13-06/2016

VEJLEDER:
HENNING HARGAARD

GRUPPE 15
NIKOLAI TOPPING – 201504495
ANDERS KNUDSEN - 201505249
TONNI FOLLMAN - 201504573
STEFAN NIELSEN - 201508282
MARTIN BURMEISTER - 09408
CHRISTIAN BØEG-JENSEN - 201504504
MIKKEL ESPERSEN - 201507348
DENNIS POULSEN - 20092858

AARHUS SCHOOL OF ENGINEERING

DENNE SIDE ER EFTERLADT BLANK MED VJLE:

Underskrifter

Anders Knudsen _____

Tonni Follmann _____

Stefan Nielsen _____

Martin Burmeister _____

Christian Bøeg-Jensen _____

Mikkel Espersen _____

Nikolai Topping _____

Dennis Poulsen _____

Resumé

Formålet med projektet er at lave et “Home Automation” system til tyveriforebygelse, for at folk kan være trygge, når man er væk fra sit hjem. “Home Automation” er designet til at fungere over hjemmets eksisterende lysnet. Systemet skal være brugervenligt og køre automatisk ud fra en brugerkonfigureret tidsplan, som kan tænde eller slukke en lampe. Systemet er en prototype som er udviklet på et 18V AC net frem for et 230V AC lysnet.

Systemet er opbygget af en styreboks og en eller flere enheder, som er tilkoblet lysnettet. En PC brugergrænseflade bruges til nemt at kunne konfigurere systemet. En kodelås sikrer, at kun autoriserede brugere kan tilgå systemet.

PC softwaren var fuld funktionelt, men den videre kommunikation til styreboksen var ikke funktionel. Der kan sendes data mellem styreboks og enheder, men dette kan ikke omsættes til noget funktionelt.

De forskellige moduler af systemet fungerer hver for sig, men kan ikke implementeres til et samlet fungerende produkt. Prototypen virker ikke i henhold til projekts mål, men vi er ikke langt fra en brugervenlig løsning til tyveriforebyggelse.

Abstract

The purpose of this project is to create a ”Home Automation” system, for theft prevention. This is created in order for people to feel safe while away from home. ”Home Automation” is designed to function on top of the homes existing electrical wiring. The system has to be user friendly, and run automatically based on a user defined time schedule. The system uses the schedule to determine when to turn on or off lighting in the house. The system is a prototype developed on a 18V AC grid, instead of the ordinary 230V AC grid.

The system consists of a control box, and one or more units, these are connected to the house’s electrical grid. A graphical interface enables the use of a PC to configure the system. A code lock ensures that only an authorised user can access the system.

The PC software is fully functional, but the communication to the control box was not functional. Data can be transmitted between the control box and units, but the data is not converted into any actual functionality.

The different modules contained in the system work individually, but can not be implemented in a functional product. The prototype does not fulfill the goals of the project, but is close to a user friendly solution to theft prevention.

1 Indledning

Denne projekt rapport omhandler et ”Home Automation” system.
Dette bygger på et projektoplæg, som forud for projektet oplistede en række krav til det udviklede system. Disse krav lå til grund for valget af ”Home Automation”.

Projektet er udarbejdet i fællesskab mellem 8 personer fra 3 forskellige retninger på Aarhus School of Engineering, herunder IKT, EE, og E på 2. semester.

Formålet med projektet er at udvikle et system, som kan forebygge indbrud ved simulering af aktivitet i et hjem. Dette opnåes ved at udvikle en hardware- og en software-del. Hardwaredelen består af sender og modtager kredsløb, som skal kunne transmittere beskeder ud på lysnettet, og i den anden ende modtage og forstå beskederne. Der er derudover udviklet en grafisk brugerflade, som gør det muligt for en bruger at interagere med systemet.

1.1 Læsevejledning

Nedenstående tabel beskriver hvem der har ansvaret for de enkelte afsnit. Der vil i overskriften for hvert afsnit stå vedkommendes initialer, dette illustreres under Tabel 1.

Hele gruppen	Alle
Tonni Follmann	TF
Stefan Nielsen	SN
Martin Burmeister	MB
Christian Bøeg-Jensen	CBJ
Dennis Poulsen	DP
Mikkel Espersen	ME
Anders Knudsen	AK
Nikolai Topping	NT

Tabel 1 Liste over Initialer

1 Overskrift (Alle)

1.1 Overskrift

1.1.1 Overskrift

1.1.1.1 Overskrift

Ovenstående eksempel viser at når alle har været med uddybes det ikke i hvert underafsnit.

1 Overskrift (TF, AK)

1.1 Overskrift (AK)

1.1.1 Overskrift (TF)

1.1.1.1 Overskrift (TF, AK)

Ovenstående eksempel viser forfatter opdelingen for hvert afsnit.

1.2 Ordliste

Slack	Kommunikationsprogram. Indeholdte kanaler til kommunikation mellem grupper, midlertidig fildeling, kalender opdateringer samt opdateringer vedrørende ændringer i GIT repositories.
Google Calendar	Brugt til organisering af gruppens møder o.lign.
Github	Bruges til versionsstyring og fildeling.
ASE	Aarhus School og Engineering
EE	Elektrisk Energiteknologi
E	Elektronik
IKT	Informations- og Kommunikationsteknologi
X10.1	Protokol benyttet i projekktet. Baseret på X10 protokollen.
SysML	System Modelling Language
UML	Unified Modelling Language
UART	Protokol til seriel kommunikationt
VHDL	Very High Speed integrated circuit Hardware Description Language.
BDD	Block Definitions Diagram
IBD	Internt Block Diagram
CPU	Central Processing Unit
SD-Kort/SD-Card	Non-volatil hukommelse til lagring af data.
SPI	Serial Peripheral Interface. Specifikation for kommunikation af seriel data.
I2C	Inter-Integrated Circuit. Protokol til kommunikation mellem integrerede kredsløb.
RTC	Real-Time Clock. Benyttes til at kontrollerer klokkeslet og dato.
GUI	Grafisk Brugerflade.
LCD	Liquid Crystal Display. Display til visning af forskellige informationer.
FURPS	Functionality, Usability, Reliability, Performance and Support
QT	Værktøj til udvikling af grafisk brugerflade
PWM	Pulse Width Modulation.

2 Projektformulering og Afgrænsning (Alle)

2.1 Projektformulering

Opgaven i dette projekt er at udvikle et ”Home Automation” system til tyveriforebyggelse. Systemet skal kunne simulere, at der er aktivitet i huset. F.eks. hvis man er på ferie, så kan en simuleret aktivitet i hjemmet være med til at forebygge indbrud. Tal fra Danmarks statistik viser, at der i hele 2015 blev anmeldt 32.974[1] indbrud i beboelses ejendomme i Danmark, mens der i samme periode blev anmeldt 2005[2] røverier. Politiet anbefaler som forebyggelse, at man får sit hjem til at se ”levende” ud, når man rejser fra det[3]. Dette vil ofte afskrække mange indbrudstyve, fra at bryde ind i lige netop det hjem. Fordelen ved et ”Home Automation” system fremfor bare at lade lyset være tændt er netop, at man får fornemmelse af, at der er liv i huset.

Baseret på ”Home Automation” konceptet vil vi udvikle et system, der fungerer ved kommunikation over hjemmets eksisterende lysnet. Derved skal der ikke trækkes ekstra kabler rundt i hjemmet. Systemet vil være baseret på X.10 protokollen, som bliver tilpasset til vores system. Systemet skal kunne køre automatisk ud fra en brugerstyret tidsplan, som brugeren skal have mulighed for at oprette og ændre.

Tidsplanen skal gemmes på et SD-Kort, så tidsplanen bevares i tilfælde af strømsvigt. Systemet skal have en LCD-skærm, hvor information om kritiske systemfejl, kan meddeles til brugeren. Da systemet skal kunne være kørende, mens brugeren af systemet er på ferie, vil vi lave så systemet kan genstarte sig selv i tilfælde af det vi kalder kritiske systemfejl. I første omgang omfatter ”Kritiske systemfejl”, når der er kommunikationsfejl over lysnettet, men dette kunne i senere udvidelser af systemet udvides til også at omfatte andre fejltilstande.

Systemet skal være i stand til at registrere hvis der opstår kommunikationsfejl, logge disse fejl, og skrive en fejlmeddelelse med systemets status på et LCD display. Brugeren skal desuden få præsenteret en oversigt over fejl, der har været siden sidste PC-tilkobling.

I en færdig prototype bør der være fokus på brugervenligheden, da det bl.a. skal kunne bruges af personer uden høj teknisk kunnen. Der skal også være fokus på transmissionshastigheden af data i mellem PC og X.10 controller, så forbrugeren ikke skal have unødvendige lange ”load” tider.

[1] Tal fra Danmarks statistik: http://www.dst.dk/da/Statistik/NytHtml?cid=20617

[2] Tal fra Danmarks statistik: http://www.dst.dk/da/Statistik/NytHtml?cid=20617

[3] https://www.politi.dk/da/servicemenu/baggrund/FAQ_indbrud_12072007.htm

2.2 Projektafgrænsning

I det udleverede projektoplæg til 2. semesterprojektet er der opstillet nogle krav til projektets indhold. Disse krav omfatter:

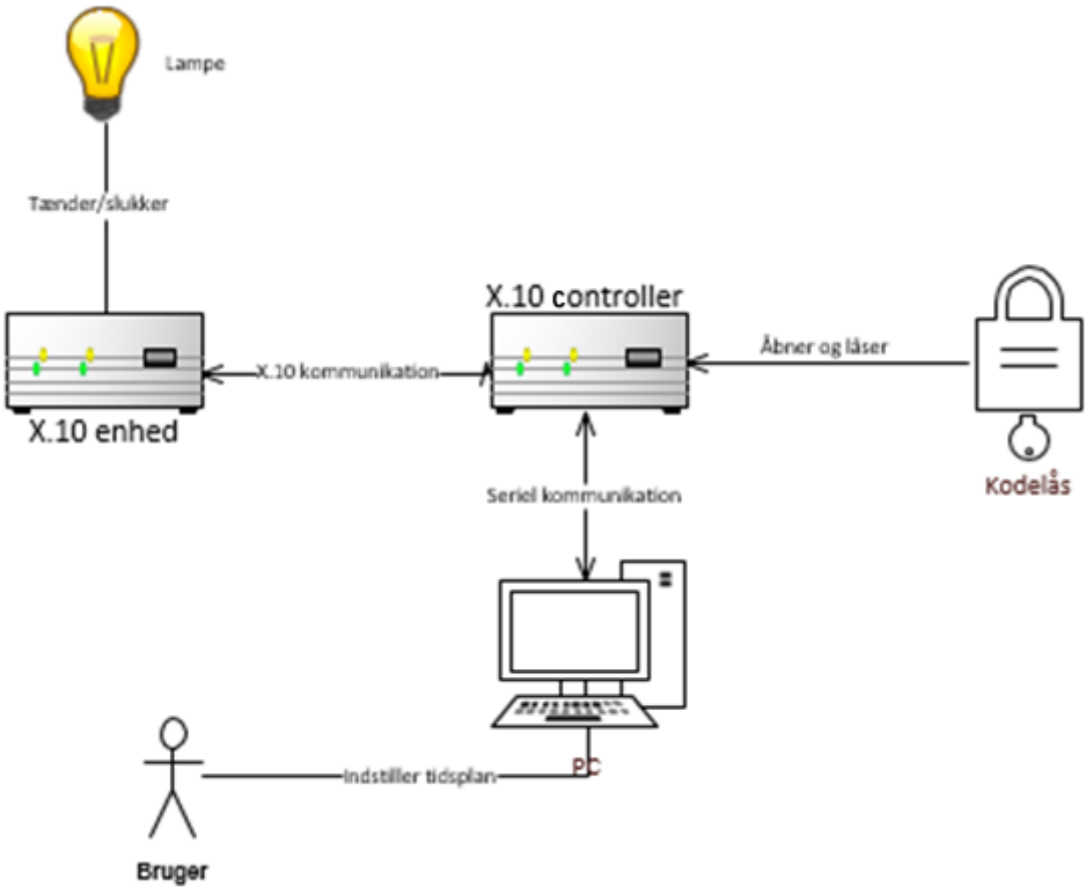
- Udviklingsboards: Arduino Mega2560 og DE2-Board
- 18 VAC/500 mA strømforsyning til simulering af lysnet
- Kommunikationen over lysnettet skal være baseret på X.10 protokollen
- Kommunikation mellem PC og X.10 controller skal være seriel f.eks. UART

Ud fra dette skal der udvikles et ”Home Automation” system (Figur 1), der kan kommunikere ved X.10 baseret kommunikation.

Det er valgfrit om man vil lave en- eller tovejs kommunikation. Det er blevet besluttet at lave tovejs kommunikation.

ASE tillader ikke studerende at arbejde med 230V til 2. semesterprojektet, derfor udvikles prototypen til at fungere med en 18VAC strømforsyning, som udleveres på værkstedet.

Ved fuld udvikling skal systemet fungere over et lysnet på 230V, som er standarden for lysnettet i Danmark.



Figur 1 - Illustration af "Home Automation".

3 Systembeskrivelse (DP, AK)

Systemet består af flere dele, som bliver beskrevet i de følgende afsnit:

Kodelåsen skal laves på et DE2-udviklingsboard, koden er allokeret i programmeringssproget VHDL. DE2-udviklingsboardet er udleveret af ASE. Selve kodelåsen programmeres som en del af undervisningen i faget Digitalt System Design. De ændringer der er foretaget i forhold til undervisningsmaterialet, vil blive beskrevet i afsnit 5.5.1.

X10.1 controlleren skal designes og udvikles i løbet af projektet. X10.1 controllerne skal bruge en Arduino Mega2560 som microcontroller og skal kunne kommunikere med en PC ved serielkommunikation. Der skal udvikles en modtagerdel og en senderdel til controlleren. X10.1 controlleren skal sende kommandoer til X10.1 enhederne over et lysnet, der simuleres med en 18 VAC strømforsyning udlånt af ASE.

X10.1 enhed/-er skal designes og udvikles i løbet af projektet. X10.1 enhederne skal bruge en Arduino Mega2560 som microcontroller. Der skal udvikles en modtagerdel og en senderdel til enhederne. X10.1 enhederne skal kunne modtage og besvare kommandoer sendt fra X10.1 controlleren, over et lysnet, der simuleres med en 18 VAC strømforsyning udlånt fra værkstedet. X10.1 enhederne kan tænde eller slukke for en lampe, ud fra den kommando der modtages fra X10.1 controlleren.

PC er brugerens grænseflade for at konfigurere "Home Automation". Softwarens formål er at give brugeren mulighed for let at kunne konfigure systemet. Dette skal gøres via en brugergrænseflade, som vil blive implementeret i programmet QT Creator.

For at få adgang til brugergrænsefladen vil der være en kodelås. Dette er for at sikre at ingen uautoriserede brugere har adgang til systemet. Autoriserede brugere vil kunne konfigurere systemet via brugergrænsefladen, hvor der vil kunne tilføjes og fjernes enheder, samt tilføje en tidsplan for de enkelte enheder. Det er denne tidsplan, som bestemmer hvornår en enhed skal være aktiv eller ej. Brugeren vil kunne indstille en tidsplan for alle ugens dage i et 24 timers interval. Når der er foretaget en ændring, vil de enkelte enheder og tidsplaner blive overført til styreboksen via serielkommunikation, som herefter vil behandle dataen.

Der vil ikke oprettes et permanent lager for enheder og tidsplaner på PC'en selv. Når PC softwaren opstartes, vil en funktion hente alt data om enheder og tidsplaner fra styreboksen. De enkelte enheder og tidsplaner vil herefter blive vist på softwaren, som brugeren herefter vil kunne konfigurere.

Validering af brugerens input vil blive foretaget via softwaren, som vil sikre at der maksimalt vil kunne blive konfigureret op til 255 enheder, hvor hver enhed vil kunne have op til 70 tidsplaner. Tidsplaner vil blive delt ud, så der kun kan konfigureres 10 tidsplaner for hver af ugens 7 dage.

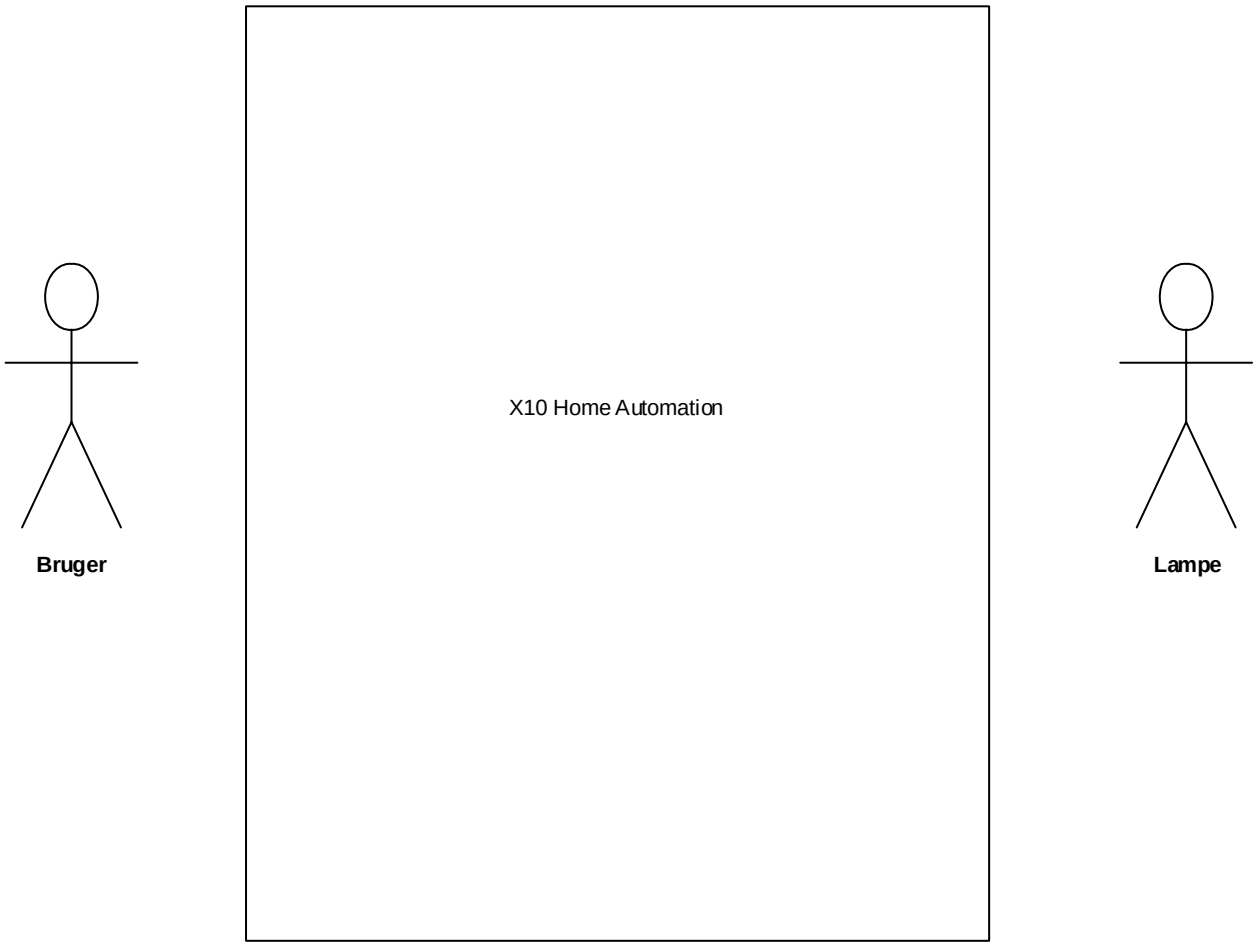
Prototypen vil bestå af en udviklet X10.1 controller, og to udviklede X10.1 enheder. Disse skal kommunikere via X10.1 kommunikation over et simuleret lysnet. I en færdig model vil controller og enheder kunne tilsluttes en stikkontakt på et 230 V lysnet.

4 Kravspecifikation (TF)

Ud fra en analyse af systembeskrivelsen, samt projekt afgrænsning, er der i gruppen blevet udarbejdet en kravspecifikation indeholdende både funktionelle og ikke funktionelle krav. Under kravspecifikation er der udarbejdet 9 use cases, som beskriver brugerens interaktion med systemet.

4.1 Aktør-kontekst

På Figur 2 ses et aktør diagram, der viser de aktører der har indflydelse på systemet.



Figur 2 – Aktør-kontekst diagram

4.2 Aktør beskrivelser

4.2.1 Bruger

Brugeren er den aktør der ønsker at benytte systemet. Brugeren har kendskab til koden til kodelåsen der kræves for konfiguration og betjening af systemet, og er den aktør der er ansvarlig for konfiguration af systemet.

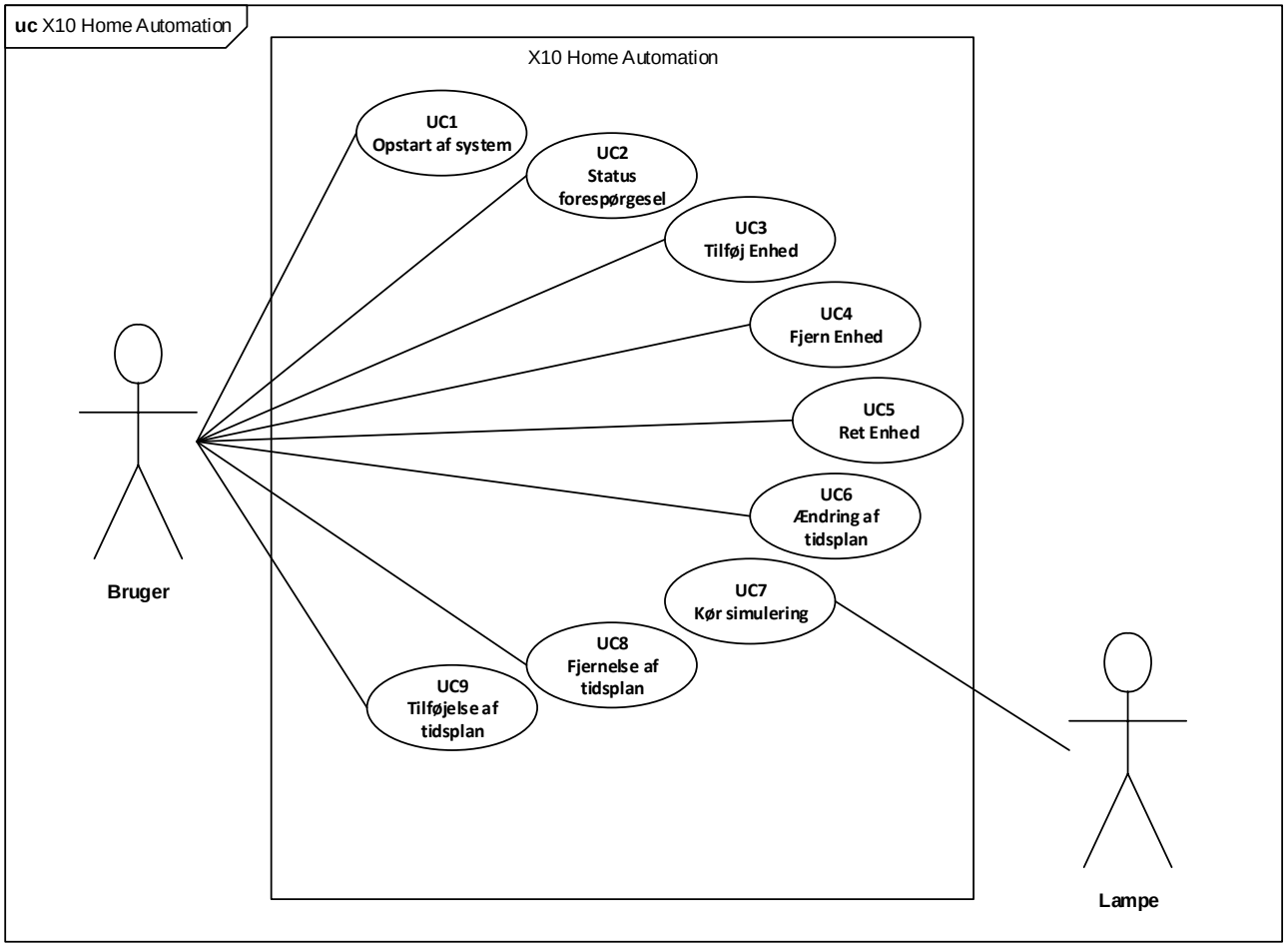
4.2.2 Lampe

Lampe er en sekundær aktør, der er forbundet til systemet. Lampen tændes og slukkes af systemet.

4.3 Funktionelle krav

De funktionelle krav beskriver systemets ønskede funktionalitet som en serie af use cases, der specificerer brugerens interaktion med systemet ved hjælp af diagrammer.

På Figur 3 ses use case diagrammet, der viser hvilke aktører, der interagerer med de enkelte use cases.



Figur 3 – Use case diagram over aktørenes interaktion med de enkelte use cases

Der følger her en kort beskrivelse af de enkelte use cases, ønskes der en fully dressed version af use casen kan denne findes i projektdokumentationen.[4]

[4] Side 8, afsnit 2.4

4.3.1 Use case 1: Opstart af System

Use casen beskriver det første scenarie som brugeren møder ved anvendelse af systemet. Scenariet startes ved at brugeren starter det medfølgende PC software, hvorefter PC softwaren anmoder brugeren om at indtaste koden til systemet. Når brugeren har indtastet den korrekte kode, vil PC softwaren vise forsiden i den grafiske brugerflade.

Udvidelserne til denne use case sikrer at der kun kan køre en instans af softwaren af gangen, og sikrer samtidig at en forkert indtastning af koden håndteres ved at vise en fejlmeddelelse.

4.3.2 Use case 2: Status Forespørgsel

Anvendes af brugeren til at få en oversigt over de tilsluttede enheders nuværende status. Dette gøres ved at brugeren trykker "Opdater Enhedsstatus", hvorefter systemet vil vise en opdateret oversigt over status på alle tilsluttede enheder.

4.3.3 Use case 3: Tilføjelse af Enhed

Beskriver hvordan brugeren tilføjer en ny enhed til systemet. Dette gøres ved at brugeren tilslutter den nye enhed til lysnettet, hvorefter enheden oprettes i systemet ved hjælp af den grafiske brugerflade. I forbindelse med oprettelsen af enheden i systemet, vil der blive givet mulighed for at tildele enheden et rum, eller annullere.

Til at håndtere de forskellige indtastningsmuligheder er der en række udvidelser til hovedscenariet som håndterer de scenarier hvor brugeren ikke indtaster et rum, eller håndtering af at enhedsadressen kolliderer med en eksisterende enhedsadresse.

4.3.4 Use case 4: Fjernelse af Enhed

Specificere hvordan brugeren kan fjerne en eksisterende enhed fra systemet via brugergrænsefladen. For at kunne håndtere forskellige indtastningsfejl eller at brugeren ombestemmer sig, er der lavet to udvidelser, der håndterer annullering fra brugerens side, samt at brugeren ikke har valgt en enhed der ønskes fjernet fra systemet.

4.3.5 Use case 5: Rediger Enhed

Rediger Enhed giver brugeren mulighed for at ændre enhedens adresse samt hvilket rum enheden er en del af i systemet.

Der er tilføjet en række udvidelser for at give brugeren mulighed for at annullere, samt at vælge kun at ændre rum eller enhedens ID.

4.3.6 Use case 6: Ændring af tidsplan

Ændring af tidsplan giver brugeren mulighed for at ændre en enheds tidsplan så en enhed kan indstilles til at tænde og slukke for en lampe på specifikke tidspunkter. Dette gøres ligeledes via den grafiske brugerflade, hvor brugeren får mulighed for at vælge ugedag samt start- og sluttidspunkt for en allerede tilføjet enhed i systemet.

En række udvidelser håndterer ikke-korrekt brugerinput, samt andre mulige problemer i løbet af hovedscenariet.

4.3.7 Use case 7: Kør simulering

Kør Simulering beskriver hvordan systemet kan tænde og slukke for lamper der er forbundet til enheder på de bruger angivne tidspunkter.

4.3.8 Use case 8: Fjernelse af tidsplan

Use casen specificerer hvordan brugeren kan fjerne en tidligere konfigureret tidsplan fra en enhed, når denne ikke længe ønskes anvendt i forbindelse med use case 7.

En kombination af hovedscenariet med en række udvidelser, giver muligheden for at håndtere forskellige fejlscenarier, så disse kan håndteres på en brugbar måde.

4.3.9 Use case 9: Tilføjelse af tidsplan

Use casen giver brugeren mulighed for at oprette en ny tidsplan til en eksisterende oprettet enhed. Dette gøres via den grafiske brugerflade.

Muligheden for forkert brugerinput eller forkert status på systemet håndteres ved hjælp af en række udvidelser der informerer brugeren vha. fejlmeddelelser.

4.4 Yderligere tekniske krav

De yderligere tekniske krav er udarbejdet fælles i gruppen. Der er fra gruppens side specificeret tekniske krav specifikt for styreboksen og enheden, der definerer stiktyper, LED farver mv.

Derudover er der udarbejdet en række ikke-funktionelle krav, der definerer de kvalitetskrav, som stilles til systemet.

For oplysninger om de yderligere tekniske krav kan de forefindes i den vedlagte projektdokumentation[5].

[5] Side 17, afsnit 2.5

5 Projektbeskrivelse

5.1 Projektgennemførelse (SN)

Dette projekt er gennemført som gruppearbejde. I projektets indledende faser blev der i fællesskab udarbejdet en samarbejdskontrakt[6], samt en overordnet filosofi for ambitionsniveauet i projektgruppen.
Der blev også aftalt at den interne kommunikation i projektgruppen skulle foregå via Slack, som kan forbindes både med Google-Kalender og Github, således at al aktivitet i projektgruppen ville fremgå i kommunikationsplatformen.
Der blev desuden aftalt at have en fast projektleder samt suppleant, og fast mødeleder samt suppleant, mens det blev aftalt at referent-rollen skulle gå på skift mellem gruppemedlemmerne.
Projektlederens rolle har været at holde styr på projektets retning, planlægning og tidsplan, mens mødelederens rolle har været at sørge for mødeindkaldelser, dagsorden til møder, samt planlægning af møder med vejleder.
Stefan Nielsen har fungeret som projektleder, med Tonni Follmann som suppleant.
Nikolai Topping har fungeret som mødeleder, med Anders Knudsen som suppleant.

Der blev i fælleskab mellem projektleder og suppleant udarbejdet en tidsplan med fokus på afleveringsfrister, som i fælleskab med de øvrige indgåede aftaler, skulle danne grundlag for planlægningen af arbejdet i projektgruppen[7].

Projektet er blevet udviklet med ASE´s udviklingsmodel for 2. semester projekt, som er vist på Figur 4.
Denne udviklingsmodel indeholder følgende faser:

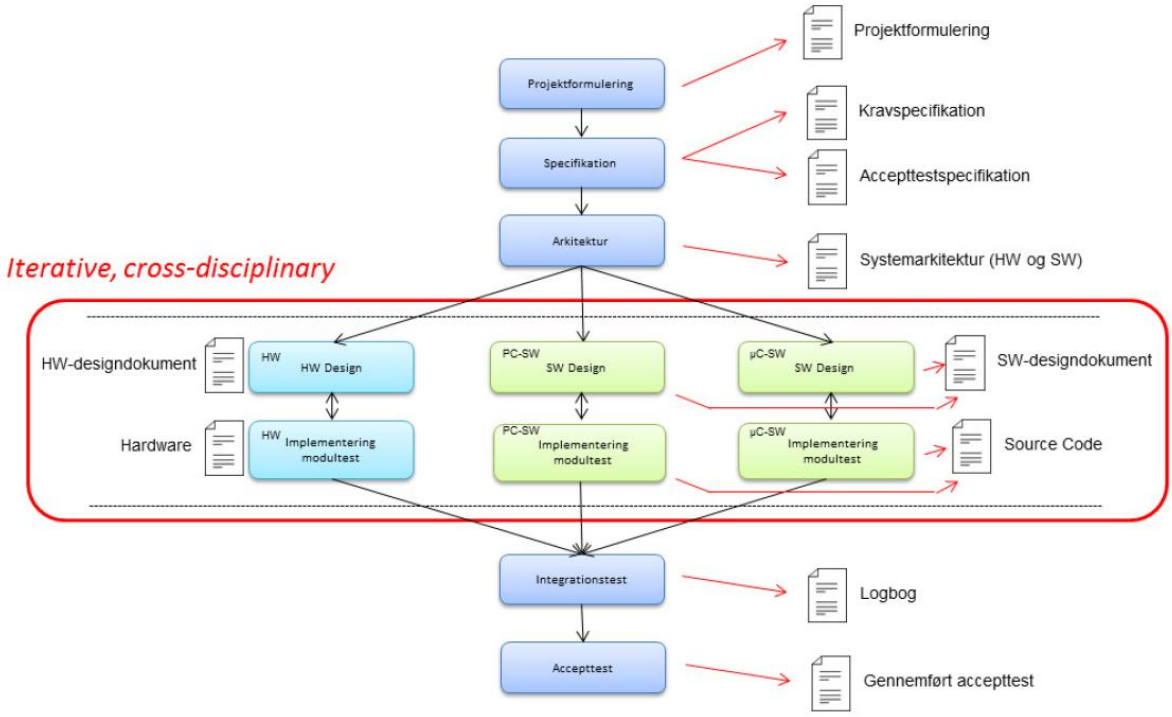
- Projektformuleringsfasen.
- Specifikationsfasen.
- Arkitekturfasen.
- Design- og implementeringsfasen.
- Accepttestfasen.

I de to første faser af udviklingen, arbejdede hele projektgruppen sammen om at få det ønskede system specificeret.
Herefter opdelte gruppen sig i 3 mindre grupper.
Grupperne er som følger:

- PC gruppe: Nikolai Topping, og Anders Knudsen.
  Hovedansvar: At lave PC-Software.
- Styreboks gruppe: Tonni Follmann, Dennis Poulsen og Stefan Nielsen.
  Hovedansvar: At lave hardware og software til styreboksen.
- Enheds gruppe: Martin Burmeister, Mikkel Espersen og Christian Bøeg-Jensen.
  Hovedansvar: At lave hardware og software til enheder.

Arbejdet i de forskellige faser af udviklingen, beskrives yderligere i rapporten.

[6] Se bilag: ”Samarbejdskontrakt”
[7] Se bilag: ”Tidsplan”



Figur 4 - ASE´s udviklingsmodel.[8]

5.2 Metode (SN)

I det følgende gives et overblik over de metoder som er anvendt i projektets specifikationsfase og arkitekturfase.

5.2.1 Use cases

Vi har i specifikationsfasen anvendt use case analyse, som et primært værktøj til at bestemme systemets funktionelle krav.

Use case analysen tager udgangspunkt i projektformuleringen, som bruges til at identificere systemets primære- og sekundære aktører. Derefter identificeres alle de opgaver eller målsætninger som aktørerne ønsker systemet kan opfylde.

Der laves en use case som beskriver formålet med opgaven, hvordan systemet skal løse den opgave (hovedscenariet - set fra brugerens synspunkt), eventuelle afvigelser i forhold til hovedscenariet (udvidelser), samt forudsætninger for at use casen kan udføres.

5.2.2 FURPS

Til at kortlægge systemets yderligere tekniske krav, blev der primært anvendt analysemetoden FURPS.

Ved at gennemgå de 5 kategorier en efter en, og stille spørgsmål til hvilke krav systemet skal opfylde inden for den pågældede kategori, kan man således identificere de yderlige tekniske krav for systemet, som ikke kom frem under use case analysen.

[8] Se bilag: ”Vejledning for gennemførelse af projekt 2”

5.2.3 SysML

I arkitekturfasen er anvendt SysML værktøjerne:

5.2.3.1 BDD – Block Definition Diagram

Når der laves et BDD for et system, opdeles systemet i mindre hardware-blokke, som er nødvendige for at opfylde systemets krav.
Et BDD giver et visuelt overblik over systemets del-komponenter, og hvilke del-komponenter der arbejder sammen, og dermed må have forbindelse til hinanden.

5.2.3.2 IBD – Internal Block Diagram

IBD´er er brugt til at danne overblik over de forskellige blokkes forbindelser, og signaltyper.

5.2.4 UML

I projektets arkitekturfase er der anvendt domæneanalyse til at identificere systemets klasser, deres indbyrdes relationer, og klassernes individuelle funktioner.
Systemets use cases gennemgåes, og gennem navneordsanalyse identificeres konceptuelle klasser, derefter identificeres klasser som boundary- domæne- eller kontrolklasser.

5.2.4.1 SD – Sequence Diagram

Sekvens diagrammerne hjælper med at identificere funktioner, som så kan indsættes i de klasser som er identificeret ved domæne analysen, og på den måde kan man komme frem til et samlet (indledende) UML klasse diagram for systemet.

Der blev først udfærdiget en matrice, som viste hvilken CPU (styreboks, enhed eller PC) der var en del af hver use case, derefter blev der lavet sekvensdiagrammer for hver CPU, for alle de use cases som den CPU indgik i, og deres udvidelser.

5.3 Specifikation og Analyse (MB, ME)

Analysearbejdet tog udgangspunkt i projektoplægget som blev givet af ASE. Projektoplægget lagde op til ”Home Automation” og dette blev valget. Forskellige emner blev foreslået, men efter tids diskussion blev alle gruppemedlemmer enige om at holde sig til det ASE havde lagt op til. Valget på Home Automation blev gjort fordi andre emner var for abstrakte og der opstod misforståelser mellem gruppemedlemmer.

I projektoplægget er ASE’s udviklingsmodel blevet fulgt. I faserne projektformulering og specifikation blev der fundet frem til hvordan slutproduktet skulle se ud, ud fra de krav som oplægget stillede. Systemets egenskaber blev specificeret ud fra overvejelser om tid og faglig kunnen. Gruppen var fast besluttet på at bygge systemet med 2-vejs kommunikation mellem Styreboks og Enhed. Der blev fundet frem til at bygge systemet med en sender- og modtagerdel til både Styreboks og Enhed.

Ud fra forskellige metoder at implementere 2-vejs kommunikation faldt valget på master/slave fordi dette er simpelt at implementere, og krævede en minimal mængde fejlhåndtering.

Gruppens retning blev fastlagt og fundamentet for projektet var lagt. Use cases og resten af krav- samt accepttestspecifikationen blev beskrevet og dette bragte gruppen videre til Arkitekturfasen.

Arbejdsfordelingen blev opdelt mellem 3 mindre grupper da ”Systemarkitekturfasen” blev indledt. Beslutningen blev truffet for at fordele arbejdsbyrden ligeligt og samtidigt skabe en generel sammenhæng omkring de emner, de enkelte grupper skulle arbejde med.

Design- og Implementeringsfaserne foregik iterativt hvor udkast til software- og hardwaredesigns blev implementeret og optimeret løbende. AK og NT arbejdede i disse faser ud fra SCRUM-udviklingsmodellen hvor de udviklede den grafiske brugerflade. De to resterende grupper havde mangel på struktur. Hele gruppens manglende struktur var stærkt medvirkende til at integrationstesten aldrig blev foretaget. Accepttesten blev foretaget selvom prototypen ikke var færdigudviklet.

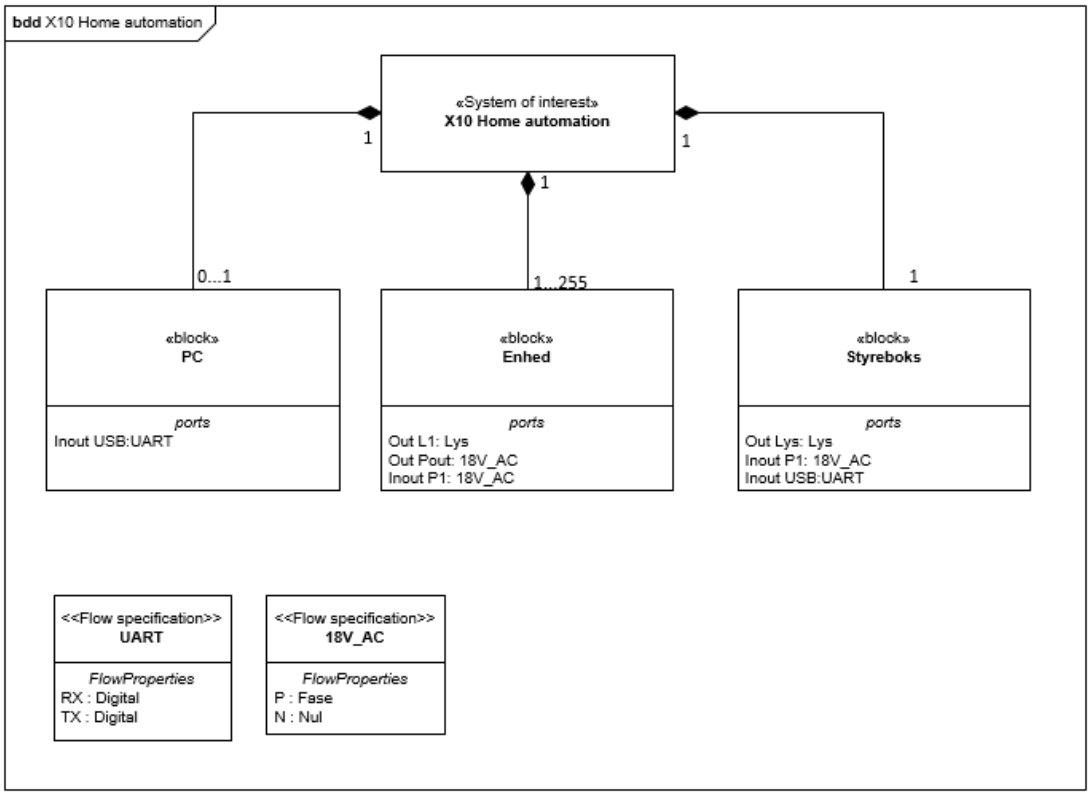
5.4 Arkitektur

I de følgende afsnit beskrives arkitekturen for hardware og software.

5.4.1 Hardwarearkitektur (DP)

Arkitekturen for hardwaren er designet af hardwaregruppen. Arkitekturen består af BDD'er, IBD'er, blokbeskrivelser og signalbeskrivelser for det system, der skal designes.

På Figur 5 ses det overordnede BDD for systemet. Systemet består af tre blokke. Hver af blokkene "Enhed" og "Styreboks" er den hardware, der skal designes kredsløb for. For mere detaljerede BDD'er for "Enhed"[9] og "Styreboks[10]" henvises til projektdokumentationen.

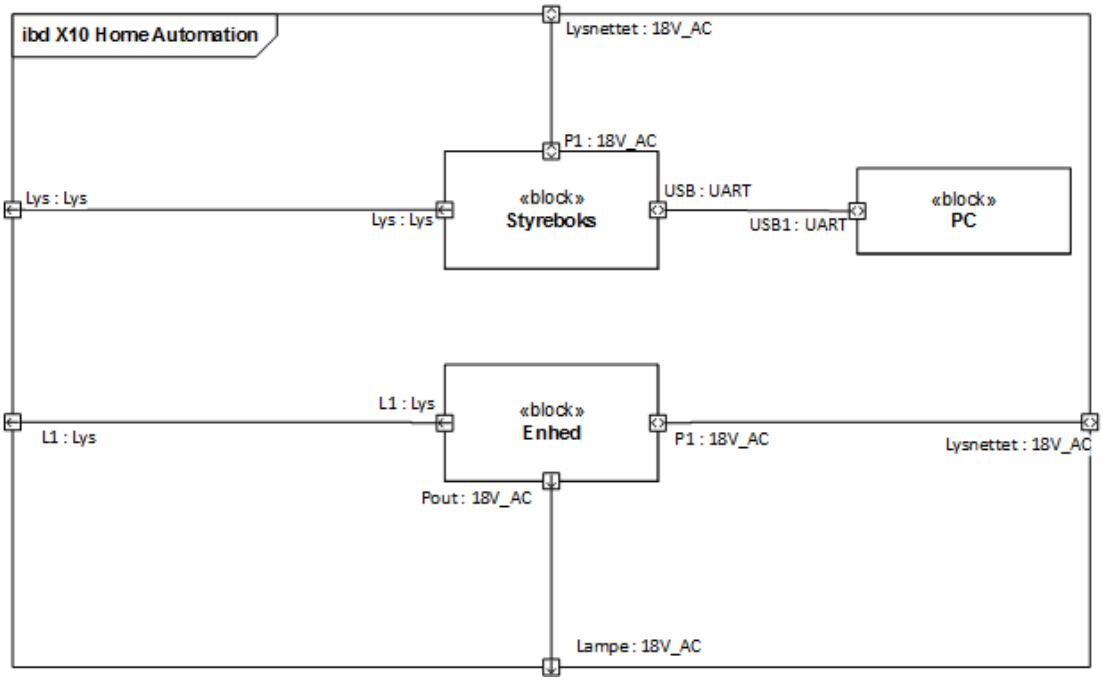


Figur 5 - BDD for systemet

Udover BDD'er laves også IBD'er, da disse viser hvordan de individuelle forbindelser i mellem blokkene er koblet sammen.

[9] Projektdokumentation side 31, afsnit 3.3
[10] Projektdokumentation side 19 afsnit 3.1

Figur 6 viser IBD for det overordnede system. Her ses de forskellige in- og outputs og forbindelserne i mellem disse. For mere detaljerede IBD'er for "Enhed"[11] og "Styreboks"[12] henvises til projektdokumentationen, her findes også signalbeskrivelser.



Figur 6 - IBD for systemet

Tabel 2 viser signalbeskrivelserne for forbindelserne i "IBD for systemet".

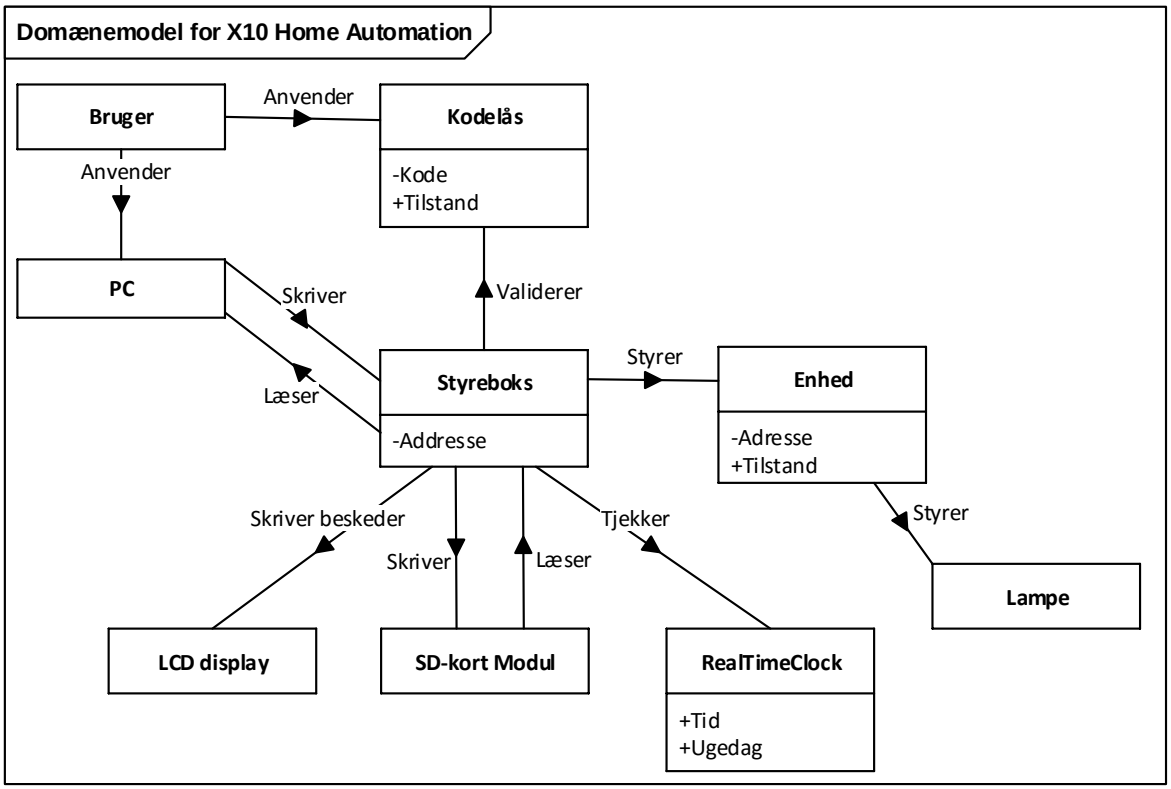
Signaltype	Definition	Beskrivelse
18V_AC	18V AC 50 Hz signal kombineret med et 120kHz X10.1 signal	18V AC 50 Hz signal fra forsyningsnettet der også indeholder kommunikationen via X10.1 protokollen der udvikles specifikt til dette produkt, se protokol afsnit for yderligere information.
Lys	Lys i det synlige spektrum	Lys i 3 farver afhængig af hvilket LED indikator der lyser.
UART	Kommunikation følger UART-standarden	Protokollen udvikles specifikt til dette produkt. Se protokol afsnit for yderligere information.

Tabel 2 - Signalbeskrivelser for "IBD for systemet"

[11] Projektdokumentation side 32, afsnit 3.3.1
[12] Projektdokumentation side 23, afsnit 3.2.1

5.4.2 Softwarearkitektur (AK, NT, DP)

Softwaren er udviklet af to softwaregrupper. En gruppe har designet softwaren for PC'en og en anden gruppe har designet softwaren for "Styreboks" og "Enhed". Softwaren tager udgangspunkt i systemets tre CPU'er. På Figur 7 ses den domænemodel, der er udarbejdet for systemet.



Figur 7 - Domænemodel for systemet

Systemets tre CPU'er: PC, Styreboks og Enhed, indeholder data og programkode der skal bruges til eksekvering af deres givne opgaver. Disse bliver derfor identificeret som domain klasser for systemet. Da der arbejdes med et distribueret system, udarbejdes individuelle applikationsmodeller for systemets CPU'er.

Til disse applikationsmodeller bruges domænemodellen til at identificere modellernes klasser: Boundary-, Controller- og Domain klaser. Ved brug af layering identificeres de forskellige lag som de valgte klasser opererer på. Lagene indeles i grænseflade-, lagring-, og logisk lag.

Grænseflade laget beskriver systemets interaktion mellem de forskellige blokke, samt forbindelse til udvendige aktører. Et eksempel på en klasse der opererer på dette lag ville være interface klassen mellem Styreboks og PC.

På lagrings laget ligger de klasser, der har til formål at lagre data. SD-Kort Modul er et eksempel på en sådan klasse.

Det logiske lag er klasser, der eksekverer de ønskede handlinger, som beskrives i de enkelte Use Cases. Disse klasser vil opstå som control klasser. I applikationsmodellerne vil der blive oprettet én control klasse per use case, for de use cases som den givne CPU bliver anvendt i.

5.4.2.1 Overordnede Sekvensdiagrammer

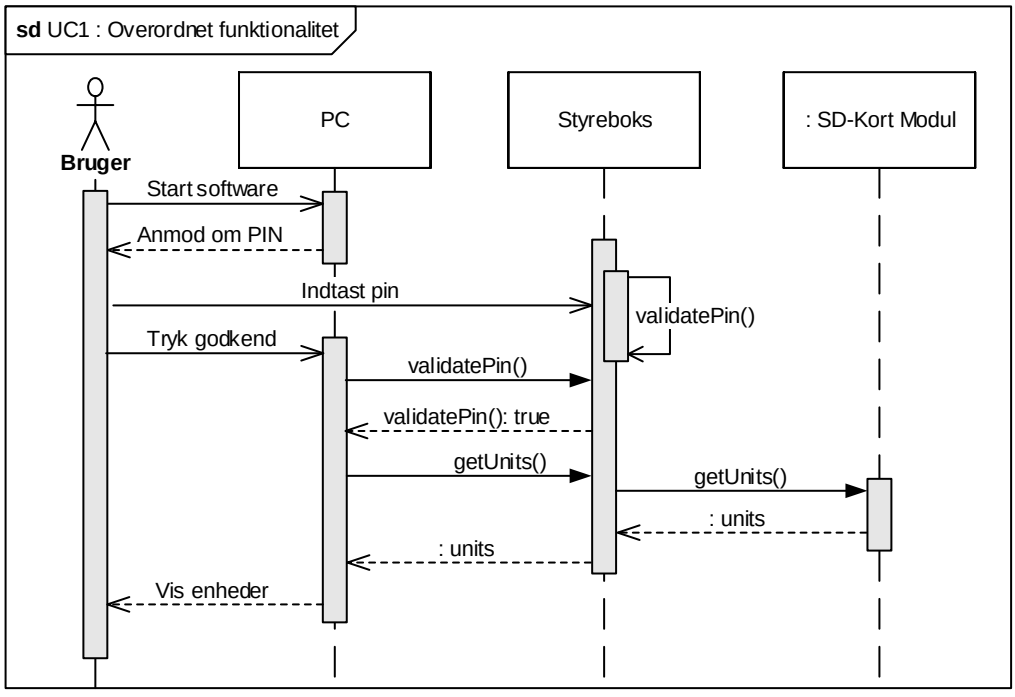
For at danne et overblik over hvilke control klasser der er relevante for de individuelle CPU'er lavede vi en matrice over use cases og deres tilhørende CPU. Matricen vises i Tabel 3, hvor et "X" repræsenterer, at en CPU deltager i den viste use case.

USE CASE/CPU	PC SOFTWARE	STYREBOKS	ENHED
UC1	X	X	
UC2	X	X	X
UC3	X	X	
UC4	X	X	
UC5	X	X	
UC6	X	X	
UC7		X	X
UC8	X	X	
UC9	X	X	

Tabel 3 Use case / CPU Matrix

Før der udfærdiges applikationsmodeller for de enkelte CPU'er, benyttes den ovennævnte matrice til opsætning af overordnede sekvensdiagrammer for hver enkelt use case. Disse bruges til at få et overblik over hvordan og hvornår der vil blive kommunikeret mellem hver CPU via deres grænseflader. Et eksempel på et sådant sekvensdiagram vises på Figur 8.

Sekvensdiagrammet er påført første udkast til nødvendige metodekald. Disse metodekald bliver videre specificeret eller ændret i arkitekturfasen.



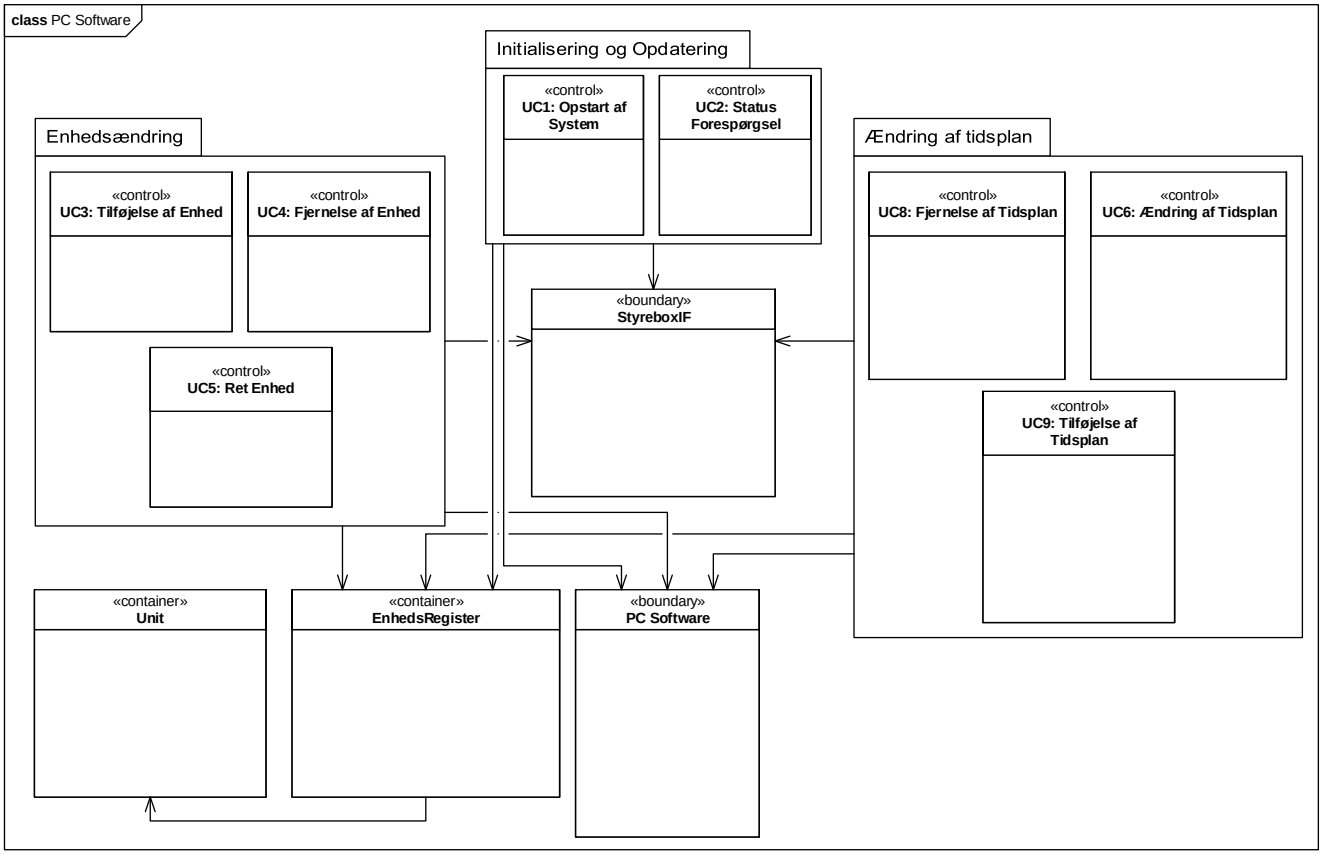
Figur 8 Overordnet sekvensdiagram for use case 1

Efter udfærdigelse af den overordnede arkitektur er der blevet opstillet et overblik over hvilke behov der er for kommunikation mellem CPU'er, og der kan derfor arbejdes på software arkitektur individuelt for hver CPU. Den følgende arkitektur beskrivelse er derfor delt op mellem de to software grupper.

5.4.3 Softwarearkitektur for PC (AK, NT)

Fra klasseidentifikationen og grænsefladerne vist i de overordnede sekvensdiagrammer dannes en applikationsmodel for softwaren der ligger på PC'en.

Som første led opsættes et tomt klassediagram. Dette klassediagram indeholder de identificerede klasser, og sammenhængen mellem dem. Disse inddeles videre i pakker for de klasser hvis funktion ligner hinanden. Det tomme klassediagram vises på Figur 9.

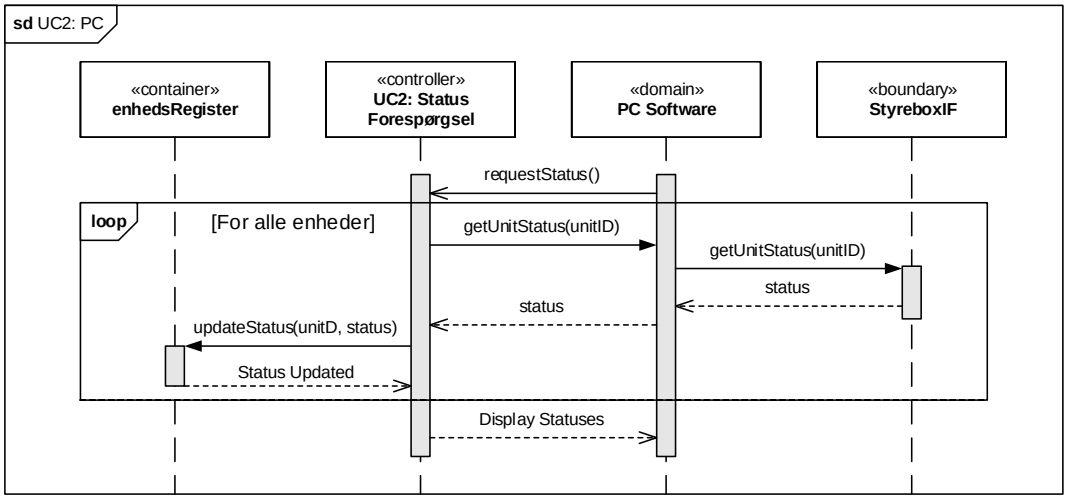


Figur 9 Tomt klassediagram for PC software

På baggrund af use cases skrives sekvensdiagrammer, der repræsenterer handlingerne, der ønskes fra de enkelte klasser. Disse sekvensdiagrammer bruges til at skabe en oversigt over det ønskede handlingsforløb, ved at identificere metoder baseret på de beskeder, der sendes mellem klasserne.

Et eksempel på sådan et sekvensdiagram ses på Figur 10. Herfra identificeres metoder som indskrives på klassediagrammet. Disse sekvensdiagrammer blev igennem design og implementations fasen opdateret løbende, for bedre at reflektere programmets forløb. For de færdige sekvensdiagrammer refereres der til projektdokumentationen.[13]

[13] Side 47, afsnit 4.5



Figur 10 Sekvensdiagram for use case 2 : PC

Klassediagrammet blev opdateret efter hvert færdiggjort sekvensdiagram, hvor det blev fyldt med metoderne som blev identificereret. Disse metoder vil senere blive brugt i implementeringen af softwaren. Det fyldte klassediagram på PC softwaren vil kunne ses i projektdokumentationen[14]. På klassediagrammet kan der ses klassen container, som kan sammenlignes med domain, hvor her blot benyttes STL bibliotekerne. Efter endt arkitektur kunne design og implementering fasen begyndes.

5.4.4 Softwarearkitektur for Styreboks (TF)

Ud fra klasseidentifikationen og de overordnede sekvensdiagrammer samt use cases er der for styreboksen udarbejdet en applikationsmodel indeholdende et klassediagram samt sekvensdiagrammer for de enkelte use cases.

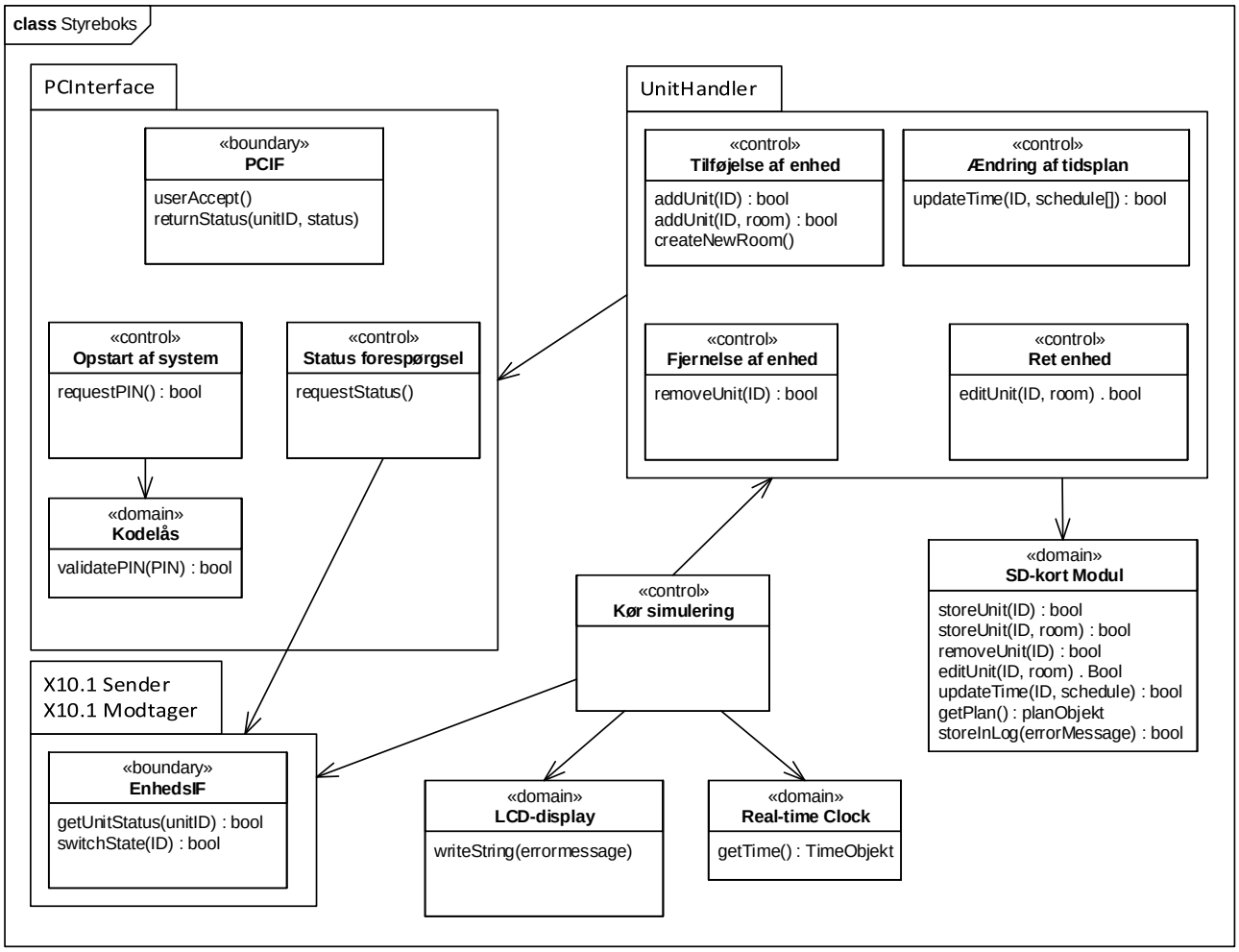
Der udarbejdes først et tomt klassediagram der kan findes i projektdokumentationen[15], hvor enkelte af klasserne samles i UML package grundet den store sammenhæng i de handlinger der skal udføres.

Herefter laves der sekvensdiagrammer for use case 1 til 7, der udarbejdes ikke sekvensdiagrammer for use case 8 og 9, da use case 8 og 9 ved nærmere inspektion har præcis samme scenarie på styreboksen som use case 6. Dette er grundet måden kommunikationen med PCen foregår på. Den i protokollen specificerede kommando vil være ens i alle 3 use cases, og derfor vil styreboksen ikke vide hvilken af de 3 use cases der er tale om når den udfører handlingen. De enkelte sekvensdiagrammer forefindes i projektdokumentationen[16].

Ud fra sekvensdiagrammerne laves der en metodeidentifikation og metoderne indføres på klassediagrammet. Det resulterende klassediagram for applikationsmodellen kan ses på Figur 11.

For en mere uddybet beskrivelse af styreboksens softwarearkitektur henvises der til projektdokumentationen[17].

[14] Side 57, afsnit 4.5.9
[15] Side 58, afsnit 4.6
[16] Side 58, afsnit 4.6
[17] Side 58, afsnit 4.6



Figur 11 - Klassediagram for styreboksens applikationsmodel

5.5 Design, Implementering og Test

Herunder beskrives design, implementering og test af projektets elementer. Sektionen er delt op i beskrivelser udarbejdet af medlemmer af de forskellige grupper, og er inddelt efter Hardware og Software

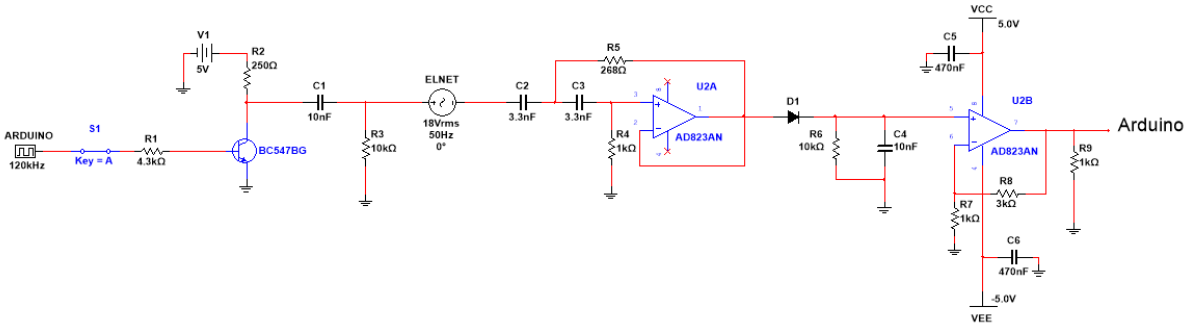
5.5.1 Kodelås (DP)

Kodelåsen er programmeret i faget ”Digital System Design”. Kodelåsen har to hardcoded koder, som kun skal være kendt af brugeren af systemet. Begge koder skal indtastes korrekt på kodelåsen, som først verificerer kode 1 og derefter kode 2. Når begge koder er indtastet korrekt går kodelåsen i tilstanden ”unlocked”. Man har i alt tre forsøg til at taste begge koder rigtigt. Bliver koderne tastet forkert tre gange går kodelåsen i tilstanden ”permanently locked”. Kodelåsen viser vha. LED’er hvor mange forkerte forsøg, der er brugt på indtastning af de to koder. Så længe kodelåsen er i tilstanden ”locked” lyser en grøn LED, og det er ikke muligt at ændre på indstillingerne på styreboksen. Kodelåsen viser hvilken tilstand, statemachinen befinder sig i vha. LED’er.

Koden er skrevet i VHDL og er lavet som en Mealy-Moore state machine.

5.5.2 Hardwaredesign, test og implementering (ME, MB)

På Figur 12 vises det samlede sender- og modtagerkredsløb koblet på elnettet. Diagrammet læses fra venstre mod højre, hvor en Arduino MEGA2560 er koblet på senderkredsløbet. På diagrammet ses at elnettet er bindeled mellem sender- og modtagerkredsløb. Desuden er en Arduino i den anden ende klar til at modtage signalet.



Figur 12 Diagram over sender-, modtager kredsløbet, koblet på elnettet

5.5.2.1 Senderkredsløb

Senderkredsløbets funktion består i at transmittere et 120 kHz firkantsignal ud på elnettet. Det centrale i kredsløbet er transistoren der fungerer som kontakt. Når der løber en mindre strøm ind på basebenet af transistoren styres en større strøm fra collectorbenet ned gennem emitterbenet. Der sendes fra Arduinoen et 120 kHz firkantsignal ind på basebenet af transistoren som får denne til at tænde og slukke 120.000 gange i sekundet. Derved bliver der skabt et 120 kHz firkantsignal på udgangen af senderkredsløbet.

Design af kredsløbet er sket på baggrund af beregninger og simuleringer[18]. Det er sket i en iterativ proces hvor der er blevet skiftet mellem design og implementering. I senderkredsløbet blev et aktivt 2. ordens højpas filter udskiftet med et passivt 1. ordens højpas filter for at gøre kredsløbet mere simpelt da det kun skulle beskytte mod 50 Hz sinussignal fra elnettet.

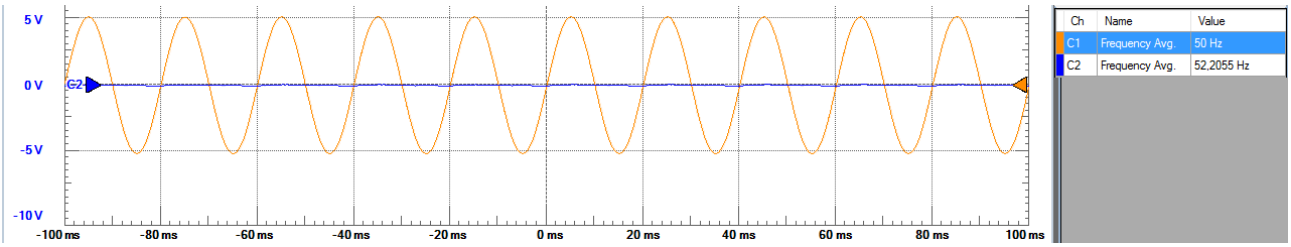
5.5.2.2 Modtagerkredsløb

Modtagerkredsløbets funktion består i at modtage og afkode et 120 kHz firkantsignal. Et 2. ordens højpas filter modtager signalet fra elnettet. Filteret skal sortere et 50 Hz sinussignal fra elnettet væk, samtidig med at lade 120 kHz firkantsignal passere. Når signalet har passeret filteret omdanner en envelope detector[19] det analoge signal til digitalt. En signalforstærker forstærker efterfølgende signalet op til et niveau hvor en Arduino mega2560 kan aflæse det.

[18] Projektdokumentation side 127, afsnit 5.1
[19] Opbygget ud fra bilag "The Envelope Detector"

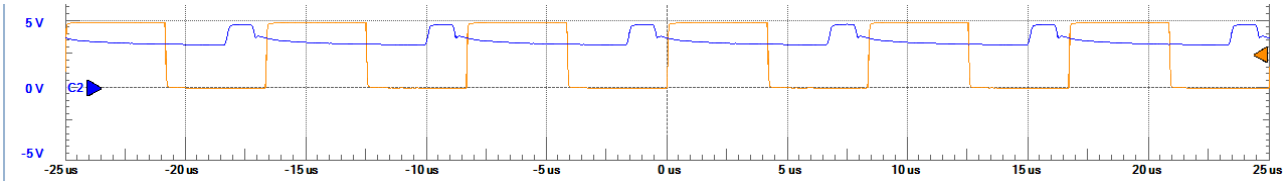
Årsagen til at et aktivt 2. ordens højpas filter er valgt i modtagerkredsløbet, er fordi det dæmper med 40dB/dekade. Dette medfører at et 50 Hz sinussignal samt eventuel støj dæmpes mere effektivt end et passivt 1. ordens højpas filter. Modtagerkredsløbet er mere sårbart overfor støj da der skal afkodes signal. Design og implementering er foregået gennem en iterativ proces med beregninger[20], simuleringer og realiseringer.

Første udkast til modtagerkredsløbet blev designet med en envelope detector fra applikationsnoten[21], men blev senere udskiftet da denne ikke virkede efter hensigten. På Figur 13 ses realiseringen af modtagerkredsløbet med et 50 Hz indgangssignal og digitalt lavt udgangssignal. Dette fastslår, at der ikke bliver transmitteret et 120 kHz firkantsignal på elnettet, og at et 50 Hz sinussignal fra elnettet ikke påvirker udgangssignalet.



Figur 13 Test af modtagerkredsløb. Gul er inputsignal, blå er udgangssignal

Tests for sender- og modtagerkredsløbene er gjort med et input på senderkredsløbet med et 120 kHz firkantsignal fra en funktionsgenerator. Det er blevet testet med et 18V AC 50 Hz sinussignal for at simulere et 230V AC elnet. På Figur 14 ses realisering af sender- og modtagerkredsløb koblet gennem et 18V AC elnet. Resultatet er et højt udgangssignal når der bliver transmitteret et 120 kHz firkant signal ud på elnettet.



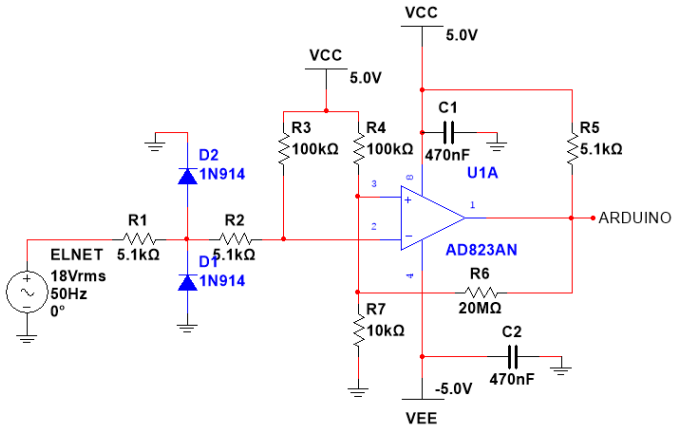
Figur 14 Gul er 120 kHz firkant indgangssignal på sender kredsløbet, og blå er udgangssignalet på modtagerkredsløbet

5.5.2.3 Zero Cross Detector

Diagrammet over zero cross[22] kredsløbet vises på Figur 15, og læses ligeledes fra venstre mod højre. Funktionaliteten af zero cross kredsløbet er at detektere nulpunktsgennemgange på 50 Hz sinuskurven. For hver rising edge på sinuskurven fås et digitalt falling edge og ved falling edge på sinuskuven fås et digitalt rising edge. Dette betyder at der kommer et analog signal ind i kredsløbet og et digitalt signal ud til en Arduino Mega2560.

[20] Projektdokumentation side 13, afsnit 5.2
[21] Se bilag: "AN236_ApplicationNote"
[22] Opbygget ud fra datablad LM339, se bilag

Der har ikke været en egentlig designproces da zero cross kredsløbet er hentet fra et datablad[23] og der er derfor ingen beregninger på det. Med hjælp fra projektvejleder Henning Hargaard blev der taget beslutning om at indsætte en ekstra diode med spærreretning mod GND. Operationsforstærkeren er blevet udskiftet med den samme, der er blevet brugt i andre kredsløb.



Figur 15 – Zero cross kredsløb

Tests er blevet udført med 50 Hz sinussignal.

5.5.2.4 Voltage Inverter

Funktionaliteten for Voltage inverteren er at forsyne AD823[24] med -5V. Dette sker ved at kredsløbet inverterer 5V fra forsyningsspændingen. Kredsløbet er taget med værdier fra databladet for ICL7660[25].

[23] Bilag: "lM339_operationsforstærker"
[24] Se vedhæftet bilag - datablad for AD823
[25] Bilag: "ICL7660_MAX1044_voltage_converter"

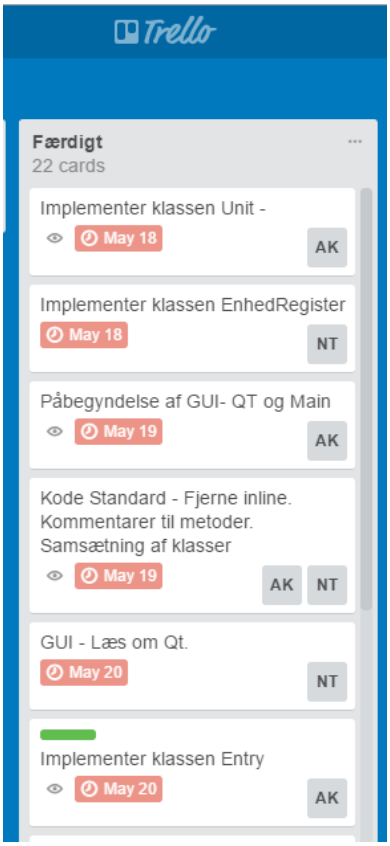
5.5.3 Softwaredesign, test og implementering – PC Software (AK, NT)

Design og implementering af PC softwaren blev foretaget ud fra systembeskrivelse, kravspecifikation og systemarkitektur. Formålet for softwaren på PC'en blev i systembeskrivelsen, beskrevet som brugerens interaktion med "Home Automation". Via brugergrænsefladen kunne brugeren konfigurere hele systemet, og bestemme hvornår systemet skulle være aktivt.

Med kravspecifikation og systemarkitektur, kunne implementering påbegyndes. Fra systemarkitekturen blev brugerens interaktion kortlagt, som kunne ses i sekvensdiagrammerne i projektdokumentation. Da vi under arkitekturfasen havde opstillet et klassediagram, indeholdende metoder fra sekvensdiagrammer, var nogle af metoderne allerede kortlagt inden implementeringen var begyndt.

For at kunne kommunikere med styreboksen, blev der aftalt en protokol[26] som skulle overholdes.

Hver dag under implementeringsfasen, blev der afholdt et møde hvor dagens sprints blev planlagt. De vigtigste opgaver blev priotereret, samt hvor længe de enkelte forventede at tage. Da vi aftalte sprints hver eneste dag, blev opgaverne forventet senest færdig til dagen efter. Alt planlægning blev foretaget via hjemmesiden Trello[27] vist på Figur 16.



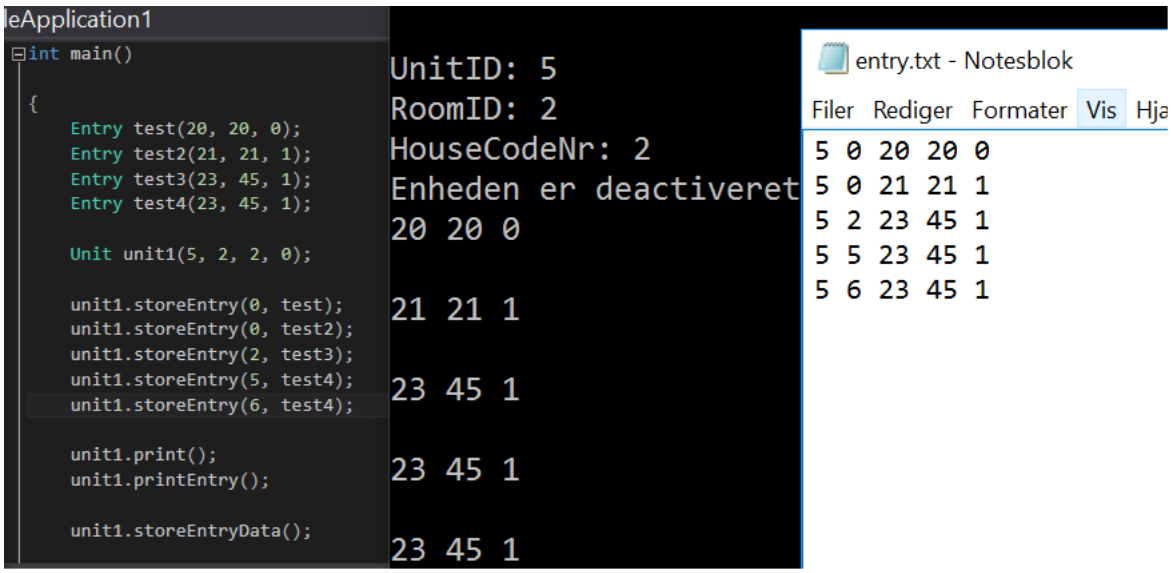
Figur 16- Billede af færdiggjorte opgaver

[26] Projektdokumentation side 37, afsnit 3.4.2
[27] https://trello.com/

Klassediagrammer blev løbende opdateret under implementering af softwaren for PC. Da vi foretog løbende test af klasser og metoder, blev metoder som at printe enheder ud implementeret, selvom vi på forhånd havde en ide om, at disse ikke ville blive benyttet i det endelig program.

De første klasser UnitRegister, Units og Entry blev implementeret i Microsoft Visual Studio. Den grafiske brugerflade er blevet implementeret i QT creator.

En af de løbende test for overstående klasser, ses på Figur 17. Her ses at implementering tester Entry og Unit oprettelse. Da styreboksen blev implementeret af en gruppe, så foretog vi test hvor vi erstattede det modul med en tekst fil. Erstatning med en tekst fil var for at simulere, at der blev skrevet og læst til et eksternt modul. De løbende test blev benyttet for at kommunikere om en opgave var færdig.

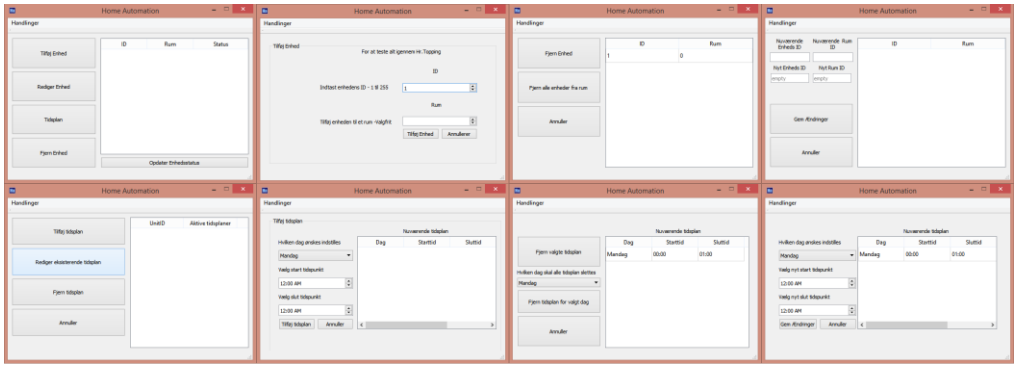


Figur 17- Test af Entry og Unit

5.5.3.1 Brugergrænseflade i QT Creator

Brugergrænsefladen blev oprettet i QT Creator. Hverken AK eller NT havde tidligere erfaring med programmet, hvilket gav en langsom start. Det overordnede formål med brugergrænsefladen var at skabe en let måde for brugeren at konfigurere ”Home Automation”.

Brugergrænsefladen er oprettet ud fra use case 1-6, samt use case 8-9, og tog udgangspunkt i sekvensdiagrammerne udarbejdet i arkitekturfasen. Under selve implementering blev der lavet ændringer, som forhøjede brugervenligheden, hvilket betød opdatering af sekvensdiagrammerne.



Figur 18 – Hele brugergrænsefladen

Et problem der opstod under fremstilling af software, var behovet for at begrænse brugerens mulighed for input. Dette skyldes at systemet maks skal kunne indeholde 256 enheder per rum, og ID 0 er reserveret til styreboksen. Et eksempel på hvordan dette løses vises på Figur 19, hvor et indbygget objekt fra QT, QIntValidator, benyttes til at begrænse tekst input i boksene. Dette begrænser input til tal fra 1 til 255.

```
22      this->findChild<QLineEdit*>("newUnitID_LineEdit")->setValidator(new QIntValidator(1,255));
23      this->findChild<QLineEdit*>("newRoomID_LineEdit")->setValidator(new QIntValidator(1,255));
```

Figur 19 QIntValidator for EditUnit

Efter at PC softwaren var færdig implementeret, blev det testet ud fra accepttesten. Accepttesten er blevet udarbejdet med udgangspunkt i use casene, og indeholder præcise tests som softwaren skulle overholde. Testen for PC softwaren blev udført med kodelås og en erstatning for styreboksen, da denne ikke var færdig implementeret. Størstedelen af PC software testen blev udført med forventede resultater, hvor de fejlede resultater hovedsageligt bestod i at accepttesten ikke tog højde for gennemførsel af foregående tests. Dette resulterede i tests der eksempelvis bad om ændring af enhed med ID "1", som i foregående test var blevet slettet eller givet et andet ID.

Under accepttesten fremstillede NT et stykke testsoftware der kunne emulere styreboksen. Denne software blev brugt til udførelse af Use Case 1, hvori pinkoden fra kodelåsen valideres, og et par dummy enheder sendes til computeren. Koden vist i Figur 20 læser kommando fra PC og sender to fiktive enheder. Disse enheder benyttes til accepttesten. Data er et array med 512 pladser, der simulerer de data blokke der findes på SD-Kort Modulet.

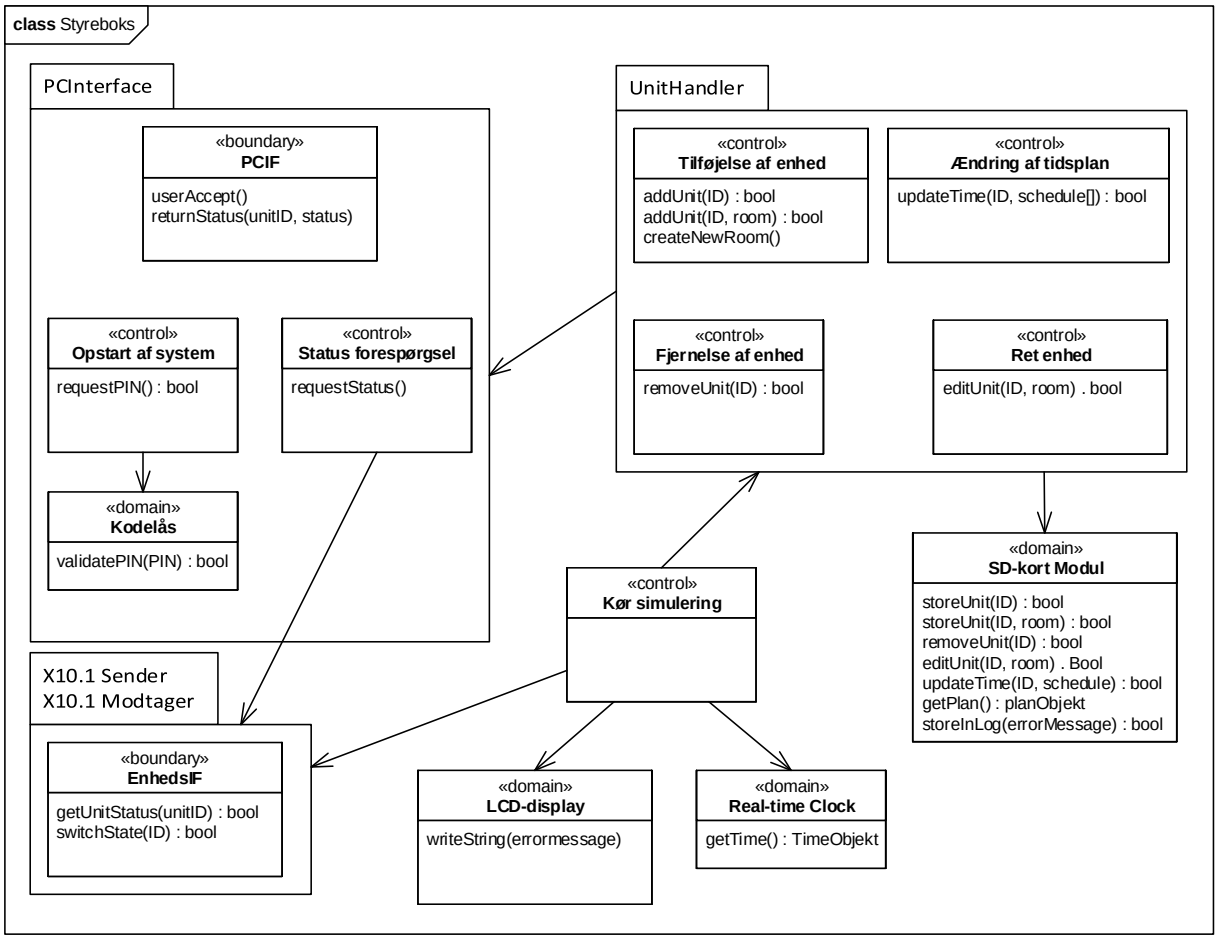
```
if(myArray[2] == 0x06) {
    for(int i = 0; i <= 1; i++) {
        data[0] = i+1;
        data[2] = i+1;
        data[3] = day;
        data[4] = entryID;
        data[5] = 0x0C;
        data[6] = 0x00;
        data[7] = 0x01;
        data[8] = entryID;
        data[9] = 0x0D;
        data[10] = 0x00;
        data[11] = 0x00;
        for(int j = 0; j < 7; j++) {
            for(int x = 0; x < 512; x++) {
                SendChar(data[x]);
            }
            entryID += 1;
            day++;
            data[4] = entryID;
            data[8] = entryID;
            data[3] = day;
        }

        day = 0;
        entryID = 1;
    }
    SendChar(0x0F); SendChar(0x0F);
```

Figur 20 Kodeudsnit til afsendelse af dummy enheder.

5.5.4 Softwaredesign, test og implementering – Styreboks Software (TF)

Der er valgt en objektorienteret tilgang til styreboksens software, hvilket muliggør en bedre struktur af koden samt for at lette vedligeholdelse af softwaren på længere sig.

Ud fra applikationsmodellen udviklet i arkitekturfasen, indledes der design af softwaren til de enkelte klasser, med udgangspunkt i det resulterende klassediagram. Der udarbejdes UML klassediagrammer for de enkelte klasser der udvikles, og i den forbindelse vælges der at implementere de enkelte UML package elementer fra applikationsmodellen som ses på Figur 21.



Figur 21 - Overordnet klassediagram for styreboksen.

5.5.4.1 Implementering og design af SD kort driver klassen (TF)

SD kort driver klassen udvikles ud fra SD simplifed specifikation[28] der er vedlagt som bilag, der benyttes atmega2560 indbyggede SPI modul til at kommunikere med SD-Kortet i SPI mode.
I forbindelse med implementeringen af SD-Kort driveren opstod der en problemstilling med håndteringen af SD-Kortets adressering. Der benyttes i SD-Kortet en 32 bits adressering, og ATmega2560 sender data i 8 bits sekvenser på SPI bussen. Dette er løst ved at typecaste en long til en pointer til en unsigned char hvorved det kan håndteres i koden som et array med fire pladser. Dette array kan nu benyttes til at sende de 4 bytes til SD-Kortet. Implementeringen i koden kan ses på Figur 22.

[28] Bilag: sd-card-spec-simplified

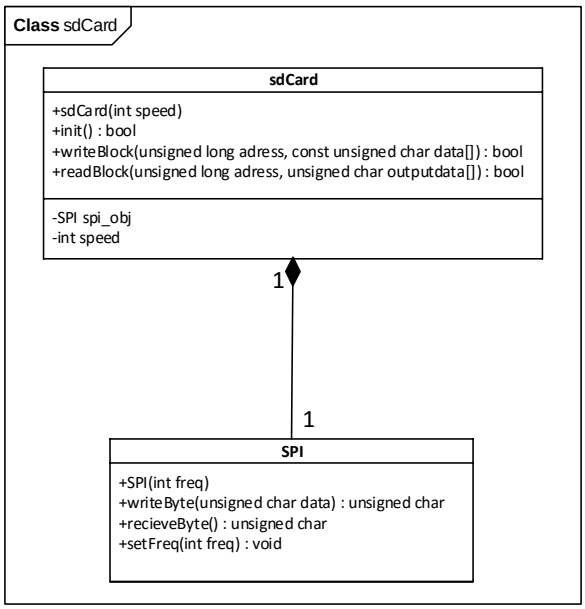
```
157 //============================================================
158 // METHOD : readBlock
159 // DESCR. : reads the block at adress, and fills it into the
160 // supplied outputdata array, data array must be 512 slots.
161 //============================================================
162 bool sdCard::readBlock( unsigned long adress, unsigned char outputdata[] )
163 {
164     unsigned char *argument_byte_pointer = (unsigned char*)&adress;
165     spi_obj.writeByte(0xFF); // clock sync
166     spi_obj.writeByte(0x51);
167     spi_obj.writeByte(argument_byte_pointer[3]);
168     spi_obj.writeByte(argument_byte_pointer[2]);
169     spi_obj.writeByte(argument_byte_pointer[1]);
170     spi_obj.writeByte(argument_byte_pointer[0]);
171     spi_obj.writeByte(0xFF); // dummy CRC;
```

Figur 22 - Uddrag af readBlock implementering for at vise håndtering af SD-Kort adressering

Den mest udfordrende del af softwaren til SD-Kort driveren ligger i at få initieringsprocessen til at forløbe korrekt, da SD-Kort er meget sarte i forhold til timingen på de enkelte kommandoer, hvilket gør at det kan være nødvendigt at gennemgå initieringsforløbet mere end en gang. Derudover er der forskel på initierinsprocessen afhængigt af hvilken type SD-kort der anvendes. I projektet er der anvendt et Kingston SD kort af typen SDHC der er et high capacity kort der kun kan læses og skrives fra i hele 512 bytes blokke. Driveren er for at holde den simpel, lavet så den kun understøtter SD-kort af denne type.

For en detaljeret beskrivelse af initieringen af SD-Kortet se projektdokumentationen[29].

Den resulterende klasse ses på Figur 23.



Figur 23 - Klassediagram for sdCard klassen

[29] Side 108, afsnit 4.9.4

Den færdige klasse testes med det vedlagte testprogram for SD-Kort[30], hvor der via en USB tilkoblet PC sendes testdata via UART indeholdende hvad der skrives på til SD-Kortet, samt hvad der udlæses. De skulle gerne være ens.

5.5.4.2 RTC Driver (TF)

RTC driver klassen er implementeret ved hjælp af en I2C driver, der benytter sig af det indbyggede I2C modul på mega2560, hvor dette anvendes til at kommunikere med en DS3231 RTC. Driveren har mulighed for at indstille tiden på RTC samt udlæse dato, tid og ugedag, hvilket giver mulighed for at styreboksen kan udføre sine simuleringer.

I forbindelse med brugen af DS3231 var der behov for at lave en funktion til konvertering mellem normale integers og Binary Coded Decimal (BCD), dette blev implementeret ved hjælp af modulus samt bit-shifting. Den resulterende kode for implementeringen, der tager en integer og konverterer den til BCD ses på Figur 24.

```
189  //=============================================================
190  // METHOD : intToBCD
191  // DESCR. : converts an 8 bit unsigned value to the binary coded decimal format
192  //=============================================================
193 unsigned char rtc::intToBCD( unsigned char val )
194  {
195      unsigned char tens = val/10 << 4; // divide by 10 bitshift for places.
196      unsigned char ones = val % 10; // get the reminder of a devision by 10
197      return tens + ones; // add the numbers to gain the BCD value
198  }
```

Figur 24 - Kode til at konvertere en integer til BCD.

Det færdige program testes vha. det vedlagte testprogram, tiden indstilles og sendes ud til en PC via UART, og det observeres herved at tiden opdateres og udlæses som forventet.

[30] Bilag: \StyreboksSoftware\sdCardDriver

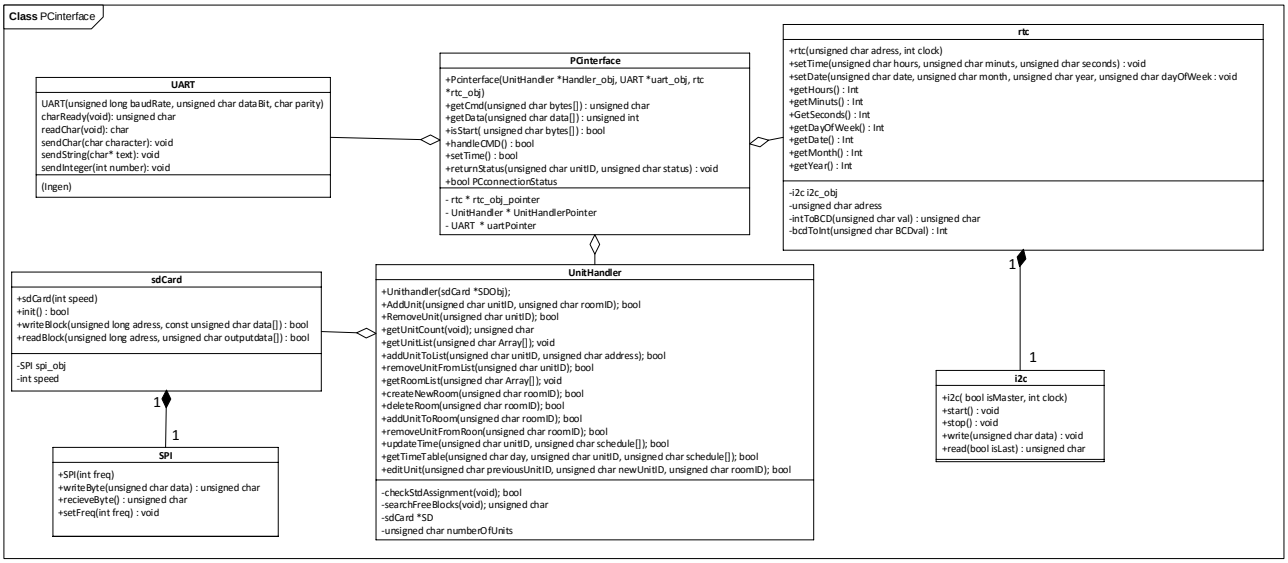
5.5.4.3 Softwaredesign, test og implementering – PCInterface (TF)

PCinterface klassen indeholder funktioner der håndterer kommunikationen med PC softwaren. Kommunikationen foregår via UART. På styreboksen implementeres dette via et objekt af typen UART, der implementerer ATmega2560s indbyggede UART.

Kommunikationen implementeres som en interruptbaseret løsning. Ved kommunikation fra PC'en genereres et interrupt der får styreboksen til at gå i konfigurationsmode, hvor simuleringen standses og behandlingen af kommandoer fra PC'en påbegyndes.

Dette er implementeret som en række funktioner i PCinterface klassen. Ved modtagelse af en kommando på UART, udføres den korrekte handling ud fra PC styreboks forbindelse protokollen, der er yderligere beskrevet i projektdokumentationen[31].

På Figur 25 ses UML klassediagrammet for PCInterface. Dette viser hvordan PCInterface klassen fungerer som bindeled mellem de andre software klasser i Styreboksen. Dette gøres ved hjælp af association, for at mindske hukommelsesforbruget, samt give mulighed for at de forskellige objekter af de andre klasser kan anvendes uden for PCInterface klassen. Dette skaber en kode der er lettere at vedligeholde, da den har en lavere kobling end eksempelvis komposition, som er anvendt mellem sdCard klassen og SPI.



Figur 25 - Klasse diagram for PCinterface klassen.

Funktionen handleCMD er den funktion der står for behandlingen af de forskellige kommandoer fra PC softwaren. Dette gøres ved hjælp af en switch case på den modtagne kommando, der herefter udfører den korrekte handling. Denne handling vil ofte indebære brug af pointeren til UnitHandler objektet, hvilket gør det yderst vigtigt at grænsefladen til UnitHandler objektet ikke ændrer sig. Desværre skred tidsplanen og PCinterface klassen er ikke helt færdigimplementeret endnu, men den grundlæggende skabelon for funktionaliteten af klassen er på plads.

[31] Projektdokumentation side 37 afsnit 3.4.2

Ønskes der en mere detaljeret gennemgang af UART driveren eller PCinterface klassen, kan denne findes i projektdokumentationen[32].

Klassen testes med systemets PC software for at sikre at kommunikationen mellem Styreboks og PC fungerer korrekt. Denne test fejlede.

5.5.4.4 Softwaredesign, test og implementering – Unithandler (SN)

Denne klasse er designet til at håndtere enheder og rum, mens der samtidigt bliver holdt en struktur på SD-kortet som gør det nemt at hente ønskede oplysninger om en enhed.
For at opnå dette var det nødvendigt at fastsætte nogle retningslinjer for allokeringen af blokke på SD-kortet, inden funktionerne blev lavet.

Der blev fastlagt følgende retningslinjer for allokeringen af datablokke på SD-kortet:

1. De første 2 blokke på SD-kortet (blok 0 og 1) bliver brugt til henholdsvis enhedsliste, og rumliste.
    a. På enhedslisten gemmes på første blok et enheds ID, med en efterfølgende counter-værdi. (bruges til beregning af startblok)
    b. På rumlisten gemmes, på første blok, et rum ID, med en efterfølgende counter for det rum.
2. For hver enhed der er tilføjet systemet allokeres 7 blokke, som hver skal indeholde tidsplanen for enheden på en given dag i ugen
3. Der bliver fastsat en adresse (start blok) for hvor fejlloggen starter. (Kommer i senere udvidelser)

Vi benytter således en counter, til at holde styr på hvor mange enheder der på et givent tidspunkt er oprettet i systemet, og beregner så på hvilken placering, den første blok skal ligge for den enhed vi er i gang med at tilføje.
Oplysninger om enhedens ID, antal enheder oprettet, rum og dag gemmes på de første 4 bytes af hver blok, der oprettes for alle enheder. Det vælges derfor at lave counteren som en unsigned char, i stedet for at lave den som en integer da SD-kortets struktur gør det nemmere at arbejde med en enkelt byte.

Der var flere ting som viste sig særligt udfordrende i udviklingen af unitHandler klassen. Én ting, netop pga. fremgangsmåden med at beregne en startblok ud fra en counter-værdi, var at håndterer sletningen af enheder, og følgevirkninger heraf.
Hvis en enhed slettes, og en ny oprettes, vil to enheder altså få den samme counter værdi.
For at komme uden om dette problem, blev det besluttet at når en enhed slettes skulle blokkene som repræsenterede enheden overskrives med 0x00 på alle bytes. Der skulle så videre laves en funktion som kunne tjekke om standard tildelingen var ledig, og en anden som kunne søge efter de tomme blokke på SD-kortet (se Figur 26).

Koden der tjekker standard tildelingen kunne let implementeres pga. den valgte struktur på enhedslisten. Mens koden til at søge efter ledige pladser på SD-kortet kom til at give lidt flere problemer, selv om det endte med at være en simpel implementering (se Figur 26).

[32] Projektdokumentation side 103 afsnit 4.9.2

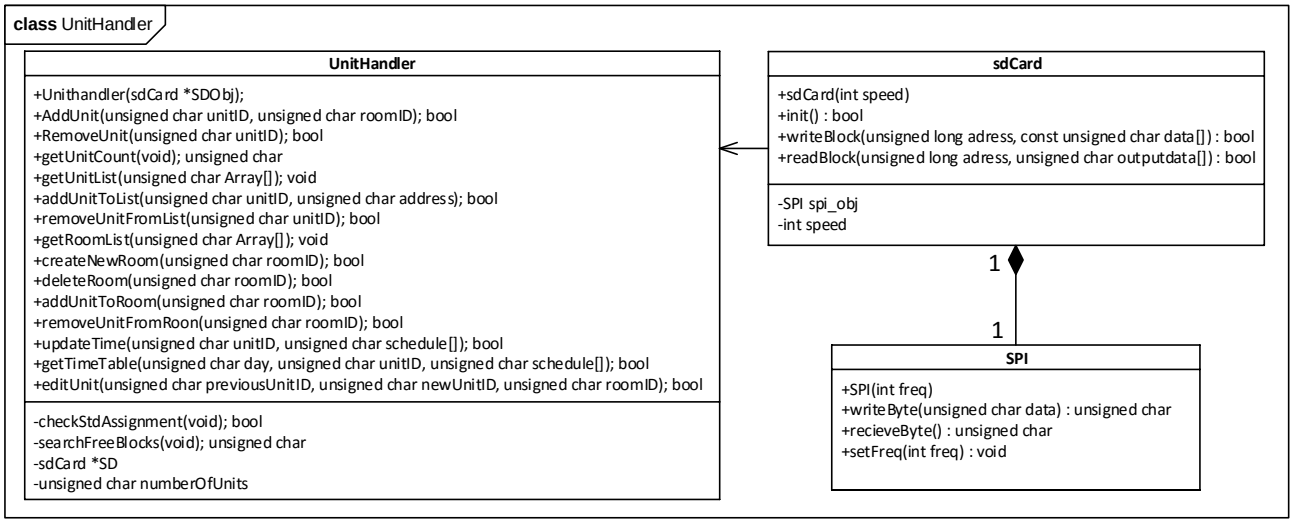
```
unsigned char unitList[512];
//Get list of known units, and their starting point (representation) number.
getUnitList(unitList);

for (unsigned char x = 1; x <= numberOfUnits; x++)
{
    for (int i = 1; i <= 511; i++)
    {
        if (i%2 != 0)
        {
            if (unitList[i] == x)
            {
                i = 512;
            }
            else if (i == 511)
            {
                return x;
            }
```

Figur 26 - Kode der søger efter tommeblokke på SD-kortet.

Det blev således muligt at håndtere følgevirkningerne af slettede enheder, ved at tjekke standard tildelingen indledningsvis, og hvis denne returnerer false, søge efter de ledige pladser og bruge denne værdi på den tilegnede byte, som det er vist på

En anden overvejelse som opstod som følgevirkning af slettede enheder var håndteringen af enhedslisten. Når en enhed blev slettet midt i listen, var det enten nødvendigt at kunne søge efter den ledige plads, eller rykke resten af listen for at udfylde pladsen. Det blev her besluttet at komprimere listen, da det ellers ville være nødvendigt at gennemsøge enhedslisten ved hvert kald af addUnit-funktionen for at vide om der var tomme pladser i listen, og da vi kun kan skrive en hel blok til SD-kortet af gangen, vil der ikke være noget at vinde ved kun at ændre to bytes i blokken.

Det endelige klasse diagram for unitHandler-klassen ses på **Error! Reference source not found.**.



Figur 27 Klassediagram for UnitHandler

Alle funktioner i klassen testes med et til formålet skrevet test-program. Test programmet gør desuden brug af UART-driveren, til at outputte testdata på PC via. Tera-term[33].

Test programmet[34] er lavet ud fra bottom-up princippet, og gør brug af en test opstilling med ATmega2560 tilkoblet SD-modul, og forbundet til PC via. den indbyggede usb forbindelse.

5.5.5 Softwaredesign, test og implementering - X10.1 Sender/Modtager (CBJ)

5.5.5.1 X10.1 Sender

Sender softwaren er kodet i C++ og er lavet som en klasse. Den er designet til at kunne virke med en ATmega2560. I vores tilfælde har vi brugt en Arduino mega2560. X10_sender har til opgave at kompilere en datapakke. Dette gør den ud fra hvem modtageren er, hvad type besked det er og hvad data der skal sendes. Herefter sendes pakken som et X10-indkodet signal over lysnettet.

Ved et X10-indkodetsignal menes der at der sendes et logisk 1 på en rising edge fra zero-cross, og der sendes et logisk 0 på en falling edge fra zero-cross. Så når det næste bit, der skal sendes, er et logisk 1, bliver der sendt et signal, når der er en rising edge fra zero-crossen.

Signalet der bliver sendt er et 1 ms burst af et 120 kHz firkant signal. Dette bliver realiseret via timer 1, der er indstillet til Mode 8: ”PWM phase and frequency correct”, som så toggler OC1A. ICR1 er loadet med en værdi der sammen med prescaleren giver en frekvens på de 120 kHz. For at det kun bliver sendt i 1 ms bruges timer 0 i normal mode sat op med en interrupt, så den i interrupt rutinen stopper PWM signalet.

Senderen er implementeret således, at der via et enkelt metodekald kan sendes en ønsket besked til modtageren. Der kompileres en data pakke og programmet bliver sat til send mode. Når den er i denne mode begynder senderen at sende hvert bit ved det relevante interrupt fra zero-cross. Efter at en pakke er blevet sendt går senderen ud af send mode.

[33] https://ttssh2.osdn.jp/index.html.en
[34] Bilag: \StyreboksSoftware\UnitHandler

Der blev først testet for, om der kunne produceres et PWM signal med den korrekte frekvens. Derefter testes for om PWM signalet kunne blive sendt i et 1 ms burst. Alt dette blev målt og verificeret med et Analog Discovery oscilloskop.

Den næste test var at kontrollere, at senderen kan kompilere en pakke og derefter sende den. En af knapperne på MSYS arduino shielded blev sat op til at sende en bestemt pakke. I første test blev de to interrupt fra zero-cross simuleret via to knapper. Der blev observeret på oscilloskopet at der blev sendt et burst ved det korrekte interrupt. Der blev derefter sammenlignet med de binære værdier som pakken bestod af i koden, og de binære værdier der kunne observeres der blev sendt. Der var nogle problemer med at sende pakken i starten, som hovedsagligt var på grund af nogle mindre fejl i en del af funktionerne. Da de var blevet rettet kunne der sendes en hel pakke.

Der blev herefter testet at de forskellige typer af beskeder også blev sendt korrekt.

Sidste test var at køre senderen i fuld hastighed, og at det virkede med vores hardware. Der kunne aflæsses på vores oscilloskop at pakken blev afsendt og at der kom et signal burst på de rigtige steder på de 50 Hz sinus. Der var også et modtager kredsløb der var sat op til at vise hvilken type pakke den havde modtaget. Denne test kørte fejlfrit.

5.5.5.2 X10.1 Modtager

Modtager softwaren er kodet i C++ og er lavet som en klasse. Det er designet til at kunne virke med en Arduino mega2560. X10_Modtager har til opgave at modtage X10 kommunikation og løbende validere om pakken er adresseret til modtagerenheden, hvad data pakken indeholder og om hele pakken er sendt intakt.

Modtageren er sat op med de samme to interrupts til zero-crossen, som i sender koden. Når der kommer et interrupt på rising eller falling, tjekkes der i 1 ms om der er et højt signal fra envelopen. Hvis der ses et højt signal tager den det som et logisk 1, hvis det skete på rising edge, eller som et logisk 0 hvis det sker på falling edge. Kommer der ikke noget på hverken rising eller falling, resettes modtageren.

Koden er implementeret så at modtageren altid vil kunne modtage et signal, medmindre microcontrolleren er i gang med at sende. Når der først bliver modtaget data, blive der tjekket om den modtaget data overholder protokol kravene. I starten af en pakke bliver der tjekket om den har de rigtige start bit. Efter dette valideres at adressen passer med microcontrollerens egen. Hvis en af disse to fejler, stopper modtageren i at gemme pakken. Den venter så på de 6 slut bit, eller til der ikke er blevet sendt data i et stykke tid.

Sidste del af pakken indeholder hvilken type besked der modtages, og dens data. Her efter modtages et paritets bit, efterfulgt af de seks slut bit. Der tjekkes så om der er paritets fejl. Der sættes så et flag om at der er modtaget en pakke.

Det er designet så der via et funktions kald kan tjekkes om der er data klar til at blive læst. Når der er data klar kan man derefter via et andet funktionskald hente hele pakkens data, og samtidig resettes modtageren så der kan modtages data igen.

Der blev i første omgang testet om en bestemt type pakke kunne modtages. Interrupt blev simuleret med trykknapper, det samme med signalet fra envelopen. Der var mange forskellige testere til forskellige iterationer, på grund af de mange fejl der var i starten. Der blev brugt LED'erne fra MSYS Arduino shieldet til at debugge programmet. I en af de første iterationer blev LED'erne brugt til at se om modtageren kom ind i de rigtige dele af funktionen på de rigtige tidspunkter. Det viste sig at problemet skyldtes fejl i trackingen af hvor meget af pakken der var blevet modtaget. Det tog et stykke tid at fikse problemet da det skyldtes et design problem, og det derfor var alle steder hvor en form for tracking af positionen blev brugt. Da fejlen var udbedret fungerede koden som den skulle.

Der blev så kørt en test hvor LED'erne viste hvilken pakke der blev modtaget. Denne test var en succes, og viste præcis den pakke der blev sendt.

Den sidste test var at kunne modtage data fra senderen gennem hardwaren, og med 50 Hz AC. Dette virkede ikke i første omgang. Efter nogle rettelser i interrupt rutinerene for zero-crossen, kunne der nu også modtages data i fuld hastighed fra senderen

5.6 Resultater og Diskussion

5.6.1 Resultater for Kodelås (DP)

Kodelåsen er testet i faget "Digitalt System Design" i forbindelse med en rapport. Og er ikke testet yderligere i dette projekt.

Følgende resultater er opnået med kodelåsen:

- Kodelåsen er i stand til at gå i "unlocked" tilstand, når de to koder indtastes korrekt.
- Kodelåsen er i stand til at gå i "permanently locked", hvis koderne tastes forkert i alt tre gange.
- Kodelåsen er i stand til at sende et lavt signal ud, når kodelåsen er i tilstanden "unlocked" og sende et højt signal ud, når kodelåsener i tilstanden "locked".

5.6.2 Resultater for PC (NT)

Følgende resultater er opnået med PC'en:

- En funktionel GUI blev fremstillet.
- GUI indeholder funktionalitet til at udføre Use Case 1-6 samt Use Case 8-9.
- Interface til kommunikation mellem styreboks og PC er i stand til at sende og modtage enheder.

5.6.3 Resultater for X10.1 styreboks (DP)

Følgende resultater er opnået med X10.1 styreboksen:

- Senderdelen er i stand til at sende et 120 kHz signal ud i 1 ms burst.
- Senderdelen er i stand til at skelne i mellem 0- og 1-bits i en kommando.
- Modtagerdelen er i stand til at filtrere uønskede frekvenser fra, så kun data signalet er tilbage.
- Modtagerdelen er i stand til at modtage et 120 kHz burst og lave det om til et digitalt signal i gennem envelope detektoren.
- Modtagerdelen er i stand til at modtage en kommando fra envelope detektoren.
- Zero Cross detektoren er i stand til at detektere zero cross på et 50 Hz lysnet.
- Er i stand til at kommunikere med en PC via en USB-forbindelse.
- Formår at være låst, hvis der modtages et højt signal fra kodelåsen på et af arduinoens ben.

5.6.4 Resultater for X10.1 enhed (DP)

Følgende resultater er opnået med X10.1 enheden:

- Senderdelen er i stand til at sende et 120 kHz signal ud i 1 ms burst.
- Senderdelen er i stand til at skelne i mellem 0- og 1-bits i en kommando.
- Modtagerdelen er i stand til at filtrere uønskede frekvenser fra, så kun data signalet er tilbage.
- Modtagerdelen er i stand til at modtage et 120 kHz burst og lave det om til et digitalt signal i gennem envelope detektoren.
- Modtagerdelen er i stand til at modtage en kommando fra envelope detektoren.
- Zero Cross detektoren er i stand til at detektere zero cross på et 50 Hz lysnettet.

5.7 Opnåede Erfaringer

Herunder findes gruppemedlemmers opnåede erfaringer. Afsnittet afsluttes med en fælles beskrivelse af de opnåede erfaringer.

5.7.1 Stefan (SN)

Jeg har personligt lært en masse om hvad det kræver, og hvad der er vigtigt når 8 forskellige mennesker skal lave et projekt, som det er tilfældet her i semester projekterne.
Jeg kom ind i projektet med den indstilling, at det vigtigste var at alle gruppemedlemmer havde en følelse af at få deres meninger hørt, og derigennem være en aktiv del af udviklingsgruppen.
Men, vi gjorde den fejl, at vi på vores indledende gruppemøde, fik lavet en abstrakt aftale der lød *"Vi vil lave et ambitiøst projekt"* hvilket i praksis gjorde det umuligt at tage hensyn til alle medlemmers ønsker, da flere var i direkte konflikt med hinanden. Det endte så med at koste en masse spildtid på at diskutere diverse beslutninger.

Jeg har altså med andre ord lært, at hvis beslutninger i en gruppe skal træffes på demokratisk vis, så er det vigtigt at der fra starten er fastlagt en rød tråd som skal følges. Et andet alternativ, som måske endda er at foretrække, vil være at have en leder som 100% fastlægger retningen, uden nødvendigvis at tage hensyn til medlemmernes ønsker.

5.7.2 Dennis (DP)

Jeg har i dette projekt gjort mig en masse erfaringer, både hvad angår kommunikation, planlægning og samarbejdet generelt i en gruppe. Jeg har desuden lært om forskellige faser i et projekt, og hvor vigtigt det er at have færdiggjort en fase, inden man går videre til den næste. Så man ikke ender med at have en masse løse ender og arbejdet i næste fase bliver også både nemmere og mere overskueligt, hvis forarbejdet er gjort godt. De review-møder, der har været afholdt, de har fungeret godt. Da der bliver fanget en masse fejl og dårlige formuleringer, som man nemt kan have set sig blind på. Det er desuden vigtigt at få foretaget disse ændringer kort tid efter et review-møde, så de ikke bliver skubbet i baggrunden.

Jeg har vekslet lidt frem og tilbage mellem to grupper (HW-design og Styreboks SW). Jeg har primært hjulpet til i designfasen på de to grupper, afhængig af hvor der var brug for hjælp. Her har jeg savnet lidt kontinuitet, og vil i et fremtidigt projekt være mere fast tilknyttet til en delgruppe.

I starten af projektet lavede vi en samarbejdskontrakt, som blev glemt i løbet af 3-4 uger. Hvilket jeg mener er synd, da den kunne være brugt mere. Heriblandt til at holde folk op på, hvis en aftale ikke blev overholdt, man kommer for sent eller udebliver.

5.7.3 Christian (CBJ)

I løbet af dette projekt har jeg erfaret nyttigheden i at lave review af en andens gruppes projekt og ligeså at få sit eget projekt reviewet. Bare det at review en andens gruppes arbejde har givet stor indsigt i hvad man måske selv kunne gøre bedre. Og så selvfølgelig det at blive reviewet, der giver andres syn på hvad man har lavet og er i gang med. Det at få andres øjne på ens projekt flere gange har gjort at man har kunne opdage mangler i ens dokumentation og planer.

Jeg har også erfaret at en gruppe struktur er vigtigt. Så hvordan arbejdet blev fordelt og planlagt. Vi kørte med 3 grupper, der vær især har fået tildelt en arbejdsbyrde som den skulle løse. Jeg syntes at det fungerede godt, og det sørgede for at alle havde nogen at arbejde sammen med og nemt vidste hvad de skulle gøre. Planlægningen har som regel været god, men arbejdsbeskrivelsen har nogle gange været ringe. Der har nemlig til tider været tvivl blandt grupperne hvad den enlige opgave har været. Det har hændt at der har været spildt arbejde på grund af dette. Dette problem blev også meget hurtigt taget op på gruppen, og blev væsentligt forbedret i løbet af projektet.

Jeg har kunne benytte en iterativ arbejdsproces da jeg arbejdede med software til Sender og Modtager til X10. Netop der har der været tydelig brug af denne tilgang, hvor jeg har skulle gå gennem flere iterationer af koden. Jeg har lavet en del af koden og testet den. Hvis koden virkede er jeg gået videre og designet den næste del. Men hvis koden ikke virkede har jeg prøvet at fejl finde koden.

5.7.4 Nikolai (NT)

I forhold til strukturen af projekt og gruppearbejde har jeg gjort mig de erfaringer, at en klar og tydelig opdeling af ansvarsområder og arbejdsopgaver er vigtig for et projekts success. Under projektet har der været forsøgt at lave en sådan inddeling, men det har været plettet af problemer. Forsøg på en demokratisk gruppestruktur, uden en defineret process til konfliktløsning, har betydet at meget tid er spildt på at diskutere mellem gruppens medlemmer, da jeg selv og flere har meget stærke holdninger.

I forhold til selve udførsel af projektets arbejde er det blevet tydeligt hvor vigtigt arkitekturfasen er. Projektes PC software bærer præg af problemer under udvikling af arkitekturen, da der er en unødvendig høj kobling mellem softwarens klasser. Dette skyldes delvist også mangel på erfaring i brug af QT, der ledte til en række midlertidige lappeløsninger. Disse endte med at blive permanente løsninger, da der jævnligt blev opdaget mangler der ikke blev dækket under arkitekturen, og disse blev løst med improvisering.

Mod slutningen af projektet, og under design og implementering har jeg lært at jeg kan stå frem som leder, og få et overblik over arbejdet, hvad der er foretaget, og hvad der laves. Det kræver fra min side af at jeg står mere frem, og engagerer mig i arbejdet, da jeg mener at jeg kan tilføre meget til en gruppe i en leder rolle.

5.7.5 Tonni (TN)

Jeg har igennem projektet gjort mig en mængde erfaringer omkring kommunikation, projektledelse, og hvad der er vigtigt for at projektforløbet kan fungere. Vi udarbejdede fra starten en samarbejdsaftale, der var alt for abstrakt formuleret, og som ikke blev anvendt nok i løbet af projektforløbet. Vi startede med en demokratisk ledelsesstruktur, hvilket resulterede i at for meget tid gik på diskussion af projektets retning, frem for udarbejdelsen af projektet. Her ville det have været bedre med en topstyret ledelsesform. Da deadlines ikke blev overholdt påtog jeg mig for mange af de opgaver der lå ufærdige, hvilket har medført en for stor en arbejdsbyrde i forhold hvad jeg kunne nå. Jeg skal fremadrettet være bedre til at sige fra og lade andre tage de arbejdsopgaver. Derudover har fejlende i projektet vist hvor vigtigt indholdet i ISE kurset er for et succesfuldt projekt, da vi i vores projekt ikke har været gode nok til fra start at få de konkrete specifikationer på plads, hvilket har skabt yderligere forsinkelser i løbet af projektet. Jeg har fungeret som suppleant til Stefan som gruppeleder, og kan konstatere at jeg bedre egner mig som udviklingsingeniør end som leder.

5.7.6 Martin (MB)

Fordi der blev begåede fejl igennem hele projektet, har det medvirket til en stor refleksion over projektets udfald.

Det vigtigste jeg vil tage med mig, er vigtigheden af en fælles forståelse samt en struktureret udviklingsproces. Da dette gav øget kompleksiteten i gruppes arbejdsform, fejlende har dog været der hvor jeg har lært adler mest af.

Jeg har fået et mere realistisk syn på hvordan man går fra ide fase til konceptudvikling. Man går systematisk til værks med konstant evaluering af processen, både for at opnå et controllerede arbejdsmiljø samt god arbejdsfordeling. Desuden er utroligt vigtigt at gruppen kender til ens ingeniør faglige kompetencer for at udnytte dem bedst muligt.

5.7.7 Mikkel (ME)

Den største lektie for mig i dette projekt har været vigtigheden af struktur og arbejdsfordeling i en udviklingsproces. At skulle overskue et projekt fra start til slut kræver god kommunikation mellem gruppemedlemmer, stærk gruppedynamik, fælles forståelse og ikke mindst et behov for struktureret arbejdsfordeling. Kommunikationen har været udmærket, men der har været for mange misforståelser. Gruppedynamikken har til gengæld været helt i top og jeg har været tilfreds med alle gruppemedlemmers indstilling og humør. Der har generelt været svag struktur i projektet og dette ansvar ligger på alles skuldre.

Mine ingeniørfaglige kompetencer er blevet skærpet på hardwaredelen som jeg primært har arbejdet på. Mine styrker vil derfor i fremtiden ligge på hardware og på at kunne overskue og strukturere et projekt.

Mine svagheder har været at jeg ikke har været nok engageret i projektet. Jeg har fokuseret på løbende eksamener og undervisning gennem semesteret. Projektet er desværre blevet nedprioriteret for ofte. Det har resulteret i manglende aktivitet fra min side.

5.7.8 Anders (AK)

Jeg har i dette projekt gjort mig erfaringer med gruppearbejde og dets struktur. Folk arbejder forskelligt selvom målet er det samme. Grundet dette har projektet vist at struktur og kommunikation er vigtigt for gennemførelse. Projektet har været ramt af manglen på fælles planlægning og uddelegering, hvilket har medført at dele af projektet er trukket i forskellige retninger, og enkle faser har taget længere tid end nødvendigt. Gruppen blev delt op i forskellige grupper, hvilket jeg godt kunne lide da det medførte at jeg fik fokus på mit arbejdsområde med software, men grundet manglende kommunikation mellem grupperne, så påvirkede det gennemførelsen af projektet i negativ retning.

Jeg har nydt at udvikle projektet i forskellige faser, da det giver en struktureret udvikling af projektet, hvor erfaringerne fra de enkle faser går videre til næste. Her har reviews været positive, da andres erfaringer på samme projekt, kan give en ny og konstruktiv udvikling på projektet. Størst erfaring med projektet, har været grundet kursus ISE, hvor jeg har lært at udvikle og designe projektet med SysML og UML. Jeg har gjort erfaring, at selvom lysten for blot at begynde at implementer, så har udviklingen af beskrivende diagrammer gjort implementering lettere.

Jeg har gjort mig erfaring, at aftaler som ikke overholdes, kan påvirke flere faser og gøre at arbejdsbyrden i gruppen bliver skæv. Her har manglende konsekvens været en mangel, selvom der var udfyldt en samarbejdskontrakt. Derudover har jeg gjort mig erfaring, at prioritering og uforudsete ting, kan påvirke hele projektet. I gruppen, har jeg nydt at have en mere supplerende rolle, hvor jeg har kunne lære af Nikolai som har større erfaring. I fremtiden vil jeg forsøge at holde et større overblik, og arbejde mod at gruppen arbejder med korte deadlines, da vores erfaring fra softwaren gruppen var positiv.

5.7.9 Fælles (Alle)

På trods af at gruppen ikke har formået at levere et færdigt produkt, har gruppen gjort sig mange erfaringer til fremtidig brug. Disse erfaringer omfatter bl.a. samarbejdsaftalen som har efterladt for stor risiko for fortolkning. Dette giver risiko for misforståelser, hvilket har resulteret i at der er blevet brugt for meget tid på at diskutere betydningen af indholdet. Dette kunne have været undgået ved at udfærdige en samarbejdsaftale, der ikke efterlader mulighed for fortolkning.

Dette sammenholdt med en for løs mødestruktur i løbet af projektet har været årsag til at den planlagte tidsplan på ingen måde har været overholdt. Fremadrettet vil det være en god ide at starte ud med væsentligt flere møder end et ugentligt møde, og så i løbet af forløbet justere antallet af møder i forhold til den på tidspunktet eksisterende arbejdsbyrde.

Ud over problemerne med samarbejdsaftalen og de få møder har vores ledelsesstruktur ikke fungeret. Vi startede ud med en projektleder, hvilket resulterede i at der opstod en manglende struktur i forløbet, da det var umuligt for en person at bevare overblikket over hele projektet. Derfor reviderede vi i løbet af projektet ledelsesformen med opdeling i tre grupper med separat ledelse, der skulle samarbejde mellem hinanden. Denne opdeling havde både positive og negative resultater, da kommunikationen grupperne imellem ikke fungerede optimalt. På den positive side blev der arbejdet mere effektivt efter opdelingen, men på den anden side opstod der igen misforståelser i forhold til arbejdet grupperne imellem. Her skal der en leder med et samlet overblik ind over til at organisere fordelingen af arbejdet mellem grupperne og løbende omorganisere arbejdsfordelingen, når behovet opstår. Dette vil give grupperne ro til at fokusere på

de enkelte arbejdsopgaver. Der vil så ved hjælp af 2-3 ugentlige møder kunne opsummeres hvordan arbejdet er skredet frem, og derved bevares en sammenhæng mellem gruppernes arbejde.

Til organisering af arbejdet er der brugt Slack som platform med integration af Google Calendar samt Github. Dette har virket rigtig godt for gruppen og vil højst sandsynligt blive brugt igen i et fremtidigt arbejde.

I forbindelse med brugen af Github til versionskontrol og håndtering af dokumenter er det meget vigtigt at der bliver fastlagt en ensartet filstruktur, og at commit beskeder beskriver hvad der er udført af arbejde. En mulig forbedring i brugen af github kan være at benytte bugtrackeren til at håndtere milestones og arbejdsopgaver, hvilket kan give et hurtigt overblik over hvem der arbejder med hvad.

Vi har været gode til at holde en god gruppe dynamik, hvilket har givet en god uformel tone gruppemedlemmerne imellem.

5.8 Fremtidigt Arbejde (Alle)

Hardware der mangler at blive færdigt før prototypen er funktionel:

1. Relæ til styring af output til lampe
2. Indikator LED modul
3. LCD display
4. Genstartsfunktion

Software der mangler for at prototypen er funktionel:

1. PCinterface klassen
2. Simulerings klassen
3. X10.1 wrapper klasse.
4. Fejlhåndtering
5. Herunder klassificering af fejltyper
6. Software til enheden.

Mulige videre udvidelser:
Generelt:

1. Implementering af funktionalitet til at udføre handling på alle enheder i et rum.
2. Understøttelse af sensorer
3. Mulighed for at hente informationer om rum og tilhørende enheder fra unitHandler klassen
4. Mulighed for dæmpning af lys
5. Support for 230 V AC
6. Strømforsyning
7.

Software PC

1. Refactor koden
2. Bedre udnyttelse af Room ID
3. Forbedre usability

6 Konklusion (Alle)

Vi kan konkludere at vores “Home automation” system ikke vil egne sig som tyveriforebyggelse. Systemet skulle være i stand til at simulere aktivitet i hjemmet ved at gøre huset levende for at afskrække eventuelle indbrudstyve.

Dette skulle gøres ved automatisk styring af belysningen i hjemmet ved kommunikation over hjemmets eksisterende lysnet. Kommunikationen over lysnettet er i nuværende iteration kun delvist funktionelt, da det er muligt at sende og modtage data, men belysningen kan ikke controlleres ud fra de overførte data.

Den grundlæggende funktionalitet i de enkelte moduler der udgør enheden, styreboksen samt PC softwaren er delvist funktionelle, men grænsefladerne til kommunikation mellem de enkelte moduler er ikke færdige.

Dette skyldes at tidsplanen ikke er blevet overholdt, grundet dårlig planlægning i gruppen.
På trods af problemet med planlægningen er vi ikke langt fra målet om at kunne levere en brugervenlig løsning til tyveriforebyggelse.