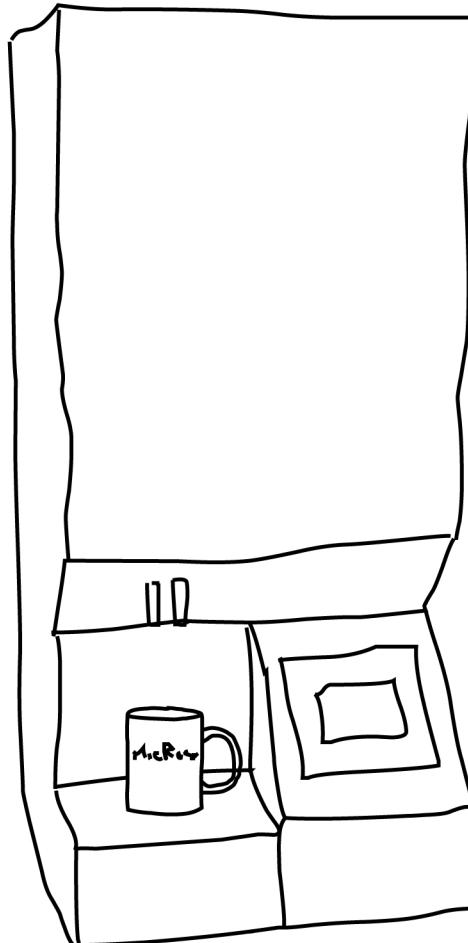


MoccaMonster3000

Rapport

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

Resume

Denne rapport beskriver semesterprojektet for 3. semester på IHA, retning EIT. Problemstillingen omhandler design og implementering af en kaffeautomat til større institutioner. Dette søges realiseret gennem et embedded system der kan integreres med institutionens eksisterende IT-systemer. Det samlede produkt vil indeholde en eller flere kaffeautomater, en database med brugeres konti og kontoinformationer, samt en webfront hvorfra brugeren optanke og administrere sin konto.

Produktet er udviklet med en PSoC3 til styring af de sensorer og aktuatorer, der kræves for at brygge kaffe efter brugerens ønske. Interaktionen med systemet sker gennem en grafisk brugergrænseflade, der er implementeret på et DevKit8000. Brugerne får adgang til grænsefladen ved at benytte deres adgangskort til institutionen. Denne løsning er en medvirkende faktor til at systemet adskiller sig fra andre kaffeautomater.

Designprocessen er inspireret af V-modellen, der sikrer at der løbende testes på færdige komponenter. Udviklingsprocessen er gennemført ved brug af ASE-modellen, hvor der arbejdes i iterationer. Ved brug af disse modeller opnås et gennemarbejdet system.

Abstract

This abstract describes the 3rd semester project at ASE, with EIT as the field of study. The thesis includes a design and implementation of a coffee vending machine for larger institutions. This is to be realized in an embedded system, which can be integrated with the preexisting IT-systems of the institution. The finalized product will include a database, which contains user account information, and a website where users are able to manage their account.

The product is developed using PSoC3 for controlling sensors and actuators, which are required to brew coffee according to the needs of the user. Using a graphic interface, implemented on a DevKit8000, the user is able to interact with the system. The users gain access to the interface using their keycard, to the institution. This solution separates the system from other coffee vending machines.

The design process is inspired by the V-model, which guarantees ongoing tests of finished components. The development process is inspired by the ASE-model, in which work is done in iterations. Using these models an elaborate system is accomplished.

Indhold

Resume.....	1
Abstract.....	1
Indhold.....	2
1. Indledning	4
2. Opgaveformulering	5
3. Projektafgrænsning	6
4. Systembeskrivelse	7
5. Krav	8
5.1. Aktørbeskrivelse.....	8
5.2. Use Case beskrivelse.....	9
5.2.1. Use Case 1 Opret konto.....	9
5.2.2. Use Case 2 Køb af kaffe.....	9
5.2.3. Use Case 3 Administrer konto.....	9
5.2.4. Use Case 4 Vedligeholdelse af systemet.....	9
6. Projektgennemførsel	10
6.1. ASE modellen	10
6.2. V-Modellen.....	11
6.3. SysML	12
6.4. Projektstyring	13
7. Systemarkitektur	14
7.1. Blokidentifikation for kaffearautomat.....	14
7.1.1. Blokbeskrivelse for kaffearautomat	15
7.1.2. Allokering af logisk funktionalitet	16
8. Design og implementering af software	17
8.1. Overordnede softwaremoduler	17
8.2. Design af Brugergrænseflade.....	18
8.2.1. Overordnet sekvensdiagram.....	18
8.2.2. Problemidentifikation.....	19
8.2.3. Klasseidentifikation	21
8.2.4. Metodeidentifikation	22
8.2.5. Uddybende kontrolklasse beskrivelse	23
8.3. Implementering af Brugergrænseflade	24
8.3.1. Indledende implementeringsovervejelser.....	24
8.3.2. Generelt om implementering.....	24
8.3.3.....	24
8.3.4. Grafiske grænseflader.....	25
8.3.5. Pakkeinddeling.....	26
8.4. Databaser og webfront	27
8.4.1. Konteringsdatabase	27
8.4.2. ADInterface	27
8.4.3. Webfront.....	27
8.5. Design af programmet til PSoC	28

8.5.1. Statebeskrivelse	29
8.6. Implementering af programmet til PSoC	30
8.6.1. Måling af sensorer.....	30
8.6.2. Styring af aktuatorer.....	30
8.7. Kommunikation mellem DevKit og PSoC	31
8.7.1. Design af spi drivere	31
8.7.2. Implementering af SPI driver	31
9. Design og implementering af hardware.....	32
9.1. Indledende designovervejelser.....	32
9.2. Overvejelser omkring sensorer.....	33
9.2.1. Temperatursensor.....	34
9.2.2. Vandniveausensor.....	35
9.3. Vanddosering.....	36
9.4. Doseringsmekanisme.....	37
9.5. Kortlæser.....	38
10. Resultat og diskussion	39
11. Udviklingsværktøjer	41
12. Opnåede erfaringer	42
13. Fremtidigt arbejde	43
14. Konklusion	44
15. Litteraturliste	45

1. Indledning

På de fleste større institutioner står der den klassiske kaffeautomat, hvor køberen indsætter en 5'er, og efter et minut kan tage en kop kaffe. Selvfølgelig kan man købe kaffeautomater, der leverer en udmærket kop kaffe, men en hurtig googlesøgning viser at udbuddet af automater fra førende leverandører ikke kan integreres med køberens allerede eksisterende adgangskort og brugerdatabase.

Formålet med dette projekt, er at udvikle denne type kaffeautomat; en kaffeautomat hvor brugeren kan købe en delikat kop kaffe, let og uden brug af kontanter. Altså at udvikle et produkt, der er integreret på den institution, den står på.

Produktet skal bestå af en webfront og en række kaffeautomater. Personer tilknyttet institutionen skal selv kunne oprette sig i systemet og bruge systemet med samme brugernavn, adgangskode og adgangskort, som de i forvejen bruger på institutionen.

Brugerne kan oprette sig i systemet ved at køre deres adgangskort forbi kortlæseren på en af institutionens kaffeautomater. Hvis brugeren i forvejen er oprettet i systemet, kan der med adgangskortet købes kaffe i alle automater på institutionen.

Til hver bruger kan der tilknyttes en kaffekonto, hvorfra kaffen betales. Denne konto kan administreres og optankes af brugeren gennem webfronten.

At alle på institutionen på forhånd har et gyldigt betalingsmiddel, vil styrke interessen for at bruge institutionens kaffeautomater.

Med udgangspunkt i brugerens behov skal der opstilles en række use cases, der beskriver vedkommendes interaktion med systemet. Det er disse use cases der skal udgøre kravspecifikationen for systemet. Systemdesignet skal udarbejdes ved en gradvis nedbrydning til specifikke hardware- og softwaremoduler, der i sidste ende skal udgøre byggestenene for produktet.



2. Opgaveformulering

Opgaven i dette projekt er at udvikle en kaffeautomat, der kan implementeres på diverse institutioner. Hver kaffeautomat skal have en touchskærm, hvor brugeren kan indtaste sin bestilling, hvoraf efter bestillingen bliver udført automatisk.

Dette medfører at det færdige produkt skal kunne:

- Dosere en forudbestemt mængde kaffepulver
- Opvarme og dosere en forudbestemt mængde vand
- Modtage brugerens bestilling på en touchskærm
- Detektere hvorvidt brugeren har placeret en kop i automaten før doseringen af vand eller pulver begyndes
- Detektere hvorvidt kaffeautomaten er løbet tør for vand eller kaffepulver

For at gøre produktet salgsbart, bør systemet kunne modtage betaling for kaffen. Dette kan gøres ved at tilknytte en personlig konto til brugeren, der kan optankes og administreres gennem en webfront. Optankning kan foregå igennem PayPal eller en lignende service. Kaffeautomaterne kan endvidere udvikles med henblik på at brugeren kan bruge et ID-kort som betalingsmiddel.

Visionen bag projektet er at udvikle et system der er velintegreret med den pågældende institutions eksisterende IT-services. For at forbedre købsoplevelsen, kan systemet implementeres så login til webfronten, sker med samme brugernavn og adgangskode som brugeren har fået tildelt af den pågældende institution. Dertil kan automaten udvikles således at køb af kaffe sker ved hjælp af de ID-kort brugerne i forvejen bruger på institutionen.

I en færdig prototype, bør der være fokus på kvaliteten af den bryggede kaffe. Derfor kan systemet videreudvikles til at kunne brygge kaffe på friskværnede bønner. Dertil bør brugeren have mulighed for at vælge mellem forskellige kaffetyper, samt mængde og styrke.

3. Projektafgrænsning

Fra IHA's side er der på forhånd defineret nogle krav til projektets indhold, hvilket indebærer at:

- Der skal indgå minimum én sensor og én aktuator
- Der skal indgå brugerinteraktion
- Der skal bruges et DevKit8000 og et PSoC3 starter kit
- Kommunikationen mellem DevKit8000 og PSoC3 starter kittet skal være en seriel kommunikation såsom SPI, I2C, RS232 eller lignende

Ud fra ovenstående udvikles et system, som på baggrund af brugerens ønsker, kan brygge en kop kaffe.

Da dette projekt gennemføres på IHA, bliver prototypen udviklet til at fungere med IHAs adgangskort og brugerdatabase. Dog skal systemet udvikles, så det med begrænsede modifikationer kan implementeres på lignende institutioner.

Da IHA ikke tillader arbejde med 230 V i forbindelse med studiet skal en færdig løsning til styring af vandvarmer anskaffes.

Det vil blive efterstræbt at implementere en fuldt funktionsdygtig prototype af systemet. Dog vil der med ovennævnte krav i mente være delelementer, der ikke kan implementeres som ønsket.

Der vil i projektet blive implementeret et system bestående af en kaffeautomat, en database og en webfront.

Ved fuld udvikling skal systemet kunne dosere friskkværnede kaffebønner samt være tilsluttet fast vandforsyning. Desuden skal automaterne kunne opdatere sig selv i forbindelse med revisioner af softwaren. Disse funktioner vil dog af tidsmæssige årsager blive nedprioriteret.

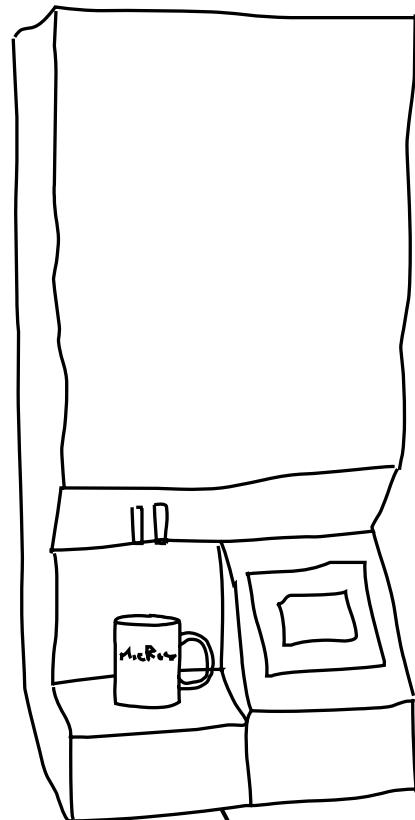
4. Systembeskrivelse

Ud fra visionen beskrevet i opgaveformuleringen er det besluttet at det endelige system består af én webfront, en konteringsdatabase samt en eller flere kaffeautomater. Systemet kan implementeres på diverse institutioner og forudsætter, at institutionen har en brugerdatabase, hvor informationer om personer tilknyttet institutionen er gemt. Derudover forudsættes det at hver bruger har et personligt brugernavn, adgangskode og adgangskort.

Institutionens brugere kan oprette sig i systemet ved at scanne sit adgangskort på en af de opsatte kaffeautomater. Derefter skal brugeren indtaste sit brugernavn, hvorefter systemet gemmer brugeren i konteringsdatabasen. Brugeren har nu adgang til både at benytte institutionens kaffeautomater, og systemets webfront. På webfronten logges ind med samme brugeroplysninger, som på institutionen øvrige IT systemer. For at købe kaffe i automaterne, skal brugeren indsætte penge på sin konto gennem webfronten. Når brugeren køber en kop kaffe, trækkes pengene automatisk.

Hvis brugeren ønsker at administrere sin konto, kan det til enhver tid gøres gennem webfronten. Brugeren har her mulighed for bla. at optanke, spærre, genåbne eller lukke sin konto.

Kaffeautomaten skal udadtil have en brugergrænseflade i form af en touchskærm. Brugergrænsefladen skal guide brugeren gennem henholdsvis oprettelse af konto, og køb af kaffe. Hvis en fejl opstår i systemet, eller systemet løber tør for kaffepulver eller vand, skal brugergrænsefladen sende en mail til en operatør, og vise en fejlmeldelse på touchskærmen.



5. Krav

Ud fra en analyse af opgaveformuleringen er der udarbejdet en række use cases, der beskriver brugerens interaktion med systemet. Disse use cases fungerer som kravspecifikation, og bruges i en tidlig del af udviklingsfasen til at fastlægge systemets funktionalitet. De fulde use cases forefindes i dokumentationen.

5.1. Aktørbeskrivelse

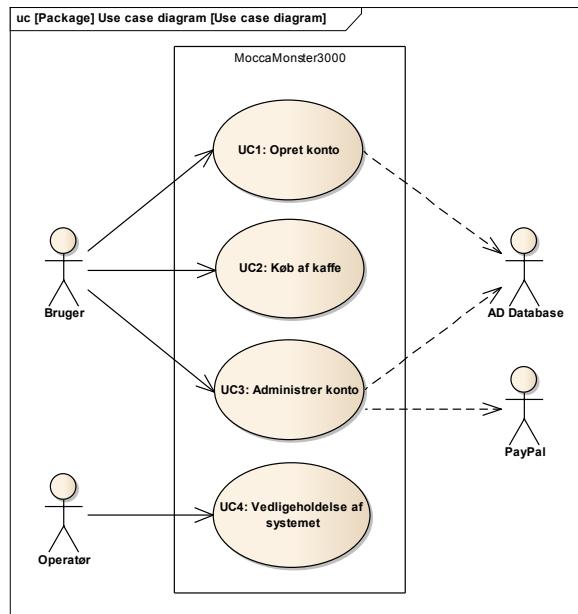
På use case diagrammet ses en række aktører, der interagerer med systemet. Disse er beskrevet i den følgende aktørbeskrivelse.

Brugeren er en primær aktør. Brugeren ønsker at købe kaffe i kaffeautomaterne. For at gøre dette, skal brugeren oprette og optanke en personlig konto.

Operatøren er en primær aktør. Operatøren har til ansvar at vedligeholde kaffeautomaterne. Det er operatøren, der står for rengøring og genopfyldning af forbrugsstoffer i kaffeautomaterne. Dertil skal operatøren udbedre eventuelle fejl i systemet.

AD (Active Directory) databasen er en off-stage aktør. Dette er bruger databasen, som indeholder informationer om de ansatte på den institution, hvor systemet er opsat. Derudover benyttes databasen til at validere brugere i forbindelse med oprettelse og administration af konti, samt ved køb af kaffe.

PayPal er en off-stage aktør. PayPal står for pengetransaktionen ved køb af kaffe.



Figur 1 - Use case diagram

5.2. Use Case beskrivelse

De identificerede use cases er beskrevet i det følgende. Hver use case beskriver et scenario hvor aktører interagerer med systemet.

5.2.1. Use Case 1 Opret konto

For at kunne benytte systemet skal brugeren oprette en personlig konto. Dette gøres ved at brugeren kører sit adgangskort forbi kortlæseren på en af institutionens kaffeautomater og indtaster sit brugernavn på brugergrænsefladen. Systemet henter relevante oplysninger om brugeren i AD databasen og opretter derefter brugerens personlige konto.

5.2.2. Use Case 2 Køb af kaffe

For at købe kaffe i en af automaterne kører brugeren sit adgangskort forbi en af kaffeautomaternes kortlæser. Dernæst indtaster brugeren sin bestilling på brugergrænsefladen og accepterer købet. Efter at brugeren har placeret sin kop i automaten, brygges kaffen og prisen for købet trækkes fra brugerens konto.

5.2.3. Use Case 3 Administrer konto

Brugeren skal have mulighed for at optanke, spærre, lukke eller genåbne sin konto. Dette gøres gennem en webfront, som brugeren kan tilgå med sit brugernavn og adgangskode. Når brugeren ønsker at logge ind på sin konto, validerer systemet de indtastede brugerinformationer med AD databasen. Hvis brugeren ønsker at optanke sin konto gøres dette via PayPal.

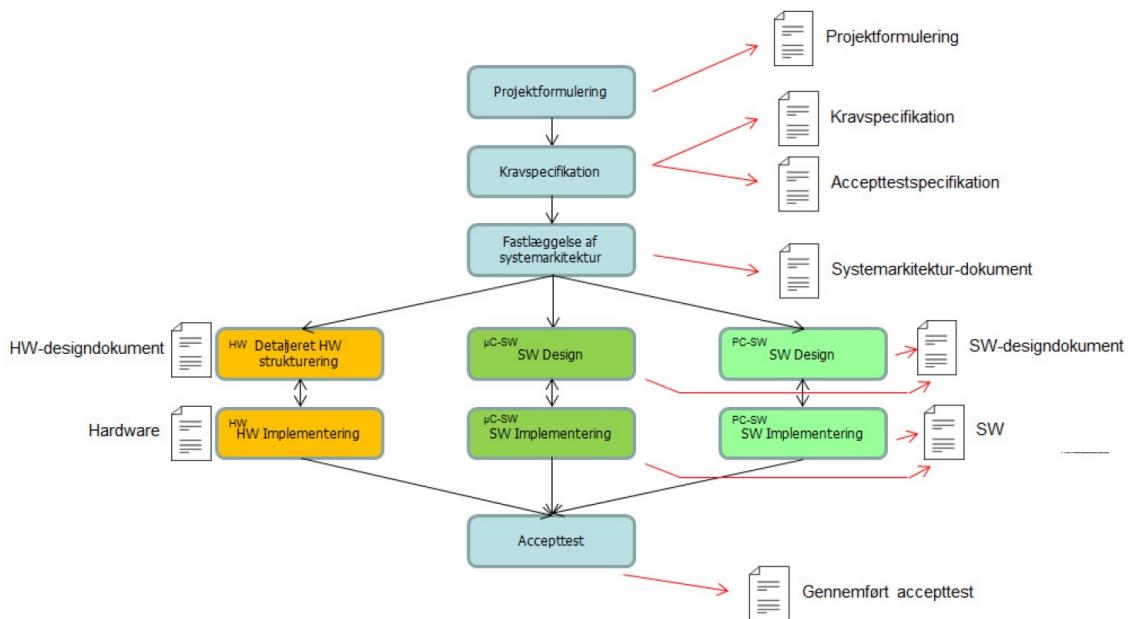
5.2.4. Use Case 4 Vedligeholdelse af systemet

Hvis systemet detekterer at der er brug for vedligeholdelse, låses systemet og der sendes en mail til operatøren. Efter at systemet detekterer at operatøren har udført den nødvendige vedligeholdelse, låses systemet op og er klar til brug.

6. Projektgennemførsel

6.1. ASE modellen

Den primære udviklingsmodel, der er benyttet i dette projekt er **ASE modellen**¹. ASE modellen er en mellemvægtig semi-iterativ udviklingsmodel, der er drevet ud fra use cases. Modellen er opbygget således at udviklerne benytter vandfallsmodellen til at fastlægge en opgaveformulering, kravspecifikation og systemarkitektur, for derefter at designe og implementere de enkelte moduler i iterationer.



Figur 2 - Ase Modellen

Ud fra projektformuleringen specificeres kravspecifikationen som en række use cases. use cases er et værktøj der beskriver diverse aktørers interaktion med systemet. Ved at definere kravspecifikationen ud fra use cases opnås et overblik over hvilke krav der stilles til systemets endelige funktionalitet. Ud fra kravspecifikationen kan systemets accepttest specificeres.

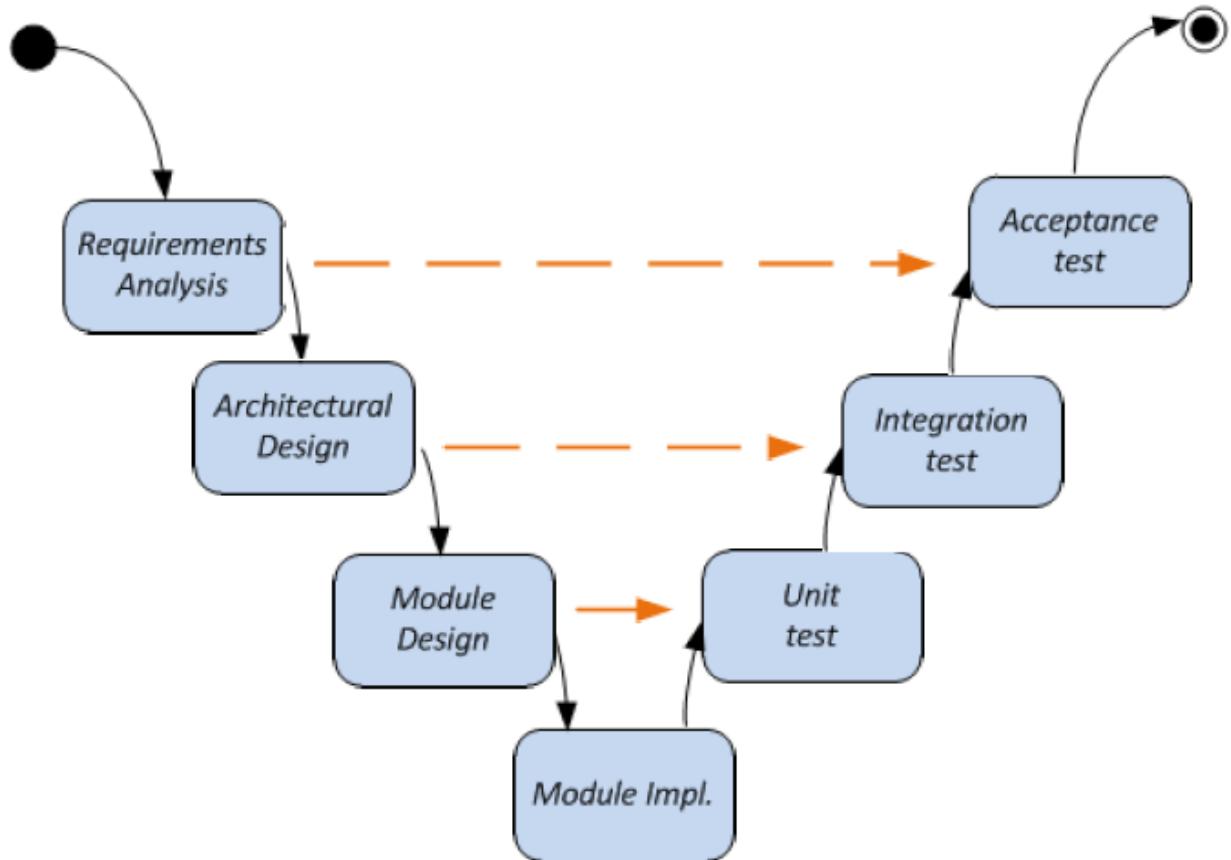
Efter kravspecifikationen er fastlagt, udarbejdes systemarkitekturen. I systemarkitekturen uddeles systemets funktionalitet i moduler og deres grænseflader til resten af systemet bestemmes. Ud fra systemarkitekturen designes systemet, ved at nedbryde det efter funktionalitet, der kan bindes til hardware og software.

¹ Modellen er udviklet på IHA. Der er ingen litterær henvisning, da teorien er baseret på undervisningen.

6.2. V-Modellen

V-modellen(Biering-Sørensen, 1988, s. 175) er en komponentbaseret udviklingsmodel, der beskriver udviklingsfaserne i et projekt. Som nævnt i ASE modellen nedbryder udviklerne gradvist systemet med formålet at specificere nødvendig hardware og software. Ved brug af V-modellen specificeres tests sideløbende med selve udviklingen af systemet.

Ved at teste på forskellige niveauer, sikres at de udviklede delsystemer virker efter hensigten.



Figur 3 - V-Modellen

Den første udviklingsfase er udarbejdelse af kravspecifikationen. Her udvikles der en tilhørende accepttest, der verificerer systemets overordnede funktionalitet. Den næste fase er systemarkitekturen. I denne fase udvikles tests af integrationen mellem implementerede moduler.

Under design og implementering som udgør den sidste fase i udviklingen udføres der løbende enhedstests, af de implementerede moduler.

6.3. SysML

Til at beskrive systemarkitekturen og det detaljerede design for produktet, er der benyttet SysML (Friedenthal, 2008). SysML udspringer af UML, men hvor UML er centreret omkring udvikling af software systemer, bruges SysML i højere grad ved systemer der både involverer software og hardware. Et af de vigtigste argumenter for brug af SysML, er at de fastlagte standarder i sproget medfører en bedre formidling af systemet, hvilket giver udviklerne et større overblik. Desuden er SysML også et stærkt redskab til at formidle systemet for omverdenen.

I dette projekt er der benyttet struktur- og adfærdsdiagrammer til at specificere og dokumentere systemet. De anvendte strukturdiagrammer er **blok definitions diagrammer** (bdd) og **interne blok diagrammer** (ibd).

Blok definitions diagrammerne er anvendt til at dokumentere nedbrydningen af systemet. Interne blok diagrammer er brugt til at beskrive grænseflader mellem diverse delsystemer.

De adfærdsdiagrammer der er benyttet i dette projekt er **sekvensdiagrammer** og **state machine** diagrammer. Disse diagrammer er velegnede til sekventielt at beskrive den logiske funktionalitet i systemet.

Til designet af software er der brugt **applikationsmodeller**. Applikationsmodellen tager udgangspunkt i en **domænemodel**, der beskriver det overordnede systemdomæne. Domænemodellen beskriver systemet ud fra koncepter fundet i de definerede use cases. Ud fra domænemodellen, kan udviklerne definere softwareklasser og deres interaktioner imellem. Dette gøres ved brug af klasse- og adfærdsdiagrammer.

Ved at bruge applikationsmodellen opbygges et softwaremodul af kontrolklasser, domæneklasser samt grænsefladeklasser. Kontrolklasserne har til funktion at koordinere og udveksle data mellem de resterende klasser. Grænsefladeklasser indeholder den funktionalitet, der skal kommunikere med omverdenen, og domæneklasser indeholder funktionalitet der er til brug for det pågældende softwaremodul. Dette er en designmodel, der hjælper udviklerne til at designe et produkt med lav kobling og høj samhørighed.

Hardwaren er designet ud fra et **black box approach**, hvor kravene til de enkelte hardwareenheder er defineret i systemarkitekturen. Ud fra disse krav opstilles enhedstest, der kan verificere om den udviklede hardwareenhed har den ønskede funktionalitet.

6.4. Projektstyring

Dette projekt strækker sig over en længere periode, med sideløbende undervisning, der danner grundlag for udviklingen af produktet. ASE modellen er derfor stærk, da man relativt sent i udviklingsprocessen går i gang med det tekniske design. Under dette projekt har ASE modellen ageret grundlag for, hvordan der er arbejdet.

Administrative roller, så som ordstyrer og referent, blev fastlagt ved første møde. Under hele projektet har der været ugentlige møder med gruppens projektvejleder. Dertil har der minimum været ét ugentligt møde fælles for hele gruppen, hvor det blev diskuteret hvad der var sket siden sidst og hvad der skulle ske til næste uge.

Under fastlæggelsen af systemets overordnede funktionalitet, er der arbejdet fælles. Dermed har alle været med til at bestemme kravene til produktet, samt til at udvikle systemarkitekturen. I denne fase blev de overordnede moduler og deres grænseflader identificeret. Ud fra systemarkitekturen blev der foretaget en risikoanalyse. I risikoanalysen blev det vurderet hvor stor relevans det enkelte modul havde for systemets funktionalitet, samt hvor svært modulet ville være at implementere. Med risikoanalysen i mente blev udviklerne uddelt på de forskellige opgaver. Dette blev gjort ud fra, hvad den enkelte udvikler ønskede at arbejde med og hvem der havde størst forudsætning for at kunne løse den pågældende opgave. Dette gav en arbejdsfordeling, hvor hver udvikler havde et stort engagement i at arbejde med de udfordringer der skulle løses. Efter at arbejdsfordelingen blev fastsat begyndte de egentlige design- og implementeringsfaser. Her blev der arbejdet i mindre grupper, hvor der løbende blev samarbejdet og testet arbejdsgrupperne imellem.

Under denne fase af udviklingsprocessen blev der arbejdet i ugentlige iterationer. Før hver iteration mødtes gruppen og fastslog hvilke opgaver, der skulle løses i den pågældende uge. Dette gjorde sig specielt gældende i den sidste måned af udviklingen, hvor der blev mere tid til projektarbejdet. I den helt afsluttende fase, hvor der blev arbejdet fuld tid på projektet, har der været en øget planlægning af arbejdsopgaver og deadlines. Selvom denne måde at arbejde på er inspireret af Scrum², er der ikke brugt Scrum. Dette valg er taget, da gruppen vurderede, at det ville være sværligt at fastholde alle dele af Scrum i et projekt, hvor der ikke var mulighed for at arbejde fuldtid.

² Se [http://en.wikipedia.org/wiki/Scrum_\(software_development\)](http://en.wikipedia.org/wiki/Scrum_(software_development))

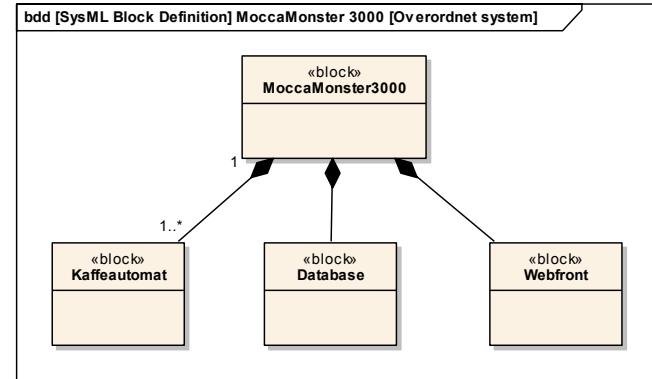
7. Systemarkitektur

I det følgende beskrives arkitekturen for systemet. Systemarkitekturen fungerer som udviklingsramme for videre design og implementering. Det er her at systemets funktionalitet, der er beskrevet i systembeskrivelsen og kravspecifikationen, nedbrydes til overordnede moduler.

Systemet kan illustreres i et bdd bestående af tre moduler. En webfront, en konteringsdatabase samt en til flere kaffeautomater.

På **webfronten** skal brugeren af systemet kunne optanke og administrere sin konto.

Konteringsdatabasen skal indeholde brugeroplysninger, transaktionslogs samt oplysninger om systemet.



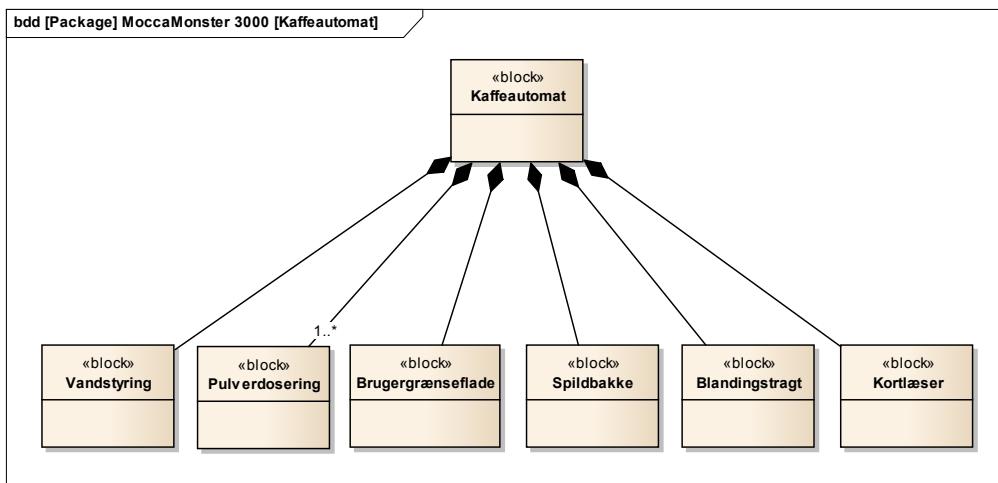
Figur 4 - bdd for systemet

Det er på **kaffeautomaten** at brugeren opretter sig i systemet, og det er her brugeren køber kaffe.

For at kunne opfylde den i kravspecifikationen beskrevne funktionalitet, skal både kaffeautomat og webfront kunne tilgå konteringsdatabasen.

7.1. Blokidentifikation for kaffeautomat

For at muliggøre at brugeren kan oprette sig i systemet og købe kaffe i en af institutionens kaffeautomater, er der en række moduler en kaffeautomat nødvendigvis skal indeholde. Dette indebærer både moduler, der skal stå for interaktionen med brugeren og konteringsdatabasen, samt moduler der kan brygge den bestilte kaffe. De identificerede moduler, der skal opfylde en kaffeautomats funktionalitet er illustreret i følgende bdd.



Figur 5 - bdd for kaffeautomat

7.1.1. Blokbeskrivelse for kaffeautomat

Interaktion mellem brugeren og kaffeautomaten sker gennem **brugergrænsefladen**, og det er derfor nødvendigt at den kan kommunikere med de andre blokke. Når brugeren kører sit adgangskort forbi **kortlæseren**, modtager brugergrænsefladen det skannede kortnummer, og skal kunne sammenholde dette med konteringsdatabasen. Derfor skal brugergrænsefladen være tilsluttet en internetforbindelse.

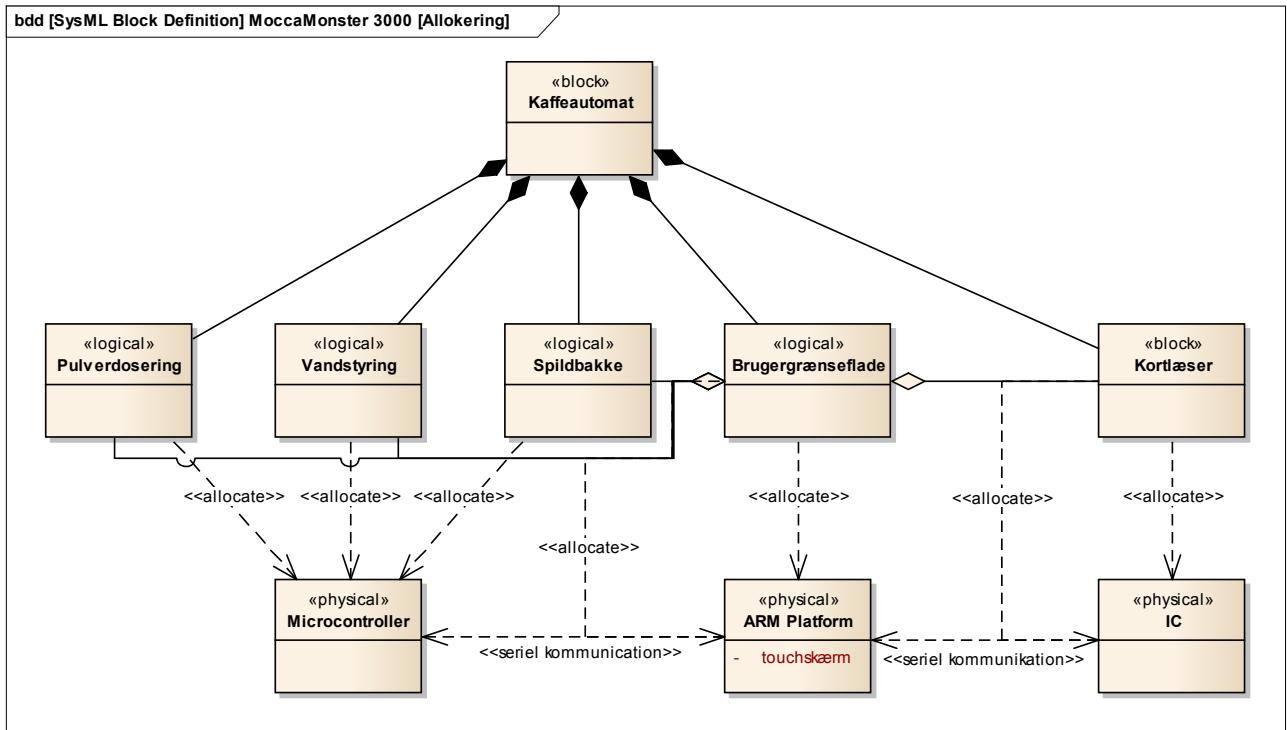
Vandstyringen har ansvaret for at opvarme og dosere den korrekte mængde vand til den kaffe brugeren har bestilt, og føre det ned i **blandingstragten**. Desuden skal vandstyring indeholde en vandniveausensor, der kan detektere om vandbeholderen er tom.

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

Brugeren placerer sin kop på en rist der agerer som låg til **spildbakken**. Over risten skal udmuddingen på blandningstragten placeres. Spildbakken skal både kunne opsamle eventuelt spild fra brygningen, men også detektere hvorvidt brugeren har placeret sin kop på risten. Dette er modvirker at automaten doserer vand og pulver, før der er en kop tilstede.

7.1.2. Allokering af logisk funktionalitet

I kaffeautomaten er der en række blokke der har en logisk funktionalitet, som er allokeret på en række fysiske enheder. Nedenstående diagram viser hvor de logiske blokke er allokeret.



Figur 6 - Allokeringdiagram for kaffeautomat

Det ses på diagrammet at pulverdoseringens, vandstyringens og spilbakvens logiske funktionalitet er allokeret på samme microcontroller. I dette projekt er det besluttet at microcontrolleren skal være af typen PSoC3. Brugergrænsefladen er allokeret på en ARM platform som skal have en touchskærm. I dette projekt er det besluttet at ARM platformen skal være et DevKit8000. Kortlæseren er allokeret på en dedikeret IC der skal vælges ud fra kravspecifikationen. IC'en vælges i hardwaredesignfasen.

Kommunikationen mellem de logiske blokke er allokeret på en seriel forbindelse. Beslutningen angående hvilken kommunikationsstandard der skal benyttes tages i designfasen.

8. Design og implementering af software

I det følgende beskrives det detaljerede softwaredesign for projektet, som specificeret i systembeskrivelsen, kravspecifikationen og systemarkitekturen. Designprocessen, samt hvilke overvejelser og valg der er gjort i forbindelse med design og implementering af softwaren, vil blive gennemgået. Softwarens endelige design og implementering beskrives med hjælp fra modeller, specificerede i projektgennemførslen.

Rapporten tager udgangspunkt i use case 2, og går derfor kun i dybden med denne use case. For en dybdegående beskrivelse af softwaren bag use case 2, samt de resterende use cases, henvises til dokumentationen³.

8.1. Overordnede softwaremoduler

Som det fremgår af systembeskrivelsen inkluderer systemet en form for brugerinteraktion og kommunikation mellem de logiske softwaremoduler.

Brugerinteraktionen sker gennem en brugergrænseflade, som implementeres på DevKittets touch-skærm. Da brugergrænsefladen derudover skal have forbindelse til konterings- og AD databasen.

PSoC'en fungerer som mellemled mellem DevKittet og automatens hardwareenheder. Kommunikationen mellem PSoC og DevKit kræver udarbejdelse af en protokol, der beskriver nødvendige beskeder.

Konteringsdatabasen primære opgave er at parre brugernavne med adgangskort. Ud over dette skal databasen indeholde informationer om brugernes transaktioner, samt informationer om pulvertyper, pris og blandingsforhold.

AD databasen er institutionens egen brugerdatabase, og indeholder informationer om personer tilknyttet institutionen. Systemet benytter denne database til at udlæse informationer om brugere af systemet

Webfronten fungerer som administrationsmodul for brugere af systemet. Det er her man indsætter penge på sin saldo eller administrerer sin konto.

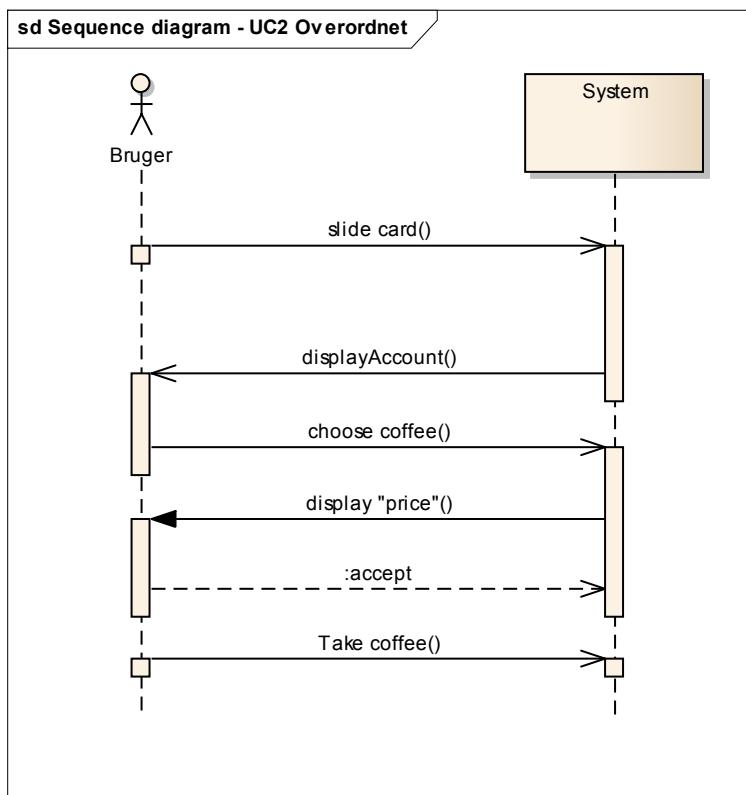
For at kunne aflæse adgangskort, skal der skrives software til at kommunikere med kortlæseren.

³ Se Dokumentation, afsnit 1.3

8.2. Design af Brugergrænseflade

8.2.1. Overordnet sekvensdiagram

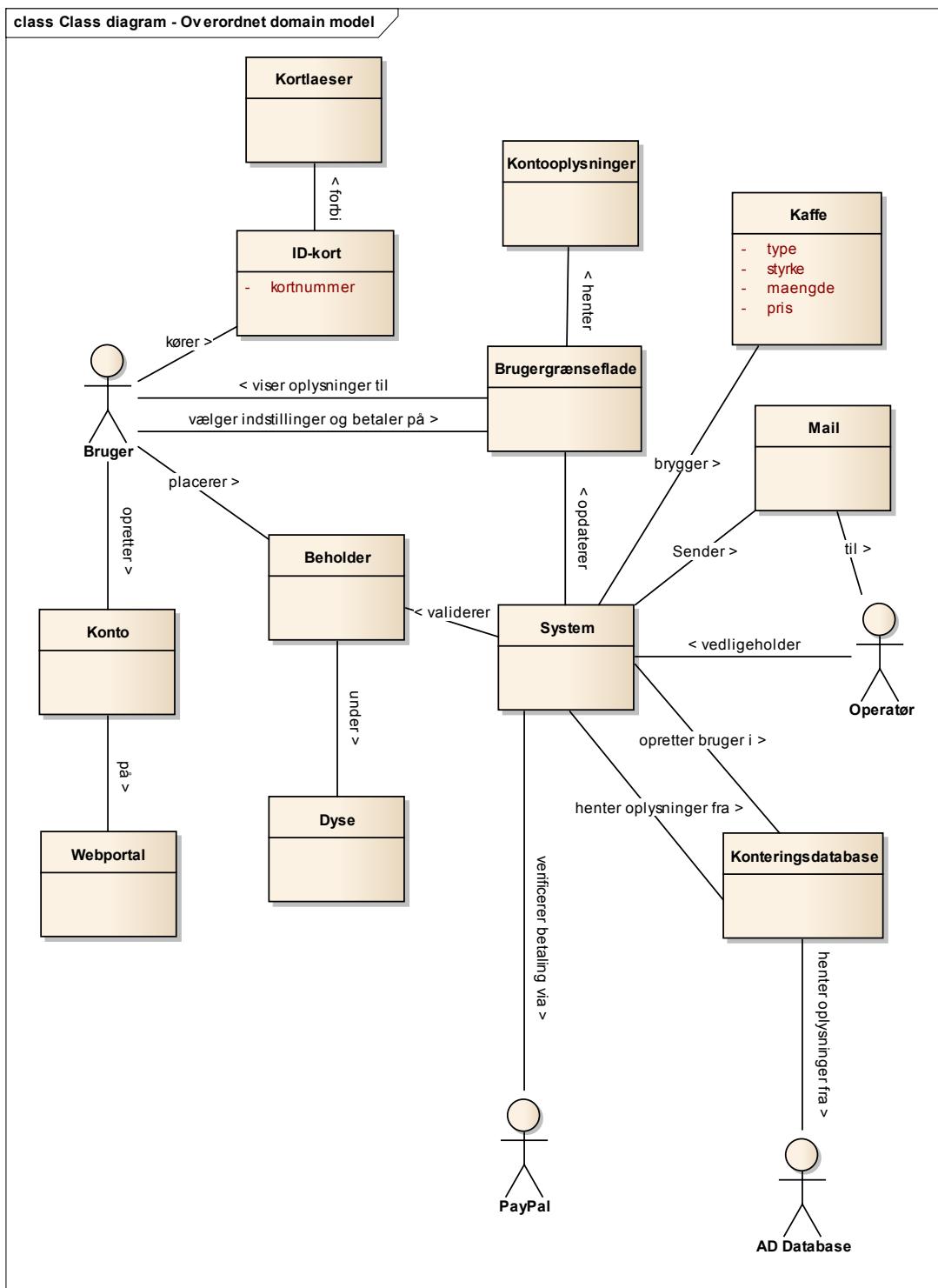
Det overordnede softwaredesign kan beskrives ud fra et sekvensdiagram. Formålet med diagrammet er at vise brugerens interaktion med systemet på en overskuelig måde. Diagrammet er derfor en simpel illustration på systemets adfærd gennem use casens forløb.



Figur 7 - Overordnet sekvensdiagram

8.2.2. Problemidentifikation

I henhold til projektgennemførslen er første skridt i designprocessen, at udarbejde en domæne model. Domæne modellen indeholder konceptuelle klasser, der er baseret på problemer, identificerede ud fra alle use casene. Softwaren skal designes til at løse de problemer.



Figur 8 - domænenmodel

Klasserne defineres ud fra grænseflade- og domæneproblemer. Grænsefladeproblemerne består af de udfordringer der opstår når to dele af systemet skal kommunikere. Domæneproblemerne er de resterende problemer systemet skal løse.

Ud fra domænemodellen defineres følgende problemer:

- Grænseflader
 - Brugergrænseflade til bruger
 - Grænseflade til kortlæser
 - Grænseflade til PSoC'en
 - Grænseflade til databaserne
- Domæner
 - Håndtering af den aktuelle brugers informationer
 - Håndtering af de specifikke parametre for den bestilte kaffe
 - Efter hvilken protokol der skal kommunikeres mellem PSoC og DevKit

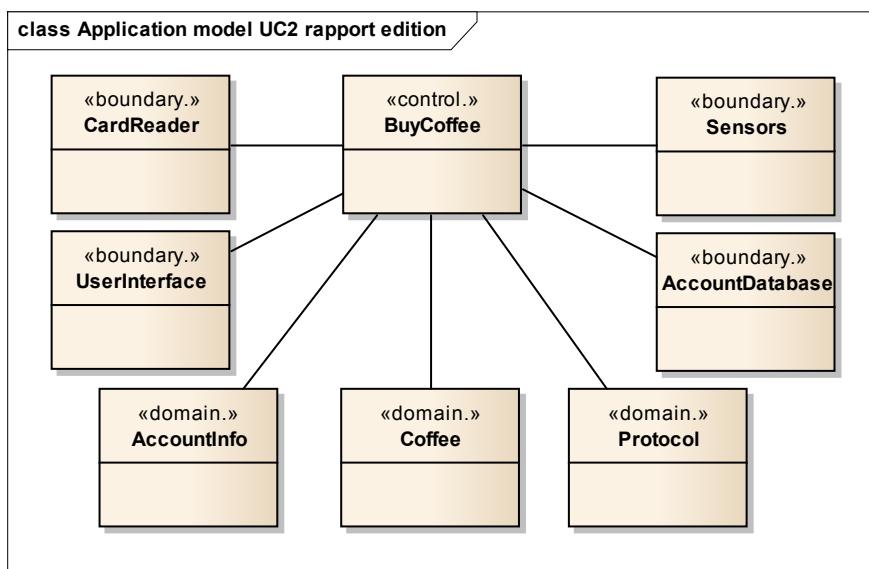
Softwaren designes så klasserne har så specifikt og indskrænket ansvar som muligt. Formålet med dette er at give hver klasse en højere selvstændighed, og mindske koblingen klasserne imellem, hvilket medfører, at ændringer i specifikke klasser kan foretages uden at påvirke det resterende softwaremodul.

8.2.3. Klasseidentifikation

Ud fra domænemodellen udarbejdes applikationsmodeller. Dette gøres for at identificere systemets klasser og hver klasses individuelle formål. Hver applikationsmodel repræsenterer en use case, og har til opgave at vise de klasser, som er involveret i den pågældende use cases funktionalitet. Til use case 2 kan der identificere følgende klasser:

- Grænsefladeklasser
 - **UserInterface** - Brugergrænseflade til bruger
 - **Sensors** - Grænseflade til den kaffebryggende del af systemet
 - **AccountDatabase** – Grænseflade til konteringsdatabasen
 - **EmailSender** – Grænseflade til operatør og bruger
- Domæneklasser
 - **AccountInfo** – Håndtering af den aktuelle brugers informationer
 - **Coffee** – Håndtering af de specifikke parametre for valgt kop
 - **Protocol** – Hvilken protokol der skal kommunikeres mellem PSoC og DevKit

Udover de definerede domæne- og grænsefladeklasser består en applikationsmodel af en enkelt kontrolklasse, der varetager use casens forløb, ved brug af de andre klasser. Ud fra de definerede klasser kan der nu udarbejdes et klassediagram for use casen.

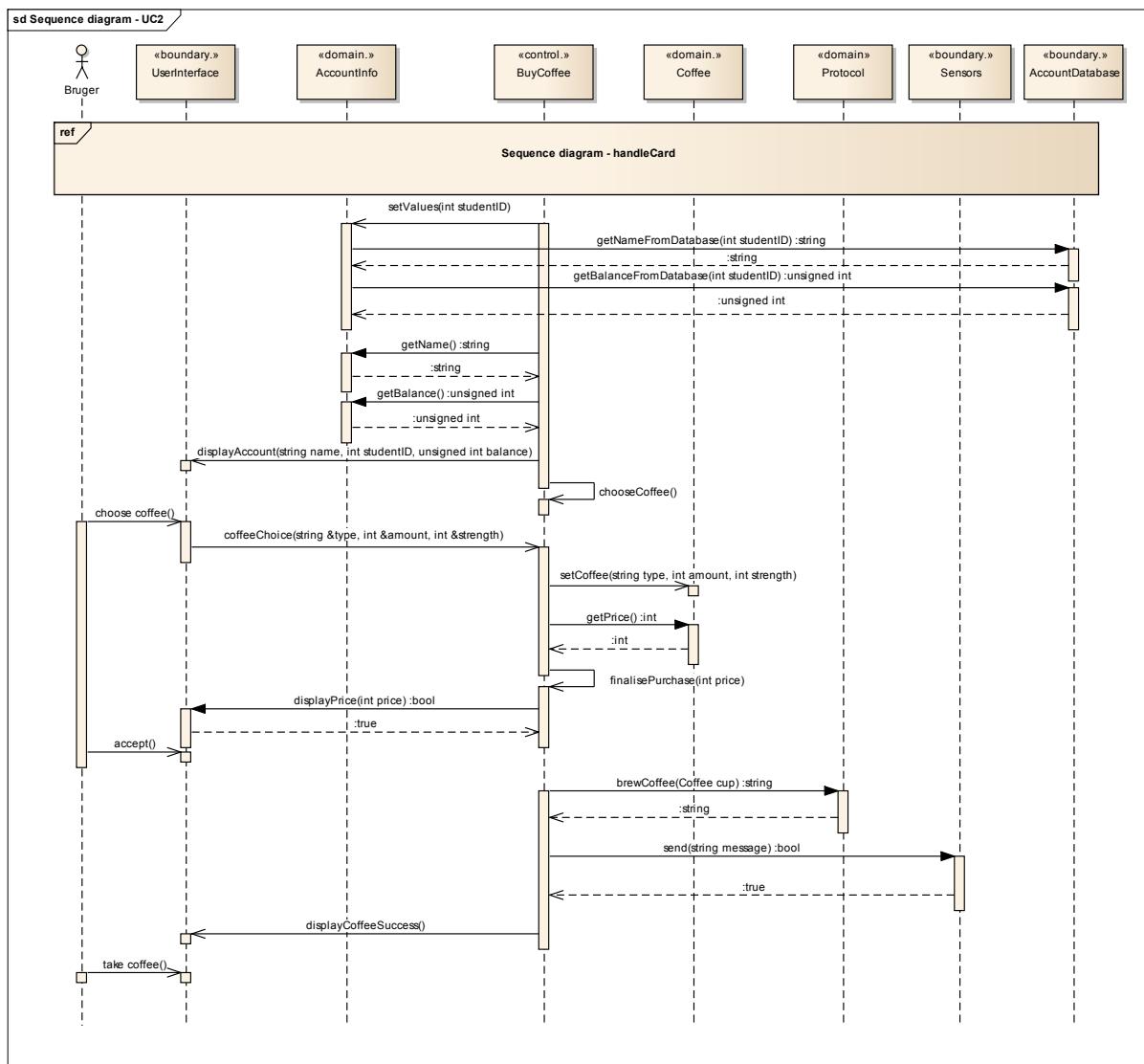


Figur 9 - Klassediagram UC2

8.2.4. Metodeidentifikation

Næste skridt i udarbejdelsen af en applikationsmodel er at designe et adfærdsdiagram for use casen. Her designes et sekvensdiagram, og udarbejdes på baggrund af de nyligt fundne klasser og sekvensen i use casene. Ud fra diagrammerne kan de grundlæggende metoder, som fuldfører use casens funktionalitet, defineres. I forlængelse af det overordnede sekvensdiagram, kan man nu også se systemets interne kommunikation når brugeren vil købe kaffe.

I sekvensdiagrammerne er det kun hovedscenariet fra use casene, som er beskrevet. Det er valgt at beskrive undtagelserne med state machines.

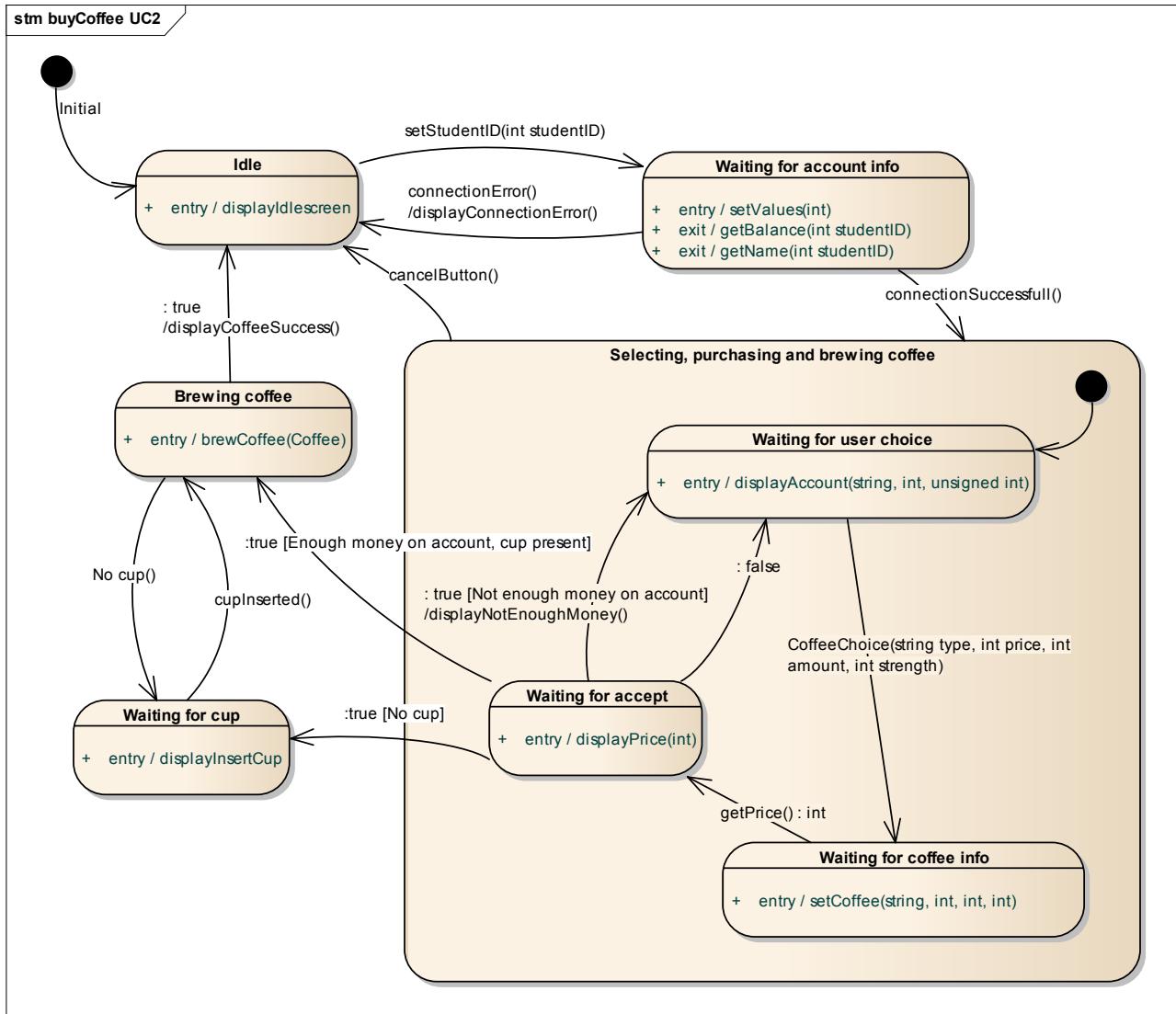


Figur 10 - Sekvensdiagram UC2

8.2.5. Uddybende kontrolklasse beskrivelse

For at give yderligere indblik i hvordan kontrol klassen varetager sine opgaver, beskrives softwaren med et state machine diagram. Formålet med state machine diagrammet er at vise kontrol klassens tilstande igennem use casens forløb. Der beskrives også hvordan use casens undtagelser håndteres.

Her identificeres metoder der håndterer undtagelser, når brugeren vil købe kaffe. Desuden ses det at det skal være muligt at annullere kaffekøbet.



Figur 11 - Statemachine UC2

8.3. Implementering af Brugergrænseflade

8.3.1. Indledende implementeringsovervejelser

Da systembeskrivelsen beskriver et eventdrevet system, kan det være fordelagtigt at implementere softwaren med tråde. Dette indebærer at tråde og håndtering af kommunikation mellem dem skal designes og implementeres.

Implementeringen af softwaren sker i Qt⁴, da det er et framework, specielt rettet imod GUI applikationer. Qt indeholder biblioteker til håndtering af tråde og trådkommunikation. Kommunikationen er udarbejdet som et beskedssystem, der bygger på en publisher/subscriber model, hvor tråde kan sende signaler, som andre tråde kan reagere på. I Qt kaldes disse beskeder signals og funktionerne til at håndtere beskeder kaldes slots.

8.3.2. Generelt om implementering

På brugergrænsefladen oprettes en tråd for hver kontrolklasse, der skal varetage den pågældende use cases forløb. Derudover oprettes tråde, der varetager skærmens UI og grænseflader til henholdsvis PSoC og kortlæser.

Da ovennævnte tråde skal kommunikere med hinanden, er der implementeret et beskedssystem. Først implementeredes et MessageQueue system, baseret på et andet OS API end Qt's eget. Det viste sig at OS API'ets beskedssystem gav problemer i forbindelse med implementering af GUI, da skærmbilleder frøs. Dette problem blev løst ved at bruge Qt's eget beskedssystem, som er bedre integreret i frameworkt. Overgangen til det nye beskedssystem medførte mindre ændringer i de implementerede metoder. Eksempelvis vil sekvensen *setValues* i sekvensdiagrammet fungere på den måde at BuyCoffee broadcaster et signal, som er forbundet med *setValues(int studentID)* slottet i AccountInfo. Når signalet broadastes vil slottets funktion blive kørt.

8.3.3.

⁴ Se <http://qt-project.org/wiki/> for videre information

8.3.4. Grafiske grænseflader

Det var et krav at der skulle laves en brugergrænseflade, hvorpå brugeren kan interagere med systemet. Som udgangspunkt blev der tegnet en skitse⁵, som skulle danne basis for køb af kaffe. Vindeuet der vises under use case 2, BuyCoffee, er et centralt vindue og var det første vindue der blev implementeret. Under udviklingen af den grafiske del af BuyCoffee blev der samtidig arbejdet på en konteringsdatabase, hvori brugerens saldo skal kunne findes, samt navne og priser på de 3 typer kaffe. Derefter skulle der laves en forbindelse mellem konteringsdatabasen og den grafiske brugergrænseflade. Da forbindelsen var etableret kunne saldo hentes direkte fra databasen i henhold til brugerens ID. Det sidste der skulle implementeres var mængde, styrke og pris. Mængden blev implementeret med steps af 0.5dl pr step, styrke blev implementeret i 3 trin: "Mild, Normal og Stærk". Prisen beregnes på basis af type og mængde. Resultatet er et fuld funktionelt BuyCoffee UI.



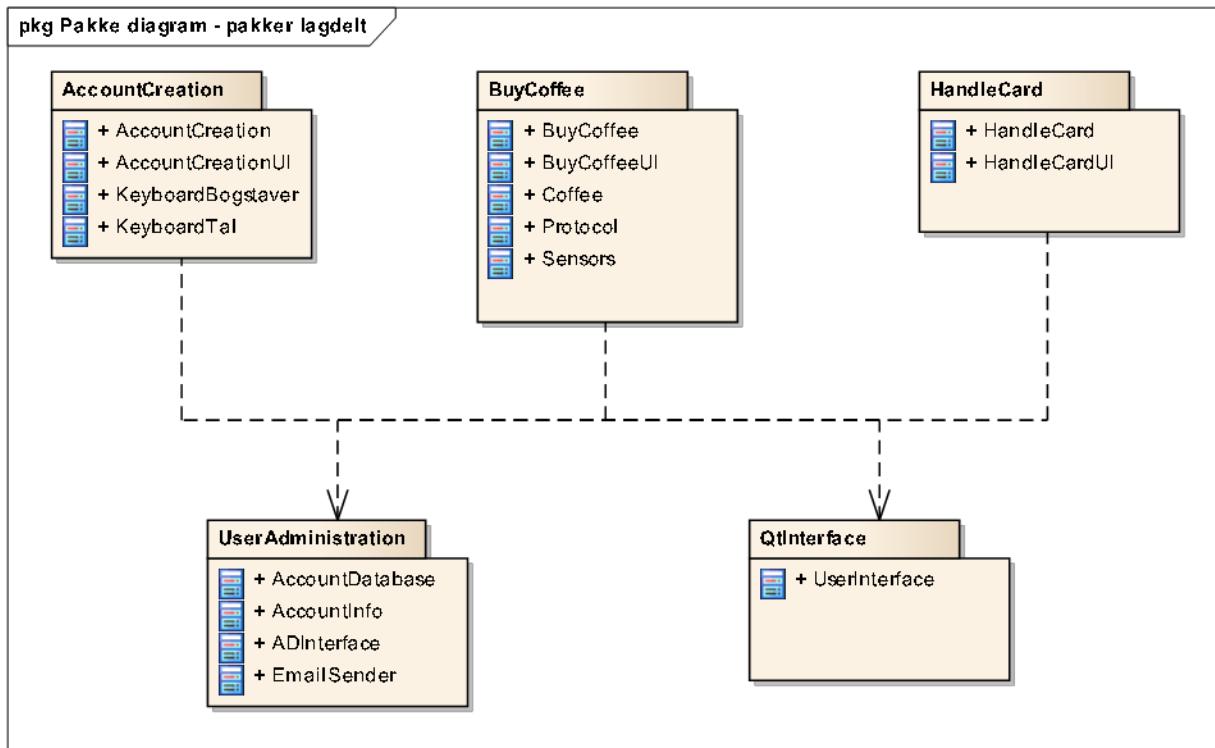
Figur 12 - BuyCoffeeUI.ui

⁵ Se Dokumentation, afsnit 1.4.3

8.3.5. Pakkeinddeling

I nedenstående pakkediagram ses alle systemets klasser. De tre øverste pakker indeholder hver en kontrolklasse, samt de klasser som kun bruges af den. Pakkerne i bunden indeholder klasser der bruges af de tre øverste, og er samlet efter funktionalitet.

I dokumentationen er pakkerne beskrevet hver for sig med detaljerede klassebeskrivelser⁶.



Figur 13 - Implementeret pakkediagram

⁶ Se Dokumentation, afsnit 4.3.4

8.4. Databaser og webfront

8.4.1. Konteringsdatabase

Konteringsdatabasesens formål er at holde informationer om brugere, transaktioner og kaffebeholderne. For at kunne opfylde kravspecifikation er det nødvendigt at både kaffeautomaten og webfronten kan tilgå konteringsdatabasen. På baggrund af dette blev databasetypen MySQL valgt, da denne database kan implementeres gratis, og opfylder alle krav.

Konteringsdatabasen er etableret på en Ubuntu server, som både indeholder webfronten og databasen. I databasen er de relevante tabeller, for at opfylde kravspecifikationen, oprettet.

Klassen, AccountDatabase, fungerer som grænsefladeklasse mellem brugergrænsefladen og konterringsdatabasen. Denne sørger for at brugergrænsefladen kan tjekke om et studiekort findes i databasen, hvor mange penge brugeren har på sin konto, hvilken kaffe der er i hvilke beholdere og hvor meget der er i dem. Klassen er implementeret med Qt frameworkets indbyggede SQL bibliotek, som benytter normal SQL syntaks.

8.4.2. ADInterface

ADInterface bruges til at udlæse informationer, om institutionens brugere, fra en AD database. AD-Interface bruges af brugergrænsefladen til at verificere, at det indtastede brugernavn er korrekt når en bruger oprettes, og henter navnet på brugeren, når der købes kaffe.

AD databasen er institutionens brugerdatabase. Desuden bruges AD databasen også til login på webfronten.

8.4.3. 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 MVC⁷ tankegangen, der gør at det er agilt og let at tilpasse til egne behov. Samtidigt har YII indbyggede funktioner til mange af de ting, som bliver nødvendige i webfronten. Derfor kunne arbejdet fokuseres på hvad der var specielt ved dette projekt.

⁷ Model View Controller

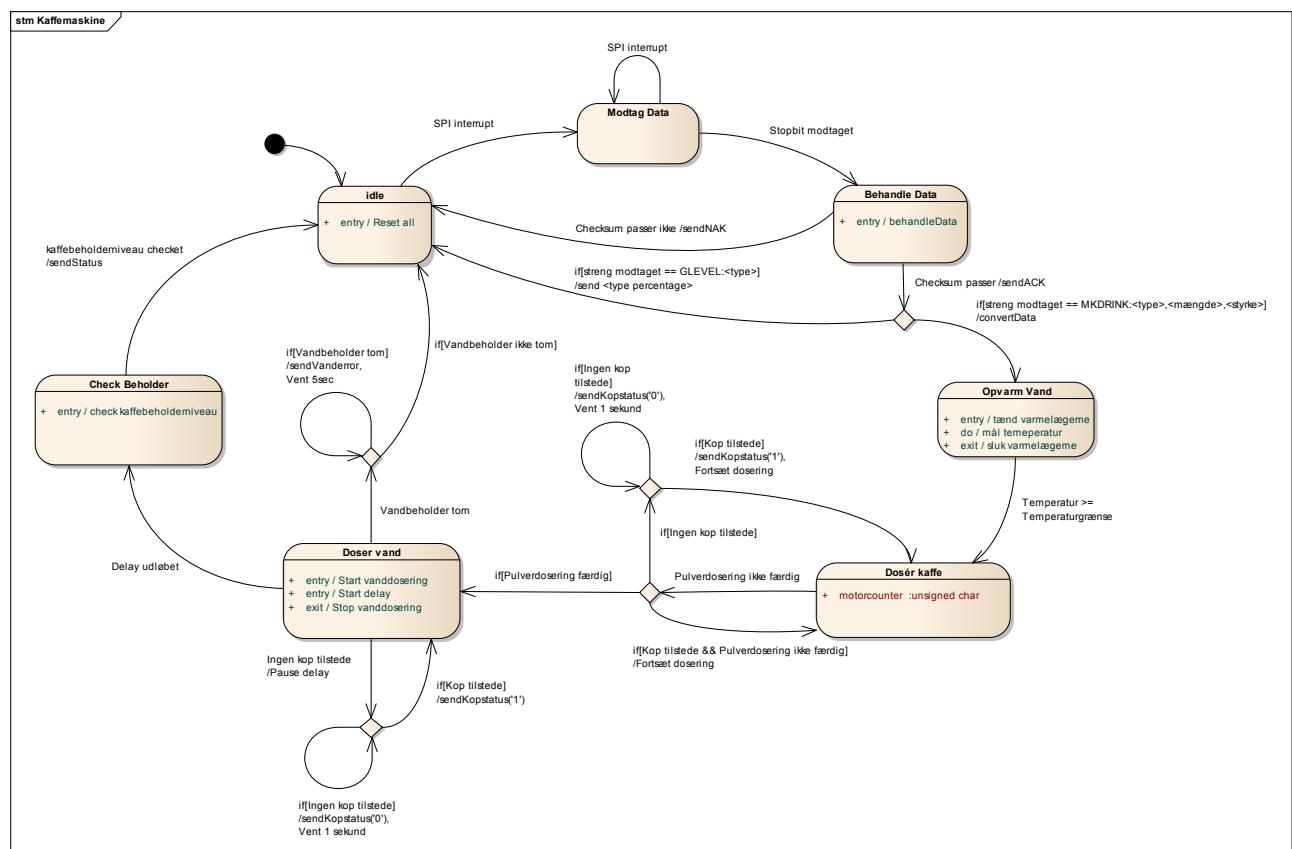
8.5. Design af programmet til PSoC

Programmet til PSoC'en er designet ud fra systemarkitekturen. Programmet skal opfylde den logiske funktionalitet i vandstyring og pulverdosering og muliggøre at PSoC'en kan kommunikere med brugergrænsefladen.

Tanken bag designet er at PSoC'en skal være en grænseflade mellem brugergrænsefladen og diverse sensorer og aktuatorer. PSoC'en skal derfor kunne modtage bestillinger på, og brygge, den ønskede kaffe. Brugergrænsefladen skal desuden kunne forespørge at få oplyst niveauet i en specifik kaffebeholder, hvorefter PSoC'en returnerer, hvor meget pulver der er tilbage i beholderen. Hvis der under brygningen forekommer fejl, såsom at der ikke er placeret en kop på spildbakken eller at der ikke er mere vand i vandbeholderen, skal programmet kunne oplyse brugergrænsefladen om den pågældende fejl.

Ved en kaffebestilling, skal brugergrænsefladen sende den ønskede mængde vand i dl og den ønskede mængde pulver i ml. Når PSoC'en underetter brugergrænsefladen om indholdet i en kaffebeholder, skal dette være angivet i procent. Dette er valgt for at mindske koblingen i systemet. Uregningen af mængden af vand og pulver, ud fra et blandingsforhold, sker derfor på brugergrænsefladen, og omregningen til værdier der er specifikke for diverse aktuatorer sker på PSoC'en. Ligeledes omregnes pulverniveausensorenes output til en procentvis repræsentation af indholdet.

Programmet til PSoC'en er designet og implementeret ud fra et state machine diagram. Dette diagram ses nedenfor.



Figur 14 - state machine PSoC

8.5.1. Statebeskrivelse

Idle er det indledende state for programmet. Ved indgangen til idle resættes relevante variable, buffere og countere, der bruges i de andre states. PSoC'en står i idle, indtil den modtager et SPI interrupt fra brugergrænsefladen.

Modtag Data er det state PSoC'en går ind i efter at der er modtaget et SPI interrupt fra DevKittet. DevKittet generer et interrupt på PSoC'en hver gang der skal sendes en besked fra DevKittet til PSoC'en. PSoC'en bliver i dette state til hele beskeden er modtaget

PSoC'en går ind i **Behandle Data**, når DevKittet har modtaget en besked. Det er i dette state at beskeden valideres og behandles. Hvis den modtagende besked fejler valideringen, sendes et NAK til DevKittet og PSoC'en returnerer til idle. Ellers sendes et ACK. Derefter behandles den sendte besked. Hvis den sendte besked er en bestilling omregnes de sendte parametre på type, mængde og styrke til aktuatorspecifikke værdier.

Det næste state PSoC'en går ind i er **Opvarm Vand**. I dette state opvarmes vandet i vandbeholderen til en fastsat temperaturgrænse. Dette gøres ved at tænde varmelegemet. Når temperaturgrænsen er nået, slukkes varmelegemet, og PSoC'en forlader Opvarm Vand.

Efter Opvarm Vand går PSoC'en ind i **Dosér Kaffe**, hvor der doseres kaffepulver. Pulveret doseres ved at køre doseringsmekanismen. Under pulverdoseringen tjekkes der løbende om der er en kop til stede på spildbakken. Hvis der ikke er en kop sendes en fejlbesked til DevKittet og der ventes 1 sekund. Dette gentages til en kop er placeret på spildbakken. Når det detekteres at en kop er placeret på spildbakken sendes en besked til DevKittet, hvorefter pulverdoseringen fortsættes. Når den ønskede mængde pulver er doseret går PSoC'en ind i næste state.

Doser Vand er den næste state. Det er i dette state at der skal doseres den bestilte mængde vand. Når PSoC'en går ind i Doser Vand, startes vanddoseringen. Vanddoseringen er styret ved at dosere vand på tid. Hvis brugeren fjerner sin kop pauses doseringen og programmet handler på samme måde, som hvis brugeren fjernede sin kop under pulverdosering. Hvis det detekteres at der ikke er mere vand i vandbeholderen, stoppes vanddoseringen. Derefter sender PSoC'en en fejlmeddeelse til DevKittet hvert 5. sekund indtil der igen er vand i beholderen. Når der er fyldt vand i beholderen, går PSoC'en til Idle.

Den sidste state er **Check Beholder**. Her tjekkes kaffebeholderne og der sendes en statusbesked til DevKittet med beholderenes niveau. Derefter går PSoC'en tilbage til idle.

8.6. Implementering af programmet til PSoC

Under implementeringen af programmet, er der den sædvanlige udfordring ved at skrive et program med lav kobling mellem metoderne i C. Dette er løst ved at implementere state machine diagrammet i *main* funktionen med en switch case struktur. Der er så vidt muligt forsøgt at dele funktionalitet op i header- og c filer. Desuden er der forsøgt at implementere programmet så *main* funktionen agerer som kontrolklasse, der styre den overordnede sekventielle gennemgang i diagrammet.

Ud fra systemarkitekturen er PSoC'ens grænseflader til hardwaren fastsat. PSoC'en skal måle på diverse analoge og digitale sensorer samt at styre forskellige aktuatorer.

8.6.1. Måling af sensorer

De **analoge sensorer** der måles med er en **temperatursensor**, og en række **pulverniveausensorer**. Alle sensorkredsløb er bygget så den mindste og maksimale værdi der skal måles giver et spændingsoutput på henholdsvis 0V og 5V. Til at måle disse sensorer er der benyttet en intern delta-sigma ADC i PSoC'en. Når en sample er taget, trigges et interrupt, og værdien lægges over i en global variabel.

Systemets to digitale sensorer, er en kopsensor og en vandniveausensor. Begge er designet så outputtet er high, hvis der henholdsvis er en kop til stede og vand i vandbeholderen, og low hvis det modsatte er tilfældet.

8.6.2. Styring af aktuatorer

De brugte **aktuatorer** er et varmelegeme, en række pulverdoseringmekanismer og en magnetventil.

Varmelegemet er designet så dets status toggles, ved en rising edge på et GPIO ben. Det er kritisk at PSoC'en ved om varmelegemet er tændt eller slukket. Derfor er der implementeret et flag, der repræsenterer varmelegemets status.

Pulverdoseringen er implementeret med en step motor, der styres med et firkantssignal. Hver rising edge trigger et step på motoren. Dette er implementeret så en motorcounter tælles op for hvert step, til der er doseret den korrekte mængde pulver.

Vanddoseringen er implementeret som en magnetventil, der bliver styret af PSoC'en med et high eller et low på et GPIO ben. Hvis en kop fjernes trigges et interrupt. Det var oprindeligt tanken at tiden skulle styres af en timer, så doseringen kunne sættes på pause, til koppen igen detekteres på spildbakken. Delays på PSoC'en vil også pauses af et interrupt, og tiden er derfor styret af delays.

8.7. Kommunikation mellem DevKit og PSoC

Under udarbejdelse af systemarkitekturen er kommunikationen mellem DevKit og PSoC allokeret på en seriel kommunikation. Da SPI er standardiseret, nem at finde dokumentation på, og understøttes både af DevKit og PSoC blev denne standard valgt. Endvidere understøtter DevKit'et to slave-enheder på samme bus, hvilket muliggjorde at kortlæseren og PSoC'en kunne sidde på samme bus.

En protokol blev designet ud fra ønsket om en fejlhåndtering, der inkluderer implementering af checksum og stopbit. Denne protokol har virket som grundlag for kommunikationen, og forefindes i dokumentationen.

8.7.1. Design af spi drive

I PSoC-Creator indgår SPI som en færdig blok, der kan håndtere den ønskede funktionalitet. Da både DevKit og PSoC skal kunne tage initiativ til at sende, udbyggedes SPI funktionaliteten med 2 gpio-linjer, der fungerede som interrupt. Til at håndtere kommunikationen på DevKit'et er der udviklet en SPI-driver. Denne SPI-driver følger den klassiske opdeling for linux device drivers. Denne opdeling er illustreret på figuren til højre.

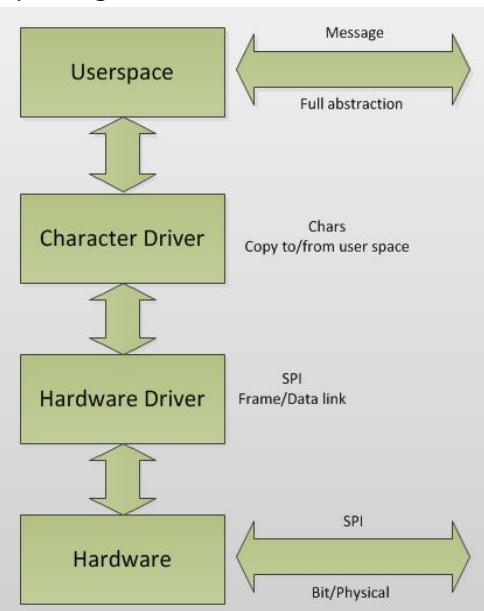
Det udviklede character device har til opgave at modtage og sende data mellem user space- applikationen og hardware driveren. Dertil er det i dette lag at de to GPIO linjer mellem DevKittet og PSoC'en håndteres. Hardware driveren har til opgave at binde character devicet med specifik hardware, der skal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

8.7.2. Implementering af SPI driver

Under udvikling af driveren har det vist sig at være hensigtsmæssigt at udarbejde to test-drivere.

De to test-drivere er udviklet til at verificere at styringen af de to GPIO linjer var implementeret korrekt. Dette gjorde det lettere at fejlsøge under udvikling af den endelige driver.

Det oprindelige design var udarbejdet med en blocking read, for kun at læse når PSoC'en ønskede at sende en besked. Dette viste sig at være problematisk, da tråden der håndterede kommunikation låste imens den ventede på data. Det blev derfor implementeret som non-blocking, der validerer om der er data klar ud fra GPIO-linjen, og derefter informerer tråden herom. Denne implementering medførte at tråden ikke låste, og derfor var i stand til både at læse og skrive som ønsket.



Figur 15 - Figur for character device opbygning

9. Design og implementering af hardware

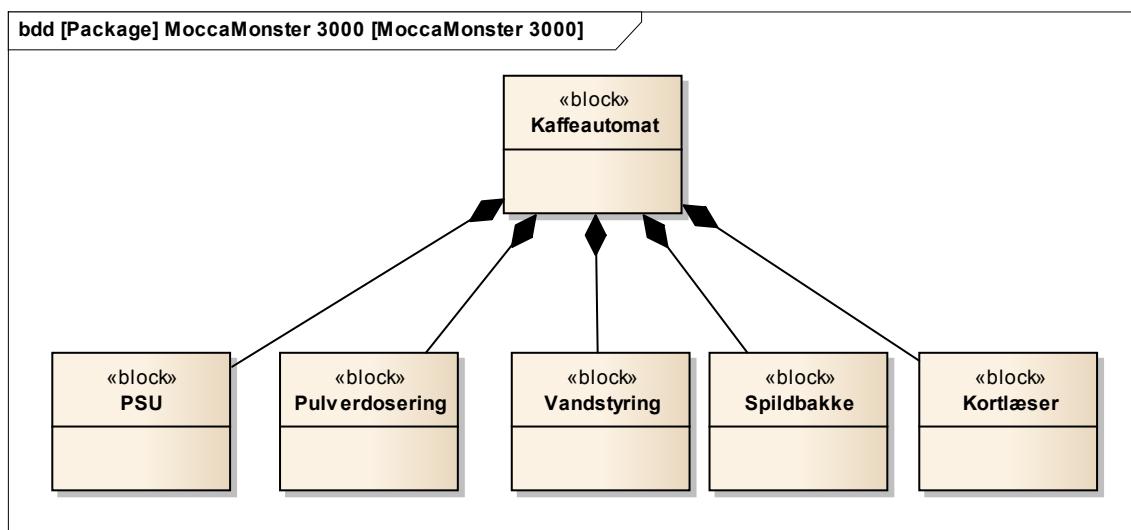
I det følgende beskrives det detaljerede hardwaredesign for projektet. Det detaljerede design tager udgangspunkt i kravspecifikationen og systemarkitekturen. Designprocessen samt hvilke overvejelser og valg der er taget vil desuden blive beskrevet.

For uddybende beskrivelse af design og implementering af hardware henvises der til dokumentationen.

9.1. Indledende designovervejelser

I den overordnede arkitektur for kaffeautomaten, ses det hvilke blokke det samlede system skal bestå af og det er beskrevet hvilken funktionalitet de skal kunne opfylde.

De relevante hardware blokke er som følger:



Figur 16 - bdd for kaffemaskinen

Med disse blokke identificeret, er det næste trin i designet at nedbryde blokkene yderligere for at bestemme de nødvendige hardwareenheder og deres interne grænseflader.

På diagrammet er en ny blok ved navn PSU introduceret. Denne blok introduceres da der designes en PSU specifikt til systemet.

9.2. Overvejelser omkring sensorer

I systemarkitekturen blev det fastlagt, at det endelige produkt ville kræve en række sensorer af forskellig art. I systemet skal der implementeres sensorer, som kan detektere følgende:

- Vandets temperatur i vandbeholderen
- Vandstanden i vandbeholderen
- Mængde af pulver i pulverreservoir
- Detektering om der er placeret en kop på spildbakken
- Mængden af spild i spildbakken

Ud fra ovenstående blev det besluttet, at følgende sensor typer skulle anvendes:

- Temperatursensor
- Nærhedssensor
- Niveausensor

En generel model for et målesystem, der kan måle på en fysisk størrelse og omdanne dette til et signal en A/D konverter kan digitalisere kan se ud som på nedenstående illustration



Figur 17 – Målesystem for analoge sensorer⁸

Føler: Føleren/sensoren mäter den ønskede fysiske størrelse. Dette kan være en ændring i modstand, et optisk input etc.

Måleomsætning: Den målte fysiske størrelse omsættes til en spænding, da denne kan forstærkes.

Forstærker: Signalet forstærkes således at hele indgangen på A/D konverteren udnyttes.

Transmission: Her omformes signalet til den type, der ønskes transmitteret. Eksempelvis kan en strøm-transmission være en god løsning, hvis der skal transmitteres over større afstande.

LP filter: Et lavpas filter kan eventuelt implementeres for at fjerne højfrekvent støj.

A/D: Analog til digital konverteren omformer et analogt signal til et digitalt signal. Hvis måleresulterne eksempelvis transmitteres som strøm vil det være nødvendigt at implementere endnu en måleomsætning da A/D konverteren arbejder med spændingsinput. A/D konverteren der anvendes i dette projekt er placeret internt i PSoC'en.

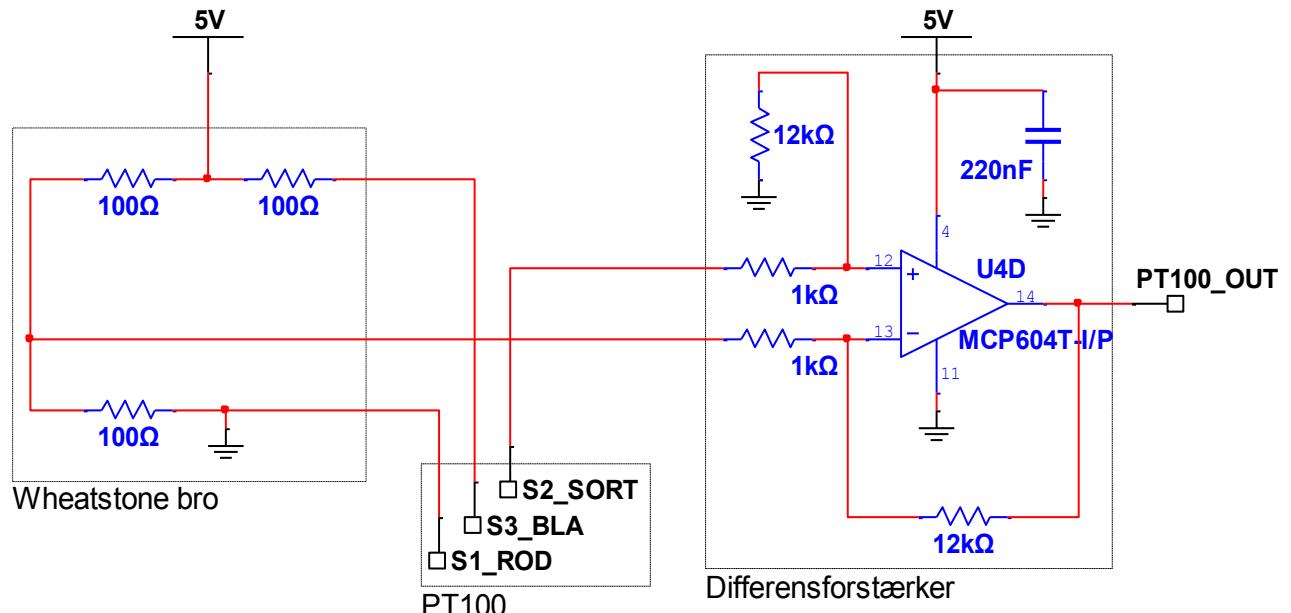
Ud fra denne model er målesystemet for diverse sensorer udviklet.

⁸ Illustration: Arne Justesen

9.2.1. Temperatursensor

"Temperatursensor" er en del af blokken "Vandstyring". Der er valgt at gå i detaljer med designet af temperatursensoren, da den giver et repræsentativt billede på hvordan det fornævnte målesystem kan implementeres.

Temperatursensoren har til formål at måle vandets temperatur i vandbeholderen. Der er valgt at anvende en temperatursensor af typen PT100. Denne type sensor ændrer sin modstand afhængigt af omgivelsernes temperatur. Ved at implementere den efter ovenstående model bliver den velegnet til at måle temperaturen i vandreservoirtet.



Figur 18 - Diagram for temperatursensor

Wheatstone broen omsætter **PT100** sensorens modstandsændring til en spændingsændring. Denne ændring i spænding kan herefter forstærkes med en **differensforstærker**. Det er nødvendigt at anvende en differensforstærker, da det er differensen mellem de to outputs fra Wheatstone broen, der skal forstærkes. Anvendelse af en instrumentationsforstærker har været på tale, da den har flere fordele, som lavt offset, lav støj og høj stabilitet. Denne løsning blev dog fravalgt af økonomske årsager.

Efter forstærkningen kan signalet transmitteres til PSoC'en.

PT100 sensorer kan indkøbes med både to og tre ledninger tilsluttet den temperaturfølsomme modstand. PT100 sensoren anvendt i dette projekt har tre ledninger. I en sensor med to ledninger vil der være en fejlmåling, i form af ledningsmodstanden. Ved at anvende en PT100 sensor med tre ledninger afhjælpes dette problem til dels, så det nu kun er ledningsmodstanden for den ene af ledningerne der vil kunne måles. Dermed er målingen mere nøjagtig.

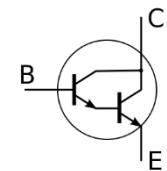
Ovenstående diagram er vist med modstandsværdier for Wheatstone broen. De specifikke modstandsværdier er valgt da PT100 sensorens måleområde er defineret mellem 0°C og 100°C. 0°C svarer til en modstandsværdi for PT100 på 100Ω. Dette svarer til at broen er i balance og temperatursensoren vil sende 0V til PSoC'en. Ved en temperatur på 100°C vil temperatursensoren sende 5V til PSoC.

9.2.2. Vandniveausensor

"Vandniveausensor" er en del af blokken "Vandstyring". Vandniveausensoren har til formål at detektere hvorvidt der er i vandbeholderen. Vandniveausensoren kunne designes med en kapacitiv føler, men sådan en løsning blev fundet for dyr.

I stedet udnyttes vandets elektriske ledeevne. Dermed kan vandniveausensoren implementeres med 2 målepunkter i vandreservovert. Vandniveauet detekteres ved at sende en strøm mellem målepunkterne ind i en Darlington transistor.

En Darlington transistor består internt af to bipolare transistorer som har fælles collector-ben. Emitter-benet på den første interne transistor er koblet til base benet på den anden interne transistor. Derved bliver base-strømmen forstærket over to omgange. Darlington transistoren er derfor velegnet til registrering af den svage strøm, der løber mellem de to målepunkter i vandbeholderne. Når vandet ikke berører det ene målepunkt, vil transistoren ikke åbne, og der kan ikke løbe strøm gennem kredsen.



Figur 19 - Darlington transistor

Under designforløbet og enhedstest fungerede kredsløbet efter hensigten. Under integrationstesten opstod der dog komplikationer med kredsen, hvilket medførte at det valgte design ikke blev implementeret iden færdige prototype. Se integrationstesten under dokumentationen for yderligere information⁹.

⁹ Se Dokumentation, afsnit 5.9

9.3. Vanddosering

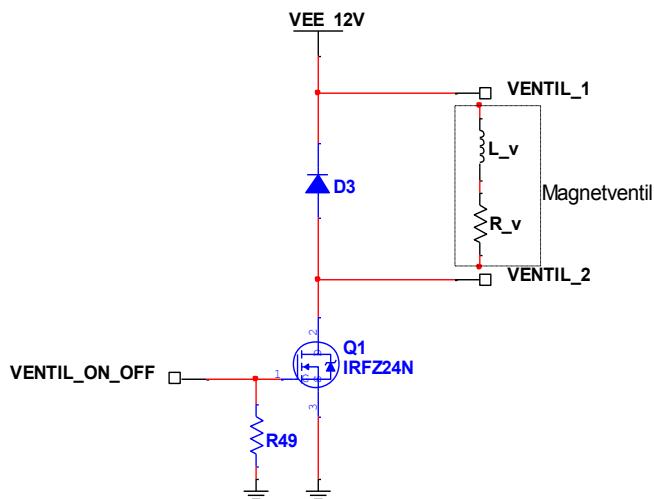
"Vanddosering" er en del af blokken "Vandstyring". Vanddoseringen skal sidde som mellemled mellem vandbeholderen og en slange, der fører vandet ned i blandingstragten. Vanddoseringen har til formål at dosere den bestilte mængde vand. For at løse dette anvendes en magnetventil, med en tilhørende regulering, der aktiveres af PSoC'en. En skitse af kaffeautomaten i tværsnit ses til højre.

En magnetventil er internt et mekanisk relæ, der bliver aktiveret når den bliver påtrykt en DC-spænding. Når relæet trækker, aktiveres et lukkespæld og der åbnes for gennemgang af vand i det tilkoblede rør.

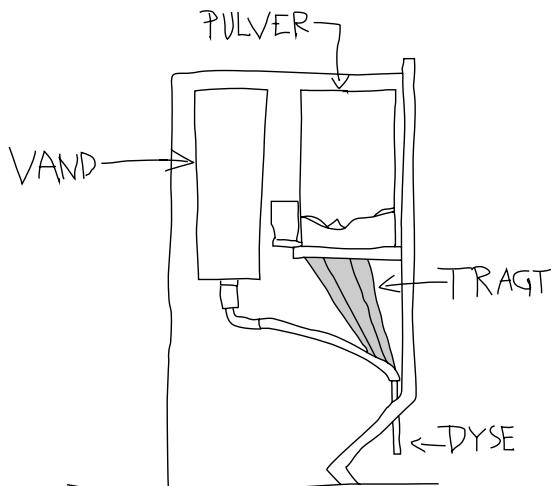
Relæet i magnetventilen består af en spole med en ohmsk modstand i spolens viklinger. Når denne spole påtrykkes en spænding, opbygges der et magnetfelt omkring den. Når magnetventilen skal lukke,afbrydes denne spænding momentant. Strømmen igennem en spole kan dog ikke ændre sig momentant, og spolen vil derfor "finde en vej", hvor strømmen kan løbe. Da der ikke længere er nogen fysisk forbindelse vil spolen i stedet inducere en meget høj spænding, nu med modsat polaritet, i et forsøg på at tvinge en strøm igennem. Denne høje spænding kan ødelægge diverse komponenter.

Dette problem kan afhjælpes med en flyback diode(D3). Når flyback dioden er koblet parallelt med spolen, er der nu en vej for strømmen at løbe. Spolen vil dermed ikke inducere den høje spænding. Med flyback koblingen er spolen og resten af kredsløbet nu beskyttet imod peakspænding ved afbrydelse af magnetventilen.

Reguleringen af magnetventilen styres med et digitalt signal fra PSoC, på benet VENTIL_ON_OFF.



Figur 21 - Magnetventil diagram



Figur 20 - Skitse af kaffeautomat i tværsnit

9.4. Doseringsmekanisme

"Doseringsmekanisme" er en del af blokken "Pulverdosering". Doseringsmekanismen skal styre dosering af pulver til brugeren. Det er, i systemarkitekturen, lagt op til at der kan være flere doseringsmekanismer, så brugeren kan vælge hvilken type vedkommende ønsker. Det er desuden et krav at doseringen af pulver skal implementeres med relativ høj præcision. Løsningen på dette er at anvende step-motorer.

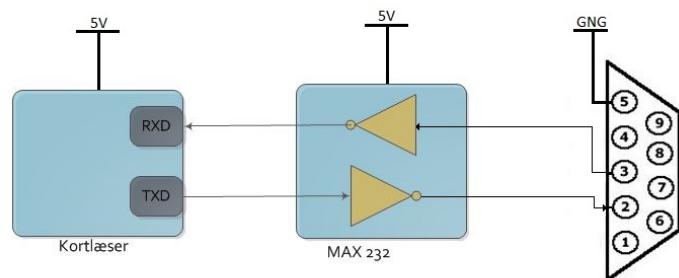
Som i almindelige elektromotorer findes der i step-motorer et antal elektriske spoler og magneter. Step-motorer adskiller sig fra traditionelle elektromotorer ved at step-motorer roterer i mindre steps og derfor kan styres præcist. Såkaldt microstepping muliggør steps på brøkdele af grader. På grund af step-motorens konstruktion kræver step-motorer ekstern elektronik for at levere strømpulser til de interne spoler i de specifik rækkefølge. Det er derfor nødvendigt at implementere en step-motor driver. Denne driver konverterer et pulssignal fra PSoC'en til fornævnte strøm-pulser. En step-motor driver kan godt implementeres på PSoC'en rent softwaremæssigt uden brug af yderligere hardware. Dette blev fravalgt, da der er et større arbejde i at implementere en sådan driver i software. Desuden mindsker det koblingen i systemet, da de forskellige moduler grænseflader simplificeres. I stedet blev det besluttet at indkøbe en færdig motor driver IC. Den anvendte driver kunne ikke programmeres, men havde i stedet en række logiske input, som på hardware niveau konfigurerede driveren.

En vigtig indstilling i forbindelse med step-motorer er opløsningen. Jo højere opløsning en step-motor kører med, jo flere steps er der per omgang og jo mindre vinkel er der mellem hvert step. Dette medfører en meget præcis motor. Ved høj opløsning, vil viklingerne i motoren aldrig være fuldt aktiverede, og dette giver mindre moment. Der er altså en balancegang mellem opløsning og moment. I dette projekt blev det først vedtaget at motoren skulle operere med 1/8 step. Det kunne dog konkluderes ved eksperimenter, at den først indkøbte motor havde for lidt moment til at løse den pågældende opgave. Der blev derefter eksperimenteret med lavere opløsning, men motoren var stadig ikke kraftig nok. Den endelige løsning blev indkøb af en kraftigere step-motor. Denne var væsentlig kraftigere, og det var nu muligt at køre med den tidligere valgt opløsning.

9.5. Kortlæser

Kortlæseren skal kunne aflæse brugerens adgangskort og sende det unikke kortnummer til DevKittet via en seriell kommunikation. Da DevKittet understøtter en SPI-bus med to slaveenheder, var det et oplagt valg at grænsefladen til kortlæseren skulle være SPI, da der skulle bruges SPI til kommunikationen med PSoC'en. En IC med disse specifikationer blev bestilt og der blev designet styrekredsløb samt en antennen. IC'en viste sig at være yderst avanceret og det interne register skulle konfigureres korrekt til både SPI kommunikation og specifikationerne for antennen. Det blev vurderet at opgaven var for omfattende, og der blev derfor fastlagt en timeframe. Da opgaven ikke var løst indenfor den pågældende timeframe blev det besluttet at der skulle findes en alternativ løsning.

En færdigt komponent (inkl. antennen), kunne købes indenfor budgettets rammer, og der blev derfor bestilt en færdig kortlæser. Denne færdigkøbte komponent havde et UART interface, hvilket medførte at der skulle designes noget simpelt hardware til spændingskonvertering, fra 5V på kortlæseren, til DevKitets 12V UART port.



Figur 22 princip i levelkonvertering

Softwaremæssigt var løsningen minimalt anderledes da der som standard i Devkittets kernel var en driver til UART-kommunikation. Det var derfor kun fil-placeringen, der ændredes i forhold til det oprindelige design.

10. Resultat og diskussion

Dette projekt er resulteret i en prototype på et system, der med få afvigelser opfylder kravspecifikationen. Alle krav for 3. semesterprojektet er integreret i det endelige produkt. Den endelige prototype er implementeret med viden og kundskaber, der ligger uden for dette og de tidligere semestres pensum. Dertil er en lang række udfordringer løst med andre faglige kundskaber, end dem uddannelsen sigter mod. Dette indebærer bl.a. at projektet har stillet en række mekaniske udfordringer.

Et af fokuspunkterne for implementeringen var at systemet skulle være stabilt og kræve minimal vedligeholdelse. I udviklingen af softwaren til systemet er dette i høj grad lykkedes. Brugergrænsefladen er implementeret således at alle forbindelser udadtil har et timeout i tilfælde af mindre fejl i systemet. Dette gør systemet bedre til fejlhåndtering og betyder at genstart sjældent er påkrævet. Softwaren til brugergrænsefladen er succesfuldt implementeret i henhold til designet, og opfylder kravene beskrevet i kravspecifikationen.

Softwaren på PSoC'en er implementeret med stor fokus på faren for at hænge fast i dele af koden. Derfor disables interrupts i situationer, hvor pågældende ISR rutiner ikke er relevante. Kommunikationen mellem brugergrænsefladen og PSoC'en er implementeret ved brug af egen udviklet protokol, og benytter sig af SPI standarden. Dette har været en stor udfordring, da det skulle sikres at sendte beskeder modtages korrekt og at ingen af parterne ville sidde fast under en transmission.

I hardwaren har der været et stort fokus på at implementere holdbare og præcise løsninger, der vil kunne tages fra prototypen og bruges i et salgsbart system. Derfor er der brugt mange ressourcer på at finde en løsning til præcis og konsekvent dosering af kaffepulver. Design af doseringenene er 3D printet, da det er en hurtig og billig måde for at verificere designet. Dele af de færdige enheder er derefter CNC fræset, for at fremstille en holdbar prototype. Til doseringen af vand er der ligeledes fundet en løsning, der præcist doserer vand med en maksimal afvigelse fra den bestilte mængde på en kvart deciliter.

Det har haft en høj prioritet at udarbejde et færdigt print, der indeholder al den designede hardware. Dette er igen møntet på at udvikle en prototype, der er stabil og tæt på et salgsbart produkt.

Til at måle på det fysiske system er der brugt en række sensorer. Temperatursensoren er implementeret i det endelige system og virker efter hensigten. Kopsensoren er enhedstestet, men fejlede i integrationen af systemet. Kopsensoren var ikke implementeret med stor nok følsomhed, hvilket gjorde at den i prototypen er blevet udladet. Ligeledes er pulverniveausensorerne enhedstestet og integrationstestet med PSoC'en. Dog virkede de ikke efter hensigten, og grundet tidsmangel er deres fejl ikke identificeret.

Til at detektere om der er vand i vandbeholderen, var dette først tænkt løst med en kapacitiv sensor. Denne type sensorer er relativt dyre i forhold til det budget IHA har stillet til rådighed. Desuden er det valgt at finde en billigere løsning, da det ellers ville øge prisen på det endelige produkt for meget, i forhold til nødvendigheden af den funktionalitet sensoren kan give automaten. Vandniveausensoren er i stedet implementeret ved at tilføre vandet en spænding og derved måle genemgangen mellem to punkter i vandbeholderen. Efter meget arbejde i at få implementeret sensoren, så den gav en konsekvent måling, blev det besluttet at der ikke var mere tid til at arbejde med sensoren.

Implementeringen af varmelegemet medførte visse udfordringer. Jævnfør IHA's regler for studerende, må der ikke arbejdes direkte med 230V. Der skulle dermed findes en måde at styre et varmelegemet, uden at arbejde direkte med 230V. Dette er løst ved at placere en tilkøbt fjernbetjent afbryder mellem varmelegeme og stikkontakt. Denne løsning er ikke optimal, da der på PSoC'en ikke kan aflæse afbryderens status, og da afbryderen kun kan toggles. Dermed kan der ikke valideres på om varmelegemet er tændt eller slukket, og et fejslag agent signal medfører at varmelegemet tænder på uønskede tidspunkter eller slet ikke slukker.

For at nedbringe bryggetiden, var det tiltænkt af vandet skulle holdes inden for et temperaturintervall når automaten ikke var i brug. Dette kunne af sikkerhedsmæssige årsager ikke lade sig gøre med tilkøbte løsning.

11. Udviklingsværktøjer

I forbindelse med udvikling og implementering af systemet, er forskellige udviklingsværktøjer blevet benyttet. Disse beskrives i følgende.

Multisim

Til design af hardware komponenter er simuleringsværktøjet Multisim anvendt. Det er brugt dels for at verificere egne beregninger, og dels for at skabe overblik over de konstruerede kredsløb. I forbindelse med anvendelse af Multisim er det bemærket, at de simulerede værdier ikke altid stemmer overens med de realiserede værdier. Dette skyldes variation i reelle komponentværdier, komponenter til rådighed, samt udefrakommende faktorer som støj.

Ultiboard

Til udlægning af print er programmet Ultiboard benyttet. Det er godt integreret med Multisim, og er derfor det oplagte valg.

Enterprise Architect

Enterprise Architect er anvendt til design af SysML-diagrammer. Det er besluttet af gruppen at benytter Enterprise Architect fremfor Microsoft Visio. Dette valg er taget, da Enterprise Architect er lettere at benytte, samt at de udviklede diagrammer har et pænere udseende.

Qt

Qt er et krydsplatforms framework, der hovedsageligt bruges til udvikling af GUI applikationer. Applikationerne skrives i sproget C++. Qt er et stærk framework, der integrerer trivielle opgaver så som håndtering af tråde og kommunikation mellem disse. Desuden findes en masse udvidelser som simplificerer programmeringsarbejdet i mange henseender.

PSoC Creator

PSoC Creator er et program specialiseret til udvikling af programmer til PSoC. Programmet tilbyder en grafisk og intuitiv tilgang til PSoC'ens mange ressourcer. PSoC Creator indeholder desuden et stærk debugging værktøj, som i alle henseender gør test og fejlfinding lettere.

Logic analyzer

Til debugging og fejlfinding på kommunikationslinjer, er en logic analyzer blevet benyttet. Logic analyzer sparer programmøren for meget debugging i koden, ved at verificere om de forskellige platforme udsender korrekte signaler.

Inventor

Inventor er et avanceret CAD program til fremstilling af tekniske tegninger til design af mekaniske løsninger. Programmet tilbyder en række værktøjer der gør design og fremstilling af mekaniske løsninger lettere. Desuden understøtter de fleste 3D-printere STL-filer som Inventor kan lave.

Makerbot Replicator

Replicator er en low-budget 3D-printer, som leverer gode resultater til en fornuftig pris. Replicator giver adgang til hurtige prototyper, så ideer hurtigt kan afprøves.

12. Opnåede erfaringer

I arbejdet med projektet er der opnået en lang række faglige erfaringer. Dette omfatter både faglig viden opnået ved at arbejde med projektet, men også et øget overblik over at arbejde i forskellige frameworks og med store embeddede systemer. Dertil har det været en stor læringsproces at skelne mellem hvad der skal bygges selv og hvad der bør tilkøbes.

I forbindelse med programmering af brugergrænsefladen, har det været en stor udfordring at inkorporere tråd programmering. Der er store muligheder i at programmere sit system med flere tråde, men det medfører også udfordringer vedrørende kommunikationen mellem dem. Dertil kommer udfordringer i at overføre tidligere erhvervede kundskaber til nye frameworks, og de udfordringer det medfører. Et eksempel på dette er at Qt har sit eget messagesystem, og at dette skal benyttes for at hindre at GUI'en låser.

En af de erfaringer der er gjort sig i forbindelse med at programmere PSoC'en, er at der er løst nogle problemer i hardware, der med fordel kunne være implementeres på PSoC'en. Dette hindrer at platformen udnyttes til fulde. Et eksempel på dette er at der findes en intern forstærkning i PSoC'ens ADC. ADC'en har en meget fin tolerance, og ville uden problemer kunne overtage denne funktion fra eksterne operationsforstærkere. I programmeringen af PSoC'en er der opnået meget læring ved at skrive et stort program med lav kobling, uden at programmere objektorienteret.

Der er opnået indsigt i hvad det vil sige at udvikle embeddede systemer på en linux platform, og hvilke fordele det medfører. Ydermere er erfaring med kommunikation mellem embeddede platforme opnået.

I hardwareudviklingen er vigtigheden af ikke at presse komponenter til det yderste. Dette gav sig til udtryk ved at den benyttede strømforsyning ikke kunne levere tilstrækkelig effekt til det implementerede system. Derudover er vigtigheden af både enhedstest og integrationstest blevet understreget, og at disse udføres på et tidligt tidspunkt i implementeringsfasen.

13. Fremtidigt arbejde

I forlængelse af projektet er der flere muligheder for, hvordan produktet kan videreudvikles. Mange af ideerne, fremlagt i opgaveformuleringen, er stadig en realistisk del af et endeligt produkt.

Sammen med adgangskortet kan der implementeres en tilhørende adgangskode. Da systemet håndterer penge, vil det give højere sikkerhed i tilfælde af at kortet bliver stjålet. Ulempen ved iden er at brugervenligheden mindskes. Et køb tager længere tid og er mere besværligt at gennemføre, hvis en kode skal indtastes. Ideen kan derfor være et tillægsprodukt, som køberen af systemet kan vælge fra webfronten. Adgangskoden kunne med fordel være den samme som til webportalen, for større brugervenlighed.

Til underholdning af bruger under kaffebrygning kan der implementeres en musikafspiller. Musikken kan bestemmes ud fra forskellige variabler. Den valgte kaffe type og styrke kan eksempelvis bestemme hvilken genre der skal spilles. Derudover kan variabler som brugerens formodede alder og køn have indflydelse.

Ud fra formodningen at sociale medier er en fast del af mange firmaers marketingsstrategi, kan en implementering af sociale medier være en mulighed. Brugerens konto kan eksempelvis kobles til vedkommendes facebook- eller twitterprofil. Når et kaffekøb er gennemført kan brugeren derefter skrive en facebook opdatering eller "tweete" via touchskærmen. Alternativt kunne et spil hvor kollegaer internt kunne dyste om hvem der købte mest kaffe, drak den stærkeste eller havde den største kop.

En app til smartphones, hvor optankning og administration af konto kunne foregå, ville være hensigtsmæssig at implementere. I forlængelse af dette kunne betaling via NFC¹⁰ også muliggøres.

Derudover kan et færdigt system inkludere fast vandforsyning og afløb til spildbakken.

¹⁰ Near Field Communication

14. Konklusion

Formålet med projektet var at udvikle en kaffeautomat, der kunne opstilles på diverse institutioner. Visionen bag produktet var at designe en kaffeautomat der, i modsætning til de gængse automater, er integreret med den pågældende institutions eksisterende IT-services.

I dette projekt er det i høj grad lykkedes at udvikle en repræsentativ prototype, der opfylder denne vision. Det endelige produkt er integreret med IHA's allerede eksisterende IT infrastruktur. Integrationen kommer til udtryk, dels i den tætte kobling med institutionens eksisterende brugerdatabase. Desuden kan der vælge en operatør, som bliver tilknyttet systemet og får notifikationer hver gang en kaffeautomat kræver eftersyn. Produktet implementeret således, at brugeren kan købe kaffe med det adgangskort, vedkommende har fået tildelt af IHA.

Udover kaffeautomaten indeholder produktet en webfront. Webfronten er også integreret med IHA's brugerdatabase, så brugerens login til webfronten er det samme som til skolens øvrige IT-services. Gennem webfronten kan brugeren optanke sin konto, og når brugeren køber kaffe er det denne konto beløbet trækkes fra.

Det er lykkedes at udvikle en kaffeautomat, der kan brygge en kop kaffe. Kaffeautomaten kan dose-re vand og kaffepulver, ud fra brugerens ønske om type, mængde og styrke. Alle basale dele, der er nødvendige for at brygge en kop kaffe er implementeret. Dog er visse features, ikke realiseret.

Store dele af prototypen er på det stadie, at de uden videre vil kunne bruges i et endeligt produkt. Prototypen fremstår som et produkt med et solidt kabinet der indeholder al nødvendig hardware der kræves for at brygge kaffe. Dertil er touchskærmen inkorporeret i kabinetet, så brugeren møder en nemt tilgængelig grafisk brugergrænseflade.

Før et endeligt system kan opsættes på diverse institutioner, er der en række nødvendige forbedringer, der skal implementeres. Det primære problem er at kaffeautomaten trækker en strøm, der ligger på grænsen af hvad den implementerede strømforsyning kan leve. Derudover er det ikke lykkedes at implementere en kopsensor, en vandniveausensor og pulverniveausensorer. Disse sensorer er nødvendige, så brugeren ikke kan købe kaffe når automaten er løbet tør for pulver eller vand. Dertil vil det også mindske systemets downtime, hvis operatøren kan fyldes pulver på kaffeautomaten, før den løber tør. I et færdigt produkt, bør kaffeautomaterne kunne tilsluttes fast vandforsyning.

Alt i alt er en fornuftig prototype implementeret, der demonstrerer et reelt potentiale for et salgsbart produkt.

15. Litteraturliste

Biering-Sørensen, S., Hansen, F. O., Klim, S., Madsen, P. T. (1988). *Håndbog i Struktureret Programudvikling*. Teknisk Forlag.

Friedenthal, S., Moore, A., Steiner, R. (2008) *A Practical Guide to SysML. The Systems Modelling Language*. Burlington, MA: Morgan Kaufmann Publishers.