2: Tekstfelt der opdateres løbende, mens brugeren indtaster. Tekstfeltet ændres ikke ved skift til bogstaver. Der er desuden opsat OK/Cancel knapper til at fortsætte eller fortryde.

4: Sletter sidstskrevne char/tal i tekstfeltet.

5: Skifter til KeyboardBogstaver, så brugeren kan indtaste bogstaver, skulle dette være nødvendigt.

3.3.4.2. Klassediagram for BuyCoffee:



BuyCoffee er kontrol klassen for use case 2. Klassens primære formål er at håndtere den sekventielle gennemgang, som forekommer i use casen. Klassen er sin egen tråd og nedarver derfor fra QThread. Klassen håndterer input fra brugeren gennem brugergrænsefladen, som bliver videresendt ved hjælp af UserInterface klassen.

Sensors er grænsefladen mellem PSoC og DevKit, og nedarver fra QThread da det er en tråd. Det er nødvendigt at gøre klassen til en tråd fordi det er vigtigt at få alt information med fra SPI interfacet. Klassen er indrettet sådan at den tjekker om der er modtaget noget en gang i sekundet. Desuden sørger denne for at få fejl fra resten af systemet og viderestille disse.

Protocol håndterer den for projektet definerede protokol¹³. Når en funktion kaldes fortolker Protocol beskeden. Beskeden kan både indeholde et objekt og standard datatyper. Disse parametre formateres således at beskeden bliver formateret til et char array, som indeholder de tegn som protokollen beskriver.

BrewingCoffeeUI er skærmen, som viser at brygningen er i gang. Denne beskrives ikke nærmere, da der blot står "Brygning i gang" på skærmen.

¹³ Se afsnit 4.7

BuyCoffeeUI

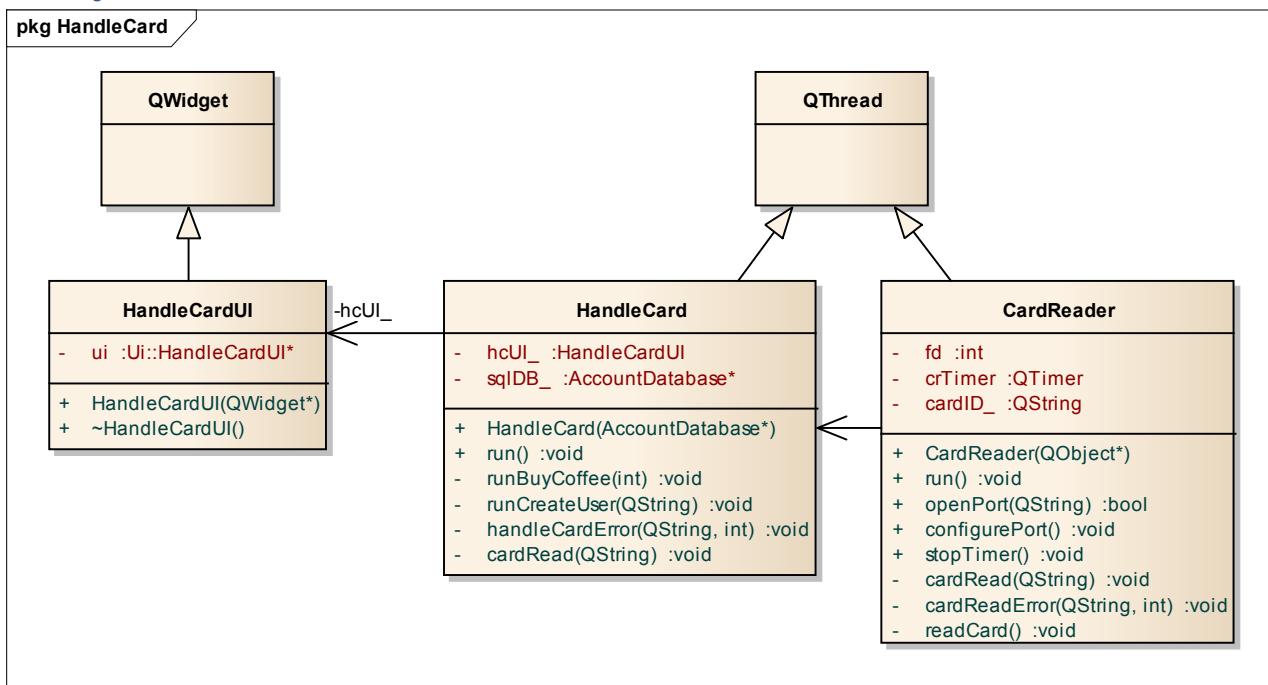


Figur 24 - BuyCoffeeUI.ui

BuyCoffeeUI er det vindue der vises når login er en success. Dette vindue kan derfor først vises, hvis en bruger har kørt sit kort forbi kortscanneren og allerede er oprettet i databasen. Vinduet er hovedvinduet til køb af kaffe. Det er i dette vindue der vælges kaffe, mængde og styrke. Der vises desuden også navn og saldo, så brugeren kan tjekke dette ved maskinen i stedet for at være tvunget til at logge på webfronten.

- 1: En label der opdateres med brugerens navn fra AD databasen.
- 2: Disse 3 knapper bruges til at vælge mellem de forskellige typer kaffe der er i maskinen. Når én knap vælges, sættes de andre til Checked(false), så der tydeligt kan ses hvilken knap der er valgt. Navnene på knapperne hentes fra konteringsdatabasen så de stemmer overens med de kaffetyper der er i de respektive beholdere. Ved valg af type opdateres den totale pris, der er udregnet i pris/enhed(dl).
- 3: Annuler knappen har kun én funktionalitet. Når der trykkes på knappen emitterer signalet *cancelClicked*. OK-knappen emitter signalet *okClicked* og tager 3 parametre med: Kaffetype, styrke og volume.
- 4: "Mængde"-label er den label, der opdateres med steps af 1 dl pr step. Steps foretages ved brug af + og - knapperne.
- 5: "Styrke"-label opdateres med de 3 steps: mild, normal og stærk. Steps foretages ved brug af + og - knapperne.
- 6: Dette grid bruges til at underrette brugeren om saldo og nuværende pris. "Saldo"-labelen opdateres via den interne database. Dette er brugerens nuværende saldo. "Pris"-label opdateres løbende undervejs, da dette afhænger af typevalg og mængde.

3.3.4.3. Klassediagram for HandleCard:



HandleCard er kontrolklassen for sekvensen, startende fra brugeren kører sit kort forbi kortlæseren til en use case startes. Når et kort er læst bliver HandleCard notificeret, hvorefter den tjekker om brugeren som er knyttet til kortet, findes i konteringsdatabasen via *cardRead*. Hvis det er tilfældet køres *runBuyCoffee* og tilsvarende hvis kortet ikke genkendes, køres *runCreateUser*. Desuden er der metoder til fejlhåndtering.

CardReader er grænsefladeklassen til systemets kortlæser. Klassen bruger en timer til at udløse en blocking read. På Linux systemer kan hardware output, gennem en driver, læses i filer. Derfor bruges streams til at læse værdien fra kortlæseren. Desuden er der konfigurationsmetoder til opsætning.

HandleCardUI



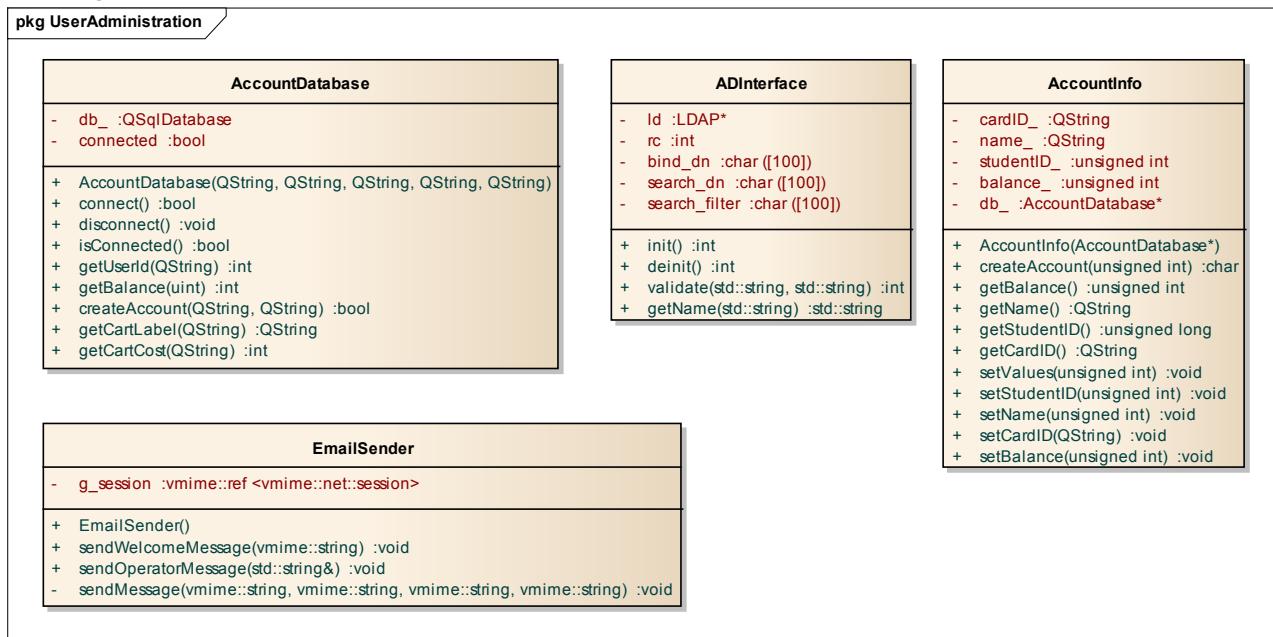
Figur 25 - HandleCardUI.ui

HandleCardUI er systemets venteskærm. Det er det ui som vises når maskinen afventer et kort id i forbindelse med en læsning.

1: Besked til brugeren.

2: Der vises et billede af et kort der køres forbi en kortscanner.

3.3.4.4. Klassediagram for UserAdministration:



Klassen **AccountDatabase** fungerer som grænseflade til systemets konteringsdatabase. Konteringsdatabasen indeholder brugerinformation, information om transaktioner, log og de forskellige pulvertyper i maskinen. For at kunne tilgå konteringsdatabasen er det nødvendigt at implementere en klasse som kan udlæse, sætte og ændre de relevante værdier i databasen. Klassen er implementeret ud fra systemarkitekturen og benyttes i alle use cases. Metoderne er defineret ud fra sekvensdiagrammet som findes i systemarkitekturen.

Metoderne kan overordnet deles ind i tre hovedgrupper:

- **Get-metoder** håndterer udlæsninger fra databasen, og formaterer disse til brug i programmet for systemet. Eksempler på get-metoder kan for eksempel være *getName* og *getBalance*, der læser brugers navn og tilgængelige saldo.
- **Update-metoder** håndterer opdateringer i database, og benyttes for eksempel når en ny bruger skal oprettes. Eksempler på update-metoder kan være *createUser*, som opretter en ny bruger i databasen og *updateTransactions*, som laver en ny post i databasen hver gang et køb er blevet foretaget.
- **Utility-metoder** håndterer funktioner som for eksempel oprettelse af forbindelse til databasen.

Klassens funktionalitet er implementeret med Qt-frameworkets egen SQL connector, som virker uanset hvilken type SQL database der er tale om. Det vil sige, at systemet uden videre ændringer kan køre på med en Sqlite database, og dette gør mulighederne for videre udvikling større.

Formålet med klassen **EmailSender** er at håndtere og sende emails til nye brugere og operatør. Brugeren modtager en email med informationer angående optankning af penge på sin konto når han er oprettet i systemet.

Operatøren modtager en email når der sker hændelser, der kræver at en operatør tilsør maskinen. Klassen er designet ud fra softwarearkitekturen, og er en del af hovedscenariet for use case 1 og 4¹⁴.

Klassen fungerer ved at de to *sender*-metoder formaterer teksten til den mail vi gerne vil sende. Hver *sender* metode benytter sig så af den private metode *sendMessage* til at sende mailen via en SMTP server.

ADInterface klassens funktionalitet er implementeret med C++ biblioteket VMIME, som er bygget til at formatere MIME emails, og sende disse via SMTP eller en anden mailprotocol. Dette bibliotek blev valgt til implementeringen af klassen, da det på en simpel måde tillod at HTML formattere emails direkte fra C++ koden. For at DevKitet kan kommunikere med, og validere på, AD databasen er der brugt et open-source biblioteket kaldet OpenLDAP. OpenLDAP er et C bibliotek og for at kunne bruge dette på en objektorienteret måde, har det været nødvendigt at pakke det ind i en C++ klasse. Denne klasse er implementeret i ADInterface.cpp hvor der er inkluderet den eksterne C headerfil ldap.h.

Kravet til klassen er begrænset og det er derfor muligt at nøjes med fire forskellige funktioner:

- *Init*

Initialiserer forbindelsen til AD serveren

- *Deinit*

Nedlægger forbindelsen til AD serveren

- *Validate*

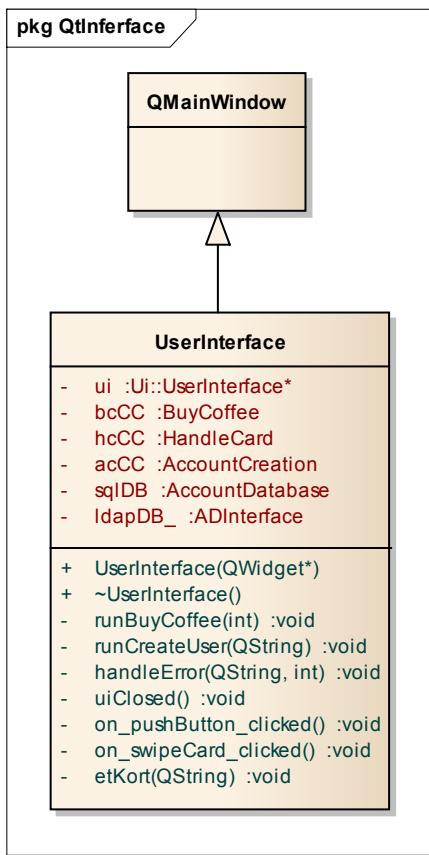
Validerer en brugers oplysninger

- *GetName*

Returnerer det fulde navn, som er forbundet med et bruger id

¹⁴ Se afsnit 1.3

3.3.4.5. Klassediagram for QtInterface:



Formålet med klassen **UserInterface** er at samle hele brugergrænsefladen og fungerer på den måde som systemets egentlige main-klasse. Da designfasen var i gang blev det besluttet at det var nødvendigt at have et centralt vindue til styring af de resterende vinduer, samt styring af eventuelle fejl. Ved hjælp af sekvensdiagrammer og klassediagrammer blev UserInterface klassen oprettet, samt de tilhørende metoder.

Klassen består af 3 filer:

- UserInterface.ui
- UserInterface.cpp
- Userinterface.h

Ui filen repræsenterer et vindue, der består af én knap og en label. Knappen bruges når operatøren er tilkaldt og skal logge ind for at godkende reparation eller opfyldning. Labelen bruges som information til brugeren, hvis der skulle opstå eventuelle fejl.

Klassens funktionalitet er udelukkende bindeled mellem de andre klasser i systemet, både i form af kode, men også vinduer. Som det ses i source koden¹⁵, består koden udelukkende af `connect()` kald samt implementering af de metoder der er listet i diagrammet

UserInterface. Klassen sørger også for at oprette kobling

rette ob-

kobling

jemter af de andre vinduer, så de kan kaldes igennem UserInterface og skabe større funktionaliteten.

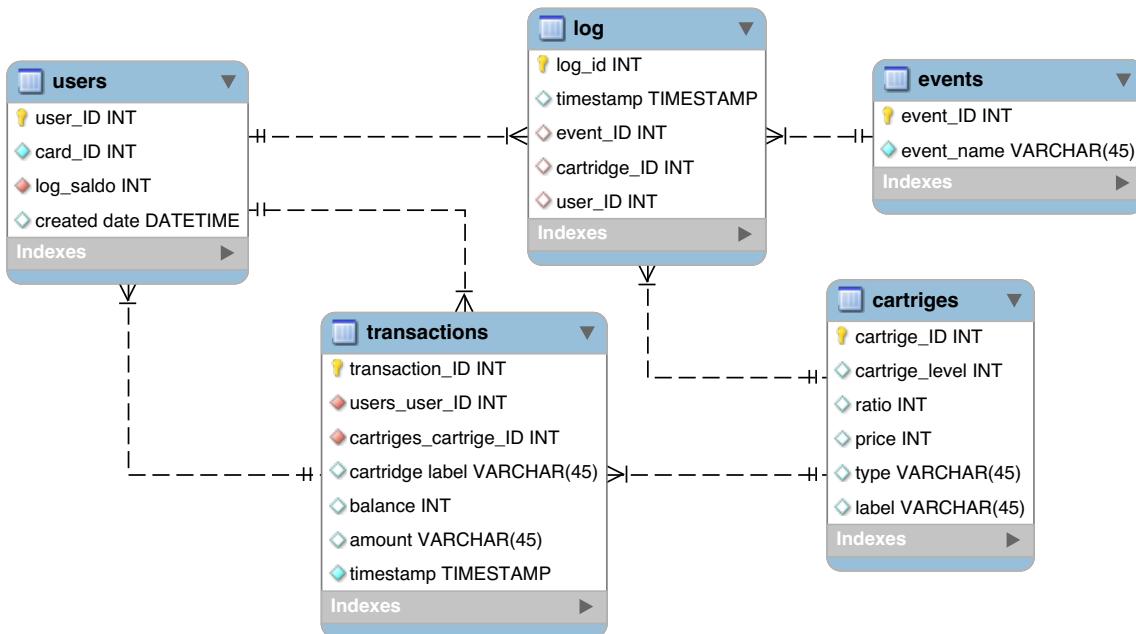
¹⁵ Se Bilag 1.3

4. Design og implementering af databaser og webfront

4.1.1. Konteringsdatabase

Konteringsdatabasen er en database til opbevaring af brugerinformationer samt informationer omkring brugernes transaktioner og maskinens konfiguration. Databasen skal kunne tilgås af både kaffeautomaten og webfronten, og dermed er der brug for en databasetype der er bredt understøttet. MySQL er databaseformatet der er valgt, og valget bunder i at denne har alle de kvaliteter systembeskrivelsen og kravspecifikationen foreskriver. Desuden er softwaren til etablering af en sådan server gratis, veldigumenteret og nem at gå til.

Designet af databasen er lavet ud fra en analyse af systembeskrivelsen og kravspecifikationen, og er illustreret i nedenstående diagram.



Figur 26 - Diagram over konteringsdatabase

Som det fremgår af diagrammet kender tabellerne til hinanden. Dette gøres for at sikre at data ikke bliver dubleret, eller ført forkert ind i tabellerne.

En kort beskrivelse af tabellerne i databasen følger

users indeholder burgernavn og kordnummer på de i systemet oprettede brugere.

cartridges indeholder informationer om de forskellige kaffebeholder monteret i systemet. Det er herfra brugerinterfacet henter informationer så som pris og blandingsforhold på kaffe og labels til knapper.

transactions indeholder informationer om transaktioner foretaget på kaffeautomaten. Transactions deler felterne **user_user_ID** og **cartridges_cartridge_ID** med tabellerne **users** og **cartridges**.

events indeholder et event ID og et eventnavn for hver logevent. Dette gøres for at sikre et ensartet log-system

log indeholder systemloggen. Det er her information om uventede hændelser i systemet bliver gemt. Log deler felterne **event_ID**, **cartridge_ID**, og **user_ID** med tabellerne **events**, **cartridges** og **users**

Databasen er etableret på en Ubunut server hvor software bundlen LAMP¹⁶ er installeret. Denne også på denne server vores webfront ligger.

4.2. Webfront

Til implementering af webfronten er der valgt at bygge på et PHP framework kaldet "YII".

YII er valgt da det er et nyt og moderne framework, bygget op om MVC(Model View Controller) tankegangen, der gør at det er agilt og let at ændre til sine egne behov. Samtidigt havde YII allerede indbygget funktionalitet for mange af de ting som er nødvendigt i webfronten og arbejdet kunne derfor fokuseres på hvad der var specielt ved dette projekt.

Det er hovedsageligt den tekniske implementering af webfronten, samt hvilke informationer der formidles, der fokuseres på og derfor er der ikke gjort noget specielt ud af det grafiske indtryk.

4.3. AD Database

I de fleste PHP installationer er der biblioteker der gør det muligt at forbinde til, og arbejde med, en LDAP server (Som Microsoft AD).

YII's login og brugerstyring er derfor overloadet med en funktion, som i stedet for at holde bruger og kode op mod en lokal databasetabel, beder skolens ADdatabase om at validere legitimationsoplysningerne.

Hvis disse kan godkendes hentes brugerens fulde navn også fra LDAP serveren.

I TABEL 1 er det kort illustreret hvordan den overloadede funktionalitet til LDAP validering er implementeret i UserIdentity.php.

Tabel 1 - Kode fra UserIdentity.php

```
<?php
class UserIdentity extends CUserIdentity
```

¹⁶ Står for Linux Apache MySQL PHP

```
{  
...  
public function authenticate()  
{  
...  
$connection = ldap_connect($options['host']);  
...  
if($connection)  
{  
...  
$user = User::model()->find('LOWER(user_ID)=?',array(strtolower($this->username)));  
  
if($user === null)  
    $this->errorCode = self::ERROR_USERNAME_INVALID;  
else if(!$bind)  
    $this->errorCode = self::ERROR_PASSWORD_INVALID;  
else  
{  
    $this->_id = $user->user_ID;  
    $this->errorCode = self::ERROR_NONE;  
}  
}  
...  
ldap_unbind($connection);
```

```
    return !$this->errorCode;  
}  
  
...  
}
```

Når sessionen er verificeret på LDAP, bliver der sat en cookie i brugerens browser, der gør at han automatisk er logget ind næste gang han besøger siden.

4.3.1.1. Highcharts

Til visuelt at formidle informationer om kontobevægelser på en intuitiv måde, er graf-biblioteket "Highcharts" brugt.

Highcharts er hurtigt, let og avanceret på samme tid og har mange indstillingsmuligheder, der gør det let at tilpasse til projektets behov.

For at integrere det med YII er der brugt en open-source indpakning kaldet "ActiveHighcharts". Ved at bruge denne YII-pakke gør, at kan man sende en instans af YII's datastruktur, " CActiveDataProvider", videre direkte fra en databaseforespørgsel.

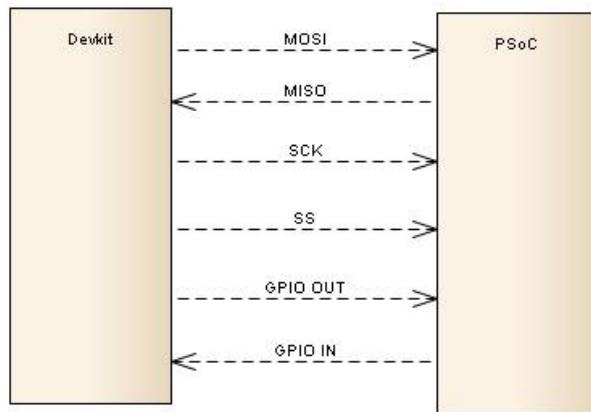
4.3.2. Database

I YII arbejder man som udgangspunkt med databaser ved at pakke databasetabeller ind i klasser nedarvet fra " CActiveRecord". Man kan herefter definere funktioner og funktionalitet på denne klasse.

Dette kan for eksempel ses i frontend/protected/models/Transactions.php

4.4. Design og implementering af Kommunikation mellem DevKit og PSOC

Kommunikation mellem DevKitet og PSoC'en er implementeret ved brug af SPI standarden¹⁷. DevKitet og PSoC'en er forbundet med en MOSI, en MISO, en SS og en Sclk linje. Dertil er de to platforme forbundet med to GPIO linjer, så begge kan tage initiativ til at kommunikere. DevKitet fungerer som master, og PSoC'en som slave.



Figur 27 – SPI forbindelser

SPI kommunikationen er opsat til at:

- Sende MSB first
- Sende 8 bit/word
- Kører SPI mode 3 (CPOL=1 , CPHA=1)
- Sende med 1 MHz

Når DevKitet ønsker at sende en besked til PSoC'en sættes GPIO-out høj, hvilket genererer et interrupt på PSoC'en. Når PSoC'en ønsker at sende en besked til DevKitet sætter den GPIOin høj. Oprindeligt var SPI funktionaliteten på DevKitet implementeret som et char device med en blocking read funktion. Dette gav problemer, da user space applikationen på DevKitet kunne sidde fast i read funktionen, hvis en besked blev sendt forkert. Derfor er char devicet implementeret så read funktionen verificerer at PSoC'en har data til at blive sendt, ved at tjekke om GPIO-in er sat høj.

¹⁷ Se http://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus

4.4.1.1. Protokol

Til dette projekt er der udarbejdet en protokol med det formål at fastsætte kommunikationen mellem DevKitet og PSoC'en.

4.4.1.2. Beskedtyper

Nedenfor ses de beskeder der kan sendes mellem DevKitet og PSoC'en

DevKit til PSoC

Besked	Returværdi	Kommentar
MKDRINK:[type],[vand],[pulver]	-	DevKitet kan sende brugerens bestillinger til PSoC'en
GLEVEL:[type]	Char niveau	Niveauet returneres i den efterspurgte kaffebeholder.

PSoC til DevKit

Besked	Returværdi	Kommentar
KOPSTATUS:[status]	-	Hvis status er 1 er der en kop på spildbakken, hvis status er 0 er der ingen kop.
STATUS:[type0], [type1], [type2]	-	En status besked der sendes efter hver succesfulde kaffebrygning. Returnere niveauet i alle kaffebeholdere.

Når DevKitet sender beseden MKDRINK:<type>,<vand>,<pulver>, sendes <type> mellem 0 og 2, <vand> sendes i antal halve dl, og <pulver> i ml. Når PSoC'en sender et pulverniveau i en af kaffebeholderne, gøres dette i procent.

Der regnes kun i hele tal, da det ikke ønskes at PSoC'en skal arbejde med decimaltal.

Specielle værdier

Efter hver besked sendes en stopbit. Dette giver mulighed for at sende beskeder af forskellige længder mellem de to platforme. Til at validere at den sendte besked er sendt korrekt, er der indført en checksum. Checksummen beregnes som summen af alle værdierne i strengen modulus 127.

Dette medfører at en besked ser ud som følger:

<Besked>,<Checksum><Stopbit>

Når DevKitet sender en streng til PSoC'en returnerer PSoC'en et ACK, hvis den udregnede checksum stemmer overens med den sendte. Ellers returnerer PSoC'en et NAK. DevKitet returnerer ikke ACK og NAK til PSoC'en, men ser bort fra den modtagede streng, hvis checksummen ikke stemmer overens med den sendte. Hvis PSoC'en ønsker at sende fejlmeddelelser, på manglende kop eller vand, sendes strenge med et interval, til den pågældende fejl er løst. Dette sikrer at DevKitet modtager vigtige fejlbeskeder, da PSoC'en sender beskeden flere gange.

Beskrivelse	værdi
Stopbit	*
ACK	S
NAK	F
Checksum	

4.4.2. SPI på PSoC

SPI kommunikationen er styret af interrupts. Hvis PSoC'en er i staten "Idle" eller "Modtag Data" er SPI interrupt aktiveret. Hver gang DevKitet ønsker at sende en besked sættes GPIO-out høj på DevKitet. Dette triggerer en ISR rutine, der læser en char fra Rx bufferen og gemmer den i et bufferarray. Når PSoC'en har modtaget en stopbit, deaktiveres SPI interruptet.

Efter at der er modtaget en stopbit, begynder PSoC'en valideringen og behandlingen af den modtagede besked.

Når PSoC'en ønsker at kommunikere med DevKitet, sendes beskeden fra en senderbuffer, en char ad gangen. Dette gøres ved at skrive en char til PSoC'ens Tx buffer. Efter dette er gjort sætter PSoC'en GPIOin høj, og afventer en Clock og SS fra DevKitet.

Når PSoC'en enten har modtaget eller sendt en char, cleares PSoC'ens Rx og Tx buffer, da dette sikrer at der ikke befinner sig data i PSoC'ens Rx og Tx buffer under næste dataudveksling. Det har under udviklingen af den nødvendige kode til SPI kommunikationen vist sig nødvendigt, da der efter første dataudveksling ofte blev sendt korrupte beskeder.

4.4.3. DevKit spi driver

Kommunikationen mellem DevKitet og PSoC'en er implementeret med en user space applikation, et character device og en hardware driver. User space applikationen indeholder nødvendige klasser til beregning af checksum samt kontrol af beskeder. Applikationen benytter character devicet som adgang til SPI linjerne. Dette er en fornuftig løsning, da der ikke inføres policy i kernen.

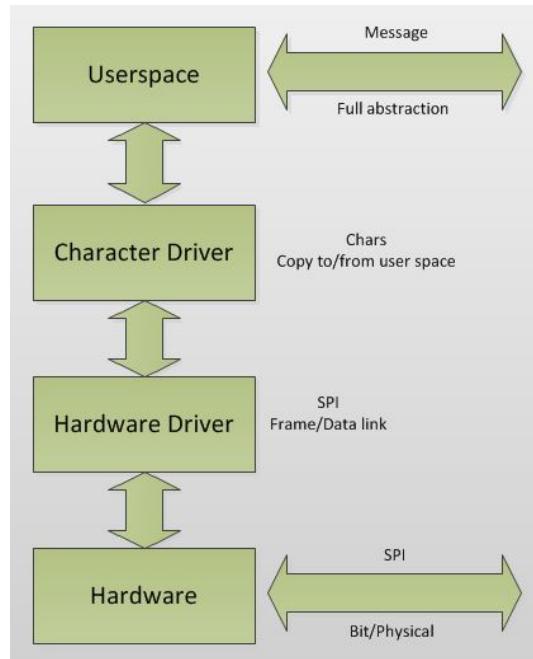
Det udviklede character device, har til opgave at modtage og sende data mellem user space applikationen og hardware driveren. Dertil er det i dette lag at de to GPIO linjer mellem DevKitet og PSoC'en håndteres. Hardware driveren har til opgave at binde character devicet med specifik hardware, der skal sende og modtage beskeder på SPI linjerne. Her sættes de fysiske parameter som hastighed, samt CPHA og CPOL modes.

De forskellige lag er illustreret på figur 24.

Når der skal sendes beskeder fra DevKitet til PSoC'en sætter character devicet GPIO-out højt, hvilket triggerer et interrupt på PSoC'en. Når DevKitet skal læse fra PSoC'en validerer den om GPIO-in er sat høj. Hvis dette er tilfælde læses der fra SPI linjen. Hvis GPIO-in er lav sendes en nul terminering til user space applikationen, som dermed ved at der ikke er data klar til læsning.

For at kunne trække SPI linjerne fra OMAP processoren til eksterne GPIO ben på add-on boardet skal CPLD'en på DevKitet konfigureres. Dette gøres i et boot-shell script¹⁸. Under udvikling af driveren er der udarbejdet simple, specifikke test applikationer til test af funktionalitet. Følgende test-applikationer er udviklet:

- gpio-test – sender interrupt til PSoC
- irq-test – modtager interrupt fra PSoC
- psoc-test – userspace applikation til test af samlede funktionalitet.



Figur 24 Illustration af SPI driverens opbygning

¹⁸ Se bilag 1.4

4.4.4. Pin konfiguration

DevKit800 SPI connection

DevKit800	Connector
MOSI	J9 Pin 1
MISO	J9 Pin 2
CLK	J9 Pin 5
SC	J9 Pin 3
PSoC interrupt in	J5 Pin 8 (GPIO130)
PSoC interrupt out	J5 Pin 7 (GPIO131)
GND	J9 Pin 7/J5 Pin 9

PSoC SPI connection

PSoC	Connector
MOSI	P6[0]
MISO	P6[1]
SCLK	P6[3]
SS	P6[2]
Interrupt in	P6[4]
Interrupt out	P6[5]

4.5. Booting DevKit

Efter opstart af MokkaMonster skal konfiguration af aktive hardware og start af QT-Applikationen ske automatisk. Dette gøres i init-systemet Systemd. Systemd er en system deamon for Linux systemer, som giver mulighed for on-demand startning af applikationer, mount point mm.

Et shell-scriptet (MokkaBoot.sh) startes med systemd. Scriptet konfigurerer cpld'en på DevKit til spi-kommunikation, initialiserer spi-driveren samt konfigurerer tty. Systemd servicen "mokkaboot.service" starter så scriptet. En anden systemd service starter efterfølgende QT-Applikationen. Kommandoen "Systemctl enable <unit>" bruges til at linke servicen til systemd deamon'en. Herefter startes applikationen automatisk efter hver boot¹⁹. Endvidere er boot-logoet på DevKit'et udskiftet til et mere sigende



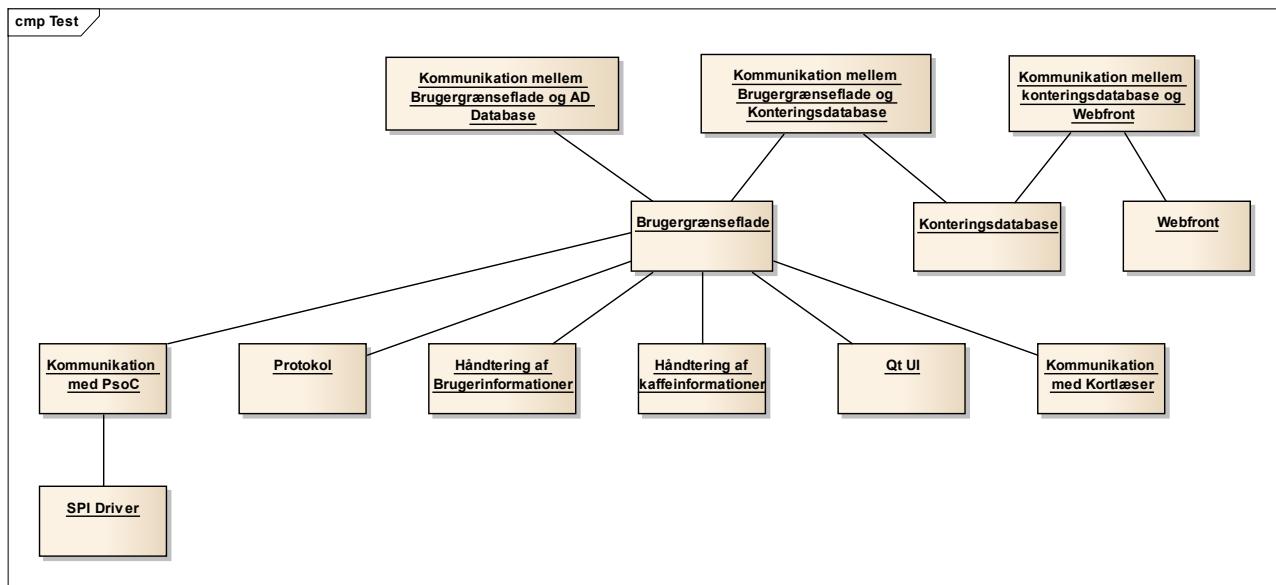
Figur 28 Bootlogo på DevKit

¹⁹ Se Bilag 1.4

4.6. Softwaretest

Som beskrevet i afsnittet om projektgennemførsel, bruges V-modellen til at teste systemet. I dette afsnit beskrives hvordan den gør sig gældende i test af softwaren. Der vil være fokus på enheds- og integrations-testene.

Da softwaren er delt op i de tre overordnede blokke, blev de tilhørende test også opdelt sådan. Derudover blev brugergrænsefladen yderligere delt op i enheder der kunne testes separat. Opdelingen vises via dette diagram.



Figur 29 – Testhierarki

Blokkene er delt op efter funktionalitet, hvor der er oprettet minimum en klasse per funktion. Bottom-up strategien (Biering-Sørensen, 1988, s.183) blev brugt, hvor der opsættes et hierarki for systemet, og man tester nede fra og op. En øvre blok kan derfor ikke testes før alle blokkene under den er testet. Til denne strategi bruges test moduler, der agerer som dele af systemet uden om en specifik blok der skal testes. På den måde kan hver bloks integration i systemet testes separat. En yderligere beskrivelse af testen følger.

4.6.1. Testbeskrivelser

Håndtering af Brugerinformationer

Test	Håndtering af Brugerinformationer
Klasse	Accountinfo
Testbeskrivelse	Blokken testes med et testmodul, der agerer som kontrolklassen. Testmodulet skriver resultatet ud på skærmen så testen kan valideres. Klassen skal kunne modtage og levere informationer om den aktive bruger af systemet. Informationerne inkluderer navn, studienummer og balance.
Input	Sætter værdier med funktionen <code>setValues()</code> . Kalder derefter <code>getValues()</code> .
Output	Der modtages samme værdier som blev sat med <code>setValues()</code> .
Resultat	Ok

Håndtering af kaffeinformationer:

Test	Håndtering af kaffeinformationer
Klasse	Coffee
Testbeskrivelse	Blokken testes med et testmodul, der agerer som kontrolklassen. Testmodulet skriver resultatet ud på skærmen så testen kan valideres. Blokken skal kunne modtage og levere informationer om den valgte kaffe. Informationerne inkluderer type, mængde, styrke og pris.
Input	Der skrives værdier med funktionen <code>setCoffee(1,2,3)</code> . Derefter udskrives type, amount og strength med <code>getType()</code> , <code>getAmount()</code> og <code>getStrength()</code> .
Forventet output	Type = 1, Amount = 2, Strength = 3.
Output	Type = 1, Amount = 2, Strength = 3.
Resultat	Ok

Protokol:

Test	Protokol #1
Klasse	Protokol
Testbeskrivelse	Blokken testes med et testmodul der agerer som kontrolklassen. Testmodulet skriver ud på skærmen så testen kan valideres. Blokken skal modtage funktionskald der opretter beskeder af typen QString med informationer og udregne beskedens checksum i henhold til afsnit om protokol.
Input	Funktionen <i>brewCoffe(cup)</i> kaldes, hvor cup er en instans af Coffee med parametre: Type=1, Amount=2, Strength=3. Checksummen udregnes til "j".
Forventet output	Funktionene returnerer "MKDRINK:1,2,3,j*"
Output	Funktionene returnerer "MKDRINK:1,2,3,j*"
Resultat	Ok

Test	Protokol #2
Klasse	Protokol
Testbeskrivelse	Blokken testes med et testmodul, der agerer som kontrolklassen. Testmodulet skriver ud på skærmen så testen kan valideres. Blokken skal modtage, regne checksum på, og validere en QString besked. Blokken skal derefter kalde den korrekte funktion på testmodulet.
Input	Funktionen <i>PsoCMsg("STATUS:52,32,1,&")</i> & er udregnet som den rigtige checksum.
Forventet output	Funktionen, status, kaldes på testmodulet med parametrene, 52,32 og 1.
Output	Funktionen, status, kaldes på testmodulet med parametrene, 52,32 og 1.
Resultat	Ok

SPI Driver:

Test	SPI Driver #1
-------------	---------------

Klasse	N/A
Testbeskrivelse	Driveren testes med PSoC-test applikationen. Testen kræver at DevKitet skal være forbundet til en PsoC, der kan modtage og sende beskeder over SPI. Endvidere skal driveren være indsat i kernen. Test applikationen benytter så driveren til at sende en fuld besked.
Input	Applikationen PSoC-test startes
Forventet out-put	Programmet tilsluttet PsoCen modtager beskeden MKDRINK:1,2,3,j*.
Output	Programmet tilsluttet PsoCen modtager beskeden MKDRINK:1,2,3,j*.
Resultat	Ok

Kommunikation med PsoC:

Test	Kommunikation med PsoC #1
Klasse	Sensors
Testbeskrivelse	Blokken testes med et testmodul, der agerer som kontrolklassen. Blokken kræver at SPI driver er testet og godkendt. Derudover skal DevKitet være forbundet til en PsoC, der kan modtage og sende beskeder over SPI. Blokken skal modtage QStringLists der er lavet af protokollen og sende beskederne over SPI. Derudover skal den opfange beskeder fra SPI og sende til Protokol. Protokollen verificerer derefter via checksum om beskeden er modtaget korrekt.
Input	Funktionen <code>send("MKDRINK:1,2,3,j")</code> kaldes.
Forventet out-put	Programmet tilsluttet PsoCen modtager beskeden MKDRINK:1,2,3,j*.
Output	Programmet tilsluttet PsoCen modtager beskeden MKDRINK:1,2,3,j*.
Resultat	Ok

Test	Kommunikation med PsoC #2
Klasse	Sensors
Testbeskrivelse	Blokken testes med et testmodul der agerer som kontrolklassen. Blokken kræver at SPI driver er testet og godkendt. Derudover skal DevKitet være forbundet til en PsoC der kan modtage og sende beskeder over SPI. Blokken skal opfange beskeder fra SPI og sende til testmodulet, der kontrollerer checksummen og skriver beskeden ud hvis den er korrekt. Ellers skriver den "Checksum fejl".
Input	Programmet på PsoC sender beskeden, "STATUS:52,32,1,&*", over SPI.
Forventet output	Testmodulet skriver "STATUS:52,32,1,&*" ud.
Output	Testmodulet skriver "STATUS:52,32,1,&*" ud.
Resultat	Ok

Kommunikation mellem Brugergrænsefladen og AD databasen

Test	Validering med AD databasen
Klasse	?
Testbeskrivelse	LDAP biblioteket og ADInterface's "validate()" funktion testes ved at kalde den med brugeroplysninger fra den kompiledé ADInterface.cpp
Input	Kontooplysninger på en AD bruger i Stud eller Staff
Forventet output	Validering lykkedes
Output	Validering lykkedes
Resultat	Ok

Test	Hente brugeroplysninger på AD databasen
Klasse	?
Testbeskrivelse	LDAP biblioteket og ADInterface's "getName()" funktion testes ved at kal-

	de den med en AD identitet fra den compilede ADInterface.cpp
Input	Kontooplysninger på en AD bruger i Stud eller Staff
Forventet output	Brugernavn
Output	Brugernavn

Kommunikation mellem Brugergrænsefalte og Konteringsdatabase:

Test	Håndtering af kaffe-labels i BuyCoffee UI
Klasse	BuyCoffeeUI
Testbeskrivelse	Der er implementeret 3 knapper til hver type kaffe der findes i systemet. Der er yderligere implementeret en label til prisen pr enhed af kaffe. Når programmet starter skal systemet hente informationer fra databasen, så de implementerede labels stemmer overens med databasen.
Input	I databasen oprettes 3 typer kaffe, samt en pris pr enhed for hver type kaffe.
Forventet output	Labels på GUI stemmer overens med databasen.
Output	Labels på GUI stemmer overens med databasen.
Resultat	Ok

Test	Håndtering af brugerinformation i BuyCoffee UI
Klasse	BuyCoffeeUI
Testbeskrivelse	Der er implementeret en label til brugerens navn. Der er yderligere implementeret en label til brugerens nuværende saldo. Når programmet starter skal systemet hente informationer fra databasen, så de implementerede labels stemmer overens med databasen
Input	I databasen oprettes 1 bruger, samt en saldo for den pågældende bruger.
Forventet output	Labels på GUI stemmer overens med databasen.

Output	Labels på GUI stemmer overens med databasen.
Resultat	Ok

Webfront:

Test	Validering af brugeroplysninger
Klasse	?
Testbeskrivelse	PHP's LDAP funktion og de to bind-strenge (for Stud og Staff) testes ved at isolere funktionerne i et PHP script, og klade med brugeroplysninger fra en browser.
Input	Kontooplysninger på en AD bruger i Stud eller Staff
Forventet output	Validering lykkedes
Output	Validering lykkedes
Resultat	Ok

Kommunikation mellem Webfront og AD databasen

Test	Validering af brugeroplysninger
Klasse	?
Testbeskrivelse	LDAP biblioteket og IdapConnector's "getName()" funktion testes ved at kalde den med en AD identitet fra den compilede IldapClient.cpp
Input	Kontooplysninger på en AD bruger i Stud eller Staff
Forventet output	Brugernavn
Output	Brugernavn
Resultat	Ok

Test	Hente brugeroplysninger på AD databasen
Klasse	?
Testbeskrivelse	PHP's LDAP funktion og de to bind-strenge (for Stud og Staff) testes ved at isolere funktionerne i et PHP script, og klade med brugeroplysninger fra en browser.
Input	Kontooplysninger på en AD bruger i Stud eller Staff
Forventet output	Brugernavn
Output	Brugernavn
Resultat	Ok

5. Hardware dokumentation

Dette dokument beskriver de forskellige hardware blokkes design og virkemåde. Nedenfor ses et bdd for de samlede hardware blokke. Dette bdd har været grundlaget for udvikling af diverse komponenter/blokke. Hver blok er brudt yderligere ned og/eller fået udarbejdet ibd i det omfang det synes nødvendigt.

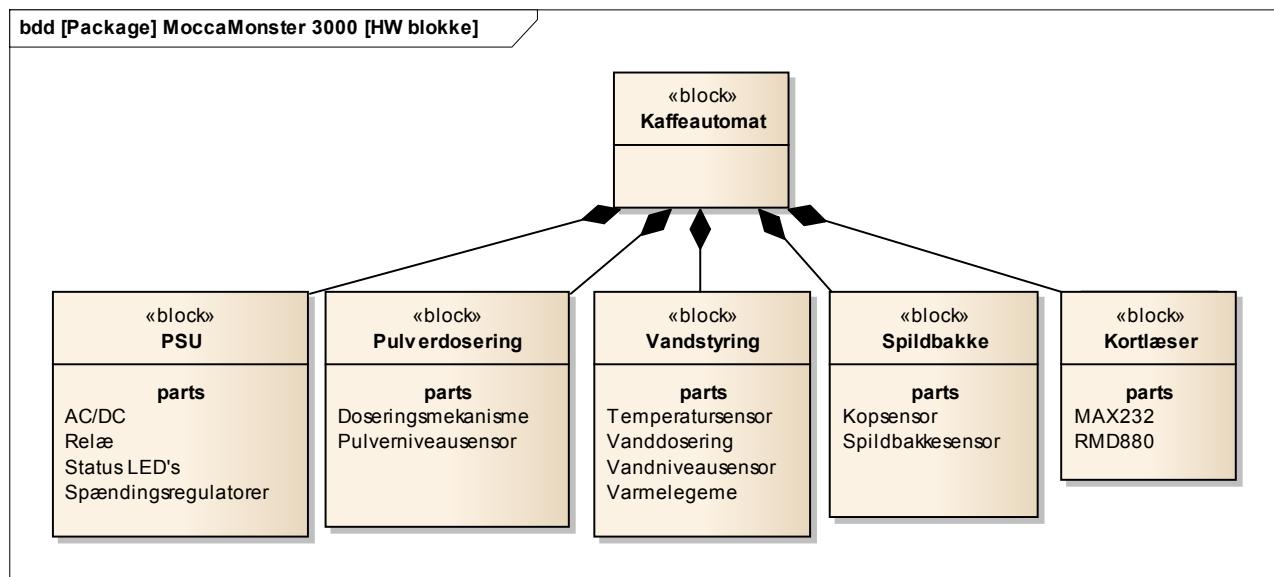


Figure 1 - Overordnet hardware bdd

5.1. PSU

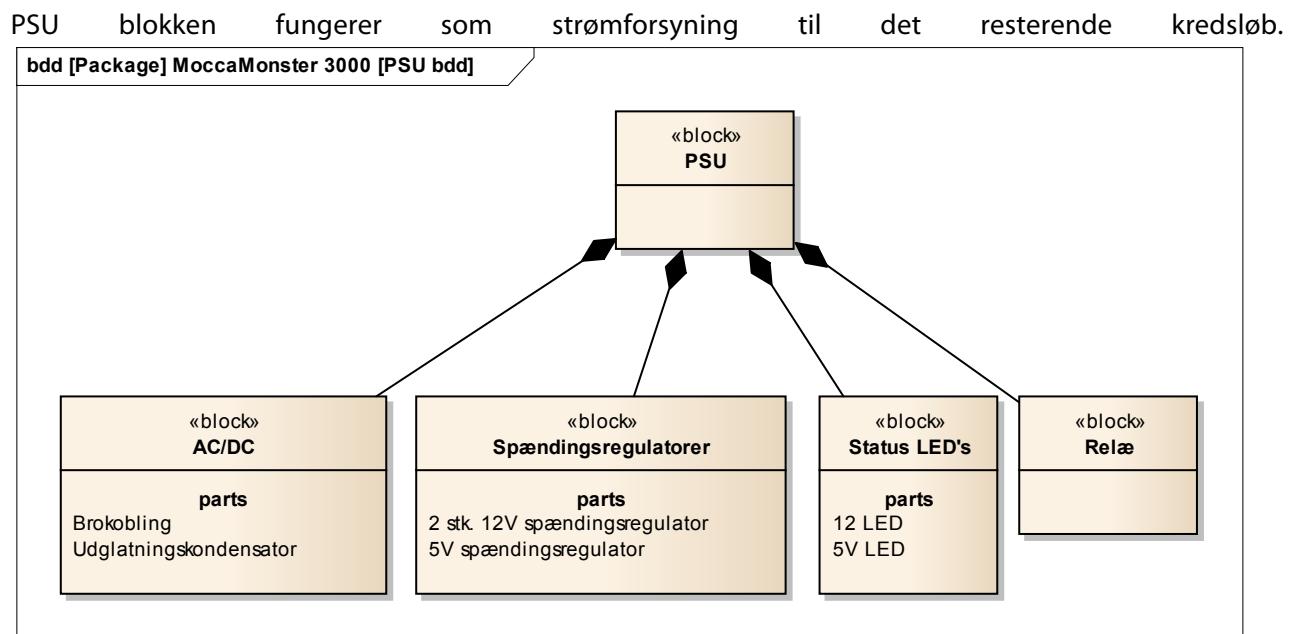
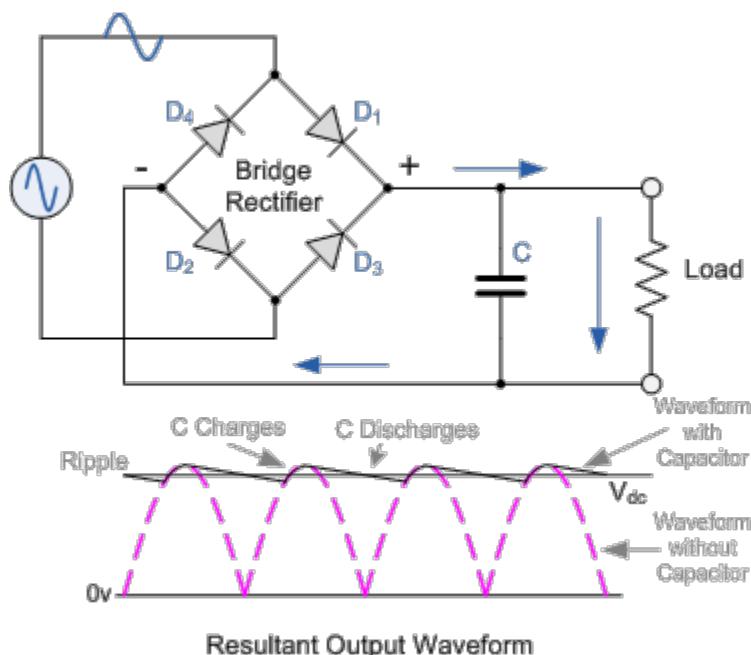


Figure 2 - PSU bdd

PSU'en får input i form af 18Vrms fra en transformer, koblet til lysnettet(230VAC). Dette 18Vrms input transformeres internt i PSU'en til de ønskede outputspændinger på henholdsvis 12VDC og 5VDC.



Figur 30 - Billede der viser omformning af sinuskurve til DC-spænding

Den sinusformede 18Vrms vekselspænding der forbindes til AC/DC blokken ønskes transformert til en DC-spænding. Dette gøres ved først at føre signalet igennem en brokobling. Denne brokobling vender de negative dele af sinuskurven til at være positive, og man har nu et sinusformet signal, der kun har positive svingninger. Signalet har dog stadig "dyk" mellem hver halve periode. For at fjerne dette anvendes en "udglatningskondensator". Denne kondensator lades op når sinussens spænding stiger, og aflader når den falder igen. Dette vil udjævne spændingen, og dermed skabe en konstant DC-spænding. For at beregne størrelsen på udglatningskondensatoren skal det samlede strømforbrug for kredsen være kendt. Dette vil være en unødvendig stor beregning, og der er derfor valgt en kondensator, der er vurderet til at være stor nok. Den valgte kondensator har en kapacitans på $220\mu F$. Det er vist ved en hedstest for PSU'en at kondensatorens kapacitans er tilstrækkelig.

DC-spændingen udgangsspændingen fra AC/DC-blokken har en amplitude på:

$$V_{DC} = 18 \cdot \sqrt{2} = 25,5VDC$$

For at regulere de 25,5VDC fra AC/DC'en til de ønskede outputspændinger anvendes spændingsregulatorer af typen 78xx. Som tidligere nævnt er der ikke foretaget beregninger på det samlede strømforbrug for kredsen. Der kan dog beregnes et forsigtigt overslag ved at regne worst case scenario, hvor magnetventilen til vanddoseringen er aktiveret, da dette er den komponent der optager den største effekt, og dermed

$$I_{magnetventil} = \frac{P}{V} = \frac{10W}{12V} = 833mA$$

største strøm.

Derudover er der i kredsløbet implementeret 3 stk. motordrivere af typen A3967SLB-T. Disse har hver et idle strømforbrug på 65mA.

$$I_{worst} = 3 \cdot 65mA + 833mA = 1028mA$$

Denne strøm skal løbe igennem LM7812. Denne har en maksimal belastningsstrøm på 1A, og det er når spændingsregulatoren er "*With adequate heatsinking*". For at mindske belastningen er kredsløbet dupliseret så det indeholder 2 stk. brokobliger, 2 stk. udglatningskondensatorer og 2 stk. LM7812. Det vurderes at systemet stadig er underdimensioneret, men af økonomiske årsager videreudvikles der ikke på denne løsning.

Det er ved test af motordriver konkluderet at det er nødvendigt med forsinket indkobling af 12VDC i forhold til 5VDC. Dette gøres med et relæ der indkobler de 12VDC. Dette relæ bliver trukket af 5VDC, og der er dermed sikret en forsinket indkobling.

Der er desuden implementeret grønne status LED's der indikerer om der er henholdsvis 5VDC og 12VDC.

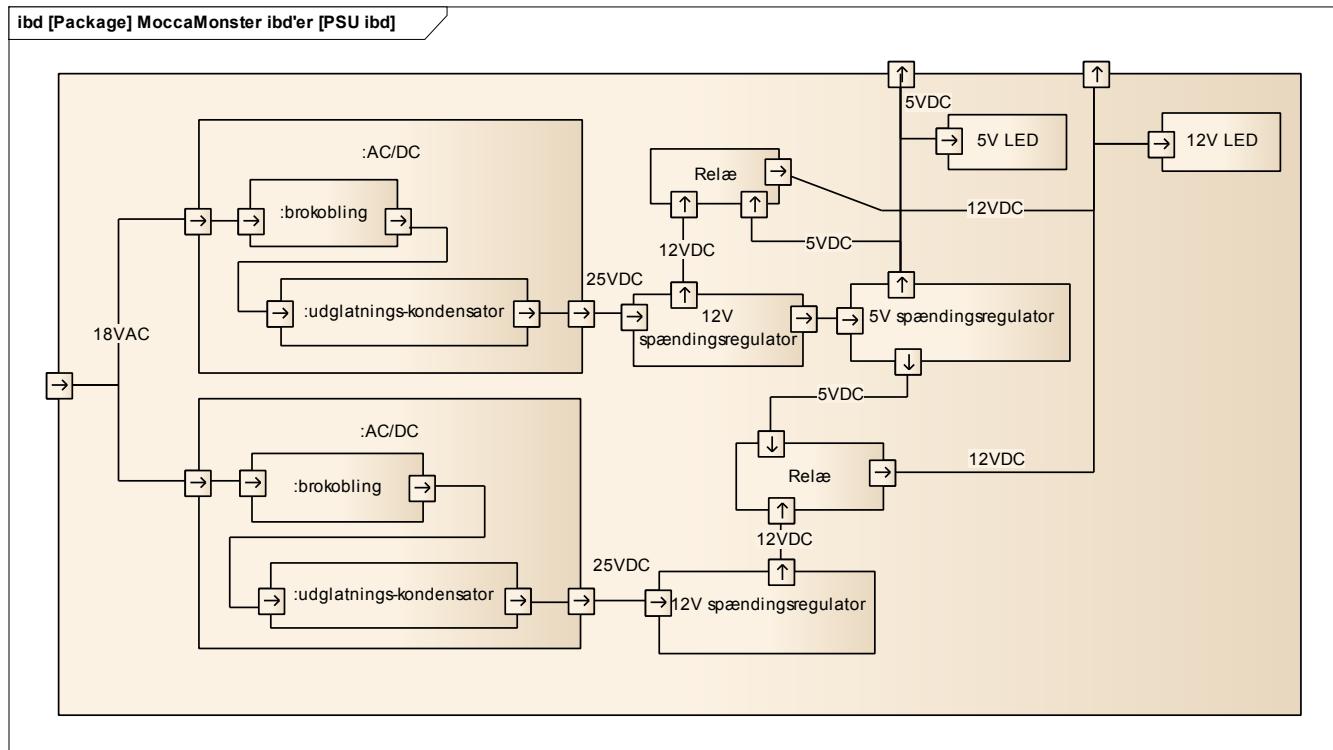


Figure 3 - idb for PSU

5.2. Pulverdosering

Pulverdoseringen skal sørge for at den korrekte mængde pulver bliver doseret og sende status på pulverniveau til PSoC. Pulverdoseringen består overordnet af en *Pulverniveausensor* og en *Doseringmekanisme*.

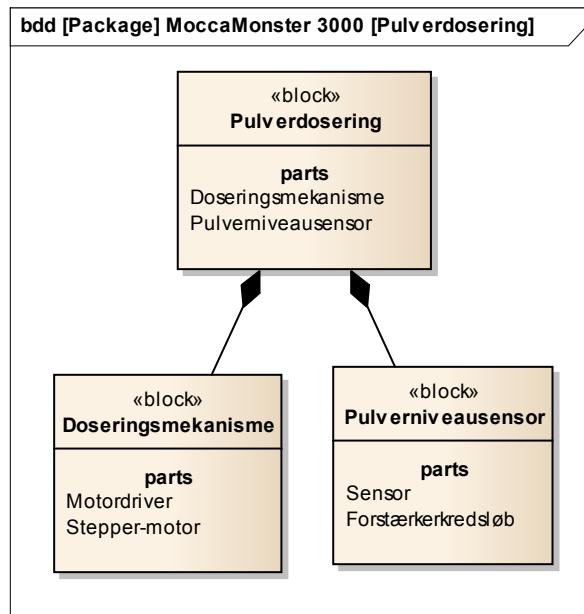


Figure 4 - bdd for "Pulverdosering"

5.2.1. Pulverniveausensor

Pulverniveausensoren skal måle hvor meget pulver der er tilbage i et pulverreservoir. Pulverniveausensoren sender et analogt signal til PSoC' en. Denne blok består af en sensor og et forstærkerkredsløb.

5.2.2. Sensor

Sensoren der bruges i dette projekt er en analog optisk sensor fra SHARP af typen GP2Y0A41SK0F. Denne model har et detektionsområde fra 4cm-30cm, hvor sensoren vil returnere ca. 0,3mV ved 30cm og ca. 2,85V ved 4cm. Disse værdier er fundet ved aflæsning fra databladet²⁰.

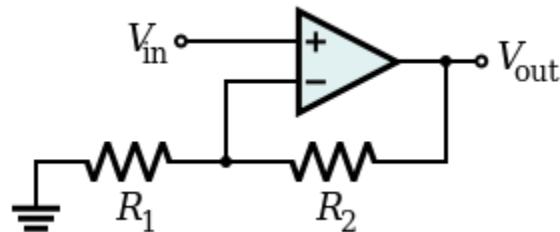


Figur 31 - SHARP afstandsmåler

²⁰ Se Bilag 4.13

5.2.3. Forstærkerkredsløb

Det analoge output direkte fra GP2Y0A41SK0F er i spændingsintervallet 0-2,85VDC. Signalet ønskes forstærket op til et spændingsinterval på 0V-5VDC. Dette gøres med en operationsforstærker af typen MCP604, koblet som ikke-inverterende forstærker, se Figur 32.



Figur 32 - Ikke-inverterende forstærker

Den generelle formel for output spændingen for en ikke-inverterende forstærker er:

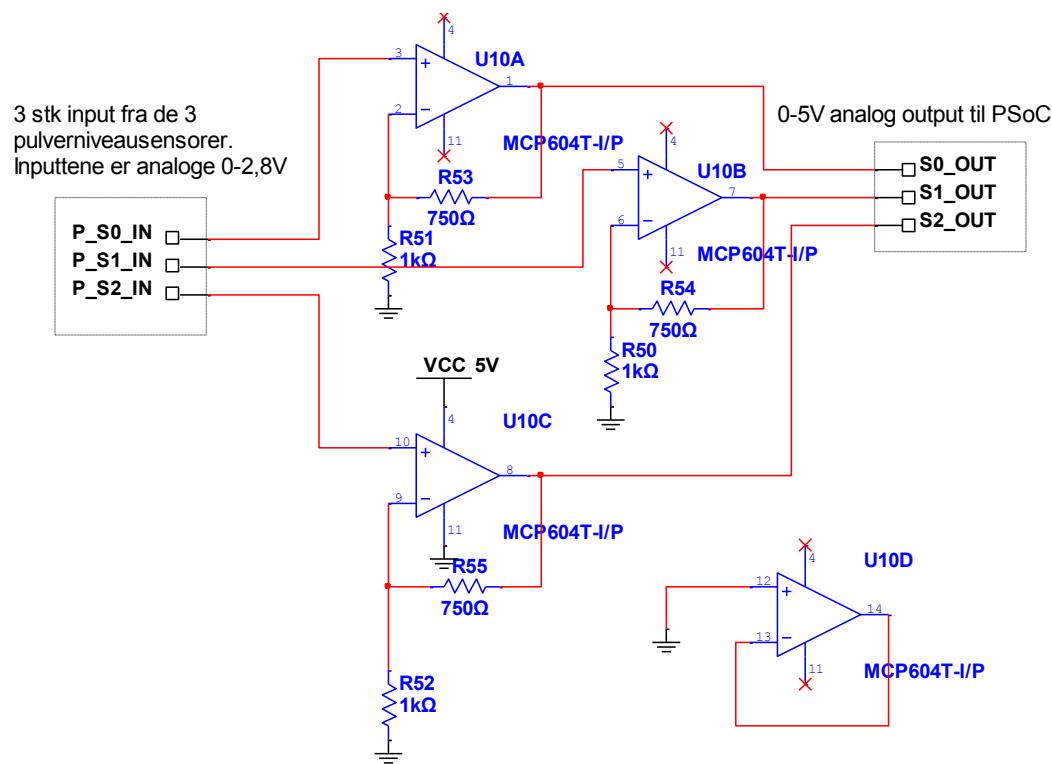
$$V_{\text{out}} = V_{\text{in}} \left(1 + \frac{R_1}{R_2} \right)$$

Modstanden R_1 vælges til $1\text{k}\Omega$.

For at opnå V_{out} på 5V beregnes værdien af modstanden R_2 :

$$5\text{V} = 2,85\text{V} \cdot \left(1 + \frac{1\text{k}\Omega}{R_2} \right) \Leftrightarrow R_2 = R_1 \left(\frac{V_o}{V_{\text{in}}} - 1 \right) = 1\text{k} \left(\frac{5}{2,85} - 1 \right) = 754\Omega$$

R_2 implementeres med en værdi på 750Ω , da denne er til rådighed fra laboratoriets lager.



Figur 33 - diagram over pulverniveausensorer

5.3. Doseringsmekanisme

Doseringsmekanismen skal sørge for at dosere pulver. Denne blok består af en Motordriver og Stepper-motor, som er samlet i følgende afsnit.

5.3.1. Motordriver og stepper-motor

Motoren til doseringsmekanismen er en stepper-motor af typen LE_3304. Denne motor styres af en driver IC af typen A3967SLB-T fra Allegro Microsystems. Driveren er fremstillet til at kunne styre bipolære stepper-motorer. Den valgte motor til dette projekt er unipolær. Unipolære stepper-motorer kan dog konfigureres til at operere som bipolære, på grund af motorernes interne konstruktion. Her anvendes udtage-ne på midterviklingerne(1 og 2 på figur 5) ikke.

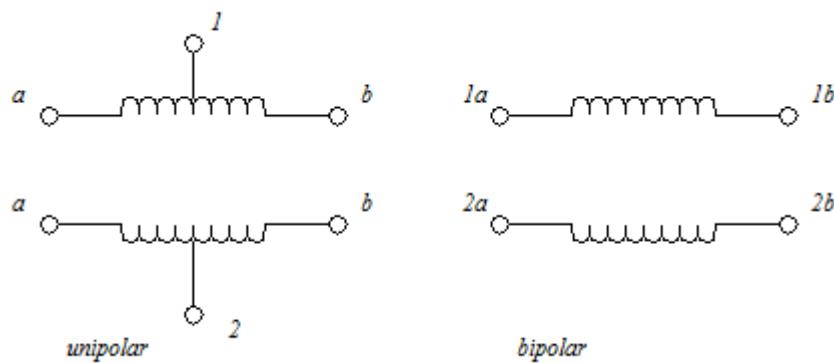


Figure 5 – Figur over ledningsudtag for unipolære og bipolære stepper motorer

I forbindelse med valg af motordriver, var det primære fokuspunkt at driveren skulle kunne levere en tilstrækkelig strøm til at drive motoren. Understøttet opløsning, og andre features havde lavere prioritet. På baggrund af dette blev førnævnte driver valgt. Denne kan håndtere en kontinuert strøm på 750mA, og en peakstrøm på 850mA.

A3967SLB-T har en del indstillingsmuligheder der er taget stilling til. Her følger nogle af de væsentligste:

REF(pin1): Er en reference spænding der begrænsrer den maksimale strøm der kan løbe gennem load-kredsløbet i A3967SLB-T. Den maksimale strøm bliver desuden begrænset af eksterne modstande, SENSE1 og SENSE2. Forholdet mellem disse kan beskrives ved ligningen:

$$I_{TRIP\max} = \frac{V_{REF}}{8R_S}$$

Denne begrænsning af strøm anvendes kun ved større forsyningsspænding end hvad motoren kræver. I denne opstilling forsynes A3967SLB-T med 12VDC, som er motorens mærkespænding. Der er derfor ikke behov for at begrænse strømmen. Derfor forbindes REF til VCC.

SLEEP(pin3): Dette aktiv lave input sætter A3967SLB-T i en dvaletilstand. I denne tilstand bliver det interne kredsløb lukket ned, og standby strømmen reduceres kraftigt fra 65mA til 20µA. Brug af sleep-funktionen har været på tale, da det ville mindske den konstante belastning på PSU'en, men blev fravalgt, på grund af mangel på pins på PSoC. Da sleep ikke skulle anvendes, blev denne hardwired til logisk high(5VDC).

DIR(pin11): DIR er et logisk input, der sætter motorens omdrejningsretning. Der er i projektet ikke behov for at kunne ændre retning, derfor forbindes denne til logisk lav (GND).

MS1(pin12) og MS2(pin13): Bestemmer opløsningen på stepper-motoren. A3967SLB-T²¹ understøtter opløsningerne full-step, half-step, quarter step og eight step. Valg af opløsning er en balancegang mellem

²¹ Se Bilag 4.7

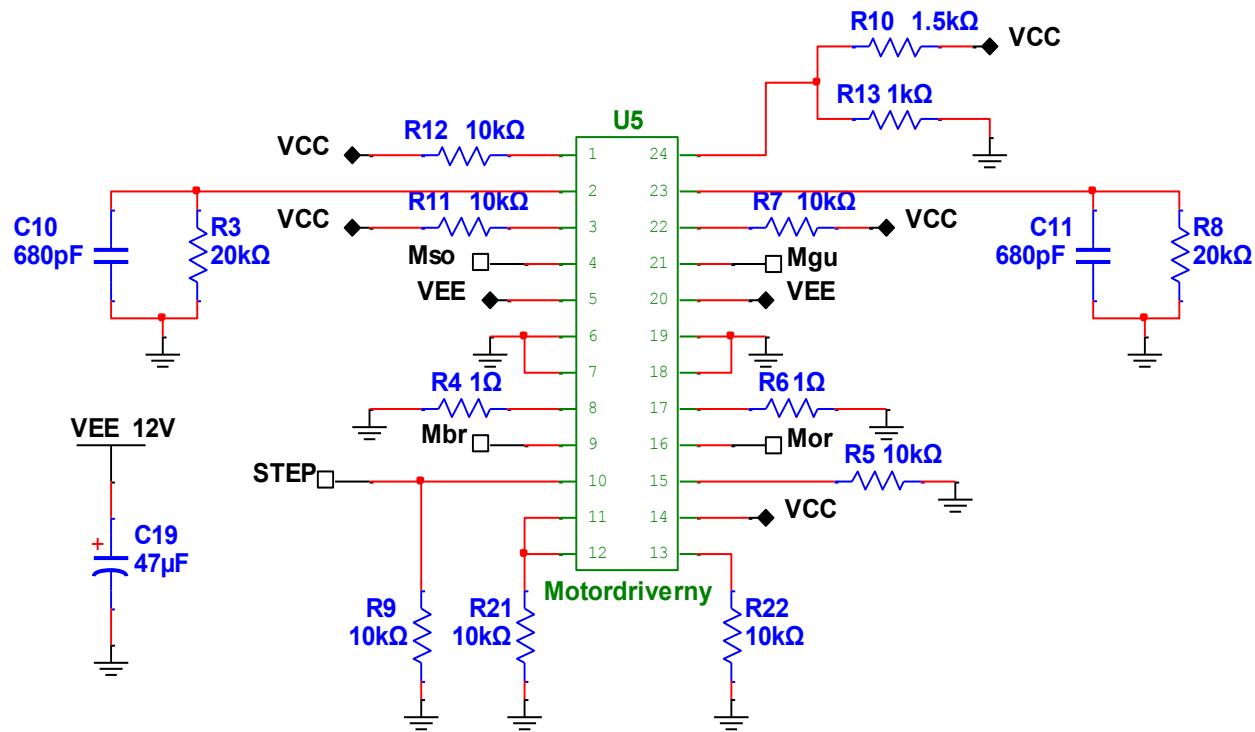
moment, hastighed og naturligvis opløsning. Ved højere opløsning opnår man større præcision, men lavere moment og hastighed. Ved lav, og dermed grovere, opløsning opnår man større moment. Det er derfor nødvendigt at finde den rette opløsning til det givne scenarie. Opløsningen blev sat til full step, efter forsøg i laboratoriet.

Micro steps/full step	Holding Torque/Micro step
1	100.00%
2	70.71%
4	38.27%
8	19.51%
16	9,80%
32	4,91%
64	2.45%
128	1.23%
256	0,61%

Kilde: <http://www.micromo.com/microstepping-myths-and-realities.aspx>

ENABLE(pin15): Dette aktivt lave input enabler/disabler outputtene fra A3967SLB-T. Denne funktion er vurderet til ikke at være nødvendig, og benet er derfor hardwired til GND. Ved et logisk high reduceres standby strømmen fra 65mA til 9mA. Grunden til dette relativt store fald i standby strøm er at driveren låser motorens placering, når motoren ikke er bevægelse. Når ENABLE deaktiveres, vil motoren derfor ikke være låst. Dette er dog ikke et problem i dette projekt, da motoren ikke skal holdes i en låst stilling.

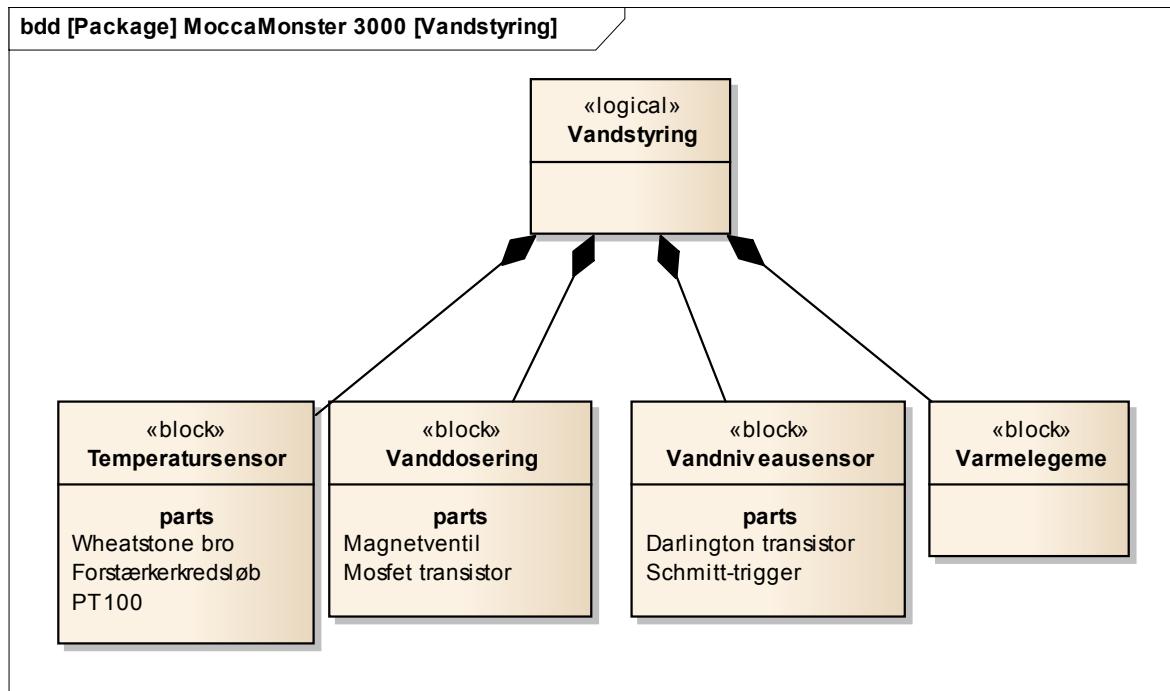
Under testfasen kom det på tale at det var nødvendigt med en forsinket indkobling af de 12VDC der forsyner load-kredsløbet på A3967SLB-T, i forhold til de 5VDC der forsyner den interne logik. Dette er implementeret i PSU'en.



Figur 34 - diagram over motordriveren

5.4. Vandstyring

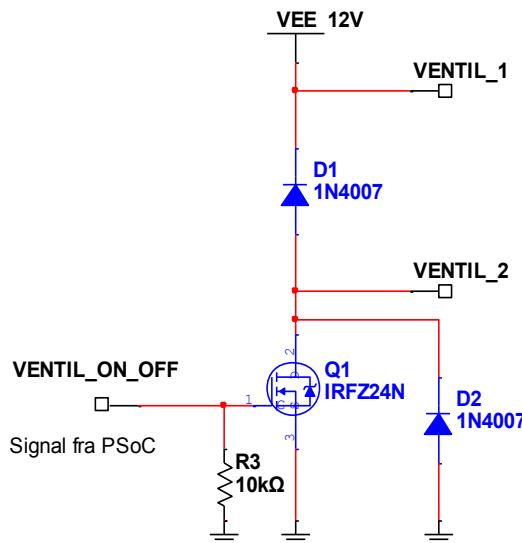
Vandstyrings blokkens overordnede funktioner består af at dosere vand til brugerens kop, opvarme vandet til rette temperatur og registrere hvor meget vand der er tilbage i vandbeholderen.



Figur 35- bdd for Vandstyring

5.4.1. Vanddosering

Vanddoseringsblokken sørger for at dosere vand ned i brugerens kop. Dette opnås ved at åbne en magnetventil med et digitalt signal.



Figur 36 - diagram for styring af magnetventil til vanddosering

5.4.1.1. Magnetventil

Magnetventilens detaljerede specifikation kendes ikke, da den er fundet på skolens lager. Den er mæret med 12V, 10W, der medfører en strøm på 833mA. Strømmen måles i laboratoriet til ca. 750mA. Modstanden i spolens viklinger måles med ohmmeter til $14,5\Omega$.

5.4.1.2. Mosfet transistor

Til dette kredsløb bruges en mosfet transistor af typen IRFZ24N. Den kan klare en max strøm på 17A, ifølge databladet, hvilket ikke giver komplikationer da magnetventilen trækker en strøm på 750mA.

På følgende graf ses det at ved en Gate-to-Source voltage på 5V, vil der kunne løbe en strøm på ca. 6A, hvilket er acceptabelt i forhold til magnetventilens forbrug.

Når mosfet transistoren modtager 5V på ben 1(gate), vil den tillade at der kan løbe strøm gennem den, og derved får hele kredsen forbindelse til GND.

Dioden D2 af typen 1N4007 fungerer som beskyttelse af mosfet transistoren. Den sikrer at spændingen, som er tilovers når magnetventilen kobles fra, forbliver mellem 0V til 12V. Denne restspændingen bliver induceret i spolens viklinger, og indeholder nødvendigvis ikke en stor strøm, men

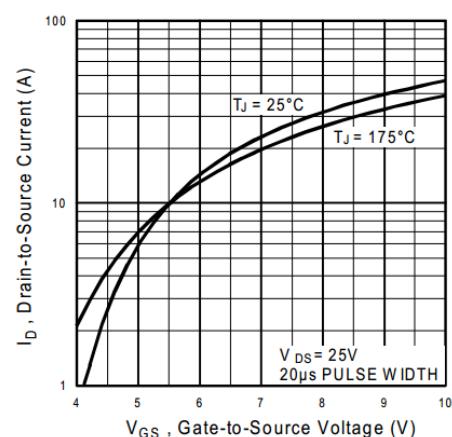
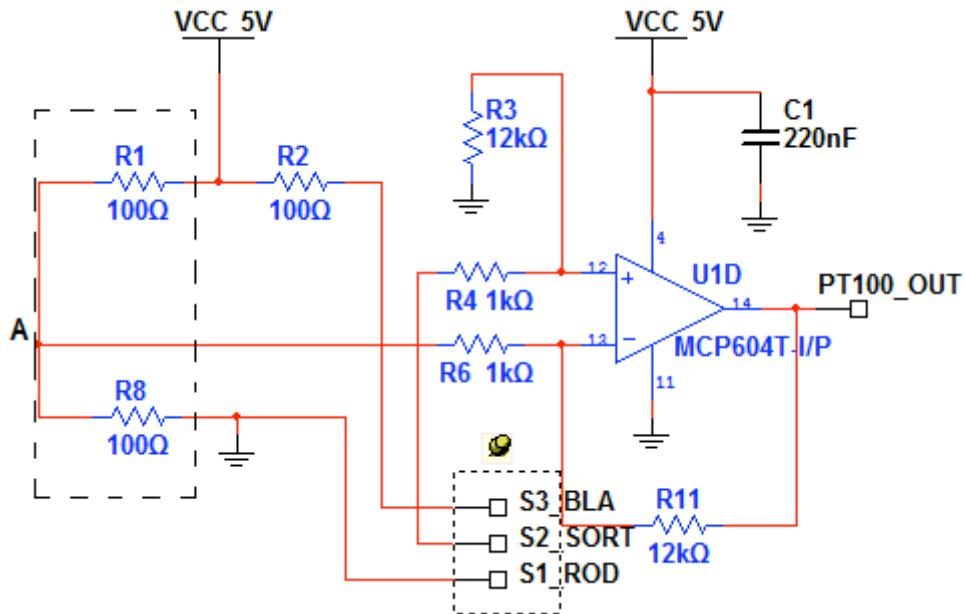


Fig 3. Typical Transfer Characteristics

dog en høj spænding.

5.4.2. Temperatursensor

Temperaturføleren skal kunne måle temperaturen i vandbeholderen og returnere et analogt signal mellem 0V-5V til PSOC' en. Overordnet består denne komponent af en wheatstone bro, PT100-føler og en forstærkerkreds.



Figur 38 - diagram for PT100 kredsløbet

5.4.2.1. PT100-føler

PT100-føleren der bruges i dette projekt er 3-benet. 3-benet medfører at ledningsmodstanden i PT100'eren formindskes, og der kompenseres derved for offset fejl. Dette medvirker at PT100'erens præcision stiger.

PT100-føleren ændrer sin modstand når temperaturen omkring denne ændres. Til dette projekt bruges føleren til at registrere når vandet i vandbeholder har nået 100 °C (kogepunkt). Derfor bestemmes følerens måleområde til 0 °C-100 °C. Dette vil give en modstandsændring (ΔR) på:

$$0^\circ\text{C} = 100\Omega$$

$$100^\circ\text{C} = 138,5\Omega$$

$$\Delta R = 138,5 - 100 = 38,5\Omega$$

Modstandsværdierne er fundet ud fra følgende tabel:
http://en.wikipedia.org/wiki/Resistance_thermometer

5.4.2.2. Wheatstone bro

Wheatstone broen vil, sammen med PT100-føleren, omsætte en temperaturændring[°C] til en ændring i spænding[V]. Broen er implementeret med 4 modstande(inkl. føleren), hvilket er nødvendigt for at overholde designkravet²², om at alle sensorer skal returnere en spænding mellem 0V-5V til PSoC' en. Broen kunne implementeres med kun 2 modstande, da der internt i PSoC programmet kan sættes en reference spænding på ADC'ens ene inputben. Referencen sættes til 2,5V da det er denne spænding som gren A konstant sender til PSoC' en.

Broen forsynes med +5VDC, og de 3 andre modstande bestemmes til 100Ω . Ved at sætte disse modstande til 100Ω , vil outputtet fra broen til differensforstærkeren være ens, ved $0\text{ }^{\circ}\text{C}$ – og derfor vil hele kredsen sende 0V til PSoC'en.

Gren

A

Gren A vil, da modstandsværdierne er konstante, sende 2,5V ud til differensforstærkeren:

$$V_{out} = \frac{100\Omega}{100\Omega + 100\Omega} * 5V = 2,5V$$

Gren

B

(PT100)

føler)

Med den fundne modstandsændring kan spændingen, der findes ved en temperatur på $100\text{ }^{\circ}\text{C}$ beregnes:

$$V_{out} = \frac{V_b(R + \Delta R)}{R + R + \Delta R} = \frac{(100\Omega + 38,5\Omega)}{100\Omega + 100\Omega + 38,5\Omega} * 5V = 2,904V$$

Wheatstone broen vil derfor sende følgende ændring i spænding ud:

$$\Delta V = 2,904V - 2,5V = 0,404V$$

²² Se afsnit 3.3.2

5.4.2.3. Forstærker

Da det ønskes at forstærke ΔV bruges en differensforstærker. En differensforstærker er en specifik kobling af en operationsforstærker. Koblingen ses på Figur 10. Differensforstærkeren skal forstærke spændingsdifferensen fra 0,4V til 5V. Det vil sige følgende forstærkning er nødvendig:

$$5V = 0,404V \cdot X \Leftrightarrow X = 12,3$$

Der opsættes følgende bestemmelse for modstandene:

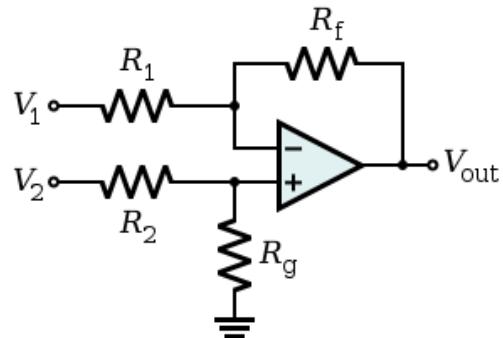
$$\frac{R_f}{R_1} = \frac{R_g}{R_2}$$

Derved kan outputtet for differensforstærkeren beregnes med følgende formel:

$$V_{out} = A(V_2 - V_1)$$

Hvor A er den differentielle forstærkning for kredsen, der beregnes således:

$$A = \frac{R_f}{R_1}$$



Figur 39 - Differensforstærker

Forstærkning vælges til 12, og modstandsværdierne findes:

$$A = \frac{R_f}{R_1} \Leftrightarrow 12 = \frac{12k\Omega}{1k\Omega}$$

Dvs.: $R_f = R_g = 12k\Omega$ og $R_1 = R_2 = 1k\Omega$.

5.4.3.

Varmelegeme

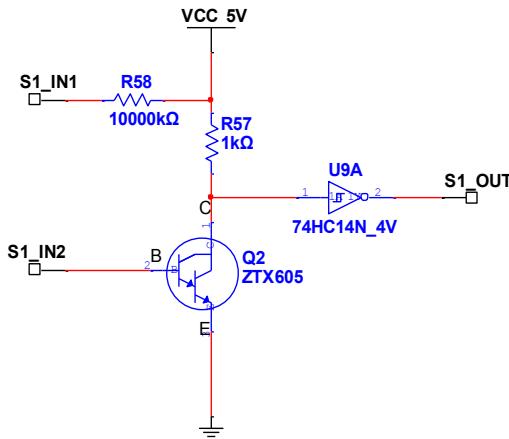
Til at opvarme vandet i vandbeholderen, blev der købt en dybkoger. Denne skal forsynes fra el nettet med 230VAC. Studerende på IHA må ikke arbejde direkte med 230V, og derfor blev der købt en fjernbetjent stikkontakt.

Fjernbetjeningen til stikkontakten skilles ad, og kobles direkte på PSoC' en, som derved kan tænde/slukke stikkontakten med et digitalt signal.

Denne løsning sikrer at de studerende i gruppen ikke arbejder med 230V, og varmelegemets funktionelit beholdes.

5.4.4. Vandniveausensor

Vandniveausensoren skal detektere om der er vand i vandbeholderen. Den sender et digitalt signal, der beskriver hvorvidt der detekteres vand eller ej. Komponenten består af en darlington transistor og en Schmitt-trigger.



Figur 40 - diagram for vandniveausensor

5.4.4.1. Darlington transistor

En darling transistor består af to bipolare transistorer som har fælles collector-ben. Emitter-benet på den første kobles til base benet på den anden transistor. Derved bliver strømmen forstærket over to omgange og Darlington transistoren kan derved registrere selv berøring af en finger.

Denne type transistor kan bruges til at detektere vand, da vandet vil kortslutte S1_IN1 og S1_IN2, og til-lade der løber strøm gennem hele kredsen.

I dette projekt bruges en BC517 PNP Darlington transistor fra Philips.

5.4.4.2. Schmitt-trigger

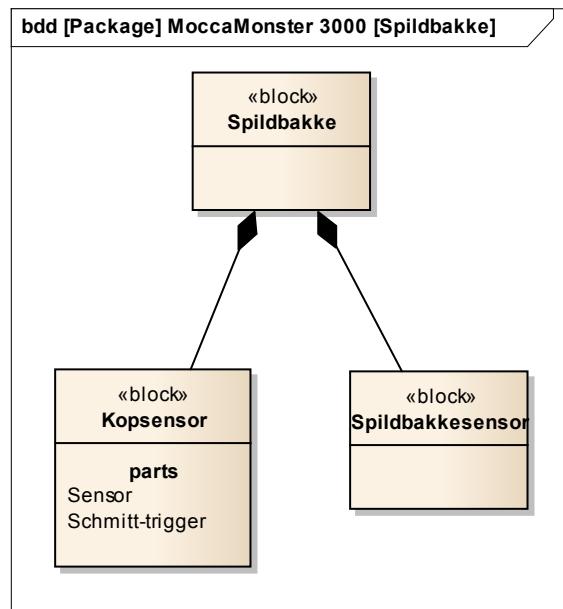
Schmitt-triggeren i vandniveausensoren har til formål at omdanne det analoge signal til et digitalt signal. I dette projekt bruges en Schmitt-trigger af typen 74HC14N. Udgangen fra en Schmitt-trigger kan enten være logisk høj eller lav. Derfor er den ideel til at konvertere en analog værdi fra en sensor, til et digitalt signal. Schmitt-triggeren der her anvendes er inverterende. Ved at koble signalet gennem to Schmitt-triggere omformes et analog input signal i intervallet 0V til 0,85V til digital lav(0V), og et signal fra 1,2V til VCC til digital høj(5V).

Denne komponent er, efter integrationstest, valgt ikke at implementeres. Se integrationstest afsnittet for forklaring²³.

²³ Se afsnit 5.9

6. Spildbakke

Denne blok indeholder to funktionaliteter. En sensor, som skal detektere hvor vidt brugeren har placeret sin kop under dysen, og en sensor som registrer om spildbakken er fyldt. Følgende bdd beskriver sammenhængen.

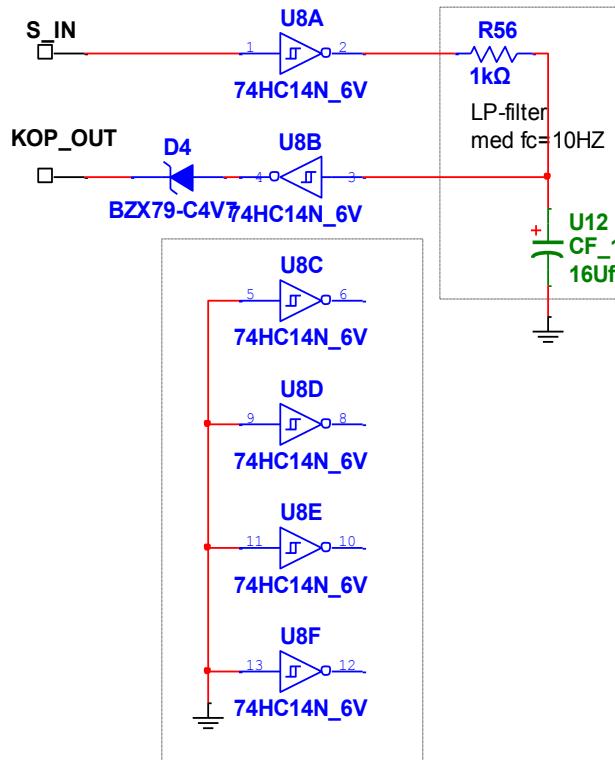


Figur 41 - Spildbakke bdd

Spildbakken blev ikke implementeret i det endelig produkt.

6.1. Kopsensor

Kopsensoren skal sende et digitalt signal til PSoC' en, hvor et digitalt høj svarer til at der er detekteret en kop. Komponentens opbygning er tilsvarende Pulverniveausensoren, uden forstærkning, men her er der uddover koblet en Schmitt-trigger.



Figur 42 - diagram for kopsensoren. Det ses at kopsensoren er implementeret med Schmitt-triggere, for at give et digitalt output

Der er koblet en zenerdiode (D4), med en zenerspænding på 4,7V på outputtet af kredsen, således sikres det at der ikke sendes en spænding større end +5VDC til PSoC' en.

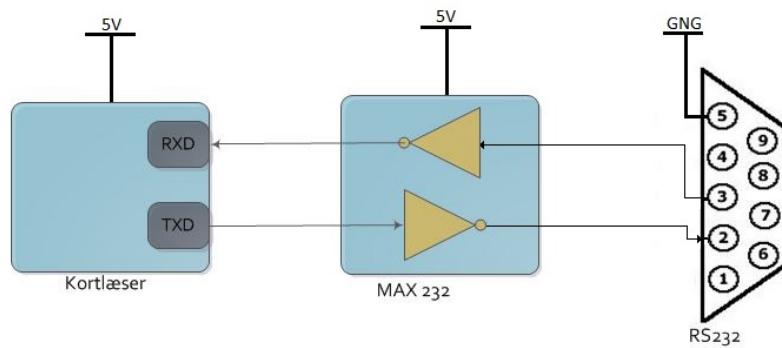
6.1.1. Schmitt-trigger

Ved at koble signalet gennem to Schmitt-triggere omformes et analog input signal i intervallet 0V til 0,85V til digital lav(0V), og et signal fra 1,2V til VCC til digital høj(5V).

Ved aflæsning fra databladet ses det at fra ca. 1,5cm til 8cm vil der være over 1,5V. Derved kan vi få en registreret en kop indenfor intervallet 1,5cm-8cm.

6.2. Kortlæser

Kortlæseren skal kunne registrere brugerens adgangskort og sende data til DevKit via en seriel kommunikation. I henhold til systemarkitekturen er chippen MFRC531²⁴ valgt, da det var den billigste IC som understøttede SPI og kan læse rfid tags på 13,56 MHz. Denne IC viste sig hurtigt at være meget avanceret og derudover skulle en antennen designes. Bl.a. skulle udgangsimpedansen til antennen sættes kort i interne register i IC'en og kommunikations interfacet skulle sættes op da den havde mange andre interfaces end blot spi. Det blev vurderet at opgaven var omfattende, og der blev derfor sat en timeframe på denne. Ud fra informationen i databladet samt andre applikationsnotes, blev der designet en passiv antennen. For at mindske støj og give en mere stabil antennen blev der designet et filter til antennen²⁵. Et styrekredsløb til IC'en blev designet efter antennens specifikationer og spi interfacet på DevKit'. Da opgaven ikke var løst indenfor den pågældende timeframe blev det besluttet at der skulle findes en alternativ løsning. Et færdigt komponent (inkl. antennen), kunne købes indenfor budgettets rammer. Chippen RMD880 havde UART interface. Dette medførte at der skulle designes noget simpelt hardware til levelkonvertering. Softwaremæssigt var løsningen minimalt anderledes da UART-kommunikation som standard understøttes på DevKittet, og derfor er det kun fil-placering som er anderledes i forhold til det oprindelige design med SPI.



Figur 43 princip i UART

RMD880 er ikke særligt intelligent og har ingen interrupt-linjer. Hvilket betyder at der skal anvendes polling for at modtage data fra kortlæseren.

²⁴ Se Bilag 4.3

²⁵ Se Bilag 4.4

En specifik 8-byte kommando skal sendes til kortlæseren, som svarer med 6 byte, hvis intet kort er registreret, og 11 byte såfremt kort registeres. 5 af disse 11 bytes er det unikke kortnummer.

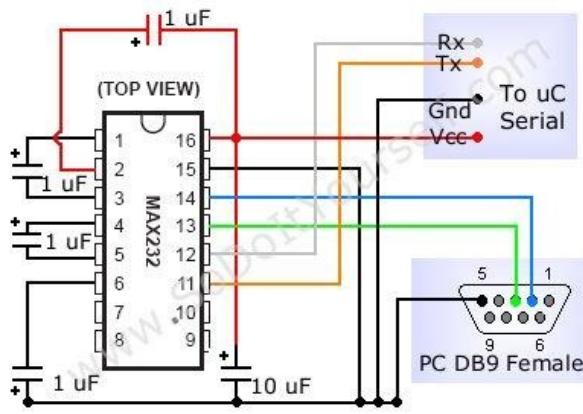
Kommandoer til RDM880

Forespørgelse på status	0xAA00032526000BB
Status uden kort	0xAA0002018380BB
Status med kort	0xAA00060000PPPPPPBBBB*

*Hvor P... er 5 bytes kortnummer. Eks:

0xAA00060000C4EAFC700BB

Da RS-232 portene på Devkit'et er 12V og RMD880-chippen kun skal 5V signaler, er der behov for levelkonvertering. Til dette anvendes en MAX 232.



Figur 44 diagram for MAX232

Det har vist sig at de tre RS-232 porte på Devkittet som standard er write-only. Derfor skal TTY00 konfigureres til at understøtte læsning. Dette gøres med konfigurationsflag i et boot-script²⁶.

²⁶ Se Bilag 1.4

6.3. Enhedstest

Dette dokument beskriver de hardware enhedstest der er lavet for at dokumentere hardwareenheder-nes funktion.

6.3.1. Temperatursensor

Ekstern grænseflade

Signal	Beskrivelse
Analog* output	Analog signal der returner en spænding der i ADC omsættes til en decimal værdi der repræsenterer den målte temperatur.
Fysisk input	Vandtemperaturen som sensoren skal måle.

Intern grænseflade

Signal	Beskrivelse
+5VDC	Forsyning til operationsforstærker, MCP604.
GND	Forsyning til operationsforstærker, MCP604
PT100 ben1 (blå)	Forbindelse til PT100 sensoren.
PT100 ben2 (sort)	Forbindelse til PT100 sensoren.
PT100 ben3 (rød)	Forbindelse til PT100 sensoren.

*Beskrivelse af "analog" findes i systemarkitekturen.

Enhedstest

Der er 2 testscenarier i denne enhedstest.

1. Test ved stuetemperatur(20grader celcius)
2. Test ved kogepunktet(100grader celcius)

Overordnet opstilling

Signal	Fra pin	På enhed	Til pin	På enhed
+5VDC	CH1, rød	Spændingsgenerator	4	U1(MCP604)
GND	CH1, sort	Spændingsgenerator	11	U1(MCP604)
Analog output	14	U1(MCP604)	CH1, rød	Oscilloskop
Modstandsændring	Blå	PT100 føler	S3_BLA	-
Modstandsændring	Sort	PT100 føler	S2_SORT	-
Modstandsændring	Rød	PT100 føler	S1_ROD	-

Test specifik opstilling

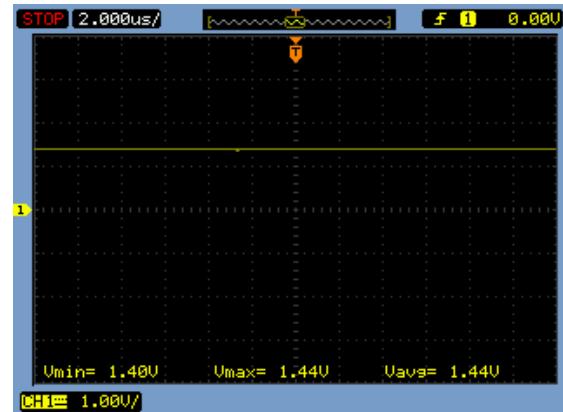
1. PT100 føleren placeres i kalibreringsenhed med temperaturen 20 grader celsius.
2. PT100 føleren placeres i kogende vand.

Forventet output

1. Der forventes at måle +1VDC($\pm 0,5$ VDC) på pin 14 U1(MCP604). Se dokumentationen for Varmenvæusensoren for udregning af denne spænding.
2. Der måles på pin 14 på U1(MCP604) +5VDC($\pm 0,5$ VDC).

Målinger

1. Oscilloskop billede af test1 - PT100 føleren placeret i kalibreringsenhed med temperaturen 20 grader celsius.



1. Oscilloskop billede af test2 - PT100 føleren placeret i kalibreringsenhed med temperaturen 100 grader celsius.



Resultat

1. Spændingsniveauet aflæses til +1,44VDC - **OK!**
2. Spændingsniveauet aflæses til +4,81VDC - **OK!**

6.3.2. Pulverniveausensor

Ekstern grænseflade

Signal	Beskrivelse
Analog* output	Analog signal der signalerer hvor meget pulver der er i en beholder.
Fysisk input	Lys.

Intern grænseflade

Signal	Beskrivelse
+5VDC	Forsyning til operationsforstærker, MPC604.
GND	Forsyning til operationsforstærker, MCP604.
Vo	Output signal fra IR-sensor. Bliver forstærket af MPC604.

*Beskrivelse af "analog" findes i systemarkitekturen.

Enhedstest

Der er 3 test-scenarier i denne enhedstest.

1. Pulverbeholderen fyldt helt op(0cm fra toppen).
2. Pulverbeholderen halvt fyldt op(8cm fra toppen).
3. Pulverbeholderen tom(15cm fra toppen).

Da sensoren sidder lidt nede i beholderen er toppen her defineret som sensorens minimale afstand til pulveret, altså ca. 4 cm.

Overordnet opstilling

+5VDC på "4" på MCP604.

GND på "11" på MCP604.

Pulverniveausensor #1

Input fra IR-sensor på "3" på MCP604.

Output på "1" på MCP604 forbindes til CH1 på oscilloskop.

Pulverniveausensor #2

Input fra IR-sensor på "5" på MCP604.

Output på "7" på MCP604 forbindes til CH1 på oscilloskop.

Pulverniveausensor #3

Input fra IR-sensor på "10" på MCP604.

Output på "8" på MCP604 forbindes til CH1 på oscilloskop.

Test specifik opstilling

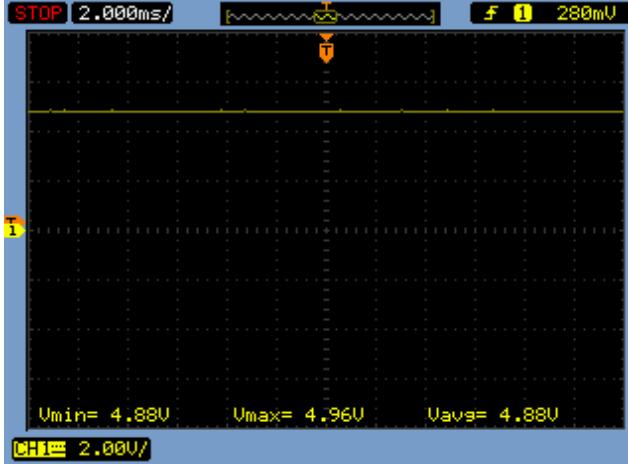
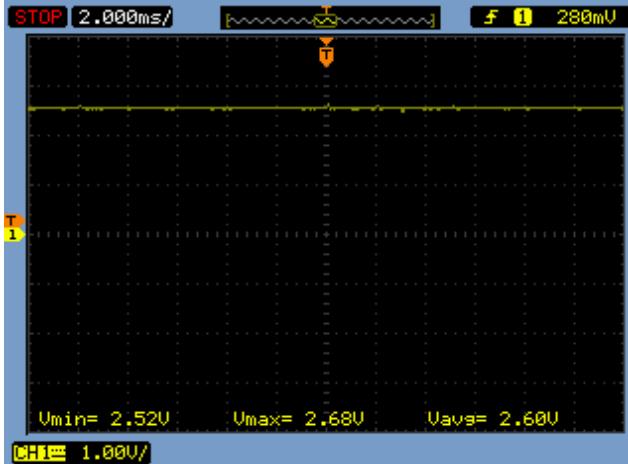
Denne opstilling er generel for test af alle 3 pulverniveausensorer.

1. Der placeres instantkaffe i et niveau svarende til de angivne niveauer for de 3 test-scenarier.
2. Pulverniveau sensoren placeres på beslag på toppen af pulverbeholder.

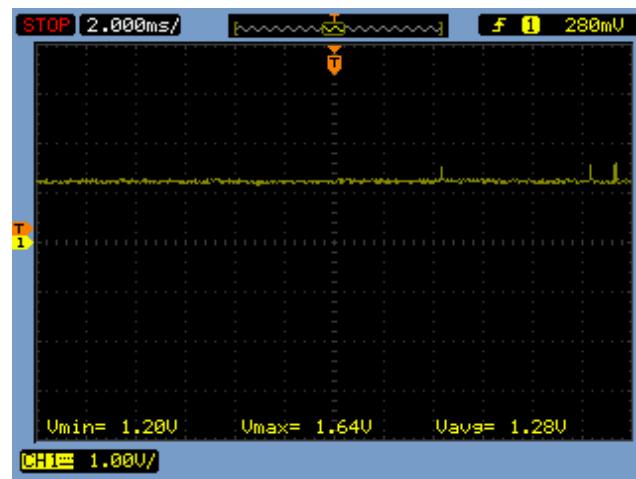
Forventet output

1. Da en fuld beholder vil svare til ca. 4 cm fra sensoren, kan der fra databladet aflæses en spænding på 2,75V. Med forstærkning på 1,75 forventes et output på 4,8V ($\pm 0,5$ V).
2. En halv fuld beholder svarer til ca. 8 cm fra sensoren, og der forventes derfor 1,55V. Med forstærkning forventes et output på 2,7V ($\pm 0,5$ V).
3. Da en tom beholder svarer til ca. 15 cm fra sensoren, forventes et output på ca. 0,9V inden forstærkningen. Med forstærkning bliver dette til 1,5V ($\pm 0,5$ V).

Målinger

<p>1. Oscilloskop billede af test1 - fyldt beholder</p>	
<p>2. Oscilloskop billede af test2 - halv fuld beholder</p>	

3. Oscilloskop billede af test3 - tom beholder



Resultat

1. Der måles +4,88VDC, hvilket er inden for tolerancen - OK!
2. Der måles +2,60VDC, hvilket er inden for tolerancen - OK!
3. Der måles +1,28VDC, hvilket er inden for tolerancen - OK!

Der er valgt kun at vise oscilloskop billeder fra sensor#1. Det er kontrolleret at de andre sensorer giver output inden for den givne tolerance.

6.3.3. Kopsensor

Ekstern grænseflade

Signal	Beskrivelse
Digital* out-put	Digital signal der signalerer hvorvidt der er en kop foran sensoren.
Fysisk input	Lys.

Intern grænseflade

Signal	Beskrivelse
+5VDC	Forsyning til U2, Schmitt trigger, 74HC14.
GND	Forsyning til U2, Schmitt trigger, 74HC14.

*Beskrivelse af "digital" findes i systemarkitekturen.

Enhedstest

Der er 2 test-scenarier i denne enhedstest.

1. Kop foran sensoren.
2. Ingen kop foran sensoren.

Overordnet opstilling

Overordnet opstilling for begge test scenarier.

Signal	Fra pin	På enhed	Til pin	På enhed
+5VDC	CH1, rød	Spændingsgenerator	14	U2
GND	CH1, sort	Spændingsgenerator	7	U2
Input fra IR-sensor	Vo	IR-sensor	1	U2
Output	4	U2	CH1, rød	Oscilloskop

Test specifik opstilling

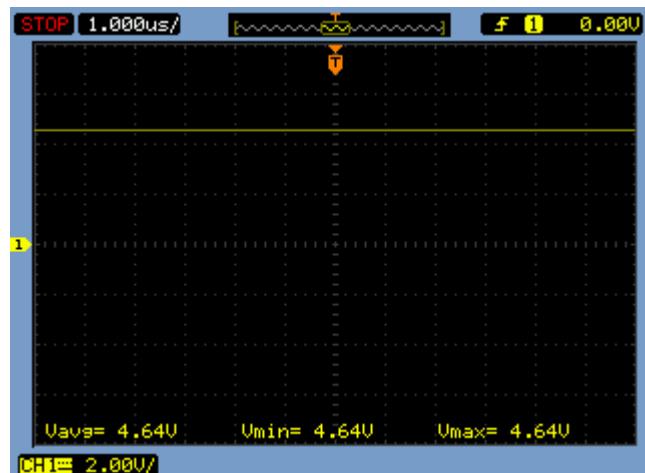
1. Der placeres en kop i en afstand af 4 cm fra sensoren.
2. Der placeres ingen kop foran sensoren. Der skal minimum være 30cm fra sensor til et objekt.

Forventet output

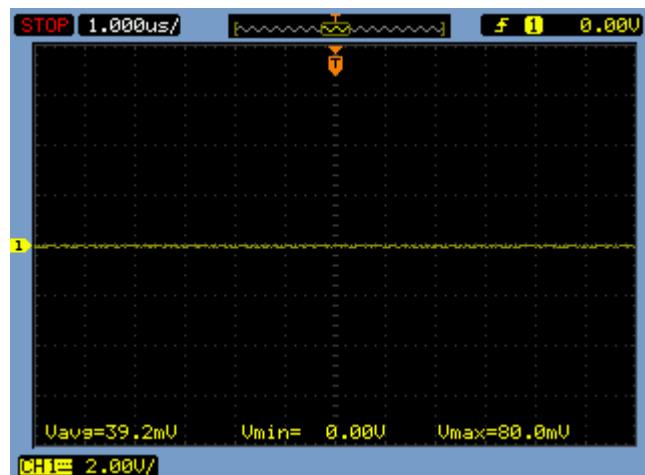
1. Der forventes målt spænding på katoden af zenerdioden(D1) der er forbundet på "4" på 74HC14 på +5VDC(-0,5V).
2. Der forventes målt spænding på "4" på 74HC14 på 0VDC($\pm 0.5V$).

Målinger

1. Oscilloskop billede af test1 - kop foran sensor.



2. Oscilloskop billede af test2 - ingen kop foran sensor.



Resultat

1. Der måles en spænding på 4,64VDC. Dette er inden for det forventede resultats interval - **OK!**
2. Der måles en spænding på 39,2mVDC. Dette er inden for det forventede resultats interval - **OK!**

Ved gennemførsel af enhedstesten blev det desuden fastlagt at sensoren kan detektere ned til en afstand på ca. 1,5cm

6.3.4. Vanddosering

Ekstern grænseflade

Signal	Beskrivelse
Digital* input	Digital signal der signalerer om magnetventilen skal være åben eller lukket.
Fysisk output	Stillingen på magnetventilen.

Intern grænseflade

Signal	Beskrivelse
+12VDC	+12VDC spændingen der skal trække magnetventilen.
GND	Forbindes til drain på IRFZ24N.

Enhedstest

Der er to test-scenarier i denne enhedstest.

1. Vand lukkes gennem magnetventilen.
2. Vand lukkes ikke igennem magnetventilen.

Overordnet opstilling

Overordnet opstilling for test scenariet.

Signal	Fra pin	På enhed	Til pin	På enhed
+12VDC	Rød, CH1	Spændingsgenerator	Katode	1N4007
GND	Sort, CH1	Spændingsgenerator	3(drain)	IRFZ24N
GND	Sort, CH1	Spændingsgenerator	Sort, CH2	Spændingsgenerator
+5VDC	Rød, CH2	Spændingsgenerator	1(gate)	IRFZ24N
Magnetventil 1	Katode	1N4007	Sort	Magnetventil
Magnetventil2	2(source)	IRFZ24N	Rød	Magnetventil

Forventet output

Der er 2 mål i denne enhedstest.

1. Ved påtrykt +5VDC på gaten af IRFZ24N skal der kunne passere vand gennem magnetventilen.
2. Ved påtrykt 0V på gaten af IRFZ24N skal der ikke kunne passere vand gennem magnetventilen.

Resultat

Det er ikke muligt at foretage fysiske målinger på resultaterne af disse tests. Vi har derfor dokumenteret kredsløbets funktionalitet ved at filme overgangen fra digitalt lav til digitalt høj på gate på IRFZ24N.

1. Det ses at der er gennemgang for vand - **OK!**
2. Det ses at der ikke er gennemgang for vand - **OK!**

6.3.5. PSU

Ekstern grænseflade

Signal	Beskrivelse
18Vrms+	AC forsyningssignal input til kredsløbet.
18Vrms-	AC forsyningssignal til input kredsløbet.
GND	GND output.
+12VDC	12VDC output.
+5VDC	5VDC output.

Enhedstest

Der er 2 test-scenarier for denne enhedstest.

1. Der skal genereres +5VDC og +12VDC spændinger.
2. +12VDC skal indkobles efter +5VDC.

Overordnet opstilling

Overordnet opstilling for test scenariet.

Signal	Fra pin	På enhed	Til pin	På enhed
18Vrms+	Rød	Transformer	"sinus1"	DB104
18Vrms-	Sort	Transformer	"sinus2"	DB104
+12VDC	14	EDR201A05	CH1, rød	Oscilloskop
+5VDC	3	LM7805	CH2, rød	Oscilloskop

GND	-	DB104	CH1, sort	Oscilloskop
GND	-	DB104	CH2, sort	Oscilloskop

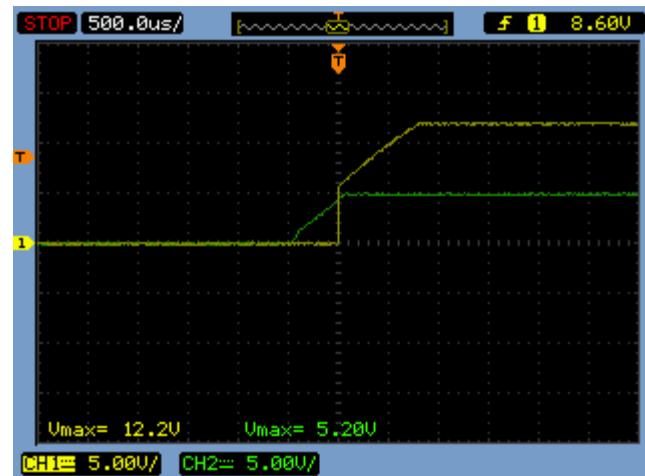
Forventet output

Der er 3 mål i denne enhedstest.

1. Der måles en spænding på +12VDC ($\pm 0,5$ VDC).
2. Der måles en spænding på +5VDC ($\pm 0,5$ VDC).
3. Det skal kunne aflæses at de +12VDC først indkobles efter de +5VDC er blevet konstant.

Målinger

1. Billede af indkobling af powersupply. Det ses at 12VDC først bliver aktiveret efter at de 5VDC er blevet konstant.



Resultat

1. Der måles spænding på 12,2VDC på CH1. Dette er indenfor intervallet 11,5V-12,5V - OK!
2. Der måles spænding på 5,2VDC på CH2. Dette er indenfor intervallet 4,5V-5,5V - OK!
3. Det ses at 12VDC spændingen på CH1 først kan måles efter 5VDC spændingen er blevet stationær - OK!

6.3.6. Motordriver

Ekstern grænseflade

Signal	Beskrivelse
CTC input	PWM signal der bestemmer motorens omdrehningshastighed.
Mekanisk output	Outputtet vil være en mekanisk bevægelse.

Intern grænseflade

Signal	Beskrivelse
+12VDC	DC forsyning til kredsløbet
+5VDC	DC forsyning til kredsløbet
GND	-

Enhedstest

Der er to test-scenarier til denne enhedstest.

1. Input med lav frekvens (ca. 10 Hz).
2. Input med høj frekvens (ca. 1k Hz) – svarende til hastigheden der doseres med.

Overordnet opstilling

Motor driveren opstilles som beskrevet i dokumentationen

Forventet output

Der er to mål i denne enhedstest.

1. Det forventes at motoren vil rotere langsomt.
2. Det forventes at motoren vil rotere i passende hastighed, lig doseringshastighed.

Resultat

Målinger fra testen ikke ikke dokumenteret, da disse er for udefinerbare. Resultatet af enhedstesten blev som forventet, hvor motorens omdrejningshastighed steg i takt med et højere frekvens input.

6.3.7. Vandniveausensor

Ekstern grænseflade

Signal	Beskrivelse
Digital* out-put	Digital signal der signalerer hvorvidt sensoren detekterer vand eller ej.
Fysisk input	Input fra sensorens 2 målepunkter

Intern grænseflade

Signal	Beskrivelse
+5VDC	Forbindes til 1k modstand der går ind på collector benet på NPN transistoren ZTX605.
GND	Forbindes til emitter benet på NPN transistoren ZTX605.

*Beskrivelse af "digital" findes i systemarkitekturen.

Enhedstest

Der er 2 overordnede test-scenarier:

- Detektering af vand
- Detektering af ikke-vand

Overordnet opstilling

Overordnet opstilling for begge test scenarier.

Signal	Fra pin	På enhed	Til pin	På enhed
+5VDC	CH1, rød	Spændingsgenerator	R1	Q1
GND	CH1, sort	Spændingsgenerator	3	Q1
+5VDC	S1_IN1	-	Rød	S1
+5VDC re-tur	S1_IN2	-	Brun	S1

For at detektere vand nedsænkes de to målepunkter i vand.

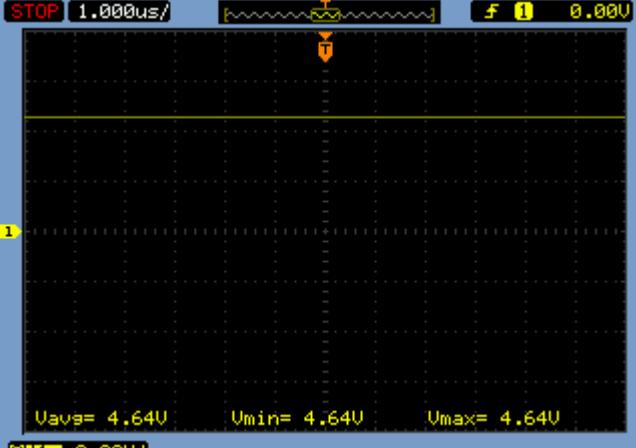
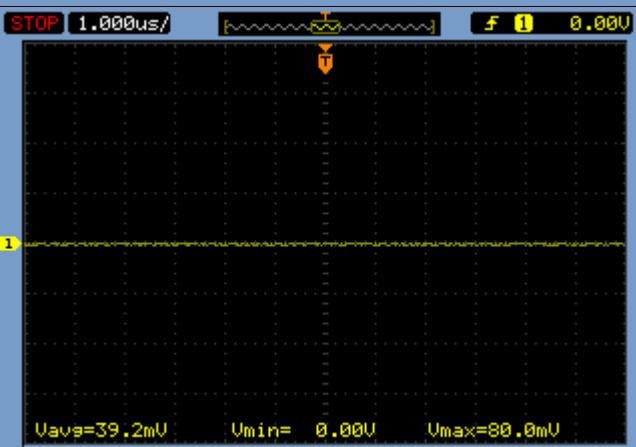
For at detektere ingen vand trækkes en målepin op af vandet, mens den anden forbliver deri.

Forventet output

Ved detektering af vand forventes +5VDC($\pm 0,5$ V).

Ved ingen detektering af vand forventes 0VDC($\pm 0,5$ V).

Målinger

Oscilloskop billede af test 1 – detektering af vand	
Oscilloskop billede af test 2 – ingen detektering af vand	

Resultat

Enhedstesten leverede det forventede resultat inden for den givne tolerance.

6.3.8. Varmelegeme

Ekstern grænseflade

Signal	Beskrivelse
Digitalt input	Det digitale input toggler stikkontaktens state, og tænder/slukker derved for varmelegemet
Fysisk output	Output fra denne komponent eksistere som varme til at opvarme vandet i vandbeholderen.

Intern grænseflade

Signal	Beskrivelse
+12VDC	DC forsyning til fjernbetjening
+230VAC	AC forsyning til varmelegeme
GND	-

Enhedstest

Der er to test-scenarier i denne enhedstest.

1. Varmelegemet er tændt.
2. Varmelegemet er slukket.

Overordnet opstilling

Opstillingen er gældende for begge test-scenarier.

- Varmelegemet kobles i den fjernstyret stikkontakt, der tilsluttes 230VAC.
- Fjernbetjening forsynes med +12VDC og GND.
- +5VDC / 0VDC tilsluttes på fjernbetjenings pin 5.

Forventet output

Når fjernbetjeningen modtager digitalt høj/lav toggler stikkontaktens state, og derved tændes/slukkes der for varmelegemet.

Resultat

Målingerne for testen er ikke dokumenteret da disse for udefinerbare. Testen gav de forventede output, da det kunne registreres at stikkontakten slåes til og fra – og varmelegemet derved tændes og slukkes.

6.3.9. Kortlæser

Ekstern grænseflade

Kortlæseren forbindes med signalerne Rx, Tx og GND til tttyo0-porten på DevKit. Der startes en test-applikation, som skriver og læser til filen /dev/tttyo0.

Intern grænseflade

Signal	Beskrivelse
+5VDC	DC forsyning til kredsløbet
GND	-

Enhedstest

Der er to test-scenarier til denne enhedstest.

1. Der placeres et adgangskort foran kortlæserens antennen
2. Der placeres ikke et adgangskort foran kortlæserens antennen

Forventet output

Der er to mål i denne enhedstest.

1. Det forventes at applikationen printer kortnummeret på konsollen.
2. Det forventes at applikationen ikke udskriver noget.

Test

Resultat for test-scenarie 1.	<pre>root@beagleboard:~# ./target Opening port... Port open! 3 file description Done opening! Nyt kort! aa 0 6 0 0 4 ae 29 cd 48 bb 0 0 0 0 Nyt kort! aa 0 6 0 0 4 ae 29 cd 48 bb 0 0 0 0 Nyt kort! aa 0 6 0 0 4 ae 29 cd 48 bb 0 0 0 0 Nyt kort! aa 0 6 0 0 4 ae 29 cd 48 bb 0 0 0 0</pre>
-------------------------------	--

Resultat

Des ses af ovenstående billede af kortlæseren agerer som forventet ved test-scenarie 1. Der er foretaget test af scenarie 2, men denne er ikke dokumenteret med billede.

6.4. HW integrationstest

Denne test case beskriver en række testscenarier for den samlede integration af hardware komponenter. Disse testcases tager alle udgangspunkt i UC2 køb af kaffe. Formålet med testen er at sikre at PSoC'en modtager de rigtige sensor input, kontrollere aktuatore korrekt, samt sender de rigtige spi beskeder. Da PSoC'en er slave-enhed, anvendes spi driveren på devit til validering af spi beskeder. Hertil anvendes en terminal der konstant læser fra driveren.

6.4.1. Test 1

Step	Handling	Forventet observation/resultat	Resultat
1	Koppen indsættes. PSoC programmet startes fra debuggeren med hardcodede værdier.	Vandvarmeren aktiveres.	Ok
2	Vandet opnår ønskede temperatur	Vandvarmeren slukkes	Ok
3	Ingen	Ønskede mængde pulver doseres	Ok
4	Ingen	Ønskede mængde vand doseres	Ok
5	Koppen fjernes	Der sendes en SPI besked med tilbageværende beholdning	Ok

6.4.2. Test case 2

Step	Handling	Forventet observation/resultat	Resultat
1	PSoC programmet startes fra debuggeren med hardcodede værdier.	Vandvarmeren aktiveres.	Ok
2	Vandet opnår ønskede temperatur	Vandvarmeren slukkes	Ok
3	Ingen	Der sendes spi besked "insæt kop"	Ok
4	Koppen indsættes	Der sendes spi besked "kop indsat"	Ok
5	Ingen	Ønskede mængde pulver doseres	Ok
6	Koppen fjernes	Der sendes spi besked "insæt kop"	Ok
7	Koppen indsættes	Der sendes spi besked "kop indsat"	Ok
8	Ingen	Ønskede mængde vand doseres	Ok
9	Koppen fjernes	Der sendes en SPI besked med	Ok

tilbageværende beholdning

6.4.3. Test case 3

Step	Handling	Forventet observation/resultat	Resultat
1	Kop indsættes. PSoC programmet startes fra debuggeren med hardcodede værdier. Vandbeholderen lidt over minimum.	Vandvarmeren aktiveres.	Ok
2	Vandet opnår ønskede temperatur	Vandvarmeren slukkes	Ok
3	Ingen	Ønskede mængde pulver doseres	Ok
4	Ingen	Vanddosering startes	Ok
5	Ingen	Vanddosering stoppes	Fejl*
6	Ingen	Der sendes spi besked "vandbeholder tom"	Fejl*

*Systemet registrerer ikke vandniveauet. Problemet blev undersøgt nærmere, og det registreres at der kan måles en stor ændring af modstanden i vandet. Denne store ændring af modstand påvirker vandniveausensoren således denne bliver ustabil. Der er ikke fundet en løsning på dette problem på grund af tidspres og prioritering af andre opgaver.

6.4.4. Test case 4

Step	Handling	Forventet observation/resultat	Resultat
1	Kop indsættes. PSoC programmet startes fra debuggeren med hardcodede værdier. Pulverbeholderen lidt over minimum.	Vandvarmeren aktiveres.	Ok
2	Vandet opnår ønskede temperatur	Vandvarmeren slukkes	Ok
3	Ingen	Der sendes spi besked "Pulverbeholder tom"	Ok

7. Pulverdispenser

Som led i projektet var der brug for at dosere én eller flere typer kaffepulver.

Da gruppen ønskede at udarbejde et færdigt og funktionelt produkt, blev der afsat ressourcer til at finde en god løsning.

Den første løsning der blev diskuteret var at bruge en "snegl", altså en spiral som ved at blive roteret trak pulver fra bunden af en pulverbeholder.

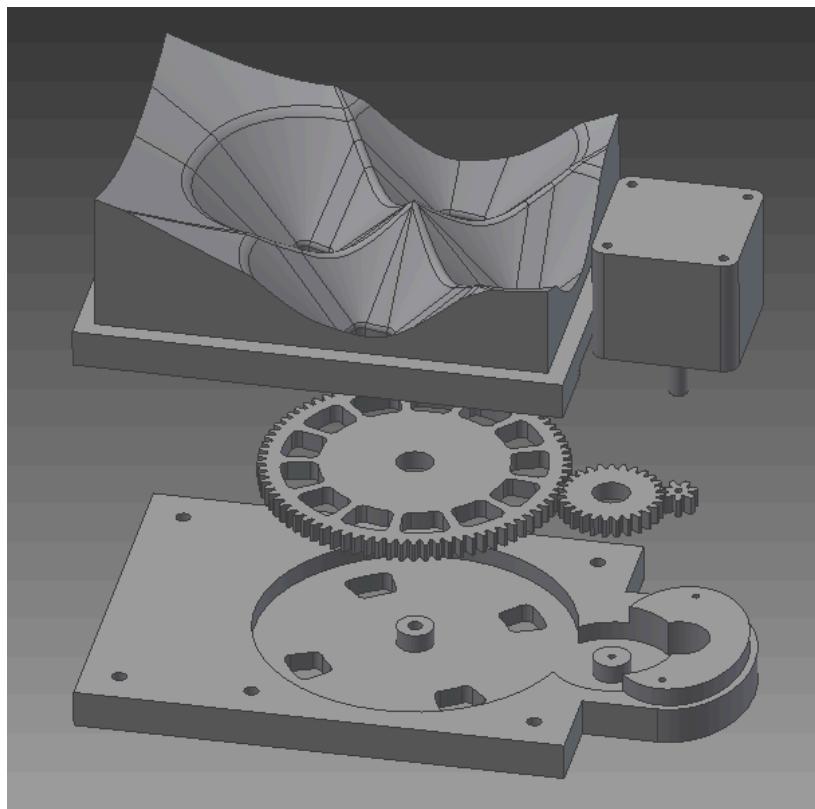
Udover at bare at kunne dosere kaffepulver var det også til en hvis grad vigtigt at kunne gøre det med en nogenlunde præcision. Doseringen en snegl kunne tilbyde er afhængig af pulverkonsistens, fugt og lignende og det kunne derfor blive yderligere behov for at implementere en strain-gauge til løbende at holde øje med vægtdifferencen på beholderen for at vide hvor meget der var doseret.

Efter et mange overvejelser, og med inspiration fra forskellige patentdatabaser, blev en innovativ løsning valgt og resultatet er beskrevet i dette afsnit.

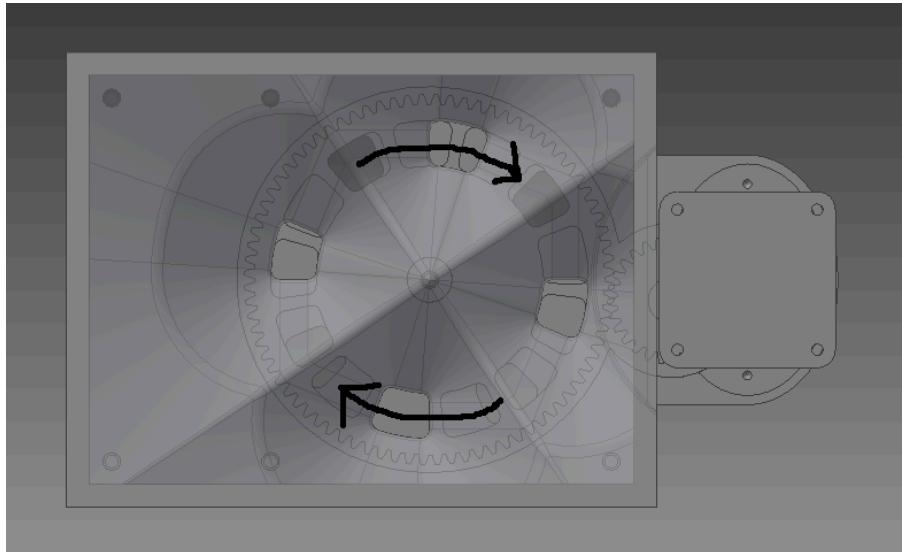
Denne løsning var kort fortalt en roterende, gennemhullet skive hvor hvert hul løbende blev fyldt op og tømt igen.

For at få en lineær dosering på hele omgangen af dette hjul har det et ulige antal huller hvorimod der et lige antal ind og udgange hertil. Herved vil der altid kun være ét hul hvor der er ophold i doseringen af gangen.

FIGUR 1 og 2 illustrerer hvordan doseringsmekanismen er realiseret.



Figur 45 - Exploderet illustration af pulverdispenser

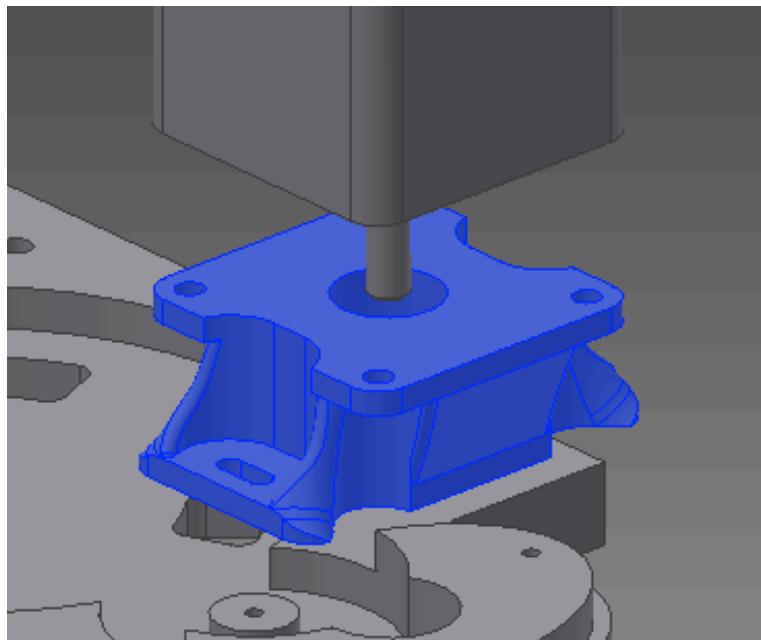


Figur 46 - Doseringmekanisme set i profil

Store dele af mekanismen er tegnet til, og fremstillet på, en 3D printer. Denne fremstillingsmetode har gjort det muligt at eksperimentere med komplekse former og hurtigt at gå fra design til virkelighed.

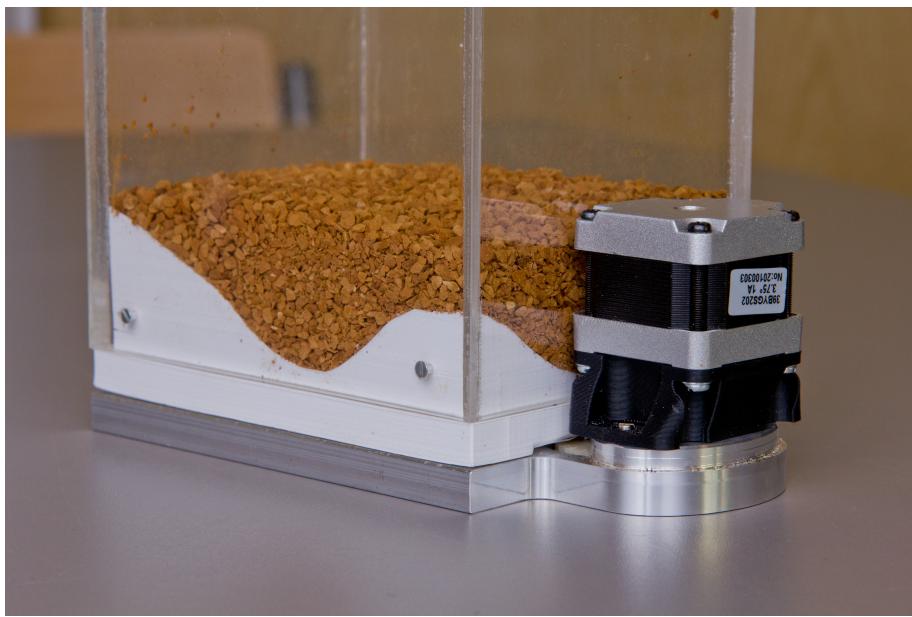
Bundpladen, der holder motor, tandhjul og tragt, er i slutproduktet blevet fremstillet af aluminium da der her er krav til at overfladen er helt plan. Af samme årsag blev tandhjulene også printet tykkere end der var behov for, for derefter fjerne den overskydende plast i en drejebænk.

For at imødegå de relativt svage stepmotorer der i første omgang var indkøbt er mekanismen meget lavgearet. Det viste sig desværre ikke at være tilstrækkeligt og nye motorer blev købt ind. Da disse havde andre dimensioner blev et konverteringsbeslag fremstillet som set i FIGUR 3.

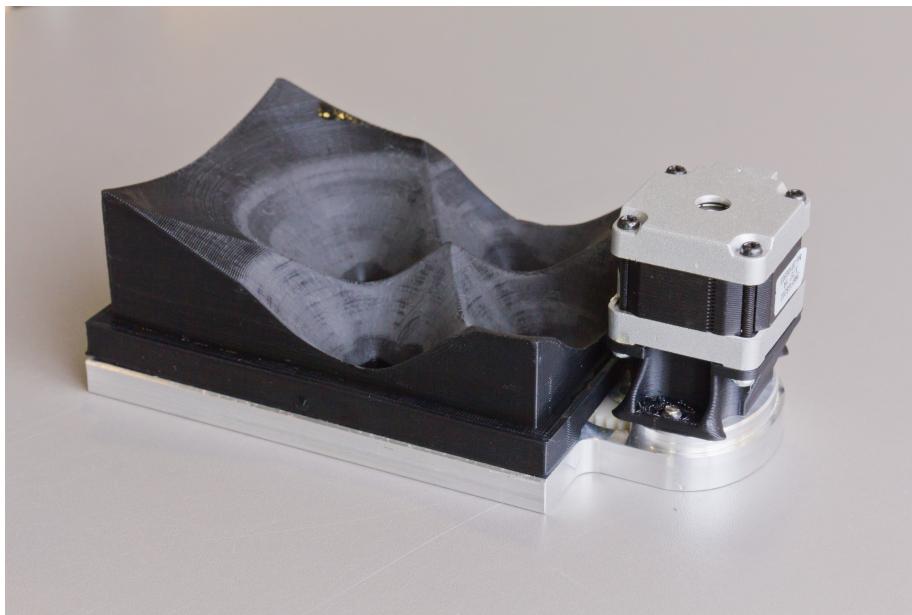


Figur 47 – Motorbeslag

Den færdige dispenser, med og uden beholder, ses i FIGUR 4 og 5.



Figur 48 - Færdig dispenser med beholder



Figur 49 - Færdig dispenser uden beholder

8. Bilagsliste

Source code

- Spi-driver
- Psoc-kode
- ProKaff
- Booting

PCB

- Mainboard-top
- Mainboard-bund
- Antenne
- Kortlæser (mangler kl. 22.48 – Skal findes i GIT historik)

Test applikationer

- Gpio-test
- Interrupt-test
- Psoc-test
- CardReader-test

Datablade

- Pt100 amplifier schematic
- Pt100
- MFRC531
- AN077925
- RFID- COMPROTOCOL
- RDM880-Spec

- A3967-Datasheet
- BC517 NPN Darlington transistor
- Brokobling – DB104
- Diode – 1N4007
- Mosfet – IRFZ24N
- Schitt trigger – 74HC14N
- Sensor – SHARP GP2Y0A1SK0F