Systemarkitektur specifikation.

Semesterprojekt 2. semester 2016, gruppe 15.

Indholdsfortegnelse

[Systemarkitektur. 2](#_Toc449048463)

[Hardware-arkitektur 3](#_Toc449048464)

[System 3](#_Toc449048465)

[Styreboks 6](#_Toc449048469)

[Enheder 10](#_Toc449048473)

[Protokoller 14](#_Toc449048477)

[X15 14](#_Toc449048478)

[PiS (PC interface to Styreboks) 16](#_Toc449048485)

[I2C/IIC/TWI 19](#_Toc449048492)

[SPI 19](#_Toc449048493)

[Typeliste 20](#_Toc449048494)

[Software-arkitektur 21](#_Toc449048497)

[PC. 22](#_Toc449048498)

[Styreboks. 29](#_Toc449048505)

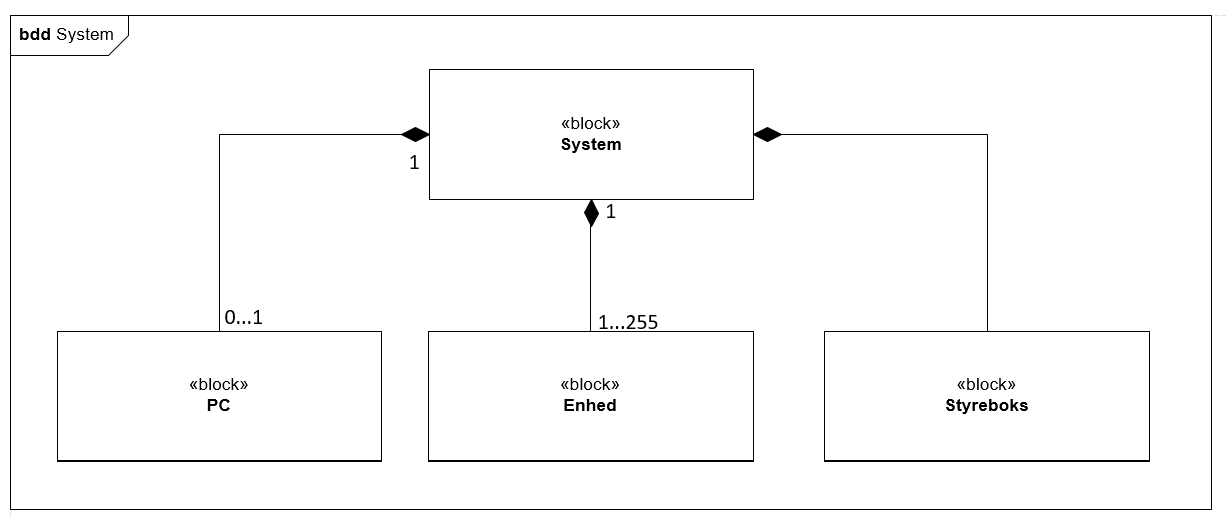
[Enhed. 35](#_Toc449048514)

# Systemarkitektur.

I det følgende beskrives arkitekturen for systemet. Systemarkitekturen bliver efterfølgende brugt som grundlag for design og implementering af systemet.

Et system består overordnet af 3 moduler herunder en styreboks, 0 til 255 enheder og eventuelt en PC, som illustreret på BDD for systemet (figur 1).

På **PC** skal brugeren kunne konfigurere systemet. Det er i kravspecifikationen lagt fast, at systemet skal være fungerende uden tilkoblet PC, det er derfor vigtigt at den af brugeren bestemte konfiguration gemmes i **styreboksen,** som kontrollerer systemets aktivitet ved normal brug.  
**Enheder** kan styres af styreboksen, via den gemte konfiguration.

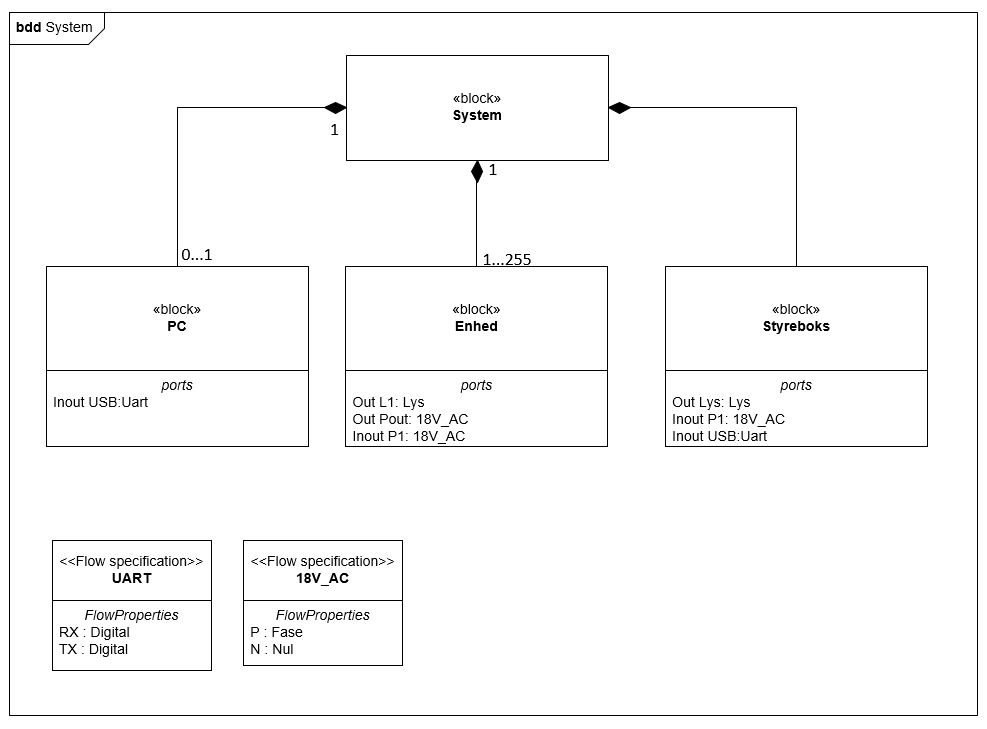


Figur - Overordnet BDD for systemet.

# Hardware-arkitektur

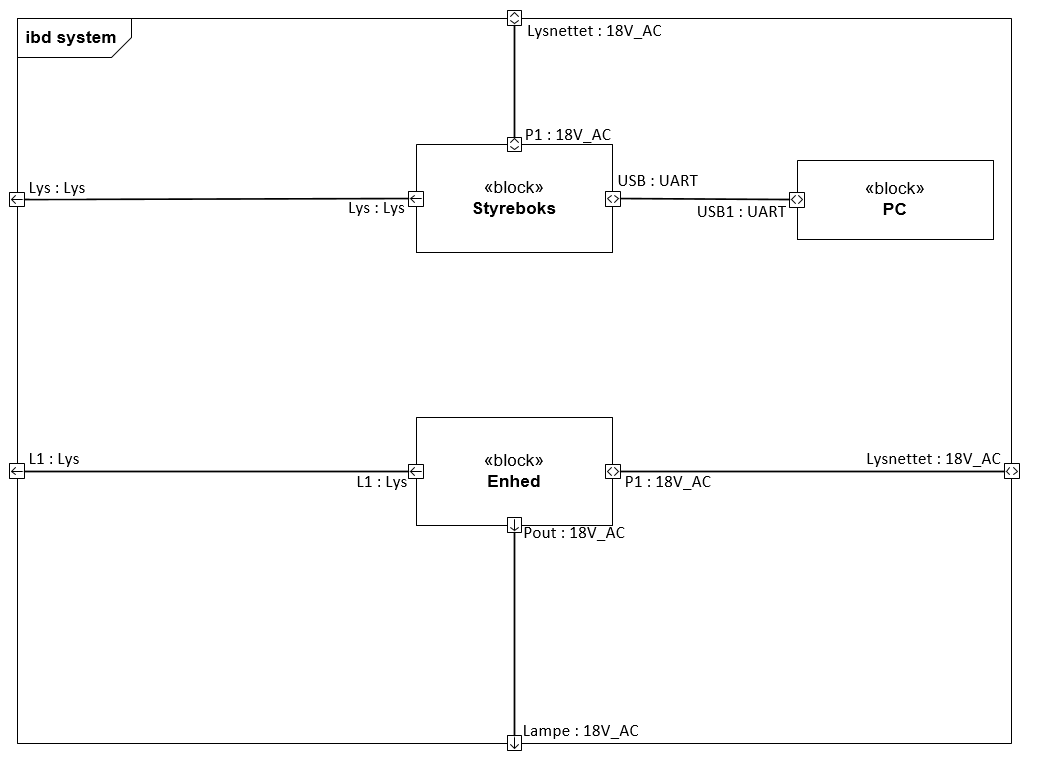
## System

Interaktionen mellem brugeren og systemet sker via en **PC**, hvor PC-softwaren til vores system er installeret. Selve indstillingen af tidplanen på vores **styreboks** sker via PC-softwaren. PC er tilsluttet styreboks via en USB-forbindelse. Via PC-softwaren er der mulighed for at indstille tidsplanen for den valgte **enhed**. Der indstilles hvornår enheden skal tænde/slukke for den lampe som er tilsluttet til enheden.



Figur - BDD system

Kommunikationen mellem styreboks og enheder foregår via. lysnettet ved hjælp af X15 protokollen.



Figur - IBD system

### Grænseflade

|  |  |  |
| --- | --- | --- |
| Grænseflade | Porte | Stiktype |
| P1 – Enhed & styreboks | Input 18V AC Hz  Input X15  Output 18 V AC 50 Hz  Output x15 | 230 V stærkstrømstik han efter dansk lovkrav |
| USB1 – Styreboks | Input 5V DC  Input max 0,5 A  Input D-  Input D+ | USB-stik type B hun (4-pin) |
| USB1 – PC | Input 5V DC  Input D-  Input D+ | USB-stik type A hun (4-pin) |
| Pout – Enhed | Output 18V AC 50 Hz | 230 V stærkstrømstik hun efter dansk lovkrav |

### Signaler til overordnet system IBD

|  |  |  |
| --- | --- | --- |
| Signaltype | Definition | Beskrivelse |
| 18V\_AC | 18V AC 50 Hz signal kombineret med et 100kHz x15 signal | 18V AC 50 Hz signal fra forsyningsnettet der også indeholder kommunikationen via x15 protokollen der udvikles specifikt til dette produkt, se protokol afsnit for yderligere information. |
| Lys | Lys i det synlige spektrum | Lys i 3 farver afhængig af hvilket LED indikator der lyser. |
| Uart | Kommunikation følger UART-standarden | Protokollen udvikles specifikt til dette produkt. Se protokol afsnit for yderligere information. |

### Signaler:

**USB: UART** er en Uart-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 x15 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 x15.

**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.

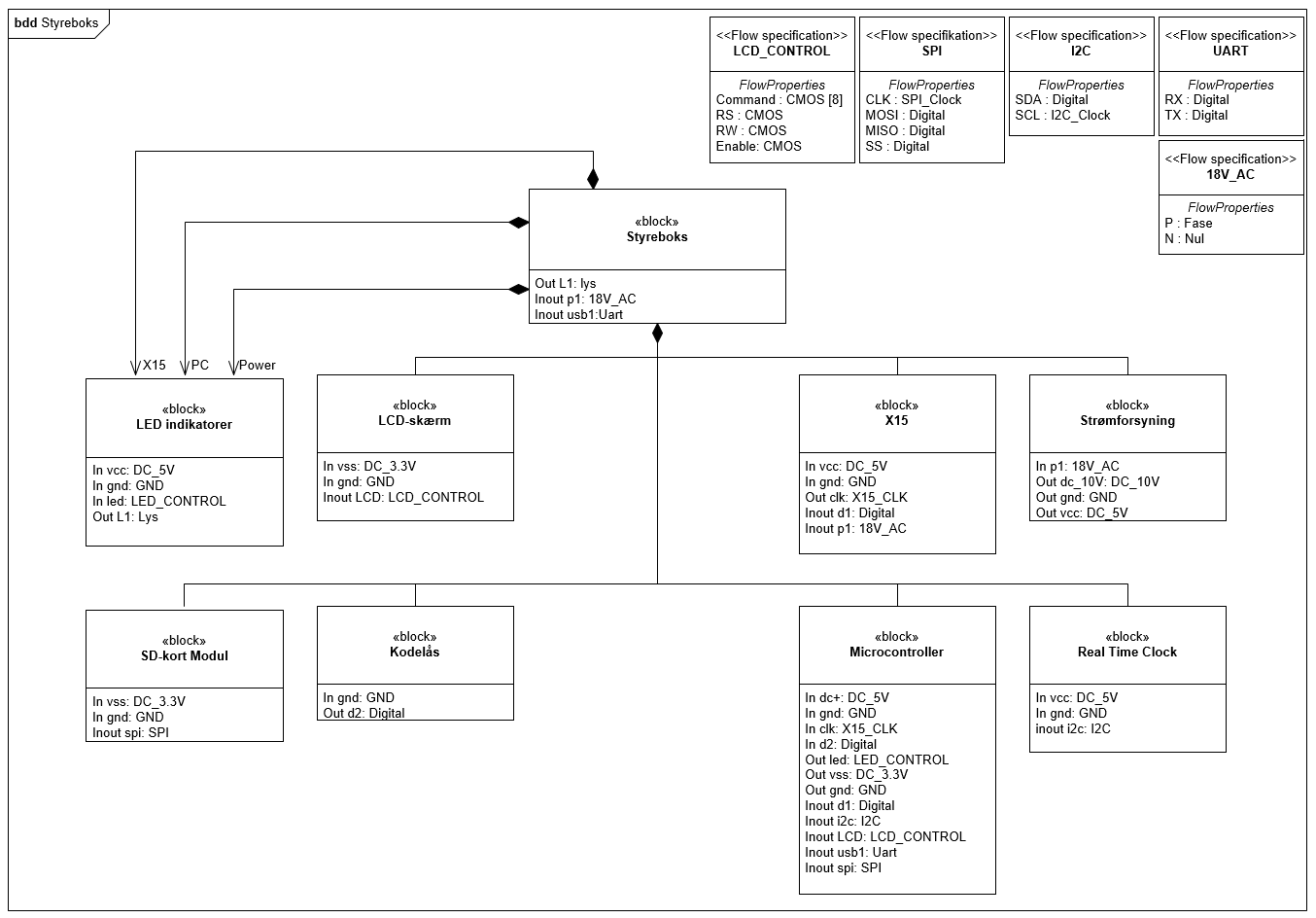
**P1: 18V\_AC** er et 18V ac 50 Hz signal som samtidig indeholder et kommunikationssignal der følger x15 protokollen. Signalet fungere som spændingsforsyning til systemets blokke samt som kommunikationssignal mellem styreboksen og enhederne via x15 protokollen.

## Styreboks

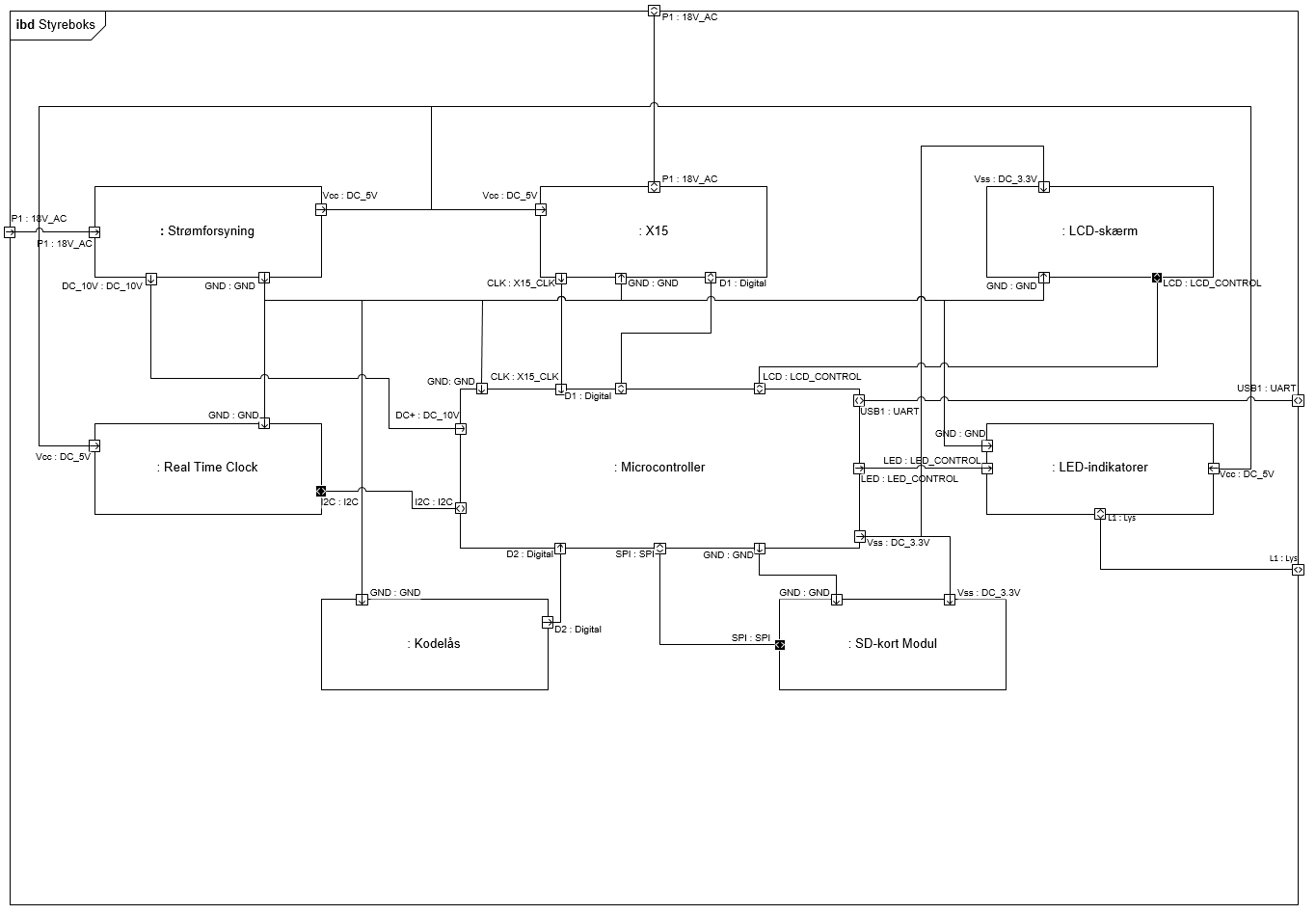
Styreboksener tilsluttet en **kodelås**,der sender et højt eller lavt signal til styreboksen, afhængig af om den indtastede kode er korrekt. Styreboksen har en **strømforsyning**, der laver en 18 VAC om til 5 VDC og 10 VDC. Styreboksenbestår af en **microcontroller**, der styrer al kommunikation mellem et modul kaldet **X15**, **Real Time Clock, SD-kort modul** og **LCD-skærm**. X15-modulet indeholder en zero-cross detektor. Real Time Clock bruges i forbindelse med tidsplanen, der holdes styr på dato og klokken. Tidsplan for enhederne gemmes på SD-kort modulet, så det gemte er tilgængelig, hvis styreboks skulle blive ramt et strømafbrud. LCD-skærmen bruges til at vise fejl meddelelser fra styreboks til bruger. Der er desuden tre **LED-indikatorer** der viser, om der sendes data fra styreboks via X15 eller om der sendes data fra styreboks til PC. Den sidste LED viser om styreboksen er tændt.

### BDD og IBD

Der er udarbejdet en BDD for styreboksen som viser styreboksen samt de moduler den består af, derudover kan der ses grænsefladerne for kommunikationen mellem styreboksens CPU og modulerne.



Figur - BDD styreboks

Ud fra styreboksens BDD er der udarbejdet følgende IBD (figur 5) hvor signaler og Porte fremgår, sammen med den forudgående BDD på figur 4 danner dette grundlaget for den videre hardwareudvikling.

Figur - IBD styreboks

### Grænseflade

|  |  |  |
| --- | --- | --- |
| Grænseflade | Porte | Stiktype |
| P1 – X15 | Input 18 V AC 50 Hz  Input X15  Output 18 V AC 50 Hz  Output X15 |  |
| P1 – Strømforsyning | Input 18 V AC 50 Hz  Output 18 V AC 50 Hz | 230 V stærkstrømstik han efter dansk lovkrav |
| LED – LED-indikatorer | Input L0 til L2 min 0 - 5 V DC |  |
| I2C – Real Time Clock | Input SCA 0 – 5 V DC  Input SCL 0 – 5 V DC  Output SCA 0 – 5 V DC  Output SCL 0 – 5 V DC |  |
| SPI – SD-modul | Input CLK 1 MHz  Input SS 0 – 3.3 V DC  Input MOSI 0 – 3.3 V DC  Output MISO 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 SCA 0 – 5 V DC  Input SCL 0 – 5 V DC  Output SCA 0 – 5 V DC  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  Output MOSI 0 – 3.3 V DC  Input MISO 0 – 3.3 V DC |  |

### Signaler til styreboks IBD

|  |  |  |
| --- | --- | --- |
| Signaltype | Definition | Beskrivelse |
| 18V\_AC | 18V AC 50 Hz signal kombineret med et 100kHz x15 signal | 18V AC 50 Hz signal fra forsyningsnettet der også indeholder kommunikationen via x15 protokollen der udvikles specifikt til dette produkt, se protokol afsnit for yderligere information. |
| Lys | Lys i det synlige spektrum | Lys i 3 farver afhængig af hvilket LED indikator der lyser. |
| DC\_10V | +10V DC signal | Forsyningsspænding til vores arduino mega2560 microcontroller der har egen dc converter monteret.  Signalet er et 10V DC signal +/- 0,5V |
| DC\_5V | +5V DC signal | Forsyningsspænding til moduler der har behov for 5V dc spænding.  Signalet er et 5V DC signal +/- 0,5V |
| DC\_3.3V | +3.3V DC signal | Forsyningsspænding til moduler der har behov for 3.3V dc spænding  Signalet er et 3.3V DC signal +/- 0,2V |
| GND | 0V DC signal | Reference spænding til DC forsyningsspændinger. |
| Digital | 0-5V DC signal | Digitalt signal  3V til 5V = 1  0V til 1,5V= 0 |
| X15\_CLK | 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 x15 protokollen. Se protokol afsnit for yderligere information. |
| LED\_CONTROL | 0-5V digitale signaler til styring af LED indikatorer | 2 0-5V digitale signaler der styre power indikator LED samt x15 kommunikationsindikator LED der begge er active high.  3V til 5V = 1  0V til 1,5V= 0 |
| LCD\_CONTROL | 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. |
| I2C | Kommunikation der følger I2C standarden | Kommunikation via I2C standarden til Real Time Clock. Se protokol afsnit for yderligere information. |
| SPI | Kommunikation der følger SPI protokollen | Kommunikation via SPI til styring af SD-kort. Se protokol afsnit for yderligere information. |

### 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 x15 eller Uart.

**P1: 18V\_AC** er et 18V ac 50 Hz signal som samtidig indeholder et kommunikationssignal der følger x15 protokollen. Signalet fungere som spændingsforsyning til systemets blokke samt som kommunikationssignal mellem styreboksen og enhederne via x15 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 x15 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 x15 og microcontrolleren hvor 3V til 5V = 1 og 0V til 1,5V = 0.

**D2: Digital** er et digitalt 0-5V signal der er activ-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 X15 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 forsyningsspændingerne på 5V og 10V.

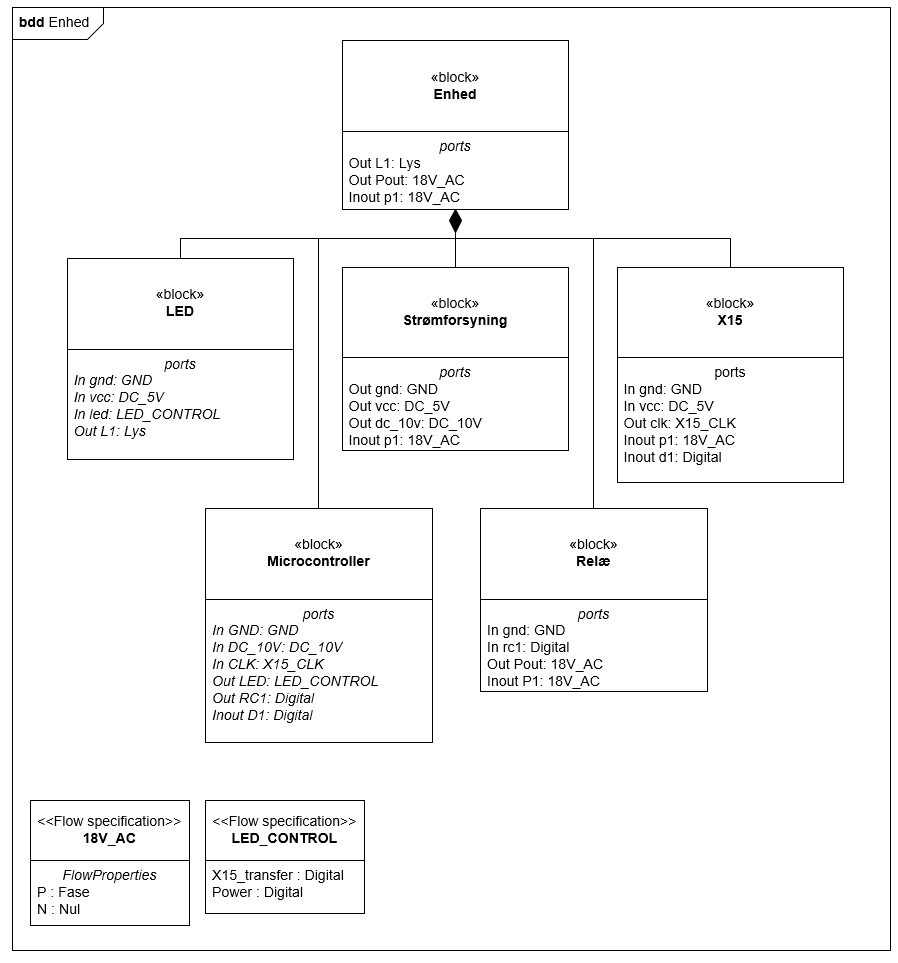
**USB1:Uart** er et Uart-kommunikationssignal til kommunikation med pc-softwaren.

## Enheder

Hver enhed har en **strømforsyning**, der laver en 18 VAC om til 5 VDC og 10 VDC. Enhedenbestår af en **microcontroller**, denne styrer al kommunikation med et modul kaldet **X15**, der indeholder en zero-cross detektor. Microcontrolleren kontrollerer også et **relæ**, der bestemmer om der skal 18 VAC til den tilsluttede lampe eller ej. Enheden har også en **LED**, der viser om der sendes data til styreboksen.

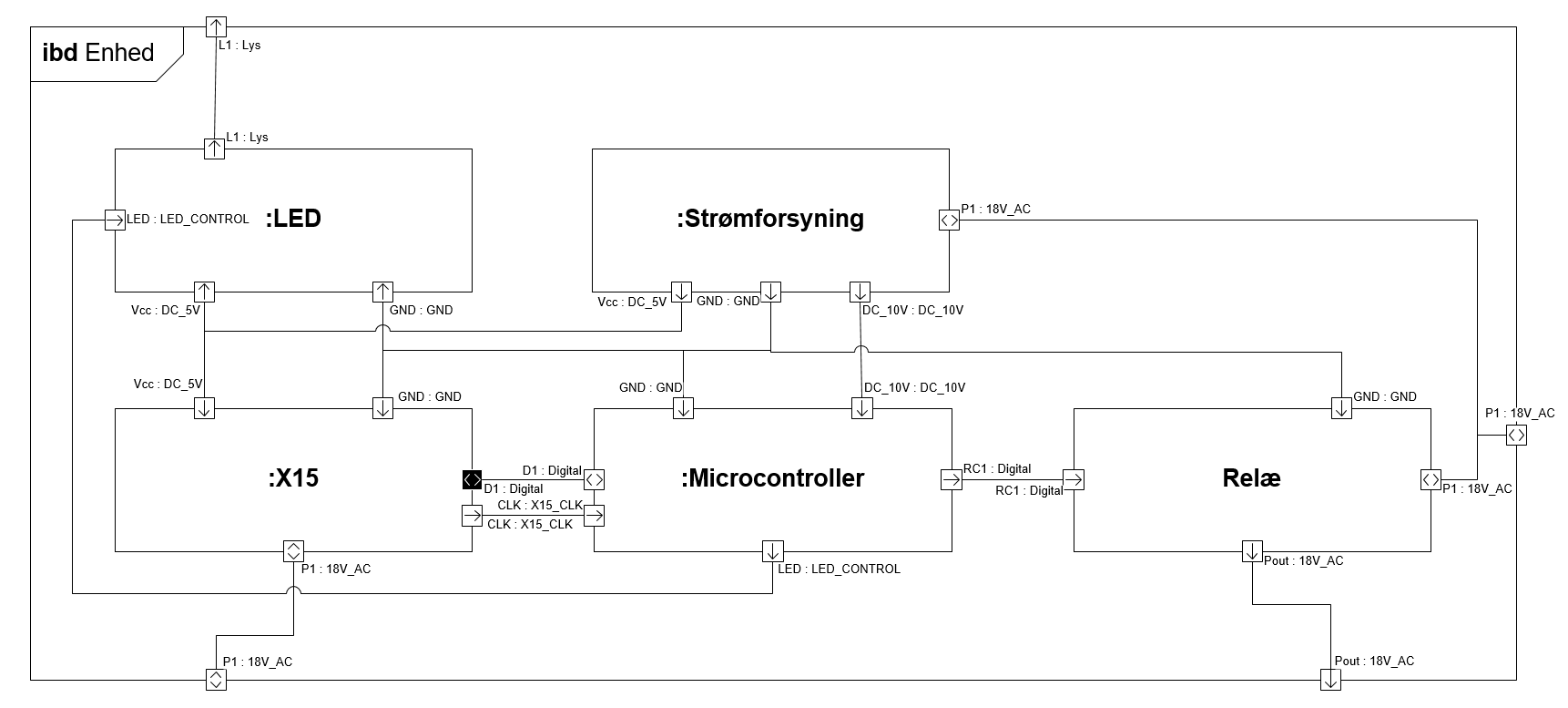
### BDD og IBD.

Ud fra den overordnede funktionalitet der er beskrevet i kravspecifikationen er der for enheden udarbejdet en BDD som ses på figur 6. På denne BDD kan det udledes hvilke blokke en Enhed består af samt hvilke grænseflader den har til andre delsystemer.



Figur - BDD enhed

Ud fra BDD for Enheden er der udarbejdet en IBD som ses på figur 7. Ud fra denne kan de interne forbindelser samt signaler udledes og derved danne grundlaget for den videre hardware udvikling sammen med den BDD der ses på figur 6.



Figur - IBD enhed

### Grænseflade

|  |  |  |
| --- | --- | --- |
| Grænseflade | Porte | Stiktype |
| P1 – strømforsyning | Input 18 V AC 50 Hz  Output 18 V AC 50 Hz | 230 V stærkstrømstik han efter dansk lovkrav |
| LED – LED | Input 0 – 5 V DC |  |
| P1 – X15 | Input 18 V­ AC 50 Hz  Input X15  Output 18 V AC 50 Hz  Output X15 |  |
| P1 – Relæ | Input 18 V AC 50 Hz  Output 18 V AC 50 Hz |  |
| 18V\_AC – Relæ | Output 18 V AC 50 Hz |  |

### Signaler til enhed IBD

|  |  |  |
| --- | --- | --- |
| Signaltype | Definition | Beskrivelse |
| 18V\_AC | 18V AC 50 Hz signal kombineret med et 100kHz x15 signal | 18V AC 50 Hz signal fra forsyningsnettet der også indeholder kommunikationen via x15 protokollen der udvikles specifikt til dette produkt, se protokol afsnit for yderligere information. |
| Lys | Lys i det synlige spektrum | Lys i 2 farver afhængig af hvilket LED indikator der lyser. |
| DC\_10V | +10V DC signal | Forsyningsspænding til vores arduino mega2560 microcontroller der har egen dc converter monteret.  Signalet er et DC 10V signal +/- 0.5V |
| DC\_5V | +5V DC signal | Forsyningsspænding til moduler der har behov for 5V dc spænding.  Signalet er et DC signal på 5V +/- 0.5V |
| GND | 0V DC signal | Reference spænding til DC forsyningsspændinger samt til digitale signaler. |
| Digital | 0-5V DC signal | Digitalt signal  3V til 5V = 1  0V til 1.5V = 0 |
| X15\_CLK | 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 x15 protokollen. Se protokol afsnit for yderligere information. |
| LED\_CONTROL | 0-5V digitale signaler til styring af LED indikatorer | 2 0-5V digitale signaler der styre power indikator LED samt x15 kommunikationsindikator LED der begge er active high. |

### 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 x15.

**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 x15 protokollen. Signalet fungere som spændingsforsyning til systemets blokke samt som kommunikationssignal mellem styreboksen og enhederne via x15 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 x15 protokollen.

**D1: Digital** er et digital 0-5V serielt signal til datatransmission mellem x15 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 X15 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 forsyningsspændingerne 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

# Protokoller

## X15

Til at kommunikere mellem Styreboks og enhed benyttes X15 kommunikation.

X15 kommunikation sendes 1 bit ad gangen hvorfor en kommando kan opdeles i flere pakker. Fast for hver kommando er 4 startbits, en huskode, 8 adressebits, 3 typebits, nul eller flere databits, en paritetsbit og 6 slutbits.

Startbits (STX): Fortæller at en besked er på vej  
Huskode: Indeholder koden for det hus hvor enheden befinder sig  
Adresse: Indeholder adressen på enheden  
Typebits: Definerer antal databit i beskeden og kommandoen/svaret i datapakken. Se afsnittet *typeliste* for uddybning.   
Databits: Indeholder bools eller fejlkoder  
Paritetsbits: Validerer om besked er modtaget intakt  
Slutbits (ETX): Fortæller at en besked er slut

I X15 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 fejlmeddelelse.

### Tabeloversigt over X15 kommunikation.

Nedenstående tabeller beskriver de forskellige datapakker der sendes mellem Styreboks og Enhed.  
Tabellerne er opdelt efter følgende eksempel:

|  |  |
| --- | --- |
| 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 |

### Kommandoer

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

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

### Svar

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

## Generelle svar

### Godkendt

|  |  |
| --- | --- |
| validated | |
| Besked modtaget og forstået | |
| Styreboks > Enhed | |
| Huskode: | 'XXXX' |
| Adresse: | 'XXXX-XXXX' |
| Type: | '100' |
| Data: | - |

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

### Fejlmeddelelser

|  |  |
| --- | --- |
| error | |
| Besked modtaget men ikke forstået | |
| Styreboks > Enhed | |
| Huskode: | 'XXXX' |
| Adresse: | 'XXXX-XXXX' |
| Type: | '110' |
| Data: | b0-b1: Fejlkode |

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

## PiS (PC interface to Styreboks)

Til at kommunikere mellem PC og Styreboks bruges PiS kommunikation som foregår over UART forbindelse.

PiS kommunikation sendes 1 byte ad gangen hvorfor en kommando kan opdeles i flere pakker. Fast for hver kommando er en startbyte, en typebyte, en længdebyte, nul eller flere databytes, en paritetsbyte og en slutbyte.

Startbyte (STX): Fortæller at en besked er på vej  
Typebyte: Definerer kommandoen/svaret i datapakken. Se afsnittet *typeliste* for uddybning.  
Længdebyte: Definerer hvor mange databytes der sendes  
Databytes: Indeholder eksempelvis adresse, rum, navn på enhed/rum, tid mm.  
Paritetsbyte: Validerer om besked er modtaget intakt  
Slutbyte (ETX): Fortæller at en besked er slut

I PiS protokollen er alle lige typenumre hvor PC tilgår Styreboks, oftest kommandoer. Alle ulige typenumre er hvor Styreboks tilgår PC, alle svar.

I oversigten over protokollerne er startbyte, paritetsbyte og slutbyte ikke synlige eftersom de altid er en del af beskeden, og ikke defineret af brugeren. Eks:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| STX | Type | Længde | Data | Paritet | ETX |

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

### Tabeloversigt over PiS kommunikation.

Nedenstående tabeller beskriver de forskellige datapakker der sendes mellem PC og styreboks.

Tabellerne er opdelt efter følgende eksempel:

|  |  |
| --- | --- |
| Use case der tages udgangspunkt i | |
| Metode/funktion | |
| Beskrivelse af type og metode/funktion | |
| Retning på dataflow: Fra > Til | |
| Type: | 'Hexadecimal tal' |
| Længde: | 'Hexadecimal tal' |
| Data: | Bytenummer: data |

## Kommandoer

|  |  |
| --- | --- |
| UC1: Opstart af system | |
| validatePin() | |
| Validering af kodelås | |
| PC > Styreboks | |
| Type: | '00' |
| Længde: | '00' |
| Data: | - |

## Svar

|  |  |
| --- | --- |
| UC1: Opstart af system | |
| pinValidated | |
| Svar på validering | |
| Styreboks > PC | |
| Type: | '01' |
| Længde: | '01' |
| Data: | B0: bool |

|  |  |
| --- | --- |
| UC1: Opstart af system | |
| getUnit() | |
| Henter tidstabel og ID | |
| PC > Styreboks | |
| Type: | '02' |
| Længde: | '00' |
| Data: | - |

|  |  |
| --- | --- |
| UC1: Opstart af system | |
| unit(ID, timeTable) | |
| Sender enheds ID og tidsplan | |
| Styreboks > PC | |
| Type: | '03' |
| Længde: | 'XX' |
| Data: | B0: bool |
|  | B1 – BX: Timetable |

|  |  |
| --- | --- |
| UC2 - Statusforespørgsel | |
| getUnitStatus(unitID) | |
| Anmoder om status på enhed | |
| PC > Styreboks | |
| Type: | '04' |
| Længde: | '01' |
| Data: | B0: Adresse |

|  |  |
| --- | --- |
| UC2 - Statusforespørgsel | |
| status | |
| Sender status på enhed | |
| Styreboks > PC | |
| Type: | '05' |
| Længde: | '01' |
| Data: | B0: Status |

|  |  |
| --- | --- |
| UC3 - Tilføjelse af enhed | |
| sendUnit(unitID) | |
| Giver styreboks besked om at ny enhed er tilføjet | |
| PC > Styreboks | |
| Type: | '06' |
| Længde: | '01' |
| Data: | B0: Adresse |
| UC3 - Tilføjelse af enhed | |
| Unit received | |
| Godkender at ny enhed er tilføjet til systemet | |
| Styreboks > PC | |
| Type: | '07' |
| Længde: | '01' |
| Data: | B0: bool |

|  |  |
| --- | --- |
| UC4 - Fjern enhed | |
| deleteUnit(unitID) | |
| Fjerner enhed fra system | |
| PC > Styreboks | |
| Type: | '08' |
| Længde: | '01' |
| Data: | B0: Adresse |

|  |  |
| --- | --- |
| UC4 - Fjern enhed | |
| Unit deleted | |
| Enhed fjernet | |
| Styreboks > PC | |
| Type: | '09' |
| Længde: | '01' |
| Data: | B0: bool |

|  |  |
| --- | --- |
| UC5 - Ret enhed | |
| modifyUnit(unit, prevUnitID) | |
| Retter enhed | |
| PC > Styreboks | |
| Type: | '0A' |
| Længde: | '02' |
| Data: | B0: Adresse |
|  | B1: Rum |

|  |  |
| --- | --- |
| UC6 - Ændring af tidsplan | |
| updateTime(unitID, schedule) | |
| Ændrer tidsplan for valgte enhed(er)/rum | |
| PC > Styreboks | |
| Type: | '0C' |
| Længde: | 'XX' |
| Data: | B0: Adresse |
|  | B1: Starttid |
|  | B2: Sluttid |
|  | B3: Dage |

|  |  |
| --- | --- |
| UC5 - Ret enhed | |
| Unit modified | |
| Enhed rettet | |
| Styreboks > PC | |
| Type: | '0B' |
| Længde: | '01' |
| Data: | B0: bool |

|  |  |
| --- | --- |
| UC6 - Ændring af tidsplan | |
| Time updated | |
| Tidsplanen er opdateret | |
| Styreboks > PC | |
| Type: | '0D' |
| Længde: | '01' |
| Data: | B0: bool |

## Generelle svar

### Godkendt

|  |  |
| --- | --- |
| validated() | |
| Godkender at besked er modtaget og forstået | |
| PC > Styreboks | |
| Type: | '0E' |
| Længde: | '00' |
| Data: | - |

|  |  |
| --- | --- |
| validated() | |
| Godkender at besked er modtaget og forstået | |
| Styreboks > PC | |
| Type: | '0F' |
| Længde: | '00' |
| Data: | - |

### Fejlmeddelelser

|  |  |
| --- | --- |
| error() | |
| Giver besked om fejl i data | |
| PC > Styreboks | |
| Type: | '10' |
| Længde: | '01' |
| Data: | B0: Fejlkode |

|  |  |
| --- | --- |
| error() | |
| Giver besked om fejl i data | |
| Styreboks > PC | |
| Type: | '11' |
| Længde: | '01' |
| Data: | B0: Fejlkode |

## I2C/IIC/TWI

I2C protokollen benyttes til at kommunikere med vores Real Time Clock modul som er baseret på en DS3231 Real Time Clock, standarden for I2C er veldefineret ved 100kHz og 400 kHz, men i forbindelse med DS3231 anvendes der en 400kHz clock-frekvens. Ønskes der yderlige information om I2C standarden kan dette findes i I2C specifikationen der er vedlagt som bilag. For mere specifik information om brugen af I2C til kommunikation med DS3231 kan dette findes i DS3231 datasheet der ligeledes er vedlagt som bilag.

## SPI

SPI protokollen benyttes til kommunikation med vores SD-kort, der bliver i vores projekt anvendt direkte adgang til de enkelte datablokke på sd-kortet i stedet for at implementere et fat filsystem. For yderligere information om SPI i forbindelse med sd-kort se SD-Card simplified spec kapitel 7 som er vedlagt som bilag. For yderligere infomation om SPI generelt kan dette findes i the avr microcontroller and embeded system kapitel 17.

# Typeliste

Type er defineret ved at det er typen af besked der sendes. Ved dette forstås både hvilken retning beskeden går, f.eks fra PC til Styreboks, og hvilken handling der udføres. Se nedenstående tabeller for sammenhæng mellem type, retning på dataflow og handling:

## PiS

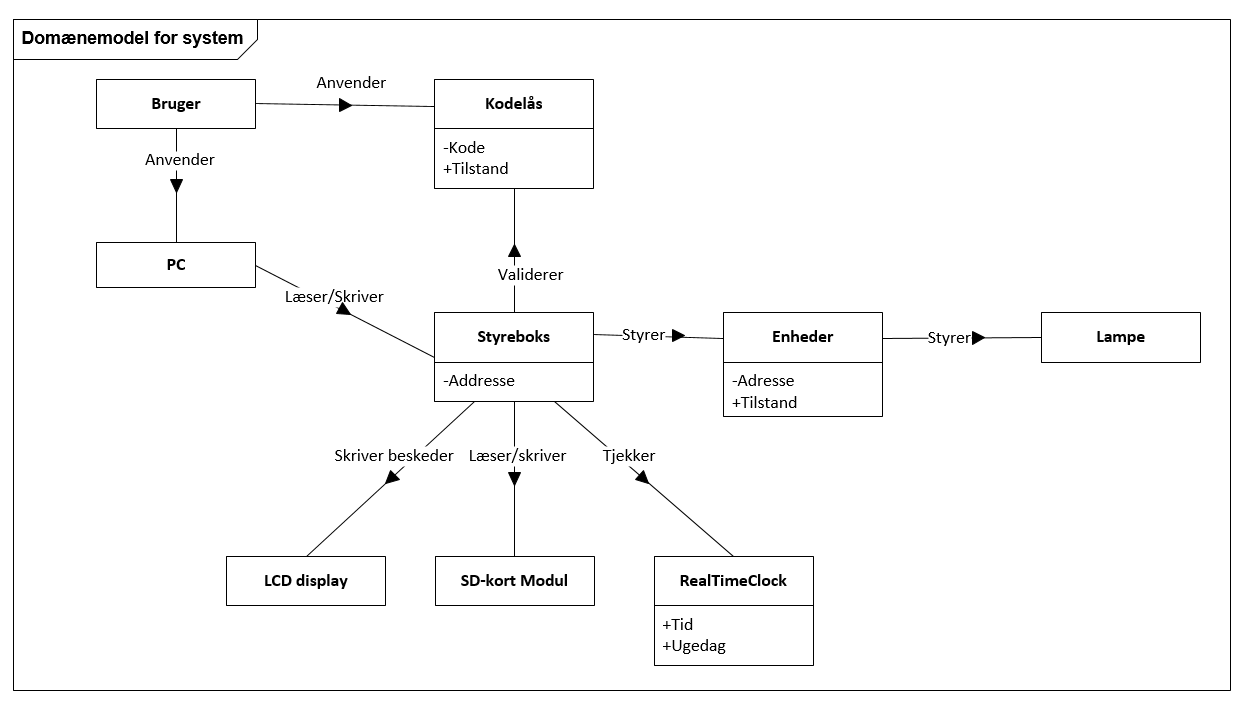
|  |  |  |  |
| --- | --- | --- | --- |
| **Type** | **Navn** | **Fra > Til** | **Handling** |
| 00 | validatePin() | PC > Styreboks | Validering af kodelås |
| 01 | pinValidated | Styreboks > PC | Svar på validering |
| 02 | getUnitStatus(unitID) | PC > Styreboks | Henter tidstabel og ID |
| 03 | unit(ID, timeTable) | Styreboks > PC | Sender enheds ID og tidsplan |
| 04 | getUnitStatus(unitID) | PC > Styreboks | Anmoder om status på enhed |
| 05 | status | Styreboks > PC | Sender status på enhed |
| 06 | sendUnit(unitID) | PC > Styreboks | Giver styreboks besked om at ny enhed er tilføjet |
| 07 | Unit recieved | Styreboks > PC | Godkender at ny enhed er tilføjet til systemet |
| 08 | deleteUnit(unitID) | PC > Styreboks | Fjerner enhed fra system |
| 09 | Unit deleted | Styreboks > PC | Enhed fjernet |
| 0A | modifyUnit(unit, prevUnitID) | PC > Styreboks | Retter enhed |
| 0B | Unit modified | Styreboks > PC | Enhed rettet |
| 0C | updateTime(unitID, schedule) | PC > Styreboks | Ændrer tidsplan for valgte enhed(er)/rum |
| 0D | Time updated | Styreboks > PC | Tidsplanen er opdateret |
| 0E | validated() | PC > Styreboks | Godkender at besked er modtaget og forstået |
| 0F | validated() | Styreboks > PC | Godkender at besked er modtaget og forstået |
| 10 | error() | PC > Styreboks | Giver besked om fejl i data |
| 11 | error() | Styreboks > PC | Giver besked om fejl i data |

## X15

|  |  |  |  |
| --- | --- | --- | --- |
| **Type** | **Navn** | **Fra > Til** | **Handling** |
| 000 | getUnitStatus(unitID) | Styreboks > Enhed | Anmoder om status på enhed |
| 001 | status | Enhed > Styreboks | Sender status på enhed |
| 010 | switchState(unitID) | Styreboks > Enhed | Sender ny status |
| 011 | Return true | Enhed > Styreboks | Sætter enhed i ny tilstand |
| 100 | validated | Styreboks > Enhed | Besked modtaget og forstået |
| 101 | validated | Enhed > Styreboks | Besked modtaget og forstået |
| 110 | error | Styreboks > Enhed | Besked modtaget, men ikke forstået |
| 111 | error | Enhed > Styreboks | Besked modtaget, men ikke forstået |

# Software-arkitektur

I det følgende afsnit defineres software-arkitekturen for systemet, ud fra den foreliggende kravspecifikation.  
Der laves sekvens diagrammer for systemets CPU´er, i forhold til de use-cases som CPU´en indgår i, og disse kommer så til at danne grundlag for et klasse diagram for den pågældende CPU.   
Vi tager udgangspunkt i en domæneanalyse-model, og de blokke som indgår i denne.



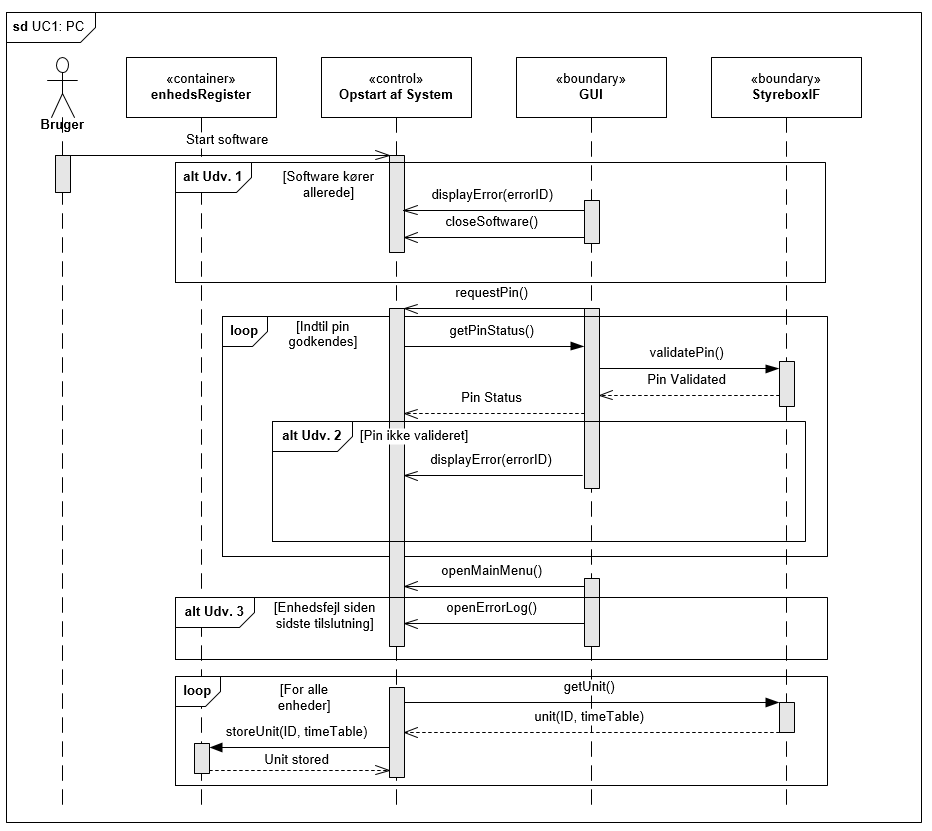
Figur - Domæne model for hele systemet.

## PC.

PC´en bruges i systemet som brugeren interface til interaktion med systemet.

### Use-case 1:

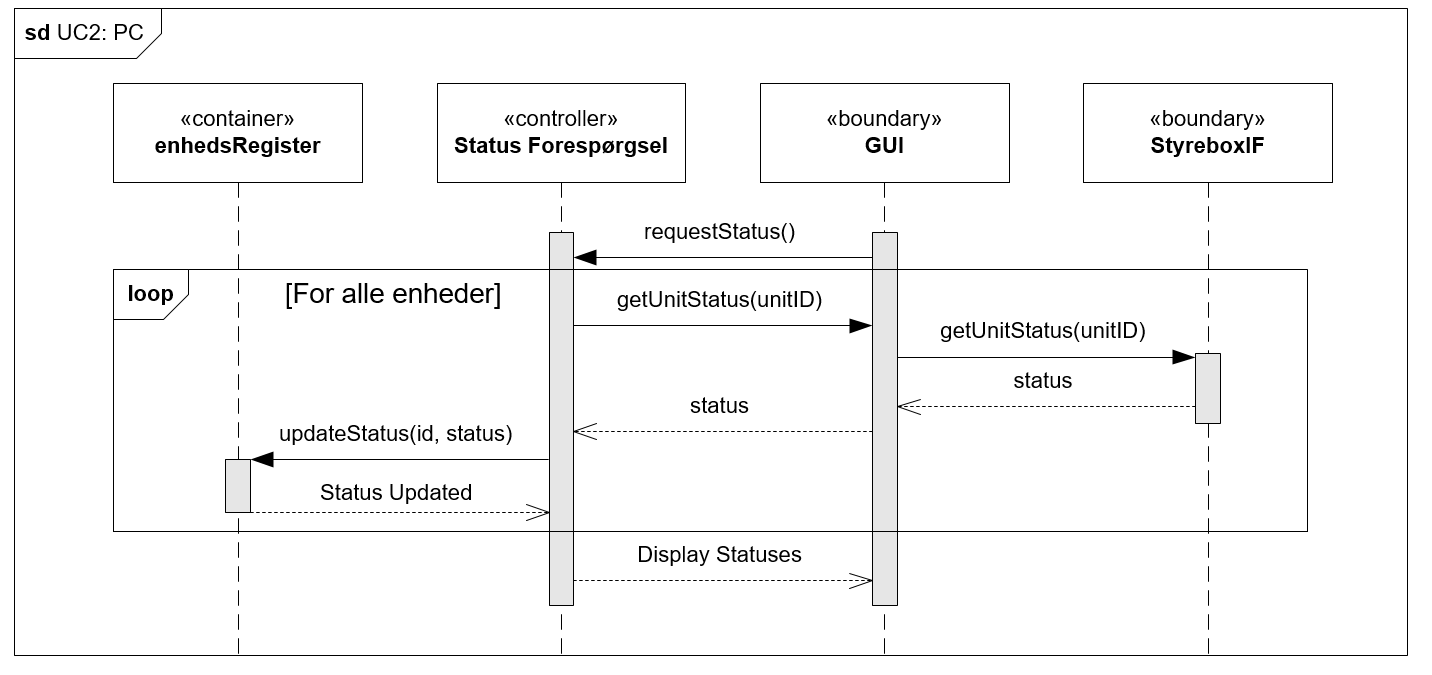
Use-case 1 omhandler opstart at PC-software. Her skal PC´en kommunikerer med styreboksen og have valideret at koden til systemet er korrekt indtastet, inden PC softwaren skal starte op.



Figur 9 - Sekvensdiagram UC1 for PC

### Use-case 2:

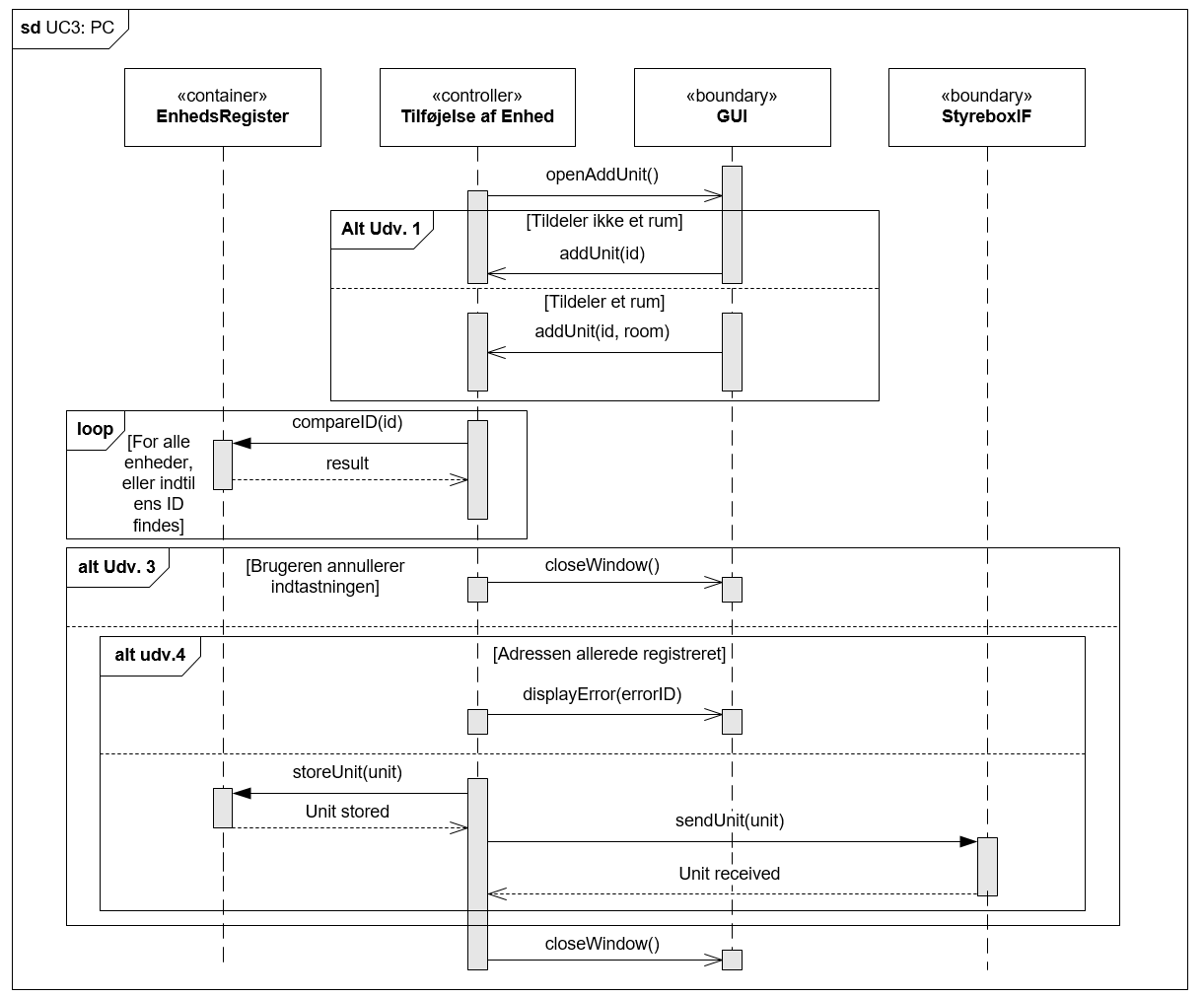
I use-case 2 laves der status forespørgsel for alle enheder, som foretages af styreboksen der sender enhedsstatus tilbage til PC´en i et loop.



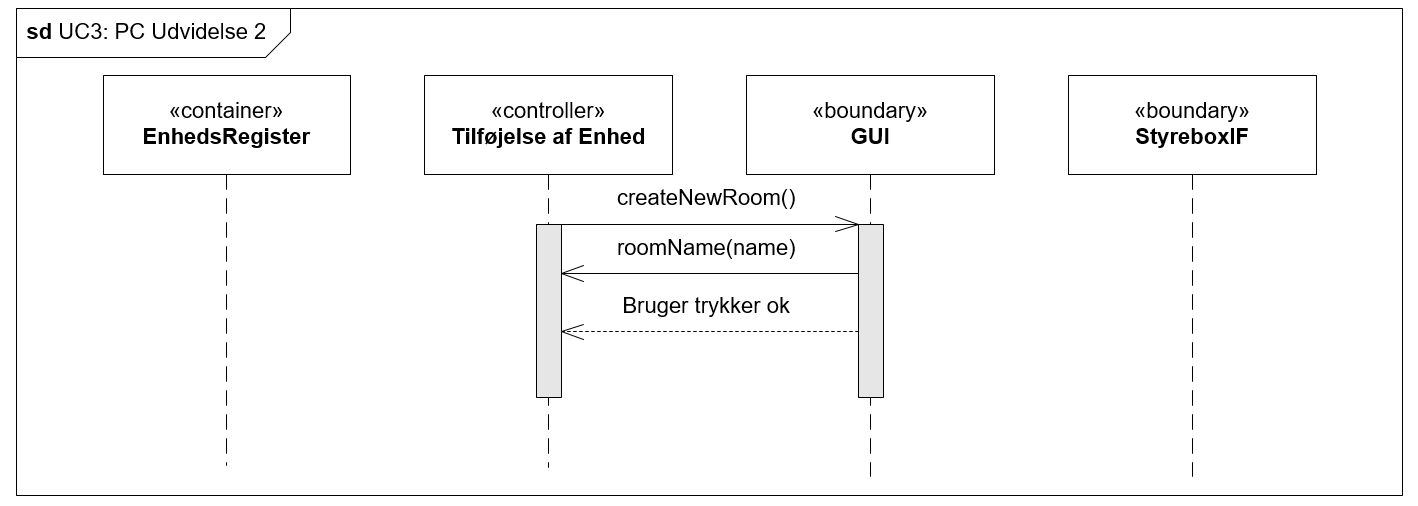
Figur 10 - Sekvensdiagram UC2 for PC

### Use-case 3:

Use-case 3 omhander at tilføje en ny enhed til systemet. Her håndterer PC softwaren alt tjek up på kendte enheder og rum, inden funktionskaldet sendes til styreboksen.



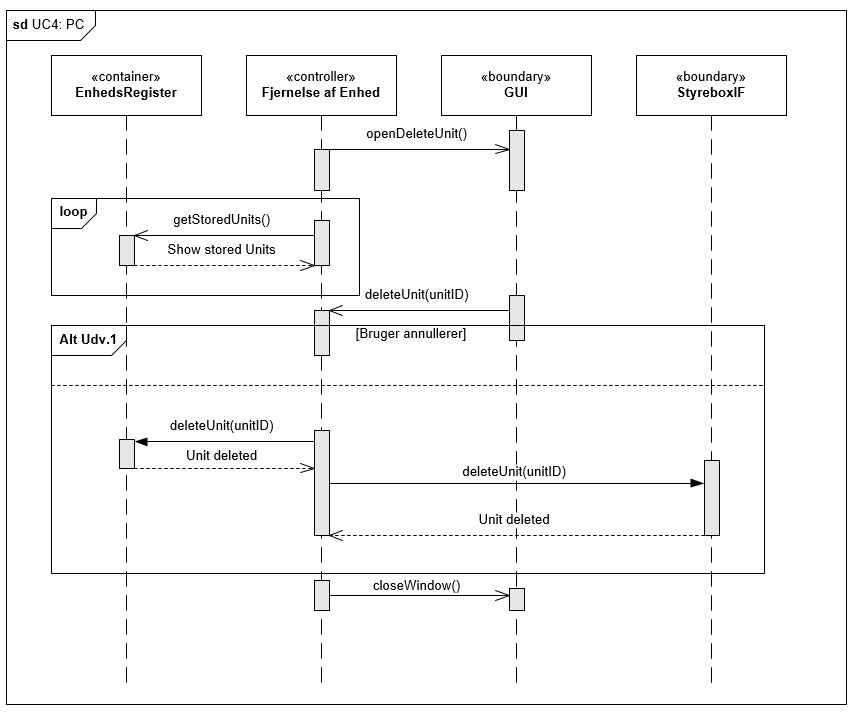
Figur - Sekvensdiagram UC3 for PC



Figur Sekvensdiagram UC3, udvidelse 2 for PC

### Use-case 4:

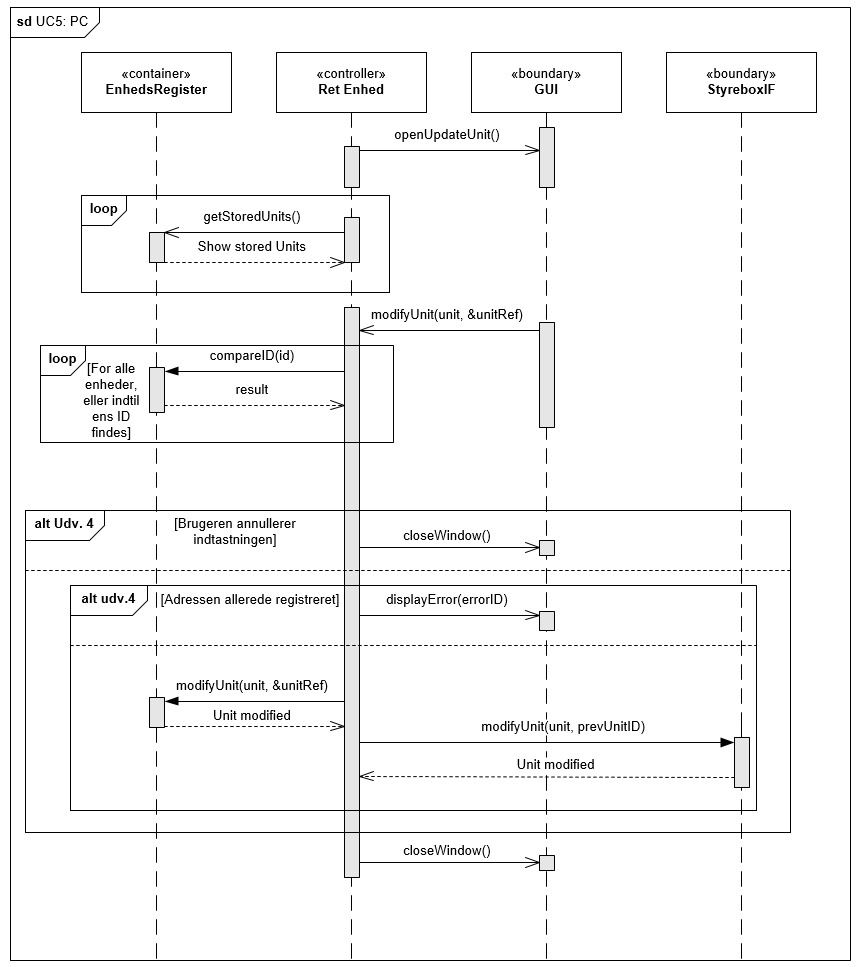
Use-case 4 omhandler at fjerne en gemt enhed fra systemets konfiguration. Dette håndteres lokal i PC softwaren inden funktionskaldet sendes til styreboksen.



Figur 13 - Sekvensdiagram UC4 for PC

### Use-case 5:

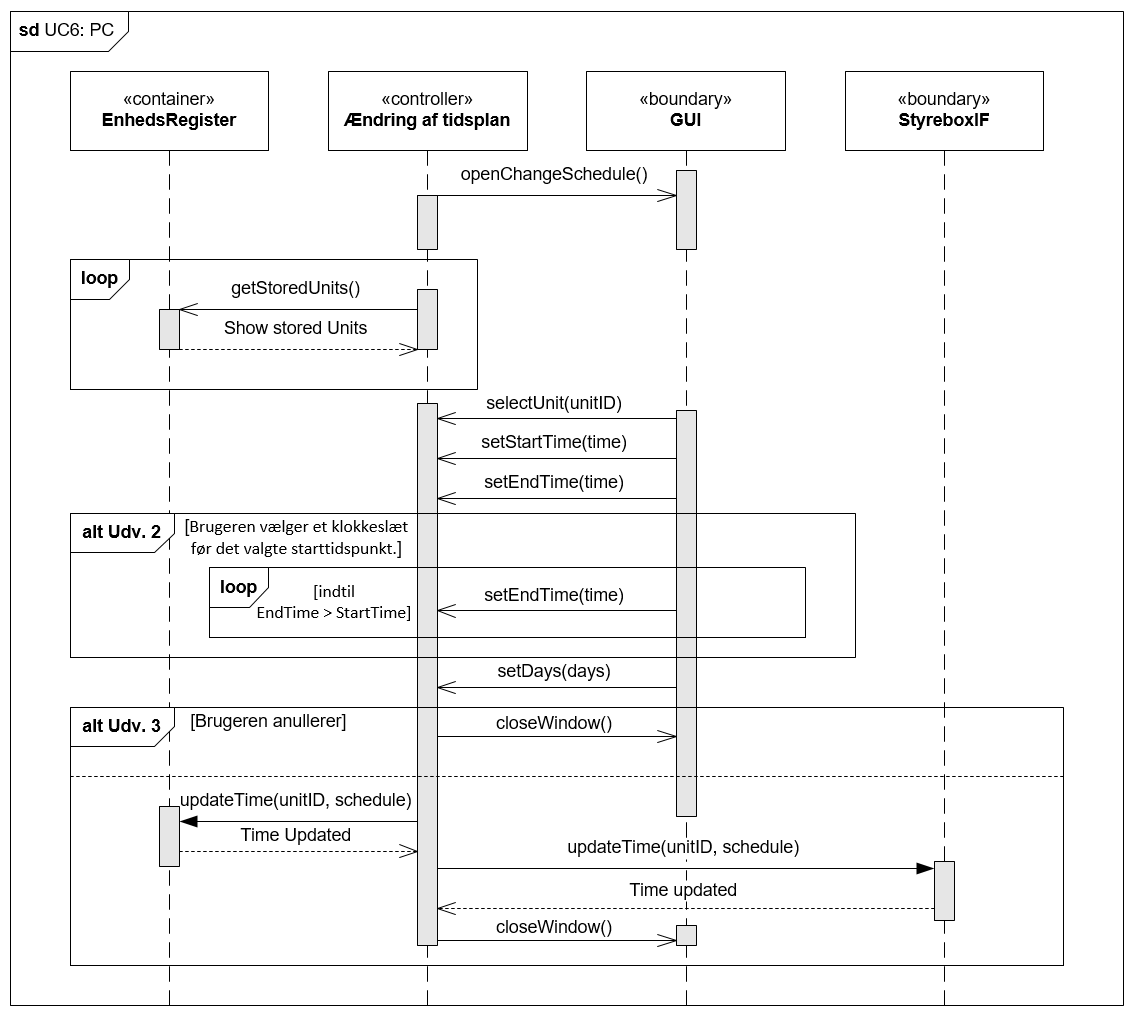
Use-case 5 omhandler at redigerer en gemt enhed i systemets konfiguration. Dette håndteres lokal i PC softwaren inden funktionskaldet sendes til styreboksen.



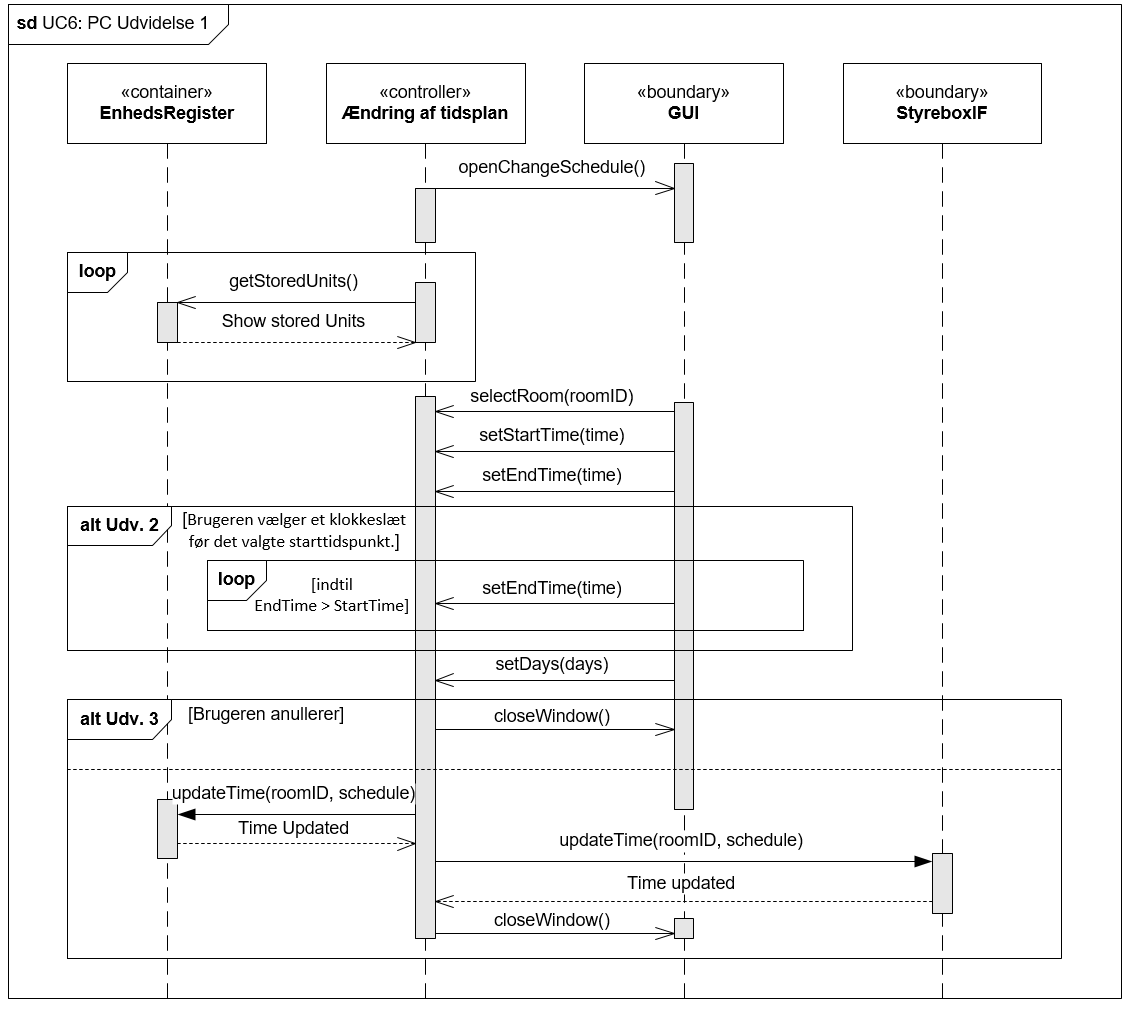
Figur 14 - Sekvensdiagram UC5 for PC

### Use-case 6:

Use-case 6 omhandler redigering af tidsplanen for en gemt enhed. Her anvender brugeren af systemet PC´en til at indtaste nye data, som først håndteres lokalt, for derefter at blive sendt til styreboksen som gemmer data i dens tidsplan.



Figur - Sekvensdiagram UC6 for PC



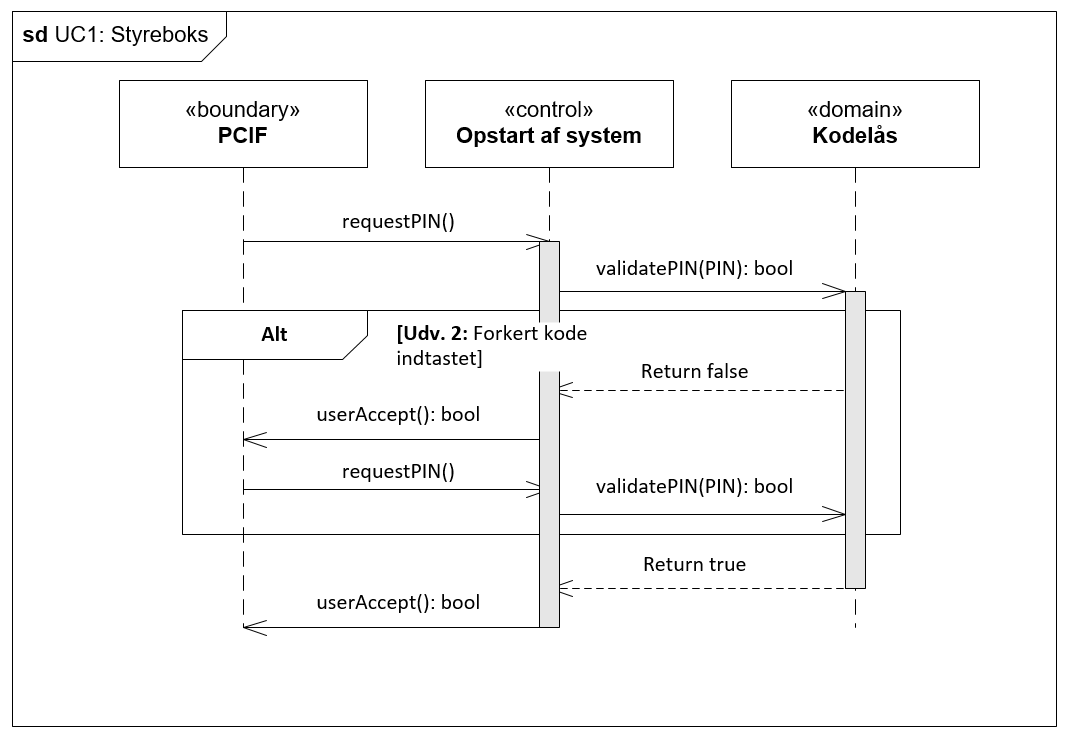
Figur - Sekvensdiagram UC6, udvidelse1 for PC

## Styreboks.

Vores styreboks indgår i alle use-cases, da denne er systemets ”hjerne” og agerer som bindeled mellem systemets andre del-systemer, dermed styre styreboksen hele systemets funktionalitet.

### Use-case 1:

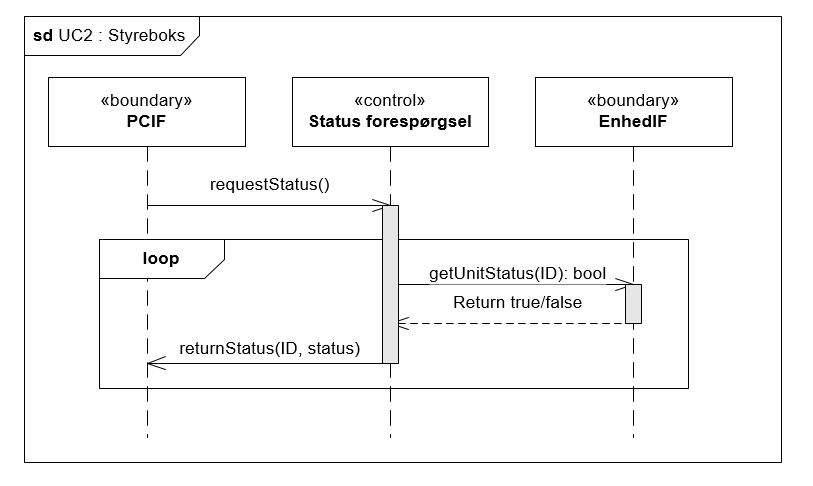
Use-case 1 omhandler opstart at PC-software, hvor styreboksen skal validere om den indtastede kode på kodelåsen er korrekt.



Figur 17 - Sekvensdiagram UC1 for styreboks

### Use-case 2:

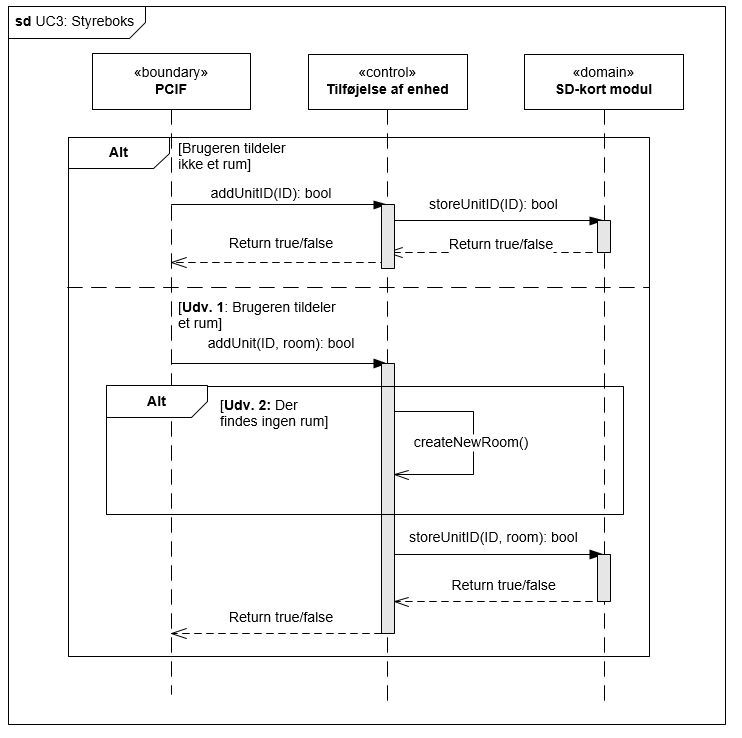
Use-case 2 omhandler indhentning af status på enhederne, her indgår styreboksen idet den kommunikerer med enhederne, og derved efterspørger deres status.



Figur 18 - Sekvensdiagram UC2 for styreboks

### Use-case 3:

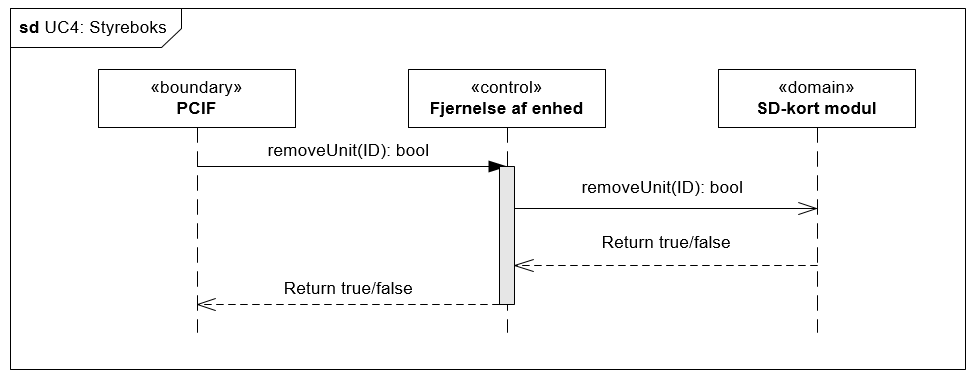
Use-case 3 omhandler at gemme en enhed i konfigurationen, her indgår styreboksen som bindeled mellem PC-interface, og SD-kortet hvor konfigurationen ligger.



Figur 19 - Sekvensdiagram UC3 for styreboks

### Use-case 4:

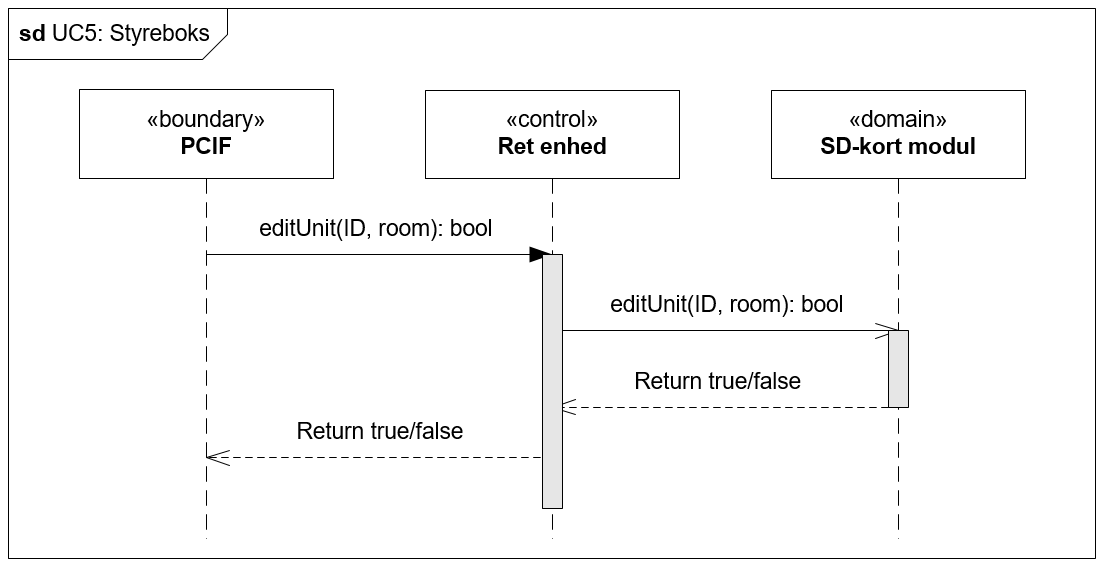
Use-case 4 omhandler at fjerne en gemt enhed fra konfigurationen, her indgår styreboksen som bindeled mellem PC-interface, og SD-kortet hvor konfigurationen ligger.



Figur - Sekvensdiagram UC4 for styreboks.

### Use-case 5:

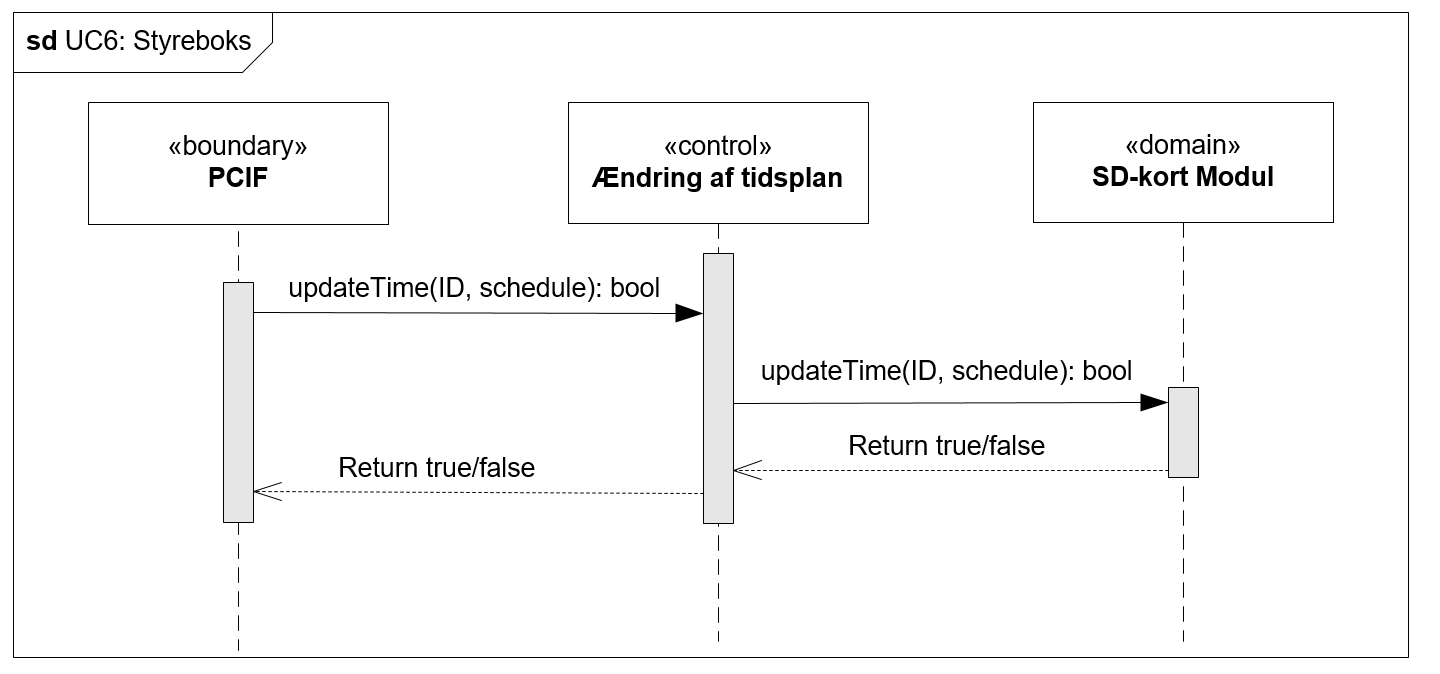
Use-case 5 omhandler at redigerer en enhed i konfigurationen, her indgår styreboksen som bindeled mellem PC-interface, og SD-kortet hvor konfigurationen ligger.



Figur 21 - Sekvensdiagram UC5 for styreboks

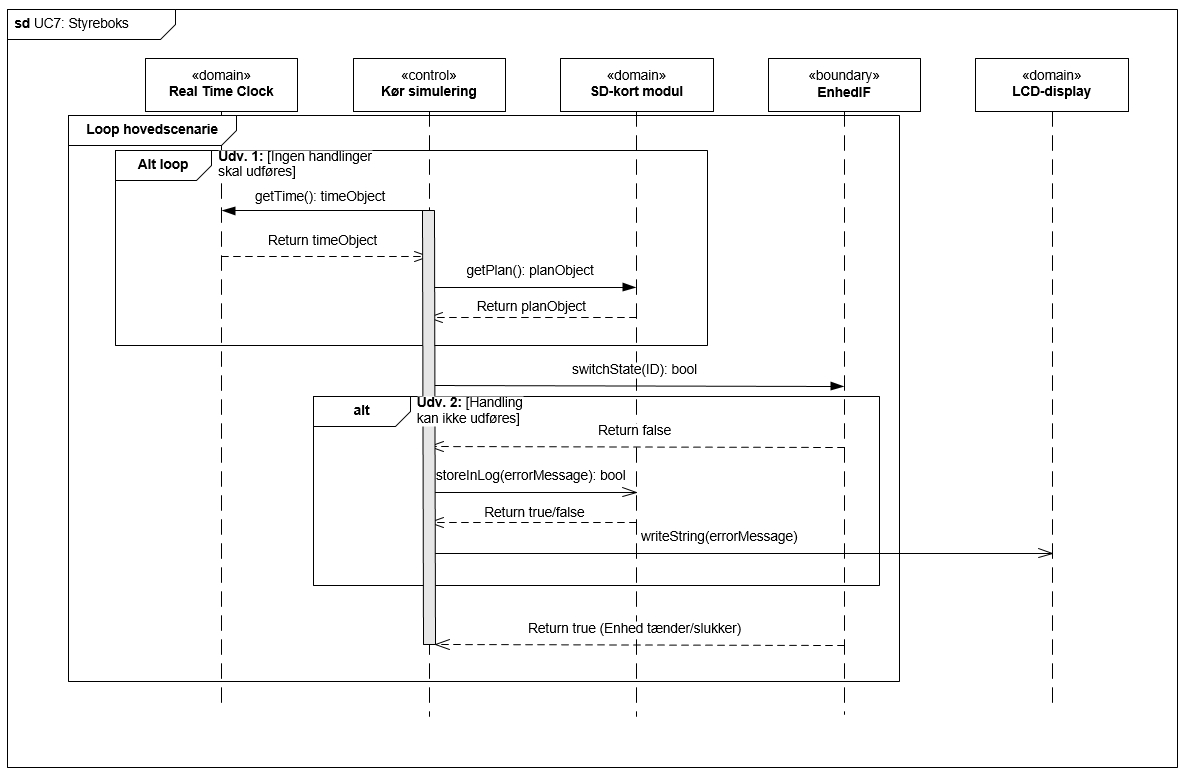
### Use-case 6:

Use-case 6 omhandler at redigerer tidsplanen for en enhed, her indgår styreboksen som bindeled mellem PC-interface, og SD-kortet hvor tidsplanerne ligger.



Figur 22 - sekvensdiagram UC6 for styreboks

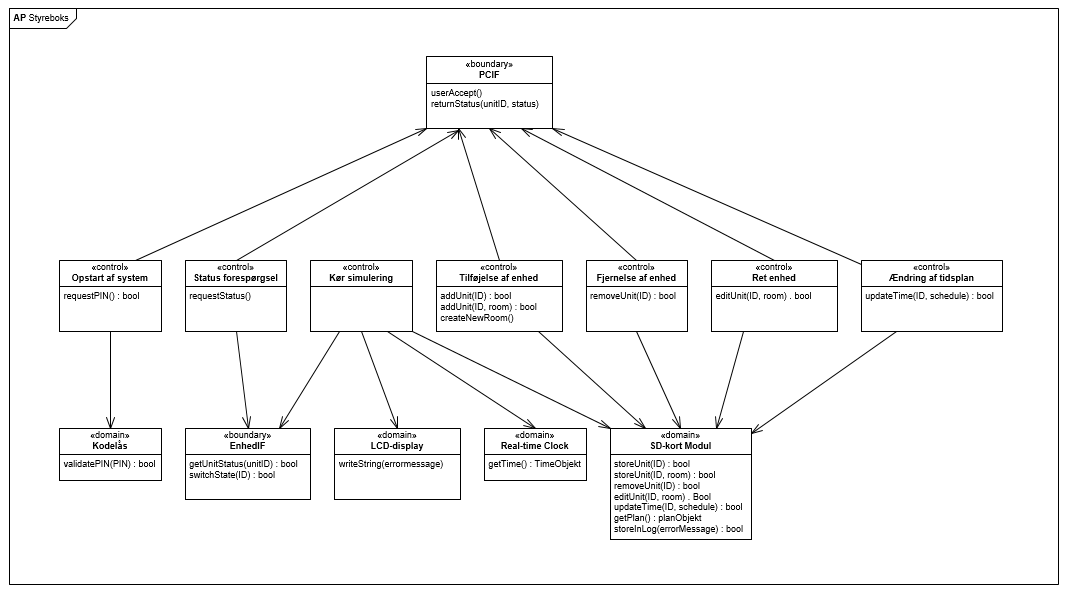
### Use-case 7:

I use-case 7 køres systemets simulering, her er det styreboksen som tænder og slukker for de tilsluttede enheder, ud fra den konfiguration som ligger i SD kortet. 

Figur - Sekvensdiagram UC7 for styreboks

### Applikationsmodel for styreboksen

Applikationsmodellen bruges til senere udvikling af UML klasse diagrammer for styreboksens CPU.



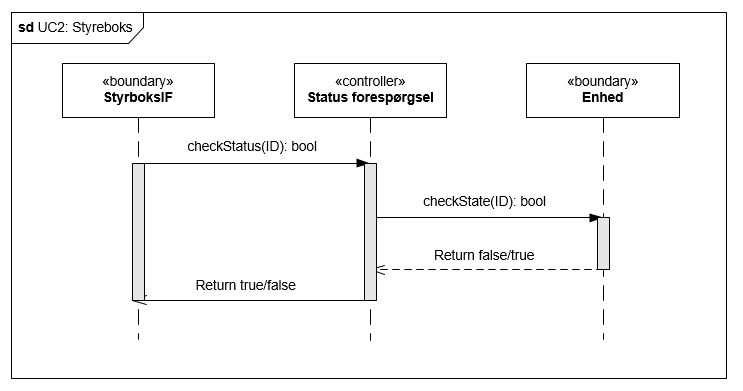
Figur - Applikationsmodel for styreboks

## Enhed.

Systemets enheder indgår i de 2 use-cases hvor der sker kommunikation mellem styreboksen og enhederne.

### Use-case 2:

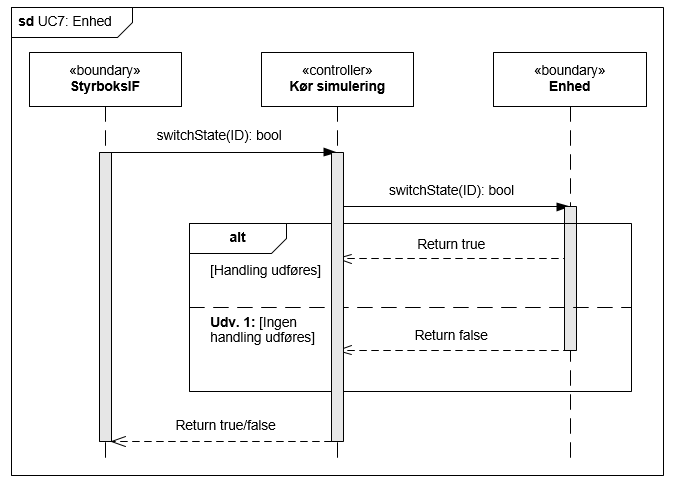
Enhederne indgår i use-case 2, da enheden modtager en anmodning fra styrebokse om at fortælle status på outputtet ”Pout”.



Figur - Sekvensdiagram UC2 for enhed

### Use-case 7:

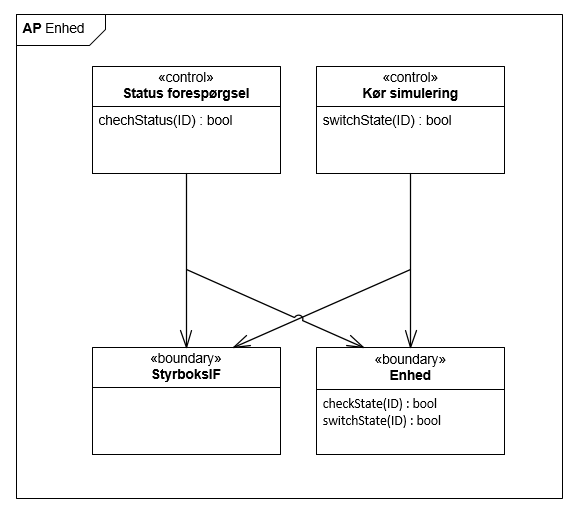
I use-case 7 modtager enhederne en anmodning om at ændre tilstanden på outputtet ”Pout”.



Figur - Sekvensdiagram UC7 for enhed

### Applikationsmodel for enhed.

Applikationsmodellen er udarbejdet ud fra enhedernes sekvensdiagrammer, og de funktionskald som i udviklingen af disse er optået. Denne bruges senere i udviklingen som grundlag for UML klasse diagram for enhedernes CPU.



Figur - Applikationsmoel for enheder