

# FORSIDE

# FORORD

<b>1</b>	<b>Kravspecifikation (TF, SN, MB, DP, ME, AK, NT) .....</b>	<b>6</b>
1.1	Aktører.....	6
1.2	Terminologiliste .....	7
1.3	Use Cases.....	8
1.4	Yderlige tekniske krav.....	17
1.5	Grafisk Brugerflade (NT, AK).....	18
<b>2</b>	<b>Systemarkitektur .....</b>	<b>19</b>
2.1	Overordnet Arkitektur (TF, SN, MB, DP, ME, CBJ) .....	19
2.2	Styreboks (TF, SN, MB, CBJ, DP, ME) .....	23
2.3	Enhed (TF, SN, MB, CBJ, DP, ME).....	31
2.4	Protokolbeskrivelse (ME, CBJ, MB).....	35
2.5	Domæneanalyse (Alle).....	39
<b>3</b>	<b>Softwarearkitektur, Design og Implementering.....</b>	<b>40</b>
3.1	Blokbeskrivelse: Domænemodel (NT) .....	40
3.2	Identifikation af control klasser (NT, AK).....	41
3.3	Overordnet Funktionalitet : Sekvensdiagrammer (NT) .....	42
3.4	Identifikation af klasser (Alle).....	46
3.5	Applikationsmodel – PC (NT, AK).....	47
3.6	Applikationsmodel – Styreboks (TF, SN, DP) .....	58
3.7	Applikationsmodel – Enhed (CBJ, MB, DP, SN, ME) .....	66
3.8	Design og Implementation – PC (NT, AK) .....	68
3.9	Design og Implementation – Styreboks (TF, SN, DP).....	101
3.10	Design og Implementering – X10.1 Sender/Modtager (CBJ).....	118
<b>4</b>	<b>Hardware Design og Implementation (ME, MB, TN, DP) .....</b>	<b>127</b>
4.1	Sender kredsløb.....	127
4.2	Modtager kredsløb .....	130
4.3	Zero cross.....	137
4.4	Voltage inverter.....	139
4.5	Samlet sender og modtager .....	141
<b>5</b>	<b>Accepttest (TF, SN, MB, DP, ME, AK, NT).....</b>	<b>143</b>
5.1	Test af Usecases.....	143
5.2	Test af yderligere tekniske krav.....	168

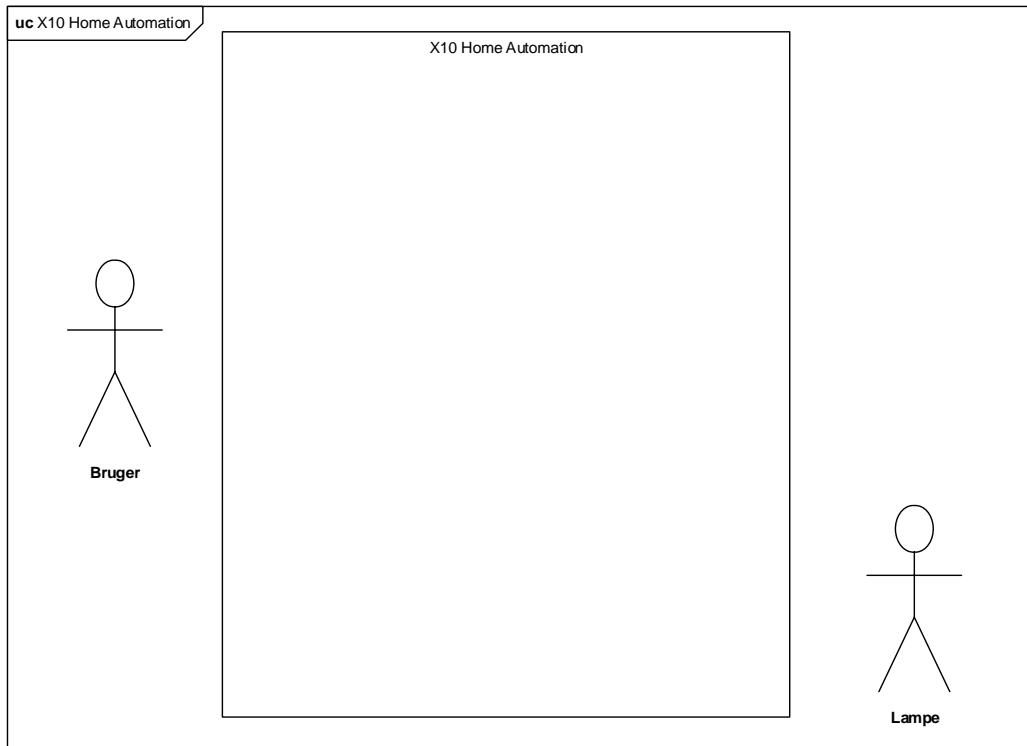


# INDHOLD

# 1 Kravspecifikation (TF, SN, MB, DP, ME, AK, NT)

## 1.1 Aktører

Dette afsnit indeholder en oversigt og beskrivelse af aktørerne i systemet. På Figur 1 ses et aktør diagram der viser hvilken rolle aktørerne har i forbindelse med brugen af systemet.



Figur 1 Aktørdiagram

### 1.1.1 Bruger

#### AKTØRNAVN BRUGER

<b>TYPE</b>	Primær
<b>BESKRIVELSE</b>	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 står for at konfigurer enhederne samt tidsplanen som simuleringen foregår ud fra.

### 1.1.2 Lampe

#### AKTØRNAVN LAMPE

<b>TYPE</b>	Sekundær
<b>BESKRIVELSE</b>	Lampen er forbundet til systemets enhed(er), og bliver styret af systemet.

## 1.2 Terminologiliste

### 1.2.1 PC Software

PC Softwareen betegner hele systemets PC-del. Dette inkluderer den grafiske brugerflade(GUI), såvel som den bagvedliggende funktionalitet.

### 1.2.2 Styreboks

Systemets styreboks indeholder en liste over enheder og handlinger. Styreboksen har til formål at simulere brugerens ønskede konfigurering, ved at sammenligne det nuværende klokkeslet med et klokkeslet tilknyttet en specifik handling.

Styreboksen indeholder derved også kredsløb til at sende og modtage X10.1 kommunikation.

### 1.2.3 Enhed

En enhed defineres i systemet som en boks indeholdende sender og modtagerkredsløb, hvor et givent stykke forbrugerelektronik kan tilsluttes. Enhedens funktionalitet er begrænset til at kunne tænde/slukke for den tilsluttede elektronik, ved brug af et relæ der kan åbne eller lukke for strømmen, på samme måde som en almindelig stikkontakt.

### 1.2.4 X10.1

X10.1 er en navngivning givet til den protokol der bruges til kommunikation mellem styreboks og enheder. Protokollen er baseret på X10, med meget få ændringer. Navngivningen bruges for at synliggøre at det ikke er den nøjagtige X10 protokol der benyttes.

### 1.2.5 Kodelås

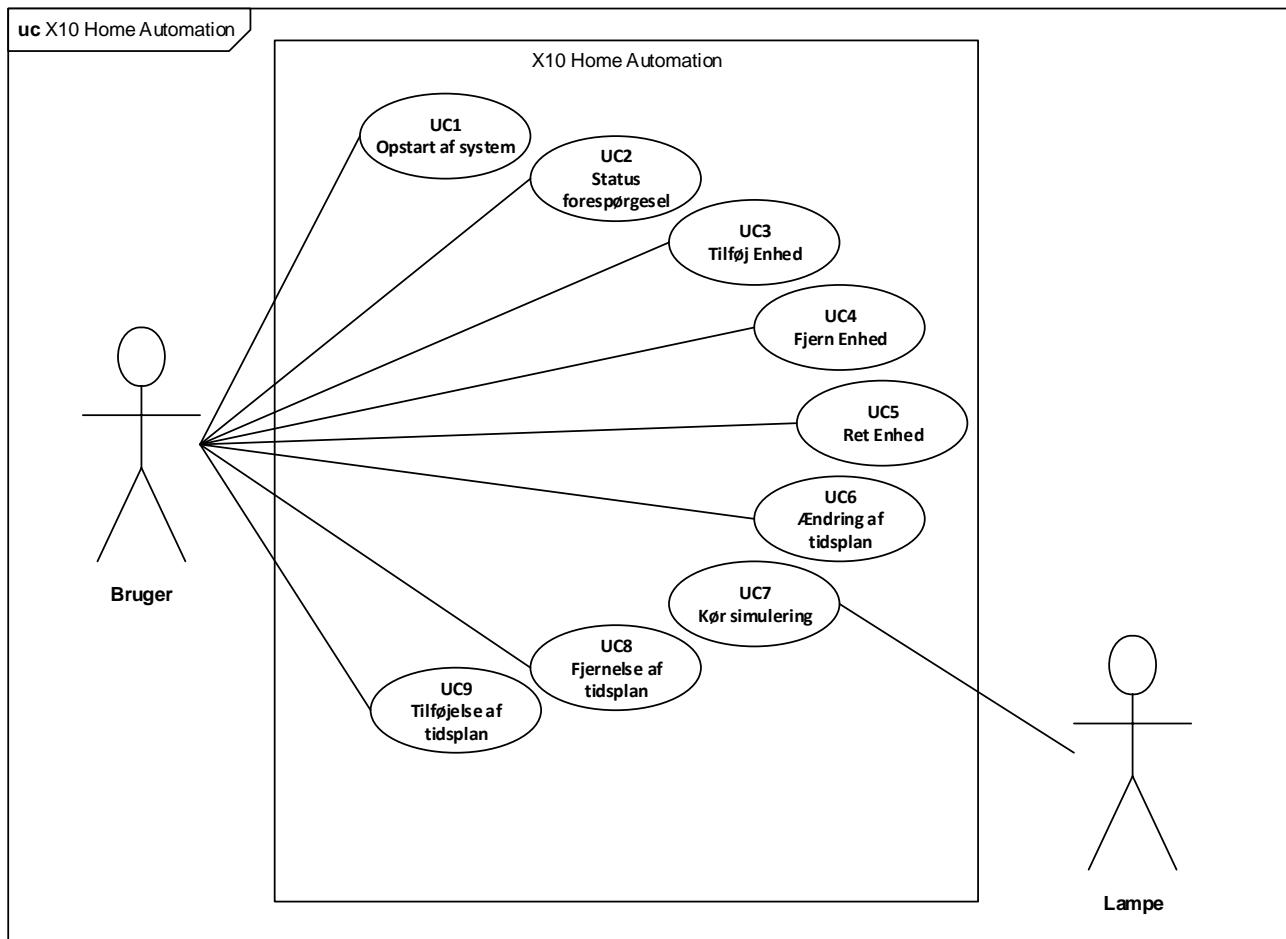
Kodelås er et eksternt interface tilkoblet styreboksen. Kodelåsen indeholder funktionalitet til at godkende/afvise en indtastet kode, og sikre at systemets indstillinger ikke kan ændres uden den korrekte kode.

### 1.2.6 Tidsplan

En tidsplan er tilknyttet en given enhed. Én tidsplan indeholder information om hvilken dag den gælder, hvilket klokkeslet den er aktiv, samt hvilken handling der skal udføres(tænde eller slukke).

### 1.3 Use Cases

På Figur 2 vises et Use Case diagram for X10 Home Automation systemet. Diagrammet har til formål at illustrere hvilke aktører der er tilknyttet til hvilke use-cases.



Figur 2 Use Case diagram for X10 Home Automation

### 1.3.1 Use Case 1: Opstart af System

---

**Navn:** Opstart af System

**Mål:** At få startet systemet op og få adgang til software

**Initiering:** Startes af bruger

**Aktører:** Bruger (primær)

**Antal samtidige forekomster:** 1

**Prækondition:** Styreboks og PC er forbundet korrekt.

**Postkondition:** Software starter op og viser grafisk brugerflade.

**Hovedscenarie:**

1. Bruger starter PC software.  
[Udvidelse 1: Software kører allerede.]
2. Software anmoder om indastning af kode.
3. Bruger indtaster kode på kodelås og trykker på godkend.  
[Udvidelse 2: Forkert kode indtastet.]
4. PC Software skifter til forsiden i grafisk bruger flade.

**Udvidelser**

[Udvidelse 1: Software kører allerede]

1. Intet nyt software vindue åbnes.

[Udvidelse 2: Forkert kode indtastes.]

1. Skærm på PC viser fejlbesked.
2. Bruger trykker "OK."
3. Fortsæt fra punkt 2.

---

### 1.3.2 Use Case 2: Status Forespørgsel

---

**Navn:** Status Forespørgsel

**Mål:** At vise en oversigt over tilsluttede enheder og deres status på PC Softwareen

**Initiering:** Startes af bruger

**Aktører:** Bruger (primær)

**Antal samtidige forekomster:** 1

**Prækondition:** UC1 er udført.

**Postkondition:** PC Software viser status oversigt.

**Hovedscenarie:**

1. Bruger vælger "Opdater Enhedsstatus"
2. PC Software viser en opdateret status på alle tilsluttede enheder.

### 1.3.3 Use Case 3: Tilføjelse af enhed

---

**Navn:** Tilføjelse af enhed

**Mål:** At brugeren tilføjer en enhed til systemet.

**Initiering:** Startes af bruger

**Aktører:** Bruger (primær)

**Antal samtidige forekomster:** 1

**Prækondition:** UC1 er udført.

**Postkondition:** Den givne enhed bliver tilføjet til systemet.

**Hovedscenarie:**

1. Brugeren tilslutter enheden til lysnettet.
2. Brugeren vælger tilføj enhed.
3. Brugeren indtaster enhedens adresse.
4. Brugeren indtaster addressen til enhedens rum.  
[Udvidelse 1: Brugeren tildeler ikke et rum.]
5. Brugeren trykker "OK".  
[Udvidelse 2: Brugeren annulerer indtastningen.]
6. PC Software viser godkendt.  
[Udvidelse 3: Adressen er allerede registreret.]
7. Skærm på PC viser hovedmenu.

**Udvidelser**

[Udvidelse 1: Brugeren tildeler ikke et rum.]

1. Fortsæt fra punkt 5.

[Udvidelse 2: Brugeren annulerer indtastningen.]

1. Skærm på PC viser hovedmenu.

[Udvidelse 3: Adressen er allerede registreret.]

1. Skærm på PC viser fejlmeddeelse.
  2. Brugeren trykker OK.
  3. Fortsæt fra punkt 7.
-

### 1.3.4 Use Case 4: Fjernelse af enhed

---

**Navn:** Fjernelse af enhed

**Mål:** At fjerne en enhed fra systemet.

**Initiering:** Startes af bruger

**Aktører:** Bruger (primær)

**Antal samtidige forekomster:** 1

**Prækondition:** UC1 er udført og mindst en enhed er registreret i systemet.

**Postkondition:** Den valgte enhed fjernes fra systemet.

**Hovedscenarie:**

1. Brugerens vælger fjern enhed
2. Brugerens vælger hvilken enhed der skal fjernes.
3. Brugerens trykker OK.
  - [Udvidelse 1: Brugerens annullerer.]
  - [Udvidelse 2: Brugerens har ikke valgt en enhed.]
4. PC Software viser Godkendt.
5. Bruger trykker "OK".
6. Skærm på PC viser hovedmenu.

**Udvidelser**

[Udvidelse 1: Brugerens annullerer]

1. Skærm på PC viser hovedmenu.

[Udvidelse 2: Brugerens har ikke valgt en enhed.]

1. Skærm på PC viser fejlmeddelelse.
  2. Brugerens trykker "OK."
  3. Fortsæt fra punkt 1.
-

### 1.3.5 Use Case 5:Rediger Enhed

---

**Navn:** Rediger Enhed

**Mål:** At rette oplysninger for en given enhed

**Initiering:** Startes af bruger

**Aktører:** Bruger (primær)

**Antal samtidige forekomster:** 1

**Prækondition:** UC1 er udført. Mindst én enhed er registreret i systemet.

**Postkondition:** Oplysningerne for den valgt enhed ændres.

**Hovedscenarie:**

1. Brugerens vælger rediger enhed.
2. Brugerens vælger hvilken enhed der skal redigeres.
3. Brugerens indtaster et nyt enheds ID.  
[Udvidelse 1: Brugerens indtaster ikke et nyt enheds ID.]
4. Bruger indtaster nyt Rum ID.  
[Udvidelse 2: Brugerens indtaster ikke et nyt Rum ID.]
5. Bruger trykker Gem Ændringer.  
[Udvidelse 3: Brugerens annullerer.]  
[Udvidelse 4: Bruger har ikke valgt en enhed til ændring.]  
[Udvidelse 5: Bruger har indtastet et nyt ID der allerede findes i systemet.]
6. PC Software viser Godkendt.
7. Brugerens trykker "OK".
8. Skærm på PC viser hovedmenu.

**Udvidelser**

[Udvidelse 1: Brugerens indtaster ikke et nyt enheds ID.]

1. Fortsæt til punkt 4.

[Udvidelse 2: Brugerens indtaster ikke et nyt Rum ID.]

1. Fortsæt til punkt 5.

[Udvidelse 3: Brugerens annullerer.]

1. Fortsæt til punkt 8.

[Udvidelse 4: Bruger har ikke valgt en enhed til ændring.]

1. Skærm på PC viser fejlbesked.
2. Bruger trykker "OK".
3. Fortsæt fra punkt 2.

[Udvidelse 5: Bruger har indtastet et nyt ID der allerede findes i systemet.]

1. Skærm på PC viser fejlbesked.
  2. Bruger trykker "OK".
  3. Fortsæt fra punkt 2.
-

### 1.3.6 Use Case 6: Ændring af tidsplan

**Navn:** Ændring af tidsplan

**Mål:** At ændre tidsplanen for en given enhed.

**Initiering:** Startes af bruger

**Aktører:** Bruger (primær)

**Antal samtidige forekomster:** 1

**Prækondition:** UC1 er udført. Der er minimum én enhed, med minimum én tidsplan tilføjet til systemet.

**Postkondition:** Den valgte tidsplan ændres.

**Hovedscenarie:**

1. Bruger vælger Tidsplan.
2. Bruger vælger ønsket enhed at ændre tidsplan for.  
[Udvidelse 1: Brugeren vælger ikke en enhed at ændre tidsplan for.]  
[Udvidelse 2: Den valgte enhed har ingen tidsplaner at ændre.]
3. Brugeren vælger tidsplanen der skal ændres.
4. Brugeren vælger hvilken dag tidsplanen skal være aktiv.
5. Brugeren indtaster starttidspunkt.
6. Brugeren indtaster sluttidspunkt.
7. Bruger trykker Gem Ændringer.  
[Udvidelse 3: Brugeren annullerer.]  
[Udvidelse 4: Bruger har indtastet sluttidspunkt før starttidspunkt.]
8. PC Software viser Godkendt.
9. Bruger trykker "OK".
10. Skærm på PC viser hovedmenu.

**Udvidelser**

[Udvidelse 1: Brugeren vælger ikke en enhed at ændre tidsplan for.]

1. Skærm på PC viser fejlbesked.
2. Bruger trykker "OK".
3. Fortsæt fra punkt 2.

[Udvidelse 2: Den valgte enhed har ingen tidsplaner at ændre.]

1. Skærm på PC viser fejlbesked.
2. Bruger trykker "OK".
3. Fortsæt fra punkt 2.

[Udvidelse 3: Brugeren annullerer.]

1. Fortsæt fra punkt 10.

[Udvidelse 4: Bruger har indtastet sluttidspunkt før starttidspunkt.]

1. Skærm på PC viser fejlbesked.
2. Brugeren trykker "OK."
3. Fortsæt fra punkt 4.

### 1.3.7 Use Case 7: Kør simulering

---

**Navn:** Kør simulering

**Mål:** Den gemte tidsplan kører som simulering

**Initiering:** Initieres af styreboks.

**Aktører:** Lampe(sekundær)

**Antal samtidige forekomster:** 1

**Prækondition:** Styreboks er konfigureret og der er tilføjet mindst 1 enhed til systemet.

**Postkondition:** Lampe tændes og slukkes på bestemte tidspunkter

**Hovedscenarie:**

1. Lampe tænder på det konfigurerede tænd tidspunkt.
2. Lampe slukker på det konfigurerede sluk tidspunkt.

**Udvidelser**

---

### 1.3.8 Use Case 8: Fjernelse af tidsplan.

---

**Navn:** Fjernelse af tidsplan

**Mål:** At fjerne en tidsplan fra en enhed.

**Initiering:** Startes af bruger

**Aktører:** Bruger (primær)

**Antal samtidige forekomster:** 1

**Prækondition:** UC1 er udført. Minimum én enhed, med minimum én tidsplan er tilføjet til systemet.

**Postkondition:** Den valgte tidsplan fjernes fra systemet.

**Hovedscenarie:**

1. Brugerens vælger tidsplan.
2. Skærm på PC viser oversigt over enheder og antal tidsplaner.
3. Brugerens vælger enhed der skal arbejdes på.  
[Udvidelse 1: Enheden har ingen tidsplan.]
4. Brugerens vælger Fjern Tidsplan.  
[Udvidelse 2: Brugerens har ingen enhed valgt.]  
[Udvidelse 3: Brugerens annulerer.]
5. Skærm på PC viser oversigt med tidsplaner for valgte enhed.
6. Brugerens vælger en tidsplan som ønskes fjernet.  
[Udvidelse 4: Brugerens ønsker at slette tidsplaner for en hel dag.]
7. Brugerens vælger Fjern Tidsplan.  
[Udvidelse 3: Brugerens annulerer.]
8. PC Software viser godkendt.
9. Brugerens trykker "OK".
10. Skærm på PC viser hovedmenu.

**Udvidelser**

[Udvidelse 1: Enheden har ingen tidsplan.]

1. Skærm på PC viser fejlbesked.
2. Fortsæt fra punkt 3.

[Udvidelse 2: Brugerens har ingen enhed valgt.]

1. Skærm på PC viser fejlbesked.
2. Fortsæt fra punkt 3.

[Udvidelse 3: Brugerens annulerer.]

1. Fortsæt fra punkt 10.

[Udvidelse 4: Brugerens ønsker at slette tidsplaner for en hel dag.]

1. Brugerens vælger hvilken dag som skal ryddes.
  2. Brugerens trykker Fjern tidsplan for valgte dag.
  3. Fortsæt fra punkt 8.
-

### 1.3.9 Use Case 9: Tilføjelse af tidsplan.

---

**Navn:** Tilføjelse af tidsplan

**Mål:** At tilføje en ny tidsplan til en eksisterende enhed.

**Initiering:** Startes af bruger

**Aktører:** Bruger (primær)

**Antal samtidige forekomster:** 1

**Prækondition:** UC1 er udført. Minimum én enhed er tilføjet til systemet.

**Postkondition:** En tidsplan tilføjes til den valgt enhed.

**Hovedscenarie:**

1. Bruger vælger Tidsplan.
2. Bruger vælger en enhed fra tabel over tilføjede enheder.
3. Bruger trykker Tilføj Tidsplan.  
[Udvidelse 1: Brugeren har ikke valgt en enhed at tilføje tidsplan til.]
4. Bruger vælger ønsket tidsplanen skal være aktiv.
5. Bruger indtaster starttidspunkt.
6. Bruger indtaster sluttidspunkt.
7. Bruger trykker Tilføj Tidsplan.  
[Udvidelse 2: Brugeren har indtastet sluttidspunkt før starttidspunkt.]  
[Udvidelse 3: Brugeren annullerer.]
8. PC Software viser Godkendt.
9. Bruger trykker "OK".
10. Skærm på PC viser hovedmenu.

**Udvidelser**

[Udvidelse 1: Brugeren har ikke valgt en enhed at tilføje tidsplan til.]

1. Skærm på PC viser fejlbesked.
2. Bruger trykker "OK".
3. Fortsæt fra punkt 2.

[Udvidelse 2: Brugeren har indtastet sluttidspunkt før starttidspunkt.]

1. Skærm på PC viser fejlbesked.
2. Bruger trykker "OK".
3. Fortsæt fra punkt 10.

[Udvidelse 3: Brugeren annullerer.]

1. Fortsæt fra punkt 10.
-

## 1.4 Yderlige tekniske krav

### 1.4.1 Enhed

1.1 Skal kunne tilsluttes dansk stikkontakt

### 1.4.2 Styreboks

2.1 Skal kunne tilsluttes dansk stikkontakt

2.2 Skal have en LCD skærm.

2.3 Skal have en LED indikator når data sendes på lysnettet.

2.4 Skal have en LED indikator når enheden er tændt.

2.5 Skal kunne genstarte i tilfælde af kritiske systemfejl.

### 1.4.3 Ikke funktionelle krav

3.1 Der skal være en gui med 5 knapper.

3.2 Systemet skal have en mean time between failure på 95%

3.3 Systemet bør kommunikere med op til 60 bit/s.

3.4 Systemet skal have en svartid på maksimalt 2 minutter.

3.5 Systemet skal kunne fungere ved tilslutning til lysnettet.

3.6 Skal kunne håndtere op til 255 enheder.

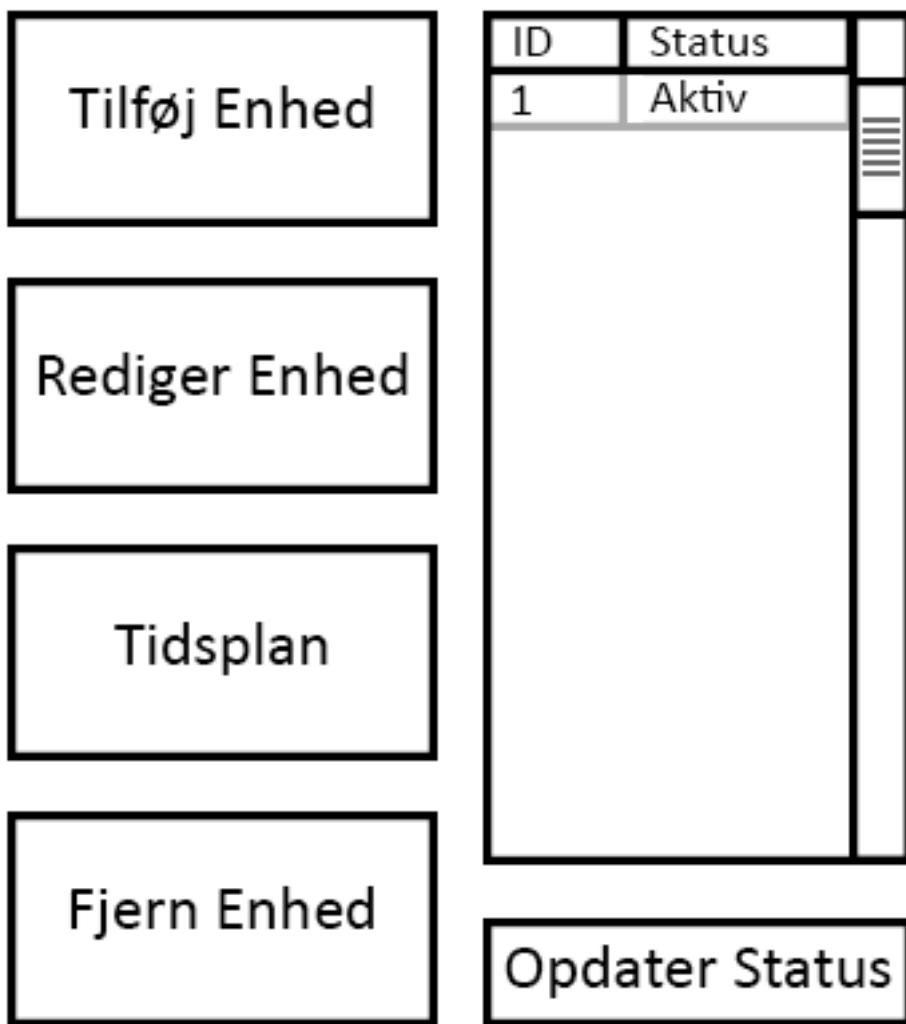
3.7 LED indikator når data sendes på lysnettet skal være gul.

3.8 LED indikator for tændt enhed skal være grøn.

## 1.5 Grafisk Brugerflade (NT, AK)

På Figur 3 vises første skitse til PC Softwareens hovedmenu. Hovedmenuen indeholder en liste over de enheder der er tilføjet til systemet, samt hvorvidt de enkelte enheder er tændte eller slukkede. Der er desuden mulighed for at navigere videre til diverse undermenuer afhængig af hvad brugeren ønsker at gøre.

Knapperne relatere sig til de enkelte use cases, hvor UC6, 8 og 9 findes under "Tidsplan".



Figur 3 Udkast til hovedmenu for PC Software

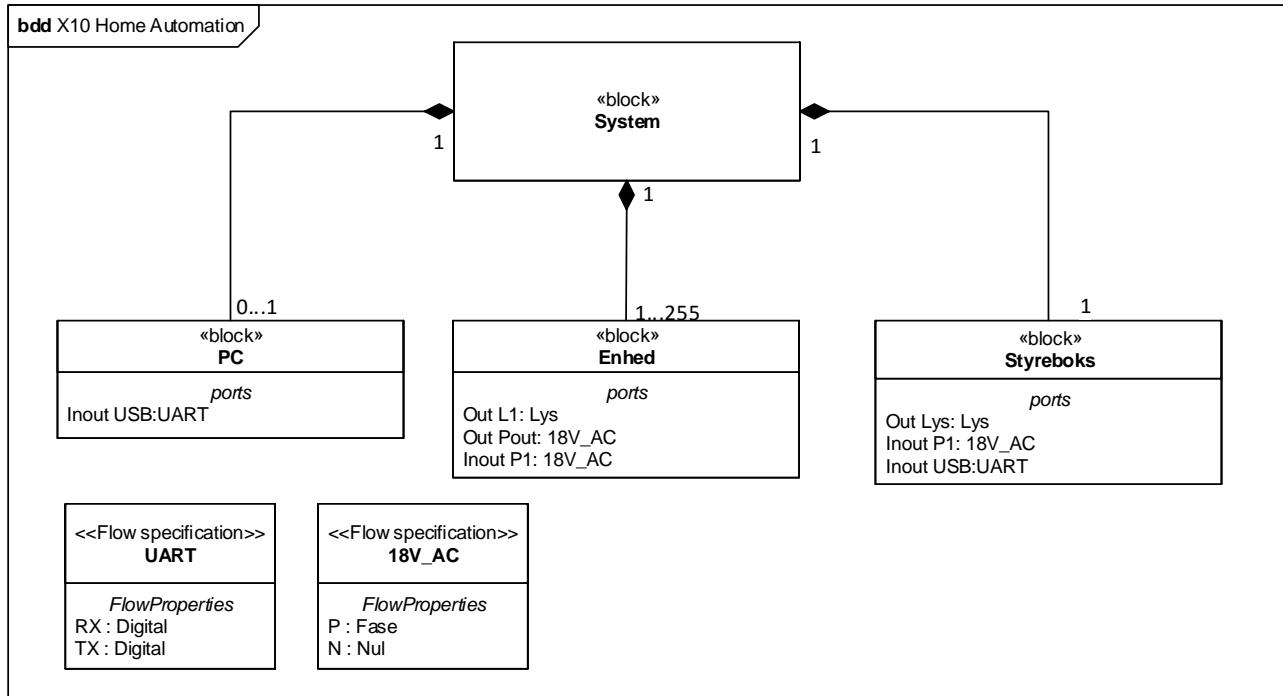
## 2 Systemarkitektur

I dette afsnit beskrives arkitekturen for X10 Home Automation. Arkitekturen dannes ud fra systemet som beskrevet i projektformuleringen og specifiseret i kravspecifikationen.

Formålet med systemarkitekturen er at identificere de blokke systemet sammensættes af, samt grænseflader imellem disse blokke. Desuden bruges systemarkitekturen som et led i projektorganisationen, da den skaber overblik over arbejdsopgaver.

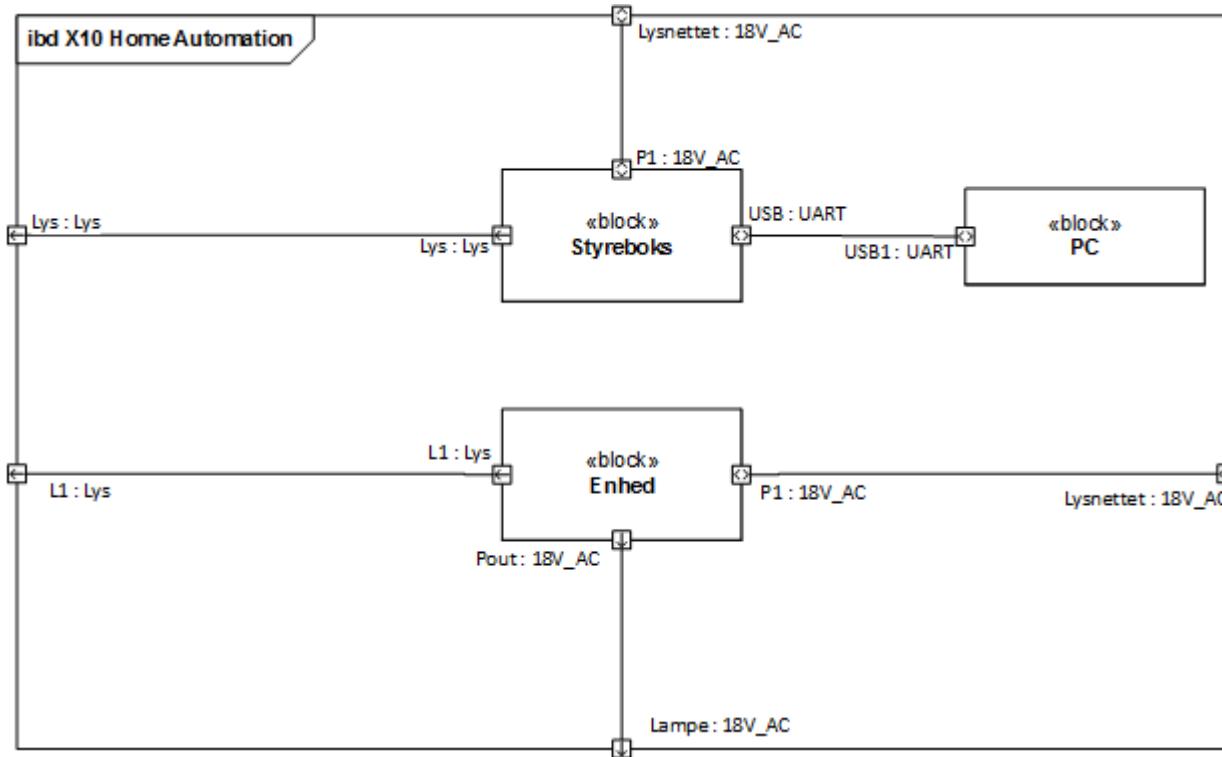
### 2.1 Overordnet Arkitektur (TF, SN, MB, DP, ME, CBJ)

Systemet der bygges er et distribueret system, som er opdelt som vist på Figur 4. Der forefindes ikke hardware diagrammer for PC, da dette er et system leveret af et eksternt parti.



Figur 4 Overordnet systemdiagram for X10 Home Automation

På Figur 5 ses et IBD for det overordnede system.



Figur 5 IDB for Home Automation

### 2.1.1 Grænseflader for det overordnede system

Grænseflade	Porte	Stiktype
<b>P1 – Enhed &amp; styreboks</b>	Input 18V AC Hz Input X10.1 Output 18 V AC 50 Hz Output X10.1 GND	Stik af dansk lovkrav
<b>USB1 – Styreboks</b>	Input 5V DC Input max 0,5 A Input D- Input D+ GND	USB-stik type B hun (4-pin)
<b>USB1 – PC</b>	Input 5V DC Input D- Input D+ GND	USB-stik type A hun (4-pin)
<b>Pout – Enhed</b>	Output 18V AC 50 Hz GND	Stik efter dansk lovkrav

## 2.1.2 Signaler for overordnet IBD

Signaltypen	Definition	Beskrivelse
<b>18V_AC</b>	18V AC 50 Hz signal kombineret med et 100kHz 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.
<b>Lys</b>	Lys i det synlige spektrum	Lys i 3 farver afhængig af hvilket LED indikator der lyser.
<b>Uart</b>	Kommunikation følger UART-standarden	Protokollen udvikles specifikt til dette produkt. Se protokol afsnit for yderligere information.

## 2.1.3 Beskrivelser

**USB: UART** er en kommunikation der står for at sende konfigurationsdata fra pc til styreboks samt at sende informationer om systemets nuværende status og konfiguration fra styreboks til pc.

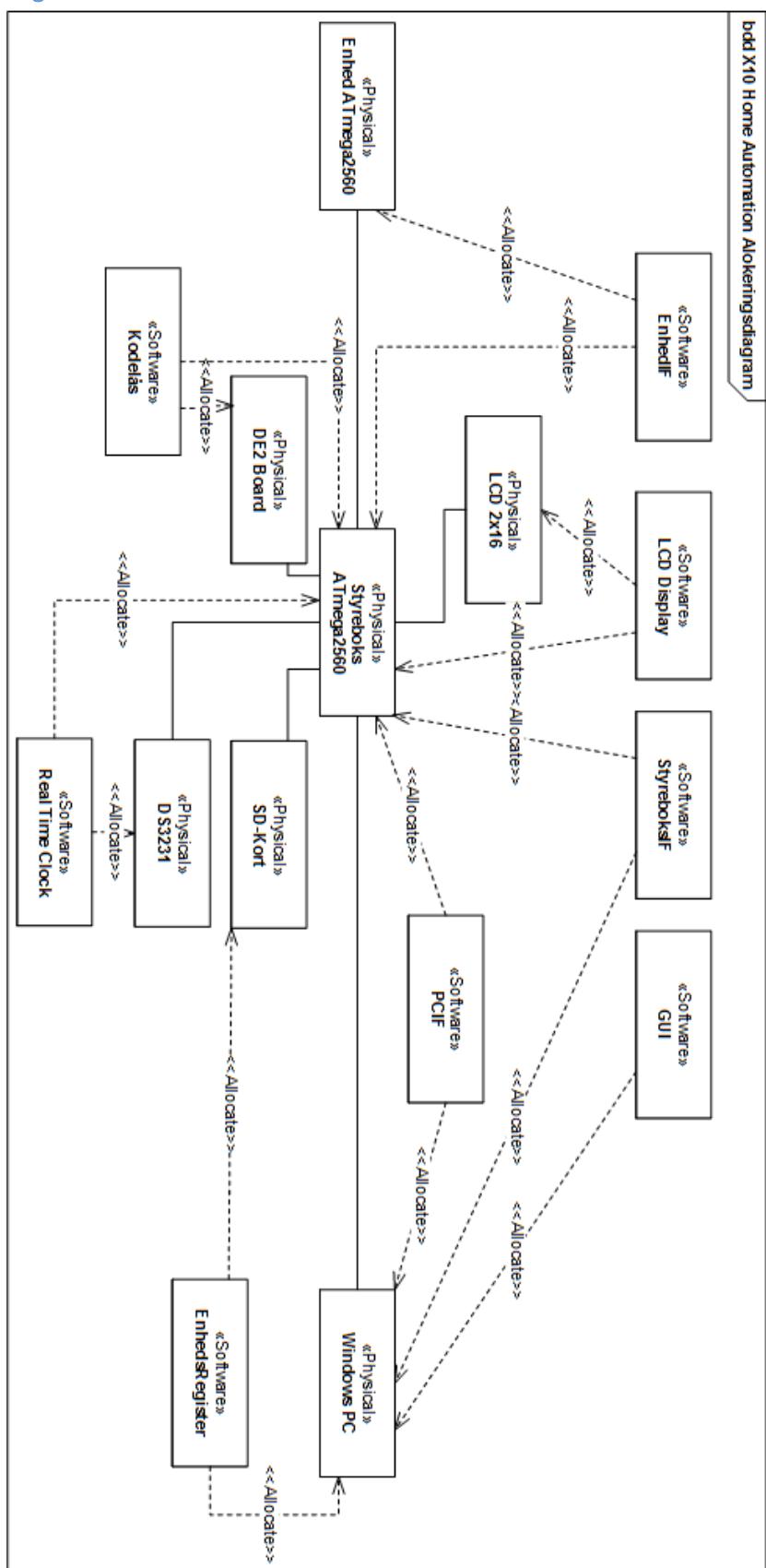
**LYS:LYS** er et synligt lyssignal fra LED indikatorerne på styreboksen der viser om styreboksen er korrekt tilsluttet lysnettet og om der er igangværende kommunikation på X10.1 eller Uart til Pc'en.

**L1:LYS** er et synligt lyssignal fra LED indikatorerne på enheden der viser om enheden er tilsluttet korrekt til lysnettet samt om der er igangværende kommunikation på X10.1.

**Pout: 18V\_AC** er et 18V AC 50 Hz signal der fungere som spændingsforsyning til en lampe, dette signal anvendes til at styre om lampen er tændt eller slukket.

**P1: 18V\_AC** er et 18V AC 50 Hz signal som samtidig indeholder et kommunikationssignal der følger X10.1 protokollen. Signalet fungere som spændingsforsyning til systemets blokke samt som kommunikationssignal mellem styreboksen og enhederne via X10.1 protokollen.

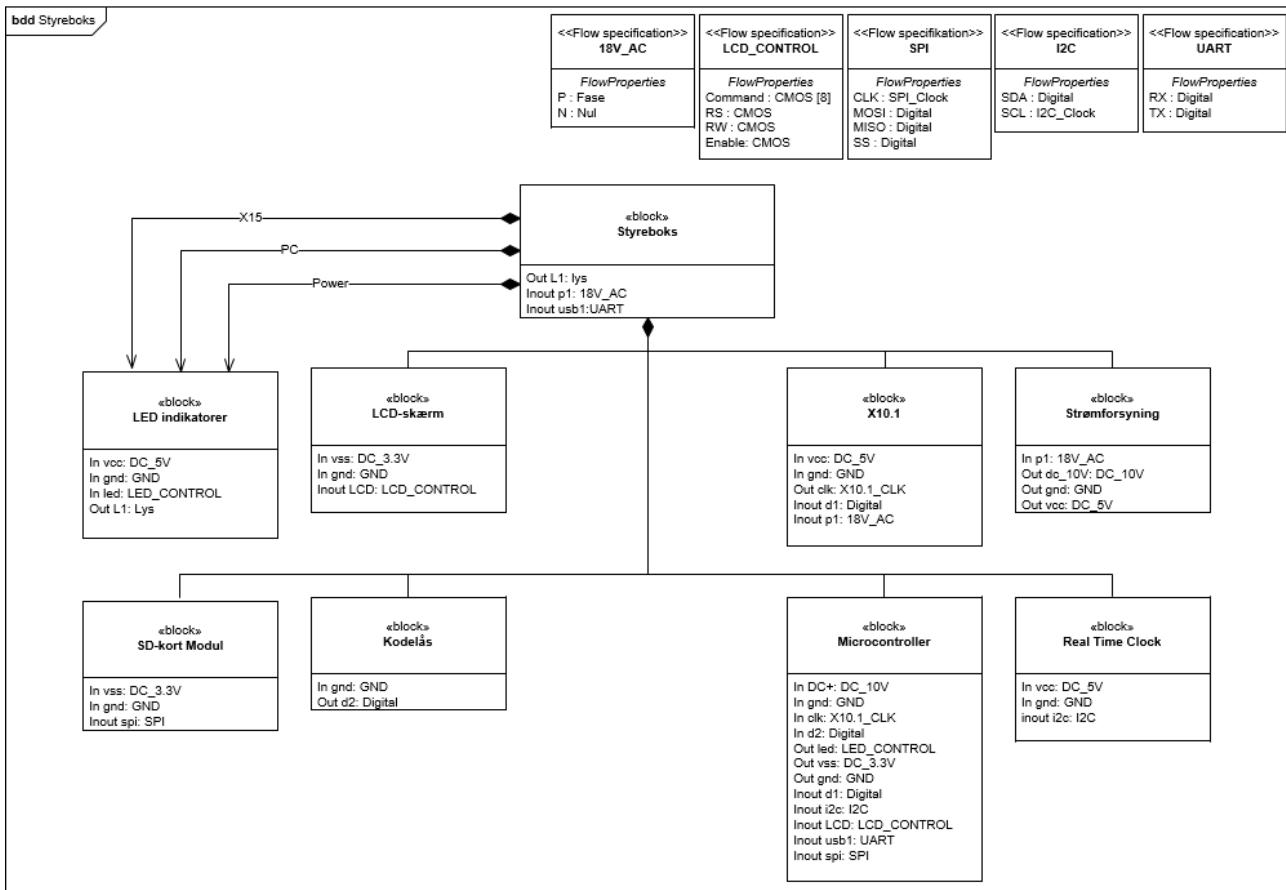
### 2.1.5 Allokeringsdiagram



Figur 6 Allokeringsdiagram for X10 Home Automation

## 2.2 Styreboks (TF, SN, MB, CBJ, DP, ME)

Analyse af styreboksen, resultere i følgende BDD vist på Figur 7. Styreboksen vil indeholde følgende blokke: **LED Indikatore**, en **LCD-skærm**, en **Microcontroller**, en **Strømforsyning**, en **Real Time Clock**, et **SD-kort Modul**, en **Kodelås**, samt en **X10.1 Sender/Modtager**. Disse blokke beskrives yderligere i det kommende afsnit.



Figur 7 BDD og flow specifikationer for styreboks

## 2.2.1 Blokbeskrivelser: Styreboks

Funktionaliteten af de enkelte blokke i styreboksen beskrives i dette afsnit.

**Kodelås** har til formål at forhindre ændringer i systemets indstillinger af uautoriserede brugere. Kodelåsen sender et højt eller lavt signal til **microcontrolleren** på styreboksen, når den er henholdsvis låst eller åben.

Styreboksens **strømforsyning** bruges til at omdanne en 18V AC spænding til 5V og 10V DC spænding. De 5V DC er forsyningsspænding til **Real Time Clock**, **X10.1** og **LED-indikatorer**. De 10V DC er spændingsforsyning til **microcontrolleren**.

**Microcontrolleren** bruges som en central computer, der håndterer kommunikation mellem blokkene internt i styreboksen, samt til eksterne moduler som PC og Enheder. **Microcontrolleren** er desuden ansvarlig for at læse nuværende tid fra **Real Time Clock**, og sammenligne det med eventuelle handlinger til systemets enheder.

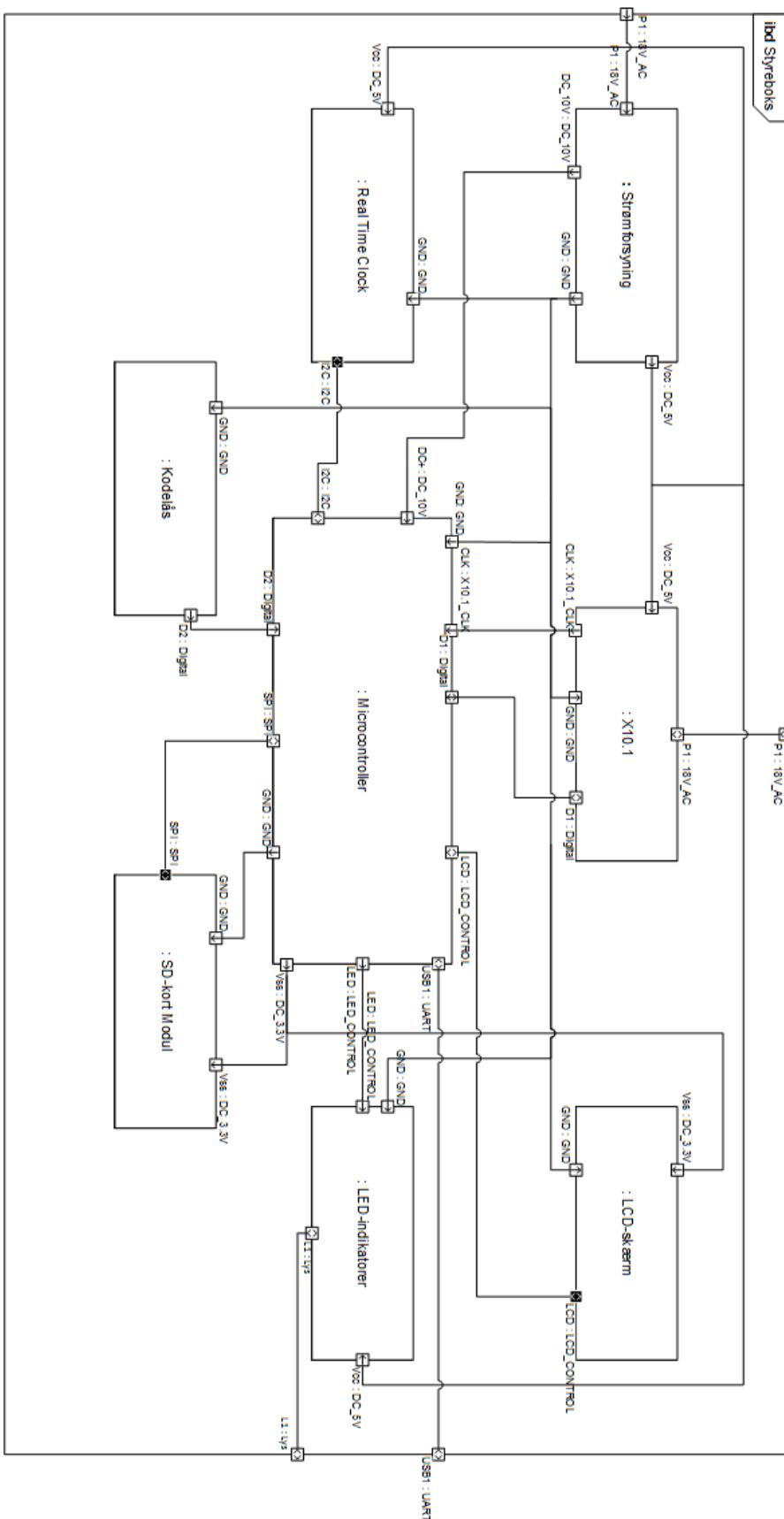
Systemet lagre alle informationer vedrørende enheder og tilhørende handlinger på **SD-Kort Modulet**. Dette gøres da **SD-Kortet** er non-volatile hukommelse, og vil derfor forblive gemt ved evt. Strømafbrydelse eller frakobling af systemet. Et **SD-Kort** benyttes da det maksimale antal enheder er sat til 255, hvilket resultere i en stor mængde potentielle handligner der skal lagres.

**LCD-Skærmen** benyttes til at give et overblik over eventuelle fejl i systemet og andre kritiske meddelelser, uden brugeren skal forbinde computeren.

**X10.1** blokken består af et sender og modtager kredsløb. Dette kredsløb benyttes til at sende kommandoer til enheder, samt at modtage svar og status for individuelle enheder.

På styreboksen forefindes tre **LED-indikatorer** der viser: Om der transmitteres data via X10.1 modulet, om der transmitteres data via UART til PC og om styreboksen er tændt.

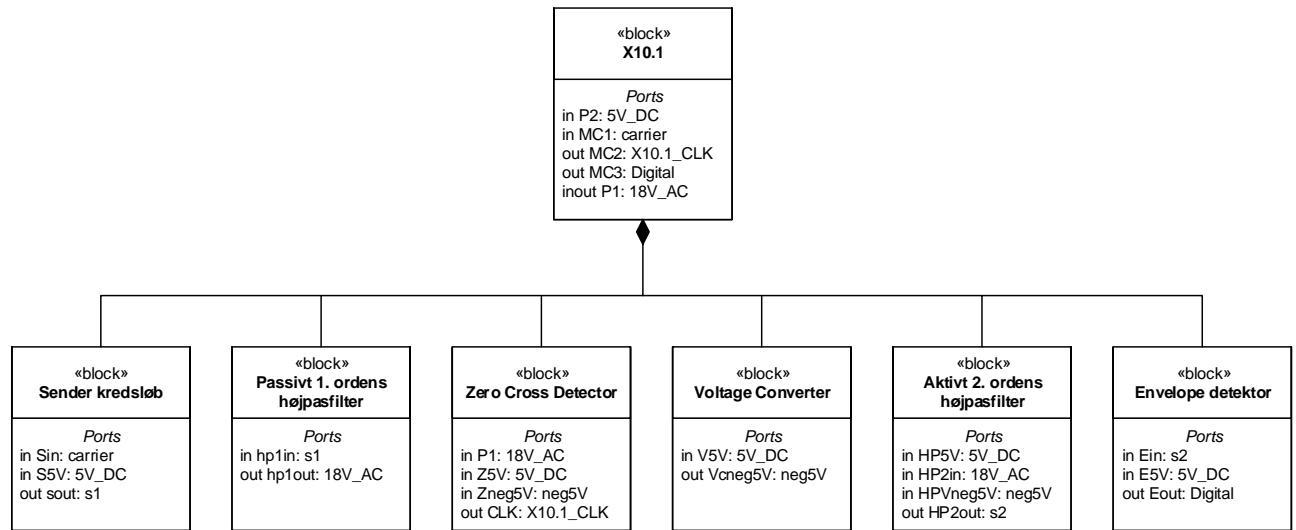
Det logiske flow mellem ovennævnte blokke vises på Figur 8, der viser et IBD for Styreboksen. Som udgangssignaler findes kommunikation mellem PC og styrebox som UART kommunikation, samt ind og udgangssignaler til X10.1 modulet.



Figur 8 IBD for Styreboks

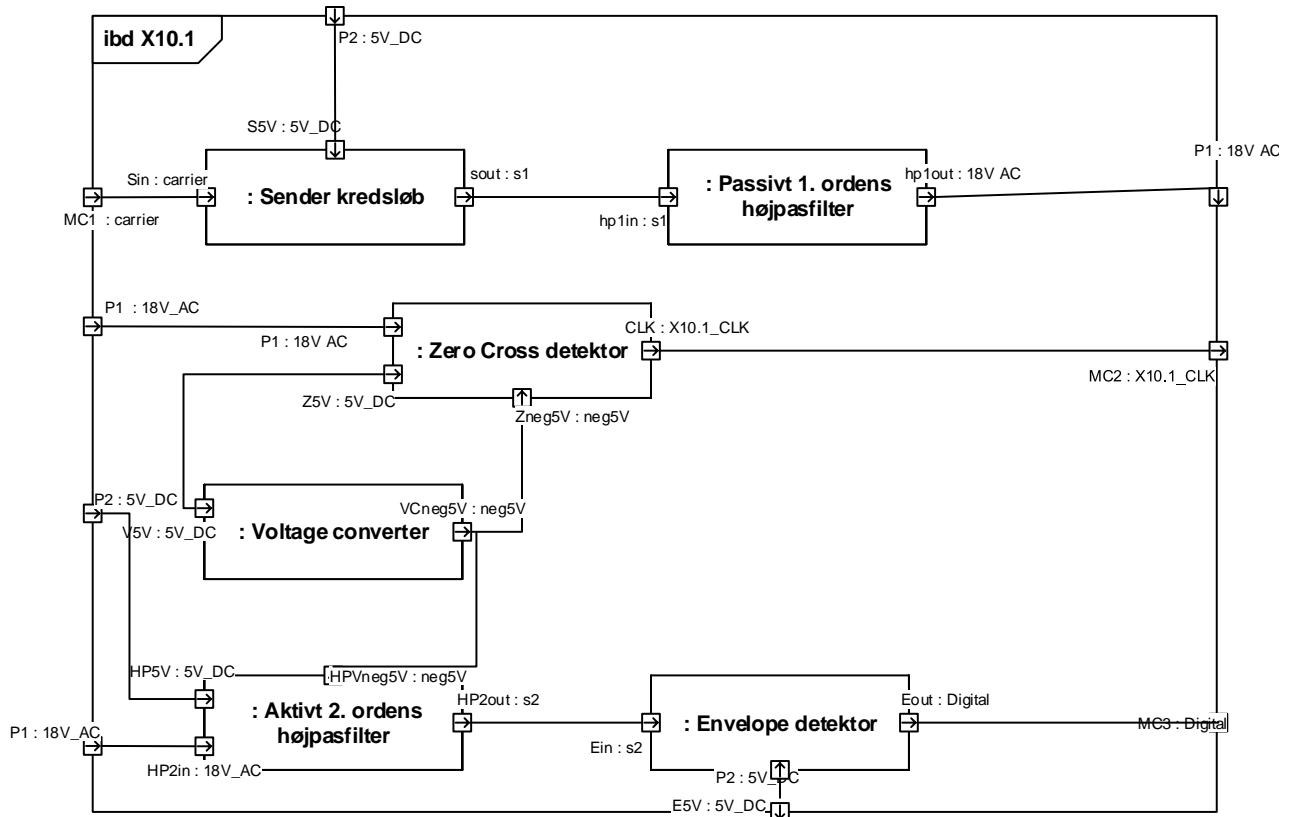
## 2.2.2 Dekomponering af X10.1 Modul

X10.1 modulet kan brydes ned i flere individuelle blokke som vist på Figur 9. Modulet består af et **Sender Kredsløb**, et **1. ordens højpasfilter**, en **Zero-Cross Detector**, en **Voltage Converter**, et **2. ordens højpasfilter** samt en **envelope detektor**.



Figur 9 BDD for X10.1 Modul

Herunder findes IBD for X10.1 modulet, der beskriver de interne forbindelser og signaler for blokkene.



### 2.2.2.1 Signaler for X10.1 Modul

Signaltypen	Definition	Beskrivelse
<b>18V_AC</b>	18V AC 50 Hz signal kombineret med et 120kHz X10.1 signal	18V AC 50 Hz signal fra forsyningsnettet der også indeholder X10.1 kommunikation.
<b>5V_DC</b>	+5V DC signal	Forsyningsspænding til moduler der har behov for 5V DC spænding. Signalet er et 5V DC signal +/- 0,5V
<b>Digital</b>	0-5V DC signal	Digitalt signal 3V til 5V = 1 (høj) 0V til 1,5V = 0 (lav)
<b>X10.1_CLK</b>	0-5V timing signal	0-5V timing signal der er genereret af zero-cross detektoren til at bestemme hvornår der skal aflæses kommunikation ud fra X10.1
<b>carrier</b>	120 kHz PWM-signal på 5V	PWM-signal genereret af microcontroller.
<b>neg5V</b>	-5V DC signal	Forsyningsspænding til moduler der har behov for -5V DC spænding. Signalet er et -5V DC signal +/- 0,5V
<b>s1</b>	Elektrisk signal 0-5 V	Behandlet elektrisk signal mellem sender kredsløb og passivt 1. ordens højpasfilter.
<b>s2</b>	Elektrisk signal 0-5 V	Behandlet elektrisk signal mellem aktivt 2. ordens højpasfilter og envelope detektor.

## 2.2.3 Signalbeskrivelse: Styreboks

Herunder findes beskrivelser af grænseflader og signaler vist på BDD og IBD for styreboksen

### 2.2.3.1 Grænseflader

Grænseflade	Porte	Stiktype
P1 – X10.1	Input/Output 18 V AC 50 Hz Input/Output X10.1	-
P1 – Strømforsyning	Input 18 V AC 50 Hz Output 18 V AC 50 Hz	Stik efter dansk lovkrav
LED – LED-indikatorer	Input L0 til L2 min 0 - 5 V DC	-
I2C – Real Time Clock	Input/Output SCA 0 – 5 V DC Input/Output SCL 0 – 5 V DC	-
SPI – SD-modul	Input CLK 1 MHz Input SS 0 – 3.3 V DC Input/Output MOSI 0 – 3.3 V DC	-
LCD – LCD-skræm	Input D4 til D7 0 – 3.3 V DC Input RS: 0 – 3.3 V DC Input EN: 0 – 3.3 V DC Input RW: 0 – 3.3 V DC	-
I2C – Microcontroller	Input/Output SCA 0 – 5 V DC Input/Output SCL 0 – 5 V DC	-
LED - Microcontroller	Output L0 til L2 0 – 5 V DC	-
SPI – Microcontroller	Output CLK 1 MHz Output SS 0 – 3.3 V DC Input/Output MOSI 0 – 3.3 V DC	-

### 2.2.3.2 Signaler

Signaltypen	Definition	Beskrivelse
<b>18V_AC</b>	18V AC 50 Hz signal kombineret med et 100kHz X10.1 signal	18V AC 50 Hz signal fra forsyningsnettet der også indeholder X10.1 kommunikation.
<b>Lys</b>	Lys i det synlige spektrum	Lys i 3 farver afhængig af hvilket LED indikator der lyser.
<b>DC_10V</b>	+10V DC signal	Forsyningsspænding til Arduino mega2560 microcontroller der har egen dc converter monteret. Signalet er et 10V DC signal +/- 0,5V
<b>DC_5V</b>	+5V DC signal	Forsyningsspænding til moduler der har behov for 5V DC spænding. Signalet er et 5V DC signal +/- 0,5V
<b>DC_3.3V</b>	+3.3V DC signal	Forsyningsspænding til moduler der har behov for 3.3V DC spænding Signalet er et 3.3V DC signal +/- 0,2V
<b>GND</b>	0V DC signal	Reference spænding til DC forsyningsspændinger.
<b>Digital</b>	0-5V DC signal	Digitalt signal 3V til 5V = 1 (høj) 0V til 1,5V= 0 (lav)
<b>X10.1_CLK</b>	0-5V timing signal	0-5V timing signal der er genereret af zero-cross detektoren til at bestemme hvornår der skal aflæses kommunikation ud fra X10.1
<b>LED_CONTROL</b>	0-5V digitale signaler til styring af LED indikatorer	To 0-5V digitale signaler der styrer power indikator LED samt X10.1 kommunikationsindikator LED der begge er active high. 3V til 5V = 1 (høj) 0V til 1,5V= 0 (lav)
<b>LCD_CONTROL</b>	0-3.3V digitale signaler til styring af LCD display	Kommunikation til LCD display som følger kommunikationsprotokollen for LCD display, se protokol afsnit for yderligere information.
<b>I2C</b>	Kommunikation der følger I2C standarden	Kommunikation via I2C standarden til Real Time Clock. Se protokol afsnit for yderligere information.
<b>SPI</b>	Kommunikation der følger SPI protokollen	Kommunikation via SPI til styring af SD-kort. Se protokol afsnit for yderligere information.

### *2.2.3.3 Beskrivelse af Signaler*

**L1:LYS** er et synligt lyssignal fra LED indikatorerne på enheden der viser om enheden er tilsluttet korrekt til lysnettet samt om der er igangværende kommunikation på X10.1 eller UART.

**P1: 18V\_AC** er et 18V ac 50 Hz signal som samtidig indeholder et kommunikationssignal der følger X10.1 protokollen. Signalet fungere som spændingsforsyning til systemets blokke samt som kommunikationssignal mellem styreboksen og enhederne via X10.1 protokollen.

**LED:LED\_CONTROL** er 3 digitale signaler 0-5V der styre de 3 LED'er der indikere om enheden modtager forsyningsspænding fra forsyningsnettet samt om der kommunikeres på lysnettet via X10.1 protokollen.

**LCD:LCD\_CONTROL** er en digital kommunikationsbus til styring af LCD displayet.

**D1: Digital** er et digital 0-5V serielt signal til datatransmission mellem X10.1 og microcontrolleren hvor 3V til 5V = 1 og 0V til 1,5V = 0.

**D2: Digital** er et digitalt 0-5V signal der er active-high som fortæller microcontrolleren om koden er korrekt indtastet hvor de digitale signaler er repræsenteret ved følgende spændinger: 3V til 5V = 1 og 0V til 1,5V= 0.

**CLK: X15\_CLK** er et timingsignal der fortæller microcontrolleren hvornår D1 kan aflæses eller skrives til i forbindelse med kommunikation via X10.1 protokollen.

**Vcc: DC\_5V** er et +5V (+/- 0,5V) dc signal der forsyner 5V modulerne med spænding.

**Vss: DC\_3.3V** er et +3.3V (+/- 0,2 V) dc signal der forsyner SD-kort modulet med 3.3V dc.

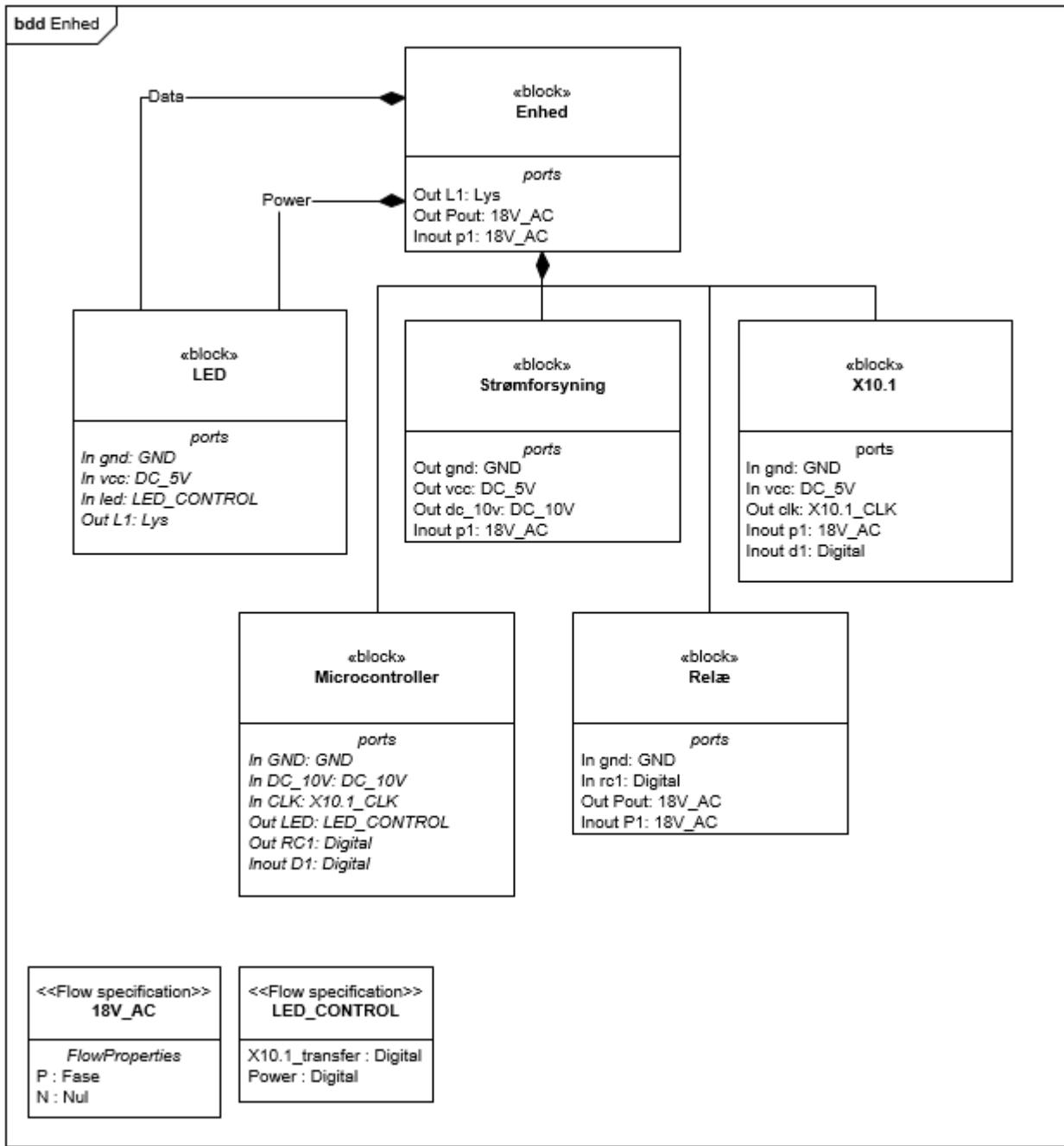
**DC\_10V:DC\_10V** er et +10V (+/- 0,5 V) dc signal der fungere som spændingsforsyning for microcontrolleren.

**GND: GND** er 0V reference spændingen til DC forsyningsspænderne på 5V og 10V.

**USB1:UART** er et kommunikationssignal til kommunikation med pc-softwaren.

## 2.3 Enhed (TF, SN, MB, CBJ, DP, ME)

Den ønskede funktionalitet fra en Enhed er blevet analyseret, og illustreret i det følgende afsnit. På Figur 10 ses BDD for enheden. Enheden består af en **strømforsyning**, et **X10.1** sender/modtager modul, et **relæ**, en **microcontroller** samt en/flere LED'er. For beskrivelser af **strømforsyning** samt **X10.1** modul henvises til afsnit 2.2.1.



Figur 10 BDD for Enhed

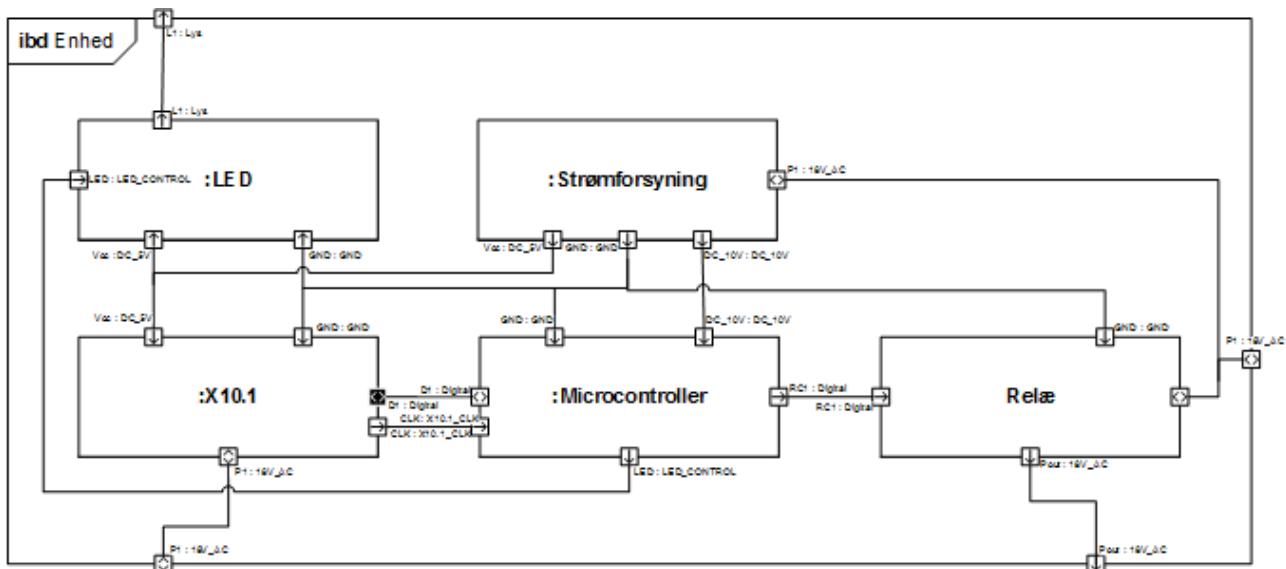
### 2.3.1 Blokbeskrivelser: Enhed

Enheden indeholder en **microcontroller** der er ansvarlig for at tolke kommunikationen der modtages og sendes vha. **X10.1** modulet. **Microcontrolleren** håndterer også tænd og sluk af relæ modulet.

**Relæet** på enheden fungerer som switch til det tilsluttede apparat, og vil tænde og slukke baseret på input fra **microcontroller**.

**LED** der indikere når der transmitteres data på X10.1 modulet, og en LED der indikere at enheden er tændt.

Ud fra BDD og blokbeskrivelserne udarbejdes et IBD der ses på Figur 11. Herpå vises de interne forbindelser for enheden, samt de signaler der sendes imellem.



Figur 11 IBD for enhed

### 2.3.2 Signalbeskrivelser: Enhed

Herunder findes beskrivelserne af signalerne der ses på BDD og IBD for enheden.

#### 2.3.2.1 Grænseflader

Grænseflade	Porte	Stiktype
P1 – strømforsyning	Input 18 V AC 50 Hz Output 18 V AC 50 Hz	Stik efter dansk lovkrav
LED – LED	Input 0 – 5 V DC	-
P1 – X15	Input 18 V AC 50 Hz Input X10.1 Output 18 V AC 50 Hz Output X10.1	-
P1 – Relæ	Input 18 V AC 50 Hz Output 18 V AC 50 Hz	-
18V_AC – Relæ	Output 18 V AC 50 Hz	-

### 2.3.2.2 Signaler

Signaltypen	Definition	Beskrivelse
<b>18V_AC</b>	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.
<b>Lys</b>	Lys i det synlige spektrum	Lys i 2 farver afhængig af hvilket LED indikator der lyser.
<b>DC_10V</b>	+10V DC signal	Forsyningsspænding til vores arduino mega2560 microcontroller der har egen dc converter monteret. Signalet er et DC 10V signal +/- 0.5V
<b>DC_5V</b>	+5V DC signal	Forsyningsspænding til moduler der har behov for 5V dc spænding. Signalet er et DC signal på 5V +/- 0.5V
<b>GND</b>	0V DC signal	Reference spænding til DC forsyningsspændinger samt til digitale signaler.
<b>Digital</b>	0-5V DC signal	Digitalt signal 3V til 5V = 1 0V til 1.5V = 0
<b>X10.1_CLK</b>	0-5V timing signal	0-5V timing signal der er genereret af zero-cross detektoren til at bestemme hvornår der skal aflæses kommunikation ud fra X10.1 protokollen. Se protokol afsnit for yderligere information.
<b>LED_CONTROL</b>	0-5V digitale signaler til styring af LED indikatorer	2 0-5V digitale signaler der styre power indikator LED samt X10.1 kommunikationsindikator LED der begge er active high.

### *2.3.2.3 Beskrivelse af signaler*

**L1:LYS** er et synligt lyssignal fra LED indikatorerne på enheden der viser om enheden er tilsluttet korrekt til lysnettet samt om der er igangværende kommunikation på X10.1.

**Pout: 18V\_AC** er et 18V AC 50 Hz signal der fungere som spændingsforsyning til en lampe, dette signal anvendes til at styre om lampen er tændt eller slukket.

**P1: 18V\_AC** er et 18V AC 50 Hz signal som samtidig indeholder et kommunikationssignal der følger X10.1 protokollen. Signalet fungere som spændingsforsyning til systemets blokke samt som kommunikationssignal mellem styreboksen og enhederne via X10.1 protokollen.

**LED:LED\_CONTROL** er 2 digitale signaler 0-5V hvor 3V til 5V = 1 og 0V til 1,5V= 0 som styrer de 2 LED'er der indikere om enheden modtager forsyningsspænding fra forsyningsnettet samt om der kommunikeres på lysnettet via X10.1 protokollen.

**D1: Digital** er et digital 0-5V serielt signal til datatransmission mellem X10.1 og microcontrolleren hvor de digitale 1 og 0 er repræsenteret som spændingerne 3V til 5V = 1 og 0V til 1,5V= 0.

**CLK: X15\_CLK** er et timingsignal der fortæller microcontrolleren hvornår D1 kan aflæses eller skrives til i forbindelse med kommunikation via X10.1 protokollen.

**Vcc: DC\_5V** er et +5V dc (+/- 0,5V ) signal der forsyner 5V modulerne med spænding.

**DC\_10V:DC\_10V** er et +10V (+/- 0,5V ) dc signal der fungere som spændingsforsyning for microcontrolleren.

**GND: GND** er 0V reference spændingen til DC forsyningsspænderne på 5V og 10V.

**RC1:Digital** er et digitalt 0-5V signal der driver relæet der styre om der er et 18V AC 50 Hz udgangssignal til den tilsluttede lampe hvor 3V til 5V = 1 og 0V til 1,5V= 0

## 2.4 Protokolbeskrivelse (ME, CBJ, MB)

I følgende afsnit beskrives protokollerne der bruges til kommunikation mellem systemets blokke.

### 2.4.1 X10.1

Til at kommunikere mellem Styreboks og enhed benyttes X10.1 kommunikation.

X10.1 kommunikation sendes 1 bit ad gangen hvorfor en kommando kan opdeles i flere pakker. Fast for hver kommando er 4 startbits, 4 huskode bits, 8 adressebits, 3 typebits, nul eller flere databits, en paritetsbit og 6 slutbits.

Startbits (STX): Fortæller at en besked er på vej. Forekommer altid som: '1110'

Huskode: Indeholder koden for det hus hvor enheden befinder sig

Adresse: Indeholder adressen på enheden

Typebits: Definerer antal databit i beskeden og kommandoens svaret i datapakken. Se afsnittet *typeliste* for uddybning.

Databits: Indeholder bools eller fejlkoder

Paritetsbits: Validerer om besked er modtaget intakt. Der benyttes even parity (se terminologi liste for uddybning). Forekommer altid som: '0' (true) eller '1' (false)

Slutbits (ETX): Fortæller at en besked er slut. Forekommer altid som: '000000'

I X10.1 protokollen er alle lige typenumre hvor Styreboks tilgår Enhed, oftest kommandoer. Alle ulige typenumre er hvor Enhed tilgår Styreboks, alle svar.

I oversigten over protokollerne er startbits, paritetsbit og slutbits ikke synlige eftersom de altid er en del af beskeden, og ikke defineret af brugeren. Eks:

STX	Huskode	Adresse	Type	Data	Paritet	ETX
-----	---------	---------	------	------	---------	-----

Alle beskeder bliver altid bekræftet med en godkendelsesbesked eller en fejlmeddeelse.

#### 2.4.1.1 Tabeloversigt over X10.1 kommunikation.

Nedenstående tabeller beskriver de forskellige datapakker der sendes mellem Styreboks og Enhed.

Tabellerne er opdelt efter følgende eksempler:

Use case der tages udgangspunkt i	
Metode/funktion	
Beskrivelse af type og metode/funktion	
Retning på dataflow: Fra > Til	
Huskode:	'4-cifret binært tal'
Adresse:	'8-cifret binært tal'
Type:	'3-cifret binært tal'
Data:	Bitnummer: data

UC7: Kør Simulering (eksempel)	
switchState(unitID)	
Sender ny status	
Styreboks > Enhed	
Huskode:	'0001'
Adresse:	'0001-0001'
Type:	'010'
Data:	b0: true

"UC7: Kør Simulering (eksempel)" som datapakke i bits:

STX	Huskode	Adresse	Type	Data	Paritet	ETX
'1110'	'0001'	'0001-0001'	'010'	'1'	'0'	'000000'

#### 2.4.1.2 Specifikke kommandoer

##### Kommando

UC2: Statusforespørgsel	
getUnitStatus(unitID)	
Anmoder om status på enheden	
Styrebooks > Enhed	
Huskode:	'XXXX'
Adresse:	'XXXX-XXXX'
Type:	'000'
Data:	-

##### Svar

UC2: Statusforespørgsel	
status	
Sender status på enheden	
Enhed > Styreboks	
Huskode:	'XXXX'
Adresse:	'XXXX-XXXX'
Type:	'001'
Data:	b0: bool

UC7: Kør Simulering	
switchState(unitID)	
Sender ny status	
Styrebooks > Enhed	
Huskode:	'XXXX'
Adresse:	'XXXX-XXXX'
Type:	'010'
Data:	b0: bool

UC7: Kør Simulering	
Return true	
Sætter enhed i ny tilstand	
Enhed > Styreboks	
Huskode:	'XXXX'
Adresse:	'XXXX-XXXX'
Type:	'011'
Data:	b0: bool

#### 2.4.1.3 Generelle Svar

Validated	
Besked modtaget og forstået	
Styrebooks > Enhed	
Huskode:	'XXXX'
Adresse:	'XXXX-XXXX'
Type:	'100'
Data:	-

Error	
Besked modtaget men ikke forstået	
Styrebooks > Enhed	
Huskode:	'XXXX'
Adresse:	'XXXX-XXXX'
Type:	'110'
Data:	b0-b1: Fejlkode

Validated	
Besked modtaget og forstået	
Enhed > Styreboks	
Huskode:	'XXXX'
Adresse:	'XXXX-XXXX'
Type:	'101'
Data:	-

Error	
Besked modtaget men ikke forstået	
Enhed > Styreboks	
Huskode:	'XXXX'
Adresse:	'XXXX-XXXX'
Type:	'111'
Data:	b0-b1: Fejlkode

#### 2.4.2 PC Styreboks Forbindelse(PSF) (TF, NT)

PC styreboks forbindelse definere kommunikationsprotokollen der anvendes mellem pc og styreboks i forbindelse med administration af styreboksen.

For PSF bruges UART med 8 databits og even paritetsbit, der anvendes en baud-rate på 57600 for at mindske tiden det tager at overføre data mellem pc og styreboks, men stadig opretholde en stabil overførsel.

Kommandoer sendt fra PC sendes efter følgende skabelon:

Start	CMD	Data	Stop
0xF0F0	0x**	Antal data bytes afhænger af kommando	0X0F0F

Som standardsvar fra styreboks bruges 0XF som godkendt(ACK) kommando.

Nedenfor ses en oversigt over kommandoer der sendes fra PC til styreboks, svar repræsenterer det der sendes tilbage fra styreboksen. Efterfølgende findes mere detaljerede beskrivelser af kommandoerne:

- Hent enheder
- Send tidsplaner
- Set nuværende tid

CMD	Beskrivelse	Antal Data	Svar
0x01	PC tilsluttet	0 bytes	ACK
0x02	PC frakoblet	0 bytes	ACK
0x03	Validate Pin	0 bytes	1 byte ( 7 don't care, 1 true false) Least significant bit er true/false bit
0x05	Anmod om enheds status	1 byte: enheds ID	0x01 hvis enhed er tændt. 0x00 hvis enhed er slukket.
0x06	Hent enheder	0 bytes	3584 Bytes. Enheds ID Rum ID Eventuelle tidsplaner
0x07	Send enhed	2 bytes: Enheds ID og Rum ID	ACK
0x08	Slet Enhed	1 byte: Enheds ID	ACK
0x09	Ret Enhed	3 bytes: Nuværende Enheds ID, nyt Enheds ID, nyt Rum ID.	ACK

0x0A	Send tidsplaner	Ved Ingen tidsplaner: 4 Bytes. Ellers 7-83 Bytes	ACK for hver modtaget byte.
0x0B	Set nuværende tid	7 bytes	ACK

#### *2.4.2.1 Hent Enheder*

Ved hent enheder sendes hent enhed kommandoen, hvorefter styreboks svarer tilbage med 3584 bytes hvis der er en enhed at sende, eller 0x00 hvis der ikke er nogen enhed at sende.

Hver enhed består af en 512 byte blok for hver dag. Hver blok indeholder enhedens ID, enhedens rum ID, hvilken dag tidsplanerne i den enkelte blok tidsplanerne hører til, tidsplanens ID, tidsplanens starttidspunkt, tidsplanens sluttidspunkt og tidsplanens ønskede handling.

Alle bytes for hver blok sendes. Hvis der ikke findes nogen/flere tidsplaner fyldes de resterende bytes med 0x00.

#### *2.4.2.2 Send Tidsplaner*

Ved tilføjelse, fjernelse eller ændring af tidsplan bruges Send Tidsplan kommandoen. Kommandoen sender enhedens ID, enhedens rum ID og hvilken dag de sendte tidsplaner tilhører. Hvis der ingen tidsplaner er at sende sendes 0x00 som den 4. byte, hvorefter der ikke sendes mere.

Hvis der er tidsplaner at sende, bliver data bytes enhedens ID, enhedens rum ID, hvilken dag tidsplanerne hører til. Disse bliver efterfulgt af gentagende Tidsplans ID, time, minut, handling. Disse gentages indtil der ikke er flere tidsplaner at sende.

#### *2.4.2.3 Set nuværende tid*

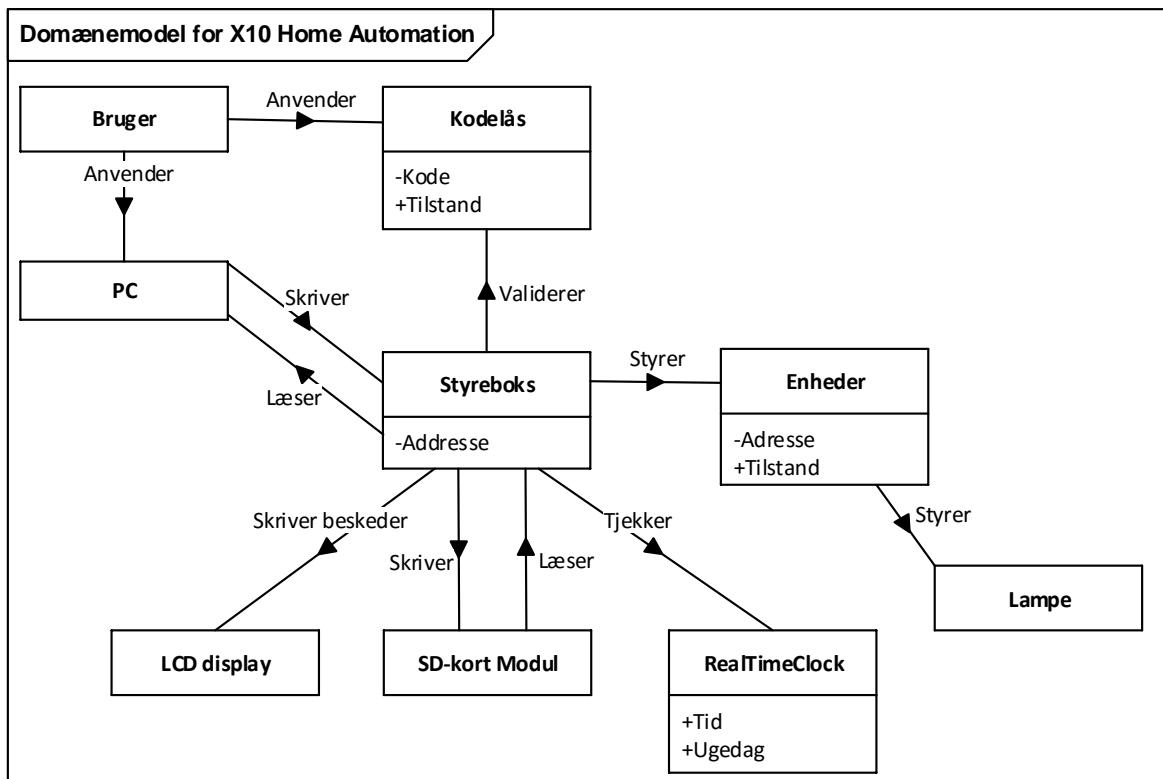
Ved set nuværende tid sendes den nuværende tid fra pc til styreboks for at synkronisere styreboksens ur med pc'ens, der sendes 7 data bytes i følgende rækkefølge:

Byte 7						Byte 1
Årstal (0-99)	Måned (0-12)	Dato	Ugedag (1-7)	Timetal (00-23)	Minuttal (0-59)	Sekundtal (0-59)

## 2.5 Domæneanalyse (Alle)

Domæneanalysen for X10 Home Automation bruges som indledende identifikation af domain og boundary klasser. Ud fra modellen vist på Figur 12 er følgende boundaries identificeret:

- Mellem PC og bruger
- Mellem PC og styreboks
- Mellem styreboks og enhed



Figur 12 Domæneanalyse for X10 Home Automation

### 3 Softwarearkitektur, Design og Implementering

I følgende afsnit beskrives softwaren for X10 Home Automation systemet. Som før beskrevet er det et distribueret system, og software arkitekturen er derfor delt op mellem de tre processorer: PC, Styreboks og Enhed. Det bemærkes at der ikke forefindes nogen unitest for softwaren – Da test af metoder foretages under accepttest.

#### 3.1 Blokbeskrivelse: Domænemodel (NT)

Herunder følger en kort beskrivelse af de enkelte blokke på domænemodellen, vist på Figur 12.

##### 3.1.1 PC

PC bliver anvendt af brugeren vha. En grafisk brugerflade. På PC findes software til at behandle brugerens input, samt at sende ændringer og tilføjelser af enheder/tidsplaner til styreboksen. PC'en læser fra styreboksen for at hente informationer vedr. Status for kodelåsen, status på tilføjede enheder eller for at hente informationer om gemte enheder.

PC'en skriver til styreboksen når der skal tilføjes eller fjernes nye enheder og tidsplaner.

##### 3.1.2 Styreboks

Styreboksen læser fra kodelåsen, om signalet indikerer at der er låst op. Dette sendes videre til PC'en når den anmoder om status på pinkoden. Der skrives til PC'en når der anmodes om information om gemte enheder. Når der fra PC anmodes om status på enheder, læser Styreboksen fra SD-kortet, og sender en amodning om status ud til de tilføjede enheder.

Styreboksen læser fra PC'en når der bliver sendt information vedr. Nye enheder, ændringer til eksisterende enheder, nye tidsplaner eller ændringer til eksisterende tidsplaner.

##### 3.1.3 Enhed

Enheden står for kontrol af tilsluttet elektrisk apparat, i vores tilfælde en lampe. Denne styres via et relæ. Enheden kan læse kommandoer der sendes fra styreboksen, og svare tilbage hvis der anmodes om den nuværende status for enheden.

### 3.2 Identifikation af control klasser (NT, AK)

Som led i at identificere hvilke control klasser der skal bruges for hver CPU skabes en matrice over systemets processorere og hvilke use cases de er relevante i. Dette vises i Tabel 1. hvori et X repræsenterer at CPU'en er inkluderet i den givne use case. For hver Use Case oprettes en control klasse.

USECASE/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	

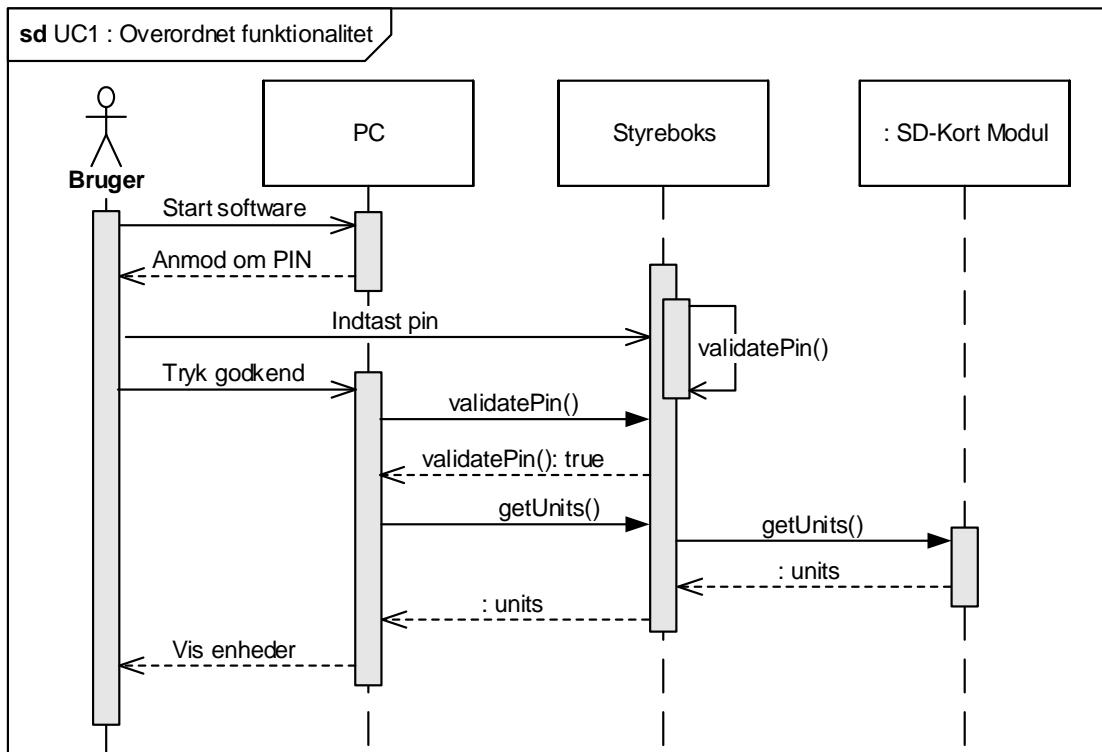
Tabel 1 Use Case / CPU matrix

### 3.3 Overordnet Funktionalitet : Sekvensdiagrammer (NT)

I følgende afsnit findes overordnede diagrammer for de forskellige use cases. Diagrammerne inkludere de tre CPU'er inkluderet i systemet. Disse benyttes til at skabe et overblik over hvordan de enkelte blokke skal fungere. Efterfølgende laves en uddybning af funktionaliteten for hver use case til hver CPU.

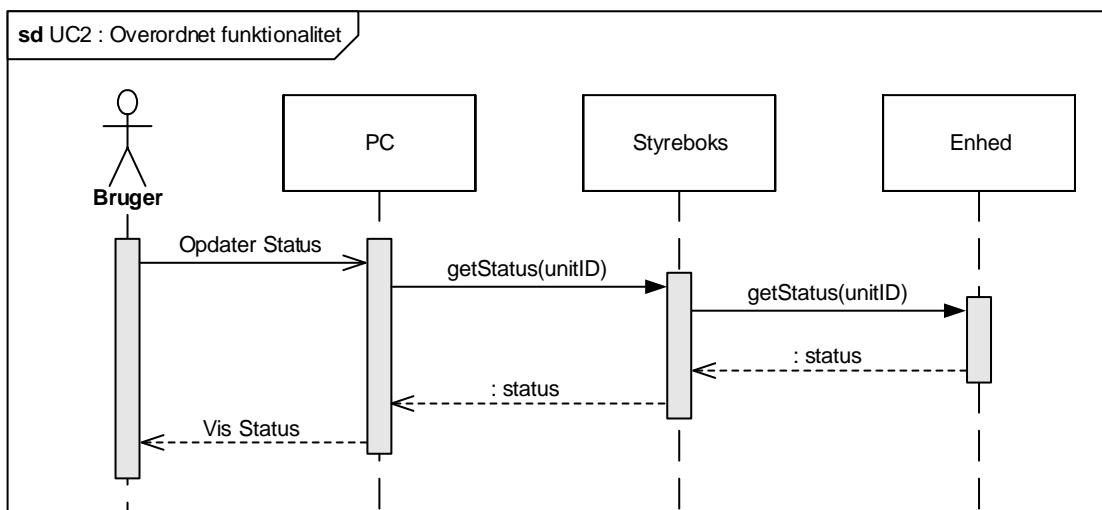
#### 3.3.1 Use Case 1

Diagrammet viser interaktionen mellem systemets blokke ved opstart af system.



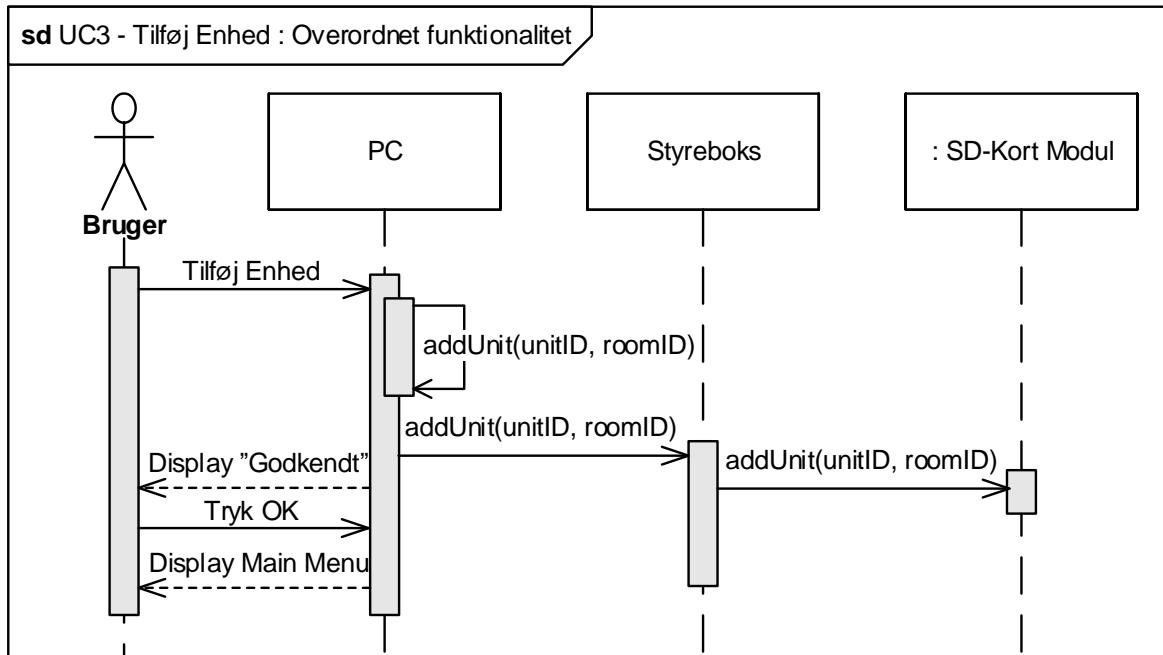
Figur 13 Overordnet UC Diagram for Use Case 1

#### 3.3.2 Use Case 2



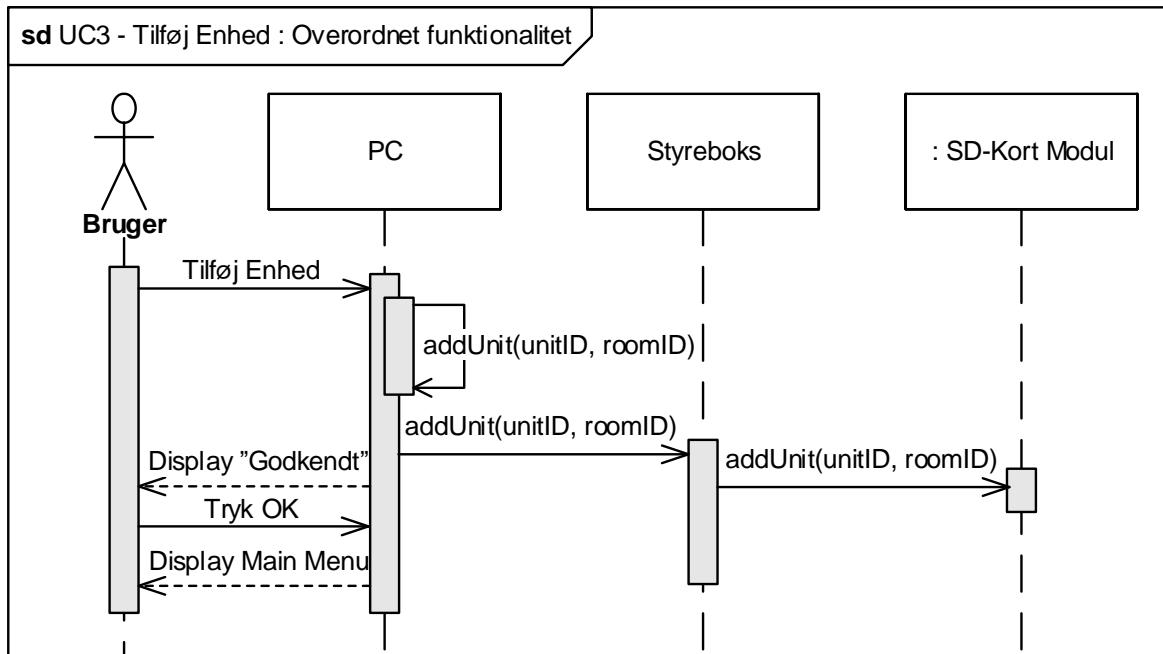
Figur 14 Overordnet UC Diagram for Use Case 2

### 3.3.3 Use Case 3

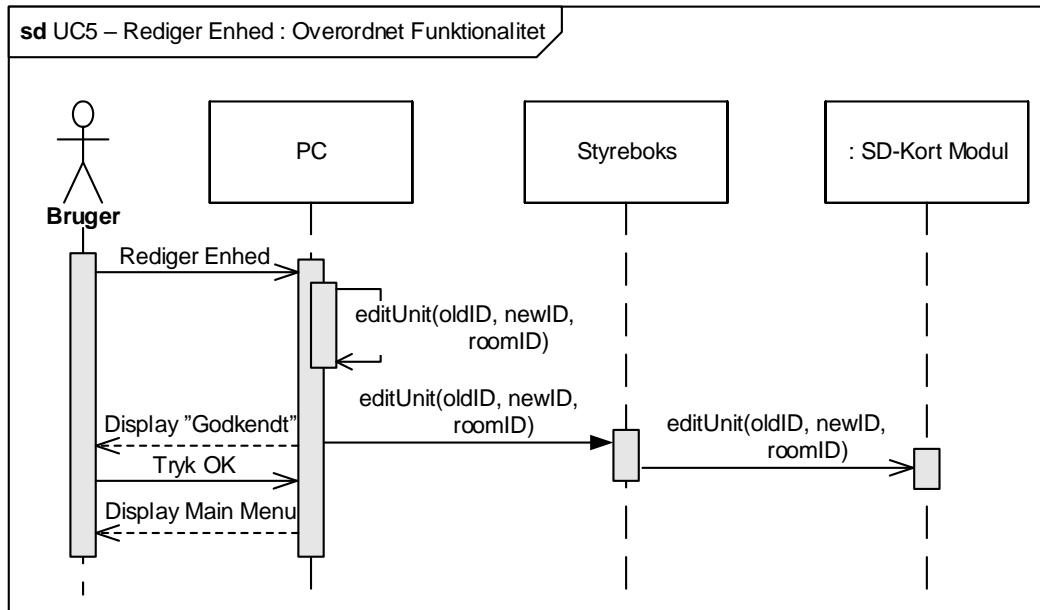


Figur 15 Overordnet UC Diagram for Use Case 3

### 3.3.4 Use Case 4

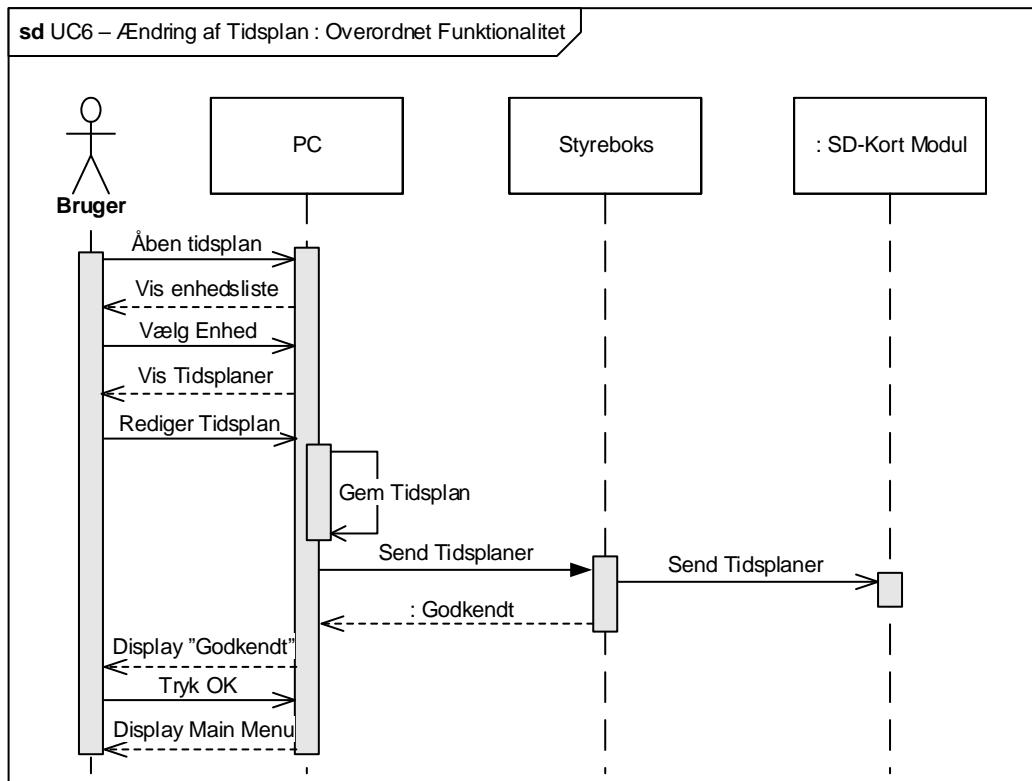


### 3.3.5 Use Case 5



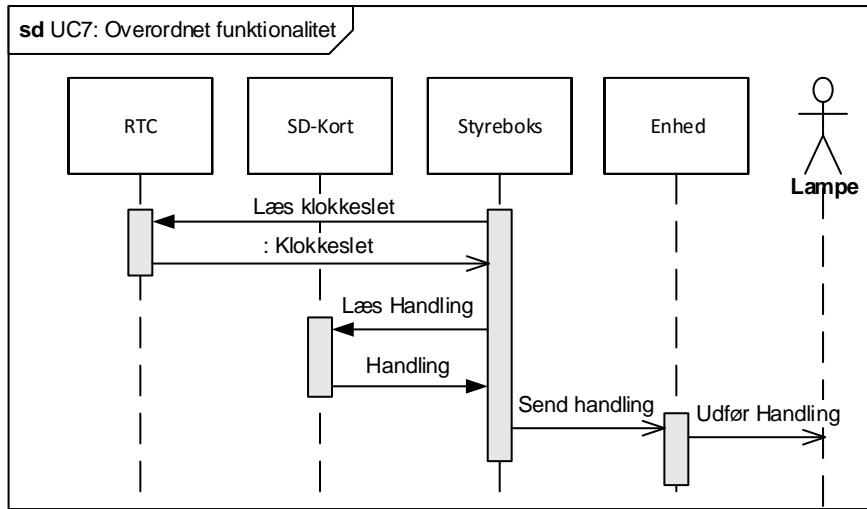
Figur 16 Overordnet UC Diagram for Use Case 5

### 3.3.6 Use Case 6



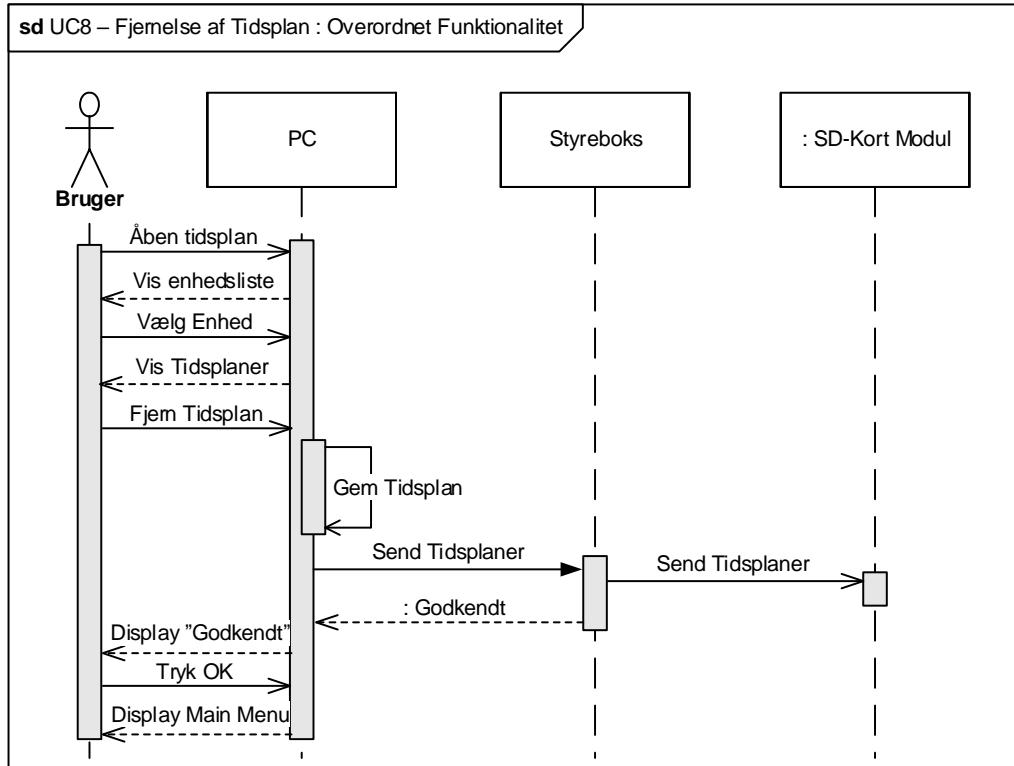
Figur 17 Overordnet UC Diagram for Use Case 5

### 3.3.7 Use Case 7



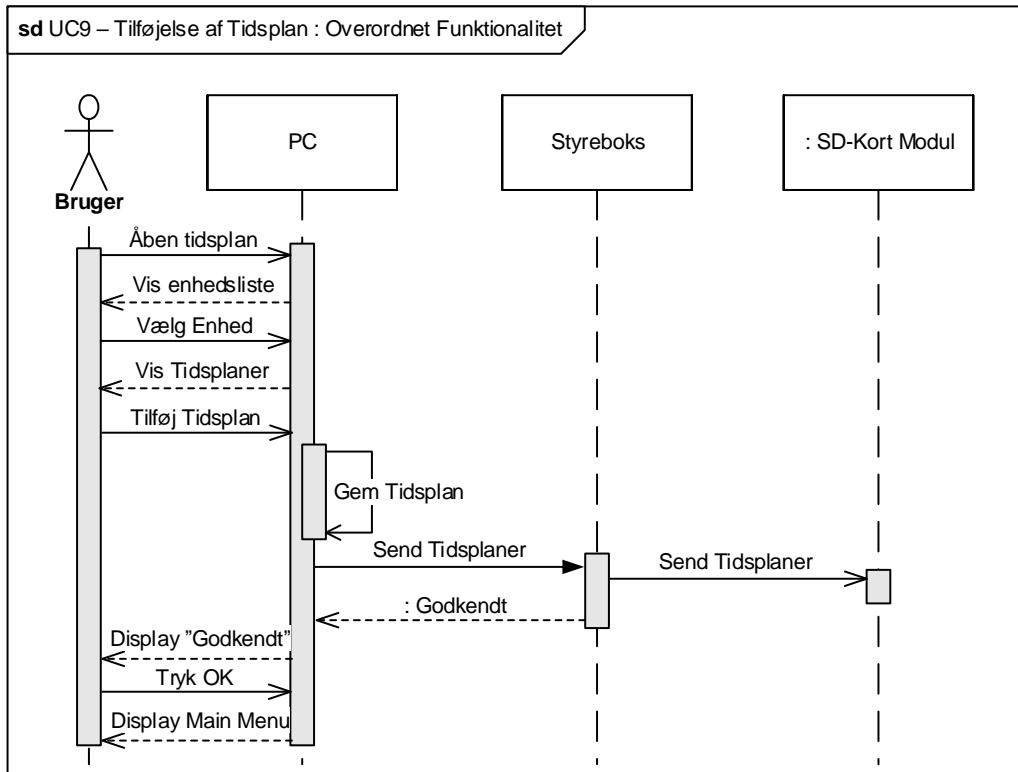
Figur 18 Overordnet UC diagram for Use Case 7

### 3.3.8 Use Case 8



Figur 19 Overordnet UC Diagram for Use Case 8

### 3.3.9 Use Case 9



Figur 20 Overordnet UC Diagram for Use Case 9

### 3.4 Identifikation af klasser (Alle)

Fra domænemodellen identificeres relevante klasser. Disse klasser bruges til at skabe applikationsmodeller for systemets tre CPU'er. Applikationsmodellerne vil dernæst blive beskrevet for hver CPU, henholdsvis PC, Styrebox og Enhed.

Der udfærdiges et klassediagram for hver CPU, hvori de relevante control, domain og boundary klasser inkluderes. Formålet med dette er at gøre det nemmere at designe software med høj samhørighed og lav kobling.

De identificerede klasser opdeles således:

#### Domain

- Kodelås – Benyttes af brugeren til at låse systemet op. Læses fra styreboksen.
- SD-Kort Modul – Bruges til lagring af information vedr. Enheder og tilhørende tidsplaner.
- LCD-Display – Skrives til fra styreboksen. Bruges til at vise kritisk information.
- Real-time Clock – Læses fra styreboksen. Bruges til at kontrollere klokkeslet, hvor de enkelte enheder skal udføre en handling.

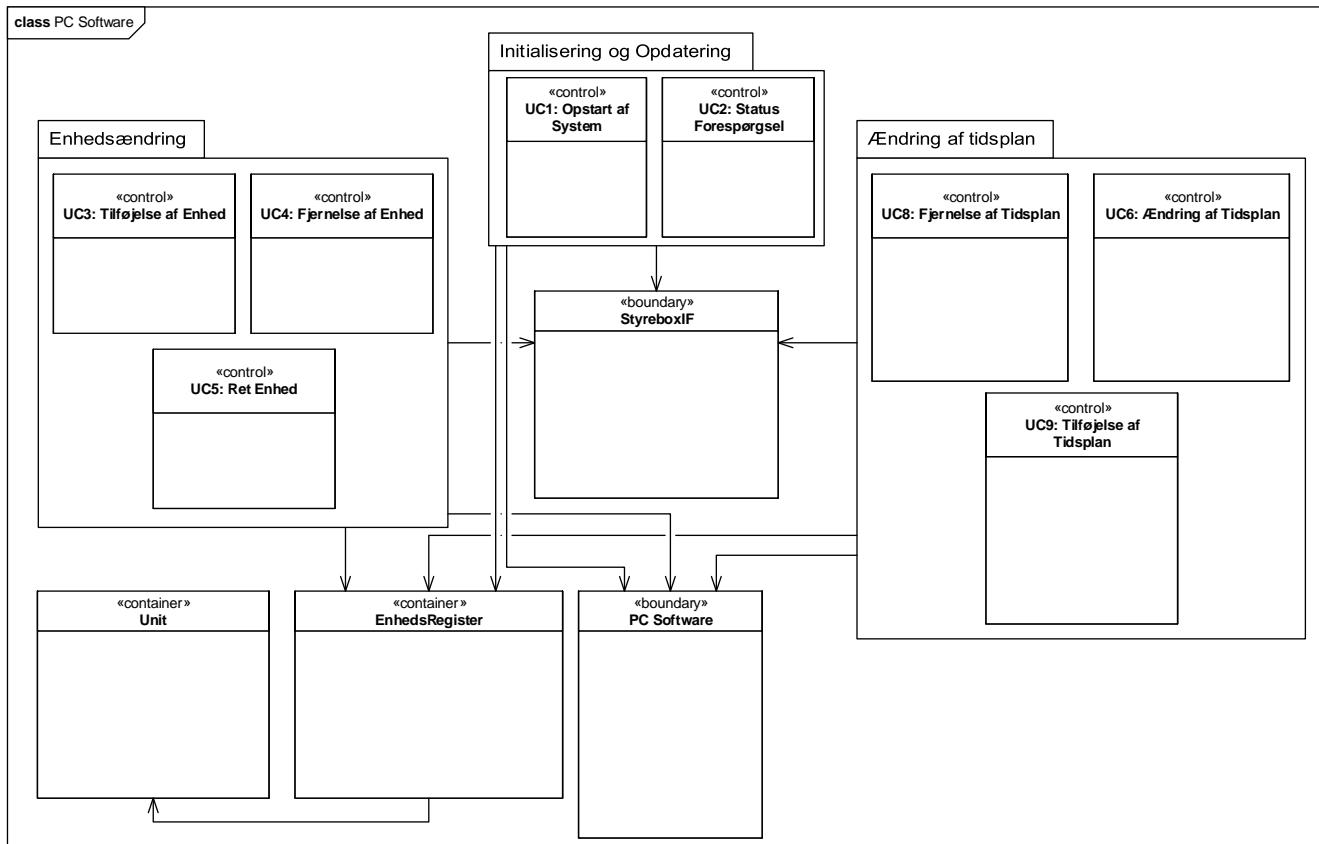
#### Boundary

- PC Software – Interface fra Bruger til PC
- StyreboxIF – Interface fra PC til Styrebox
- PCIF – Interface fra Styrebox til PC
- EnhedsIF – Interface fra Styrebox til Enhed

### 3.5 Applikationsmodel – PC (NT, AK)

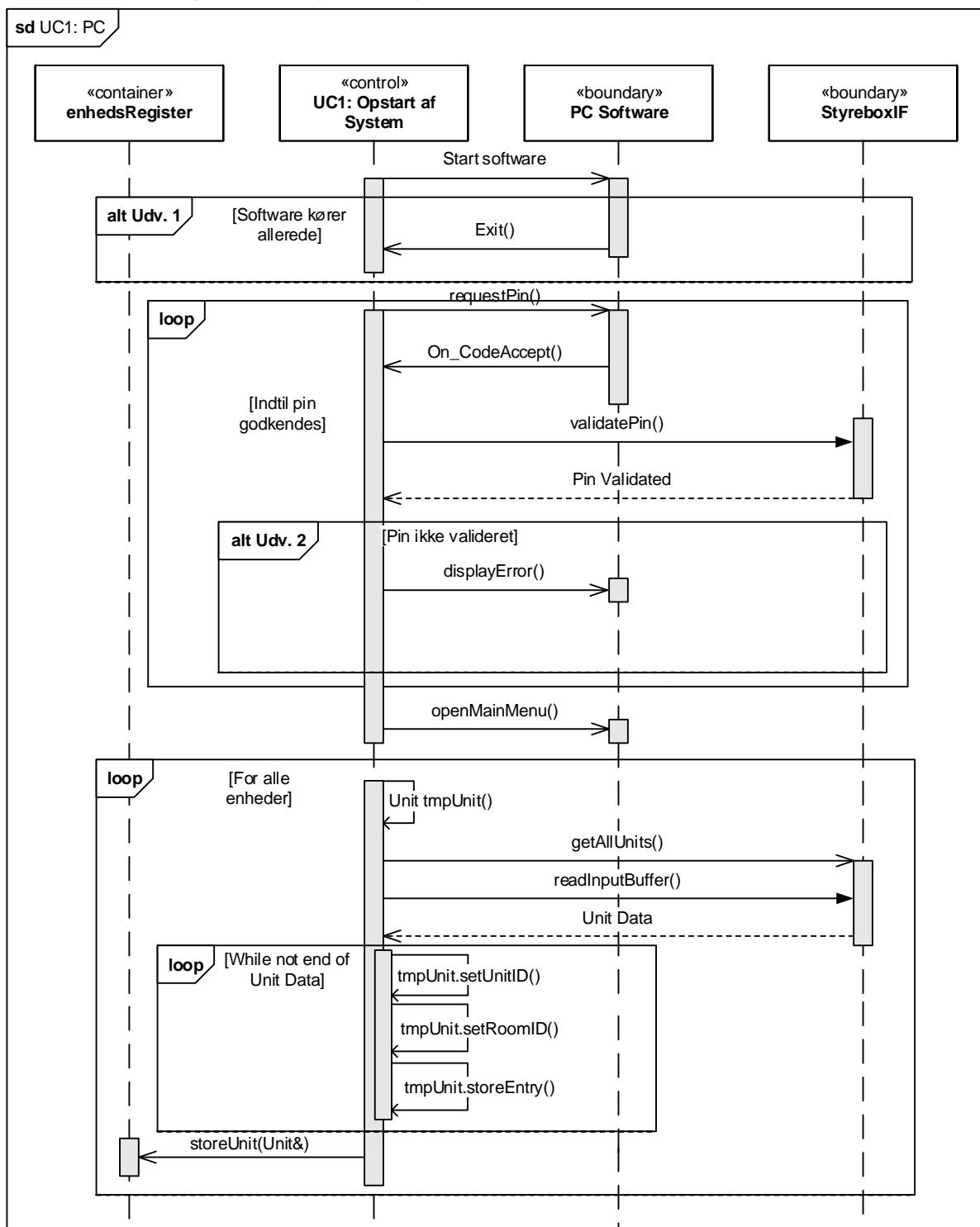
I dette afsnit findes applikationsmodellen for PC'en. Applikationsmodellen består af et indledende klassediagram, med tilhørende sekvensdiagrammer. Sekvensdiagrammerne benyttes til at identificere metoder. Disse metoder skrives ind i klassediagrammet, og er en del af implementationen af softwaren.

På Figur 21 ses det tomme klassediagram for PC. De forskellige klasser er indelt i forhold til ansvarsområde, for at skabe samhørighed mellem kasserne og deres funktionalitet.



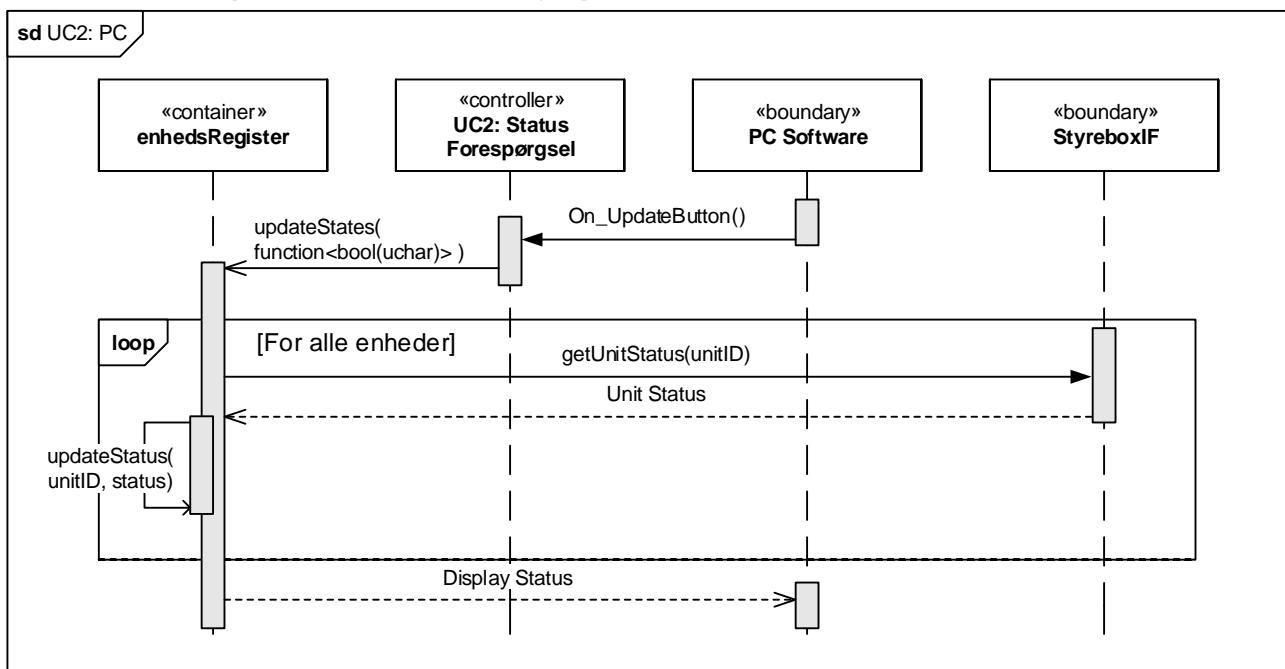
Figur 21 Indledende klassediagram for PC Software

### 3.5.1 Sekvensdiagram UC 1: Opstart af System



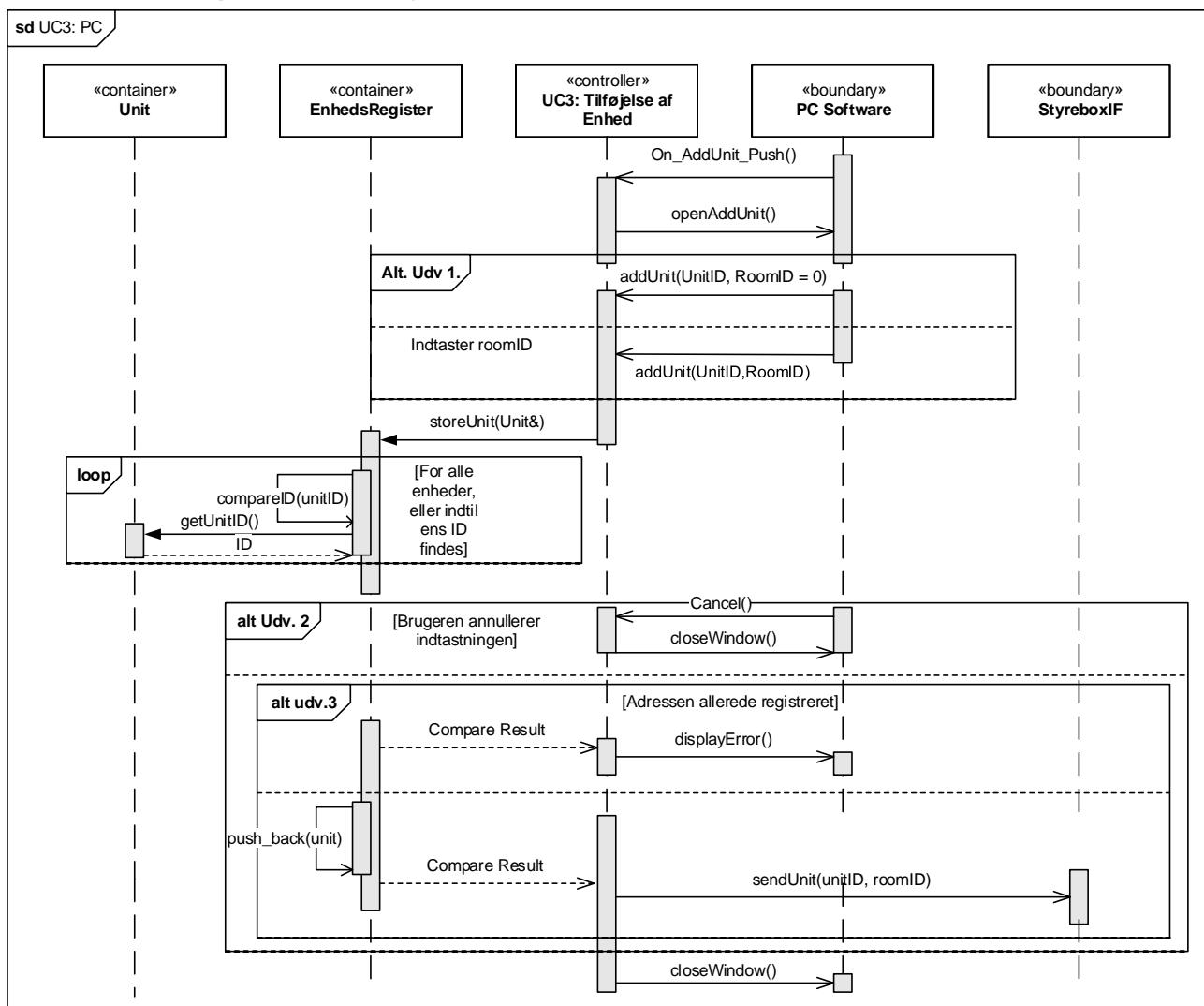
Figur 22 Sekvensdiagram for UC1 : PC

### 3.5.2 Sekvensdiagram for UC 2 Status Forespørgsel

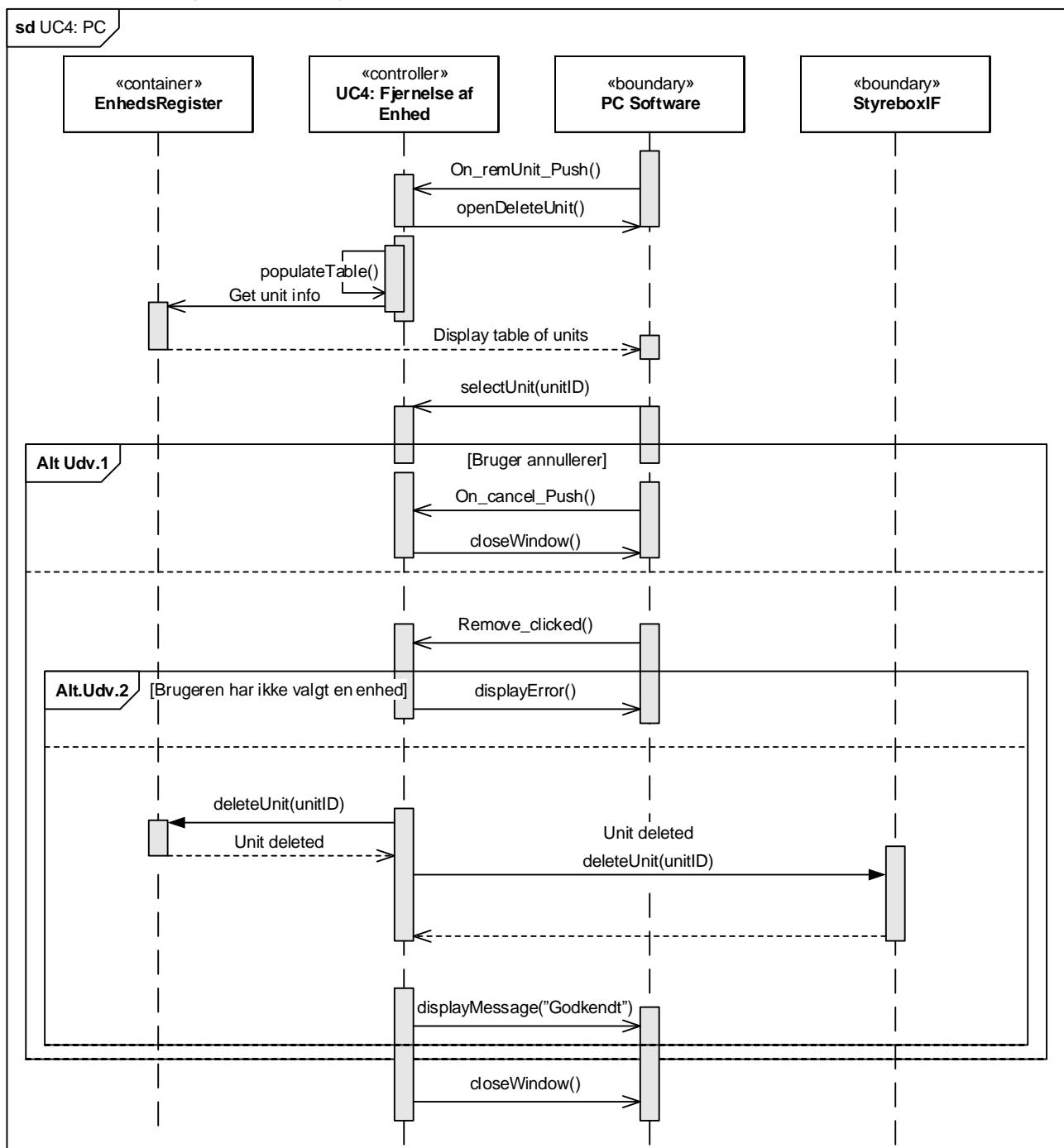


Figur 23 Sekvensdiagram for UC 2 : PC

### 3.5.3 Sekvensdiagram for UC3 Tilføjelse af Enhed

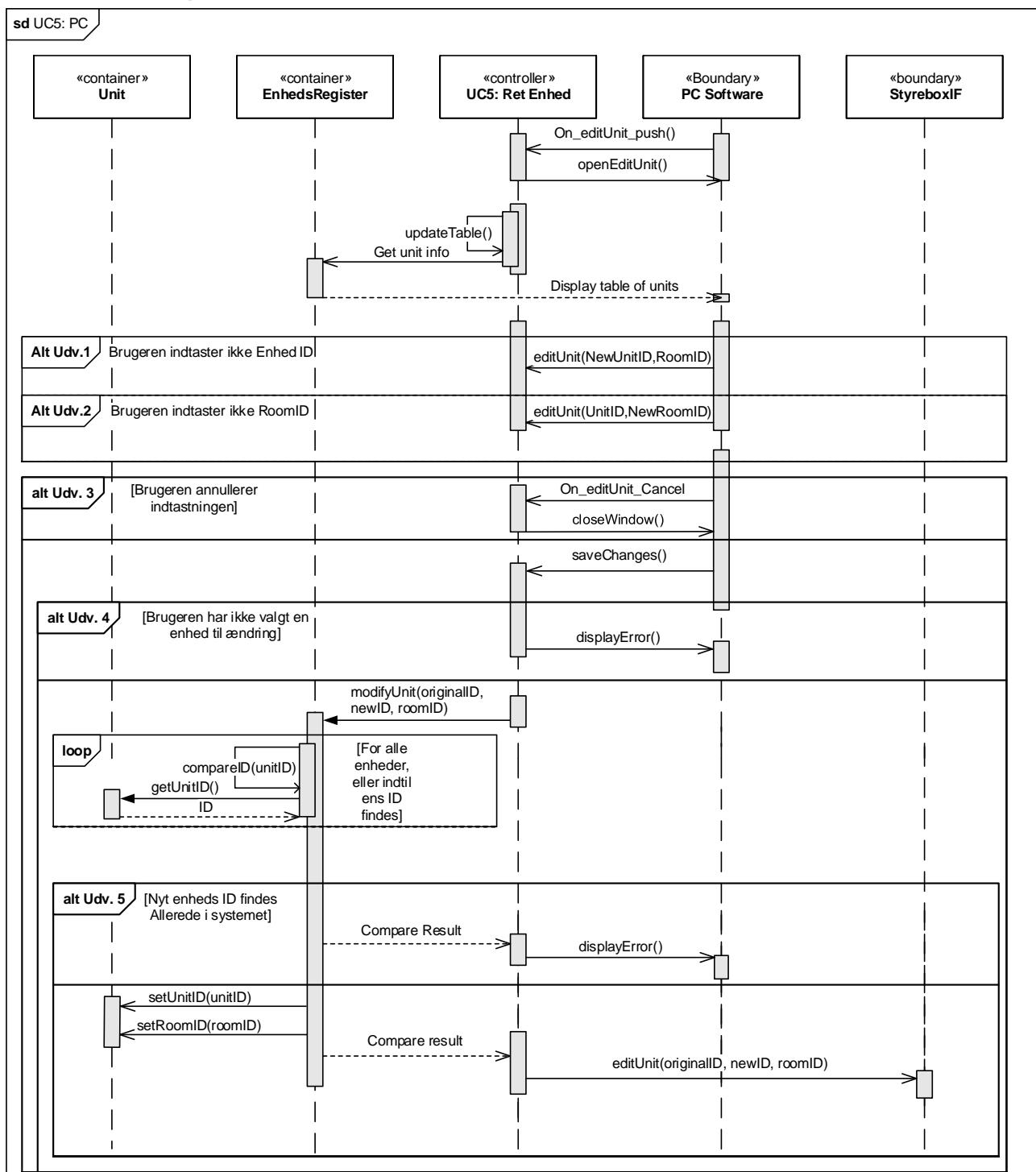


### 3.5.4 Sekvensdiagram for UC 4 Fjernelse af Enhed



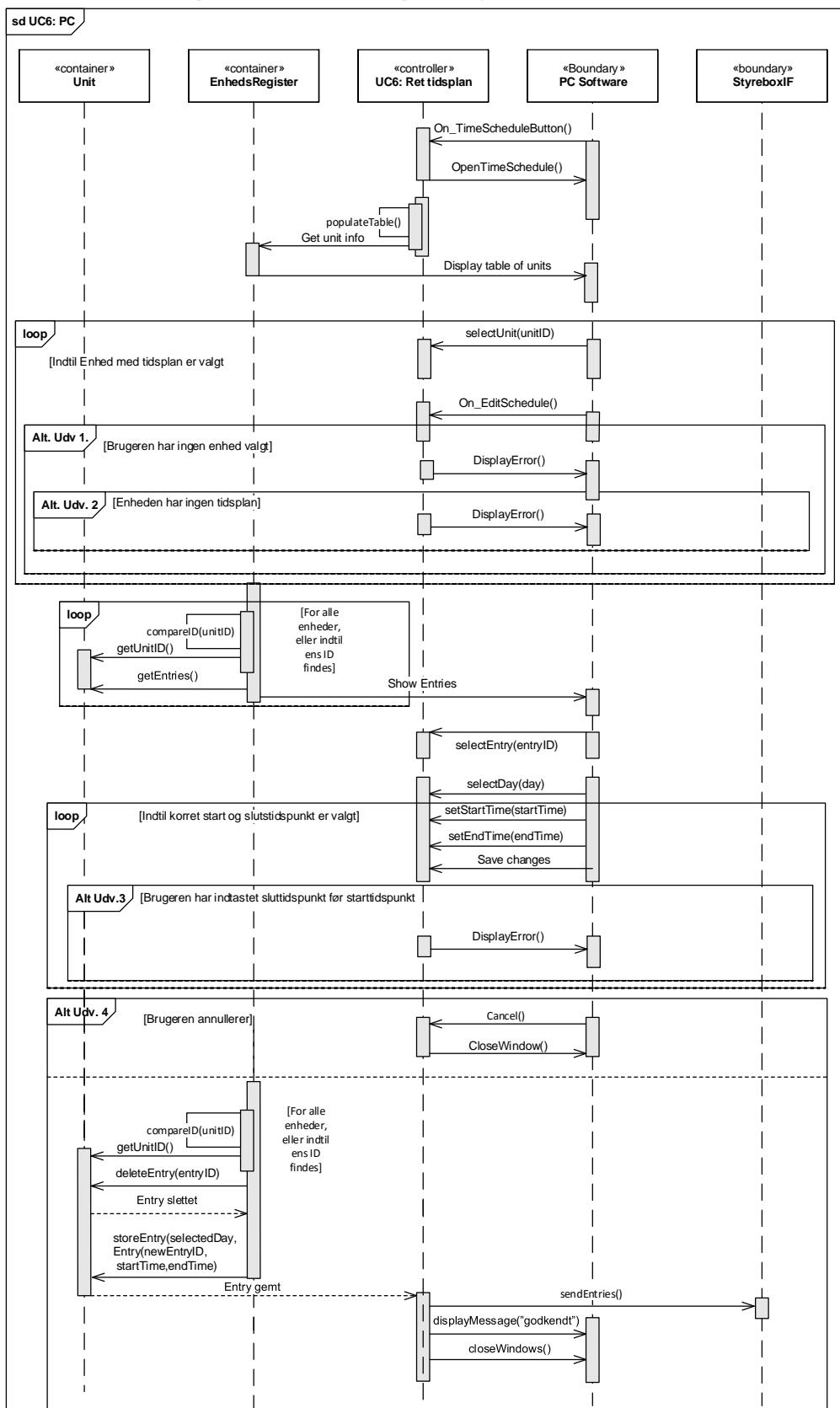
Figur 24 Sekvensdiagram for UC4 : PC

### 3.5.5 Sekvensdiagram for UC5 Ret Enhed



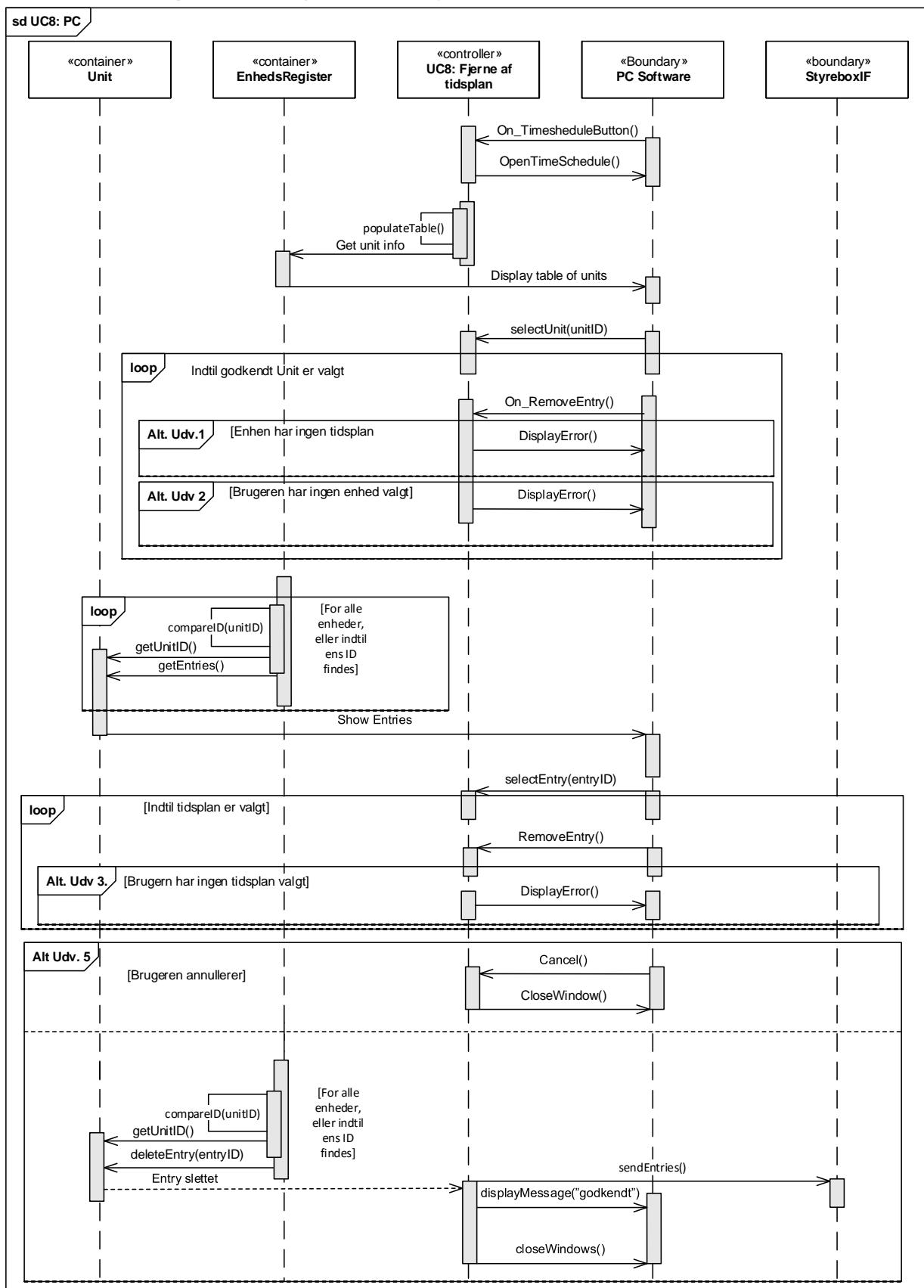
Figur 25 Sekvensdiagram for UC5 : PC

### 3.5.6 Sekvensdiagram for UC6 Ændring af Tidsplan



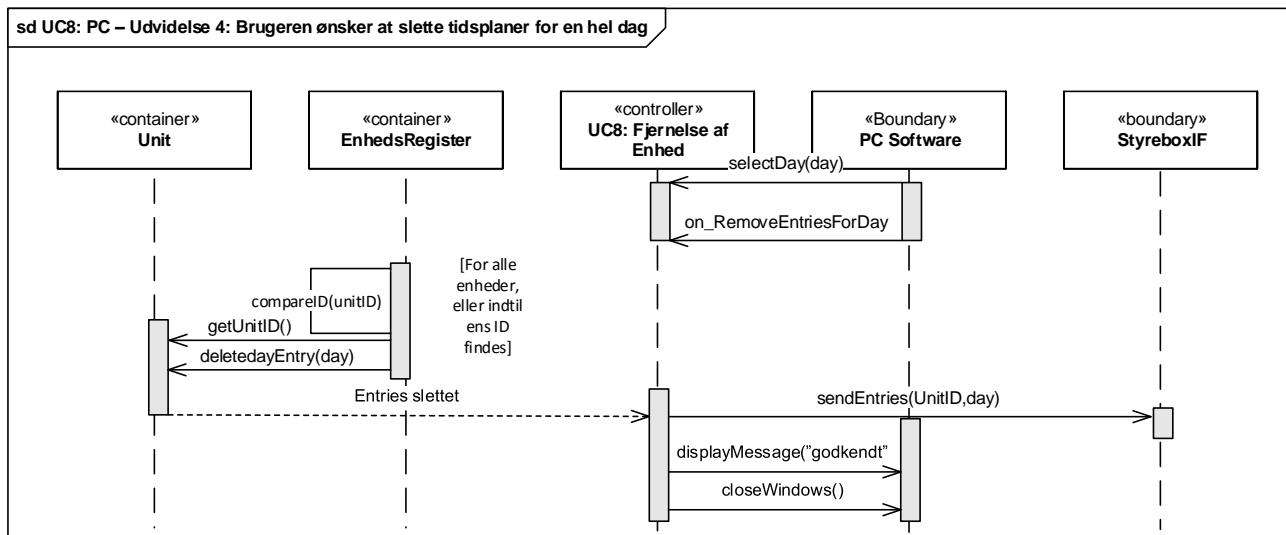
Figur 26 Sekvensdiagram for UC6 Ændring af tidsplan

### 3.5.7 Sekvensdiagram for UC8 Fjernelse af Tidsplan



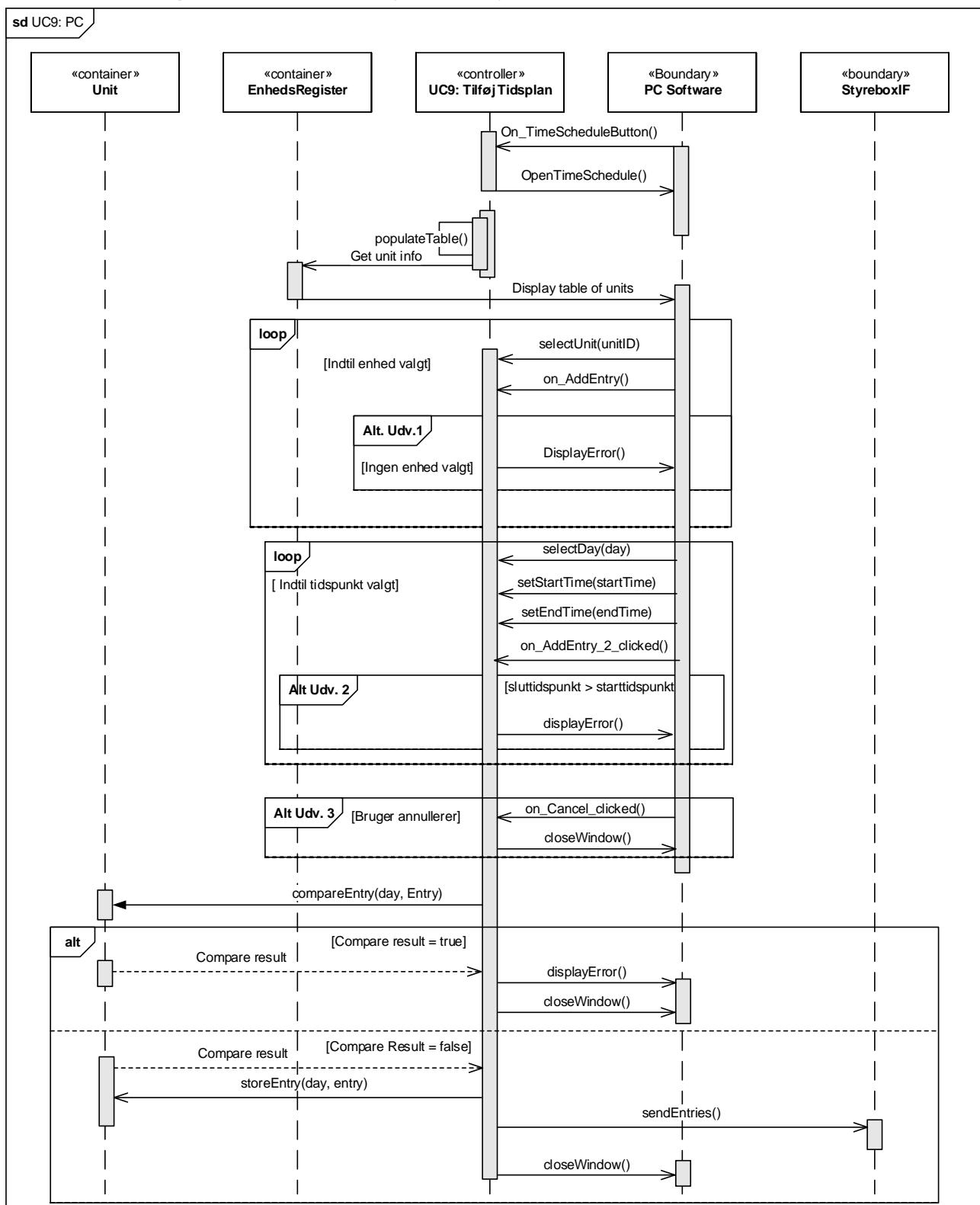
Figur 27 Sekvensdiagram for UC8 : PC

### 3.5.7.1 Udvilelse 4 for Use Case 8



Figur 28 Sekvensdiagram for Use Case 8 : PC - Udvilelse 4

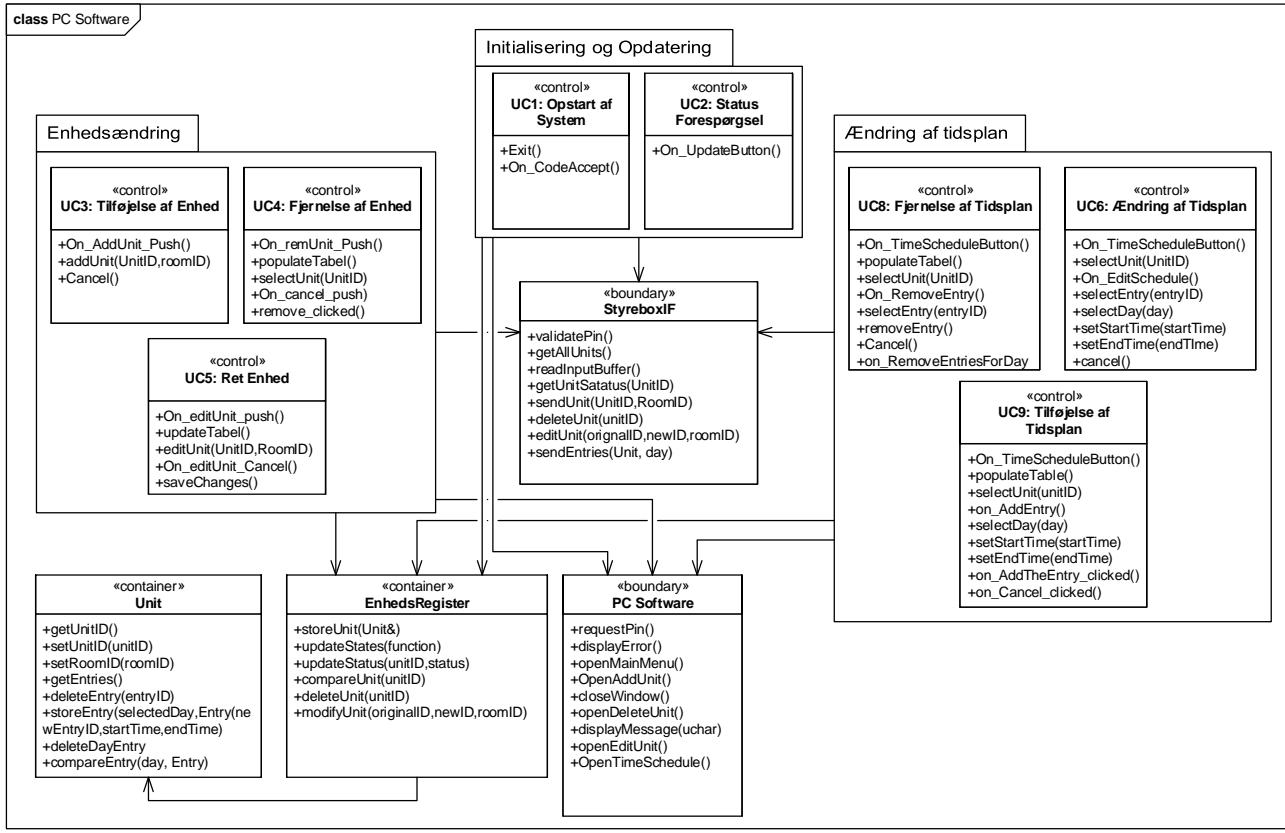
### 3.5.8 Sekvensdiagram for Use Case 9 Tilføjelse af Tidsplan



Figur 29 Sekvensdiagram for UC9 : PC

### 3.5.9 Metodeidentifikation for PC

Ud fra klassediagrammerne identificeres metodekald mellem de individuelle blokke, som derefter indføres på PC'ens klassediagram. Dette ses på Figur 30 hvor metoderne fra sekvensdiagrammet er indført.



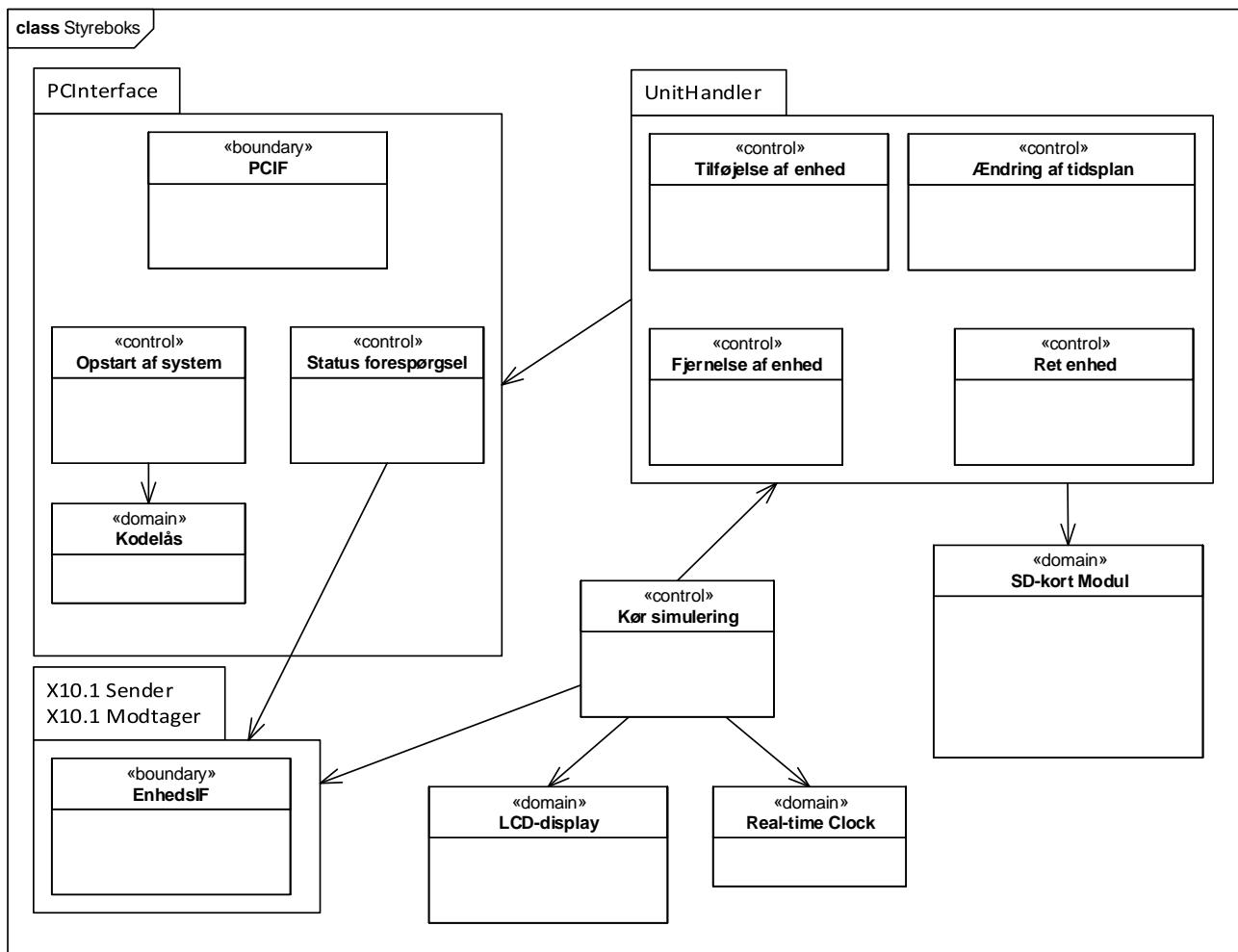
### Figur 30 Udfyldt klassediagram for PC

### 3.6 Applikationsmodel – Styreboks (TF, SN, DP)

I dette afsnit findes applikationsmodellen for Styreboksen. Applikationsmodellen består af et indledende klassediagram, med tilhørende sekvensdiagrammer. Sekvensdiagrammerne benyttes til at identificere metoder. Disse metoder skrives ind i klassediagrammet, og er en del af implementationen af softwaren.

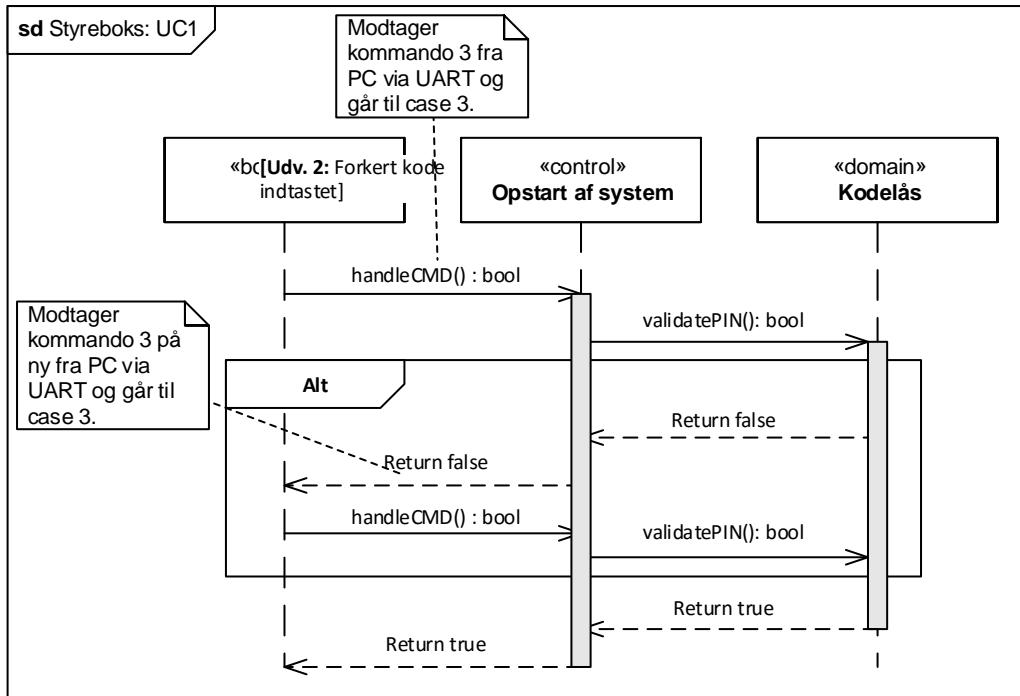
På Figur 31 ses det tomme klassediagram for Styreboksen. De forskellige klasser er indelt i forhold til ansvarsområde, for at skabe samhørighed mellem kasserne og deres funktionalitet.

Der forefindes ingen control klasser for Use Case 8 og Use Case 9. Dette skyldes at fra styreboksens side vil funktionaliteten være ens med den der ses i Use Case 6.



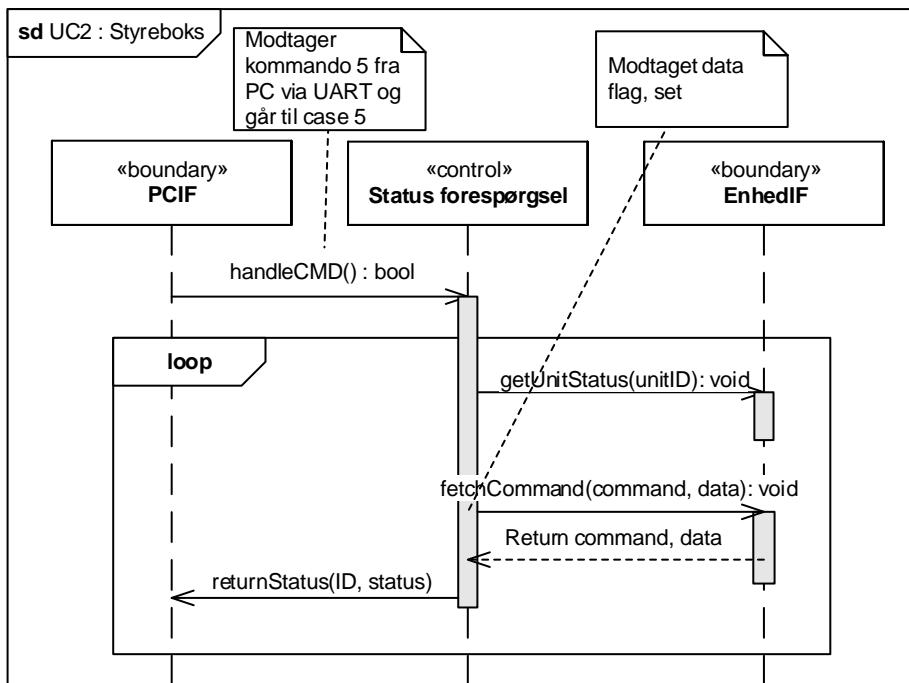
Figur 31 Indledende klassediagram for Styreboks

### 3.6.1 Sekvensdiagram for UC1 Opstart af System



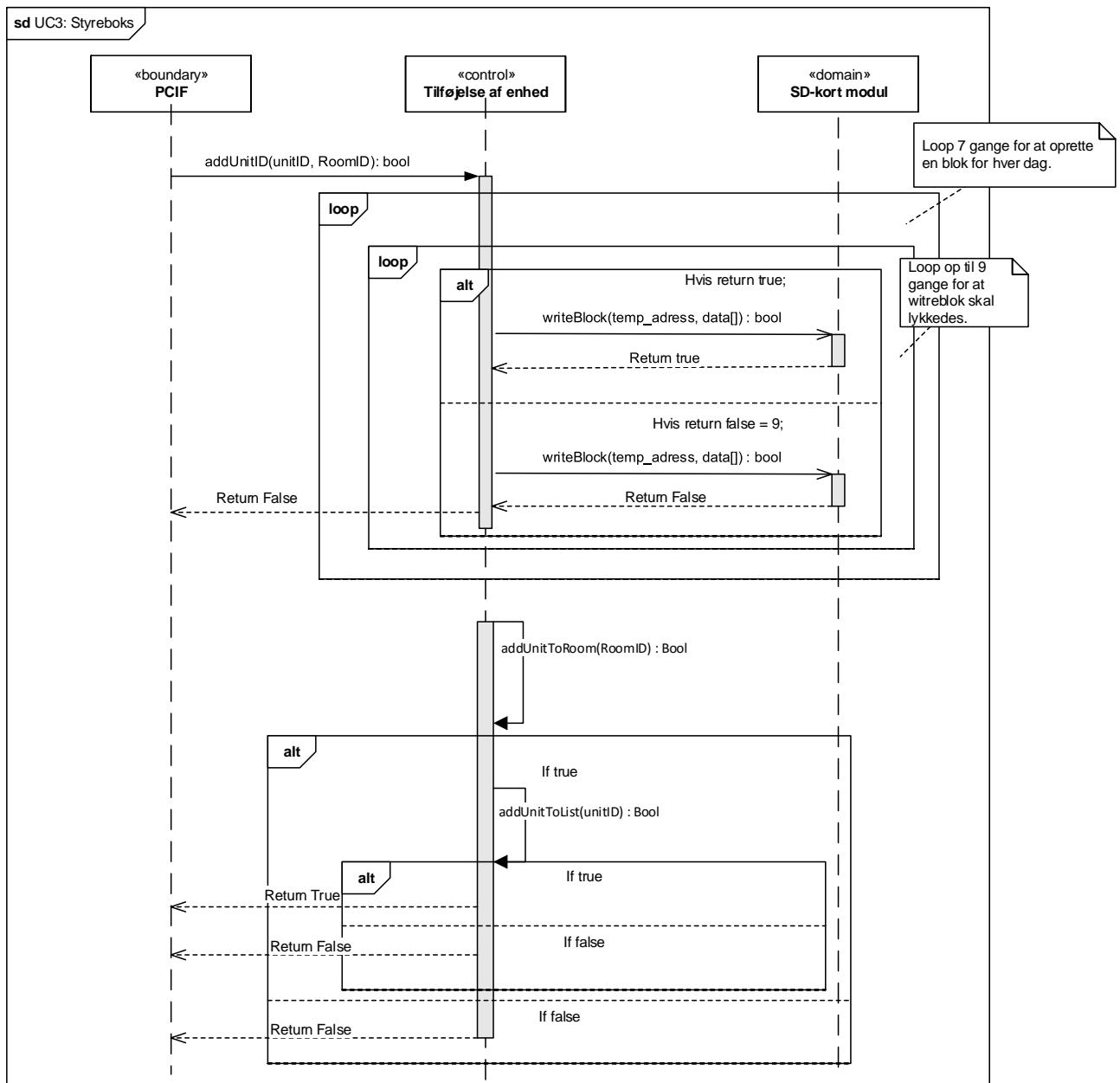
**Figur 32** Sekvensdiagram for UC 1 : Styreboks

### 3.6.2 Sekvensdiagram for UC2 Status Forespørgsel



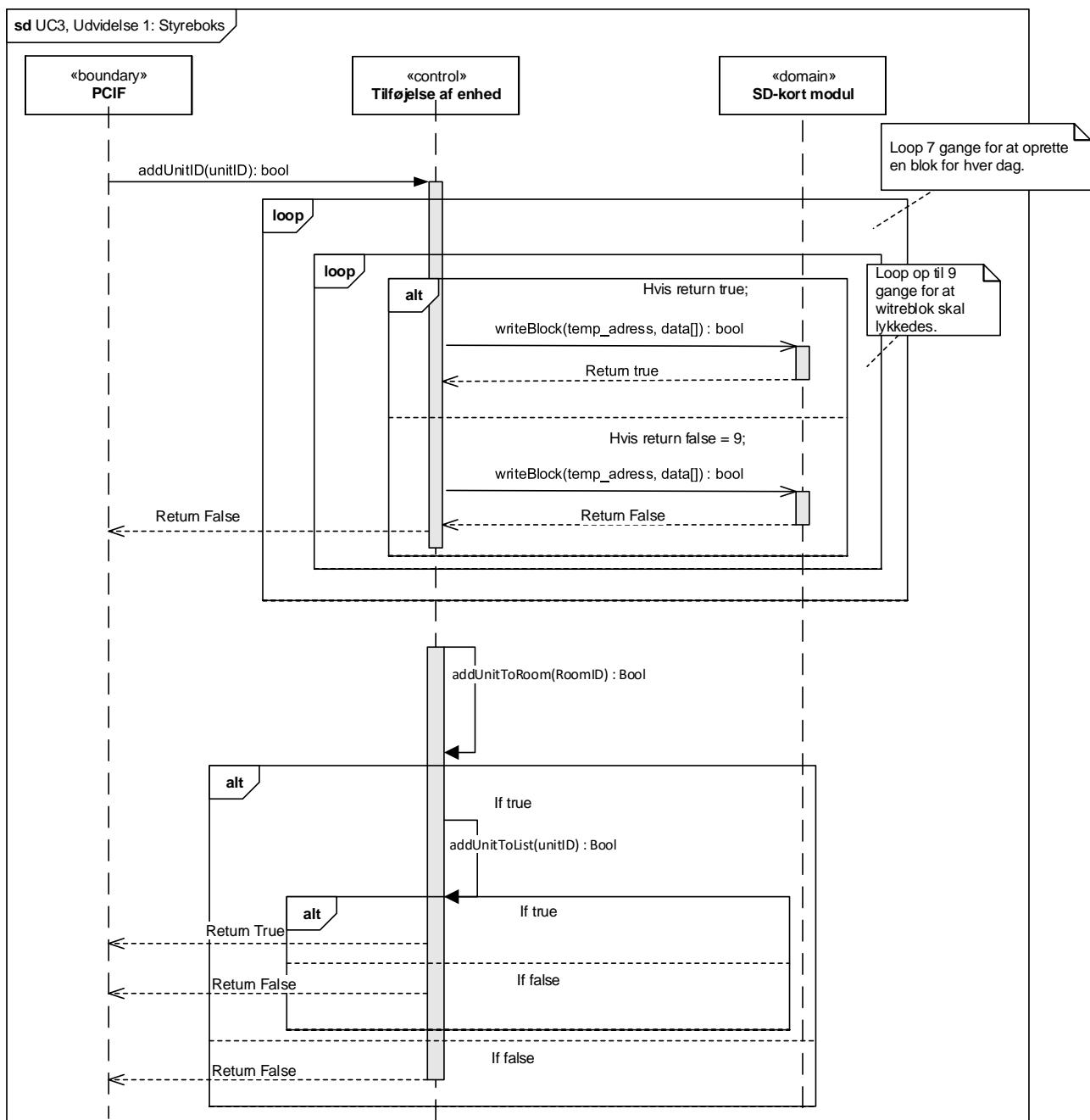
### Figur 33 Sekvensdiagram for UC 2 : Styreboks

### 3.6.3 Sekvensdiagram for Use Case 3 Tilføjelse af Enhed



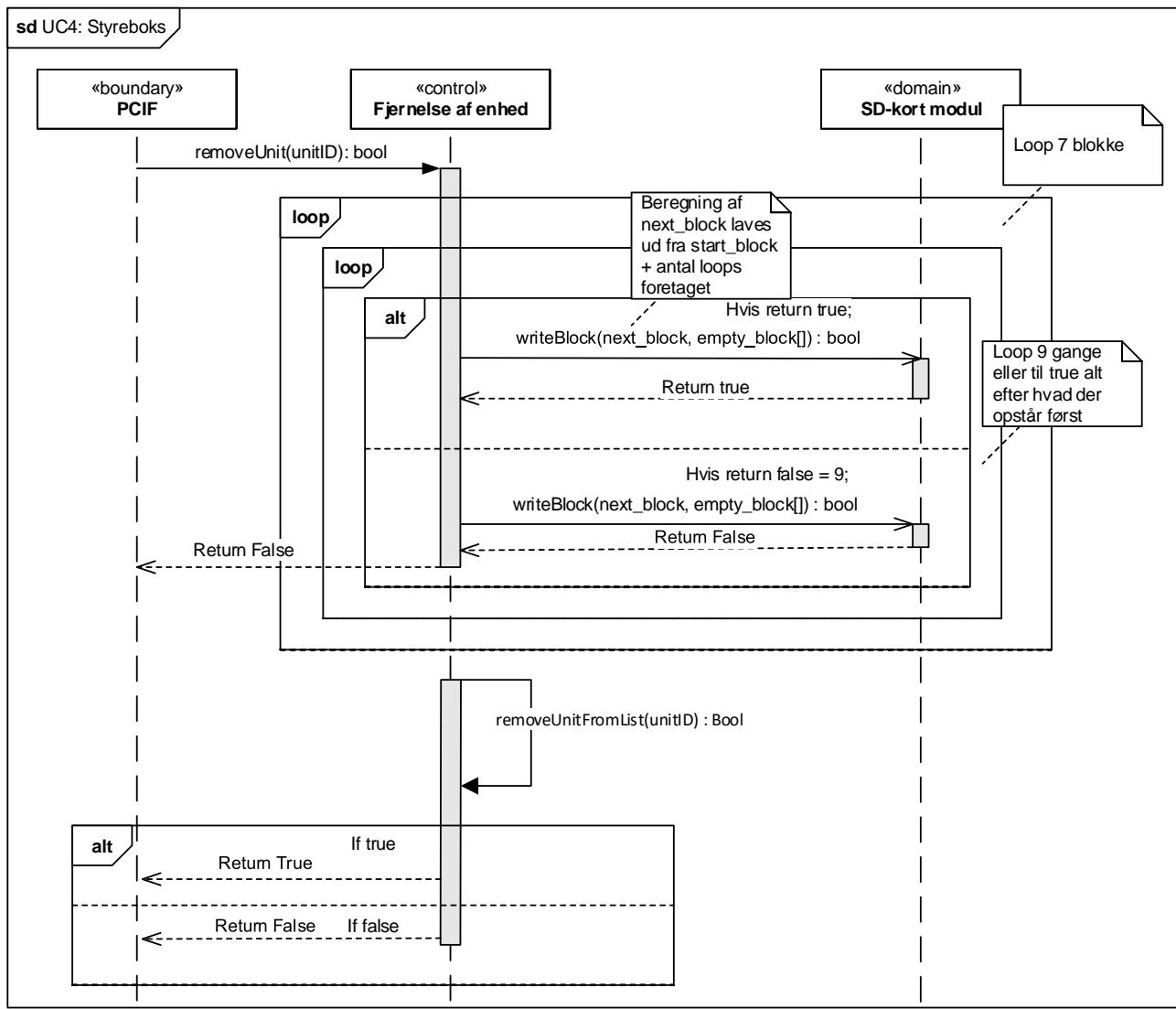
Figur 34 Sekvensdiagram for UC3 : Styreboks

### 3.6.3.1 Sekvensdiagram for Use Case 3, Udvigelse 1 : Bruger tildeler ikke et rum



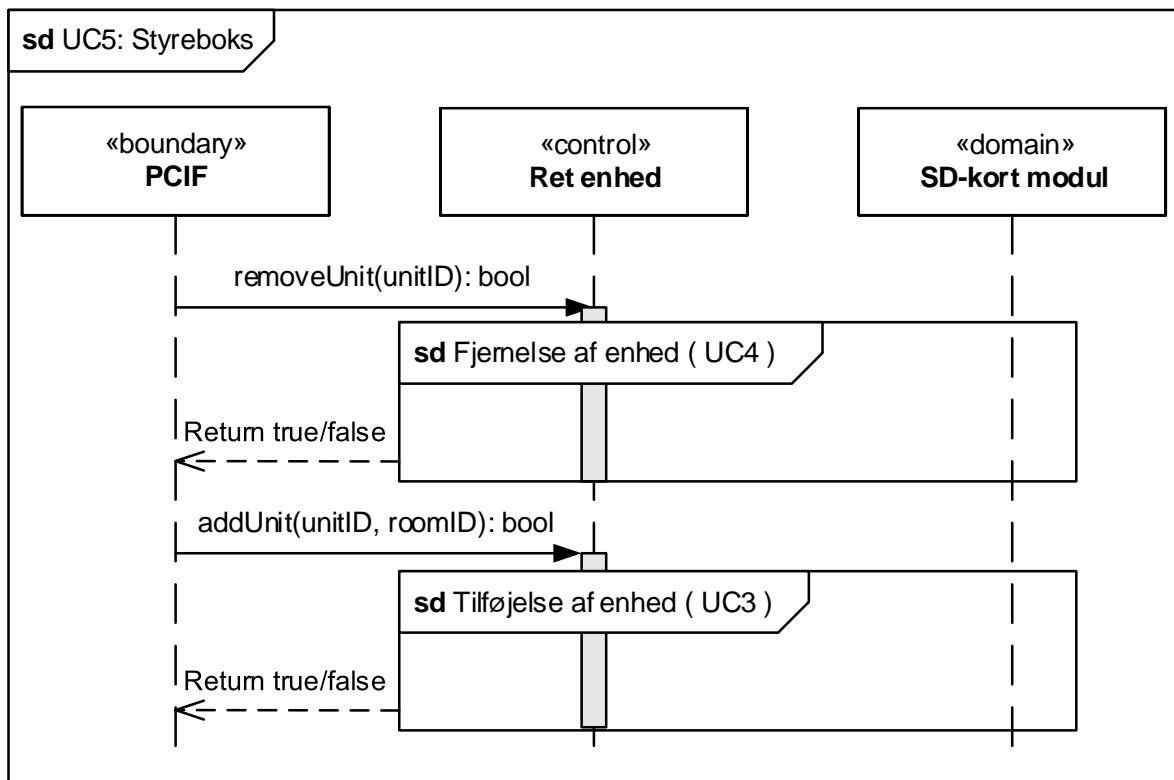
Figur 35 Sekvensdiagram for UC3 Udv. 1 : Styreboks

### 3.6.4 Sekvensdiagram for Use Case 4 Fjernelse af Enhed



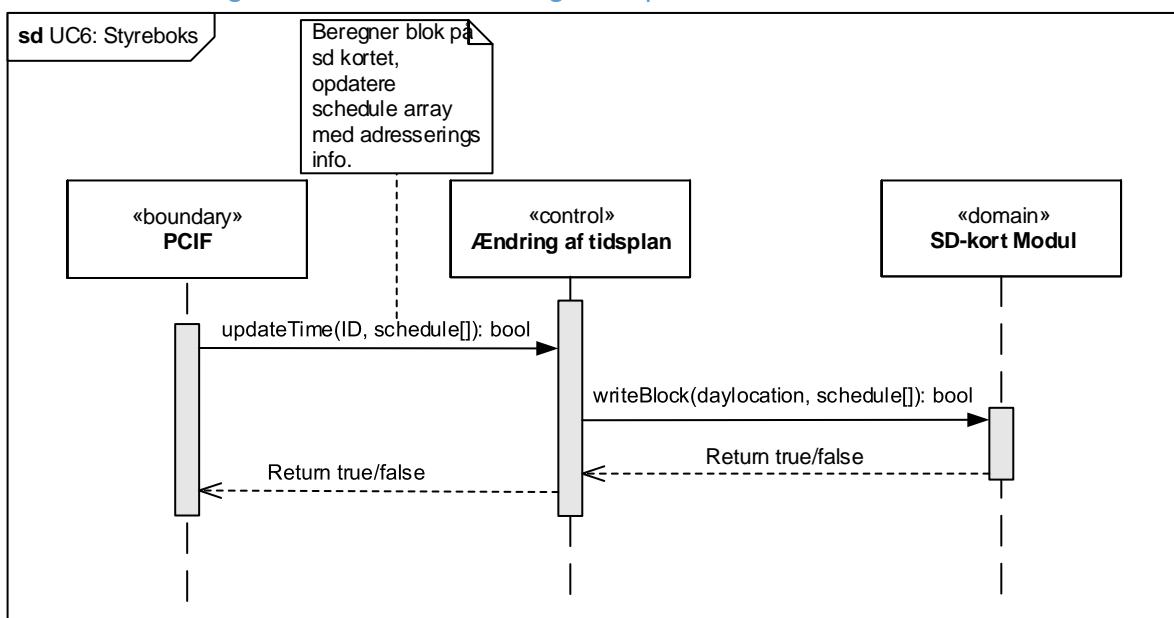
Figur 36 Sekvensdiagram for UC 4 : Styrebox

### 3.6.5 Sekvensdiagram for Use Case 5 Rediger Enhed



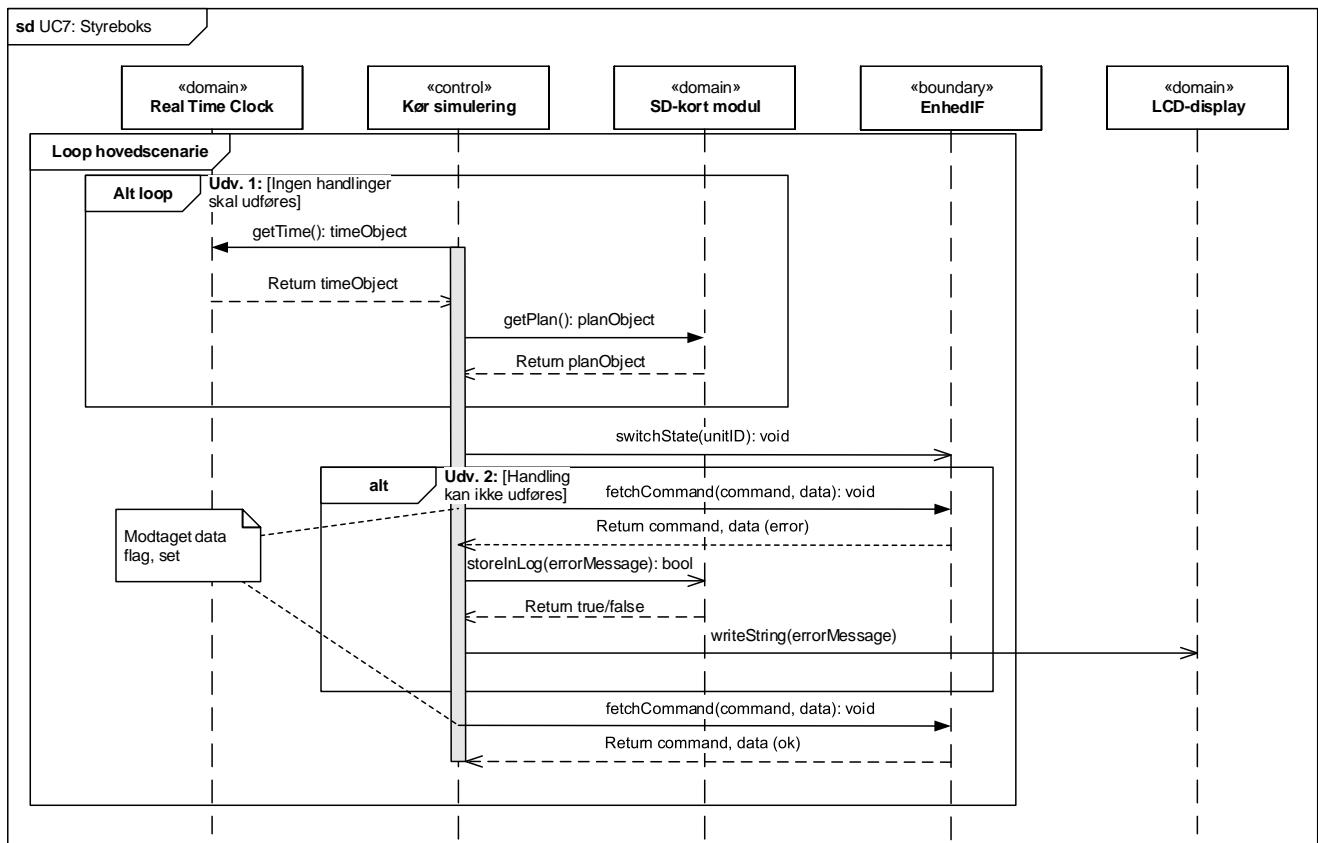
**Figur 37 Sekvensdiagram for UC5 : Styrebokx**

### 3.6.6 Sekvensdiagram for Use Case 6 Ændring af Tidsplan



### **Figur 38 Sekvensdiagram for UC6 : Styrebox**

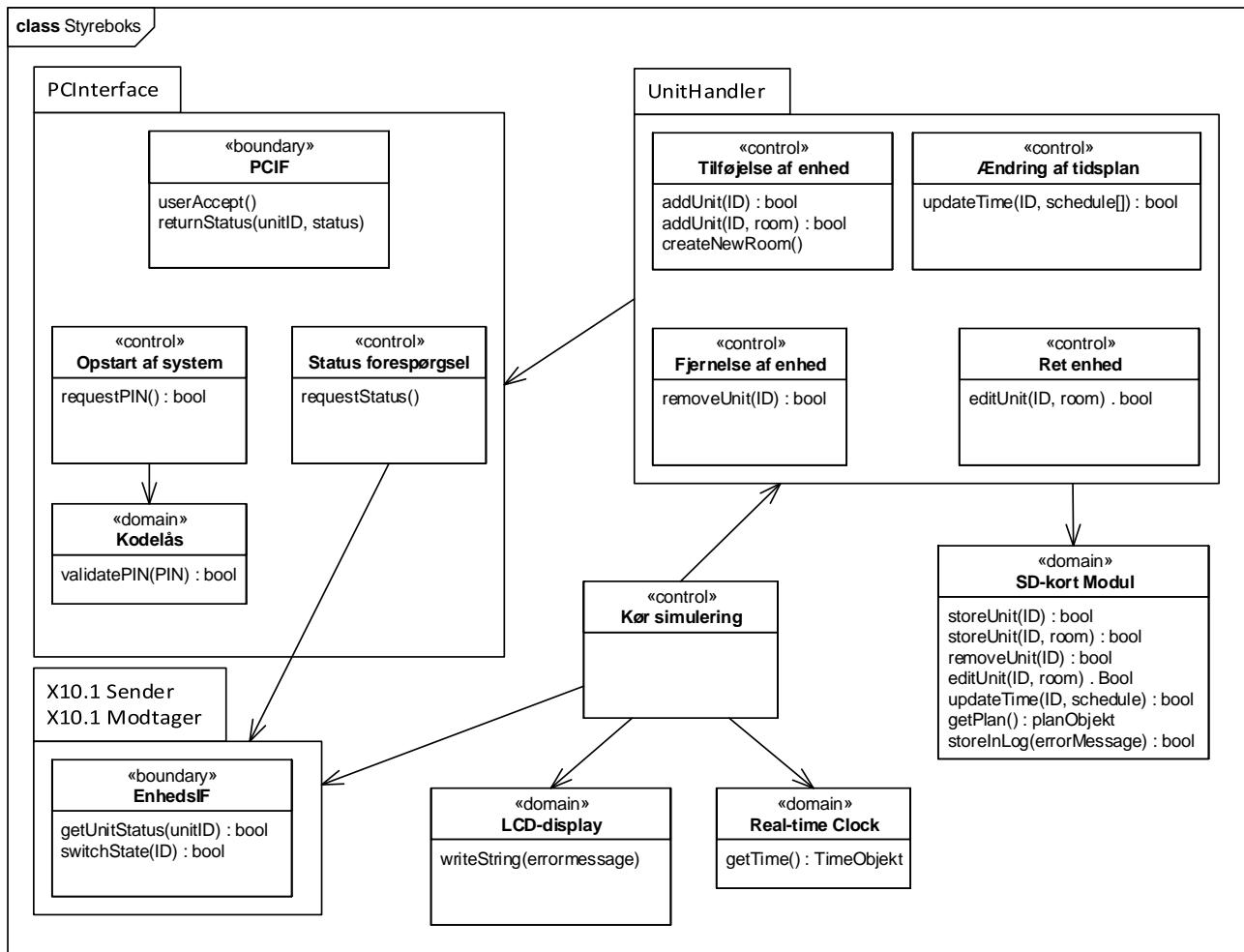
### 3.6.7 Sekvensdiagram for Use Case 7 Kør Simulering



Figur 39 Sekvensdiagram for UC7 : Styreboks

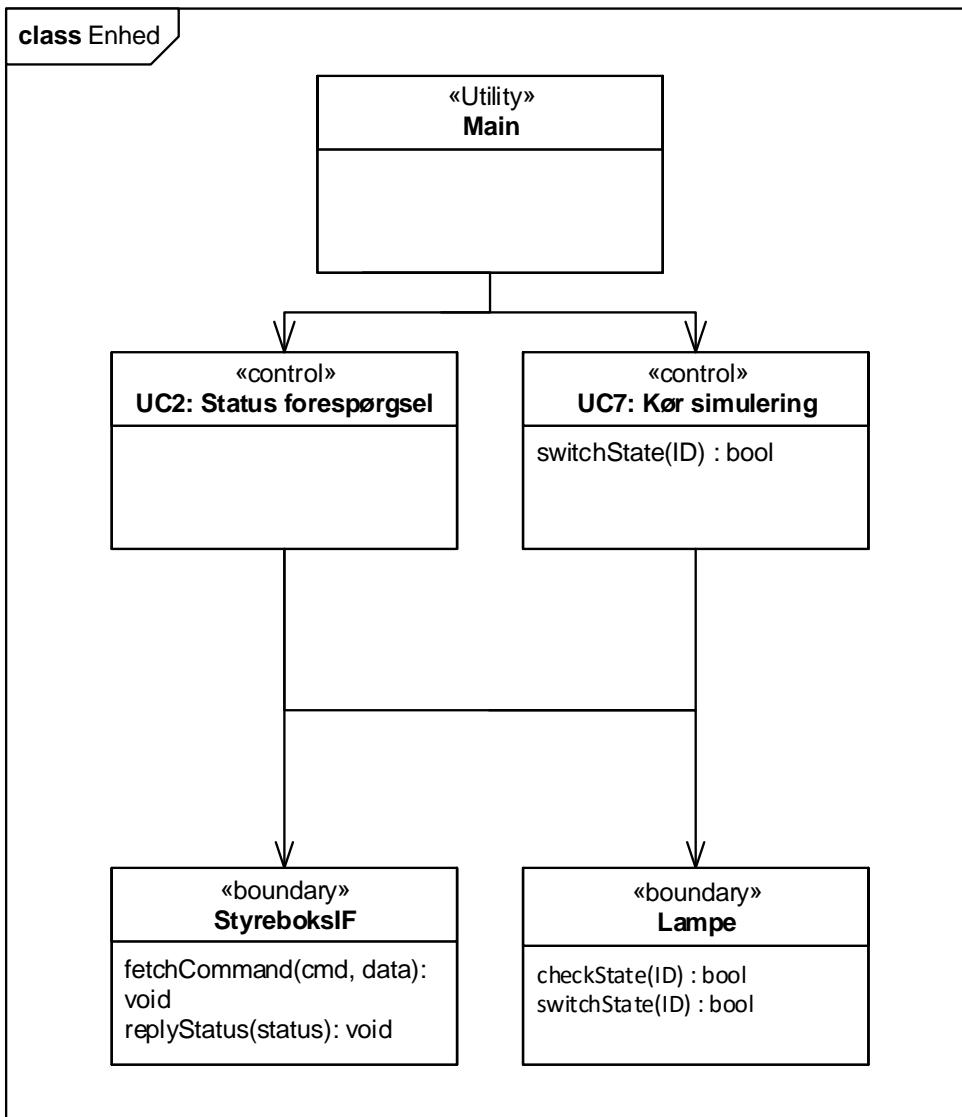
### 3.6.8 Metodeidentifikation – Styreboks

Ud fra klassediagrammerne identificeres metodekald mellem de individuelle blokke, som derefter indføres på styreboksens klassediagram. Dette ses på Figur 40 hvor metoderne fra sekvensdiagrammet er indført.

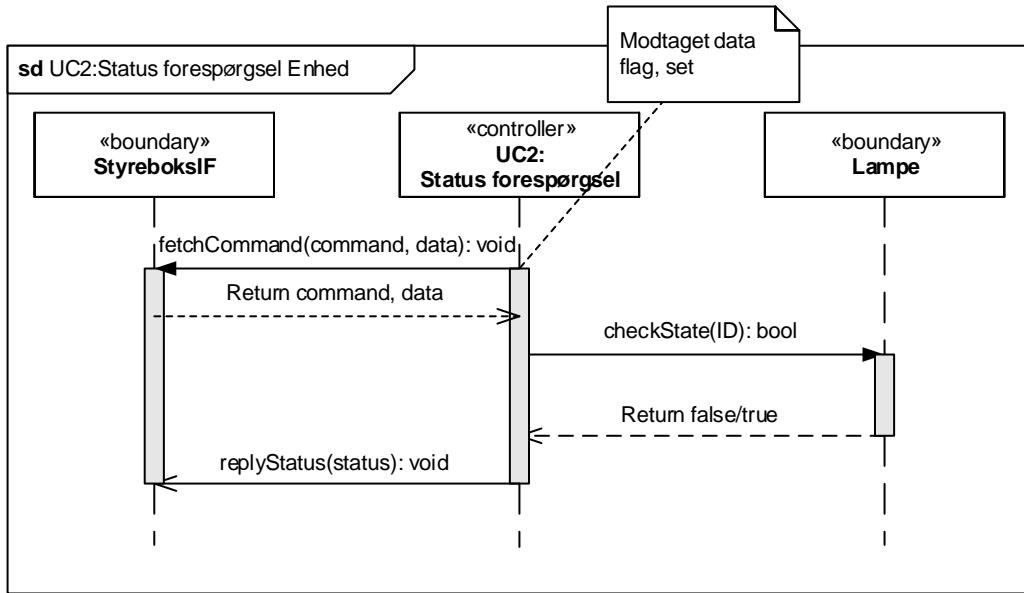


Figur 40 Udfyldt klassediagram for Styreboksen

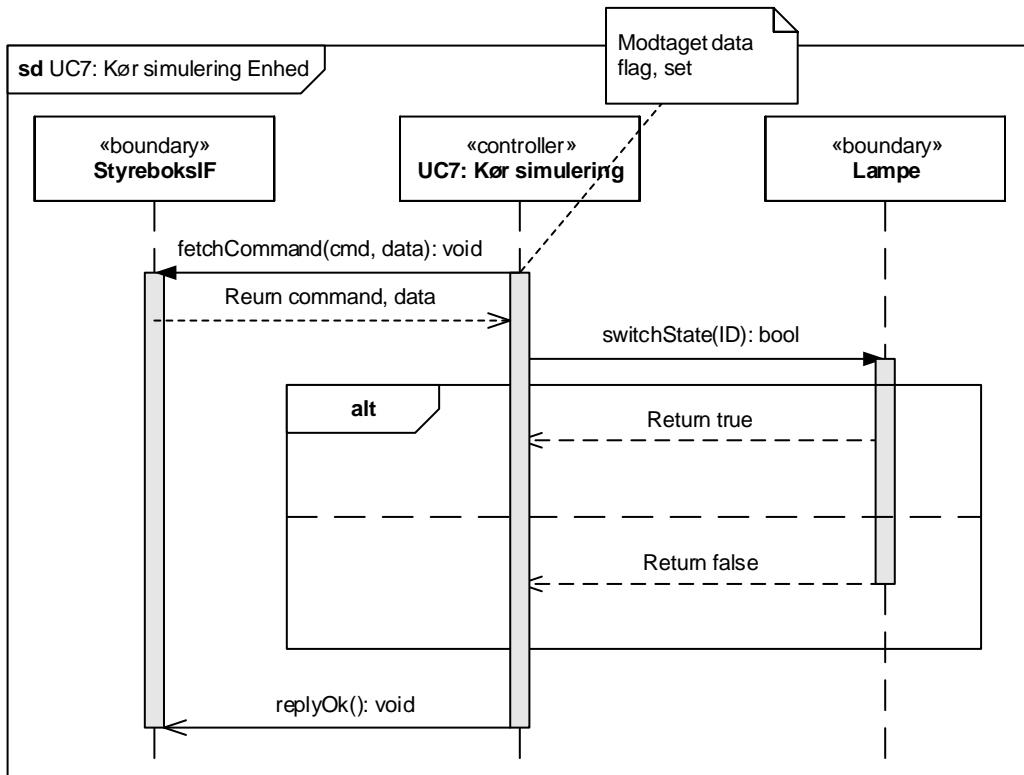
### 3.7 Applikationsmodel – Enhed (CBJ, MB, DP, SN, ME)



Figur 41 Klassediagram for Enhed



Figur 42 Sekvensdiagram for Status Forespørgsel : Enhed



Figur 43 Sekvensdiagram for UC7 : Kør Simulering

### 3.8 Design og Implementation – PC (NT, AK)

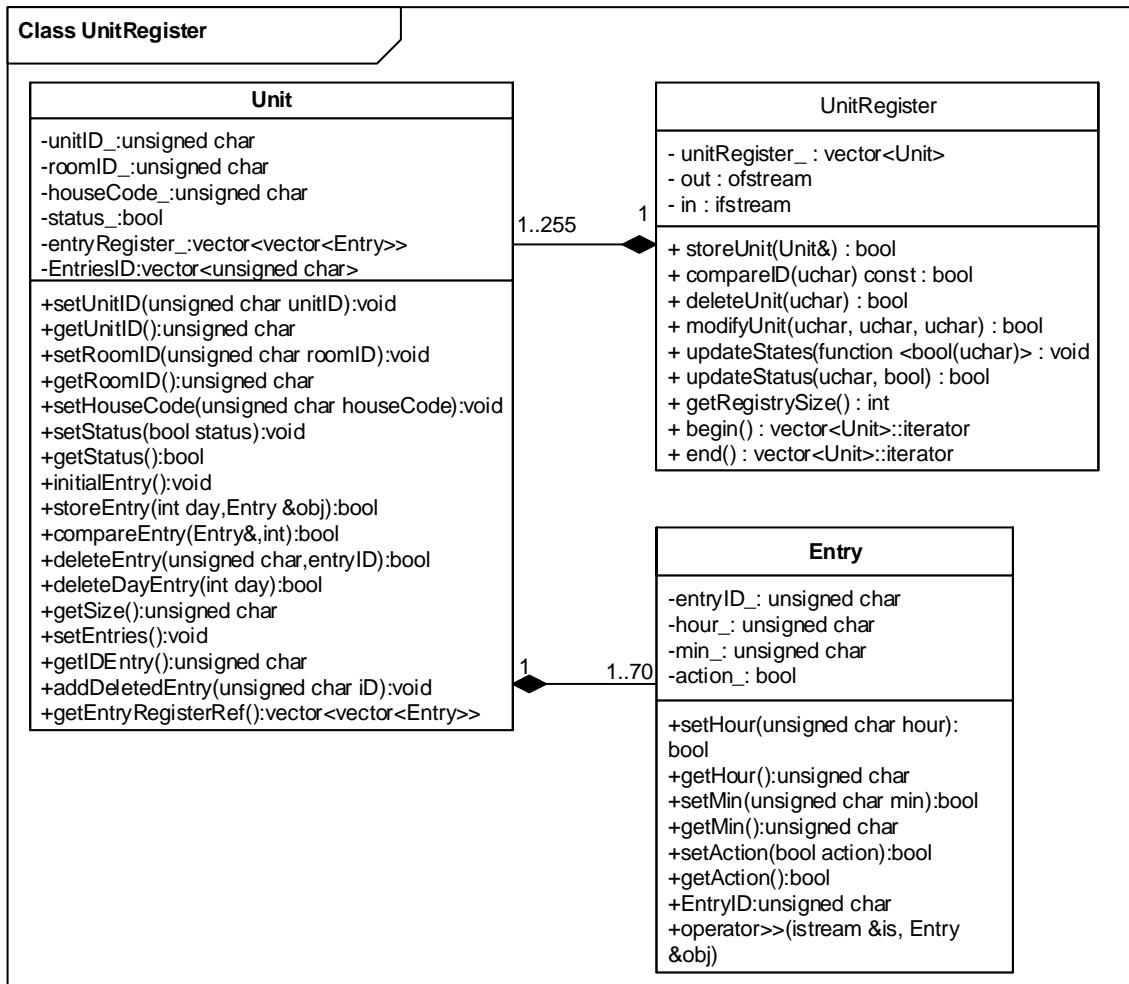
Herunder findes beskrivelser af design og implementation af PC software, herunder den grafiske brugerflade. Beskrivelsen af den grafiske brugerflade er opdelt efter den grafiske brugerflades otte vinduer. Disse vinduer repræsentere de use cases PC'en tager del i. Desuden findes **UnitRegister**, **Unit** og **Entry** klasser, som benyttes af alle use cases. Der vil findes en overordnet beskrivelse af de relevante klasser og deres ansvar, samt tilhørende billeder af den grafiske brugerflade. Samlet oversigt over metodebeskrivelser inkluderes til sidst i dette afsnit.

#### 3.8.1 UnitRegister

Klassen **UnitRegister** bruges til lagring af systemets enheder. Disse gemmes i en vector indeholdende **Unit** objekter. Disse **Unit** objekter indeholder et enheds ID, et rum ID, og en status. Desuden indeholder **Unit** en to-dimensionel vector af **Entry** objekter. **Unit** har en liste over ledige ID'er, der kan tildeles til et **Entry** når det lagres.

**Entry** klassen indeholder et ID for den givne entry, samt time og minut hvor en handling skal udføres, og en handling som skal udføres.

Fælles for de tre klasser er at **Set** og **Get** metoder benyttes til at hente og ændre klassernes attributter. Dette gøres for at indkapsle klassernes attributter, og for at validere ændringer til attributterne.



Figur 44 Klassediagram for UnitRegister

### 3.8.2 CommInterface

Klassen **CommInterface** håndtere kommunikation mellem PC og Styreboks. Klassen indeholder metoder til at sende nye enheder og ændringer til eksisterende enheder. Tilføjelse og ændring af entries til styreboksen gøres via én kommando, **sendEntries**, da det for styreboksens side er nødvendigt at modtage alle entries for en given enhed og dag, ikke kun den der ændres eller tilføjes.

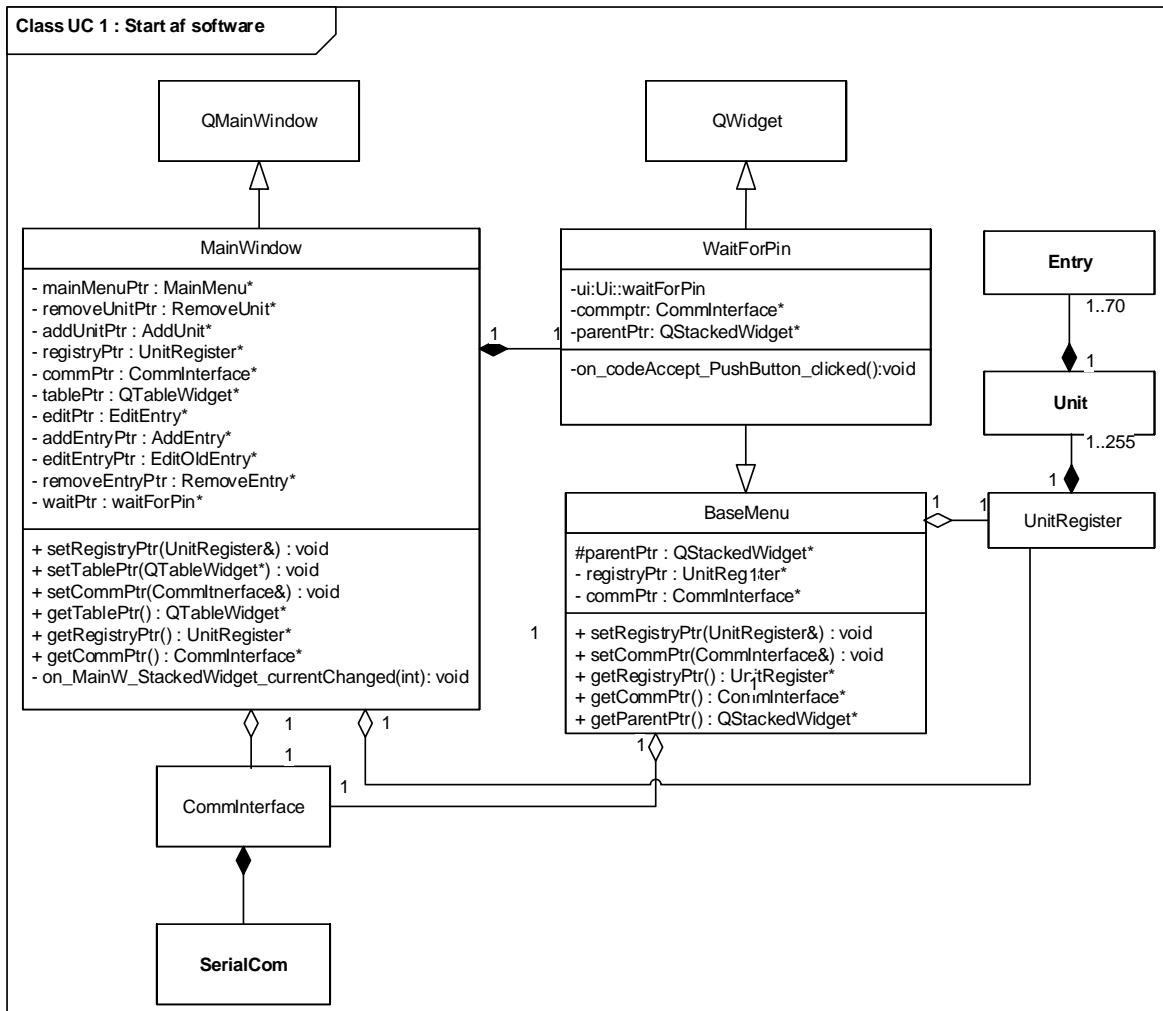
CommInterface
# portOpened : bool # unitRegPtr : UnitRegister* - serialCom : SerialCom  + setRegPtr(UnitRegister&): void + openComPort(int, int, int, int, int) : bool + closeComPort(): void + PCConnected(): bool + PCDisconnected(): void + validatePin(): bool + getAllUnits() : void + getUnitStatus(uchar): bool + sendUnit(uchar, uchar): bool + deleteUnit(uchar) : bool + editUnit(uchar, uchar, uchar): bool + sendEntries(Unit&, int) : bool + readAckCommand() : bool + sendCommand(char*, int) : void + readInputBuffer() : int

Figur 45 Klassen CommInterface

### 3.8.3 WaitForPin – Use Case 1 Opstart af System

Klassen WaitForPin vises når PC Softwaren startes. Klassen indeholder funktionalitet til at anmode styreboksen om status på pinkoden. Menuen WaitForPin har en enkelt knap:

**on\_codeAccept\_PushButton\_clicked()**. Denne knap kalder **validatePin()** på CommInterface, og skifter til **MainMenu** hvis koden er indtastet korrekt.



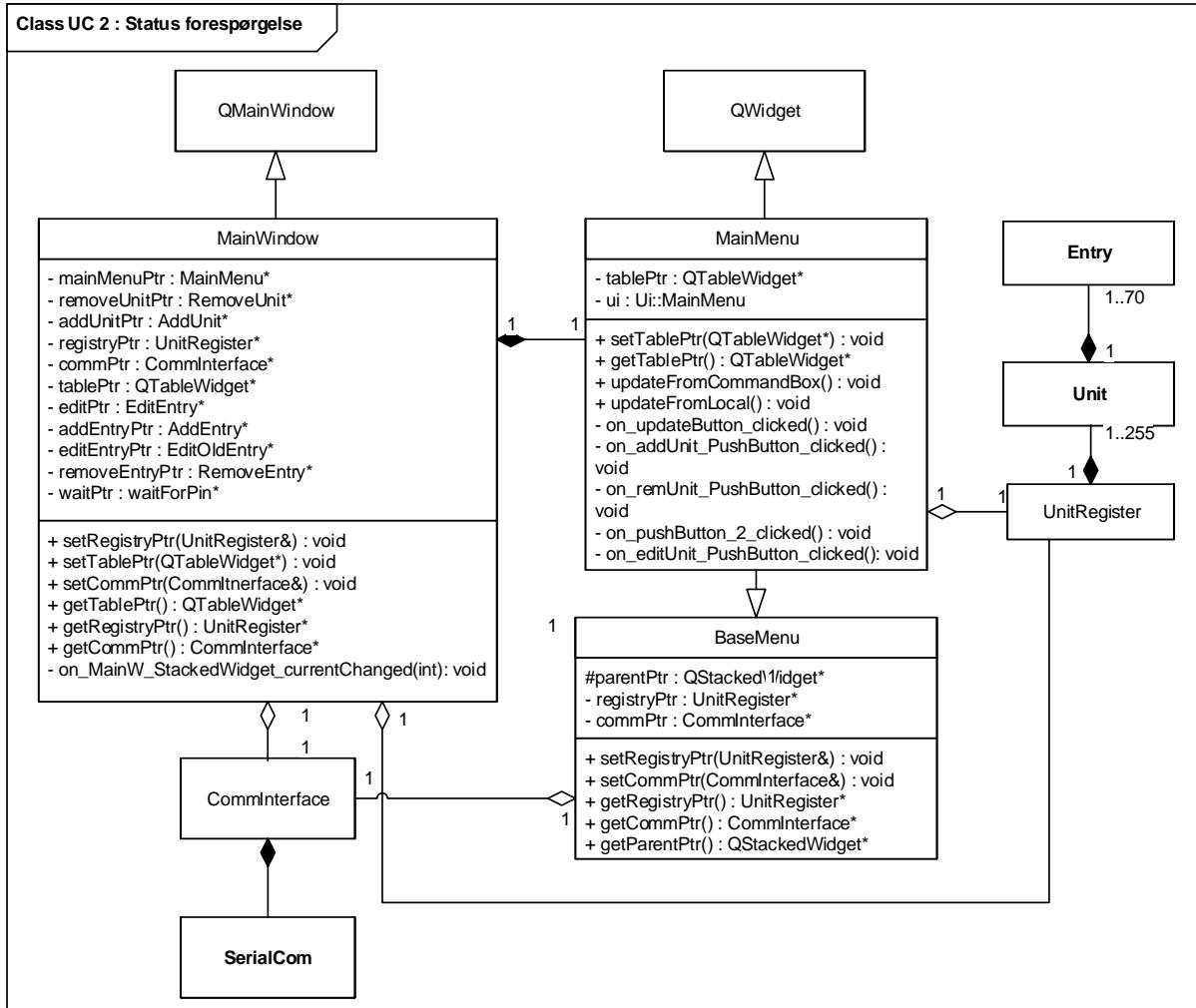
Figur 46 Klassediagram for WaitForPin



Figur 47 Billede af WaitForPin menu

### 3.8.4 MainMenu

MainMenu er brugerens adgang til programmers diverse undermenuer. Ydermere indeholder MainMenu en tabel hvori systemets tilsluttede enheder og deres status vises. MainMenu bruges desuden til udførelse af Use Case 2. Metoden **updateFromLocal()** kaldes automatisk når der skiftes til MainMenu. Denne opdaterer tabellen fra det lokale lager, så eventuelt nyligt tilføjede enheder vises i oversigten.



Figur 48 Klassediagram for MainMenu

MainMenu har 5 Knapper:

#### 1 Tilføj Enhed: **on\_addUnitPushButton\_clicked()**

Denne knap skifter til Use Casen for tilføjelse af enhed.

#### 2 Rediger Enhed: **on\_editUnitPushButton\_clicked()**

Denne knap skifter til Use Casen for ændring af enhed.

#### 3 Tidsplan : **on\_pushButton\_2\_clicked()**

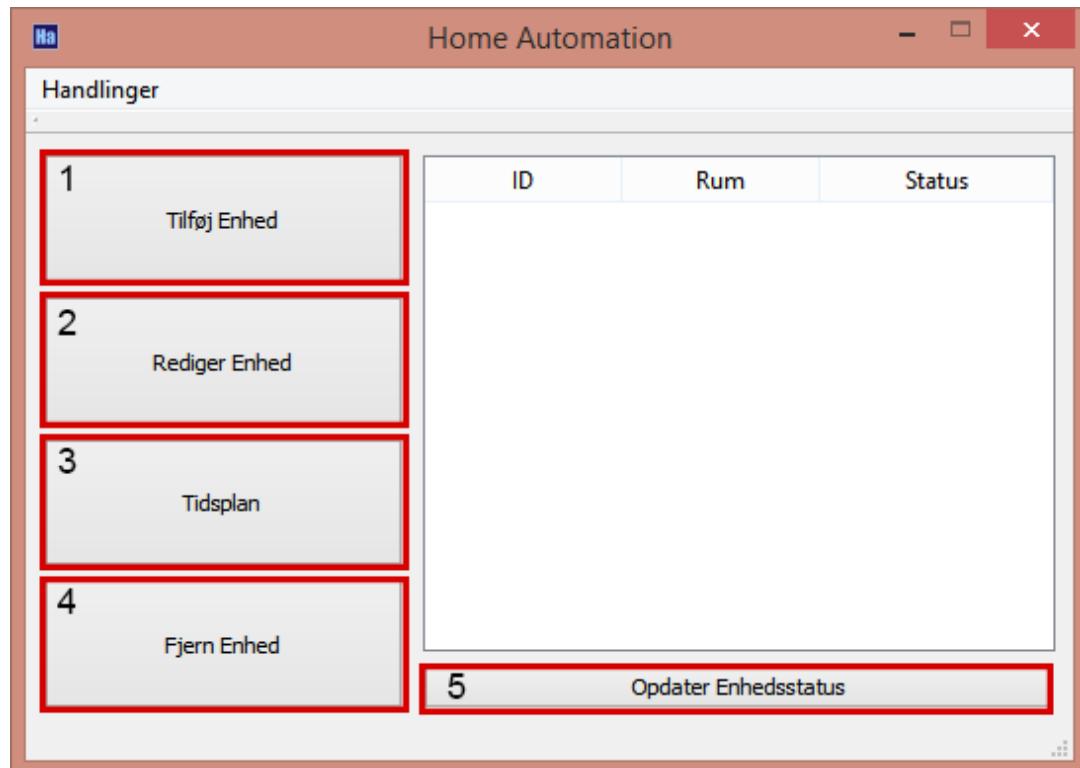
Denne knap åbner vindue for manipulation af tidsplaner, herunder tilføjelse, fjernelse og ændring deraf.

#### 4 Fjern Enhed : **on\_remUnitPushButton\_clicked()**

Denne knap skifter til Use Casen for fjernelse af enhed.

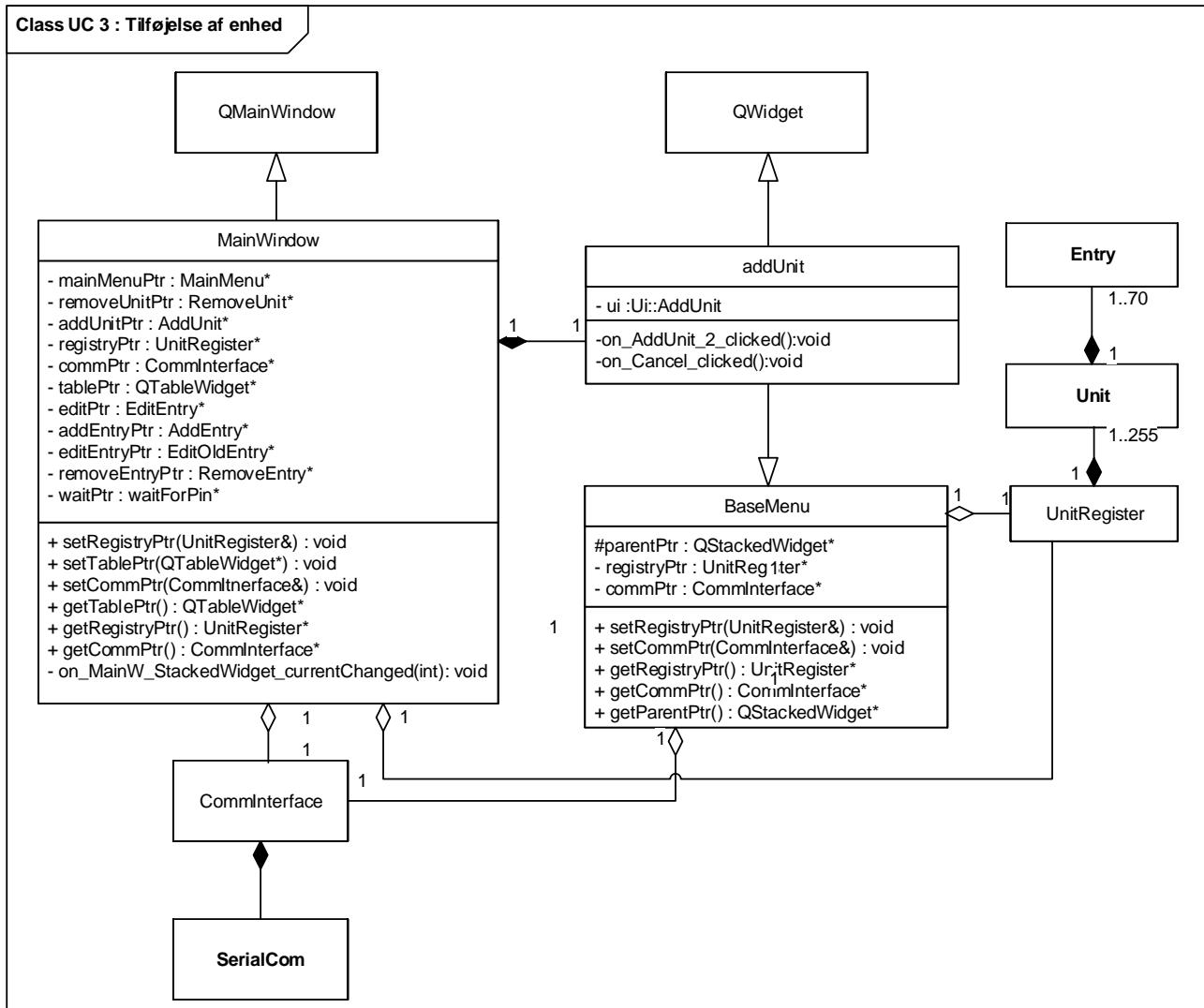
##### **5 Opdater Enhedsstatus : on\_updateButton\_Clicked()**

Denne knap udfører Use Case 2 – Status Forespørgsel. Registret af tilføjede enheder gennemløbes i metoden **updateFromCommandBox()**, og status opdateres i tabellen.



### 3.8.5 AddUnit

AddUnit bruges til at udføre Use Case 3, heri indtastes et Enheds ID og Room ID. Disse valideres ved indbyggede funktioner i QT til at sætte range for input boksen. Dette gøres for at forhindre bruger indtaster ugyldige værdier.



Figur 49 Klassediagram for addUnit

Skærmvinduet har 4 aktive felter:

### 1 Spinbox

Bruges til input af Enheds ID. Kan kun indeholde værdier fra 1-255

### 2 Spinbox

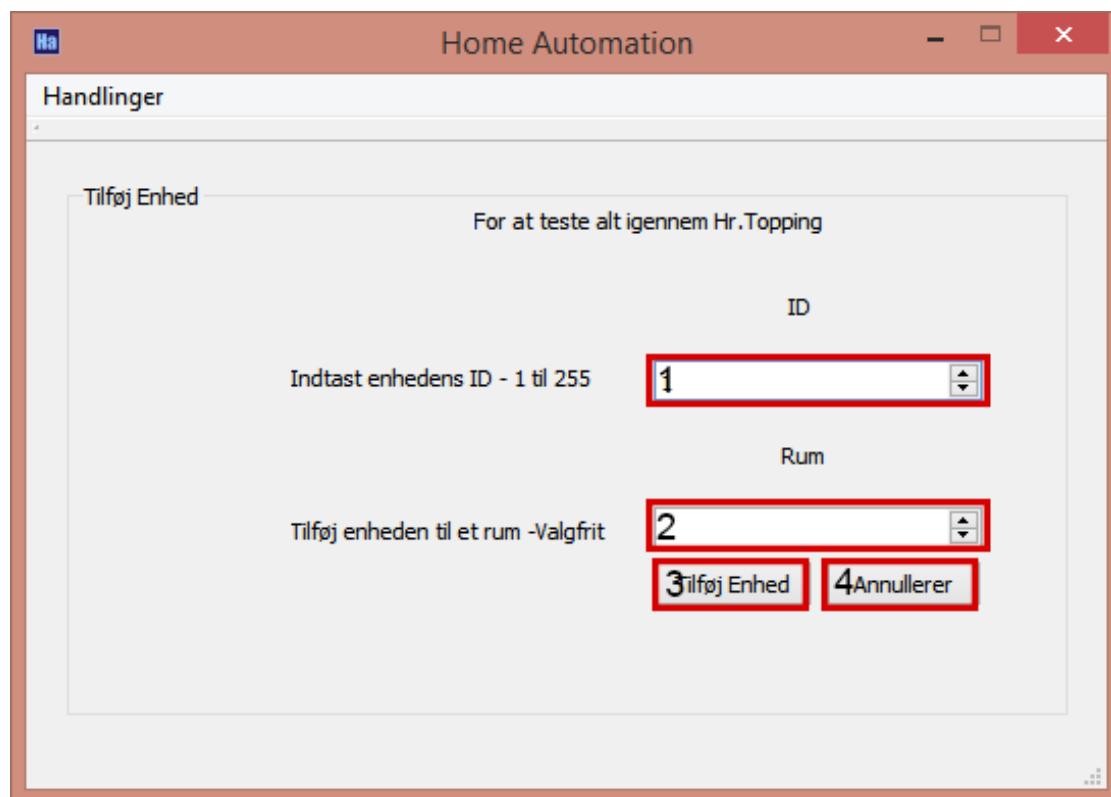
Bruges til input af enhedens rum ID. Kan kun indeholde værdier fra 1-255.

### 3 Tilføj Enhed : on\_AddUnit\_2\_Clicked()

Ved tryk på denne knap tilføjes en enhed med indtastet enheds ID og rum ID til enhedsregister. Den oprettede enhed sendes videre til styreboksen ved kald til CommInterface.

### 4 Annuler : on\_Cancel\_Clicked()

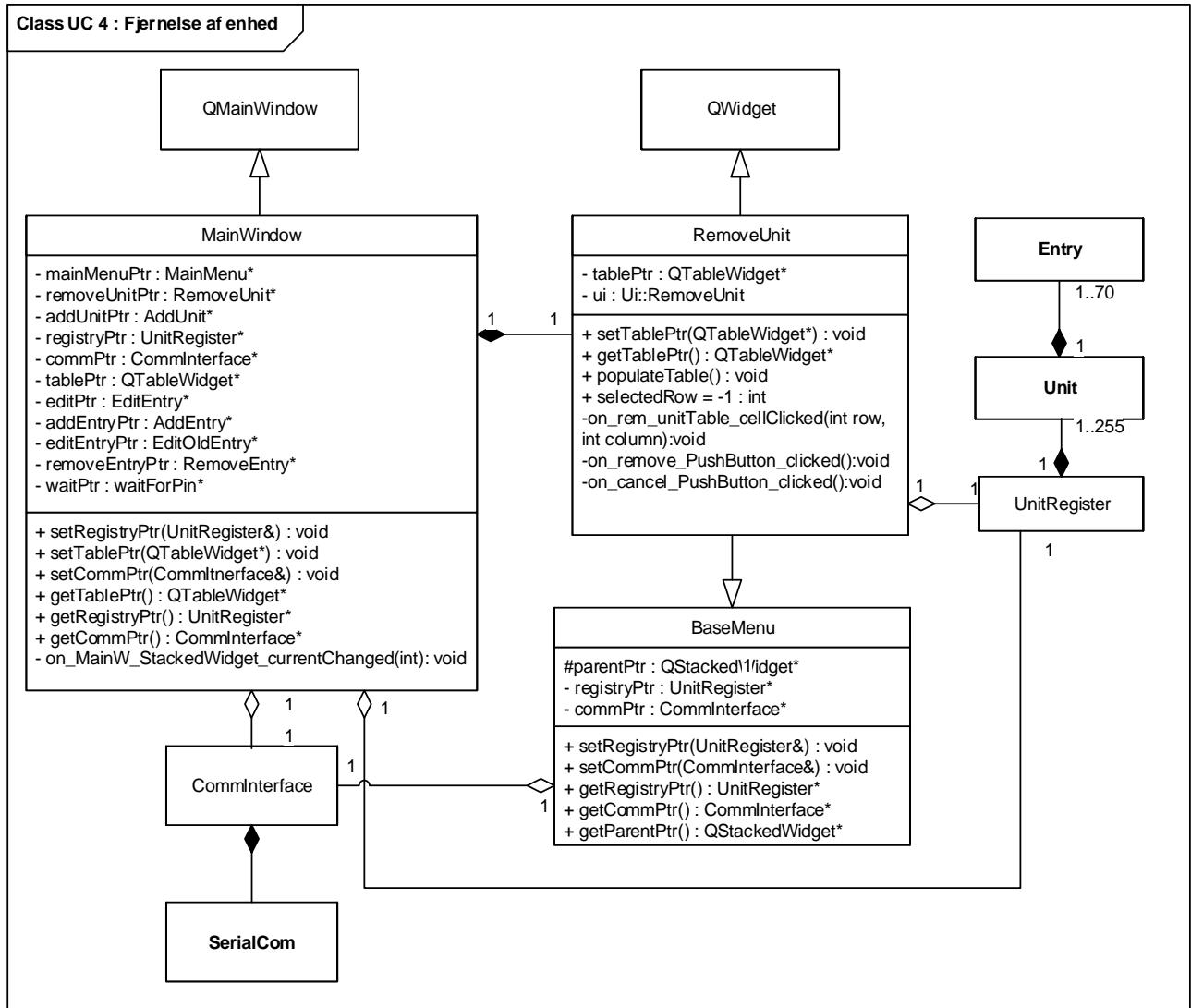
Denne knap returnerer programmet til MainMenu.



Figur 50 Skærmvindue for AddUnit

### 3.8.6 RemoveUnit

EditUnit indeholder funktionalitet til udførsel af Use Case 4. RemoveUnit indeholder en tabel over tilføjede enheder, og mulighed for at vælge en enhed der skal fjernes.



Figur 51 Klassediagram for Fjernelse af Enhed

Skærmvinduet gør brug af 3 knapper, samt en funktion der kaldes når der markeres en enhed i tabellen.

### 1 Fjern Enhed : on\_Remove\_PushButton\_Clicked()

Denne knap sletter den valgte enhed fra UnitRegister, og sender kommandoen for at fjerne enheden fra styreboksen. Hvis ingen enhed er valgt vises en fejlbesked og intet slettes.

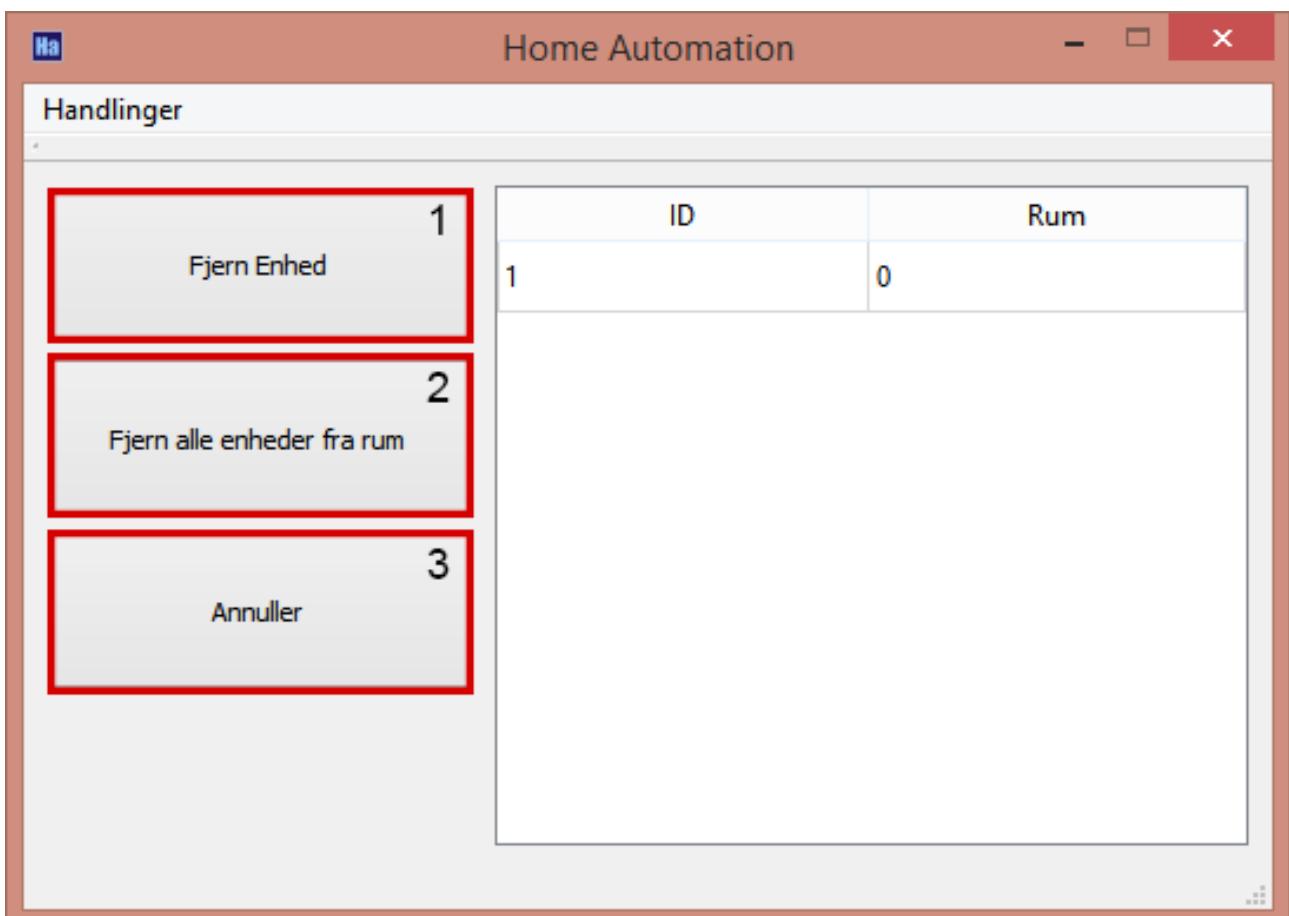
### 2 Fjern alle enheder fra rum

Funktionaliteten for denne knap er ikke implementeret.

### 3 Annuler : on\_cancel\_PushButton\_Clicked()

Denne knap returnerer brugeren til MainMenu.

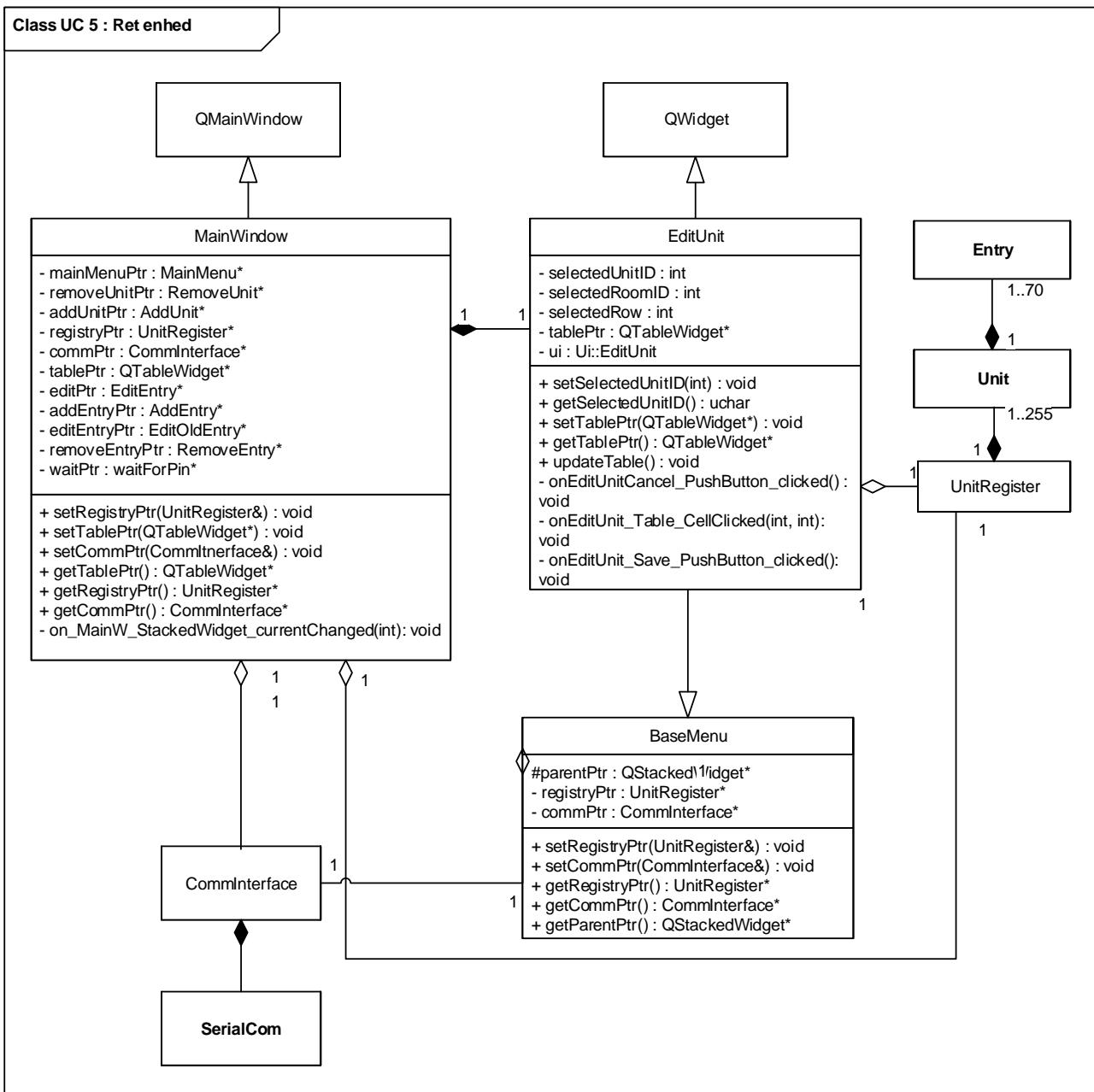
Desuden kaldes metoden **on\_rem\_UnitTable\_cellClicked** når der vælges en enhed i tabellen. Denne funktion opdatere attributten **selectedRow** som bruges til at bestemme hvilken enhed der skal fjernes.



Figur 52 Skærmvindue for Fjern Enhed

### 3.8.7 EditUnit

EditUnit bruges til at manipulere enheders informationer, i form af Enheds ID og Room ID.



Figur 53 Klassediagram for EditUnit

Menuen EditUnit indeholder 4 tekstfelter, hvoraf 2 ikke direkte kan skrives til af brugeren. Herunder er to knapper der bruges til at gemme og annullere. Vinduet viser en tabel over enheder der er tilføjet i systemet, med deres Enheds ID og Room ID.

### 1 Nuværende Enheds ID

Når en enhed vælges i tabellen opdateres **1** med den valgte enheds ID.

### 2 Nuværende Rum ID

Når en enhed vælges i tabellen opdateres **2** med den valgte enheds Room ID.

### 3 Nyt Enheds ID

Brugeren indtaster det nye ønskede ID for enheden. En QIntValidator bruges til at sørge for at kun integers fra 1-255 kan indtastes i feltet.

### 4 Nyt Rum ID

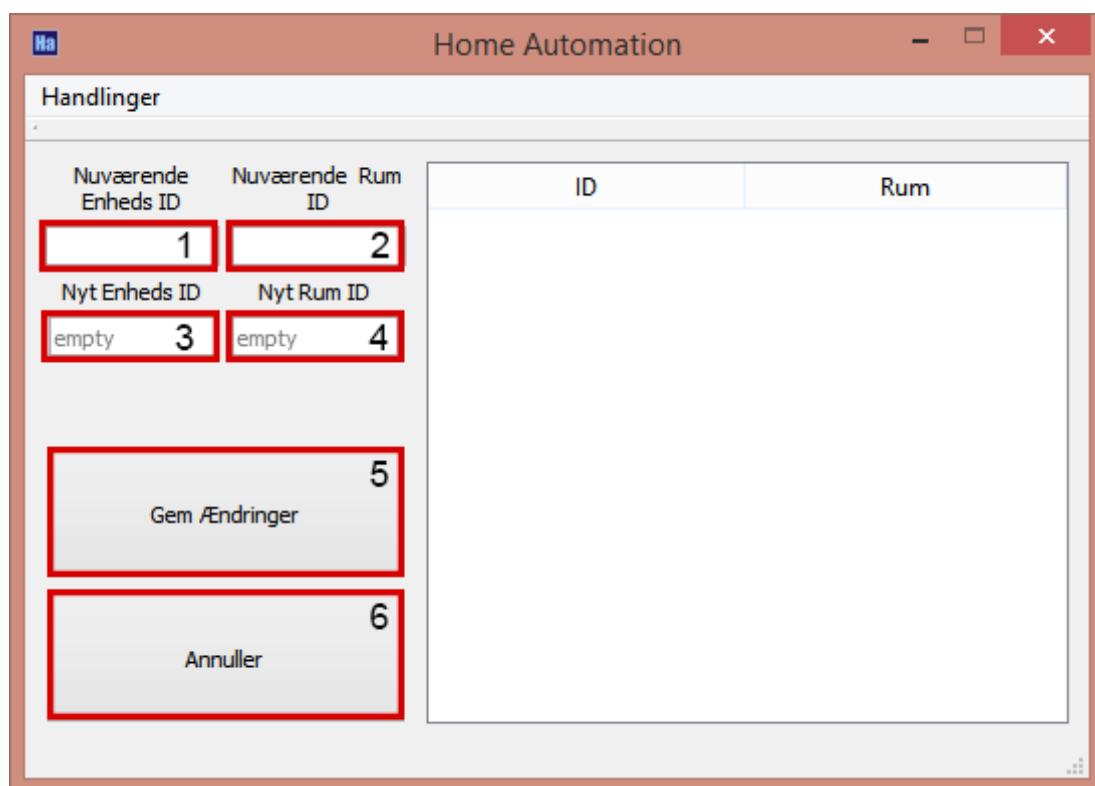
Brugeren indtaster det nye ønskede Room ID for enheden. En QIntValidator bruges til at sørge for at kun integers fra 1-255 kan indtastes i feltet.

### 5 Gem Ændringer : onEditUnit\_Save\_PushButton\_Clicked()

Denne knap kalder metode for at gemme de indtastede ændringer i enhedsregistret, samt at sende dem til styreboksen. Hvis ikke en enhed er valgt, eller de indtastede værdier er de samme som de nuværende vises en fejbesked og intet ændres.

### 6 Annuler : onEditUnitCancel\_PushButton\_Clicked()

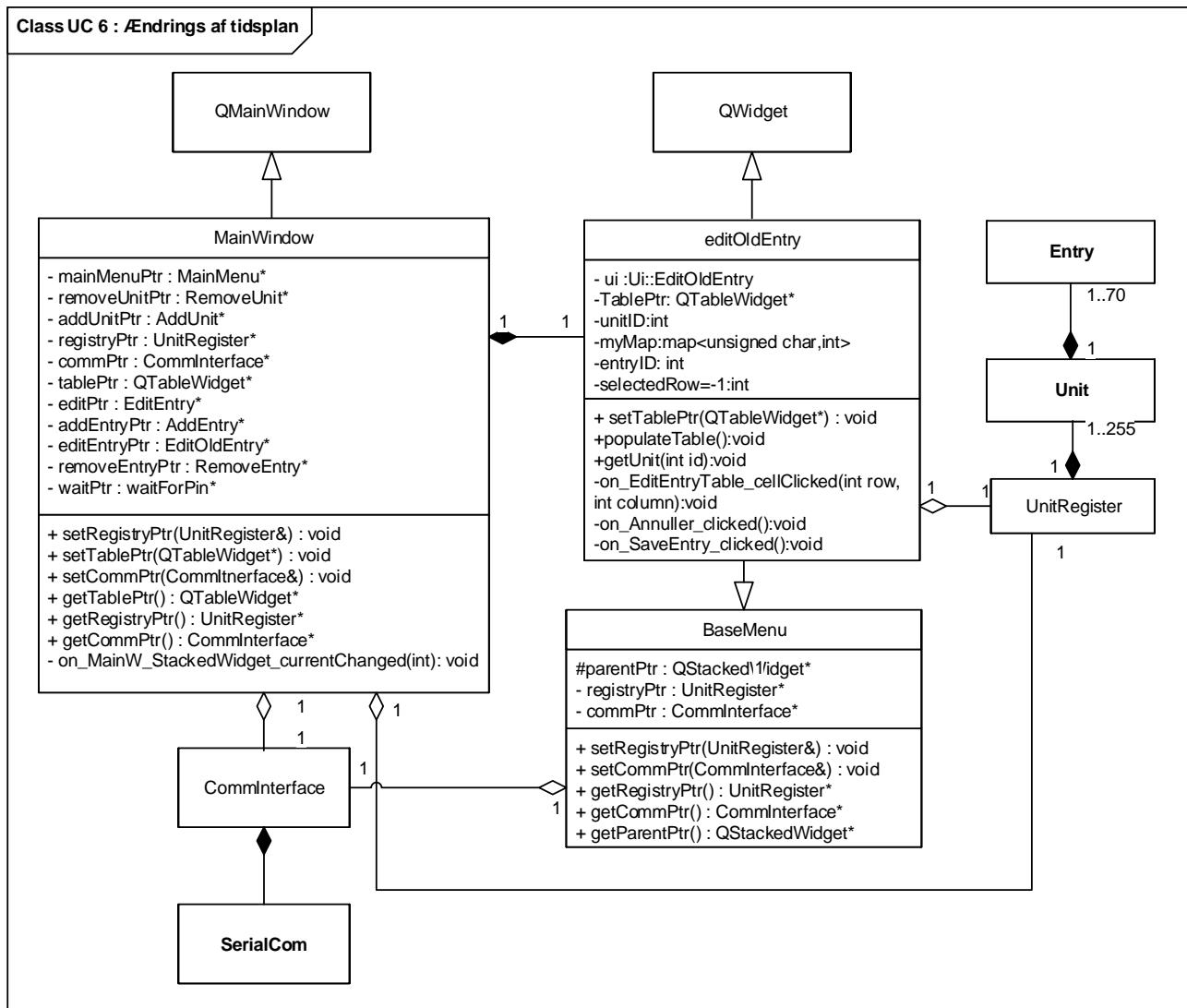
Denne knap returnerer brugeren til hovedmenuen.



Figur 54 Skærmbillede fra EditUnit

### 3.8.8 EditOldEntry

EditOldEntries benyttes til at redigere eksisterende tidsplaner for en valgt enhed. Her ændres starttidspunkt og sluttidspunkt.



Figur 55 Klassediagram for EditOldEntry

Menuen for EditOldEntry indeholder 5 aktive felter.

#### 1 Hvilken dag ønskes indstilles

Denne dropdown menu indeholder dagene fra Mandag til Søndag. Brugeren vælger en dag som tidsplanen skal være aktiv.

#### 2 Vælg nyt start tidspunkt

Denne spinbox bruges til at vælge et starttidspunkt. Indtastningsmetoden skifter automatisk mellem 12- og 24-timers ur, afhængig af computerens sprog.

#### 3 Vælg nyt slut tidspunkt

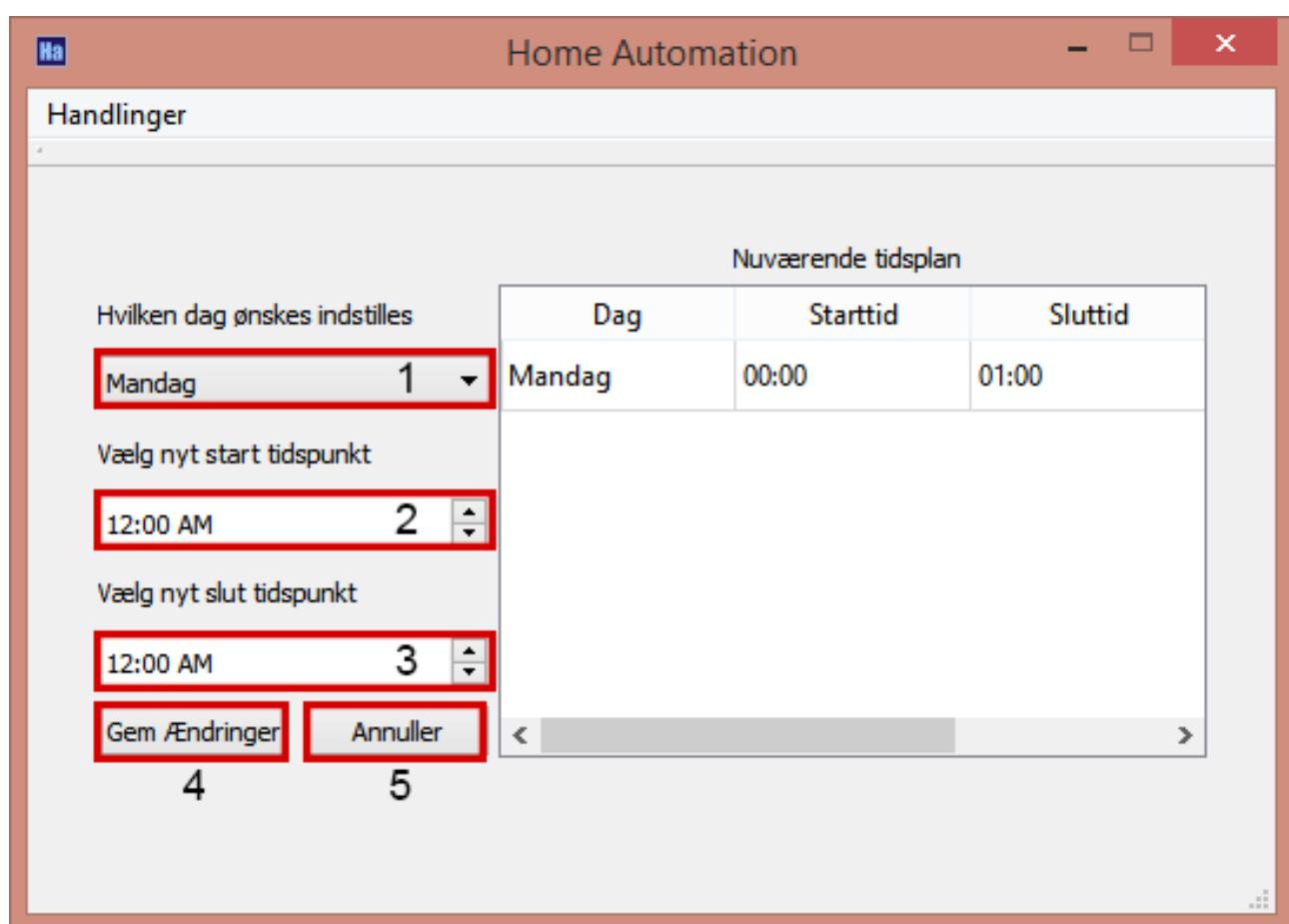
Denne spinbox bruges til at vælge et sluttidspunkt. Indtastningsmetoden skifter automatisk mellem 12- og 24-timers ur, afhængig af computerens sprog.

#### 4 Gem Ændringer : on\_SaveEntry\_clicked()

Denne knap gemmer de indtastede ændringer for enheden, og får informationen sendt til styreboksen.

#### 5 Annuler : on\_Annuler\_clicked()

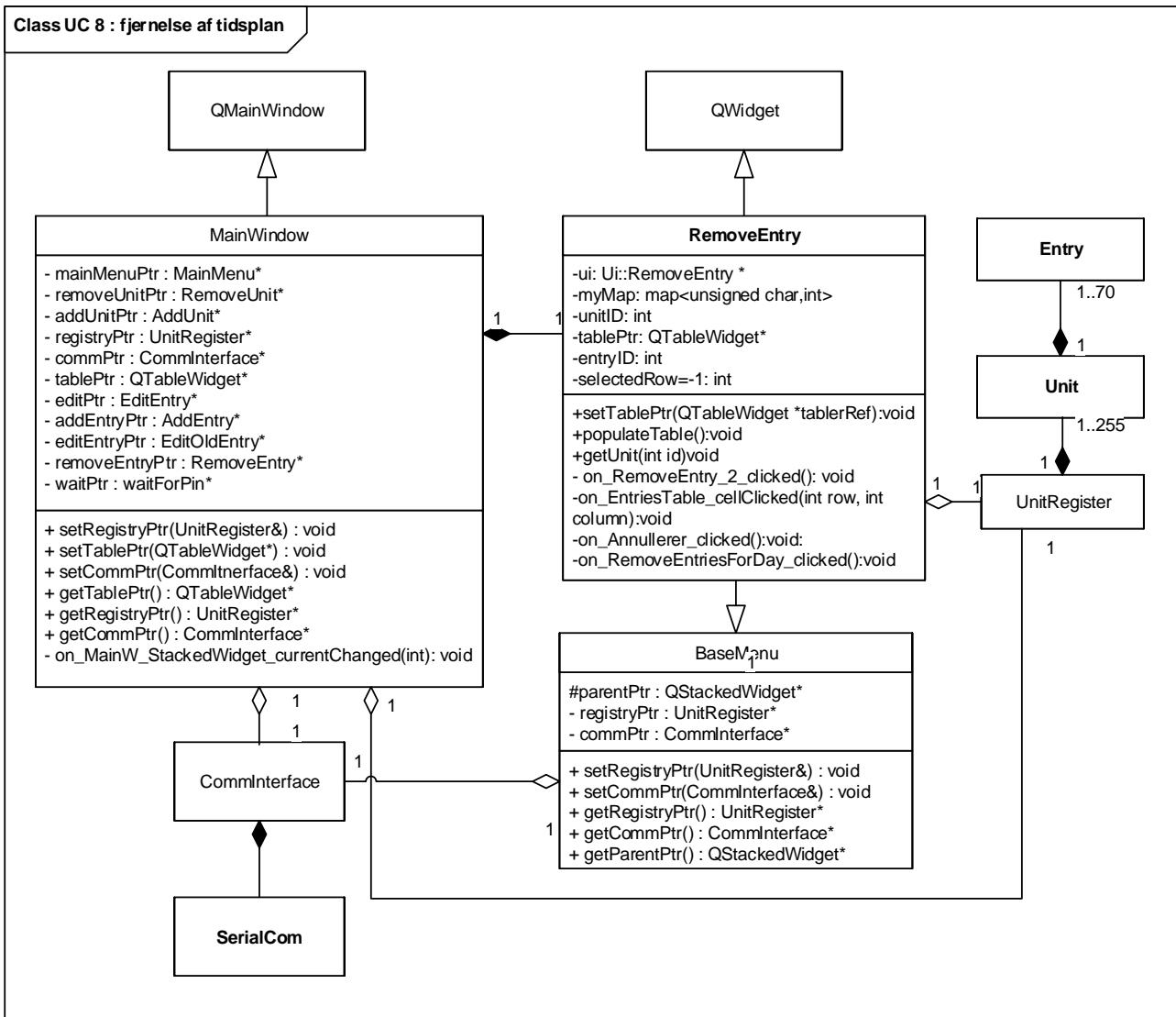
Denne knap returnerer brugeren til MainMenu.



Figur 56 Skærmvindue for EditOldEntry

### 3.8.9 RemoveEntry

RemoveEntry benyttes til at fjerne entries fra en enhed.



Figur 57 Klassediagram for RemoveEntry

Til fjernelse af tidsplaner findes 4 aktive felter.

**1 Fjern Valgte Tidsplan : on\_RemoveEntry\_2\_Clicked()**

Ved tryk på denne knap fjernes den valgte tidsplan. Er ingen tidsplan valgt skrives fejlbesked, og intet slettes.

**2 Hvilken dag skal alle tidsplan slettes**

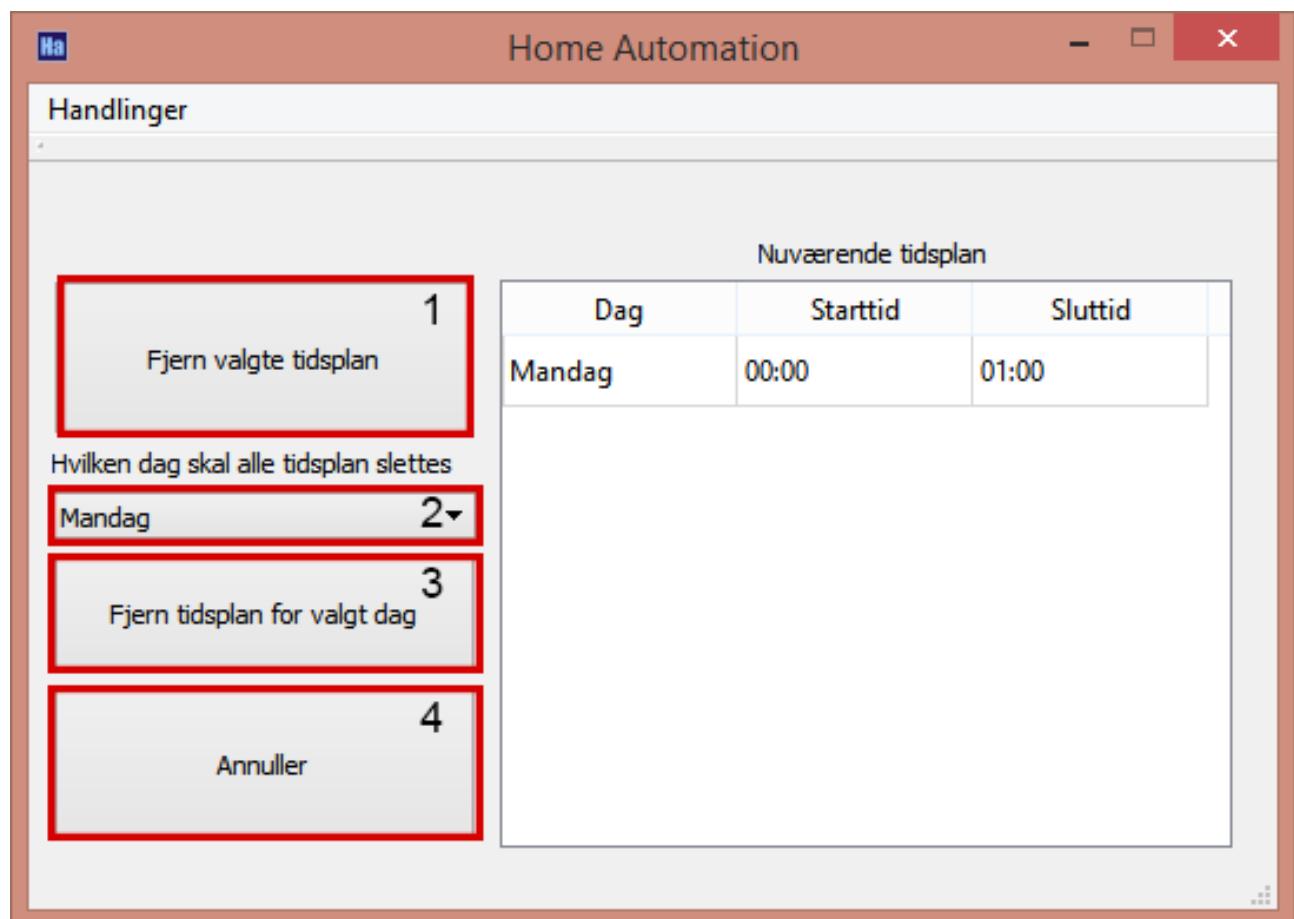
Dropdown menu indeholdende dagene mandag til søndag.

**3 Fjern tidsplan for valgt dag : on\_RemoveEntriesForDay\_clicked()**

Ved tryk på denne knap slettes alle tidsplaner for dagen valgt i 2.

**4 Annuler : on\_Annuler\_clicked()**

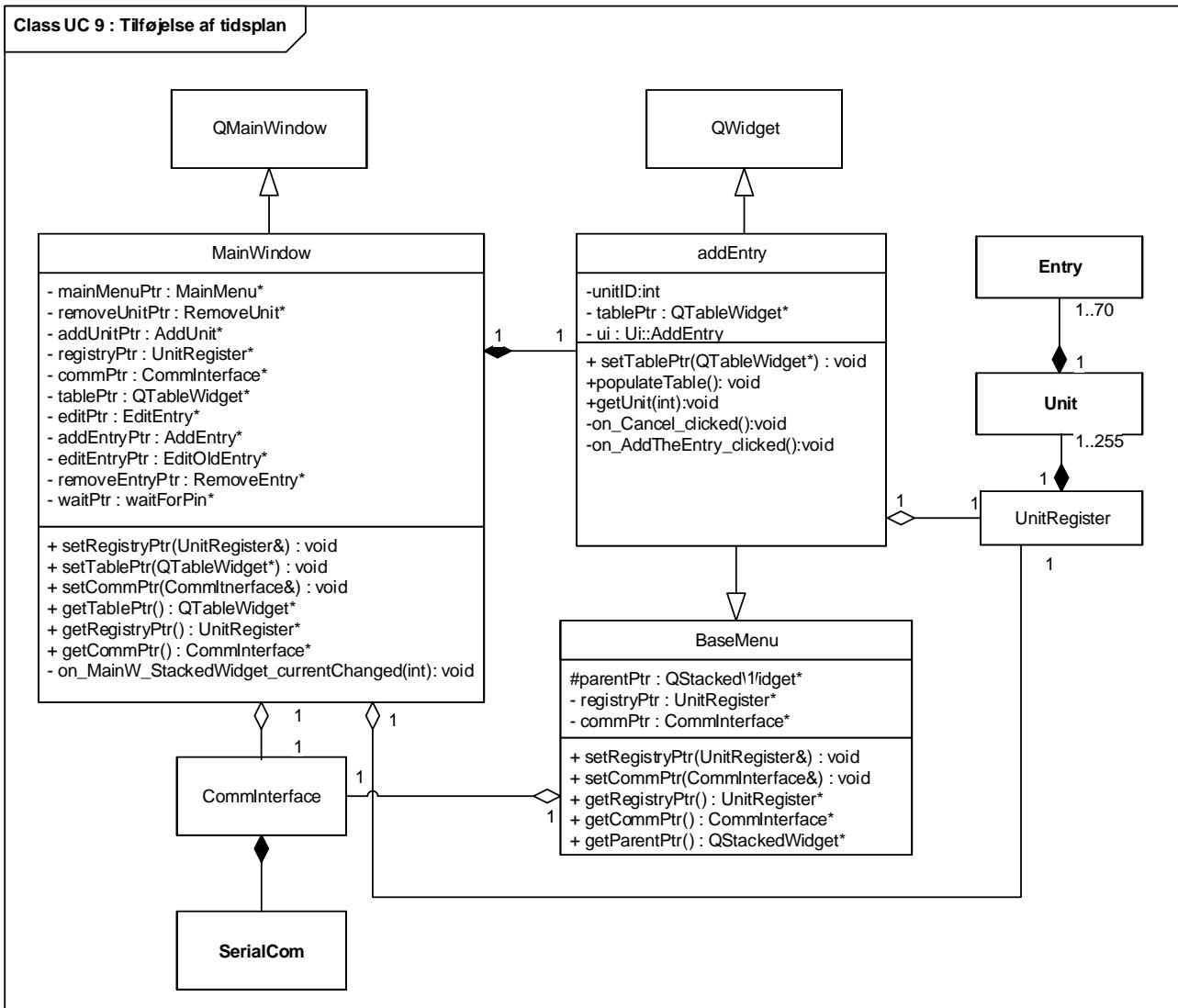
Ved tryk på denne knap returneres til MainMenu.



Figur 58 Skærmvindue for RemoveEntry

### 3.8.10 AddEntry

AddEntry klassen bruges til at tilføje en tidsplan til en valgt enhed.



Figur 59 Klassediagram for AddEntry

Menuen til AddEntry indeholder 5 aktive felter.

#### 1 Hvilken dag ønskes indstilles

Dropdown menu der indeholder dagene fra mandag til søndag. Brugeren vælger en dag fra menuen, og tidsplanen tilføjes til den.

#### 2 Vælg start tidspunkt

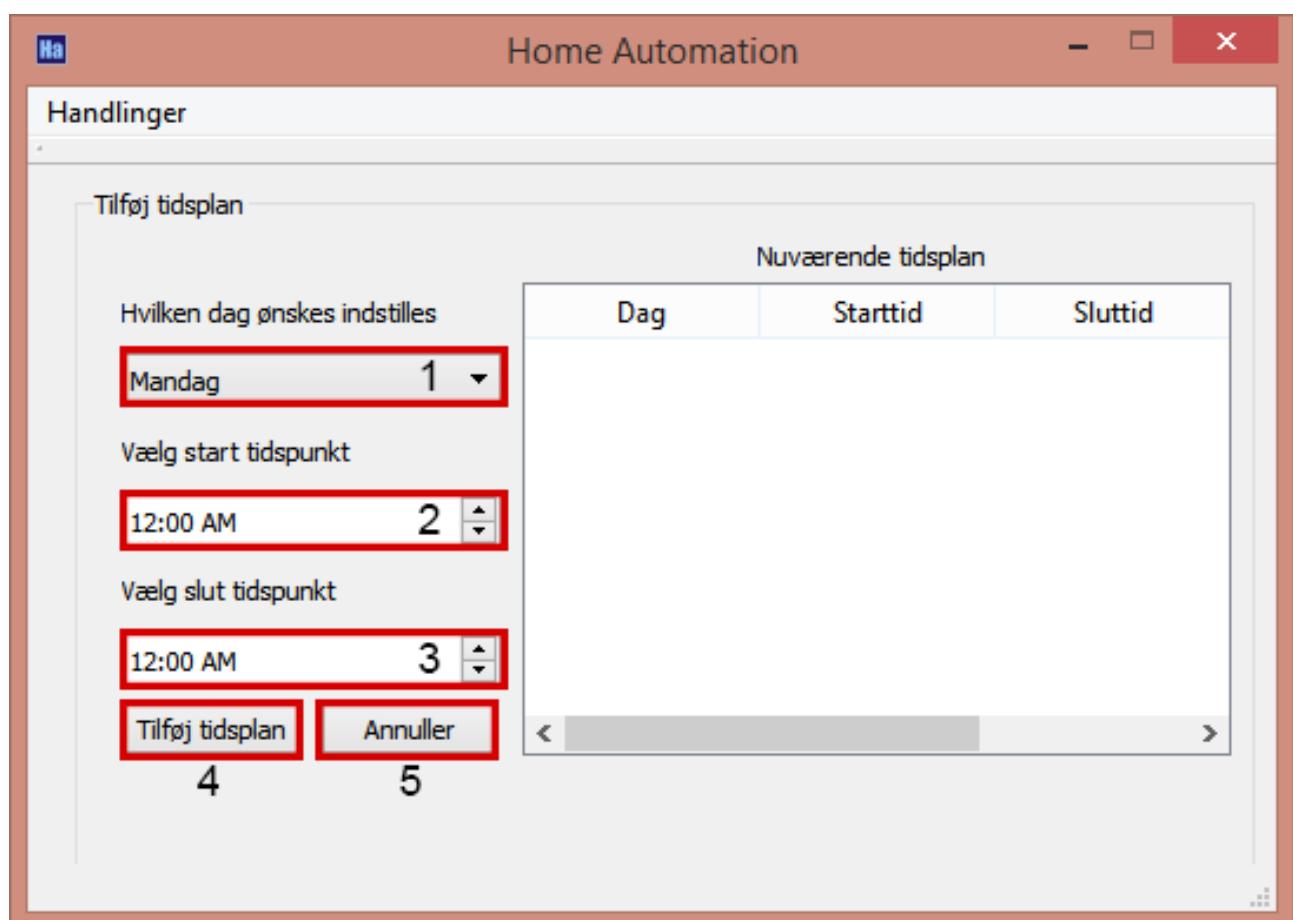
Denne spinbox bruges til at vælge et starttidspunkt. Indtastningsmetoden skifter automatisk mellem 12- og 24-timers ur, afhængig af computerens sprog.

#### 3 Vælg slut tidspunkt

Denne spinbox bruges til at vælge et sluttidspunkt. Indtastningsmetoden skifter automatisk mellem 12- og 24-timers ur, afhængig af computerens sprog.

#### 4 Tilføj Tidsplan : on\_AddTheEntry\_clicked()

Ved tryk på denne knap føjes tidsplanen til den valgt enhed. Hvis tidsplanens tidspunkter er indstillet til en ugyldig værdi, vises fejlbesked og intet gemmes.



Figur 60 Skærmvindue for AddEntry

### 3.8.11 Metodebeskrivelser

Herunder findes tabeller med beskrivelser af klassernes metoder.

#### 3.8.11.1 UnitRegister

Funktion	storeUnit(Unit&)
Parametre	Unit objekt
Returværdi	Bool
Beskrivelse	Lagre den givne enhed i unitRegister_ vectoren.

Funktion	compareID(uchar unitID)
Parametre	Enheds ID
Returværdi	Bool
Beskrivelse	Sammenligner et givet Enheds ID med Enheds ID'er gemt i unitRegister_. Returnerer true hvis ID'et findes i registret, false hvis ikke.

Funktion	mobidyUnit(uchar originalUnitID, uchar unitID, uchar roomID)
Parametre	Oprindelig Enheds ID, nyt Enheds ID og originalt/nyt Room ID
Returværdi	Bool
Beskrivelse	Ændre Enheds ID og Room ID for den valgte enhed. Returnerer true hvis det er succesfuldt, false hvis ikke.

Funktion	updateStates(function<bool(uchar)> f)
Parametre	Benytter en lambda expression
Returværdi	Ingen
Beskrivelse	Bruges til at opdatere status for alle enheder i registret. Kilde for funktionalitet beskrevet i bilag – Stack Overflow.
Funktion	updateStatus(uchar unitID, bool status)
Parametre	Enheds ID for enheden der skal opdateres og status der skal skrives til enheden.
Returværdi	Bool
Beskrivelse	Returnerer true hvis ændring er succesfuld.

Funktion	getRegistrySize()
Parametre	Ingen
Returværdi	Int
Beskrivelse	Returnerer størrelsen på unitRegister

Funktion	begin()
Parametre	Ingen
Returværdi	vector<Unit>::iterator
Beskrivelse	Returnerer en iterator der peger på starten af unitRegister

Funktion	end()
Parametre	Ingen
Returværdi	vector<Unit>::iterator
Beskrivelse	Returnerer en iterator der peger på enden af unitRegister

### 3.8.11.2 Unit

Funktion	Unit(unsigned char unitID, unsigned char roomID, unsigned char houseCode, bool status)
Parametre	Enheds iD, Room ID, Housecode(sættes til 0) og en initial status.
Returværdi	Ingen
Beskrivelse	Klassens constructor. Sætter enhedens ID og status attributter.

Funktion	setUnitID(unsigned char unitID)
Parametre	Enheds iD
Returværdi	Ingen
Beskrivelse	Sætter Enheds ID for Unit.

Funktion	getUnitID()
Parametre	Ingen
Returværdi	Unsigned char
Beskrivelse	Returnerer Enheds ID for Unit

Funktion	setRoomID(unsigned char roomID)
Parametre	Room ID
Returværdi	Ingen
Beskrivelse	Indstiller Room ID for enheden til det givne ID.
Funktion	getRoomID()
Parametre	Ingen
Returværdi	Unsigned Char
Beskrivelse	Returnerer Room ID for Unit

Funktion	setHouseCode(unsigned char houseCode)
Parametre	Hosue Code til Unit
Returværdi	Ingen
Beskrivelse	Indstiller Housecode for Unit

Funktion	getHosueCode()
Parametre	Ingen
Returværdi	Unsigned char
Beskrivelse	Returnerer house code fra Unit

Funktion	<code>setStatus(bool status)</code>
Parametre	Status for enheden : True for tænd, false for slukket.
Returværdi	Ingen
Beskrivelse	Indstiller status for Unit.

Funktion	<code>getStatus()</code>
Parametre	Ingen
Returværdi	Boolean
Beskrivelse	Returnerer enhedens status. True hvis tændt, false hvis slukket.

Funktion	<code>initialEntry()</code>
Parametre	Ingen
Returværdi	Ingen
Beskrivelse	Opsætter to dimensionel vector indeholdende Entries.

Funktion	<code>storeEntry/int day, Entry&amp; obj)</code>
Parametre	Dag hvor entry skal gemmes, og den Entry der skal gemmes.
Returværdi	Bool
Beskrivelse	Gemmer Entry i den valgte dag. Returnerer true hvis successful, false hvis ikke.

Funktion	<code>compareEntry(Entry&amp;, int d)</code>
Parametre	Entry object og dagen det befinder sig i.
Returværdi	Boolean
Beskrivelse	Returnerer true hvis en ens entry findes, false hvis ikke.

Funktion	<code>deleteEntry(unsigned char entryID)</code>
Parametre	ID for den Entry der skal slettes.
Returværdi	Boolean
Beskrivelse	Sletter entry med valgte ID. Returnerer true hvis successful, false hvis ikke.

Funktion	<code>deleteDayEntry(int day)</code>
Parametre	Den dag som skal tømmes for Entries
Returværdi	Boolean
Beskrivelse	Sletter alle entries for den valgte dag. Returnerer true hvis successful, false hvis ikke.

Funktion	<code>getSize()</code>
Parametre	Ingen
Returværdi	Unsigned char
Beskrivelse	Returnerer det totale antal Entries for Unit

Funktion	setEntries()
Parametre	Ingen
Returværdi	Ingen
Beskrivelse	Tilføjer ledige Entry Ids til vector indeholdende ledige Entry ID's .

Funktion	getIDEntry()
Parametre	Ingen
Returværdi	Unsigned char
Beskrivelse	Returnerer et ledig Entry ID som kan tildeles til en enhed. Fjerner dette ID fra de ledige ID's

Funktion	addDeletedEntry(unsigned char ID)
Parametre	Unsigned char med Entry ID.
Returværdi	Ingen
Beskrivelse	Føjer det givne Entry ID til listen over ledige IDs.

### 3.8.11.3 Entry

Funktion:	Entry(unsigned char entryID,unsigned char hour, unsigned char min, bool action)
Parametre:	unsigned char entryID,unsigned char hour, unsigned char min, bool action)
Returværdi:	Ingen.
Beskrivelse:	Constructor

Funktion:	Bool setHour(unsigned char hour)
Parametre:	Unsigned char
Returværdi:	Bool
Beskrivelse:	Funktionen sætter attributen hour for entry

Funktion:	Unsigned char getHour()
Parametre:	Ingen
Returværdi:	Unsigned char
Beskrivelse:	Funktionen returner attributten hour for entry

Funktion:	Bool unsigned char setMin(unsigned char min)
Parametre:	Unsigned char
Returværdi:	Bool
Beskrivelse:	Funktionen sætter attributten min for entry

Funktion:	Unsigned char getMin()
Parametre:	Ingen
Returværdi:	Unsigned char
Beskrivelse:	Funktionen returner attributten min for entry

Funktion:	Bool setAction(bool action)
Parametre:	Bool action
Returværdi:	Bool
Beskrivelse:	Funktionen sætter attributten action for entry

Funktion:	Bool getHour()
Parametre:	Ingen
Returværdi:	Bool
Beskrivelse:	Funktionen returner attributten action for entry

### 3.8.11.4 CommInterface

Funktion	setRegPtr(UnitRegister& regRef)
Parametre	Addressen til unitregister
Returværdi	Ingen.
Beskrivelse	Sætter pointeren til UnitRegister til addressen regRef

Funktion	openComPort(int port, int baud, int dataBit, int paritet, int stopBit)
Parametre	Port: Com Porten der benyttes. Baud: Baudrate der sendes med. dataBit: Antal databits. Paritet: Hvilken type paritet der benyttes. stopBit
Returværdi	Bool: Sand hvis porten er åbnet, falsk hvis ikke.
Beskrivelse	Bruges til at åbne com porten til kommunikation mellem PC og styreboks.

Funktion	closeComPort
Parametre	Ingen
Returværdi	Ingen
Beskrivelse	Lukker com porten

Funktion	sendCommand(char* cmd, int cmdSize)
Parametre	Char* eller string, samt længden på string eller char array.
Returværdi	Ingen
Beskrivelse	Sender en kommando til styreboksen.

Funktion	readInputBuffer()
Parametre	Ingen
Returværdi	Int
Beskrivelse	Læser input bufferen til en fil. Returnere en int med antallet af bytes hentet.

Funktion	PCConnected()
Parametre	Ingen
Returværdi	Bool
Beskrivelse	Kaldes når PC softwaren startes.

Funktion	PCDisconnected()
Parametre	Ingen
Returværdi	Ingen
Beskrivelse	Sendes når PC Softwaren lukkes.

Funktion	validatePin()
Parametre	Ingen
Returværdi	Bool
Beskrivelse	Anmoder styreboks om status på pin kode. Returnere true hvis pinkode er indtastet. False hvis ikke.

Funktion	getUnitStatus(unsigned char unitID)
Parametre	Enheds ID for den status der ønskes at hente status for.
Returværdi	Bool
Beskrivelse	Anmoder styreboks om status på given enhed. Returnerer true hvis enheden er tændt, false hvis ikke.

Funktion	getAllUnits()
Parametre	Ingen
Returværdi	Ingen
Beskrivelse	Anmoder styreboks om at sende alle lagrede enheder. Enheder sendes én af gangen. Funktionen er rekursiv, og kalder sig selv indtil styreboks returnere en datamængde der ikke svarer til en komplet enhed.

Funktion	sendUnit(unsigned char unitID, unsigned char roomID)
Parametre	Enheds ID og Room ID
Returværdi	Bool
Beskrivelse	Returnerer true hvis styreboksen godkender lagring af enhed. False hvis ikke.

Funktion	deleteUnit(unsigned char unitID)
Parametre	Enheds ID
Returværdi	Bool
Beskrivelse	Anmoder styreboksen om at slette enhed med givet ID. Returnerer true hvis sletning er successful, false hvis ikke.

Funktion	editUnit(unsigned char previousID, unsigned char newID, unsigned char roomID)
Parametre	Det nuværende Enheds ID, det nye Enheds ID og originalt/nyt Room ID
Returværdi	Bool
Beskrivelse	Anmoder styreboksen om at ændre den givne enhed. Returnerer true hvis ændringen er successful, false hvis ikke.

Funktion	sendEntries(Unit& obj, int day)
Parametre	Addressen til en given enhed, og dagen for de entries der skal sendes.
Returværdi	Bool
Beskrivelse	Anmoder styreboks om at gemme alle entries for en given dag for den valgte enhed. Returnerer true hvis lagring var successful, false hvis ikke.

Funktion	readAckCommand()
Parametre	Ingen
Returværdi	Bool
Beskrivelse	Læser om ACK kommando er sendt fra styreboks. Returnerer true hvis kommando er modtaget, false hvis ikke.

### 3.8.11.5 MainWindow

Funktion:	MainWindow(UnitRegister&ref,CommInterface &commRef,QWidget *parent = 0)
Parametre:	UnitRegister&ref,CommInterface &commRef,QWidget *parent = 0)
Returværdi:	Ingen.
Beskrivelse:	Constructor

Funktion:	void setRegistryPtr(UnitRegister&)
Parametre:	Adressen til UnitRegister
Returværdi:	Ingen.
Beskrivelse:	Funktionen sætter en pointer til klassen UnitRegister

Funktion:	<code>Void setTablePtr(QTableWidget*)</code>
Parametre:	Pointer til QTableWidget
Returværdi:	Ingen.
Beskrivelse:	Funktionen sætter en pointer til tabellen

Funktion:	<code>void setCommPtr(CommInterface&amp;)</code>
Parametre:	Adressen til CommInterface
Returværdi:	Ingen.
Beskrivelse:	Funktionen sætter en pointer til klassen commInterface

Funktion:	<code>QTableWidget* getTablePtr()</code>
Parametre:	Ingen
Returværdi:	Ingen.
Beskrivelse:	Funktionen returner en pointer til tabellen

Funktion:	<code>UnitRegister* getRegistryPtr()</code>
Parametre:	Ingen
Returværdi:	Returner pointer til UnitRegister
Beskrivelse:	Funktionen returner en pointer til UnitRegister

Funktion:	<code>CommInterface* getCommPtr()</code>
Parametre:	Ingen
Returværdi:	Returner pointer til CommInterface
Beskrivelse:	Funktionen returner en pointer til CommInterface

Funktion:	<code>Void on_MainW_StackedWidget_currentChanged(int)</code>
Parametre:	Ingen
Returværdi:	Ingen.
Beskrivelse:	Funktionen tjekker om noget er skifte i tabellerne. Hvis der er sket en ændring, så opdater den tabellen.

### 3.8.11.6 MainMenu

Funktion:	MainMenu(QStackedWidget *parent, UnitRegister& regRef, CommInterface& commRef)
Parametre:	(QStackedWidget *parent, UnitRegister& regRef, CommInterface& commRef)
Returværdi:	Ingen.
Beskrivelse:	Constructor

Funktion:	Void setTablePtr(QTableWidget*)
Parametre:	Pointer til QTableWidget
Returværdi:	Ingen.
Beskrivelse:	Funktionen sætter en pointer til tabellen

Funktion:	QTableWidget* getTablePtr()
Parametre:	Ingen
Returværdi:	Ingen.
Beskrivelse:	Funktionen returner en pointer til tabellen

Funktion:	Void updateFromCommandBox()
Parametre:	Ingen
Returværdi:	Ingen.
Beskrivelse:	Funktionen opdaterer tabellen med status for enheder fra styreBoksen

Funktion:	Void updateFromLocal()
Parametre:	Ingen
Returværdi:	Ingen.
Beskrivelse:	Funktionen opdaterer tabellen med status for enheder

Funktion:	Void on_updateButton_clicked()
Parametre:	Ingen
Returværdi:	Ingen.
Beskrivelse:	Funktionen kalder funktionen updateFromCommandsBox()

Funktion:	Void on_addUnit_PushButton()
Parametre:	Ingen
Returværdi:	Ingen.
Beskrivelse:	Funktionen åbner "Tilføj Enhed"

Funktion:	Void on_remUnit_PushButton_clicked()
Parametre:	Ingen
Returværdi:	Ingen.
Beskrivelse:	Funktionen åbner "Fjern Enhed"

Funktion:	Void on_PushButton_2_clicked()
Parametre:	Ingen
Returværdi:	Ingen.
Beskrivelse:	Funktionen åbner "Tidsplan"

Funktion:	Void on_editUnit_PushButton_clicked()
Parametre:	Ingen
Returværdi:	Ingen.
Beskrivelse:	Funktionen åbner "Rediger Enhed"

### 3.8.11.7 AddUnit

Funktion:	AddUnit(QStackedWidget *parent, UnitRegister& regRef, CommInterface& commRef)
Parametre:	(QStackedWidget *parent, UnitRegister& regRef, CommInterface& commRef)
Returværdi:	Ingen.
Beskrivelse:	Constructor

Funktion	Void on_AddUnit_2_clicked()
Parametre	Ingen
Returværdi	Ingen
Beskrivelse	Funktionen tilføjer enhed til unitRegister

Funktion:	void on_cancel_clicked()
Parametre:	Ingen
Returværdi:	Ingen.
Beskrivelse:	Funktionen afbryder tilføjelse af enheden

### 3.8.11.8 RemoveUnit

Funktion:	RemoveUnit(QStackedWidget *parent, UnitRegister& regRef, CommInterface& commRef)
Parametre:	(QStackedWidget *parent, UnitRegister& regRef, CommInterface& commRef)
Returværdi:	Ingen.
Beskrivelse:	Constructor

Funktion:	void setTablePtr(QTableWidget*)
Parametre:	Pointer til tabellen
Returværdi:	Ingen.
Beskrivelse:	Funktionen sætter en pointer til tabellen. Benyttes for at tilgå tabellen.

Funktion:	<code>QTableWidget* getTablePtr()</code>
Parametre:	Ingen
Returværdi:	Ingen.
Beskrivelse:	Funktionen returner en pointer til tabellen

Funktion:	<code>void populateTable()</code>
Parametre:	Ingen
Returværdi:	Ingen.
Beskrivelse:	Funktionen udfylder tabellen med tilføjede tidsplaner.

Funktion:	<code>Void on_rem_unitTable_cellClicked(int row,int column)</code>
Parametre:	<code>Int row, int column</code>
Returværdi:	Ingen.
Beskrivelse:	Funktionen sætter hvilken celle som er valgt i tabellen

Funktion:	<code>Void on_remove_PushButton_clicked()</code>
Parametre:	Ingen
Returværdi:	Ingen.
Beskrivelse:	Funktionen fjerner valgte Unit

Funktion:	<code>Void on_cancel_PushButton_clicked()</code>
Parametre:	Ingen
Returværdi:	Ingen.
Beskrivelse:	Funktionen afbryder fjernelse af enheden

### 3.8.11.9 EditUnit

Funktion:	<code>EditUnit(QStackedWidget *parent, UnitRegister&amp; regRef, CommInterface&amp; commRef)</code>
Parametre:	<code>(QStackedWidget *parent, UnitRegister&amp; regRef, CommInterface&amp; commRef)</code>
Returværdi:	Ingen.
Beskrivelse:	Constructor

Funktion:	<code>Void setSelectedUnitID(int)</code>
Parametre:	<code>Int</code>
Returværdi:	Ingen.
Beskrivelse:	Funktionen benyttes for at sætte attributten selectedUnitID

Funktion:	<code>Uchar getSelectedUnitID()</code>
Parametre:	Ingen
Returværdi:	<code>Uchar</code>
Beskrivelse:	Funktionen returner hvilket UnitID som er valgt

Funktion:	Void setTablePtr(QTableWidget*)
Parametre:	Pointer til QTableWidget
Returværdi:	Ingen.
Beskrivelse:	Funktionen sætter en pointer til tabellen

Funktion:	QTableWidget* getTablePtr()
Parametre:	Ingen
Returværdi:	Ingen.
Beskrivelse:	Funktionen returner en pointer til tabellen

Funktion:	Void updateTable()
Parametre:	Ingen
Returværdi:	Ingen.
Beskrivelse:	Funktionen opdaterer tabellen med tilføjede enheder

Funktion:	Void onEditUnitCancel_PushButton_clicked()
Parametre:	Ingen
Returværdi:	Ingen.
Beskrivelse:	Funktionen afbryder redigering af enheder

Funktion:	Void onEditUnit_Table_PushButton_clicked(int,int)
Parametre:	Int, int
Returværdi:	Ingen.
Beskrivelse:	Funktionen sætter hvilken cell som er valgt i tabellen

### 3.8.11.10 EditOldEntry

Funktion:	EditOldEntry(QStackedWidget *parent, UnitRegister& regRef, CommInterface& commRef)
Parametre:	(QStackedWidget *parent, UnitRegister& regRef, CommInterface& commRef)
Returværdi:	Ingen.
Beskrivelse:	Constructor

Funktion:	void setTablePtr(QTableWidget*)
Parametre:	Pointer til tabellen
Returværdi:	Ingen.
Beskrivelse:	Funktionen sætter en pointer til tabellen. Benyttes for at tilgå tabellen.

Funktion:	<code>void populateTable()</code>
Parametre:	Ingen
Returværdi:	Ingen.
Beskrivelse:	Funktionen udfylder tabellen med tilføjede tidsplaner.

Funktion:	<code>void getUnit(Int)</code>
Parametre:	Int
Returværdi:	Ingen.
Beskrivelse:	Funktionen er et public slot, som benyttes for at modtage hvilket UnitID, der ønskes arbejdes på.

Funktion:	<code>Void on_EditEntryTable_cellClicked(int row,int column)</code>
Parametre:	Int row, int column
Returværdi:	Ingen.
Beskrivelse:	Funktionen sætter hvilken celle som er valgt i tabellen

Funktion:	<code>Void on_Annullerer_clicked()</code>
Parametre:	Ingen
Returværdi:	Ingen.
Beskrivelse:	Funktionen afbryder redigering af eksisterende tidsplan

Funktion:	<code>Void on_saveEntry_clicked()</code>
Parametre:	Ingen
Returværdi:	Ingen.
Beskrivelse:	Funktionen gemmer ændringer af eksisterende tidsplan

### 3.8.11.11 RemoveEntry

Funktion:	<code>RemoveEntry(QStackedWidget *parent, UnitRegister&amp; regRef, CommInterface&amp; commRef)</code>
Parametre:	<code>(QStackedWidget *parent, UnitRegister&amp; regRef, CommInterface&amp; commRef)</code>
Returværdi:	Ingen.
Beskrivelse:	Constructor

Funktion:	<code>void setTablePtr(QTableWidget*)</code>
Parametre:	Pointer til tabellen
Returværdi:	Ingen.
Beskrivelse:	Funktionen sætter en pointer til tabellen. Benyttes for at tilgå tabellen.

Funktion:	<code>void populateTable()</code>
Parametre:	Ingen
Returværdi:	Ingen.
Beskrivelse:	Funktionen udfylder tabellen med tilføjede tidsplaner.

Funktion:	<code>void getUnit(Int)</code>
Parametre:	Int
Returværdi:	Ingen.
Beskrivelse:	Funktionen er et public slot, som benyttes for at modtage hvilket UnitID, der ønskes arbejdes på.

Funktion:	<code>void on_RemoveEntry_2_clicked()</code>
Parametre:	Ingen
Returværdi:	Ingen.
Beskrivelse:	Funktionen fjerner den valgte tidsplan

Funktion:	<code>Void onEntriesTable_cellClicked(int row,int column)</code>
Parametre:	Int row, int column
Returværdi:	Ingen.
Beskrivelse:	Funktionen sætter hvilken celle som er valgt i tabellen

Funktion:	<code>Void on_Annullerer_clicked()</code>
Parametre:	Ingen
Returværdi:	Ingen.
Beskrivelse:	Funktionen afbryder fjernelse af Tidsplan

Funktion:	<code>Void on_RemoveEntriesForDay_clicked()</code>
Parametre:	Ingen
Returværdi:	Ingen.
Beskrivelse:	Funktionen fjerner alle tidsplan for valgte dag

Funktion:	<code>Void onEditUnit_Save_PushButton_clicked()</code>
Parametre:	Ingen
Returværdi:	Ingen.
Beskrivelse:	Funktionen gemmer ændringer af eksisterende enhed

### 3.8.11.12 AddEntry

Funktion:	AddEntry(QStackedWidget *parent, UnitRegister& regRef, CommInterface& commRef)
Parametre:	(QStackedWidget *parent, UnitRegister& regRef, CommInterface& commRef)
Returværdi:	Ingen.
Beskrivelse:	Constructor

Funktion:	void setTablePtr(QTableWidget*)
Parametre:	Pointer til tabellen
Returværdi:	Ingen.
Beskrivelse:	Funktionen sætter en pointer til tabellen. Benyttes for at tilgå tabellen.

Funktion:	void populateTable()
Parametre:	Ingen
Returværdi:	Ingen.
Beskrivelse:	Funktionen udfylder tabellen med tilføjede tidsplaner.

Funktion:	void getUnit(Int)
Parametre:	Int
Returværdi:	Ingen.
Beskrivelse:	Funktionen er et public slot, som benyttes for at modtage hvilket UnitID, der ønskes arbejdes på.

Funktion:	void on_Cancel_clicked()
Parametre:	Ingen
Returværdi:	Ingen.
Beskrivelse:	Funktionen afbryder tilføjelse af tidsplan til enheden

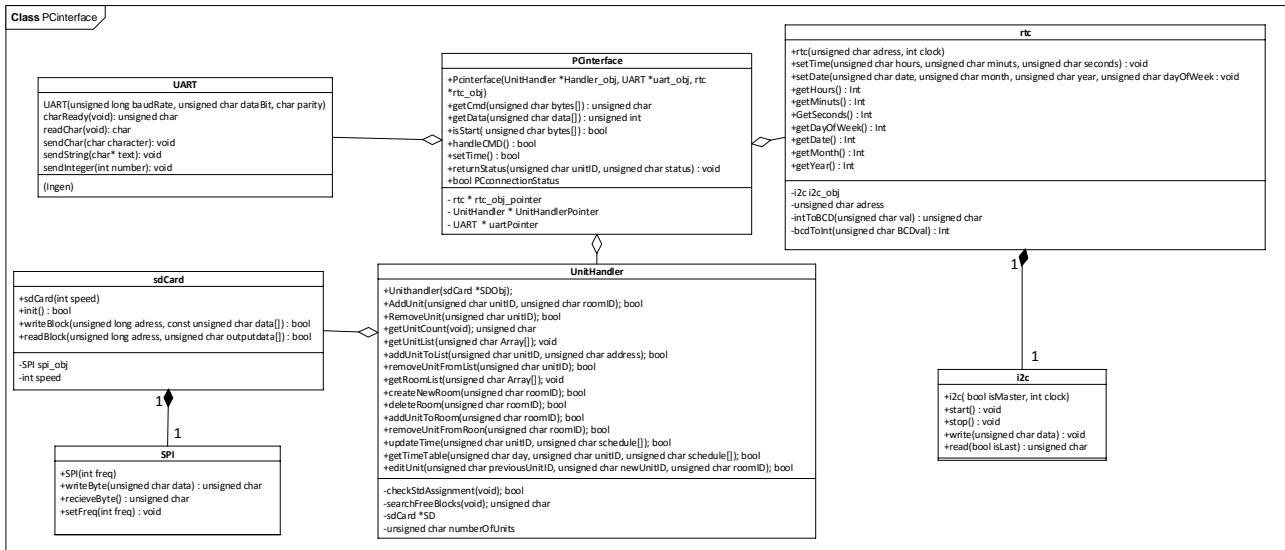
Funktion:	void on_AddTheEntry_clicked()
Parametre:	Ingen
Returværdi:	Ingen.
Beskrivelse:	Funktionen tilføjer tidsplan til enheden

## 3.9 Design og Implementation – Styreboks (TF, SN, DP)

Herunder findes beskrivelser af design og implementation af styreboksen software. Softwaren er delt op i tre underemner, PC Interface, Real Time Clock, SD-Kort Driver og Unit Handler.

### 3.9.1 PC Interface (TF)

Til kommunikationen mellem PC og styreboksen anvendes der en interrupt baseret uart kommunikation der derefter behandles af PC interface klassen, og ud fra den modtagne uart kommunikation udføres der forskellige handlinger på styreboksen. Denne kommunikation er kernen til at konfigurere systemet.



Figur 61 – PC interface UML klassediagram

Som det ses på Figur 61 anvender PC interface klassen en stor del af de tidligere beskrevne klasser, dette skyldes at det skal være muligt at håndtere hele konfigurationen af styreboksen fra den grafiske brugerflade på pc'en. For at muliggøre dette er PC interface klassen bindeleddet mellem styreboksens moduler og pc'en, dette bevirket at PC interface klassen håndtere en del funktionalitet som tidligere er blevet beskrevet som separate klasser.

Kommunikationen følger PC interface protokollen, og implementere denne i styreboksen via den indbyggede uart i atmega2560.

I Tabel 2 beskrives PC interface klassens funktioner.

Funktion	Beskrivelse
<b>PCinterface(UnitHandler *Handler_obj, UART *uart_obj, rtc *rtc_obj)</b>	Klassens constructor, skal modtager pointers til et UnitHandler objekt, et UART objekt samt et rtc objekt.
<b>getCmd(unsigned char bytes[])</b>	Henter en kommando fra uart og gemmer den i bytes[]
<b>getData(unsigned char data[])</b>	Henter den data fra uart der følger efter den modtagne kommando og gemmer det i data[].
<b>isStart (unsigned char bytes[] )</b>	Henter 2 bytes på uart, gemmer dem i bytes[], og returnere true, hvis de to bytes matcher start sekvensen på en kommunikation fra pcen.
<b>handleCMD()</b>	HandleCMD er den funktion der håndtere samtlige kommandoer fra pcen, og udføre de nødvendige handlinger ud fra en switch case i funktionen.
<b>setTime()</b>	Anvendes til at sætte tiden på RTC når tiden modtages fra PCen.
<b>returnStatus(unsigned char unitID, unsigned char status)</b>	Sender unit ID og den modtagne status til PC i forbindelse med statusrequests.
<b>Bool PCconnectionStatus</b>	Anvendes til at kunne udlæse om der er en PC tilsluttet eller ej i forbindelse med skift mellem simulering og konfigurations states.
<b>Rtc_obj_pointer</b>	Pointer til real time clock objektet.
<b>UnitHandlerPointer</b>	Pointer til UnitHandler Objektet
<b>uartPointer</b>	Pointer til Uart driver objekt.

Tabel 2- funktionsbeskrivelser for PC Interface klassen

### 3.9.2 UART Driver (DP)

**Ansvar:** Kommunikation mellem styreboks og PC

#### 3.9.2.1 Metoder:

Funktion:	UART( unsigned long baudRate = 9600, unsigned char dataBit = 8, char parity = 'N' )
Parametre:	Den ønskede baudrate, de ønskede antal databit, den ønskede paritet.
Returværdi:	Ingen.
Beskrivelse:	Constructor for klassen UART. Mulighed for indstilling af baudrate mellem 110 og 155200, default er sat til 9600 bps. Mulighed for indstilling af antal databit mellem 5 og 8, default er sat til 8. Mulighed for indstilling af paritet enten even, odd eller ingen, default er sat til ingen.

Funktion:	unsigned char charReady( void )
Parametre:	Den ønskede baudrate, de ønskede antal databit, den ønskede paritet.
Returværdi:	Returnerer 0, hvis der ikke er modtaget en ny character.
Beskrivelse:	Returnerer 0, hvis der ikke er modtaget en ny character.

Funktion:	char readChar( void )
Parametre:	Den ønskede baudrate, de ønskede antal databit, den ønskede paritet.
Returværdi:	Returner character når denne er modtaget
Beskrivelse:	Returner character når denne er modtaget

Funktion:	void sendChar( char character )
Parametre:	Den character der ønskes sendt
Returværdi:	
Beskrivelse:	Vent på at sender registeret er klar. Send en character afsted

Funktion:	void sendString( char* textPtr)
Parametre:	Den linje tekst, der ønskes sendt
Returværdi:	
Beskrivelse:	Send linje en character ad gangen indtil "/0" (sidste plads) nås.

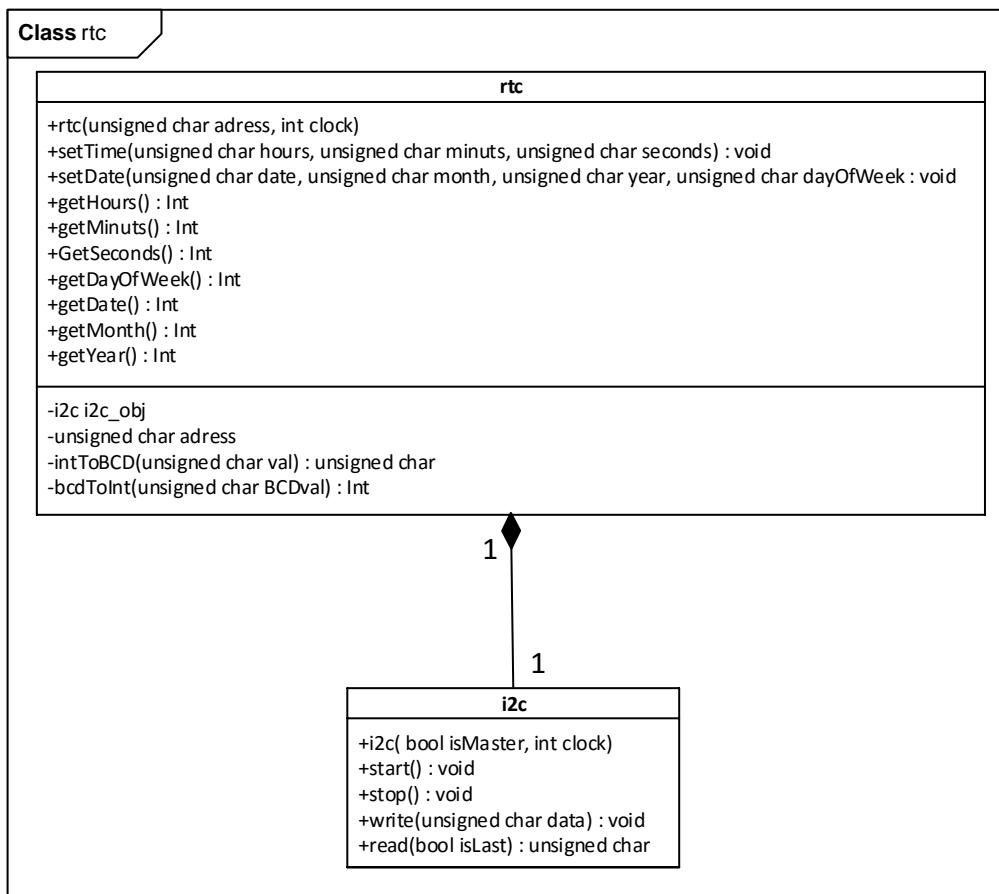
Funktion:	void sendInteger( int number)
Parametre:	Den integer der ønskes sendt
Returværdi:	
Beskrivelse:	Konverte interger til ASCII-array, og send dette.

### 3.9.3 Real-Time Clock (TN)

Der er i forbindelse med projektet behov for at styreboksen kan udføre handlinger ud fra en given tidsplan, for at dette er muligt er det nødvendigt for styreboksen at have en real time clock der sikre at tiden kan udlæses præcist så handlinger kan udføres på de korrekte tidspunkter.

I dette projekts tilfælde anvenders der et ds3231 modul, hvor der kommunikeres direkte med ds3231 fra vores atmega2560 via I2C/TWI protokollen.

Ud fra ds3231 chippens datasheet ses det at ds3231 understøtter I2C med en clock på 100 kHz samt 400 kHz. Der anvendes her i projektet en hastighed på 400 kHz for at mindske forsinkelsen mellem den reelle tid og den udlæste tid.



Figur 62 - UML klassediagram for rtc klassen

### 3.9.3.1 I2C driver klasse

Klassen i2c anvendes her som kommunikationsvej mellem vores rtc klasse og ds3231 modul. Denne implementeres via det indbyggede TWI interface i vores atmega2560.

I Tabel 3Tabel 2- funktionsbeskrivelser for PC Interface klassen beskrives i2c klassens funktioner.

Funktion	Beskrivelse
<b>i2c(bool isMaster, int clock)</b>	Dette er klassens constructor, isMaster sættes til true når i2c anvendes i master mode som er tilfældet her i projektet. Clock sættes til den ønskede frekvens, hvilket kan være enten 100 kHz eller 400 kHz.
<b>start()</b>	Sender i2c start sekvensen som beskrevet i atmega2560 samt ds3231 datasheet. Skal sendes inden der kan skrives eller læses fra i2c.
<b>stop()</b>	Sender i2c stop sekvensen som beskrevet i atmega2560 samt ds3231 datasheets, Skal sendes når en i2c transmission afsluttes.
<b>write(unsigned char data)</b>	Skriver 1 byte ( 8 bit ) af data på i2c bussen.
<b>read(bool isLast)</b>	Læse TWI bufferen på atmega2560 og returnere 1 byte ( 8 bit ) data fra i2c bussen.

Tabel 3- funktionsbeskrivelser for i2c klassen

### 3.9.3.2 Rtc klassen

Rtc klassen har et i2c objekt der benyttes til selve kommunikationen mellem ds3231 og atmega2560. Dette foregår med den i ds3231 datasheetet angivede 7 bits i2c adresse 1101000 efterfuldt af en retningsbit der er 0 for write, 1 for at læse. Adresseringen af hukommelsesblokkene til at indstille tiden samt udlæse den nuværende tid ser ud som på Figur 63.

ADDRESS	BIT 7 MSB	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0 LSB	FUNCTION	RANGE			
00H	0	10 Seconds			Seconds				Seconds	00–59			
01H	0	10 Minutes			Minutes				Minutes	00–59			
02H	0	12/24	AM/PM	10 Hour	Hour				Hours	1–12 + AM/PM 00–23			
			10 Hour										
03H	0	0	0	0	0	Day			Day	1–7			
04H	0	0	10 Date		Date				Date	00–31			
05H	Century	0	0	10 Month	Month				Month/ Century	01–12 + Century			
06H	10 Year				Year				Year	00–99			
07H	A1M1	10 Seconds			Seconds				Alarm 1 Seconds	00–59			
08H	A1M2	10 Minutes			Minutes				Alarm 1 Minutes	00–59			
09H	A1M3	12/24	AM/PM	10 Hour	Hour				Alarm 1 Hours	1–12 + AM/PM 00–23			
			10 Hour										
0AH	A1M4	DY/DT	10 Date		Day			Alarm 1 Day	1–7				
					Date			Alarm 1 Date	1–31				
0BH	A2M2	10 Minutes			Minutes				Alarm 2 Minutes	00–59			
0CH	A2M3	12/24	AM/PM	10 Hour	Hour				Alarm 2 Hours	1–12 + AM/PM 00–23			
			10 Hour										
0DH	A2M4	DY/DT	10 Date		Day			Alarm 2 Day	1–7				
					Date			Alarm 2 Date	1–31				
0EH	EOSC	BBSQW	CONV	RS2	RS1	INTCN	A2IE	A1IE	Control	—			
0FH	OSF	0	0	0	EN32kHz	BSY	A2F	A1F	Control/Status	—			
10H	SIGN	DATA	DATA	DATA	DATA	DATA	DATA	DATA	Aging Offset	—			
11H	SIGN	DATA	DATA	DATA	DATA	DATA	DATA	DATA	MSB of Temp	—			
12H	DATA	DATA	0	0	0	0	0	0	LSB of Temp	—			

**Note:** Unless otherwise specified, the registers' state is not defined when power is first applied.

Figur 63 - Tabel der viser adresse og registre på ds3231, taget fra datasheet for ds3231.

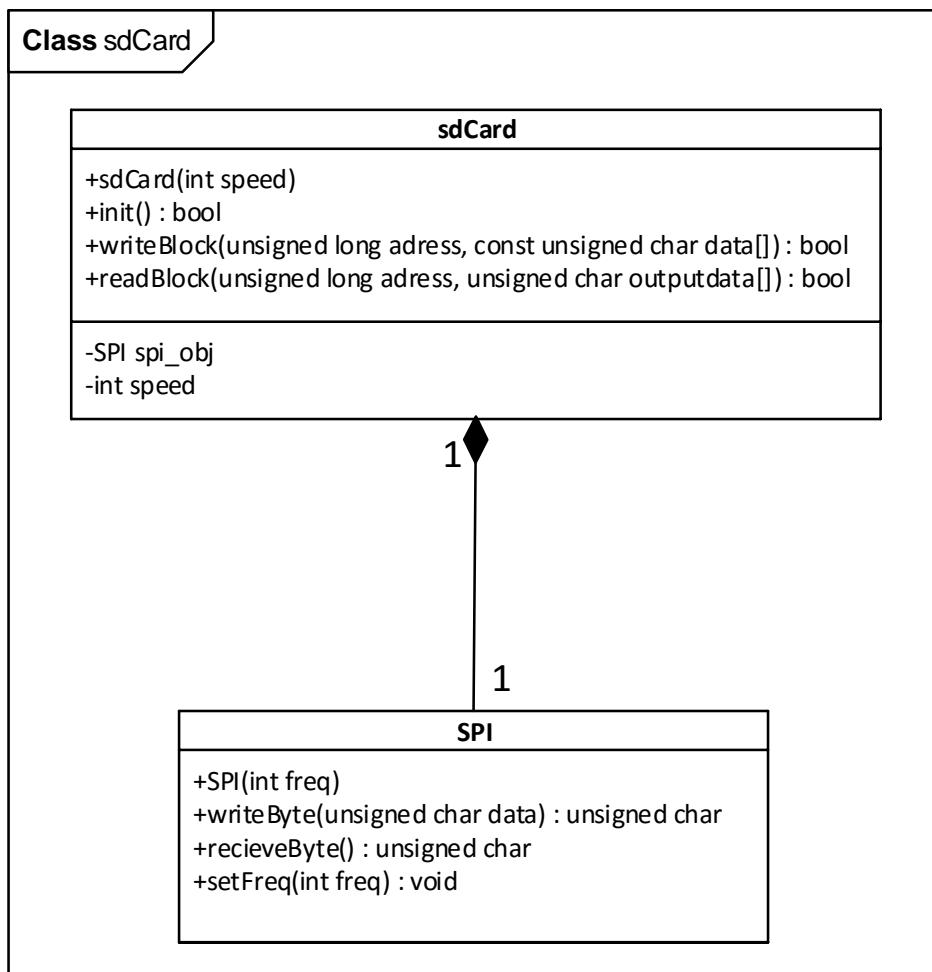
Som det ses på figur 2 anvender ds3231 et andet format end standard integers til at gemme tiden, dette format kaldes for Binary Coded Decimal og anvender de 4 most significant bits til at representere 10ere og de 4 laveste til at representere 1ere af det samlede tal, i nogle tilfælde anvendes der ikke alle 4 bit, hvilket der selvfølgelig tages højde for i driveren. Til at håndtere konvertering mellem BCD og integers er der 2 funktioner i rtc klassen som bruges internt, se tabelreference for yderligere info omkring denne konvertering. Når der skrives til registret på adresse 00H aktiveres rtc og denne vil begynde at køre.

For yderligere info om DS3231 se datasheet der er vedlagt som bilag.

Funktion/attribut	Beskrivelse
<b>rtc(unsigned char adress, int clock)</b>	Constructor der initiere et nyt objekt af typen rtc, tager en 8 bit adresse ( 7 bit + direktion bit sat til 0) samt den ønskede i2c clockfrekvens.
<b>setTime(unsigned char hours, unsigned char minuts, unsigned char seconds)</b>	Funktion til at indstille tiden på ds3231, den anvender hjælpefunktionen intToBCD til at konvertere de 8 bit intergers den modtager til BCD som ds3231 kan benytte.
<b>setData(unsigned char data, unsigned char month, unsigned char year, unsigned char dayOfWeek)</b>	Funktion til at indstille dato samt ugedag på ds3231, denne anvender hjælpefunktionen intToBCD for at konvertere de modtagne parametre til BCD inden det sendes til ds3231.
<b>getHours()</b>	Returnere det nuværende time antal fra ds3231.
<b>getMinuts()</b>	Returnere det nuværende minut antal fra ds3231.
<b>GetSeconds()</b>	Returnere det nuværende antal sekunder fra ds3231.
<b>getDayOfWeek()</b>	Returnere den nuværende ugedag (1 – 7) fra ds3231.
<b>getDate()</b>	Returnere den nuværende dato fra ds3231.
<b>getMonth()</b>	Returnere den nuværende måned (0-12) fra ds3231.
<b>getYear</b>	Returnere det nuværende årstal (00-99) fra ds3231.
<b>intToBCD(unsigned char val)</b>	Tager en 8 bit integer og returnere den tilsvarende BCD værdi
<b>bcdToInt(unsigned char BCDval)</b>	Tager en BCD værdi og returnere en integer svarende til denne værdi.
<b>I2c i2c_obj</b>	I2c driver objekt der håndtere i2c kommunikationen.
<b>Unsigned char adress</b>	Benyttes til at gemme adressen på ds3231.

### 3.9.4 SD-Kort (TN)

I forbindelse med projektet er der behov for en permanent lagring af data som også kan bevare de gemte data ved eksempelvis en strømafbrydelse. Til dette anvendes der et kingston 8 GB sdhc kort der anvendes i SPI mode, kortet følger sd simplified specification for sdhc kort, der er vedlagt som bilag, hvilket gør det muligt at skrive en fornuftig driver til at håndtere kortet, driveren her vil virke med alle sdhc kort der følger sd simplified specification. UML klassediagrammet for sdCard klassen ses på Figur 61, hvor det også ses at der anvendes et SPI objekt til kommunikation med sd kortet, begge klasser beskrives i de efterfølgende sektioner.



Figur 64 - sdCard UML klassediagram

### 3.9.4.1 SPI driver klasse

Klassen SPI anvendes her som kommunikationsvej mellem vores sdCard klasse og vores 8gb SDHC kort. Denne implementeres via det indbyggede SPI interface i vores atmega2560, hvor atmega2560 anvendes i master mode.

I Tabel 4 beskrives SPI klassens funktioner.

Funktion	Beskrivelse
<b>SPI(int freq)</b>	Klassens konstructor, freq er den frekvens angivet i kHz der ønskes anvendt som clock frekvens på spi modulet. Der kan vælges mellem følgende frekvenser: 125 kHz 250 kHz 1000 kHz 4000 kHz
<b>writeByte(unsigned char data)</b>	Skriver 1 byte (8 bit) data til SPI bussen på MOSI porten.
<b>receiveByte()</b>	Læse 1 byte (8 bit) data fra SPI bussen på MISO porten.
<b>setFreq(int freq)</b>	Ændre clockfrekvensen på SPI bussen til den ønskede frekvens, angives i kHz og kan være en af følgende frekvenser: 125 kHz, 250 kHz, 1000 kHz, 4000 kHz.

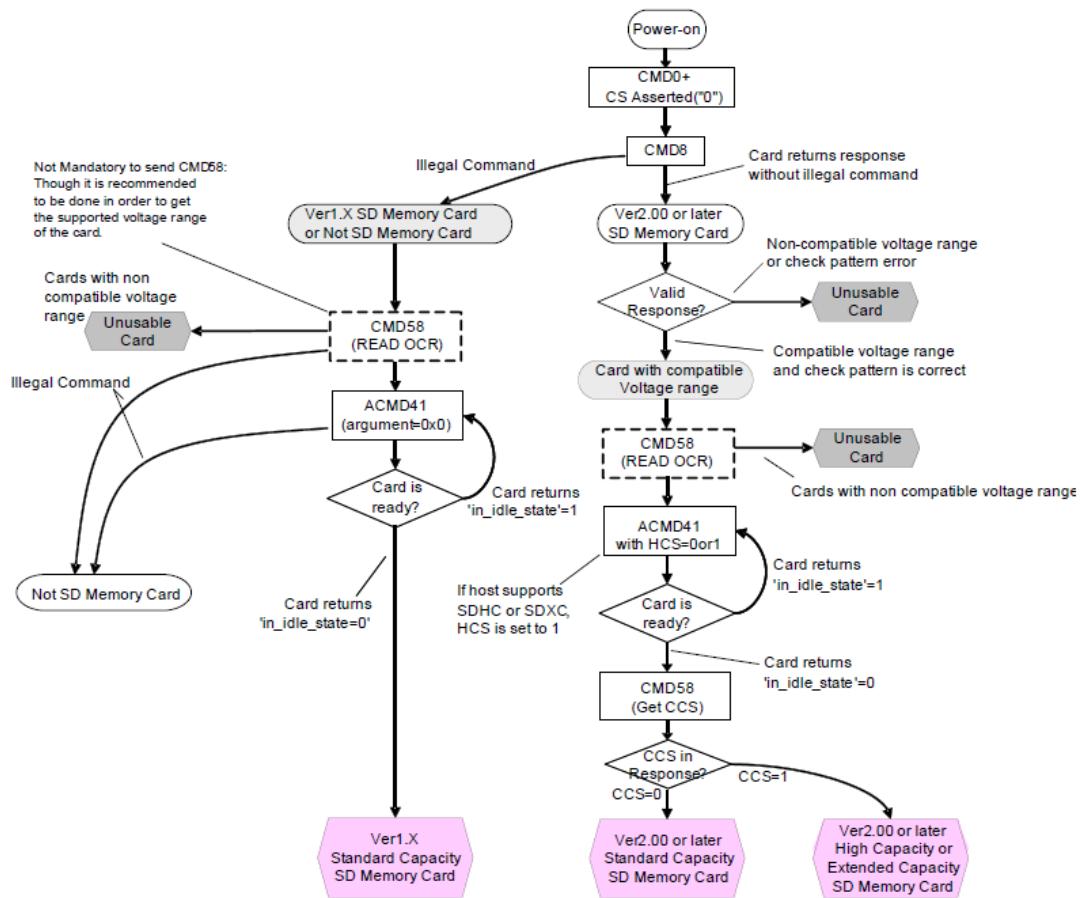
Tabel 4- funktionsbeskrivelser for SPI klassen

### 3.9.4.2 sdCard klassen

sdCard klassen følger som nævnt sd simplified specifikation som er vedlagt som bilag. Der er i nogle sammenhænge set bort fra dele af specifikationen der vedrøre ældre sd-kort samt mmc kort, da vi i projektet kun anvender sd-kort af typen SDHC.

Initialiseringsprocessen for et sd-kort er rimelig kompliceret, og kræve en længere sekvens af kommandoer der skal sendes til sd-kortet og kan ses på Figur 65. Der er udeladt alle dele af initieringsprocessen der ikke vedrøre kort af typen SDHC.

Det er ifølge specifikationen nødvendigt at starte med en SPI clock frekvens på under 400 kHz for samt at der sendes mindst 70 clock cycles inden initieringssekvensen startes. Når initieringsprocessen er udført og kortet er klar til brug kan frekvensen hæves til den ønskede clock frekvens for at øge datatransmissionshastigheden.



Figur 65 - Initieringssekvens for sd-kort, udklip fra sd simplified specifikation.

I denne version af sdCard driveren anvendes initieringsprocessen for V2.00 eller senere high capacity kort, er kortet ikke af denne type vil initieringsprocessen fejle.

Ud over den specielle initieringsproces, skal man herudover være opmærksom på at sd-kort af denne type kun kan læse og skrive en hel block af gangen. En block på et sd-kort er 512 bytes (1 byte = 8 bit), hvorfor sd-kortet adresseres via blocklocationer, dvs. adressen 0 = første 512 bytes på kortet, adressen 1, er de næste 512 bytes på kortet.

Funktion/attribut	Beskrivelse
<b>sdCard(int speed)</b>	Constructor der initiere et nyt objekt af typen sdCard, speed angives til den ønskede frekvens i kHz der ønskes når initialiseringsprocessen er udført. Speed kan være en af følgende frekvenser: 125 kHz 250 kHz 1000 kHz 4000 kHz
<b>Init()</b>	Udføre initialiseringsprocessen af sd-kortet, returner true hvis initialiseringsprocessen lykkedes, false hvis den fejler.
<b>writeBlock(unsigned long adress, const unsigned char data[])</b>	Modtager en 32 bits adresse samt et 512 bytes array, og skriver det 512 bytes array til den block der ligger på den angivne adresse på sd-kortet. Returnerer true hvis det lykkedes at skrive på sd-kortet, false hvis der opstår en fejl.
<b>readBlock(unsigned long adress, unsigned char data[])</b>	Tager en 32 bits adresse samt et 512 bytes array og henter den block på sd-kortet der ligge på den 32 bits adresse der modtages. Denne block fyldes nu i det 512 byte array der er modtaget som parameter. Lykkedes det ikke at læse blokken returneres der false.
<b>SPI spi_obj</b>	SPI objekt til at håndtere selve SPI kommunikationen.
<b>speed</b>	Speed attributten indeholder den ønskede hastighed der blev valgt ved oprettelsen af objektet.

Tabel 5 - funktionsbeskrivelser for sdCard klassen.

Angives der adresser der ligger uden for det antal blokke der er tilgængeligt på sd-kortet vil funktionerne returnere false, hvorved der kan reageres på de opståede fejl.

Ønskes der yderligere informationer om sd-kort via SPI protokollen henvises der til sd simplified specifikation der er vedlagt som bilag.

### 3.9.5 Unit Handler (SN)

**Ansvar:** Skal håndterer (oprette, redigerer og slette) alle enheder i systemet.

Denne klasse implementerer en række funktioner som gør det muligt at systemet gemmer oplysninger om enheder, rum og tidsplaner på det SD-kort, som er en del af vores styreboks.

Funktionerne i denne klasse sørger også for at allokeringen af datablokke på SD-kortet sker på en struktureret måde, som gør det muligt nemt at hente ønskede data fra SD-kortet.

#### 3.9.5.1 Allokering af data-blokke på SD-kort.

Allokeringen af blokke på SD-kortet sker ud fra følgende regler:

1: de første 2 blokke på SD-kortet (blok 0 og 1) bliver brugt til henholdsvis enheds-liste, og rum-liste.

2: Hver enhed tilføjet til systemet får allokeret 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)

UnitHandler
+Unithandler(sdCard *SDObj); +AddUnit(unsigned char unitID, unsigned char roomID); bool +RemoveUnit(unsigned char unitID); bool +getUnitCount(void); unsigned char +getUnitList(unsigned char Array[]); void +addUnitToList(unsigned char unitID, unsigned char address); bool +removeUnitFromList(unsigned char unitID); bool +getRoomList(unsigned char Array[]); void +createNewRoom(unsigned char roomID); bool +deleteRoom(unsigned char roomID); bool +addUnitToRoom(unsigned char roomID); bool +removeUnitFromRoom(unsigned char roomID); bool +updateTime(unsigned char unitID, unsigned char schedule[]); bool +getTimeTable(unsigned char day, unsigned char unitID, unsigned char schedule[]); bool +editUnit(unsigned char previousUnitID, unsigned char newUnitID, unsigned char roomID); bool
-checkStdAssignment(void); bool -searchFreeBlocks(void); unsigned char -sdCard *SD -unsigned char numberOfUnits

### 3.9.6 Funktioner

Funktion:	UnitHandler(sdCard *SDObj)
Parametre:	Pointer til et sdCard objekt.
Returværdi:	Ingen.
Beskrivelse:	Constructor for klassen, skal bruge en pointer til et sd-Objekt ved oprettelsen.

Funktion:	bool AddUnit(unsigned char unitID, unsigned char roomID 0x00)
Parametre:	unitID som er den unikke identifier for den enhed der forsøges oprettet. roomID, som er muligheden for at gøre enheden til en del af en gruppe (rum), standart-værdi som er intet rum = 0x00
Returværdi:	Boolsk.
Beskrivelse:	AddUnit-funktionen kalder først funktionen checkStdAssignment(); for at se om vi kan anvende vores standart tildeling af blokke på sd-kortet. Hvis checkStdAssignment(); returnerer false, vil funktionen kalde endnu en funktion: searchFreeBlocks(); som søger efter frie blokke på sd-kortet, som kan anvendes til oprettelsen. Når start-blokken er lokaliseret, allokeres 7 blokke på sd-kortet til den oprettede enhed, og der gemmes på hver blok tidsplanen for den givne dag i ugen.

Funktion:	bool RemoveUnit(unsigned char unitID)
Parametre:	unitID som er den unikke identifier for den enhed der skal slettes.
Returværdi:	Boolsk.
Beskrivelse:	Funktionen henter først unit-listen, og gennemsøger denne efter et matchende unitID, og henter her fra den værdi som benyttes til at beregne start blokken for den enhed. Funktionen tjekker så om enheden er tildelt et rum, eller standart værdien 0x00, hvis enheden er del af en rum, vil funktionen først trække et fra antallet af enheder i det rum, og vil så herefter over skrive de 7 blokke, som før repræsenterede enheden med 0x00, og kalder til sidst funktionen removeUnitFromList(), for at alle data på SD-kortet som repræsenterede den enhed er slettet.

Funktion:	unsigned char getUnitCount()
Parametre:	Ingen.
Returværdi:	unsigned char .... Retunerer numberOfUnits.
Beskrivelse:	Retunerer variablen numberofUnits, som i klassen bruges til at holde styr på antallet af gemte enheder.

Funktion:	void getUnitList(unsigned char Array[])
Parametre:	Unsigned char Array[]
Returværdi:	Ingen.
Beskrivelse:	Fylder det array man giver funktionen som parameter med enhedslisten fra blok 0 på SD-kortet.

Funktion:	bool addUnitToList(unsigned char unitID, unsigned char address)
Parametre:	unitID som er den unikke identifier for den enhed der skal tilføjes listen. Address som er den værdi vi bruger til at beregne start-blokken for enheden på sd-kortet.
Returværdi:	Boolsk.
Beskrivelse:	Funktionen henter først enhedslisten, og gennemsøger denne efter første ledige plads. Hvis ikke der er flere ledige pladser på listen, vil funktionen returnerer false, ellers vil den skrive unitID på første ledige plads, og address på pladsen lige efter, og returnerer true hvis det lykkedes at skrive til sd-kortet, eller false hvis det fejler.

Funktion:	bool removeUnitFromList(unsigned char unitID)
Parametre:	unitID som er den unikke identifier for den enhed der skal fjernes fra listen.
Returværdi:	Boolsk.
Beskrivelse:	Funktionen kalder først getUnitList(), og gennemsøger så dette array for placeringen af den specificerede enhed. Når placeringen er fundet, oprettes et nyt array, hvor alle pladser indtil placeringen af enheden kopieres lige over fra det første array, og alle pladser efter den specificerede enhed shiftes 2 pladser. Til sidst skrives det nye array til plads 0 på SD-kortet.

Funktion:	void getRoomList(unsigned char Array[])
Parametre:	Unsigned char Array[]
Returværdi:	Ingen.
Beskrivelse:	Fylder det array man giver funktionen med rum-listen fra blok 1 på SD-kortet.

Funktion:	bool createNewRoom(unsigned char RoomID)
Parametre:	Unsigned char RoomID som er identifier for det rum der skal oprettes.
Returværdi:	Boolsk.
Beskrivelse:	Funktionen kalder først getRoomList(), og gennemsøger så det fyldte array for tomme pladser. Hvis der ikke er flere tomme pladser returneres false, ellers skrives roomID på første ledige plads, og det opdaterede array bliver skrevet til blok 1 på SD-kortet.

Funktion:	bool deleteRoom(unsigned char RoomID)
Parametre:	Unsigned char RoomID, som er unik identifier for det rum som ønskes slettet.
Returværdi:	Boolsk.
Beskrivelse:	<p>Funktionen starter med at kalde getRoomList(), og gennemsøger så det udfyldte array for det rum der ønskes slettet, og hiver informationer ud fra arrayet om hvor mange enheder der på det givne tidspunkt er tilføjet rummet, og gemmer den info lokalt (i samme scope).</p> <p>Når placeringen er fundet, oprettes et nyt array, hvor alle pladser indtil placeringen af det specificerede rum kopieres lige over fra det første array, og alle pladser efter det specificerede rum shiftes 2 pladser, hvorefter det nye array skrives til plads 1 på SD-kortet.</p> <p>Til sidst gennemsøges startblokke i SD-kortet (efter lister, hver syvende blok) efter enheder som er tildelt til det rum som ønskes slettet.</p> <p>Når en enhed som er en del af det rum bliver fundet, kaldes funktionen removeUnitFromRoom(), og disse enheder fjernes fra rummet. Alt dette sker i et loop, indtil der er fjernet lige så mange enheder som ved funktionskaldet var en del af det rum vi ønskede at slette.</p>

Funktion:	bool addUnitToRoom(unsigned char roomID)
Parametre:	Unsigned char roomID, som er unik identifier for det rum hvor der ønsked tilføjet en enhed.
Returværdi:	Boolsk.
Beskrivelse:	Funktionen kalder først getRoomList(), og gennemsøger så det fyldte array for pladsen for det specificerede rum. Når placeringen er fundet lægges 1 til antallet for dette rum, og det opdaterede array skrives til sidst til SD-kortet.

Funktion:	bool removeUnitFromRoom(unsigned char unitID, unsigned char roomID)
Parametre:	unitID som er den unikke identifier for den enhed der skal fjernes fra rummet specificeret. roomID som er den unikke identifier for det rum der skal opdateres.
Returværdi:	Boolsk.
Beskrivelse:	Først kaldes getRoomList(), og det fyldte array gennemsøges efter pladsen for det specificerede rum. Hvis rummet findes trækkes én fra antallet tilhørende dette rum, og den opdaterede liste skrives til SD-kortet. Hvis ikke rummet findes i listen, går funktionen blot videre til næste punkt. Herefter kaldes getUnitList(), og det fyldte array gennemsøges efter den specificerede enhed. Når enheden er fundet i enhedslisten beregnes start-blokken for enheden, og alle 7 blokke der repræsenterer enheden hentes ned og rettes til rum 0x00, og skrives igen til SD-kortet.

Funktion:	bool UpdateTime(unsigned char unitID, unsigned char schedule[])
Parametre:	<p>Unsigned char unitID, som er den unikke identifier for enheden der skal opdateres.</p> <p>unsigned char schedule[], som er det array med hele tidsplanen for en dag der modtages fra PC.</p>
Returværdi:	Boolsk.
Beskrivelse:	Funktionen kalder først getUnitList(), og gennemsøger det fyldte array for placeringen af den specificerede enhed, og gemmer lokalt værdien der bruges til at beregne start-blokken for enheden. Funktionen tjekker så det array som er modtaget som parameter, for hvilken dag denne repræsenterer for enheden, og benytter så disse oplysninger til at beregne hvilken blok der skal opdateres. funktionen opdaterer så det modtagne array (schedule) med start blok repræsentationen, og skriver til sidst den nye opdaterede tidsplan til SD-kortet.

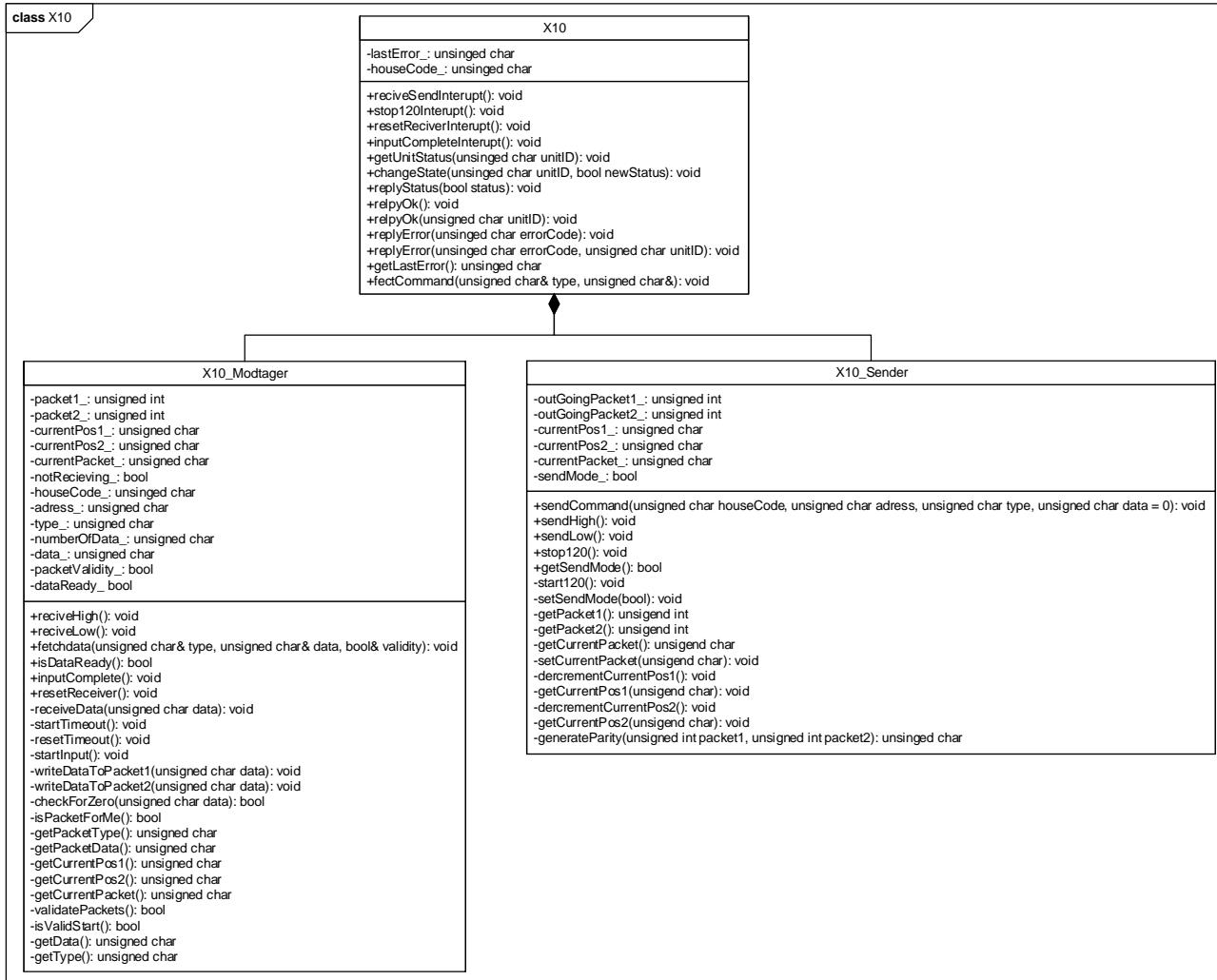
Funktion:	bool getTimeTable(unsigned char day, unsigned char UnitID, unsigned char schedule[])
Parametre:	<p>Unsigned char day, fortæller hvilken dag vi ønsker at få tidsplanen for.</p> <p>Unsigned char UnitID, som er den unikke identifier for den enhed som ønskes en tidsplan for.</p> <p>Unsigned char schedule[], som er det array der skal fyldes med tidsplanen for den givne dag.</p>
Returværdi:	Boolsk.
Beskrivelse:	Funktionen kalder først getUnitList(), og gennemsøger denne efter den specificerede enhed. Når denne er fundet, bruges parameteren "day" sammen med denne værdi til at beregne hvilken blok vi skal hente fra SD-kortet. Til sidst læses den beregnede blok ind i schedule-arrayet som funktionen modtog som parameter.

Funktion:	bool editUnit(unsigned char previousUnitID, unsigned char newUnitID, unsigned char roomID)
Parametre:	<p>Unsigned char previousUnitID, som er det unitID som ved funktionskaldet er gemt i styreboksen.</p> <p>Unsigned char newUnitID, som er det nye unitID vi ønsker gemt.</p> <p>Unsigned char roomID, som er det nye roomID vi ønsker enheden tilføjet (standart er 0x00).</p>
Returværdi:	Boolsk.
Beskrivelse:	<p>Funktionen kalder først getUnitList(), og gennemsøger det fylde array efter "previousUnitID", og dennes start-blok position, som gemmes i en lokal variable.</p> <p>Denne enheds unikke identifier opdateres så til "newUnitID" og den opdaterede liste skrives igen til SD-kortet.</p> <p>Nu læses den første blok for enheden fra SD-kortet, for at finde ud af hvilket rum denne enhed i forvejen er tilføjet, og dette gemmes i en lokal variable.</p> <p>Så kaldes getRoomList(), og det fylde array gennemsøges efter et roomID som matcher det roomID som enheden i forvejen havde, og her trækkes én fra antallet af enheder i det rum.</p> <p>Funktionen gennemsøger nu rumlisten for at se om det rum som enheden skal tilføjes i forvejen eksisterer. Hvis rummet i forvejen eksisterer lægges en til antallet af enheder i rummet, og den opdaterede liste skrives til SD-kortet. Hvis ikke rummet i forvejen eksisterede, vil funktionen oprette rummet med createNewRoom() og herefter tilføje en enhed til det rum med addUnitToRoom().</p> <p>Til sidst er et loop, hvor de 7 blokke som repræsenterer enheden bliver hentet fra SD-kortet en ad gangen, får opdateret unitID og roomID og igen bliver skrevet til SD-kortet.</p>

Funktion:	unsigned char searchFreeBlocks()
Parametre:	Ingen.
Returværdi:	Unsigned char.
Beskrivelse:	Funktionen kalder først getUnitList(), og gennemsøger så det fylde array efter den "numberOfUnits" som er blevet slettet. Når en værdi er fundet vil den blive returneret.

Funktion:	bool checkStdAssignment()
Parametre:	Ingen.
Returværdi:	Boolsk.
Beskrivelse:	Funktionen kalder først getUnitList(), og gennemsøger så det fylde array efter en værdi som er lig numberOfUnits. Hvis denne værdi findes, vil funktionen returnerer false, da standart tildelingen så allerede er i brug.

### 3.10 Design og Implementering – X10.1 Sender/Modtager (CBJ)



#### 3.10.1 X10.1

**Ansvar:** Kommunikation mellem styrebox og Enhed

##### X10 ( unsigned char houseCode, unsigned char adress )

<b>Parametre:</b>	Den house code der skal kommunikeres på, adressen der kan modtages data på
<b>Rerurværdi:</b>	Ingen
<b>Beskrivelse:</b>	Construktur for klassen X10. Der skal indtastes en house code der skal kommunikeres på og en adresse der kan modtages data på

##### void receiveSendInterrupt ( bool highLow )

<b>Parametre:</b>	Den værdi der modtages true for high og false for low
<b>Rerurværdi:</b>	Ingen
<b>Beskrivelse:</b>	Metode kald til interrupt rutine. Placeres i de to interrupt rutine, som sidder på zerocrossen, en med true og en med false

**void stop120Interrupt ( void )**

<b>Parametre:</b>	Ingen
<b>Rerurværdi:</b>	Ingen
<b>Beskrivelse:</b>	Metode kald til interupt rutine. Stopper senderens 120kHz PWM

**void resetReceiverInterrupt ( void )**

<b>Parametre:</b>	Ingen
<b>Rerurværdi:</b>	Ingen
<b>Beskrivelse:</b>	Metode kald til interupt rutine. Resetter modtageren.

**void inputCompleteInterrupt ( void )**

<b>Parametre:</b>	Ingen
<b>Rerurværdi:</b>	Ingen
<b>Beskrivelse:</b>	Metode kald til interupt rutine. Stopper modtageren i at læse input fra envelopen

**void getUnitStatus ( unsigned char unitID )**

<b>Parametre:</b>	Den ønskedes enheds adresse der skal kommunikeres med
<b>Rerurværdi:</b>	Ingen
<b>Beskrivelse:</b>	Sendere en anmodning til en bestemt unit om dens nuværende status

**void changeState ( unsigned char unitID, bool newState )**

<b>Parametre:</b>	Den ønskedes enheds adresse der skal kommunikeres med, den nye status enheden skal have
<b>Rerurværdi:</b>	Ingen
<b>Beskrivelse:</b>	Skifter status på en bestemt enhed

**void replyStatus ( bool status )**

<b>Parametre:</b>	Den nuværende status
<b>Rerurværdi:</b>	Ingen
<b>Beskrivelse:</b>	Sender enhedens nuværende status

**void replyOk ( void )**

<b>Parametre:</b>	Ingen
<b>Rerurværdi:</b>	Ingen
<b>Beskrivelse:</b>	Sender en standard OK respons til styrboksen

**void replyOk ( unsigned char unitID )**

<b>Parametre:</b>	Den ønskedes enheds adresse der skal kommunikeres med
<b>Rerurværdi:</b>	Ingen
<b>Beskrivelse:</b>	Sender en standard OK respons til en specifik enhed

**void replyError ( unsinged char error )**

<b>Parametre:</b>	Fejl koden der skal sendes
<b>Rerurværdi:</b>	Ingen
<b>Beskrivelse:</b>	Sender en specifik fejlkode til styrboksen

**void replyError (unsinged char error, unsigned char unitID )**

<b>Parametre:</b>	Fejl koden der skal sendes, den ønskedes enheds adresse der skal kommunikeres med
<b>Rerurværdi:</b>	Ingen
<b>Beskrivelse:</b>	Sender en specifik fejlkode til en specifik enhed

**unsigned char getLastError ( void )**

<b>Parametre:</b>	Ingen
<b>Rerurværdi:</b>	Den seneste modtaget fejlkode
<b>Beskrivelse:</b>	Returnere den sensete modtaget fejlkode

**void fetchCommand ( unsinged char& command, unsigned char& data )**

<b>Parametre:</b>	Variable hvor commando bliver gemt, variable hvor data bliver gemt
<b>Rerurværdi:</b>	Ingen
<b>Beskrivelse:</b>	Henter den seneste modtaget pakkes kommando og den data der muligvis er i den

### 3.10.2 X10.1 Modtager

**Ansvar:** Sørger for at der kan modtages kommunikation og validerer at denne er til sig og intakt.

**X10\_Modtager (unsigned char houseCode, unsigned char address)**

<b>Parametre:</b>	Den house code der skal kommunikeres på, adressen der kan modtages data på
<b>Rerurværdi:</b>	Ingen
<b>Beskrivelse:</b>	Construkter for klassen X10_Modtager. Modtager house code og adresse som der skal modtages data på

**void receiveHigh ( void )**

<b>Parametre:</b>	Ingen
<b>Rerurværdi:</b>	Ingen
<b>Beskrivelse:</b>	Søger for at der kun modtages et High når nogle prækonditioner er mødt

**void receiveLow ( void )**

<b>Parametre:</b>	Ingen
<b>Rerurværdi:</b>	Ingen
<b>Beskrivelse:</b>	Søger for at der kun modtages et Low når nogle prækonditioner er mødt

**void fetchdata ( unsinged char& type, unsigned char& data, bool& validity )**

<b>Parametre:</b>	Variable hvor commando bliver gemt, variable hvor data bliver gemt, variable hvor validity bliver gemt
<b>Rerurværdi:</b>	Ingen
<b>Beskrivelse:</b>	Henter den seneste modtaget pakkes type, den data der muligvis er i den og om pakken er intakt

**bool isDataReady ( void )**

<b>Parametre:</b>	Ingen
<b>Rerurværdi:</b>	Returnere true hvis der er data klar til at blive læst
<b>Beskrivelse:</b>	Returnere true hvis der er data klar til at blive læst

**void inputComplete ( void )**

<b>Parametre:</b>	Ingen
<b>Rerurværdi:</b>	Ingen
<b>Beskrivelse:</b>	Stopper modtageren i at fortsætte med at tjekke for data fra evelopen

**void resetReceiver ( void )**

<b>Parametre:</b>	Ingen
<b>Rerurværdi:</b>	Ingen
<b>Beskrivelse:</b>	Nulstiller hele modtageren, så den er klar til at modtage data på ny

**void receiveData ( unsigned char data )**

<b>Parametre:</b>	Den modtaget data 1 eller 0
<b>Rerurværdi:</b>	Ingen
<b>Beskrivelse:</b>	Modtager enten et 1 eller 0, og skriver denne ind i den nuværende pakke og tæller positionen op. Ved prædefinerede positioner tjekkes den nuværende modtaget data om den er ved den rigtige modtager, for rigtig STX. Hvis der findes ud af at den modtaget data ikke er ved den rigtige modtager, venter den på ETX og derefter resetter modtageren.

**void startTimeout ( void )**

<b>Parametre:</b>	Ingen
<b>Rerurværdi:</b>	Ingen
<b>Beskrivelse:</b>	Starter timeout timeren for modtageren

**void resetTimeout ( void )**

<b>Parametre:</b>	Ingen
<b>Rerurværdi:</b>	Ingen
<b>Beskrivelse:</b>	Nulstiller timeout timeren

**void startInput ( void )**

<b>Parametre:</b>	Ingen
<b>Rerurværdi:</b>	Ingen
<b>Beskrivelse:</b>	Starter timer til at tjekke for modtaget data i prædefineret tid

**void writeDataToPacket1 ( unsigned char data )**

<b>Parametre:</b>	Den modtaget data 1 eller 0
<b>Rerurværdi:</b>	Ingen
<b>Beskrivelse:</b>	Skriver den modtaget data til pakke 1

**void writeDataToPacket2 ( unsigned char data )**

<b>Parametre:</b>	Den modtaget data 1 eller 0
<b>Rerurværdi:</b>	Ingen
<b>Beskrivelse:</b>	Skriver den modtaget data til pakke 2

**bool checkForZero ( unsigned char data )**

<b>Parametre:</b>	Den modtaget data 1 eller 0
<b>Rerurværdi:</b>	Returnere true hvis den modtaget data er lig 0
<b>Beskrivelse:</b>	Tjekker om den seneste data er et 0 eller et 1

**bool isPacketForMe ( void )**

<b>Parametre:</b>	Ingen
<b>Rerurværdi:</b>	Returnere true hvis house code'en og adressen i den modtaget data passer med sin egen house code og adresse
<b>Beskrivelse:</b>	Returnere true hvis house code'en og adressen i den modtaget data passer med sin egen house code og adresse

**unsigned char getPacketType ( void )**

<b>Parametre:</b>	Ingen
<b>Rerurværdi:</b>	Returnere typen fra pakken
<b>Beskrivelse:</b>	Returnere typen i den modtaget pakke

**unsigned char getPacketData ( void )**

<b>Parametre:</b>	Ingen
<b>Rerurværdi:</b>	Returnere data'en fra pakken
<b>Beskrivelse:</b>	Returnere data'en i den modtaget pakke

**unsigned char getCurrentPos1 ( void )**

<b>Parametre:</b>	Ingen
<b>Rerurværdi:</b>	Returnere den nuværende position der skrives til i pakke 1
<b>Beskrivelse:</b>	Returnere den nuværende position der skrives til i pakke 1

**unsigned char getCurrentPos2 ( void )**

<b>Parametre:</b>	Ingen
<b>Rerurværdi:</b>	Returnere den nuværende position der skrives til i pakke 2
<b>Beskrivelse:</b>	Returnere den nuværende position der skrives til i pakke 2

**unsigned char getCurrentPacket ( void )**

<b>Parametre:</b>	Ingen
<b>Rerurværdi:</b>	Returnere den nuværende pakke der skrives til
<b>Beskrivelse:</b>	Returnere den nuværende pakke der skrives til

**bool validatePackets ( void )**

<b>Parametre:</b>	Ingen
<b>Rerurværdi:</b>	Returnere true hvis pakken er validt
<b>Beskrivelse:</b>	Tjekker om parity er korrekt

**bool isValidStart ( void )**

<b>Parametre:</b>	Ingen
<b>Rerurværdi:</b>	Returnere true hvis starten på pakken er validt
<b>Beskrivelse:</b>	Tjekker om de fire første bit er de samme som STX

**unsigned char getData ( void )**

<b>Parametre:</b>	Ingen
<b>Rerurværdi:</b>	Returnere data på pakken
<b>Beskrivelse:</b>	Returnere data på pakken

**unsigned char getType ( void )**

<b>Parametre:</b>	Ingen
<b>Rerurværdi:</b>	Returnere type på pakken
<b>Beskrivelse:</b>	Returnere type på pakken

### 3.10.3 X10.1 Sender

**Ansvar:** Sørger for at den ønskede data der skal sendes bliver pakket rigtigt sammen, og sendt.

#### X10\_Sender ( void )

<b>Parametre:</b>	Ingen
<b>Rerurværdi:</b>	Ingen
<b>Beskrivelse:</b>	Construktør for klassen X10_Sender.

**void sendCommand ( unsinged char houseCode, unsigned char address, unsigned char type, unsigned char data = 0 )**

<b>Parametre:</b>	House code som skal sendes på, adresse som skal modtage data'en, typen på beskeden der bliver sendt, data som muligvis skal sendes med
<b>Rerurværdi:</b>	Ingen
<b>Beskrivelse:</b>	Kompile og sende en angivet type besked til en angivet modtager.

#### void sendHigh ( void )

<b>Parametre:</b>	Ingen
<b>Rerurværdi:</b>	Ingen
<b>Beskrivelse:</b>	Søger for at der kun sendes et High når nogle prækonditioner er mødt

#### void sendLow ( void )

<b>Parametre:</b>	Ingen
<b>Rerurværdi:</b>	Ingen
<b>Beskrivelse:</b>	Søger for at der kun sendes et Low når nogle prækonditioner er mødt

#### void stop120 ( void )

<b>Parametre:</b>	Ingen
<b>Rerurværdi:</b>	Ingen
<b>Beskrivelse:</b>	Stopper senderens 120 kHz PWM

#### bool getSendMode ( void )

<b>Parametre:</b>	Ingen
<b>Rerurværdi:</b>	Returnere true hvis der senderen er i gang med at sende
<b>Beskrivelse:</b>	Returnere true hvis der senderen er i gang med at sende

#### void start120 ( void )

<b>Parametre:</b>	Ingen
<b>Rerurværdi:</b>	Ingen
<b>Beskrivelse:</b>	Starter senderens 120 kHz PWM og timer til at stoppe den

**void setSendMode ( bool )**

<b>Parametre:</b>	Bool med status på om der bliver sendt
<b>Rerurværdi:</b>	Ingen
<b>Beskrivelse:</b>	Setter status på om der er ved at blive sendt data

**unsinged int getPacket1 ( void )**

<b>Parametre:</b>	Ingen
<b>Rerurværdi:</b>	Returnere packet1
<b>Beskrivelse:</b>	Returnere packet1

**unsinged int getPacket2 ( void )**

<b>Parametre:</b>	Ingen
<b>Rerurværdi:</b>	Returnere packet2
<b>Beskrivelse:</b>	Returnere packet2

**unsigned char getCurrentPacket ( void )**

<b>Parametre:</b>	Ingen
<b>Rerurværdi:</b>	Returnere hvilken pakke der bliver arbejdet på
<b>Beskrivelse:</b>	Returnere hvilken pakke der bliver arbejdet på

**void setCurrentPacket ( unsigned char )**

<b>Parametre:</b>	Hvilken pakke der skal arbejdes på
<b>Rerurværdi:</b>	Ingen
<b>Beskrivelse:</b>	Setter hvilken pakke der skal arbejdes på

**void decrementCurrentPos1 ( void )**

<b>Parametre:</b>	Ingen
<b>Rerurværdi:</b>	Ingen
<b>Beskrivelse:</b>	Decrementer positionen på pakke 1

**unsigned char getCurrentPos1 ( void )**

<b>Parametre:</b>	Ingen
<b>Rerurværdi:</b>	Returnere positionen på pakke 1
<b>Beskrivelse:</b>	Returnere positionen på pakke 1

**void decrementCurrentPos2 ( void )**

<b>Parametre:</b>	Ingen
<b>Rerurværdi:</b>	Ingen
<b>Beskrivelse:</b>	Decrementer positionen på pakke 2

**unsinged char getCurrentPos2 ( void )**

<b>Parametre:</b>	Ingen
<b>Rerurværdi:</b>	Returnere positionen på pakke 2
<b>Beskrivelse:</b>	Returnere positionen på pakke 2

**unsigned char generateParity ( unsinged int packet1, unsinged int packet2 )**

<b>Parametre:</b>	Pakke 1, pakke 2
<b>Rerurværdi:</b>	Returnere parity på hele pakken
<b>Beskrivelse:</b>	Genere parity til hele pakken

## 4 Hardware Design og Implementation (ME, MB, TN, DP)

Herunder skrives design og implementation for hardware delene af X10 Home Automation Systemet.

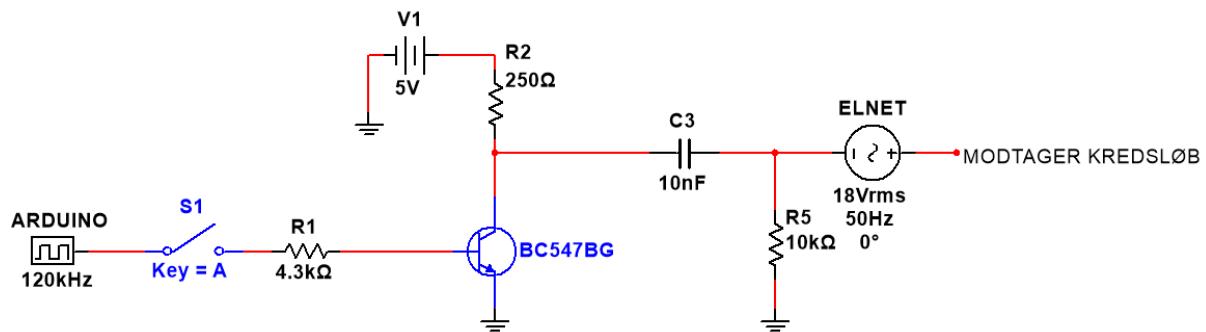
### 4.1 Sender kredsløb

#### 4.1.1 Design

Nedenfor ses på Figur 66 et diagram over sender-kredsløbet. Det læses fra venstre mod højre og kredsløbet består af to moduler – et transmitterkredsløb koblet sammen med et 1. ordens højpas filter.

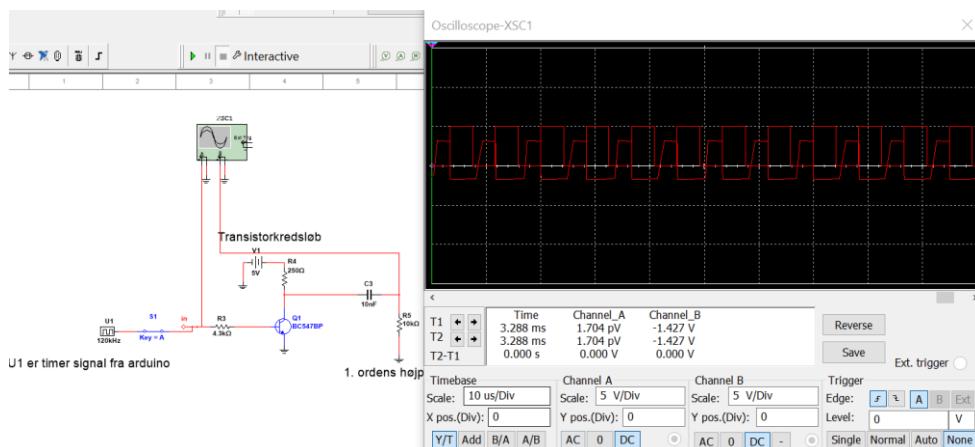
Transistoren består af et collector-, base- og emitterben. Når en mindre strøm tilføres basebenet styres en større strøm fra collectorbenet ned gennem emitterbenet. Det giver transistoren en funktion som en kontakt. Det bruges til at skabe et 120 kHz firkantsignal som sendes ud på elnettet. Signalet der bliver sendt ud på elnettet styres fra Arduinoen ved at denne udsender 120 kHz firkantsignal ind på basebenet af transistoren. Det får transistoren til at reagere ved at tænde og slukke 120.000 gange i sekundet. Dette generer et 120 kHz firkantsignal i knudepunktet mellem transistorens collectorben, R2 og C3. Al strøm og spænding der ryger ud på elnettet trækkes derfor fra forsyningsspændingen.

Højpas filteret skal sikre at strømmen fra elnettet ikke løber ind i transmitterkredsløbet og i værste fald løber ind i Arduinoen og gør skade. I modsætning til modtager kredsløbet er der her brugt et 1. ordens højpas filter fordi det kun skal tage hensyn til et signal, som er de 18V AC 50 Hz sinussignal fra elnettet.



Figur 66 – Multisim diagram over sender kredsløb

Kontakten S1 bruges til at simulere om Arduinoen sender et signal eller ej. Spændingskilden V1 symboliserer spændingsforsyningen der anvendes i laboratoriet.



Figur 67 – Simulering af sender kredsløb I Multisim

#### 4.1.1.1 Transmitterkredsløb

Transmitterkredsløbet består af en  $4.3\text{ k}\Omega$  modstand (R1), en BC547B transistor og en modstand på  $250\Omega$  (R2).

Der er et spændingsfald mellem base- og emitterbenet på 0.7 V. Ohms lov bruges til at udregne R1 fra Figur 66:

$$V_{base} := 5 \text{ V} \quad i_{base} := 1 \text{ mA}$$

$$R1 := \frac{V_{base} - 0.7 \text{ V}}{i_{base}} = 4.3 \text{ k}\Omega$$

Figur 68 – Udregning på R1 fra Figur 66

Ud fra databladet for transistor BC547B aflæses  $\beta=20$ . Strømmen på collector benet bliver derfor 20 mA når strømmen på basebenet er 1 mA. R2 udregnes ved hjælp af Ohms lov, nu bare med værdier ift. collectorbenet:

$$V_{collector} := 5 \text{ V} \quad i_{collector} := 20 \text{ mA}$$

$$R2 := \frac{V_{collector}}{i_{collector}} = 250 \text{ }\Omega$$

Figur 69 – Udregning af R2 fra Figur 66

#### 4.1.1.2 Højpas filter

Højpas filteret er et passivt 1. ordens højpas filter bestående af en modstand  $R5=10\text{ k}\Omega$  og en kondensator  $C3=10\text{ nF}$ . Kondensatorens størrelse er valgt ud fra den viden om at jo større værdi på kondensatoren jo større lækstrøm.

Værdierne i højpas filteret er valgt ud fra følgende beregninger:

$$T(s) = \frac{Ks}{s+\alpha} \Rightarrow T(s) = \frac{K \cdot \left( s \cdot \frac{1}{\alpha} \right)}{\left( s \cdot \frac{1}{\alpha} \right) + 1} \Rightarrow K = 1 \quad T(s) = \frac{s \cdot \frac{1}{\alpha}}{\left( s \cdot \frac{1}{\alpha} \right) + 1}$$

Figur 70 – Omskrivning af overføringsfunktion T(s) for højpas filteret

$$V_{out} := \frac{R}{R + \frac{1}{C \cdot s}} \cdot V_{in} \Rightarrow \frac{V_{out}}{V_{in}} = \frac{R}{R + \frac{1}{C \cdot s}} \Rightarrow \frac{V_{out}}{V_{in}} = \frac{RCs}{RCs + 1}$$

Figur 71 – Spændingsdeler over modstand i højpas filteret

$$T(s) = \frac{V_{out}}{V_{in}} \Rightarrow \frac{s \cdot \frac{1}{\alpha}}{\left( s \cdot \frac{1}{\alpha} \right) + 1} = \frac{RCs}{RCs + 1}$$

Figur 72 – Overføringsfunktionen T(s) og spændingsdeleren sættes lig hinanden

Ud fra de to funktioner på Figur 72 kan modstanden i højpas filteret udregnes.

$$f := 1600 \text{ Hz} \quad \omega_0 := f \cdot 2 \pi$$

$$\alpha := \omega_0 \quad C := 10 \text{ nF}$$

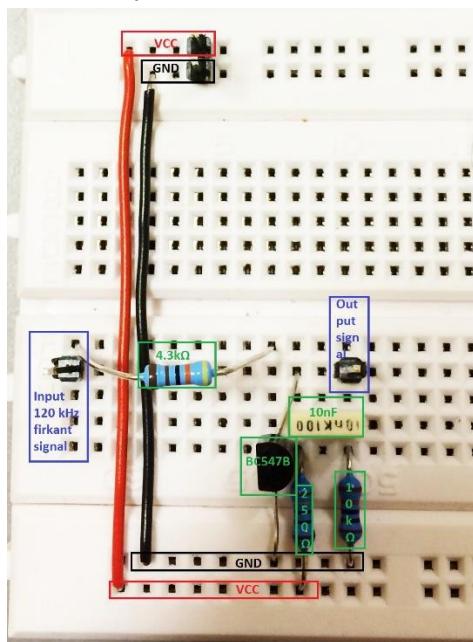
$$R \cdot C = \frac{1}{\alpha} \xrightarrow{\text{solve}, R} \frac{1}{32000 \cdot \pi \cdot nF \cdot Hz} = 9947.184 \Omega$$

Figur 73 – Modstanden i højpas filteret udregnes

Cutoff-frekvensen er sat til 1600 Hz da den eneste funktion er at sortere de 50 Hz sinussignal fra elnettet væk, og at en lille kondensator var at foretrække. 10 nF kondensator blev valgt fordi en passende modstand er nemmere at finde frem for at tage udgangspunkt i en modstand og derefter finde en kondensator.

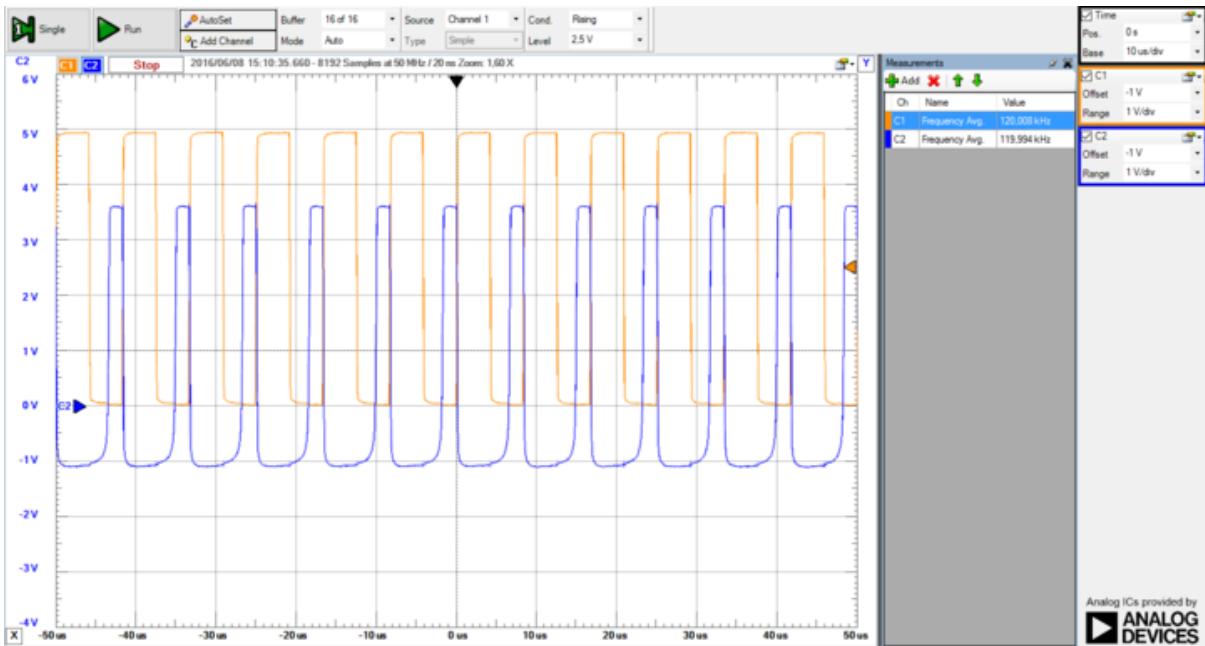
Modstanden R i udregningerne svarer til R5 på Figur 66. C svarer til C3 på Figur 66.

#### 4.1.2 Implementation



Figur 74 – Sender kredsløb opbygget på fumlebrædt

Figur 74 viser opstillingen af sender-kredsløbet som der er blevet brugt til at foretage de nødvendige målinger.



Figur 75 – Osciloskop billede af sender kredsløbet hvor gul er indgangssignal og blå er udgangssignal

Det ses ud fra Figur 75 at indgangssignalet passerer højpas filteret og derved ikke bliver filtreret fra. Realiseringen stemmer godt overens med simuleringen fra Figur 67.

## 4.2 Modtager kredsløb

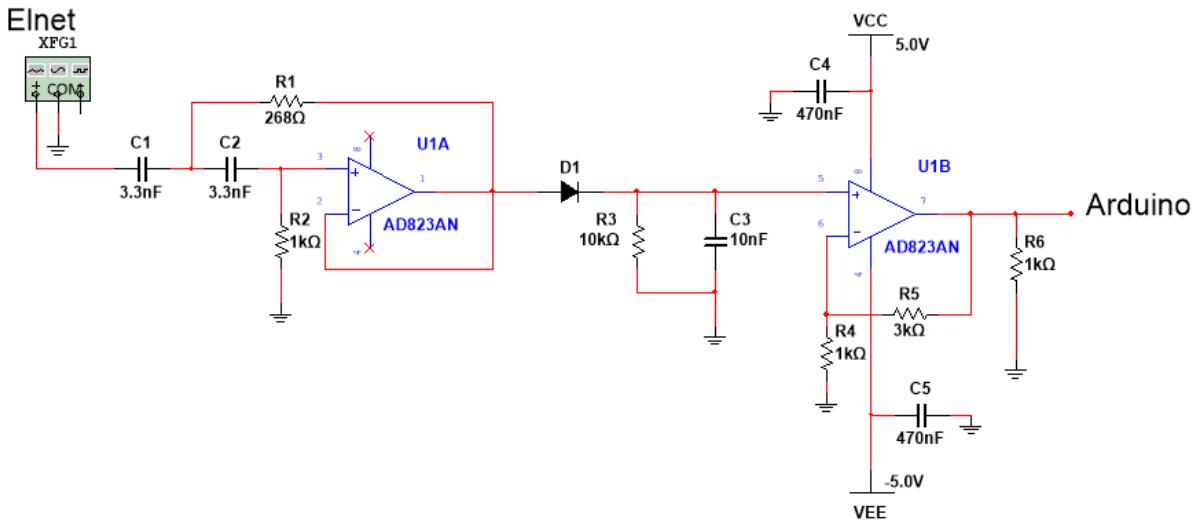
### 4.2.1 Design

Nedenfor ses på Figur 76 et diagram over modtager-kredsløbet. Det læses fra venstre mod højre og kredsløbet består af tre moduler – et aktivt 2. ordens højpas filter koblet sammen med en envelope detector videre til en signalforstærker.

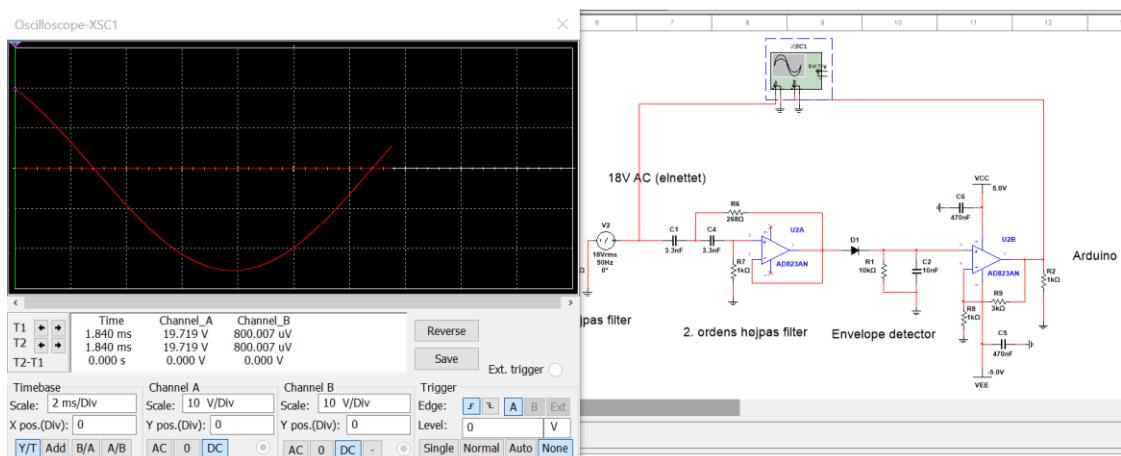
Højpas filteret i modtager kredsløbet bruges til at sortere 50 Hz sinussignal samt støj fra elnettet væk. Modtager kredsløbet er mere følsomt over for støj og på denne baggrund er der taget beslutning om at bruge et 2. ordens højpas filter da det dæmper dobbelt så meget som et 1. ordens højpas filter.

Envelope detectorens funktion er at konvertere 120 kHz firkantsignal om til et digitalt højt signal. Når der ikke kommer et 120 kHz firkantsignal er der et digitalt lavt signal på udgangen af envelope detectoren.

Signalforstærkeren forstærker det digitale signal så Arduino MEGA2560 kan aflæse signalet. Hvis signalet ikke forstærkes er der risiko for at Arduinoen ikke kan aflæse det digitale signal, fordi Arduinoen kun kan aflæse signaler der ligger mellem 3-5.5 V i følge databladet for Arduino MEGA2560 målt på et digitalt højt signal.

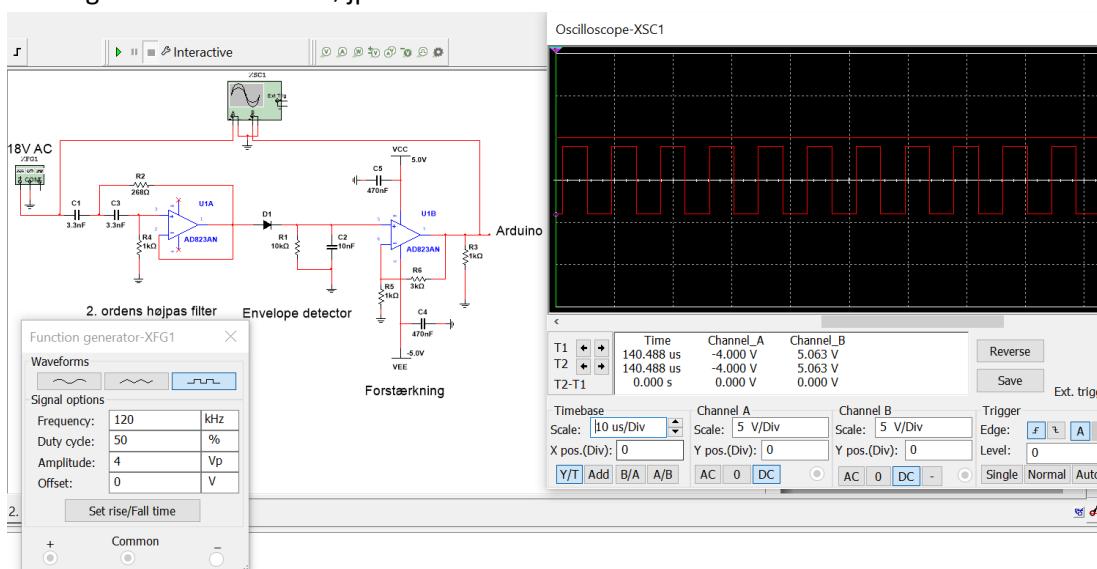


Figur 76 – Multisim diagram over modtager kredsløb



Figur 77 – Simulering af modtager kredsløb med 50 Hz sinussignal som indgangssignal på højpas filteret

På Figur 77 ses det at udgangssignalet på modtager kredsløbet er lavt er fordi indgangssignalet på 50 Hz sinussignal sorteres fra af højpas filteret.



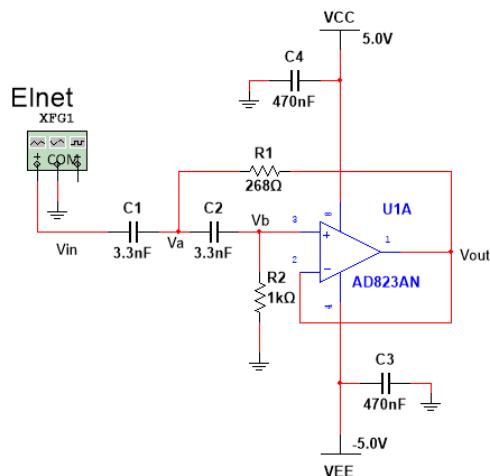
Figur 78 – Simulering af modtager kredsløb med 120 kHz firkantsignal som indgangssignal på højpas filteret

På Figur 78 ses det at udgangssignalet på modtager kredsløbet er højt fordi indgangssignalet på 120 kHz firkantsignal passerer højpas filteret og envelope detectoren omdanner signalet til et digitalt højt signal.

#### 4.2.1.1 Højpas filter

Højpas filteret er et aktivt 2. ordens højpas filter bestående af 2 kondensatorer på 3.3 nF, 2 modstande på henholdsvis 1 kΩ og 268 Ω, og en AD823 operationsforstærker. AD823 er en IC som indeholder to operationsforstærkere hvor den anden bruges i signalforstærker kredsløbet.

Da der er negativ feedback på operationsforstærkeren og forskellen i indgangsspændingerne er lig nul, må udgangsspændingen være lig med indgangsspændingen på ben 3.



Figur 79 – Multisim diagram over 2. ordens højpas filter

$$V_b = V_{out}$$

Figur 80 – Spændingerne sættes ens da det giver færre ukendte

$$Z_1 = \frac{1}{C_1 \cdot s} \quad Z_2 = \frac{1}{C_2 \cdot s}$$

Figur 81 – Kondensatorernes impedanceres defineres for at simplificere regnestykket

$$\frac{V_A - V_{out}}{Z_2} = \frac{V_{out}}{R_2}$$

Figur 82 – KCL laves i knudepunktet Vb

$$V_A := \frac{V_A - V_{out}}{Z_2} = \frac{V_{out}}{R_2} \xrightarrow{\text{solve}, V_A} \frac{R_2 \cdot V_{out} + Z_2 \cdot V_{out}}{R_2}$$

Figur 83 – Va defineres på baggrund af at strømmen gennem Z1 lig med strømmen gennem R2

$$\frac{V_{in} - V_A}{Z_1} = \frac{V_A - V_{out}}{Z_2} + \frac{V_A - V_{out}}{R_1}$$

Figur 84 – KCL laves i knudepunktet Va

$$\frac{V_{in} - V_A}{Z_1} = \frac{V_A - V_{out}}{Z_2} + \frac{V_A - V_{out}}{R_1} \xrightarrow{\text{simplify}} -\frac{R_2 \cdot V_{out} - R_2 \cdot V_{in} + Z_2 \cdot V_{out}}{R_2 \cdot Z_1} = \frac{V_{out} \cdot (R_1 + Z_2)}{R_1 \cdot R_2}$$

Figur 85 – Mathcad indsætter  $V_A$  fra Figur 83 i ligningen fra Figur 82

$$Z_1 := \frac{1}{C_1 \cdot s} \quad Z_2 := \frac{1}{C_2 \cdot s}$$

Figur 86 – Kondensatorernes symbolske impedans værdier

$$\frac{R_2 \cdot V_{out} - R_2 \cdot V_{in} + Z_2 \cdot V_{out}}{R_2 \cdot Z_1} = \frac{V_{out} \cdot (R_1 + Z_2)}{R_1 \cdot R_2} \xrightarrow{\text{solve}, V_{in}} \frac{V_{out} + s \cdot C_1 \cdot R_1 \cdot V_{out} + s \cdot R_1 \cdot C_2 \cdot V_{out} + s^2 \cdot C_1 \cdot R_1 \cdot C_2 \cdot R_2 \cdot V_{out}}{s^2 \cdot C_1 \cdot R_1 \cdot C_2 \cdot R_2}$$

Figur 87 – Mathcad insætter impedansernes symbolske værdier fra Figur 86

$$V_{in} = V_{out} \cdot \frac{1 + s \cdot C_1 \cdot R_1 + s \cdot R_1 \cdot C_2 + s^2 \cdot C_1 \cdot R_1 \cdot C_2 \cdot R_2}{s^2 \cdot C_1 \cdot R_1 \cdot C_2 \cdot R_2}$$

Figur 88 –  $V_{out}$  sættes uden for parentes

$$\frac{V_{in}}{V_{out}} = \frac{1 + s \cdot C_1 \cdot R_1 + s \cdot R_1 \cdot C_2 + s^2 \cdot C_1 \cdot R_1 \cdot C_2 \cdot R_2}{s^2 \cdot C_1 \cdot R_1 \cdot C_2 \cdot R_2} \xrightarrow{\text{collect}, s} \frac{V_{in}}{V_{out}} = \frac{s \cdot C_1 \cdot R_1 + s \cdot R_1 \cdot C_2 + s^2 \cdot C_1 \cdot R_1 \cdot C_2 \cdot R_2 + 1}{s^2 \cdot C_1 \cdot R_1 \cdot C_2 \cdot R_2}$$

Figur 89 –  $V_{out}$  sættes på den anden side af lighedstegnet og samler  $s$

$$\left( \frac{s \cdot C_1 \cdot R_1 + s \cdot R_1 \cdot C_2 + s^2 \cdot C_1 \cdot R_1 \cdot C_2 \cdot R_2 + 1}{s^2 \cdot C_1 \cdot R_1 \cdot C_2 \cdot R_2} \right)^{-1} \xrightarrow{\text{collect}, s} \frac{s^2 \cdot C_1 \cdot R_1 \cdot C_2 \cdot R_2}{C_1 \cdot R_1 \cdot C_2 \cdot R_2 \cdot s^2 + (C_1 \cdot R_1 + R_1 \cdot C_2) \cdot s + 1}$$

Figur 90 – Den reciprokke værdi udregnes

$$K := 1$$

$$T(s) = \frac{K \left( \frac{s}{\omega_0} \right)^2}{\left( \frac{s}{\omega_0} \right)^2 + 2 \cdot \zeta \left( \frac{s}{\omega_0} \right) + 1}$$

Figur 91 – Standard overføringsfunktion for 2. ordens højpas filter hvor  $K=1$  da forstærkningen sættes til 1

$$T(s) = \frac{\left( \frac{s}{\omega_0} \right)^2}{\left( \frac{s}{\omega_0} \right)^2 + 2 \cdot \zeta \left( \frac{s}{\omega_0} \right) + 1} = \frac{s^2 \cdot C_1 \cdot R_1 \cdot C_2 \cdot R_2}{C_1 \cdot R_1 \cdot C_2 \cdot R_2 \cdot s^2 + (C_1 \cdot R_1 + R_1 \cdot C_2) \cdot s + 1}$$

Figur 92 – Standard overføringsfunktionen sammenlignes med funktionen fra Figur 90

$$C1 = C2$$

Figur 93 – For nemhedens skyld bruges ens værdier på kondensatorerne

$$\frac{1}{\omega_0} = \sqrt{C_1 \cdot R_1 \cdot C_2 \cdot R_2} \rightarrow \frac{1}{\omega_0} = \sqrt{R_1 \cdot C_2^2 \cdot R_2}$$

$$\frac{2 \cdot \zeta}{\omega_0} = C_1 \cdot R_1 + R_1 \cdot C_2 \rightarrow \frac{2 \cdot \zeta}{\omega_0} = 2 \cdot R_1 \cdot C_2$$

Figur 94 – De to overføringsfunktioners nævnere sammenlignes og udtrykkes ved Zeta og cutoff-frekvens

$$C_2 := 3.3 \text{ nF} \quad w_0 := 90 \text{ kHz} \cdot 2 \pi = (565.48668 \cdot 10^3) \frac{\text{rad}}{\text{s}} \quad \zeta := 0.5 \quad \omega_0 := 565.48668 \cdot 10^3 \frac{\text{rad}}{\text{s}}$$

$$R_1 := \frac{2 \cdot \zeta}{\omega_0} = 2 \cdot R_1 \cdot C_2 \xrightarrow{\text{solve}, R_1} \frac{2.6793761351753062539 \cdot 10^{-7} \cdot s}{nF \cdot rad} = 267.938 \Omega$$

Figur 95 – R1 findes ud fra definerede værdier

$$\frac{1}{\omega_0} = \sqrt{R_1 \cdot C_2^2 \cdot R_2} \xrightarrow{\text{solve}, R_2} \frac{1.0717504540701225231 \cdot 10^{-15} \cdot s^2}{nF^2 \cdot rad^2 \cdot \Omega} = 1.072 \text{ k}\Omega$$

Figur 96 – R2 bestemmes

#### 4.2.1.2 Envelope detector

Envelope detectoren består af en modstand på 10 kΩ og en kondensator på 10 nF.

For at lave 120 kHz firkantsignal om til digitalt signal skal kondensatorens afladningstid ligge indenfor intervallet mellem 8.33 μs og 10 ms. Dette gøres for at undgå at kondensatoren bliver helt afladt mellem perioderne. Afladningstiden τ bestemmes til 0.1 ms. Kondensatorens størrelse sættes til 10 nF og en modstand findes ud følgende formel:

$$C := 10 \text{ nF} \quad t := 0.1 \text{ ms}$$

$$\tau = R \cdot C$$

$$R := \frac{C}{\tau} = 10 \text{ k}\Omega$$

Figur 97 – Modstanden i envelope detectoren bestemmes

#### 4.2.1.3 Signalforstærker

Signalforstærkeren består af 1 kΩ modstande, en modstand på 3 kΩ, en AD823 operationsforstærker hvor der er tilkoblet de nødvendige to kondensatorer på 470 nF, en 5 V forsyningsspænding på ben 8 og en -5 V forsyningsspænding på ben 4. -5 V forsyningsspændingen bliver i realiseringen skabt ved en inverterende spændingsforsyning.

$$A_{CL} = \frac{V_{out}}{V_{in}} = 1 + \frac{R_2}{R_1}$$

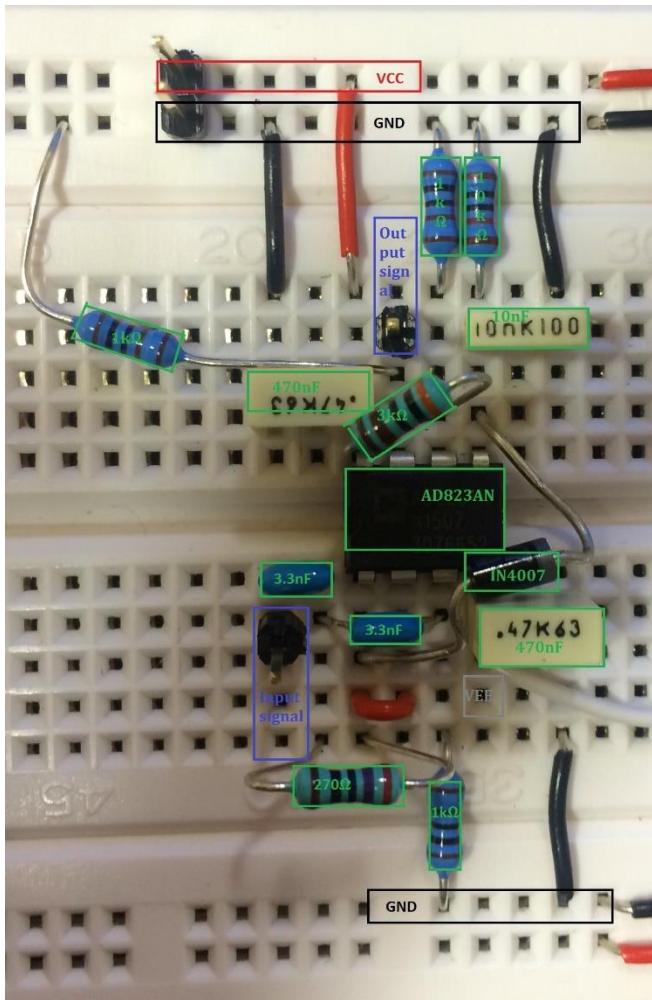
Figur 98 – Generel formel for forstærkning

$$R_4 := 1 \text{ k}\Omega$$

$$4 = 1 + \frac{R_5}{R_4} \xrightarrow{\text{solve}, R_5} 3 \cdot k\Omega$$

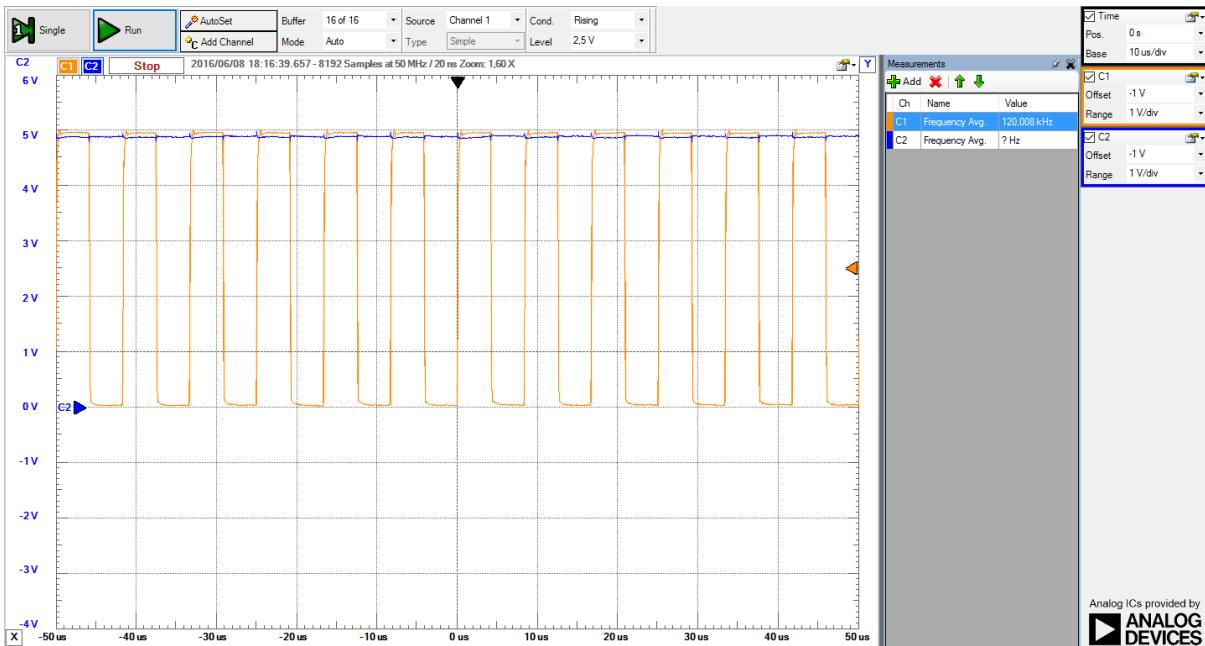
Figur 99 – R2 udregnes

#### 4.2.2 Implementation

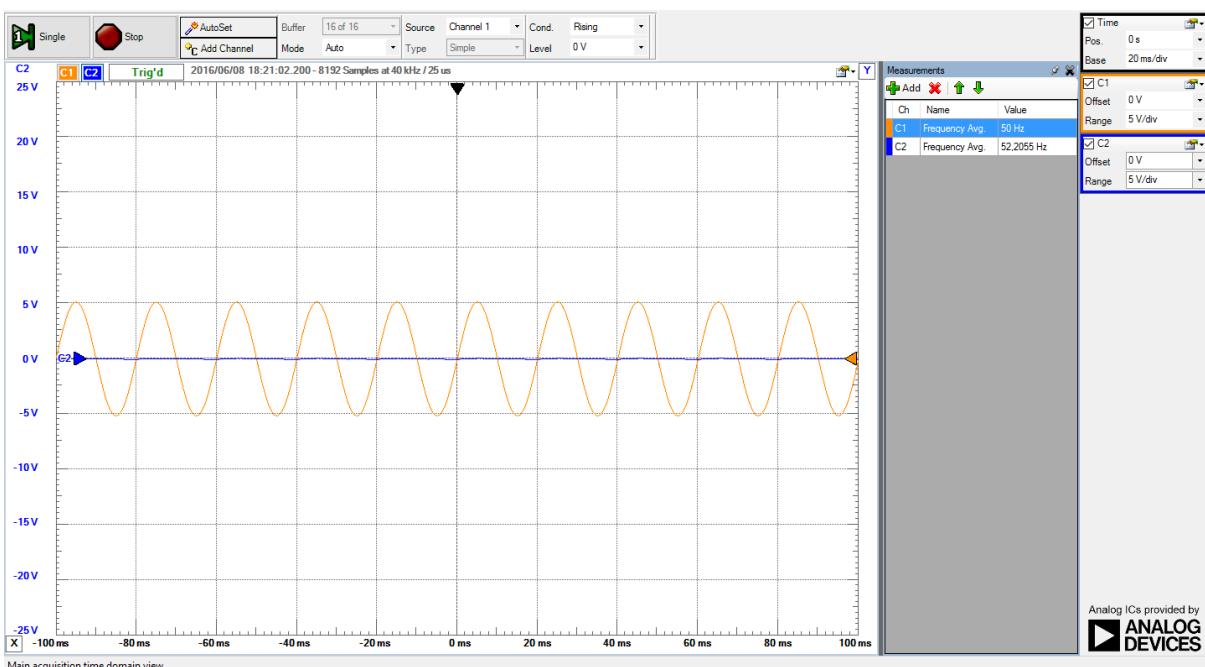


Figur 100 – Modtager kredsløb opbygget på fumlebrædt

Figur 100 viser opstillingen af modtager-kredsløbet som der er blevet brugt til at foretage de nødvendige målinger.



Figur 101 – Oscilloskop billede af modtager-kredsløbet med 120 kHz firkant indgangssignal (gul) og et digitalt højt udgangssignal (blå)



Figur 102 - Oscilloskop billede af modtager-kredsløbet med 50 Hz sinus indgangssignal (gul) og et digitalt lavt udgangssignal (blå)

## 4.3 Zero cross

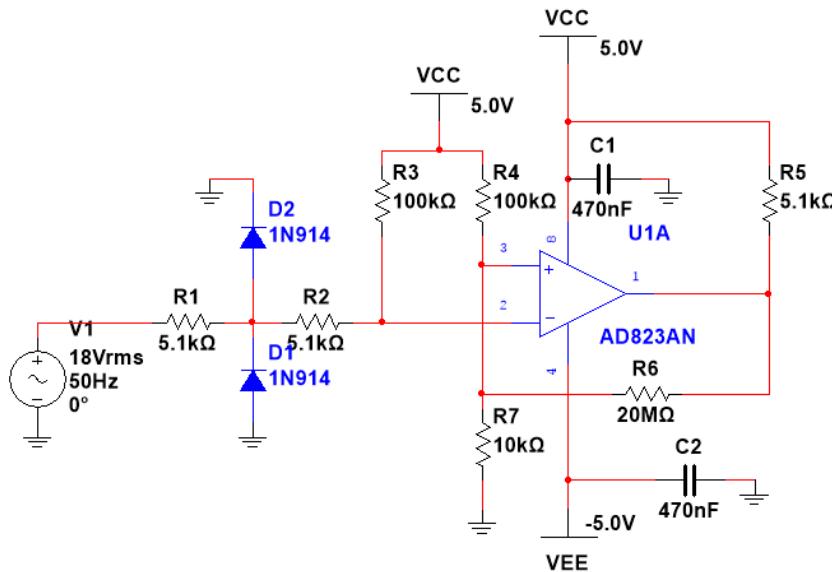
### 4.3.1 Design

Nedenfor ses på Figur 103 et diagram over zero cross-kredsløbet som består af et modul.

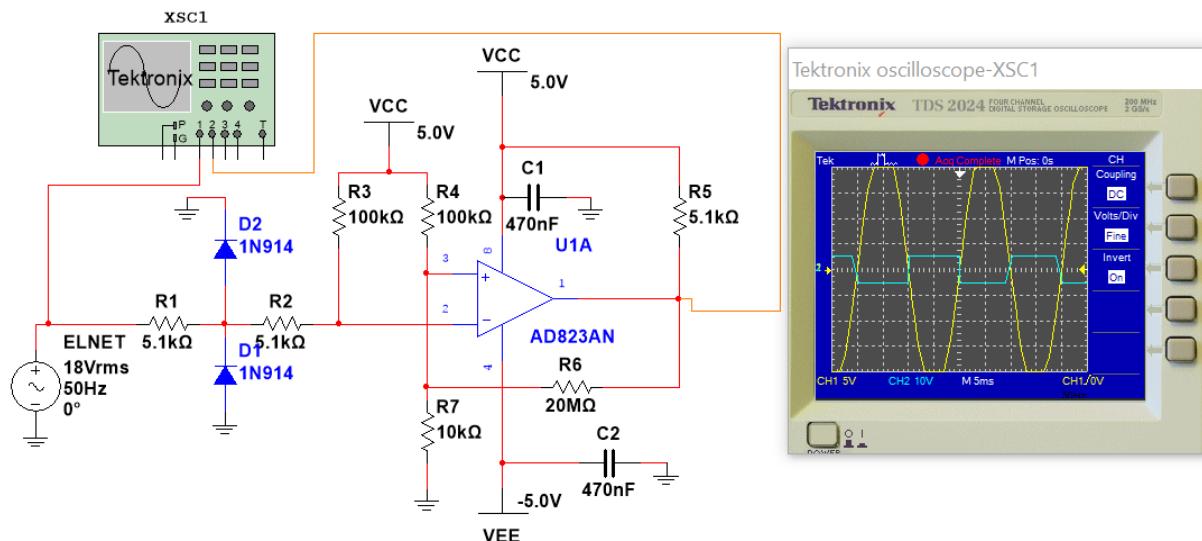
Kredsløbet bruges til at detektere nul-gennemgangene i elnettets sinuskurve. Ved rising edge på sinuskurven fås et digitalt lavt signal og omvendt på falling edge.

Kredsløbet er grundlæggende bygget ud fra databladet for LM339 operationsforstærker hvori der findes et kredsløb for zero cross. Det er modifieret med hjælp fra vejleder Henning Hargaard.

Operationsforstærkeren er blevet udskiftet med model AD823 som også bruges i andre kredsløb. Desuden er en ekstra diode sat ind i knudepunktet hvor D1 allerede sad med modsat spærretning.

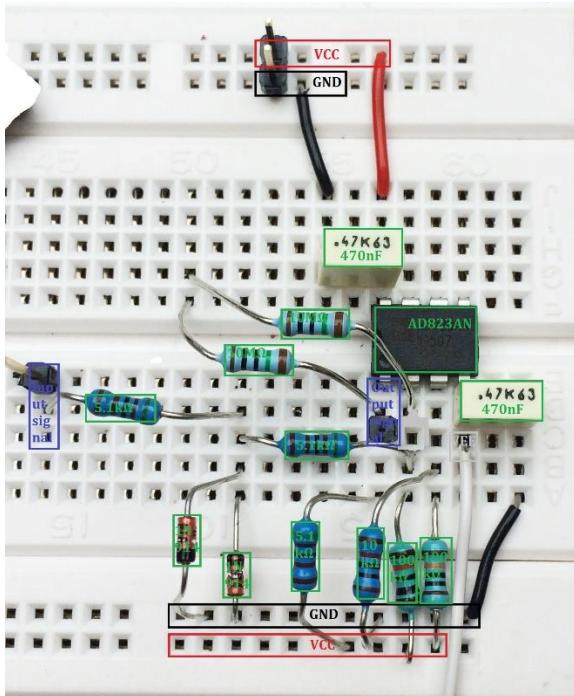


Figur 103 – Multisim diagram over zero cross-kredsløbet



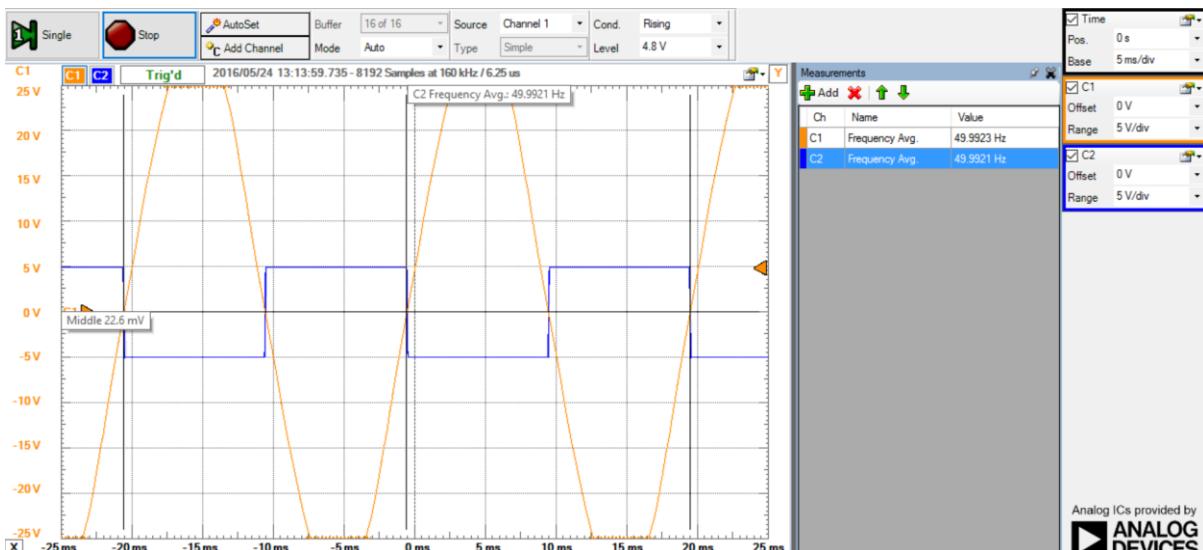
Figur 104 – Simulering af zero cross-kredsløb i Multisim

### 4.3.2 Implementation



Figur 105 – Zero cross-kredsløbet opbygget på fumlebrædt

Figur 105 viser opstilling af zero cross-kredsløbet som der er blevet brugt til at foretage de nødvendige målinger.



Figur 106 – Oscilloskop billede af zero cross-kredsløbet hvor gul er indgangssignal og blå er udgangssignal

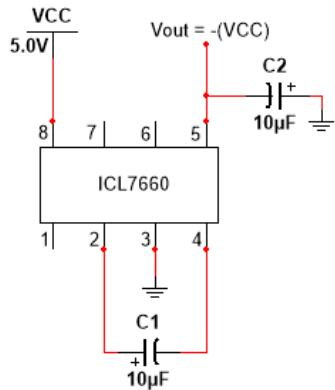
Det ses ud fra Figur 106 at zero cross-kredsløbet detekterer i nul-gennemgangene som de skal. Dette stemmer overens med simuleringen i Multisim fra Figur 104.

## 4.4 Voltage inverter

### 4.4.1 Design

Nedenfor ses på Figur 107 et diagram over voltage inverter-kredsløbet som består af et modul.

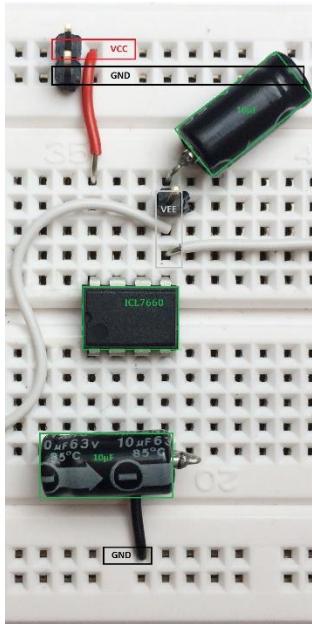
Voltage inverter-kredsløbet inverterer 5V fra forsyningsspændingen. -5V skal bruges til AD823 operationsforstærkerne på ben 4. Kredsløbet er opbygget ud fra databladet for ICL7660.



Figur 107 – Multisim diagram over Voltage inverter-kredsløbet

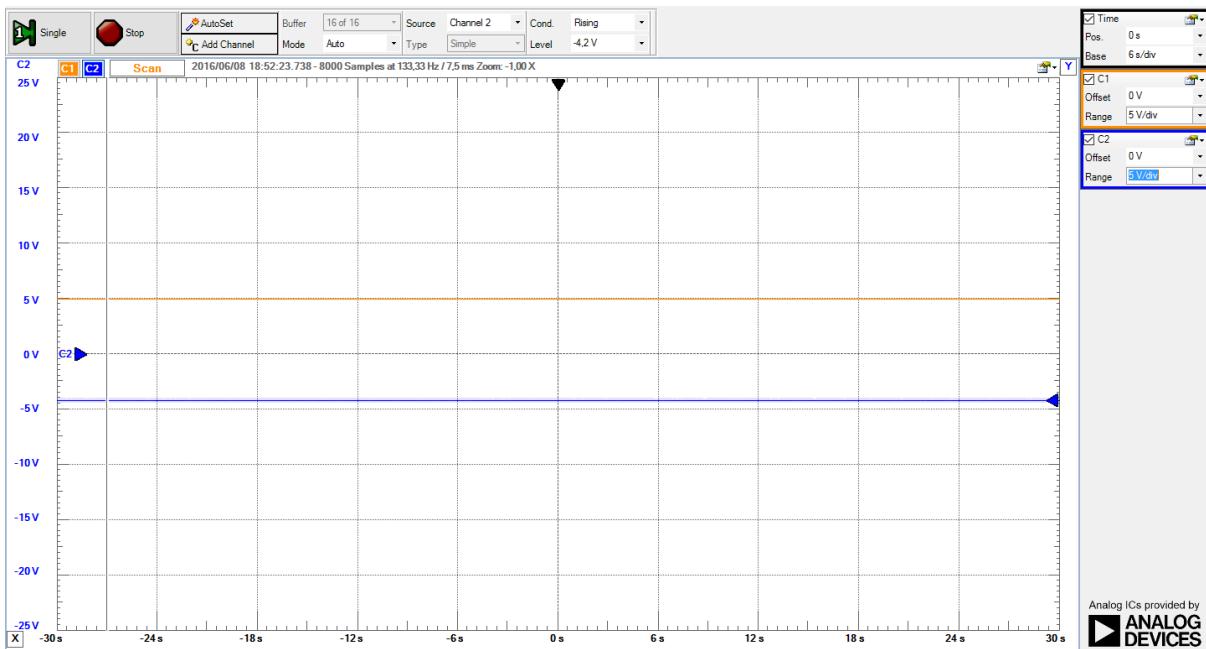
Voltage inverteren består af ICL7660 og to elektrolytkondensatorer på  $10 \mu F$ .

### 4.4.2 Implementation



Figur 108 – Voltage inverter-kredsløbet opbygget på fumlebrædt

Figur 108 viser opstillingen af voltage inverter-kredsløbet som der er blevet brugt til at foretage de nødvendige målinger.

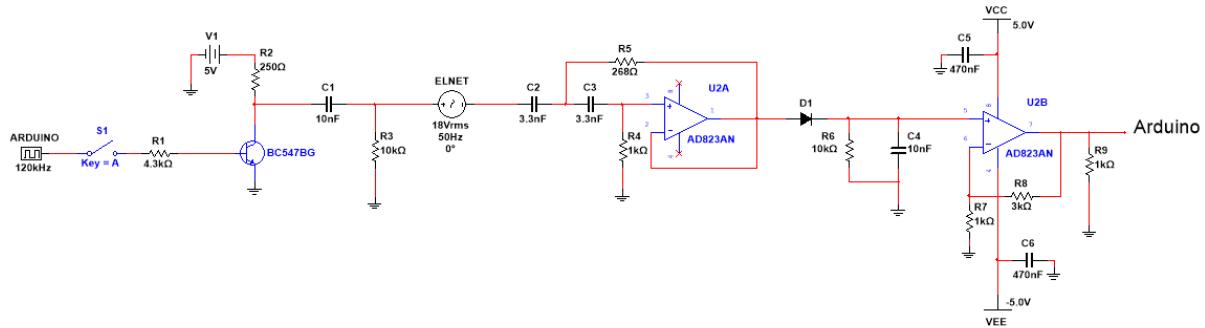


Figur 109 – Oscilloskop billede af voltage inverter-kredsløbet hvor gul er 5V indgangssignal og blå er -5V udgangssignal

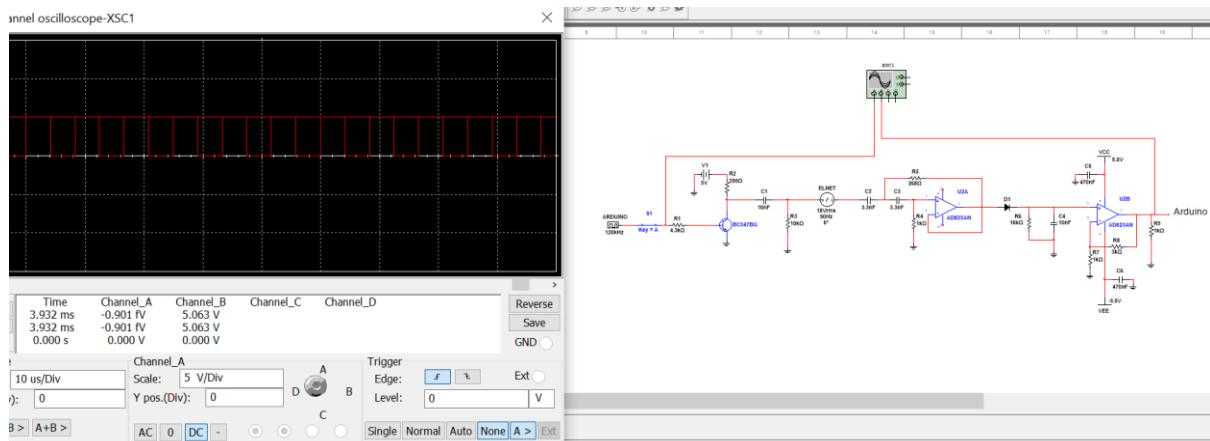
## 4.5 Samlet sender og modtager

### 4.5.1 Design

Nedenfor ses på Figur 110 et diagram over sender og modtager-kredsløbet.

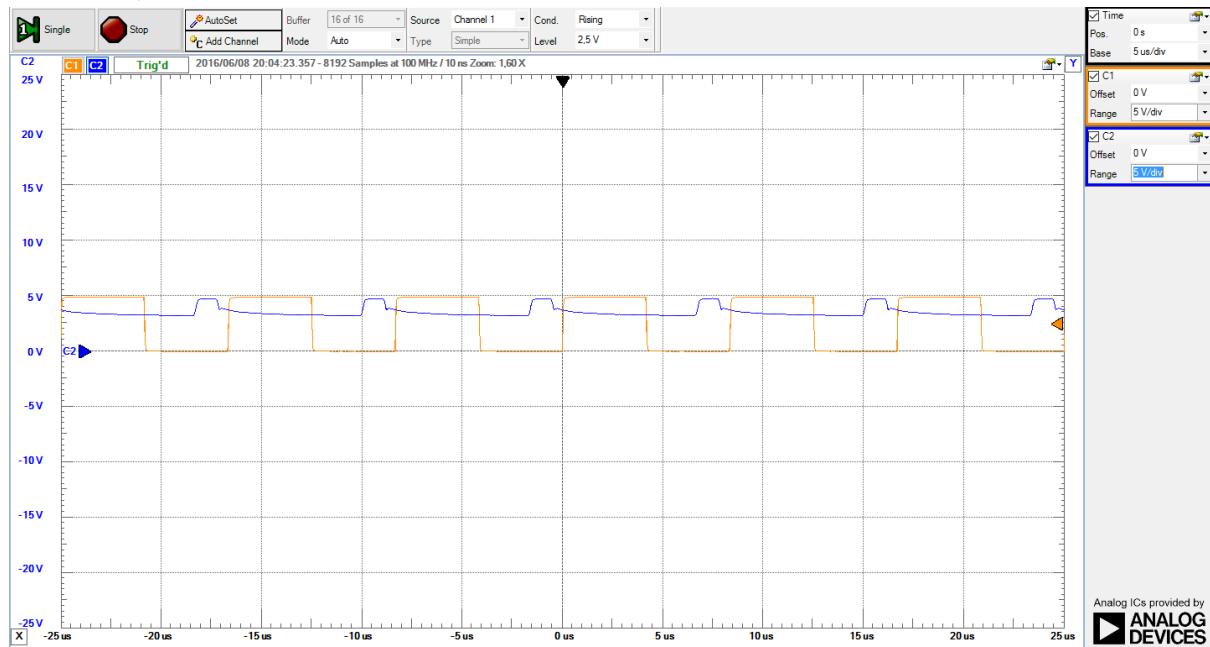


Figur 110 – Multisim diagram over sender og modtager kredsløb



Figur 111 – Simulering af sender og modtager kredsløb i Multisim

#### 4.5.2 Implementation



Figur 112 – Osciloskop billede af sender og modtager-kredsløbet hvor gul er 120 kHz firkant indgangssignal på sender kredsløbet og blå er udgangssignalet på modtager

Det ses ud fra Figur 111 at realiseringen stemmer overens med simuleringen. Der fås et digitalt højt signal på udgangen af modtager kredsløbet når der transmitteres et 120 kHz firkantsignal ud på elnettet.

## 5 Accepttest (TF, SN, MB, DP, ME, AK, NT)

Accepttesten udføres på et 18 VAC lysnet.

Systemet er sat op af en PC, en styreboks (Funduino Mega2560), en kodelås (DE2-board) og to enheder tilkoblet. PC'en er tilkoblet styreboksen via USB. På både PC og styreboks, er der installeret den udviklede software. Kodelåsen er tilkoblet styreboksen via en digital forbindelse, og er forudindstillet med to koder "høj-høj-lav-lav" og "lav-lav-høj-høj". De to tilkoblede enheder kaldet "enhed 1" og "enhed 2" er konfigureret og tilføjet til hvert deres rum i PC-software. Enhed 1 er tildelt til rumID "1" og enhed 2 er tildelt rumID "2". Enhed 1 tildeles en tidsplan gældende for Mandag, med start tidspunkt 12:00 og slut tidspunkt 13:00.

### 5.1 Test af Usecases

#### 5.1.1 Use Case 1: Opstart af system

Use case under test		Use case 1: Opstart af system		
Scenarie		Hovedscenarie Her testes samtidig for kravene: 2.1, 3.1.		
Forudsætninger		Styreboks og PC er forbundet korrekt. SerialCOM er installeret på PC.		
Step	Handling	Forventet observation/resultat	Faktisk observation/resultat	Vurdering (OK/FAIL)
1	Dobbeltklik på softwareikon på skrivebordet for at starte PC softwareen.	PC softwaren starter op og anmoder brugeren om indtastning af kode på kodelås.		
2	Indtast 1.kode "høj-høj-lav-lav" på styreboksens kodelås vha. SW(3-0), efterfulgt af tryk på Key(3).	Der observeres at LEDR(5) og LEDR(4) lyser på styreboksens kodelås.		
3	Indtast 2.kode "lav-lav-høj-høj" på styreboksens kodelås vha. SW(3-0), efterfulgt af tryk på Key(3).	Der observeres at LEDR(6) og LEDR(4) lyser på styreboksens kodelås.		
4	Herefter trykkes der godkend på PC softwareen vinduet.	PC software skifter til forsiden af den grafiske brugerflade og viser 5 knapper.		

<b>Use case under test</b>	Use case 1: Opstart af system			
<b>Scenarie</b>	Udvidelse 1: Software kører allerede			
<b>Forudsætninger</b>	Styreboks og PC er forbundet korrekt, en version af PC softwaren er allerede startet på PC'en.			
Step	Handling	Forventet observation/resultat	Faktisk observation/resultat	Vurdering (OK/FAIL)
1	Dobbeltklik på ikon på skrivebordet for at starte PC softwaren.	Der observeres at intet yderlige vindue af PC softwaren opstartes.		

<b>Use case under test</b>	Use case 1: Opstart af system			
<b>Scenarie</b>	Udvidelse 2: Forkert kode indtastet			
<b>Forudsætninger</b>	Bruger er i gang med Usecase 1 hovedscenarie, og er nået til punkt 2			
Step	Handling	Forventet observation/resultat	Faktisk observation/resultat	Vurdering (OK/FAIL)
1	Indtast kode "høj-høj-høj-høj" på styreboksen kodelås vha. SW(3-0), efterfulgt af tryk på Key(3).	Der observeres at LEDR(4) og LEDR(2) lyser på styreboksen kodelås..		
2	Indtast kode "høj-høj-høj-høj" på styreboksen kodelås vha. SW(3-0), efterfulgt af tryk på Key(3).	Der observeres at LEDR(4) og LEDR(3) lyser på styreboksen kodelås.		
3	Indtast kode "høj-høj-høj-høj" på styreboksen kodelås vha. SW(3-0), efterfulgt af tryk på Key(3).	Der observeres at LEDR(7), LEDR(3) og LEDR(2) lyser på styreboksen kodelås.		
2	Herefter trykkes der godkend på PC softwaren vinduet.	Der observeres at PC softwaren ikke skifter vindue.		

### 5.1.2 Use Case 2: Status Forespørgsel

<b>Use case under test</b>	Use case 2: Status forespørgsel			
<b>Scenarie</b>	Hovedscenarie			
<b>Forudsætninger</b>	Styrebooks og PC er forbundet korrekt, Use case 1 er gennemført, enhed konfigureret til systemet			
Step	Handling	Forventet observation/resultat	Faktisk observation/resultat	Vurdering (OK/FAIL)
1	Tryk på "Opdater enhedstatus" på den grafiske brugerflade	PC softwaren henter status på enhederne og viser statusoversigt.		

### 5.1.3 Use Case 3: Tilføjelse af enhed

<b>Use case under test</b>	Use case 3: Tilføjelse af enhed			
<b>Scenarie</b>	Hovedscenarie Her testes samtidig for krav: 1.1			
<b>Forudsætninger</b>	Use case 1 er udført			
Step	Handling	Forventet observation/resultat	Faktisk observation/resultat	Vurdering (OK/FAIL)
1	Tilslut enhed til lysnettet ved at koble stik fra enhed til stikkontakt	Enhed tilsluttet korrekt lysnettet gennem stikkontakt		
2	Tryk på "Tilføj enhed" i den grafiske brugerflade.	Skærm på PC viser skærmvindue med formular til tilføjelse af enhed		
3	Indtast "3" i feltet "Indtast enhedens ID".	Der observeres at ID "3" står i feltet "Indtast enhedens ID"		
4	Indtast "1" i feltet "Tilføj enheden til et rum"	Der observeres at "1" står i feltet "Tilføj enheden til et rum"		
5	Tryk på knappen "Tilføj enhed"	Skærm på PC viser "Tilføjelse af enheden succesfuld".		
6	Tryk på knappen "OK"	"Tilføj enhed" vinduet lukker. PC vinduet skifter til forsiden, hvor der observeres at den nye enhed er tilføjet til Enhedstabellen.		

<b>Use case under test</b>		Use Case 3: Tilføjelse af enhed		
<b>Scenarie</b>		Udvidelse 1: Brugeren tildeler ikke et rum		
<b>Forudsætninger</b>		Bruger er i gang med at udføre use case 3: "Tilføjelse af enhed" og er nået til punkt 4 i hovedscenarie		
Step	Handling	Forventet observation/resultat	Faktisk observation/resultat	Vurdering (OK/FAIL)
1	Tryk "Tilføj enhed" på PC skærm uden at tildele rumID til enheden	Skærm på PC viser "Tilføjelse af enheden succesfuld".		
2	Tryk på knappen "OK"	"Tilføj enhed" vinduet lukker. PC vinduet skifter til forsiden, hvor der observeres at den nye enhed er tilføjet til Enhedstabellen med Rum ID "0".		

<b>Use case under test</b>	Use Case 3: Tilføjelse af enhed			
<b>Scenarie</b>	Udvidelse 3: Brugerens annullering af indtastningen			
<b>Forudsætninger</b>	Bruger er i gang med at udføre use case 3: "Tilføjelse af enhed" og er nået til punkt 6: "Bruger trykker OK" i hovedscenarie			
Step	Handling	Forventet observation/resultat	Faktisk observation/resultat	Vurdering (OK/FAIL)
1	Tryk "Annuler" på PC skærm	Indtastningsvinduet lukkes. PC Skærm viser hovedmenu.		

<b>Use case under test</b>	Use Case 3: Tilføjelse af enhed			
<b>Scenarie</b>	Udvidelse 4: ID er allerede registreret			
<b>Forudsætninger</b>	Bruger er i gang med at udføre use case 3: "Tilføjelse af enhed" og er nået til punkt 3 i hovedscenarie			
Step	Handling	Forventet observation/resultat	Faktisk observation/resultat	Vurdering (OK/FAIL)
1	Skriv "1" i feltet "Indtast enhedens ID"	Feltet til "Indtast enhedens ID" opdateres til "1"		
2	Tryk på "Tilføj Enhed"	Skærm på PC viser "Tilføjelse af enheden fejlede"		
3	Tryk på "Ok"	Skærm på PC viser hovedmenu.		

<b>Use case under test</b>	Use case 3: Tilføjelse af enhed			
<b>Scenarie</b>	Enheden tildeles ugyldig ID			
<b>Forudsætninger</b>	Bruger er i gang med at udføre use case 3: "Tilføjelse af enhed" og er nået til punkt 6 i hovedscenarie			
<b>Step</b>	<b>Handling</b>	<b>Forventet observation/resultat</b>	<b>Faktisk observation/resultat</b>	<b>Vurdering (OK/FAIL)</b>
3	Indtast "256" i feltet "Indtast enhedens ID".	Der observeres at det ikke er muligt at indtaste ID "256"		
6	Tryk "Annuler" på PC skærm	Indtastningsvinduet lukkes. Skærm på PC software viser hovedmenu.		

#### 5.1.4 Use Case 4: Fjernelse af enhed

<b>Use case under test</b>		Use Case 4: Fjernelse af enhed		
<b>Scenarie</b>		Hovedscenarie		
<b>Forudsætninger</b>		Use case 1 er udført og minimum 1 enhed er tilføjet til systemet		
Step	Handling	Forventet observation/resultat	Faktisk observation/resultat	Vurdering (OK/FAIL)
1	Tryk på "Fjern Enhed" i PC Softwaren.	Skærm på PC viser skærmvindue med oversigt over kendte enheder		
2	Tryk på enhed med ID "2".	Den valgte enhed markeres.		
3	Tryk på "Fjern Enhed"	Skærm på PC viser "Enheden er blevet fjernet".		
4	Tryk på "OK".	"Enheden er blevet fjernet" besked lukkes og tabellen over kendte enheder opdateres.		

<b>Use case under test</b>	Use Case 4: Fjernelse af enhed			
<b>Scenarie</b>	Udvidelse 1: Brugerens annullerer			
<b>Forudsætninger</b>	Bruger er i gang med at udføre use case 4: "Fjernelse af enhed" og er nået til punkt 3: "Brugerens trykker Fjern enhed" i hovedscenarie			
<b>Step</b>	<b>Handling</b>	<b>Forventet observation/resultat</b>	<b>Faktisk observation/resultat</b>	<b>Vurdering (OK/FAIL)</b>
1	Tryk "Annuler" i PC Softwaren.	Indtastningsvinduet lukkes. Skærm på PC viser hovedmenu.		

<b>Use case under test</b>	Use Case 4: Fjernelse af enhed			
<b>Scenarie</b>	Der trykkes fjern enhed uden en enhed er valgt			
<b>Forudsætninger</b>	Use case 1 er udført og minimum 1 enhed er tilføjet til systemet			
<b>Step</b>	<b>Handling</b>	<b>Forventet observation/resultat</b>	<b>Faktisk observation/resultat</b>	<b>Vurdering (OK/FAIL)</b>
1	Tryk på "Fjern Enhed" i PC Softwaren.	Skærm på PC viser skærmvindue med oversigt over kendte enheder		
2	Tryk på "Fjern Enhed"	Skærm på PC viser "Ingen enhed fjernet"		
3	Tryk på "OK"	Skærm på PC returnere til skærmvindue med oversigt over kendte enheder.		

### 5.1.5 Use Case 5: Rediger Enhed

<b>Use case under test</b>		Use Case 5: Rediger enhed		
<b>Scenarie</b>		Hovedscenarie		
<b>Forudsætninger</b>		Use case 1 er udført og mindst en enhed er tilføjet til systemet		
Step	Handling	Forventet observation/resultat	Faktisk observation/resultat	Vurdering (OK/FAIL)
1	Tryk på "Rediger enhed" i PC Softwaren.	Skærm på PC viser skærmvindue med oversigt over kendte enheder		
2	Tryk på enhed med ID "2".	Enhed med ID "2" markeres i tabellen. Feltet "Nuværende Enheds ID" opdateres med ID "2". Feltet "Nuværende Rum ID" opdateres med ID "2"		
3	Skriv "4" i feltet "Nyt Enheds ID"	Feltet "Nyt Enheds ID" opdateres med "4"		
4	Skriv "3" i feltet "Nyt Rum ID"	Feltet "Nyt Rum ID" opdateres med "3"		
5	Tryk på "Gem Ändringer".	Skærmen på PC viser "Enhed: 2 opdateret til: Enheds ID: 4 Rum ID: 3".		
6	Tryk på "OK"	Tabellen opdateres og enhed med ID "4" og rum "3" vises.		

<b>Use case under test</b>	Use Case 5: Rediger enhed			
<b>Scenarie</b>	Bruger indtaster ugyldig værdi			
<b>Forudsætninger</b>	Use case 1 er udført og mindst en enhed er tilføjet til systemet			
Step	Handling	Forventet observation/resultat	Faktisk observation/resultat	Vurdering (OK/FAIL)
1	Tryk på "Ret enhed" i PC Softwaren.	Skærm på PC viser skærmvindue med oversigt over kendte enheder		
2	Tryk på enhed med ID "2".	Enhed med ID "2" markeres i tabellen. Feltet "Nuværende Enheds ID" opdateres med ID "2". Feltet "Nuværende Rum ID" opdateres med ID "2"		
3	Skriv "test" i feltet "Nyt Enheds ID"	Feltet forbliver tomt		
4	Skriv "test" i feltet "Nyt Rum ID"	Feltet forbliver tomt		
5	Tryk på "Gem Ændringer".	Skærmen på PC viser "Ingen enheds ændringer registreret".		
6	Tryk på "OK"	Tabellen vises uændret.		

<b>Use case under test</b>	Use Case 5: Rediger enhed			
<b>Scenarie</b>	Bruger ændre Enheds ID men ikke Rum ID			
<b>Forudsætninger</b>	Use case 1 er udført og mindst en enhed er tilføjet til systemet			
<b>Step</b>	<b>Handling</b>	<b>Forventet observation/resultat</b>	<b>Faktisk observation/resultat</b>	<b>Vurdering (OK/FAIL)</b>
1	Tryk på "Ret enhed" i PC Softwaren.	Skærm på PC viser skærmvindue med oversigt over kendte enheder		
2	Tryk på enhed med ID "2".	Enhed med ID "2" markeres i tabellen. Feltet "Nuværende Enheds ID" opdateres med ID "2". Feltet "Nuværende Rum ID" opdateres med ID "2"		
3	Skriv "3" i feltet "Nyt Enheds ID"	Feltet "Nyt Enheds ID" opdateres til "3"		
4	Efterlad feltet "Nyt Rum ID" tomt	Feltet forbliver tomt		
5	Tryk på "Gem Ændringer".	Skærmen på PC viser: "Enhed: 2 opdateret til Enheds ID: 3 Rum ID: 2"		
6	Tryk på "OK"	Tabellen opdateres og enhed med ID "3" og rum "2" vises.		

<b>Use case under test</b>	Use Case 5: Rediger enhed			
<b>Scenarie</b>	Bruger ændre Rum ID men ikke Enheds ID			
<b>Forudsætninger</b>	Use case 1 er udført og mindst en enhed er tilføjet til systemet			
Step	Handling	Forventet observation/resultat	Faktisk observation/resultat	Vurdering (OK/FAIL)
1	Tryk på "Ret enhed" i PC Softwaren.	Skærm på PC viser skærmvindue med oversigt over kendte enheder		
2	Tryk på enhed med ID "2".	Enhed med ID "2" markeres i tabellen. Feltet "Nuværende Enheds ID" opdateres med ID "2". Feltet "Nuværende Rum ID" opdateres med ID "2"		
3	Efterlad feltet "Nyt Enheds ID" tomt	Feltet forbliver tomt		
4	Indtast "3" i feltet "Nyt Rum ID"	Feltet "Nyt Rum ID" opdateres til "3"		
5	Tryk på "Gem Ændringer".	Skærmpen på PC viser: "Enhed: 2 opdateret til Enheds ID: 2 Rum ID: 3"		
6	Tryk på "OK"	Tabellen opdateres og enhed med ID "2" og rum "3" vises.		

<b>Use case under test</b>	Use Case 5: Rediger enhed			
<b>Scenarie</b>	Bruger vælger ingen enhed til ændring			
<b>Forudsætninger</b>	Use case 1 er udført og mindst en enhed er tilføjet til systemet			
<b>Step</b>	<b>Handling</b>	<b>Forventet observation/resultat</b>	<b>Faktisk observation/resultat</b>	<b>Vurdering (OK/FAIL)</b>
1	Tryk på "Rediger enhed" i PC Softwaren.	Skærm på PC viser skærmvindue med oversigt over kendte enheder		
2	Tryk på "Gem Ændringer"	Skærm på PC viser "Ingen enhed valgt til ændring"		
3	Tryk på "Ok"	Tabellen vises uændret		

<b>Use case under test</b>	Use Case 5: Ret enhed			
<b>Scenarie</b>	Bruger annullerer			
<b>Forudsætninger</b>	Use case 1 er udført og mindst en enhed er tilføjet til systemet			
<b>Step</b>	<b>Handling</b>	<b>Forventet observation/resultat</b>	<b>Faktisk observation/resultat</b>	<b>Vurdering (OK/FAIL)</b>
1	Tryk på "Ret enhed" i PC Softwaren.	Skærm på PC viser skærmvindue med oversigt over kendte enheder		
2	Tryk på "Annuler"	Skærm på PC returnerer til hovedmenuen.		

### 5.1.6 Use Case 6: Ændring af tidsplan

<b>Use case under test</b>		Use Case 6: Ændring af tidsplan		
<b>Scenarie</b>		Hovedscenarie		
<b>Forudsætninger</b>		Use case 1 er udført og der er minimum en enhed, med en tidsplan tilføjet til systemet.		
Step	Handling	Forventet observation/resultat	Faktisk observation/resultat	Vurdering (OK/FAIL)
1	Tryk på "Tidsplan" i den grafiske brugerflade.	Skærm på PC viser skaermvindue med oversigt over kendte enheder		
2	Tryk på enhed med ID "1"	Enhed med ID "1" markeres		
3	Tryk på "Rediger eksisterende tidsplan"	Vinduet for redigering af eksisterende tidsplan vises.		
4	Tryk på den øverste tidsplan.	Den øverste tidsplan markeres.		
5	Vælg "Onsdag" i boksen "Hvilken dag ønskes indstillet"	Onsdag vises i boksen "Hvilken dag ønskes indstillet"		
6	Vælg "13:00" i feltet "Vælg nyt start tidspunkt"	"13:00" vises i feltet "Vælg nyt start tidspunkt"		
7	Vælg "15:00" i feltet "Vælg nyt slut tidspunkt"	"15:00" vises i feltet "Vælg nyt slut tidspunkt"		
8	Tryk "Gem ændringer"	Skærm på PC viser "Tidsplanen er nu opdateret"		
9	Tryk "Ok"	Skærm på PC viser hovedmenu		

<b>Use case under test</b>	Use Case 6: Ændring af tidsplan			
<b>Scenarie</b>	Brugerens annulerer indtastningen			
<b>Forudsætninger</b>	Use case 1 er udført og der er minimum en enhed, med en tidsplan tilføjet til systemet.			
<b>Step</b>	<b>Handling</b>	<b>Forventet observation/resultat</b>	<b>Faktisk observation/resultat</b>	<b>Vurdering (OK/FAIL)</b>
1	Tryk på "Tidsplan" i den grafiske brugerflade.	Skærm på PC viser skærmvindue med oversigt over kendte enheder		
2	Tryk på enhed med ID "1"	Enhed med ID "1" markeres		
3	Tryk på "Rediger eksisterende tidsplan"	Vinduet for redigering af eksisterende tidsplan vises.		
4	Tryk på "Annuler"	Skærm på PC viser hovedmenuen		

<b>Use case under test</b>		Use Case 6: Ændring af tidsplan		
<b>Scenarie</b>		Bruger indtaster sluttidspunkt før starttidspunkt		
<b>Forudsætninger</b>		Use case 1 er udført og der er minimum en enhed, med en tidsplan tilføjet til systemet.		
Step	Handling	Forventet observation/resultat	Faktisk observation/resultat	Vurdering (OK/FAIL)
1	Tryk på "Tidsplan" i den grafiske brugerflade.	Skærm på PC viser skærmvindue med oversigt over kendte enheder		
2	Tryk på enhed med ID "1"	Enhed med ID "1" markeres		
3	Tryk på "Rediger eksisterende tidsplan"	Vinduet for redigering af eksisterende tidsplan vises.		
4	Tryk på den øverste tidsplan.	Den øverste tidsplan markeres.		
5	Vælg "Onsdag" i boksen "Hvilken dag ønskes indstillet"	Onsdag vises i boksen "Hvilken dag ønskes indstillet"		
6	Vælg "15:00" i feltet "Vælg nyt start tidspunkt"	"15:00" vises ifeltet "Vælg nyt start tidspunkt"		
7	Vælg "13:00" i feltet "Vælg nyt slut tidspunkt"	"13:00" vises i feltet "Vælg nyt slut tidspunkt"		
8	Tryk "Gem ændringer"	Skærm på PC viser "Ugyldigt tidspunkt – Sluttidspunkt før starttidspunkt"		
9	Tryk "Ok"	Skærm på PC viser vindue for redigering af eksisterende tidsplan.		

<b>Use case under test</b>	Use Case 6: Ændring af tidsplan			
<b>Scenarie</b>	Bruger vælger ikke en enhed at ændre tidsplan for			
<b>Forudsætninger</b>	Use case 1 er udført og der er minimum en enhed, med en tidsplan tilføjet til systemet.			
Step	Handling	Forventet observation/resultat	Faktisk observation/resultat	Vurdering (OK/FAIL)
1	Tryk på "Tidsplan" i den grafiske brugerflade.	Skærm på PC viser skærmvindue med oversigt over kendte enheder		
2	Tryk på "Rediger eksisterende tidsplan"	PC skærm viser "Ingen enhed valgt til ændring."		
3	Tryk på "Ok"	PC Skærm viser vinduet for tidsplaner		

<b>Use case under test</b>	Use Case 6: Ændring af tidsplan			
<b>Scenarie</b>	Den valgte enhed har ingen tidsplaner at ændre			
<b>Forudsætninger</b>	Use case 1 er udført og der er minimum en enhed uden en tidsplan, tilføjet til systemet.			
Step	Handling	Forventet observation/resultat	Faktisk observation/resultat	Vurdering (OK/FAIL)
1	Tryk på "Tidsplan" i den grafiske brugerflade.	Skærm på PC viser skærmvindue med oversigt over kendte enheder		
2	Tryk på enhed med ID "1"	Enhed med ID "1" markeres		
3	Tryk på "Rediger eksisterende tidsplan"	Skærm på PC viser "Ingen tidsplan at ændre"		
4	Tryk på "Ok"	Skærm på PC viser vinduet for tidsplaner		

### 5.1.7 Use Case 7: Kør Simulering

<b>Use case under test</b>		Use case 7: Kør simulering		
<b>Scenarie</b>		Hovedscenarie samt Udvidelse 1. Her testes samtidig for kravene: 2.3, 2.4 3.8, 3.9		
<b>Forudsætninger</b>		Én lampe enhed er tilføjet til systemet, og konfigureret til at skifte tilstand med 2 minutters interval.		
Step	Handling	Forventet observation/resultat	Faktisk observation/resultat	Vurdering (OK/FAIL)
1	Tilslut styreboks til lysnettet.	<p>Der observeres visuelt at indikatoren for tændt styreboks lyser grønt.</p> <p>Der observeres visuelt at lampen skifter tilstand med 2 minutters interval.</p> <p>Der observeres visuelt at indikatoren for data transmission lyser gult umiddelbart før lampen skifter tilstand.</p>		

### 5.1.8 Use Case 8: Fjern tidsplan

<b>Use case under test</b>		Use case 8: Fjern tidsplan		
<b>Scenarie</b>		Hovedscenarie		
<b>Forudsætninger</b>		Use case 1 er udført og der er minimum en enhed med en tidsplan tilføjet til systemet		
Step	Handling	Forventet observation/resultat	Faktisk observation/resultat	Vurdering (OK/FAIL)
1	Tryk på "tidsplan"	<i>Skærm på PC viser skærmvindue med tidsplan</i>		
2	Tryk på enhed med UnitID "1" i tabellen	<i>Enhed med ID "1" markeres på skærmen</i>		
3	Tryk på "Fjern tidsplan"	<i>Skærm på PC viser skærmvindue med fjern tidsplan</i>		
4	Vælg øverste tidsplan i tabellen	<i>Øverste tidsplan markeres på skærmen</i>		
5	Tryk på "Fjern valgte tidsplan"	<i>Skærm på PC viser "Tidsplanen er fjernet".</i>		
6	Tryk på knappen "OK"	<i>"Fjern tidsplan" vinduet lukker. PC vinduet skifter til hovedmenuen.</i>		

<b>Use case under test</b>	Use case 8: Fjern tidsplan			
<b>Scenarie</b>	Fjern tidsplaner tilføjet til specifik dag			
<b>Forudsætninger</b>	Use case 1 er udført og der er minimum en enhed med en tidsplan aktiv mandag tilføjet til systemet			
Step	Handling	Forventet observation/resultat	Faktisk observation/resultat	Vurdering (OK/FAIL)
1	Tryk på "tidsplan"	<i>Skærm på PC viser skærmvindue med tidsplan</i>		
2	Tryk på enhed med UnitID "1" i tabellen	<i>Enhed med ID "1" markeres på skærmen</i>		
3	Tryk på "Fjern tidsplan"	<i>Skærm på PC viser skærmvindue med fjern tidsplan</i>		
4	Vælg "mandag" i comboboxen	<i>Observeres at dagen står i comboboxen.</i>		
5	Tryk på "Fjern tidsplan for valgte dage"	<i>Skærm på PC viser "Dagen er ryddet".</i>		
6	Tryk på knappen "OK"	<i>"Fjern tidsplan" vinduet lukker. PC vinduet skifter til hovedmenuen.</i>		

<b>Use case under test</b>		Use case 8: Fjern tidsplan		
<b>Scenarie</b>		Ingen tidsplan for valgte enhed		
<b>Forudsætninger</b>		Use case 1 er udført og der er minimum en enhed uden nogen tidsplaner.		
Step	Handling	Forventet observation/resultat	Faktisk observation/resultat	Vurdering (OK/FAIL)
1	Tryk på "tidsplan"	<i>Skærm på PC viser skærmvindue med tidsplan</i>		
2	Tryk på enhed med UnitID "1" i tabellen	<i>Enhed med ID "1" markeres på skærmen</i>		
3	Tryk på "Fjern tidsplan"	<i>Skærm på PC viser skærmvindue med fejlmeddelse "Ingen tidsplan at fjerne"</i>		
4	Tryk på knappen "OK"	<i>Skærm på PC viser skærmvindue med tidsplan</i>		

<b>Use case under test</b>		Use case 8: Fjern tidsplan		
<b>Scenarie</b>		Bruger annullerer		
<b>Forudsætninger</b>		Use case 1 er udført og der er minimum en enhed med en tidsplan aktiv mandag tilføjet til systemet		
Step	Handling	Forventet observation/resultat	Faktisk observation/resultat	Vurdering (OK/FAIL)
1	Tryk på "tidsplan"	<i>Skærm på PC viser skærmvindue med tidsplan</i>		
2	Tryk på enhed med UnitID "1" i tabellen	<i>Enhed med ID "1" markeres på skærmen</i>		
3	Tryk på "annuller"	<i>Indtastningsvinduet lukkes. Skærm på PC viser hovedmenu.</i>		

### 5.1.9 Use Case 9: Tilføjelse af Tidsplan

<b>Use case under test</b>		Use Case 9: Tilføjelse af tidsplan		
<b>Scenarie</b>		Hovedscenarie		
<b>Forudsætninger</b>		Use case 1 er udført og der er minimum en enhed tilføjet til systemet		
Step	Handling	Forventet observation/resultat	Faktisk observation/resultat	Vurdering (OK/FAIL)
1	Tryk på "Tidsplan" i den grafiske brugerflade.	Skærm på PC viser skærmvindue med oversigt over kendte enheder		
2	Tryk på enhed med ID "1".	Enhed med ID "1" markeres på skærmen		
3	Tryk "Tilføj Tidsplan"	Vindue til tilføjelse af tidsplan vises		
4	Vælg "Mandag" fra drop down menu "Hvilken dag ønskes indstillet"	"Mandag" vises i drop down menuen.		
5	Vælg start tidspunkt "14:00"	Start tidspunkt vises som "14:00"		
6	Væl slut tudspunkt "15:00"	Slut tidspunkt viser "15:00"		
7	Tryk "Tilføj Tidsplan"	Skærm på PC viser "Tilføjelse af tidsplan godkendt."		
8	Tryk "OK"	Skærm på PC viser hovedmenu.		

<b>Use case under test</b>	Use Case 9: Tilføjelse af tidsplan			
<b>Scenarie</b>	Udvidelse 2: Brugerens vælger et sluttidspunkt før det valgte starttidspunkt			
<b>Forudsætninger</b>	Use case 1 er udført og der er minimum en enhed tilføjet til systemet			
<b>Step</b>	<b>Handling</b>	<b>Forventet observation/resultat</b>	<b>Faktisk observation/resultat</b>	<b>Vurdering (OK/FAIL)</b>
1	Tryk på "Tidsplan" i den grafiske brugerflade.	Skærm på PC viser skærmvindue med oversigt over kendte enheder		
2	Tryk på enhed med ID "1".	Enhed med ID "1" markeres på skærmen		
3	Tryk "Tilføj Tidsplan"	Vindue til tilføjelse af tidsplan vises		
4	Vælg "Mandag" fra drop down menu "Hvilken dag ønskes indstillet"	"Mandag" vises i drop down menuen.		
5	Vælg start tidspunkt "15:00"	Start tidspunkt vises som "15:00"		
6	Vælg slut tidspunkt "13:00"	Slut tidspunkt viser "13:00"		
7	Tryk "Tilføj Tidsplan"	Skærm på PC viser "Tilføjelse af tidsplan fejlede."		
8	Tryk "OK"	Skærm på PC returnere til hovedmenuen.		

<b>Use case under test</b>	Use Case 9: Tilføjelse af tidsplan			
<b>Scenarie</b>	Brugerens har ikke valgt en enhed at tilføje tidsplan til			
<b>Forudsætninger</b>	Use case 1 er udført og der er minimum en enhed tilføjet til systemet			
<hr/>				
<b>Step</b>	<b>Handling</b>	<b>Forventet observation/resultat</b>	<b>Faktisk observation/resultat</b>	<b>Vurdering (OK/FAIL)</b>
1	Tryk på "Tidsplan" i den grafiske brugerflade.	Skærm på PC viser skærmvindue med oversigt over kendte enheder		
2	Tryk på "Tilføj Tidsplan"	Skærm på PC viser "Ingen enhed valg til tilføjelse af tidsplan"		
3	Tryk på "OK"	Vinduet for tidsplaner vises.		

<b>Use case under test</b>	Use Case 9: Tilføjelse af tidsplan			
<b>Scenarie</b>	Brugerens annulerer tilføjelse af tidsplan			
<b>Forudsætninger</b>	Use case 1 er udført og der er minimum en enhed tilføjet til systemet			
<hr/>				
<b>Step</b>	<b>Handling</b>	<b>Forventet observation/resultat</b>	<b>Faktisk observation/resultat</b>	<b>Vurdering (OK/FAIL)</b>
1	Tryk på "Tidsplan" i den grafiske brugerflade.	Skærm på PC viser skærmvindue med oversigt over kendte enheder		
2	Tryk på enhed med ID "1"	Enheden med ID "1" markeres		
3	Tryk på "Tilføj Tidsplan"	Vinduet for tilføjelse af tidsplan vises		
4	Tryk på "Annuler"	Skærm på PC returnerer til hovedmenuen		

## 5.2 Test af yderligere tekniske krav.

<b>Krav 2.2 &amp; 2.5</b>		Skal kunne genstarte i tilfælde af <kritiske systemfejl>, og vise vigtige beskeder om systemets status på LCD skærmen.		
Step	Handling	Forventet observation/resultat	Faktisk observation/resultat	Vurdering (OK/FAIL)
<b>1</b>	Simulering startes uden enheder tilkoblet systemet.	Styreboksen vil genstarte, og efter lidt tid vise beskeden: "System fejl: Kan ikke kommunikerer med enheder. Tjek forbindelse!" på styreboksens LCD skærm.		

<b>Krav 3.6</b>		Skal kunne håndtere op til 255 enheder		
Step	Handling	Forventet observation/resultat	Faktisk observation/resultat	Vurdering (OK/FAIL)
<b>1</b>	Enhed 1 og 2 tilsluttes	Enhed 1 og 2 er tilsluttet		
<b>2</b>	Enhed 1 tændes	Enhed 1 er tændt		
<b>3</b>	Enhed 2 tændes	Enhed 2 er tændt		
<b>4</b>	Enhed 1 slukkes	Enhed 1 er slukket		
<b>5</b>	Enhed 2 slukkes	Enhed 2 er slukket		

<b>Krav 3.2</b>		Systemet skal have en MTBF på minimum 95%		
Step	Handling	Forventet observation/resultat	Faktisk observation/resultat	Vurdering (OK/FAIL)
<b>1</b>	En enhed indstilles til at skifte tilstand med 1 times mellemrum	Enheden tidsplan modtages af systemet.		
<b>2</b>	En timer indstilles på 8 timer. Enheden observeres med en times mellemrum under test.	Enheden skifter tilstand hver time. Systemet er aktivt i minimum 7 timer og 36 minutter.		

<b>Krav 3.3</b>	Systemet bør kommunikere med op til 60 bit/s
-----------------	--

Step	Handling	Forventet observation/resultat	Faktisk observation/resultat	Vurdering (OK/FAIL)
1	En timer startes i softwaren og en <tænd kommando> sendes til en enhed.	Den gule LED på styreboksen indikere at der transmitteres data.		
2	Styreboksen modtager en svarpakke fra enheden.  Timeren stoppes.  Tiden aflæses.  Mængden af bits sendt deles med den brugte tid.	Data transmissions hastigheden udregnes til at opfylde de givne krav.		

<b>Krav 3.4</b>	Systemet skal have en svartid på maksimalt 2 minutter.
-----------------	--

Step	Handling	Forventet observation/resultat	Faktisk observation/resultat	Vurdering (OK/FAIL)
1	Benyt tiden under testen af krav 3.3 og sammenlign med svartid på maksimalt 2 minutter	Svartiden fra testen af krav 4.3 er maksimalt 2 minutter.		

<b>Krav 3.5</b>		Systemet skal kunne fungere ved tilslutning til lysnettet.		
Step	Handling	Forventet observation/resultat	Faktisk observation/resultat	Vurdering (OK/FAIL)
1	Systemet tilsluttet lysnettet.	Systemet starter.		