

# **Indholdsfortegnelse**

---



# Ordliste 1

---

- AC** Alternating Current (Vekselstrøm)
- API** Application Programming Interface (Softwaregrænseflade til SMS kommunikation)
- Carriage return** ASCII kommando for at returnerer cursoren. (Kommunikations protokollerne som ETX)
- CSS** Child Security System (Børnesikkerheds System)
- ETX** End of text (Seriell kommunikation)
- HMI** Human Machine Interface (Brugergrænseflade man kan interagere med)
- ISR** Interrupt Service Routine (Microcontroller functions kaldt i tilfælde af et interrupt signal)
- RS232** Recommended Standard 232 (Seriell digital datakommunikation)
- STK500** Atmel Mega32 development board
- STX** Start of text (Seriell kommunikation)
- UART** Universal Asynchronous Receiver/Transmitter
- UC** Use Case
- UI** User Interface (Brugergrænseflade)
- VAC** Volt Alternating Current (Vekselstrøm)
- X10** Protocol for communication among electronic



# Kravspecifikation 2

---

Versionshistorik	
<b>v1.1</b>	19-05-2014 Indledning + DE2 som aktør
<b>v1.0</b>	24-03-2014 Hele gruppen (efter 1. review)
<b>v0.5</b>	20-03-2014 Hele gruppen

## 2.1 Indledning

Med udgangspunkt i børnesikkerhed i hjemmet vil vi udvikle et produkt, som kan hjælpe familier med børn, til at få et mere sikkert hjem.

Af problemstillinger som kan opstå i en almindelig husholdning kan nævnes:

- Fare for at et barn tænder for en kogeplade, eller andre elektriske varme aggregater, og efterfølgende kan brænde sig
- Fare for at et barn kan skære sig på køkkenknive som ligger i en skuffe

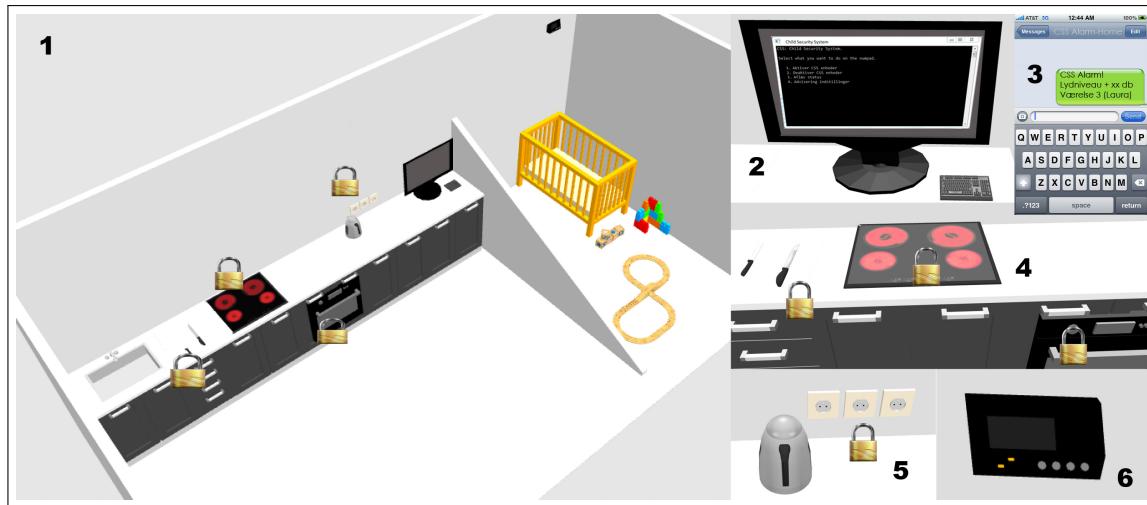
Den anden del af systemet er en babyalarm. Næsten alle mennesker i Danmark har deres mobiltelefon i nærheden hele tiden, så i stedet for at skulle have en babyalarm med rundt også, så kan man koble sin mobil til systemet og få besked når barnet giver lyd fra sig.

Dette ender ud i tre produkter:

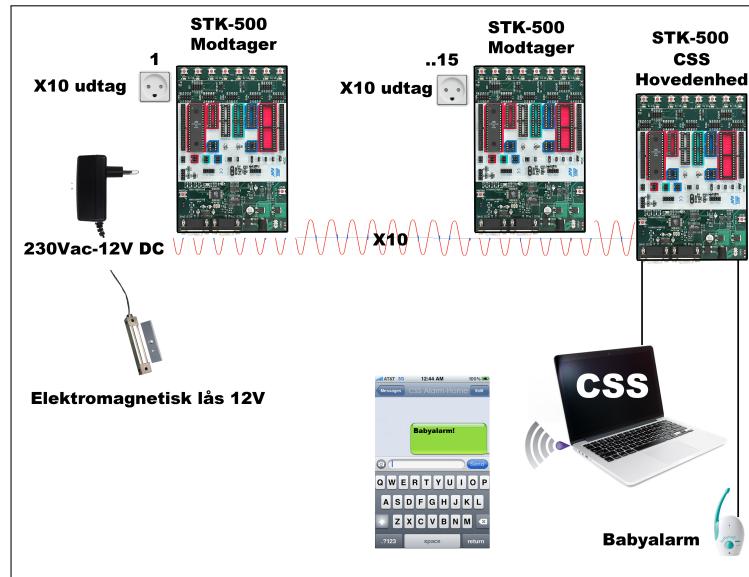
- Afbryder til valgt 230 Vac stikkontakt
  - Beskyttelse mod kogeplader og lignende
- Låsemekanisme til at låse skabe og skuffer
  - Aflæsning af skuffe med køkkenknive
- Babyalarm til lyddetektering
  - SMS-beskeder i stedet for en ekstra ”boks” i lommen

Systemet skal være nemt at sætte op og skal kommunikere over det eksisterende 230 V vekselspændings netværk i hus installationen.

En central enhed håndterer styringen i mellem enhederne og der skal være mulighed for at tilkoble en computer som kan bruges til at styre og aflæse systemet. Hele systemet kan aktiveres med et kodetryk.

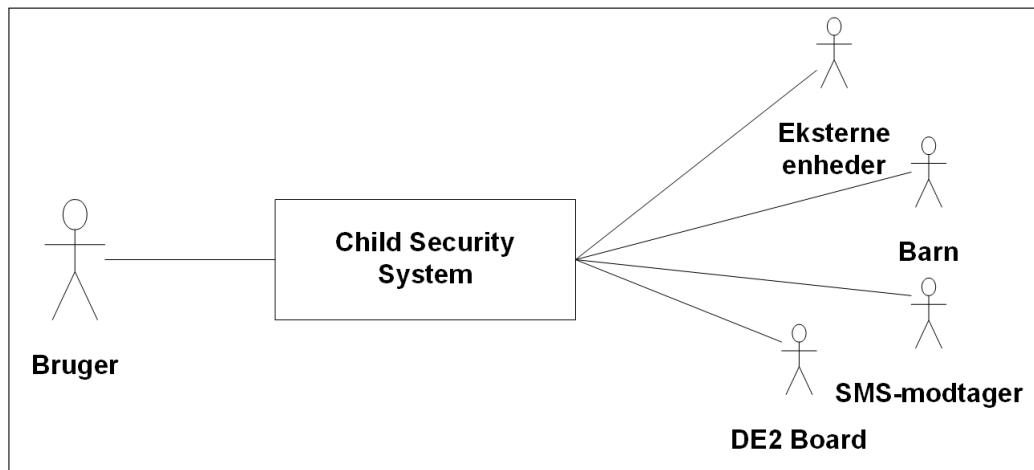
*Figur 2.1.* Installationsoversigt

1. Samlet oversigtstegning af CSS.
2. CSS programmet med tilhørende DE2 kodelås
3. SMS besked udsendt af systemet idet lydniveauet i værelse 3 (Laura) har været over det tilladte.
4. Overblik over hvad systemet er tiltænkt at børnesikre. Køkken skuffe med skarpe genstande, kogeplader, ovn.
5. 230V udtag. X10 styret, således at det bestemmes om der udtaget skal være aktivt.
6. Babyalarm. Illustrationen vil variere i forhold til virkeligheden.

*Figur 2.2.* Oversigt

Ud fra en kommando fra CSS programmet på computeren styres ønskede 230V udtag i hjemmet. Dette er muligt ved at benytte sig af X10 protokollen. Testmiljøet er illustreret via figur 2.2. Her sender CSS programmet besked til Hovedenheden som giver Modtageren besked på at hhv. tænde eller slukket for et givent udtag. Hvad brugen tilslutter i de forskellige udtag står frit for. Ydermere er der på X10 senderen koblet en lyddetektor som via computeren sender en sms ud via API.

## 2.2 Aktører



*Figur 2.3.* Kontekst diagram

### 2.2.1 Bruger

Type Beskrivelse	Bruger aktøren er ejeren af systemet eller den voksne med adgang til Computeren. Vil typisk være forældre, barnevært osv. (Primær)
------------------	------------------------------------------------------------------------------------------------------------------------------------

### 2.2.2 Eksterne enheder

Type Beskrivelse	Eksterne enheder, omfatter hvad man ønsker at aflæse eller slukke for. Vil typisk være skabe, komfur, el-kedel osv. (Sekundær)
------------------	--------------------------------------------------------------------------------------------------------------------------------

### 2.2.3 Barn

Type Beskrivelse	Barnet eller børnene i huset, som systemet skal beskytte. (Sekundær)
------------------	----------------------------------------------------------------------

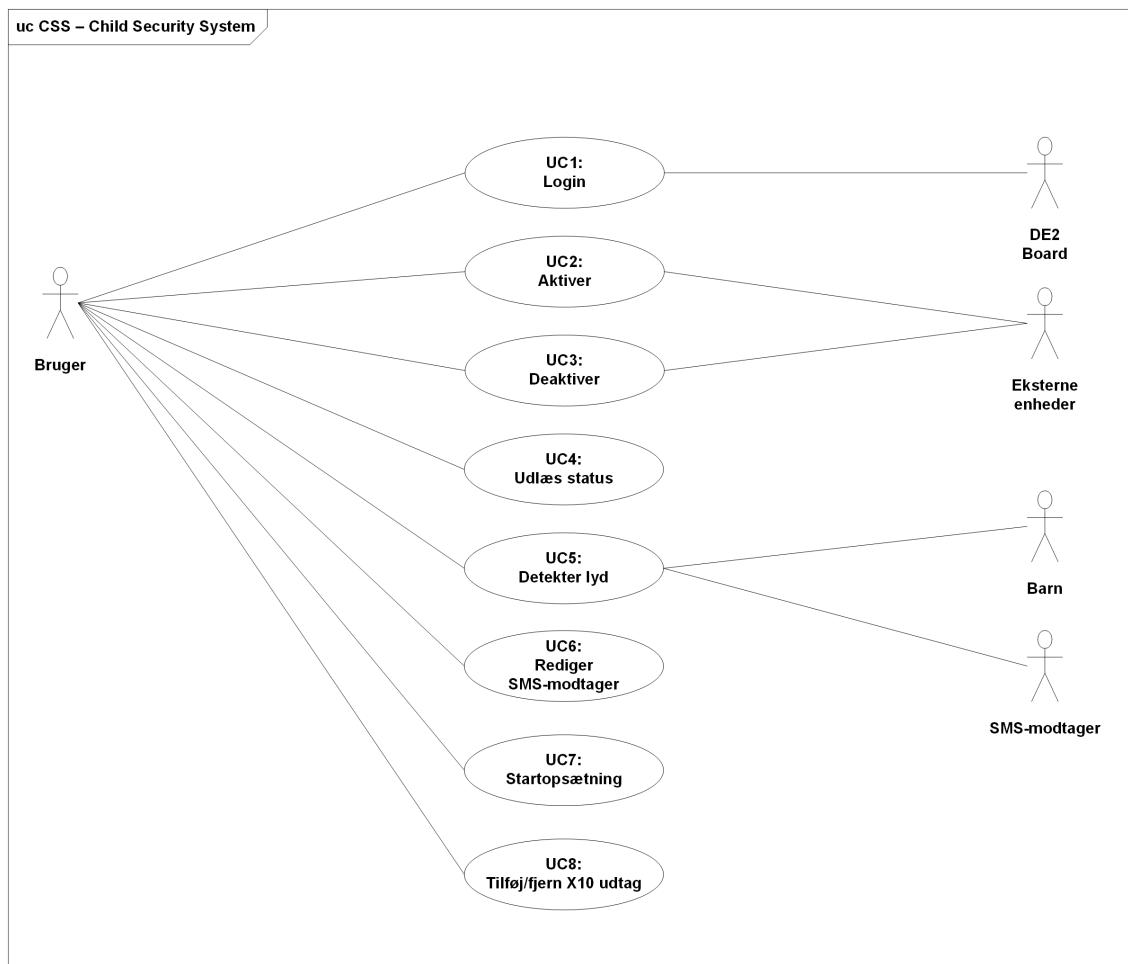
### 2.2.4 SMS modtager

Type Beskrivelse	Typisk forældrene eller barnevært. Den person der skal have besked om gråd eller anden støj fra børneværelset. (Sekundær)
------------------	---------------------------------------------------------------------------------------------------------------------------

### 2.2.5 DE2 Board

Type Beskrivelse	DE2 Board programmeret som kodelås i DSD øvelse 7 (Sekundær)
------------------	--------------------------------------------------------------

## 2.3 Usecases



Figur 2.4. Usecase diagram

### 2.3.1 UC1: Login

<b>Mål</b>	At Bruger kan logge ind ved hjælp af adgangskode
<b>Initialisering</b>	Bruger vælger login i interface
<b>Aktører og Stakeholders</b>	Bruger(Primær), DE2 Board(Sekundær)
<b>Referencer</b>	Ingen
<b>Antal af samtidige hændelser</b>	1
<b>Forudsætning</b>	At interfacet er tændt
<b>Efterfølgende tilstand</b>	At bruger er logget ind og hovedmenu vises på skærmen. Hele systemet er klar til brug
<b>Hovedforløb</b>	<ol style="list-style-type: none"> <li>1. Bruger vælger login i interfacet</li> <li>2. Bruger indtaster 3 adgangskoder adskilt af "Enter" på DE2 board [Undtagelse 2a] Bruger vælger Annuller</li> <li>3. Bruger får adgang til hovedmenuen</li> </ol>
<b>Undtagelser</b>	<ol style="list-style-type: none"> <li>2a. Bruger vælger annuller og kommer tilbage til startskærm</li> </ol>

### 2.3.2 UC2: Aktiver

<b>Mål</b>	At Bruger kan aktivere enkelte eller alle enheder, i systemet
<b>Initialisering</b>	Bruger vælger ”Aktiver” hovedmenu
<b>Aktører og Stakeholders</b>	Bruger(Primær), Eksterne enheder(Sekundær)
<b>Referencer</b>	UC1: Login
<b>Antal af samtidige hændelser</b>	1
<b>Forudsætning</b>	Bruger er logget ind (UC1: Login)
<b>Efterfølgende tilstand</b>	Enkelte eller alle enheder er aktiveret
<b>Hovedforløb</b>	<ol style="list-style-type: none"> <li>1. Bruger vælger ”Aktiver” i hovedmenu</li> <li>2. UI viser mulige enheder samt ”Vælg alle”, ”Aktiver” og ”Tilbage”</li> <li>3. Bruger markerer ønskede enheder til aktivering</li> <li>4. Bruger vælger ”Aktiver”  <b>[Undtagelse 4a]</b> Bruger vælger ”Tilbage”</li> <li>5. Systemet aktiverer valgte enheder  <b>[Undtagelse 5a]</b> Ingen valgte enheder</li> <li>6. UI viser besked om at enheder, er aktiverede</li> <li>7. UI returnerer til hovedmenu</li> </ol>
<b>Undtagelser</b>	<ol style="list-style-type: none"> <li>4a. UI returnerer til hovedmenu og UC2 afbrydes</li> <li>5a. Hvis ingen unit er valgt udskrives en fejl på skærmen og beder brugeren om at vælge en unit og går til UC2.2</li> </ol>

### 2.3.3 UC3: Deaktiver

<b>Mål</b>	At Bruger kan deaktivere enkelte eller alle enheder, i systemet.
<b>Initialisering</b>	Bruger vælger ”Deaktiver”
<b>Aktører og Stakeholders</b>	Bruger(Primær), Eksterne enheder(Sekundær)
<b>Referencer</b>	UC1: Login
<b>Antal af samtidige hændelser</b>	1
<b>Forudsætning</b>	Bruger er logget ind (UC1: Login)
<b>Efterfølgende tilstand</b>	Enkelte eller alle enheder er deaktiveret
<b>Hovedforløb</b>	<ol style="list-style-type: none"> <li>1. Bruger vælger ”Deaktiver” i hovedmenu</li> <li>2. UI viser mulige enheder samt ”Vælg alle”, ”Deaktiver” og ”Tilbage”</li> <li>3. Bruger markerer ønskede enheder til deaktivering</li> <li>4. Bruger vælger ”Deaktiver”           <ul style="list-style-type: none"> <li><b>[Undtagelse 4a]</b> Bruger vælger ”Tilbage”</li> </ul> </li> <li>5. Systemet deaktiverer valgte enheder           <ul style="list-style-type: none"> <li><b>[Undtagelse 5a]</b> Ingen valgte enheder</li> </ul> </li> <li>6. UI viser besked om at enheder, er deaktiverede</li> <li>7. UI returnerer til hovedmenu</li> </ol>
<b>Undtagelser</b>	<ol style="list-style-type: none"> <li>4a. UI returnerer til hovedmenu og UC3 afbrydes</li> <li>5a. Hvis ingen enheder er valgt udskrives en fejl på skærmen og beder brugeren om at vælge en enhed og går til UC3.2</li> </ol>

### 2.3.4 UC4: Udlæs status

<b>Mål</b>	At udlæse status
<b>Initialisering</b>	Bruger vælger ”Udlæs status”
<b>Aktører og Stakeholders</b>	Bruger(Primær)
<b>Referencer</b>	Ingen
<b>Antal af samtidige hændelser</b>	1
<b>Forudsætning</b>	Systemet er tændt
<b>Efterfølgende tilstand</b>	Systemet viser hovedmenu
<b>Hovedforløb</b>	<ol style="list-style-type: none"> <li>1. Bruger vælger ”Udlæs status”</li> <li>2. Status vises</li> <li>3. Bruger vælger ”Tilbage”</li> </ol>
<b>Undtagelser</b>	Ingen

### 2.3.5 UC5: Detekter lyd

<b>Mål</b>	At underrette SMS-modtager ved lyddetektion
<b>Initialisering</b>	Barn <sup>1</sup> afgiver lyd
<b>Aktører og Stakeholders</b>	SMS-modtager(Primær), Barn(Sekundær)
<b>Referencer</b>	Ingen
<b>Antal af samtidige hændelser</b>	1
<b>Forudsætning</b>	At systemet er tændt og har forbindelse til internettet
<b>Efterfølgende tilstand</b>	Lyddetektor stadig aktiv
<b>Hovedforløb</b>	<ol style="list-style-type: none"> <li>1. Lyddetektor er aktiveret</li> <li>2. Lyddetektor detekterer lyd</li> <li>3. Systemet underrettes</li> <li>4. Systemet afsender SMS</li> </ol>
<b>Undtagelser</b>	Ingen

### 2.3.6 UC6: Rediger SMS-modtager

<b>Mål</b>	At bruger kan ændre SMS-modtager i systemet
<b>Initialisering</b>	Bruger vælger "Rediger SMS-modtager"
<b>Aktører og Stakeholders</b>	Bruger(Primær)
<b>Referencer</b>	UC1: Login
<b>Antal af samtidige hændelser</b>	1
<b>Forudsætning</b>	Bruger er logget ind (UC1: Login)
<b>Efterfølgende tilstand</b>	Hovedmenu vises
<b>Hovedforløb</b>	<ol style="list-style-type: none"> <li>1. Bruger vælger "Rediger SMS-modtager"</li> <li>2. Bruger fortager ændringer af telefonnummer hvortil avisering sendes og bekræftes</li> </ol> <p>[Undtagelse 2a] Bruger vælger Annuler</p>
<b>Undtagelser</b>	<ol style="list-style-type: none"> <li>2a. Bruger vælger annuler og kommer tilbage til hovedmenu</li> </ol>

<sup>1</sup>Grunden til at initialisering foretages af et Barn, er fordi at formålet med lyddetektoren er at fungere som babyalarm.

### 2.3.7 UC7: Startopsætning

UC7: Startopsætning	
Mål	At brugeren kan opsætte systemet første gang.
Initialisering	Bruger starter systemet første gang
Aktører og Stakeholders	Bruger(Primær)
Referencer	UC8: Tilføj/Fjern X10 udtag
Antal af samtidige hændelser	1
Forudsætning	Ingen
Efterfølgende tilstand	Systemet er fuldt opsat
Hovedforløb	<p>1. Bruger sætter følgende kabler sammen:</p> <p>Serielt RS-232 kabel mellem hovedenhedens COM-port og computer</p> <p>Medfølgende styrekabel til lyddetektor forbindes mellem hovedenhed og lyddetektor</p> <p>Strømkabel fra et ledigt 230 Vac udtag til hovedenhedens AC indgang</p> <p>2. Bruger tænder for hovedenhed og computer på Tænd/Sluk knappen</p> <p>3. CSS programmet startes på computeren (UC1: Login gennemføres)</p> <p>4. UC8: Tilføj/fjern X10 udtag udføres</p> <p>5. Punkt 4 gentages med antallet af X10 udtag der ønskes opsat</p> <p>6. UC6: Ændre SMS-modtager udføres</p>

### 2.3.8 UC8: Tilføj/fjern X10 udtag

UC8: Tilføj/fjern X10 udtag	
Mål	At brugeren kan tilføje en ny enhed til CSS
Initialisering	Bruger
Aktører og Stakeholders	Bruger(Primær)
Referencer	UC1: Login
Antal af samtidige hændelser	1
Forudsætning	Bruger er logget ind (UC1: Login)
Efterfølgende tilstand	Et nyt X10 udtag er tilføjet

...fortsat fra forrige side

<b>UC8: Tilføj/fjern X10 udtag</b>	
<b>Hovedforløb</b>	<p>1. Bruger vælger menupunkt "Tilføj/fjern X10 udtag" (UC1 gennemføres) og programmet udskriver i forvejen indstillede X10 udtag og mulighed for at vælge tilføj eller fjern</p> <p><b>Tilføj valgt</b></p> <ul style="list-style-type: none"> <li>a) Bruger indstiller addresseswitchen til en adresse på X10 udtaget</li> <li>b) Bruger indtaster den fire cifrede kombination som er indstillet på X10 udtaget efterfulgt af "Enter"</li> </ul> <p><b>[Undtagelse 1b.a]</b> Adressen er ikke unik</p> <p><b>[Undtagelse 1b.b]</b> Adressen har ikke den rette længde</p> <ul style="list-style-type: none"> <li>c) Programmet udskriver beskeden "Indtast navn"</li> <li>d) Bruger indtaster et selvvalgt navn for X10 udtaget efterfulgt af "enter"</li> </ul> <p><b>[Undtagelse 1d.a]</b> Navnet har ikke den rette længde</p> <ul style="list-style-type: none"> <li>e) Bruger sætter X10 udtaget i det ønskede 230 Vac udtag</li> </ul> <p><b>Fjern valgt</b></p> <ul style="list-style-type: none"> <li>a) Den ønskede enhed markeres og der trykkes Fjern</li> </ul> <p>2. Programmet returnerer til hovedskærmen</p>
<b>Undtagelser</b>	<p>1b.a. Programmet udskriver fejlmeddelelsen "Adressen er ikke unik. Vælg en ny."</p> <p>Gå til UC8.1a</p> <p>1b.b. Programmet udskriver fejlmeddelelsen "Adressen har ikke den rette længde. Vælg en ny."</p> <p>Gå til UC8.1a</p> <p>1d.a. Programmet udskriver fejlmeddelelsen "Navnet skal minimum have to og maximum 50 karaktere"</p> <p>Gå til UC8.1c</p>

## 2.4 Ikke-funktionelle krav

### Brugbarhed (Usability)

1. UI skal kunne bruges efter gennemlæst manual.

### Pålidelighed (Reliability)

2. Levetid: 5 år uden hardware nedbrud
3. Software oppetid: Minimum 1 måned før genstart

### Ydeevne (Performance)

4. System respons må maksimalt være 2,5 sekunder
5. Startuptid fra power-off til funktionel tilstand maksimalt 2 minutter
6. Systemkapaciteten er på maksimalt 15 CSS udtag
7. Ved lyddetektion må der maksimalt gå 1 minut før SMS-besked er afsendt

### Vedligeholdelse (Supportability)

8. X10 udtag kan udskiftes separat ved simpel omkodning ved hjælp af addresseswitchen
9. Systemet er plug'n'play i en almindelig husholdning
10. X10 udtag kan tilføjes og installeres løbende

### Generelle krav

11. Systemet skal virke på det eksisterende 230 Vac netværk i almindelige husstande
12. Kommunikationen mellem X10 udtag og hovedenheden skal ske på X10 protokollen
13. Systemet skal kunne afsende SMS-beskeder
14. Systemet skal automatisk logge ud efter 1min uden aktivitet

### CSS enheder

15. Udtag skal kunne være i en 1,5 moduls Fuga stikdåse
16. Udtag skal have en LED indikator som viser at den er aktiv
17. Hovedenheden skal kunne virke på 230 Vac/13 A tilslutning

### Eksterne enheder

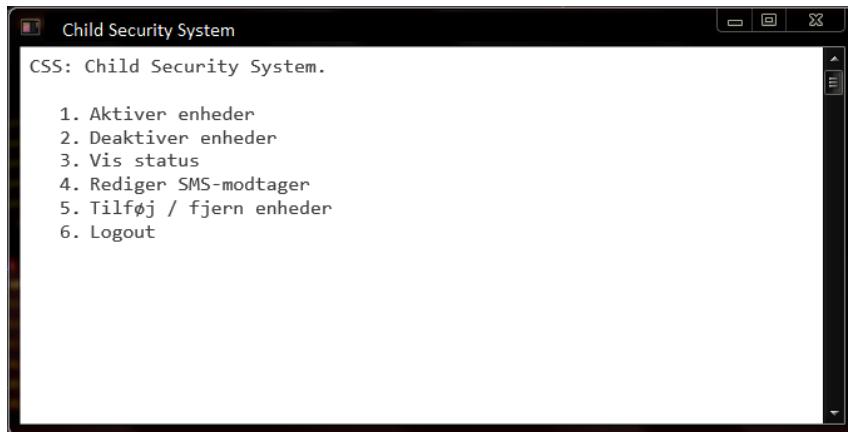
18. Lyddetektoren skal registrere lyde på over 68 dB
19. Der må maksimalt afsendes 1 SMS-besked pr. minut ved gentagende reaktion fra lyddetektoren
20. Låse enheder må maksimalt være 8x5x3 cm
21. Låse enhederne skal kunne holde 5 kilogram

## 2.5 Begrænsninger

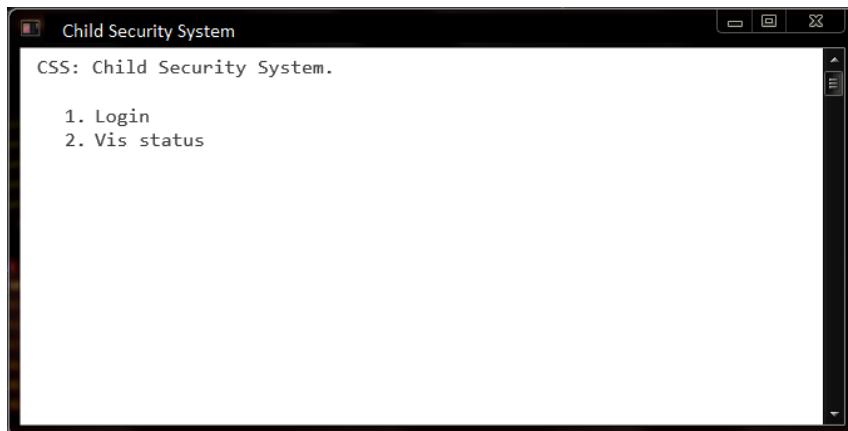
- Prototypen udføres i et 18 Vac testmiljø
- I stedet for magnetlåse til at simulere låsemekanismen bruges en lysindikator
- Prototypen udføres med et STK500 kit, hvorfor krav til dimensionerne frafalder

## 2.6 HMI(Human Machine Interface)

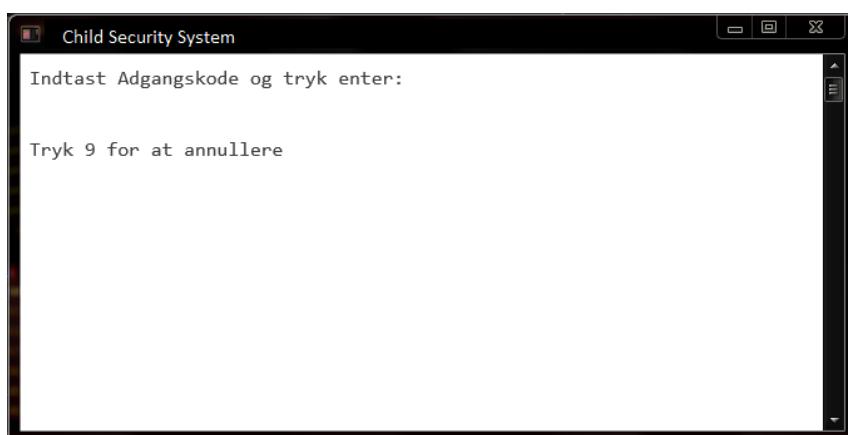
Billederne er inverteret for læsbarhedens skyld.



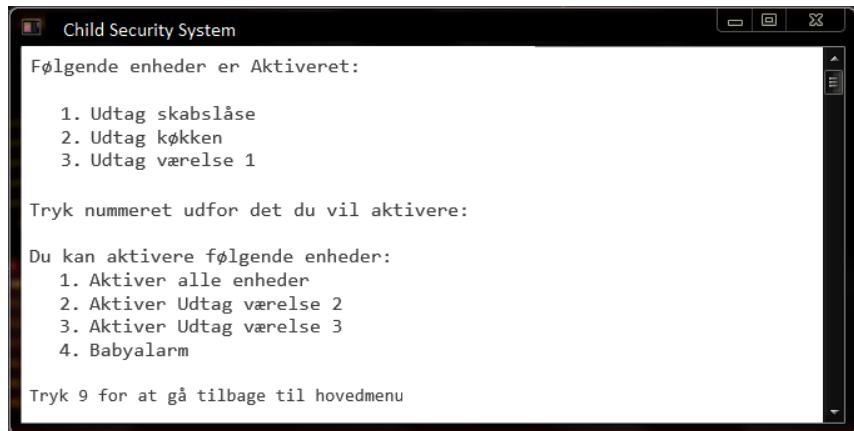
*Figur 2.5.* CSS Menu



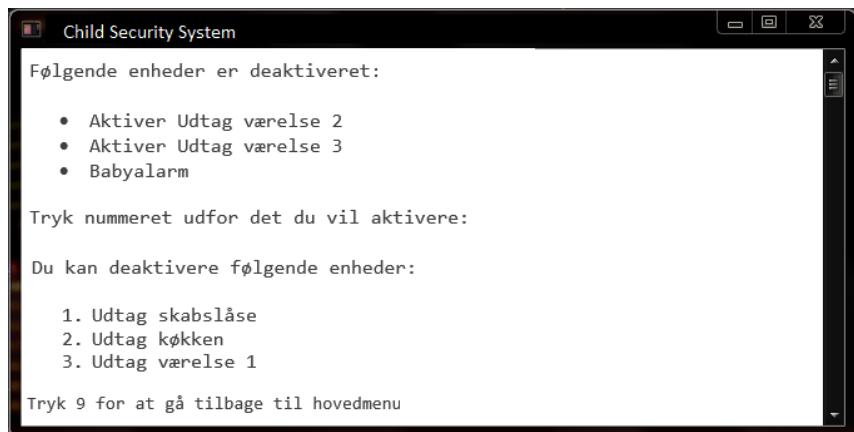
*Figur 2.6.* CSS Pre-Login



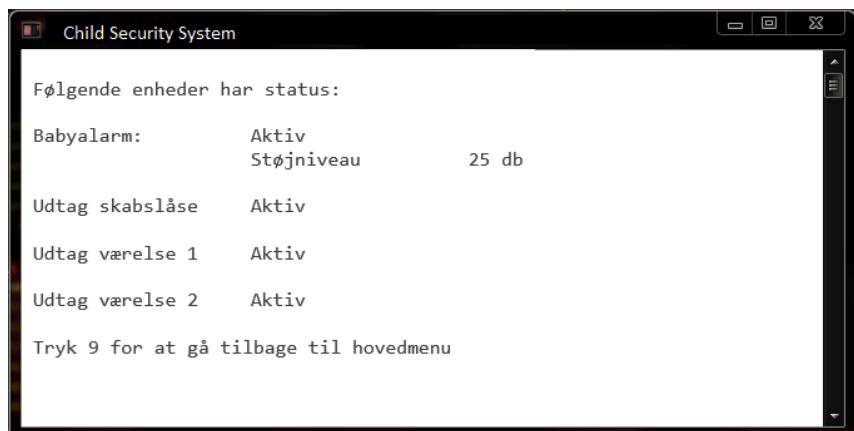
*Figur 2.7.* CSS Login



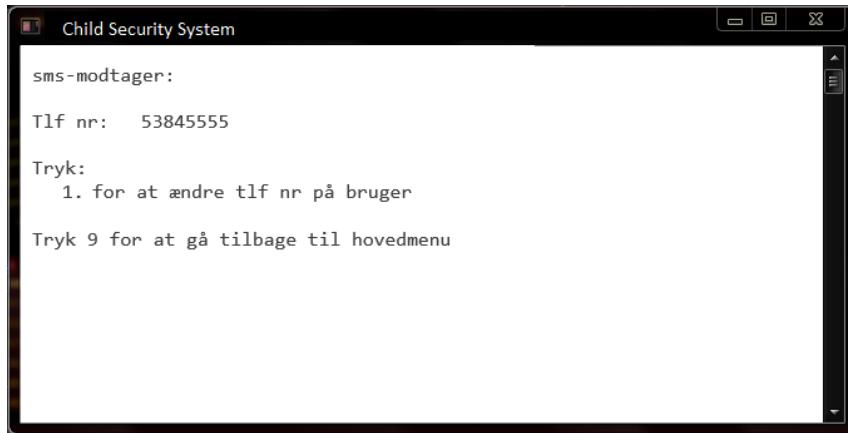
*Figur 2.8.* CSS Aktiver



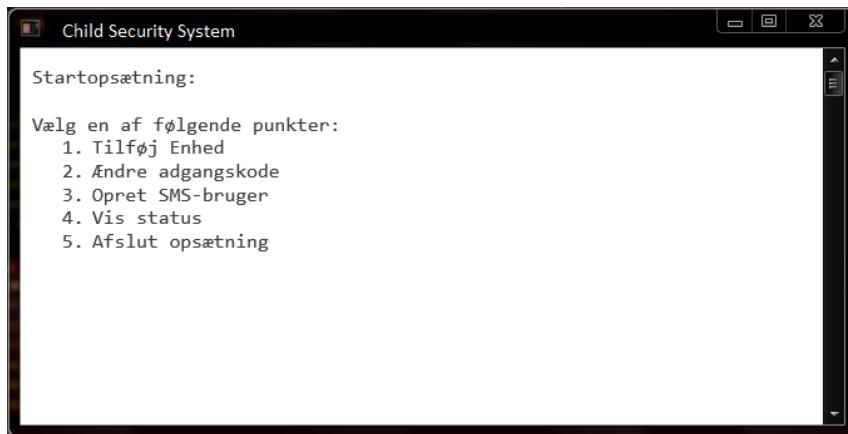
*Figur 2.9.* CSS Deaktivér



*Figur 2.10.* CSS Vis Status



*Figur 2.11.* CSS Advisering



*Figur 2.12.* CSS Startopsætning

# Forundersøgelse 3

## 3.1 GSM

<b>Løsning</b>	GSM Modul
<b>Producent</b>	Cinterion
<b>Interface</b>	I2C, SPI, USB
<b>Beskrivelse</b>	Hardware modul der kan tilkobles X10'eren via SPI
<b>Krav</b>	SIM kort og indgående programerings kendskab
<b>Fordeler</b>	Mest pålidelige løsning og ingen forsinkelse på SMS'er
<b>Ulemper</b>	Kræver viden inden for Java eller Microsoft Windows Mobile programering
<b>Pris</b>	563,23 - 656,34 + SMS takst
<b>Link</b>	<a href="http://dk.farnell.com/cinterion/mc75i/module-gsm-gprs-edge-quad-band/dp/1718875">http://dk.farnell.com/cinterion/mc75i/module-gsm-gprs-edge-quad-band/dp/1718875</a> <a href="http://dk.farnell.com/cinterion/tc65i/module-gsm-gprs-quad-band-tcp-ip/dp/1718877">http://dk.farnell.com/cinterion/tc65i/module-gsm-gprs-quad-band-tcp-ip/dp/1718877</a>

<b>Løsning</b>	API
<b>Producent</b>	Clickatell
<b>Interface</b>	HTTP, HTTPS, FTP, SMPP, XML, SOAP, SMTP, COM obj.
<b>Beskrivelse</b>	Software baseret API modul
<b>Krav</b>	Forbindelse til internettet
<b>Fordele</b>	Let at programere
<b>Ulemper</b>	Kräver forbindelse til internettet
<b>Pris</b>	0,762 kr. pr. SMS
<b>Link</b>	<a href="https://www.clickatell.com/apis-scripts/">https://www.clickatell.com/apis-scripts/</a>



<b>Løsning</b>	Arduino + GSM shield
<b>Producent</b>	Arduino
<b>Interface</b>	Internt
<b>Beskrivelse</b>	Single-board computer med GSM modul
<b>Krav</b>	SIM kort
<b>Fordele</b>	Let at programere
<b>Ulemper</b>	
<b>Pris</b>	149,- + 515,- + SMS takst
<b>Link</b>	<a href="http://arduino.cc/">http://arduino.cc/</a>



### 3.1.1 GSM-valg

Vi har valgt at bruge Clickatell løsningen, da den er let at implementere, fleksibel og billig i opstarts omkostninger.

## 3.2 Lås

<b>Løsning</b>	Elektrisk karm lås TFS-A21
<b>Producent</b>	Ukendt
<b>Tilslutning</b>	12V DC - 0.6A
<b>Beskrivelse</b>	Elektrisk karm lås med bevægelig pal
<b>Krav</b>	Skal monteres med slutblæk
<b>Fordele</b>	
<b>Ulemper</b>	Slutstykket begrænser montering (udfræsning). Den bevægelige pal skal smøres.
<b>Pris</b>	65 kr
<b>Link</b>	<a href="http://goo.gl/SDvjkD">http://goo.gl/SDvjkD</a>



<b>Løsning</b>	Elektromagnetisk lås 60kg
<b>Producent</b>	KingGo
<b>Tilslutning</b>	12 V DC - 0.3A
<b>Beskrivelse</b>	Elektromagnetisk lås uden bevægelige dele
<b>Krav</b>	Skal monteres med metal stykke
<b>Fordele</b>	Skal kun skrues fast
<b>Ulemper</b>	
<b>Pris</b>	115 kr
<b>Link</b>	<a href="http://goo.gl/ewKYfa">http://goo.gl/ewKYfa</a>



### 3.2.1 Låsvalg

Valget et faldet på den elektromagnetiske lås fra KingGo. Denne lås er valg da den er simpel og let at sætte op, da der ikke skal fræses ud for at benytte denne type lås. Ydermere så vil låsen automatisk låse sig fast, hvis modtager pladen er ude for rækkevidde og denne fysisk skubbes hen til elektromagneten. I testmiljøet vil en 12V lyskilde agere lås.

### 3.3 Babyalarm

<b>Navn</b>	Philips SCD505
<b>Rækkevidde</b>	330
<b>Lyd:</b>	Justerbar lydniveau. Lys i forældreenheden angiver lydniveau ved babyen
<b>Batteri</b>	Delvist genopladelig
<b>Stråling</b>	Høj
<b>Pris</b>	549 kr
<b>Link</b>	<a href="http://goo.gl/pw06P9">http://goo.gl/pw06P9</a>



<b>Navn</b>	Supernova D7
<b>Rækkevidde</b>	600m
<b>Lyd</b>	Ukendt
<b>Batteri</b>	2 genopladelige batterier
<b>Pris</b>	999 kr
<b>Stråling</b>	Lav
<b>Link</b>	<a href="http://goo.gl/JFZcf5">http://goo.gl/JFZcf5</a>



#### 3.3.1 Babyalarm sammenligning

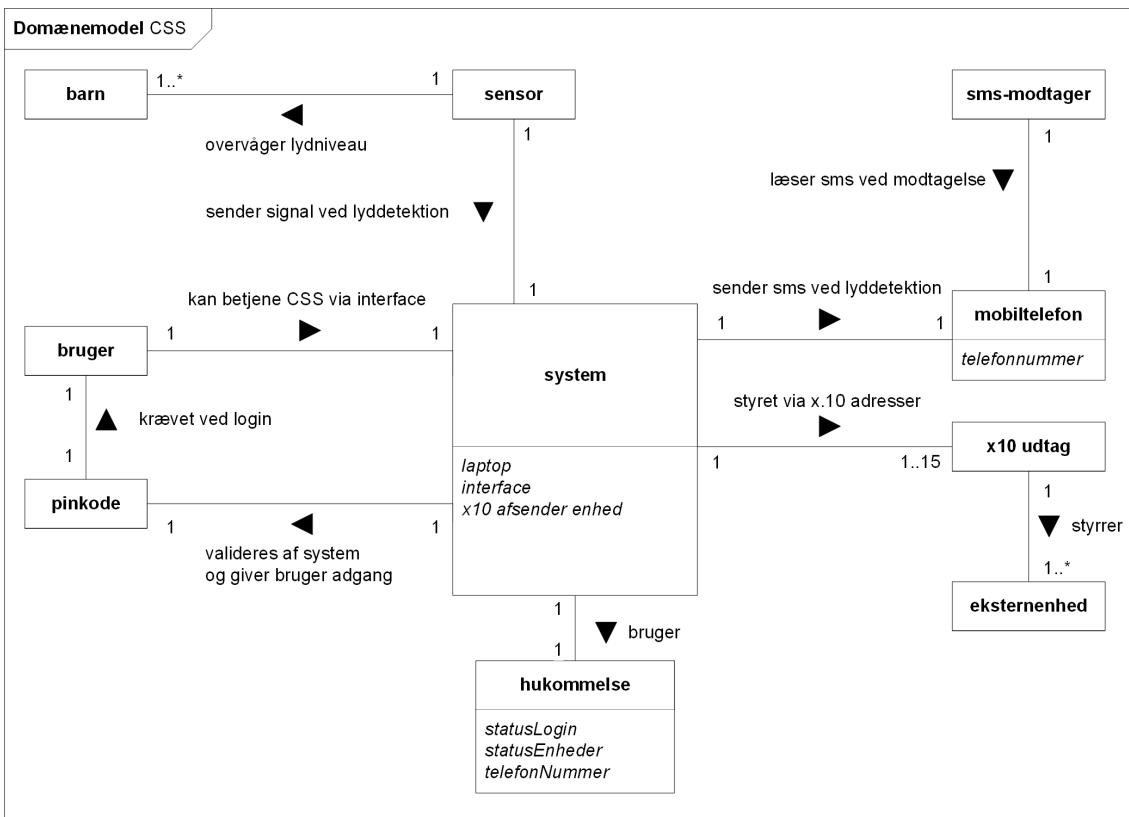
Vi har valgt at lave en forundersøgelse på babyalarmer for at få en indikation af hvad markedet tilbyder af babyalarmer imod det vi kommer til at tilbyde med vores babyalarm.

Vi har valgt at vores babyalarm skal reagere på lydniveauer højere end 40 dB. Da vores babyalarm ikke bliver trådløs så undgår vi ting som stråling og batteri tid. Vores rækkevidde bliver dog betydelig mindre end den typiske babyalarm, igen på grund af at den ikke er trådløs. Vores babyalarm er tænkt som en stationær enhed som kan placeres i et barneværelse og altså ikke som en transportabel babyalarm som man typisk ser.

# System Arkitektur 4

---

## 4.1 Domænemodel



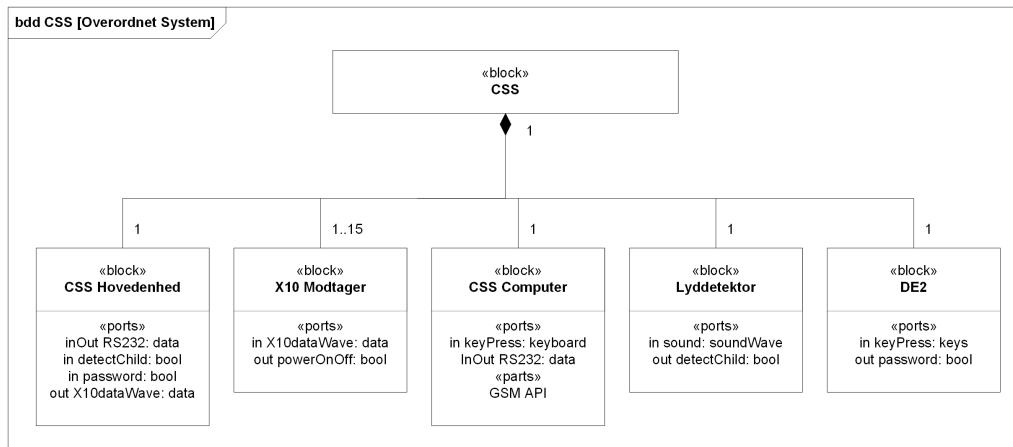
Figur 4.1. Domænemodel

Domænemodel er udarbejdet i samarbejde med kunden. Denne har til opgave at give et struktureret billede af systemets funktionalitet og sammenhæng. Domænemodellen gør ikke brug af fagudtryk, men pile og kortfattede samt præcise sætninger anvendes for at beskrive sammenhængen mellem blokkene. Dette er med til at opnå en højere forståelse, af systemet som helhed, for kunden.

## 4.2 Hardware

### 4.2.1 Hardware beskrivelse

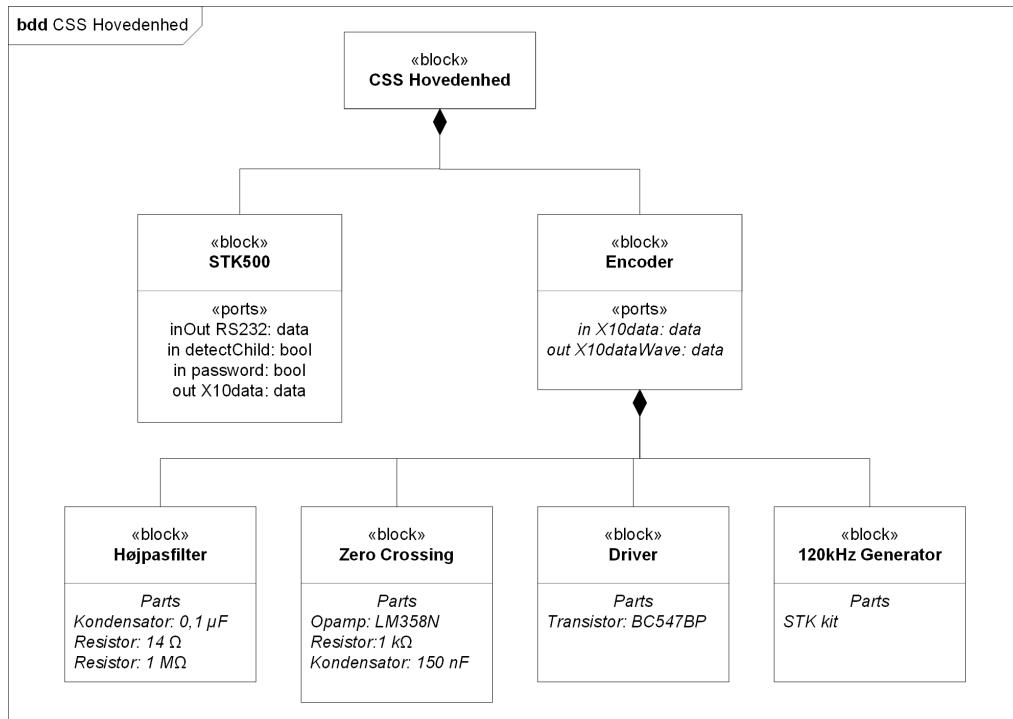
### 4.2.2 BDD Hardware



Figur 4.2. BDD Hardware

BDD diagrammet giver et overblik over hvad det samlede system består af. Vi ser en port beskrivelse som viser hvilke signaler hver blok består af.

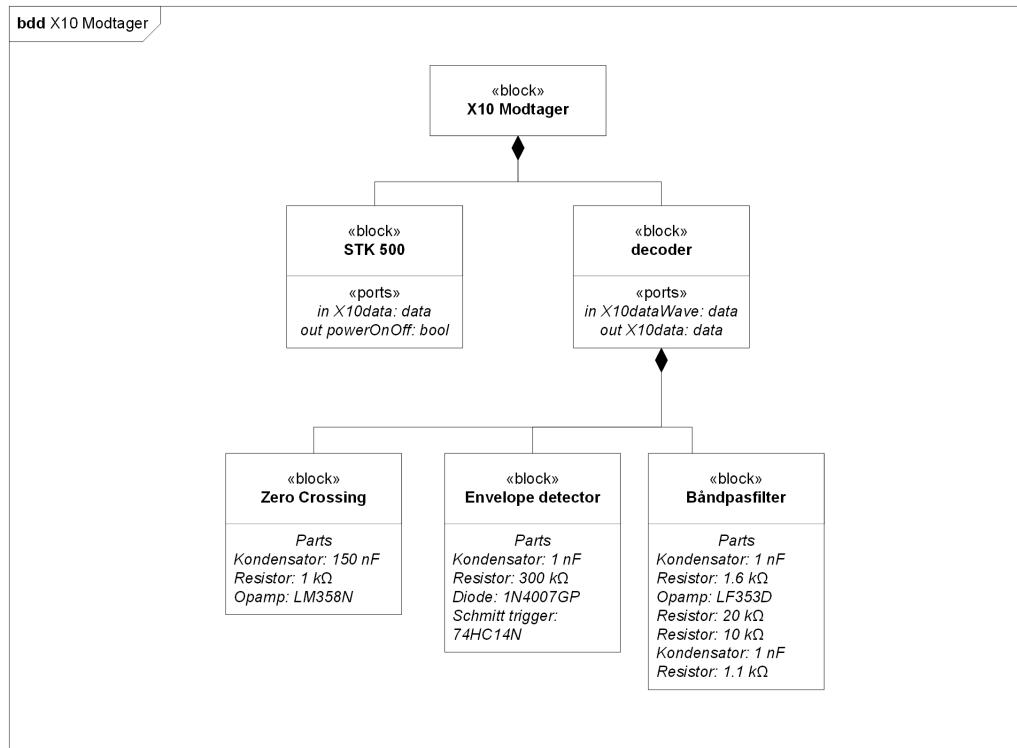
### 4.2.3 BDD Hovedenhed



Figur 4.3. BDD Hovedenhed

BDD diagrammet giver et overblik over hvad CSS hovedenheden består af. Vi ser en portbeskrivelse for STK kittet og encoder samt et overblik over hvilke komponenter encoderen består af.

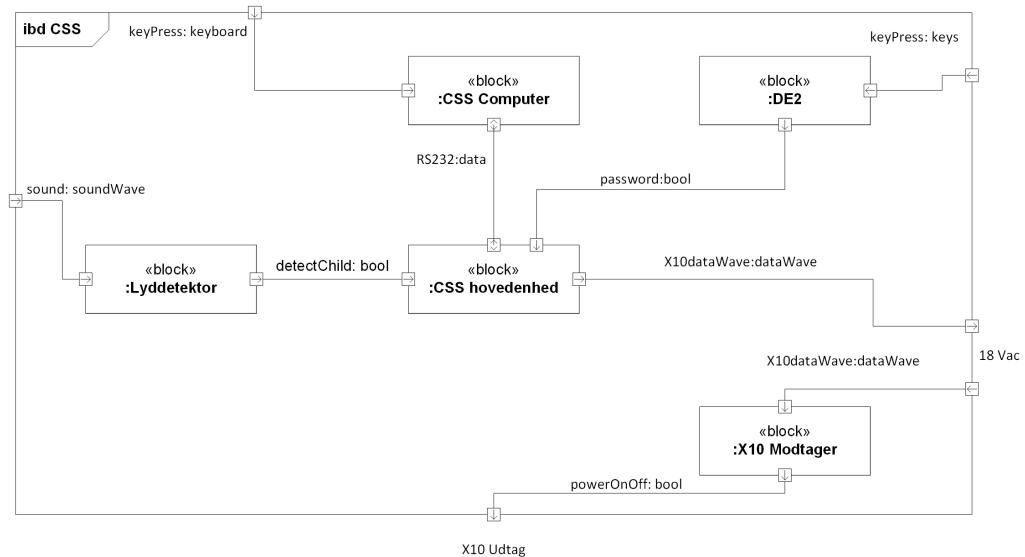
#### 4.2.4 BDD Modtager



*Figur 4.4.* BDD Modtager

BDD diagrammet giver et overblik over hvad X10 modtageren består af. Vi ser en portbeskrivelse for STK kittet og decoderen samt et overblik over hvilke komponenter decoderen består af.

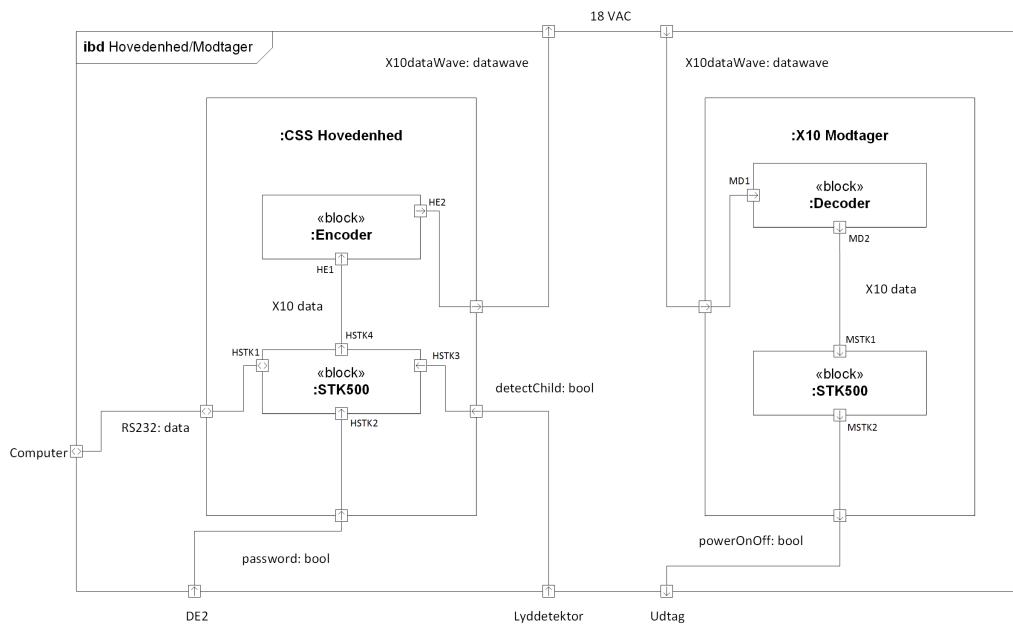
#### 4.2.5 IBD Hardware



*Figur 4.5.* IBD Hardware

IBD diagrammet giver et internt overblik over hvordan hele vores system er forbundet. Vi ser hvilke type signaler der bliver sendt imellem vores forskellige blokke.

#### 4.2.6 IBD Hovedenhed og Modtager



*Figur 4.6.* IBD Hovedenhed og Modtager

IBD diagrammet giver et internt overblik over hvordan vores X10 hovedenhed og X10 modtager er forbundet. Vi ser hvilke type signaler der bliver sendt imellem vores forskellige blokke.

### 4.2.7 Grænseflade

For at opnå forståelse for signaler mellem blokkene laves en grænseflade der beskriver de enkelte blokkes porte og hvilke signaler der løber mellem disse.

#### Blok beskrivelse

Til at beskrive blokkene nærmere er anvendt tabeller som ses herunder. Her er hvert signal i en respektiv blok kommenteret og blokkens funktion er kort beskrevet.

**Tabel 4.1.** Tabel med beskrivelse af respektive blokke

Bloknavn	Funktion	Signaler	Kommentar
Encoder	modtage kommando og encode til 120 kHz bursts	120 kHz	Data ud
		X10 data	X10 data kommando ind
STK500 Hovedenhed	Genererer burst og detekterer på zero-crossing	RS232	Laptop forbindelse
		X10 data	X10 data kommando linje
		Bool	lyd detektion
		Bool	Password accept
Decoder	Modtager 120 kHz og decoder til X10 data	120 kHz	120 kHz ind
		X10 data	Kommando linje
		120 kHz	120 kHz data ind
STK 500 Decoder	Modtager burst og detekterer på zero-crossing	X10 data	X10 data ind
		Bool	Power I/O ekstern enhed

### Signal beskrivelse

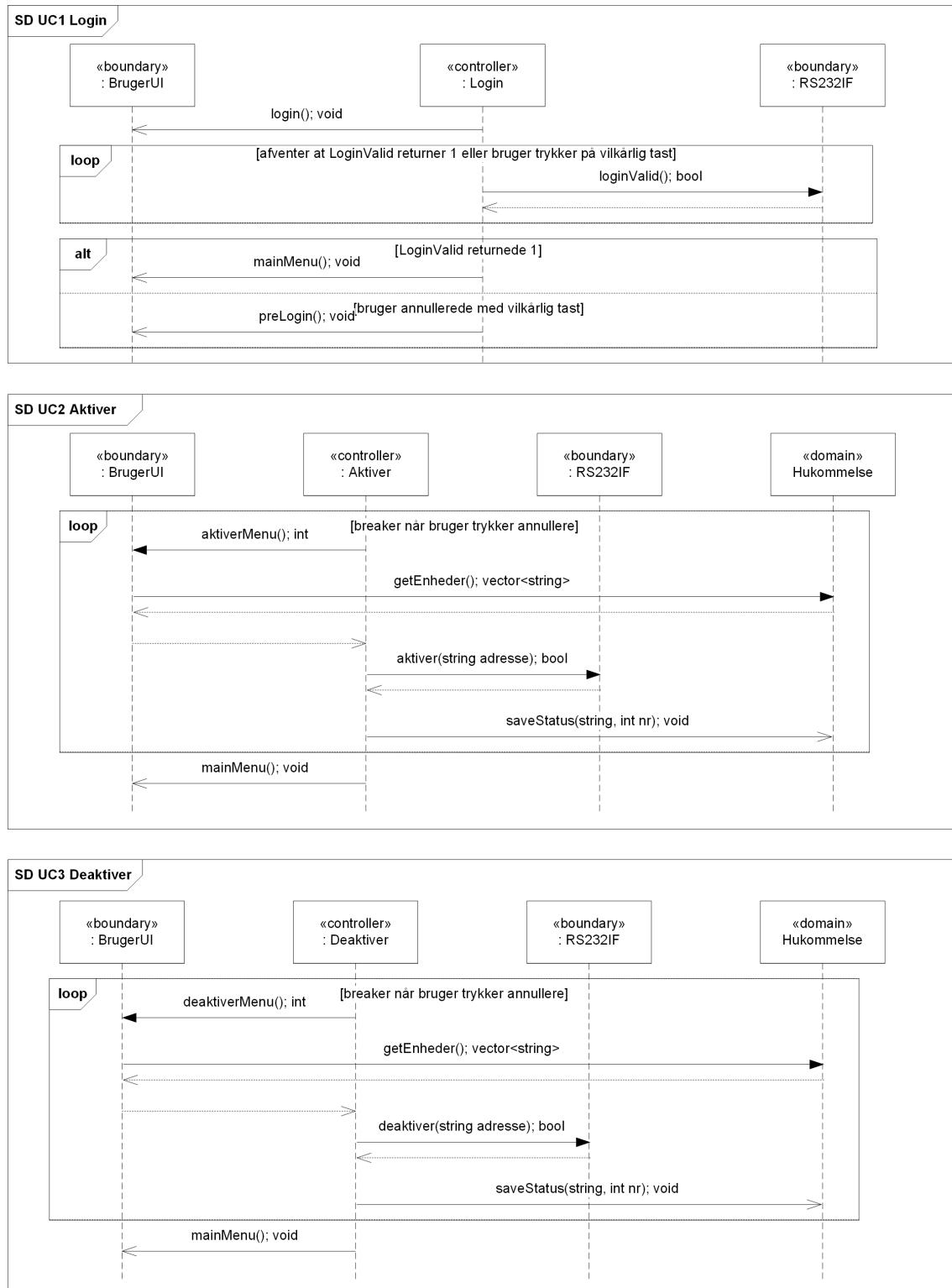
For at fuldende beskrivelsen af grænsefladen er der lavet en signaltabel som kan ses herunder. Hvert signal er beskrevet og tilknyttet en kort kommentar. Området et signal er defineret under er også beskrevet. Blok og terminal indgår også.

*Tabel 4.2.* Tabel over signaler med terminaler

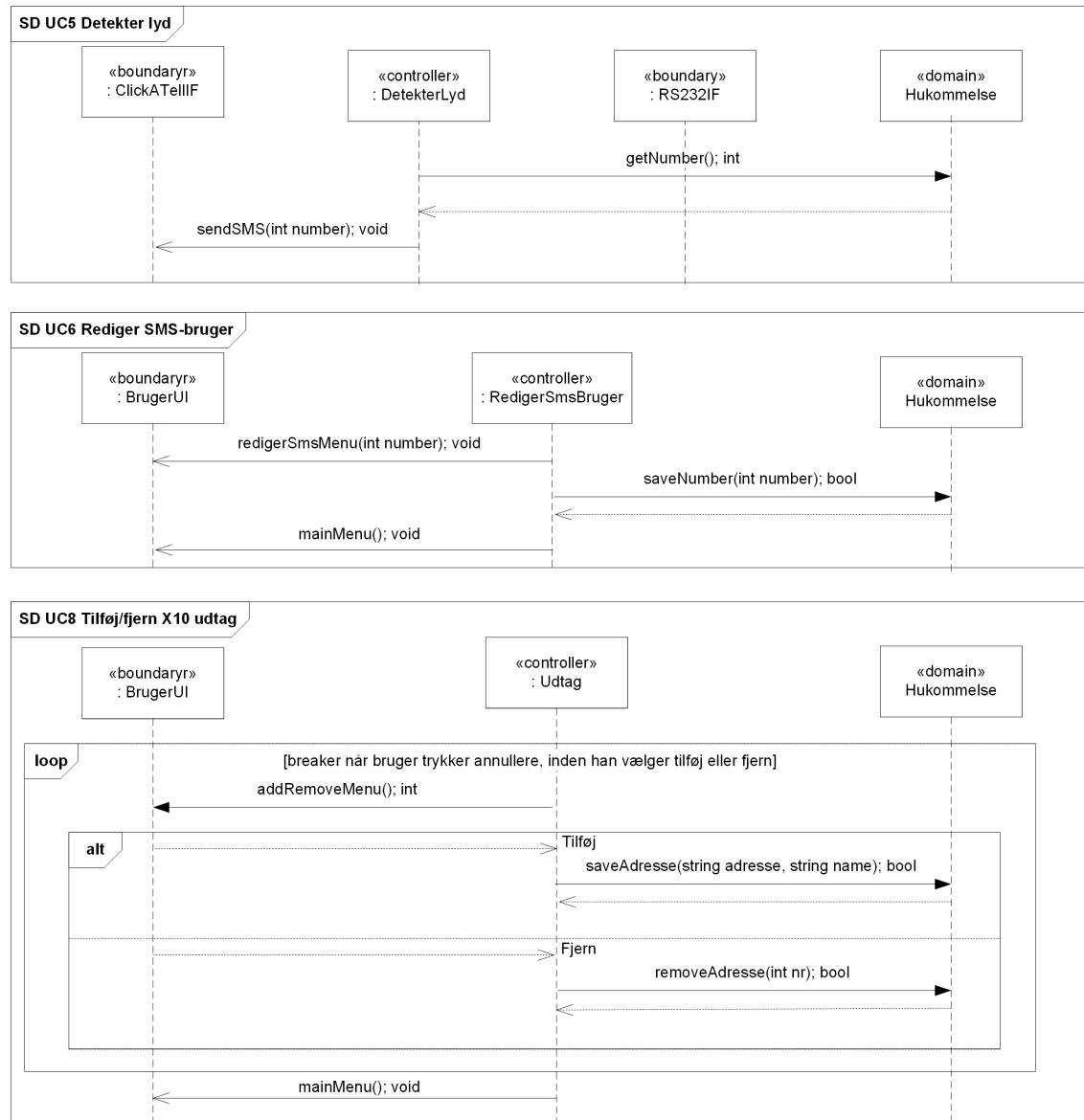
Signal-navn	Funktion	Område	Port 1	Port 2	Kommentar
120 kHz	sende kommando på 18V nettet		Encoder, HE2	Decoder, DM1	
X10 data	kommando		STK500, HSTK4	Encoder, HE1	
			Decoder, MD2	STK500, MSTK1	
bool	digital signal	5V/0V	DE2, N/A	STK500, HSTK2	
			Lyddetektor, N/A	STK500, HSTK3	
			STK500, MSTK2	Udtag, N/A	

## 4.3 Software

### 4.3.1 Applikations model for PC



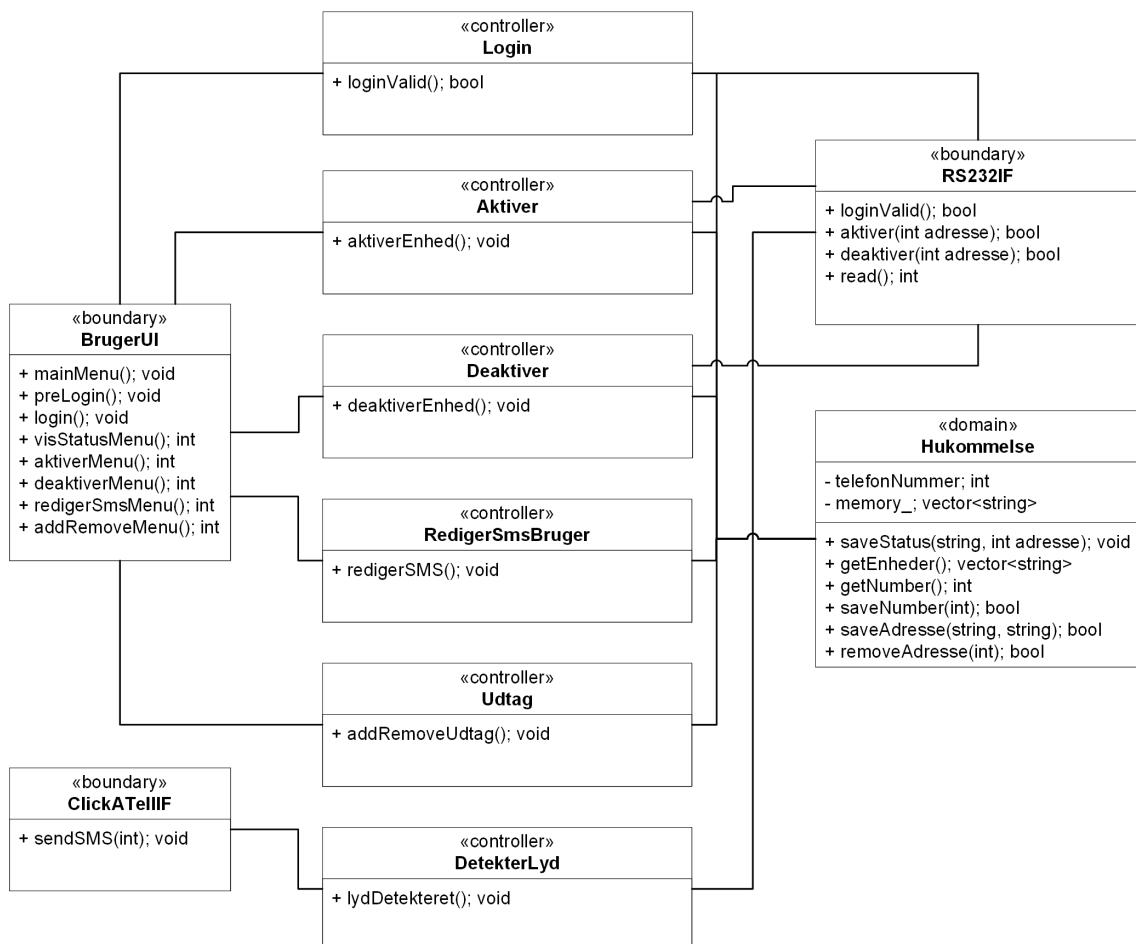
Figur 4.7. Use-case 1-3 sekvensdiagram for PC

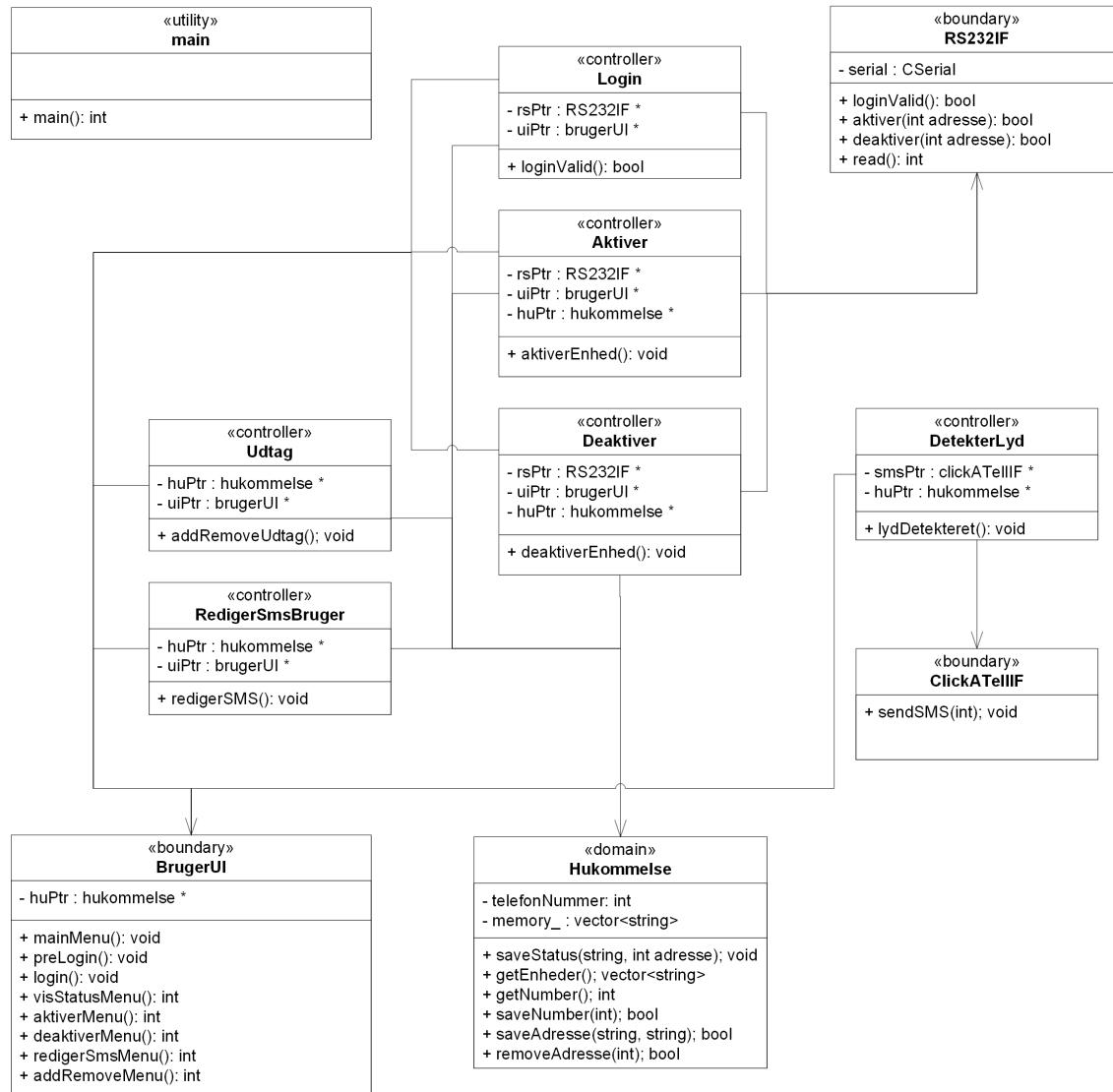


**Figur 4.8.** Use-case 5-8 sekvensdiagram for PC

Alle kaldene til controller klasserne kommer fra main.cpp programmet som tager imod bruger inputs i preLogin menuen og mainMenuen. Når mainMenuen er vist så står main.cpp og spørger på read() metoden i RS232IF klassen som tester om den har modtaget data fra STK kittet. Dette gør den for at se om login status har ændret sig eller om der skal sendes en sms til brugeren pga. babyalarmen.

Metode navnene i controller klasserne kan ses på næste sidste i klassediagrammet som er lavet på baggrund af Sekvens diagrammerne.

*Figur 4.9.* Klassediagram for PC



**Figur 4.10.** Statisk klassediagram for PC

Main opretter alle objekterne og pointerne til de klasser hvis constructor skal bruge dem. Derudover styre main hvilke controllers der bliver kaldt alt efter bruger input.

### 4.3.2 Klassebeskrivelse for PC

Her følger klassebeskrivelser for alle klasser til PC.

#### Hukommelse klasse

**Ansvar:** at håndtere hukommelsen så man kan lukke programmet og åbne det igen, og så stadig se det gemte telefonnr og de gemte enheder med status og navn.  
når der nævnes at ting gemmes i hukommelsen så menes der både hukommelse.txt filen og vectoren memory\_

void saveStatus(string, int adresse);

**Parametre:** string til bestemmelse af om status er aktiv eller deaktiv. Int adresse til bestemmelse af adressen

**Returværdi:** ingen

**Beskrivelse:** gemmer status på enheden på pågældende adresse

vector<string> getEnheder();

**Parametre:** ingen

**Returværdi:** vector som indeholder strings. Hver string i vectoren er en linje i hukommelses filen.

**Beskrivelse:** Skal returnere hukommelsen hvor adresse, navn og status på alle enheder er gemt

int getNumber();

**Parametre:** ingen

**Returværdi:** gemte telefonnummer

**Beskrivelse:** returnere det gemte telefonnummer

bool saveNumber(int number);

**Parametre:** number der skal gemmes

**Returværdi:** ingen

**Beskrivelse:** gemmer telefonnummeret. Både i hukommelsen og i den private variabel telefonNummer

bool saveAdresse(string adresse, string navn);

**Parametre:** 2 strings. første skal være adressen og anden parametre er navnet på enheden

**Returværdi:** ingen

**Beskrivelse:** Skal gemme enheden i hukommelsen. Status på nye oprettede enheder skal initialiseres som false (deaktive)

```
bool removeAdresse(int nr);
```

**Parametre:** modtager en integer som repræsentere nr på enheden i vectoren. Nr er dog ikke den direkte plads i vectoren da enhederne ligger fra plads nr 1 til 46. Hver enhed fylder 3 pladser i vectoren.

De ligger på følgende måde: adresse, navn, status

**Returværdi:** ingen

**Beskrivelse:** Skal slette enheden i hukommelsen

### Login klasse

**Ansvar:** at kontrollere forløbet under UC1

```
bool loginValid();
```

**Parametre:** ingen

**Returværdi:** returne true hvis login går godt. returnere false hvis brugeren annullere login forsøg.

**Beskrivelse:** loginValid metoden styrer forløbet under login forsøg. Kalder brugerUIs login() menu vha. en pointer. Derefter kalder den RS232IF validLogin metode. Enter while(!kbhit) hvor den kalder RS232IF read() metode indtil den returnere 1 eller brugeren trykker på en vilkårlig tast.

### Aktiver klasse

**Ansvar:** at kontrollere forløbet under UC2

```
void aktiverEnhed();
```

**Parametre:** ingen

**Returværdi:** ingen

**Beskrivelse:** enter en while(1) loop. Kalder brugerUI aktiverMenu(). Hvis returværdien er 27 (annullere) så skal forloopet breakes og mainMenu() skal kaldes. Ellers skal RS232IF metode aktiver() kaldes med adresse som parametre. Status skal gemmes vha hukommelses metoden saveStatus som skal have "true" som første parametre og nr. på enheden i anden parametre.

### Deaktiver klasse

**Ansvar:** at kontrollere forløbet under UC3

```
void deaktiverEnhed();
```

**Parametre:** ingen

**Returværdi:** ingen

**Beskrivelse:** enter en while(1) loop. Kalder brugerUI deaktiverMenu(). Hvis returværdien er 27 (annullere) så skal forloopet breakes og mainMenu() skal kaldes. Ellers skal RS232IF metode deaktiver() kaldes med adresse som parametre. Status skal gemmes vha hukommelses metoden saveStatus som skal have "false" som første parametre og nr. på enheden i anden parametre.

### **DetekterLyd klasse**

**Ansvar:** at hente nummer og sende det til clickATellIFs metode sendSMS(nummer)

void lydDetekteret();

**Parametre:** ingen

**Returværdi:** ingen

**Beskrivelse:** henter telefonnummer i hukommelse vha getNumber() og kalde ClickATellIF metoden sendSMS(int ) med det nummer den fik fra hukommelsen.

### **RedigerSmSBruger klasse**

**Ansvar:** at kontrollere forløbet under UC6

void redigerSMS(int number);

**Parametre:** nye nummer

**Returværdi:** ingen

**Beskrivelse:** gemmer nye nummer i hukommelsen vha saveNumber(number)

### **Udtag klasse**

**Ansvar:** at kontrollere forløbet under UC8

void addRemoveUdtag();

**Parametre:** ingen

**Returværdi:** ingen

**Beskrivelse:** kalder først brugerUI addRemoveMenu som returnere om brugeren ville annullere, tilføje eller fjerne en enhed. Hvis brugeren annullere udskrives mainMenu() og metoden returnere til main.

Hvis brugeren valgte tilføje skal han indtaste navn og adresse. Adressen skal valideres så den kun indholde 4 cifre som alle er enten 1 eller 0. Hvis det går godt skal det indtastede holdes op imod de allerede gemte adresse i hukommelsen. Hvis adressen allerede er brugt bedes bruger om at indtaste ny adresse indtil den er valideret. Når den er valideret sendes navn og adresse til hukommelsen vha saveAdresse(adresse, navn) metoden. Hvis saveAdresse returnere true skal der udskrives "Enhed tilføjet", ellers udskrives "Enhed blev ikke tilføjet".

Hvis brugeren valgte fjerne bedes brugeren om at angive den enhed som skal fjernes. bruger inputtet sendes til removeAdresse(input). Hvis den returnere true udskrives "Enhed fjernet", ellers udskrives "Enhed blev ikke fjernet"

Brugeren bliver efterfølgende præsenteret med et opdateret ui som igen giver ham mulighed for at tilføje, fjerne eller annullere. Dette looper indtil han vælger at annullere.

**ClickATellIF klasse**

**Ansvar:** sende brugeren en sms

void sendSMS(int number);

**Parametre:** telefonnummer

**Returværdi:** ingen

**Beskrivelse:** sender sms til bruger via clickatell API

**RS232IF klasse**

**Ansvar:** at være binde-led imellem pc og STK kittet ifølge protokollen

bool loginValid();

**Parametre:** ingen

**Returværdi:** altid true

**Beskrivelse:** spørger STK kittet om der er logget ind ved at sende 1 kommando ifølge protokollen

void aktiver(string adresse);

**Parametre:** adresse på enhed

**Returværdi:** ingen

**Beskrivelse:** beder om aktivering af enhed på adressen, ifølge protokol

void deaktiver(string adresse);

**Parametre:** adresse på enhed

**Returværdi:** ingen

**Beskrivelse:** beder om deaktiver af enhed på adressen, ifølge protokol

int read();

**Parametre:** ingen

**Returværdi:** int fra 0-3

**Beskrivelse:** tjekker først om der er modtaget 7 eller flere bytes i bufferen. Hvis der ikke er, returneres 0. Hvis der er modtaget 7 eller flere læses 7 bytes. De tjekkes imod 3 parametre. Hvis t / T er det første bogstav den læste så returneres 1. Hvis f / F er det første bogstav den læste så returneres 2. Hvis b / B er det første bogstav den læste så returneres 3.

**BrugerUI klasse**

**Ansvar:** at udskrive text og i nogle tilfælde tage imod bruger input

void mainMenu();

**Parametre:** ingen

**Returværdi:** ingen

**Beskrivelse:** udskriver main menu på skærmen.

void preLogin();

**Parametre:** ingen

**Returværdi:** ingen

**Beskrivelse:** udskriver pre-login menu på skærmen.

void login();

**Parametre:** ingen

**Returværdi:** ingen

**Beskrivelse:** udskriver login menu på skærmen.

int visStatusMenu();

**Parametre:** ingen

**Returværdi:** altid 0

**Beskrivelse:** udskriver status og navne på enhederne. Når brugeren trykker 27 udskrives mainMenu og returneres 0.

int aktiverMenu();

**Parametre:** ingen

**Returværdi:** integer

**Beskrivelse:** udskriver hvilke aktive enheder der er og hvilke der er deaktive, og altså kan aktiveres. Skal tage imod bruger input som skal matche nummeret på enheden. Brugeren kan annullere ved at trykke 27. Hvis der annulleres returneres 27. Ellers returneres tallet på den valgte enhed.

int deaktiverMenu();

**Parametre:** ingen

**Returværdi:** integer

**Beskrivelse:** udskriver hvilke deaktive enheder der er og hvilke der er aktive, og altså kan deaktivieres. Skal tage imod bruger input som skal matche nummeret på enheden. Brugeren kan annullere ved at trykke 27. Hvis der annulleres returneres 27. Ellers returneres tallet på den valgte enhed.

```
int redigerSmsMenu();
```

**Parametre:** ingen

**Returværdi:** integer

**Beskrivelse:** Henter nr fra hukommelsen vha. getNumber(). udskriver nummeret at brugeren kan taste 1 for at ændre nr eller 27 for at annullere. Der tages imod bruger input. Hvis brugeren vælger at annullere returneres 0. Hvis brugerne vælger 1. så bedes han om at indtaste et 8 cifret tlf nr. Inputtet bliver valideret ved at det skal være mindre end 9999999 og større end 10000000. Hvis det går godt returneres nummeret.

```
int addRemoveMenu();
```

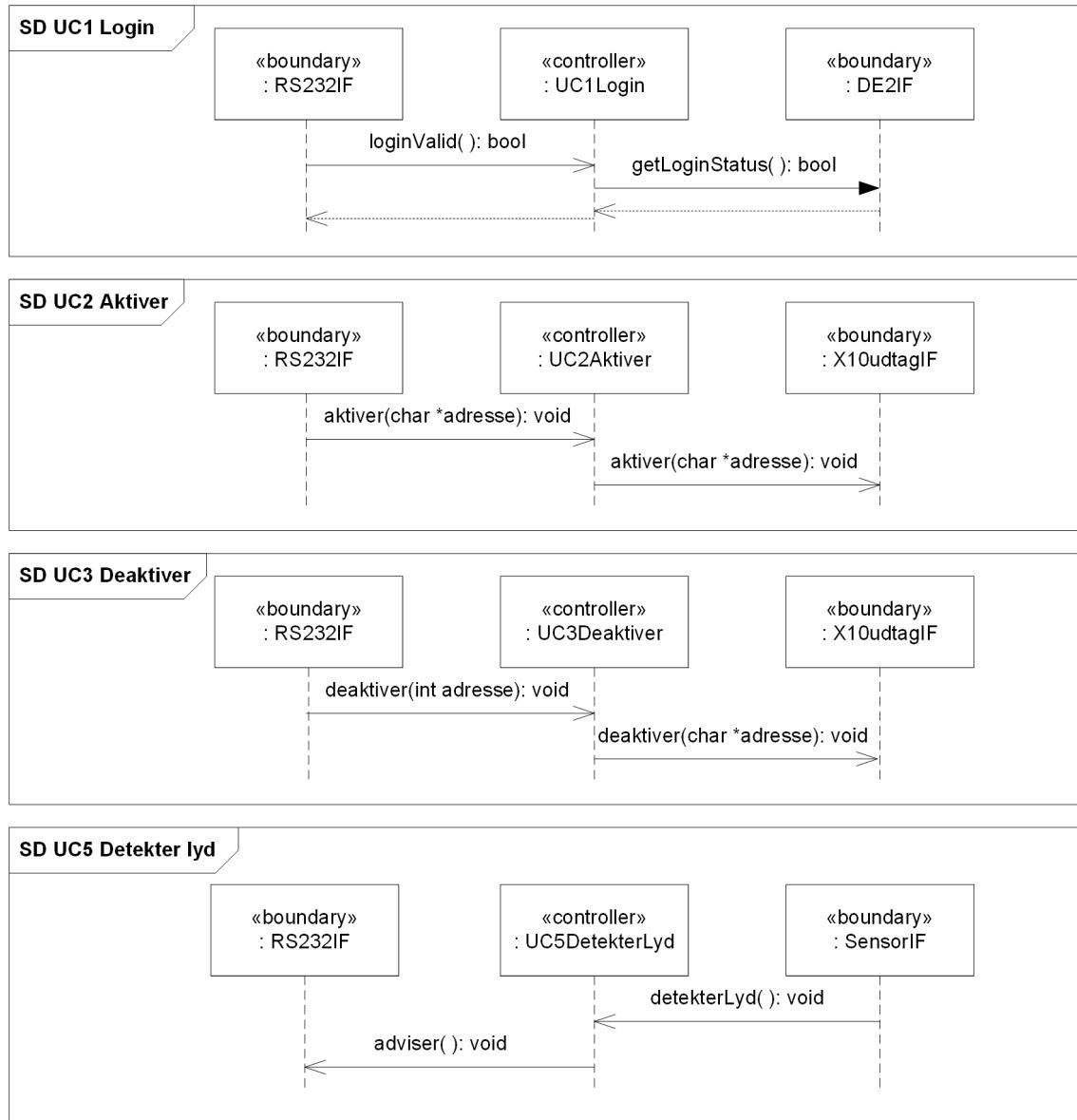
**Parametre:** ingen

**Returværdi:** integer

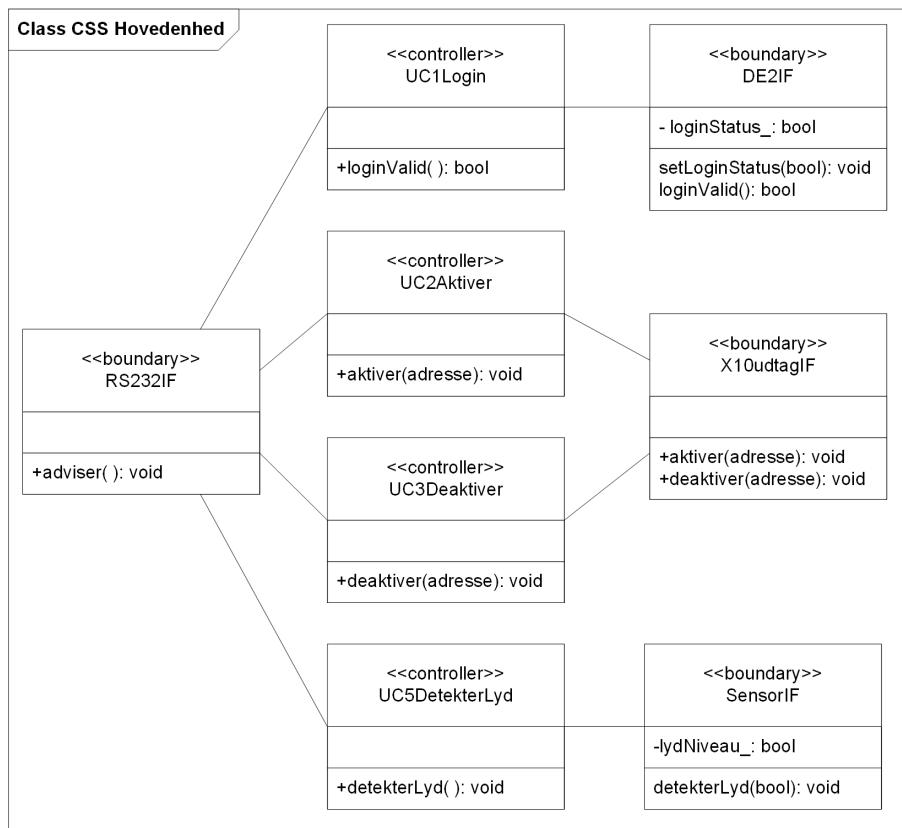
**Beskrivelse:** Alle enheder udskrives på skærmen med navn og adresser. Hvis der er 15 enheder bedes brugeren om at fjerne nogle før han kan tilføje nogle nye. Han kan også vælge at annullere ved at trykke 27. Trykkes der 27 returneres 27. Hvis der vælges at fjerne enheder returneres 1. Hvis brugeren vælger at tilføje returneres 2.

### 4.3.3 Applikations model for CSS hovedenhed

For CSS hovedenheden er udviklet en række diagrammer ud fra applikationsmodel metoden. Der er et sekvensdiagram på figur 4.11 for alle de aktuelle use-cases som beskriver systemets virkemåde. Ud fra dette er der lavet et klassediagram på figur 4.12 som dækker de forskellige use-cases med controller klasser og kommunikationen til PC via RS232 og X10 udtag via X10.



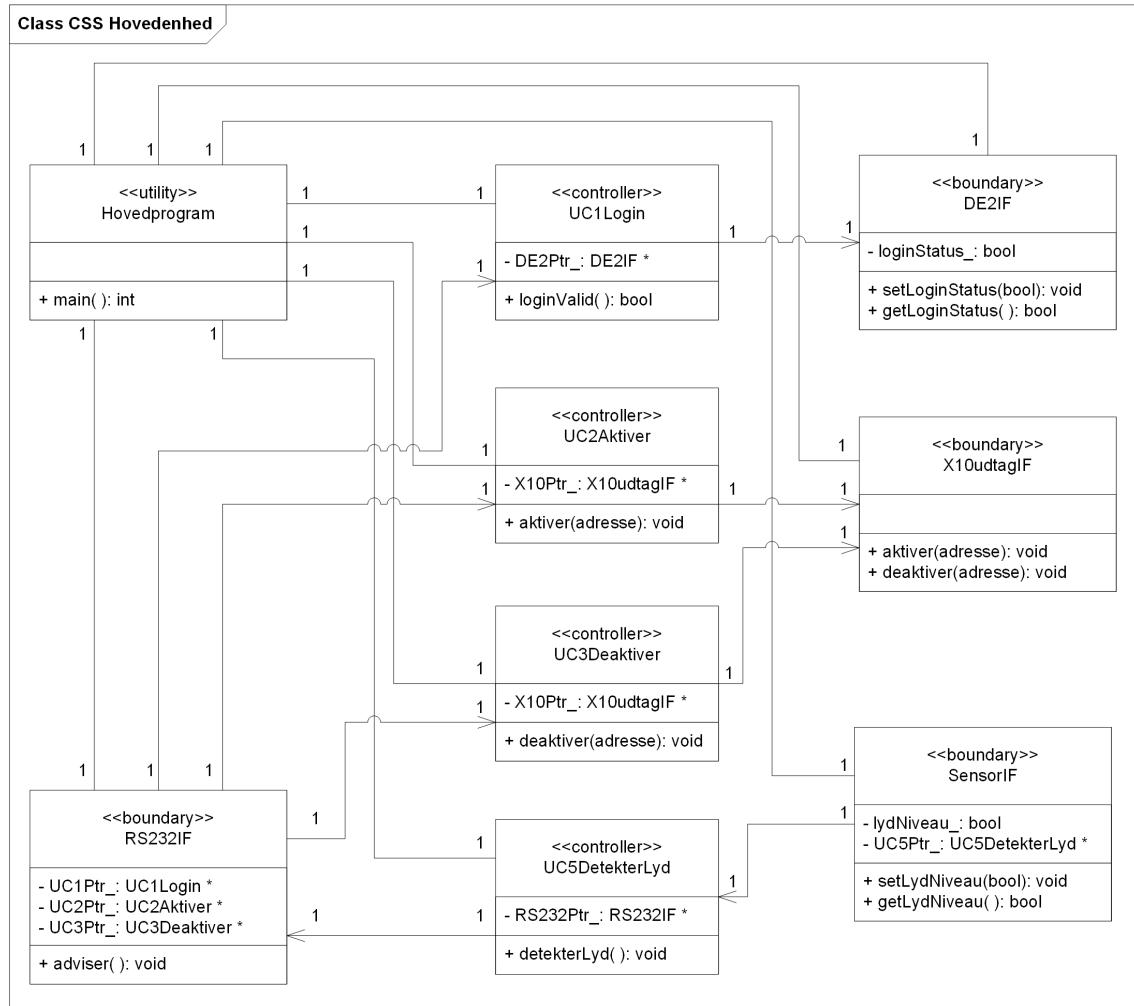
Figur 4.11. Use-case sekvensdiagrammer for CSS hovedenhed



**Figur 4.12.** Klassediagram for CSS hovedenhed

#### 4.3.4 Klassediagram og beskrivelse for CSS hovednehed

Ud fra applikationsmodellens klasse diagram (Figur 4.12) er udledt et statisk klasse diagram, se figur 4.13.



**Figur 4.13.** Statisk klasse diagram for CSS hovedenhed

Her følger klassebeskrivelser for alle klasser til CSS hovedenheden.

#### RS232IF

**Ansvar:** At varetage kommunikation mellem CSS hovedenhed og PC over RS232 protokollen.

**Attributer:**

- **UC1Login \* UC1Ptr\_**  
Pointer til associeret UC1 objekt
- **UC2Aktiver \* UC2Ptr\_**  
Pointer til associeret UC2 objekt
- **UC3Deaktiver \* UC3Ptr\_**  
Pointer til associeret UC3 objekt

**Metoder:**

void adviser();

**Parametre:** Ingen

**Returværdi:** Ingen

**Beskrivelse:** Sender kommando "SB<cr>" over RS232

### **UC1Login**

**Ansvar:** At varetage UC1 Login forløbet.

**Attributer:**

- **DE2IF \* DE2Ptr**

Pointer til associeret DE2 objekt

**Metoder:**

bool loginValid();

**Parametre:** Ingen

**Returværdi:** Ingen

**Beskrivelse:** Kalder getLoginStatus() metoden i DE2IF og returnerer værdien her fra

### **UC2Aktiver**

**Ansvar:** At varetage UC2 Aktiver forløbet.

**Attributer:**

- **X10udtagIF \* X10Ptr**

Pointer til associeret X10udtag objekt

**Metoder:**

void aktiver(int adresse);

**Parametre:** Adresse på enhed

**Returværdi:** Ingen

**Beskrivelse:** Kalder aktiver() metoden i X10udtagIF med den modtagede adresse

### **UC3Deaktiver**

**Ansvar:** At varetage UC3 Deaktiver forløbet.

**Attributer:**

- **X10udtagIF \* X10Ptr**

Pointer til associeret X10udtag objekt

**Metoder:**

void deaktiver(int adresse);

**Parametre:** Adresse på enhed

**Returværdi:** Ingen

**Beskrivelse:** Kalder deaktiver() metoden i X10udtagIF med den modtagede adresse

### UC5DetekterLyd

**Ansvar:** At varetage UC5 Detekter Lyd.

**Attributer:**

- **RS232IF \* RS232Ptr \_**

Pointer til associeret X10udtag objekt

**Metoder:**

void detekterLyd();

**Parametre:** Ingen

**Returværdi:** Ingen

**Beskrivelse:** Kalder adviser() metoden i RS232IF

### DE2IF

**Ansvar:** At holde styr på aktuel loginstatus på DE2 boardet.

**Attributer:**

- **bool loginStatus \_**

1: Login bekræftet på DE2 board

0: Login ikke bekræftet på DE2 board

**Metoder:**

void setLoginStatus(bool status);

**Parametre:** status: 1 hvis bekræftet og 0 hvis ikke bekræftet på DE2 board

**Returværdi:** Ingen

**Beskrivelse:** Sætter attribut loginStatus\_ til aktuel status på DE2 board

bool getLoginStatus();

**Parametre:** Ingen

**Returværdi:** status: 1 hvis bekræftet og 0 hvis ikke bekræftet på DE2 board

**Beskrivelse:** Returnerer aktuel login status

### X10udtagIF

**Ansvar:** At varetage kommunikation mellem CSS hovedenhed og X10 modtager over X10 protokollen.

**Attributer:** Ingen

**Metoder:**

void aktiver(int adresse);

**Parametre:** adresse: Adresse på X10 enhed som ønskes aktiveret

**Returværdi:** Ingen

**Beskrivelse:** Konverterer adressen bitvis til ASCII karakterer (0010 -> "0010") og sender kommandoen "SAXXXX<cr>" (hvor XXXX er adressen over) over X10

void deaktiver(int adresse);

**Parametre:** adresse: Adresse på X10 enhed som ønskes deaktiveret

**Returværdi:** Ingen

**Beskrivelse:** Konverterer adressen bitvis til ASCII karakrterer (0010 -> "0010") og sender kommandoen "SDXXXX<cr>"(hvor XXXX er adressen over) X10

### SensorIF

**Ansvar:** At holde styr på aktuel lyd detektering og give besked hvis lyd registreres.

**Attributer:**

- **bool loginStatus\_**  
1: Lyd detekteret  
0: Ingen lyd detekteret
- **UC5DetekterLyd \* UC5Ptr\_**  
Pointer til associeret UC5 objekt

**Metoder:**

void setLydNiveau(bool niveau);

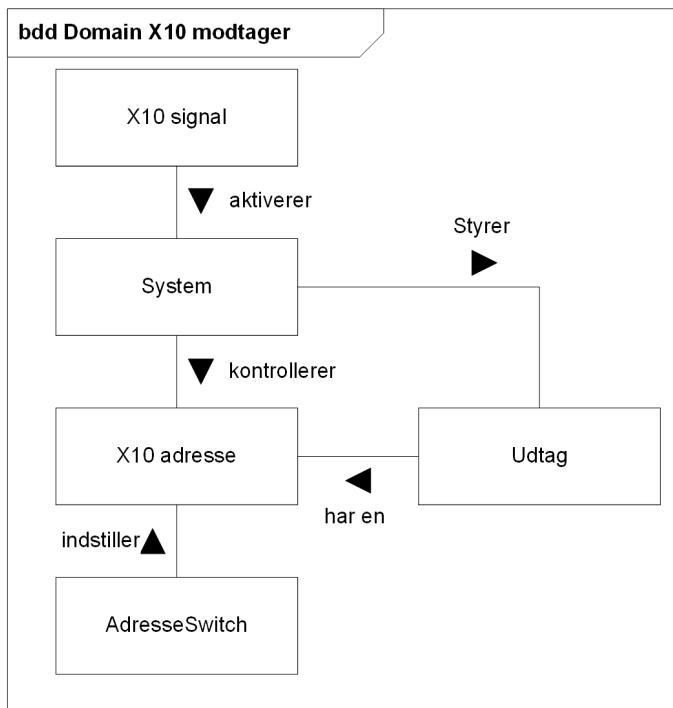
**Parametre:** niveau: 1 hvis lyd detekteret ellers 0

**Returværdi:** Ingen

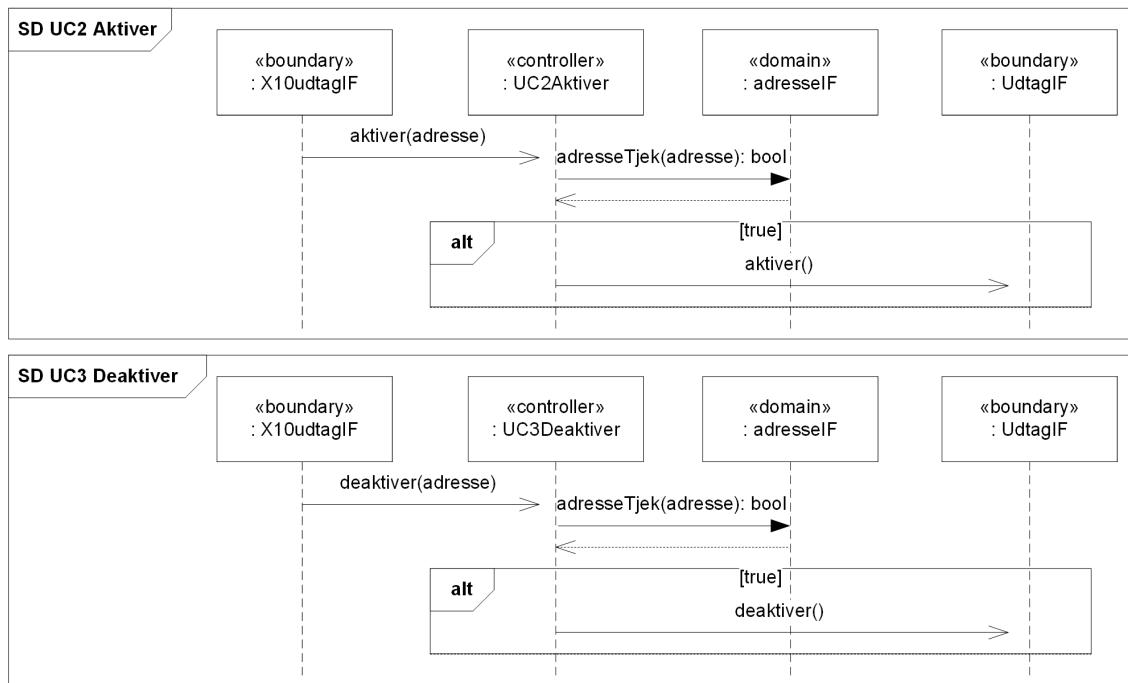
**Beskrivelse:** Kalder detekterLyd() metoden, i UC5DetekterLyd, med parameter 1, hvis niveau er 1, ellers intet.

### 4.3.5 Applications model for X10 modtager

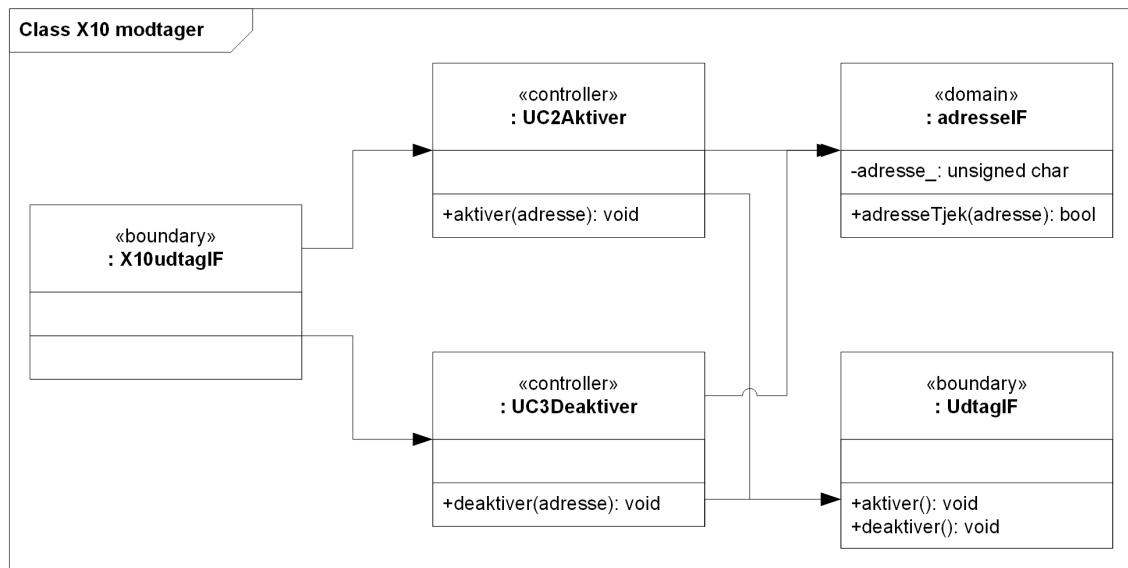
For X10 modtageren er der udviklet en række diagrammer ud fra applikationsmodel metoden. På figur 4.14 er domænemodellen som resten af diagrammerne er udviklet ud fra. Der er et sekvensdiagram på figur 4.15 for alle de aktuelle use-cases som beskriver systemets virkemåde. Ud fra dette er der lavet et klassediagram på figur 4.16 som dækker de forskellige use-cases med controller klasser og kommunikationen til CSS hovedenheden via X10.



*Figur 4.14.* Domænemodel for X10 modtager



Figur 4.15. Use-case sekvensdiagrammer for X10 modtager



Figur 4.16. Klassediagram for X10 modtager

#### 4.3.6 Klassebeskrivels for X10 modtager

##### Hukommelse klasse

void saveLogin(bool);

**Parametre:** ingen

**Returværdi:** ingen

**Beskrivelse:** gemmer login status og bevare denne i 10 minutter

void saveStatus(bool, int adresse);

**Parametre:** bool til bestemmelse af om status er aktiv eller deaktiv. Int adresse til bestemmelse af status på adressen

**Returværdi:** ingen

**Beskrivelse:** gemmer status på enheden på pågældende adresse

void getEnheder();

**Parametre:** ingen

**Returværdi:** ingen

**Beskrivelse:** gemmer login status og bevare denne i 10 minutter

int getNumber();

**Parametre:** ingen

**Returværdi:** gemte telefonnummer

**Beskrivelse:** returnere det gemte telefonnummer

void saveNumber(int number);

**Parametre:** number der skal gemmes

**Returværdi:** ingen

**Beskrivelse:** gemmer telefonnummeret

##### Login klasse

void loginValid();

**Parametre:** ingen

**Returværdi:** ingen

**Beskrivelse:** N/A

##### Aktiver klasse

void aktiverEnhed(int adresse);

**Parametre:** adresse på enhed

**Returværdi:** ingen

**Beskrivelse:** N/A

#### Deaktiver klasse

void deaktiverEnhed(int adresse);

**Parametre:** adresse på enhed

**Returværdi:** ingen

**Beskrivelse:** N/A

#### DetekterLyd klasse

void lydDetekteret();

**Parametre:** ingen

**Returværdi:** ingen

**Beskrivelse:** henter telefonnummer i hukommelse og sender det til ClickATell klassen

#### RedigerSmSBruger klasse

void redigerSMS(int number);

**Parametre:** nye nummer

**Returværdi:** ingen

**Beskrivelse:** gemmer nye nummer i hukommelsen

#### Udtag klasse

bool addUdtag(int adresse, string name);

**Parametre:** adresse og navn på udtag

**Returværdi:** true hvis operation gik godt, false hvis ikke

**Beskrivelse:** tilføjer udtag til hukommelse ved at gemme navn og adresse

#### ClickATellIF klasse

void sendSMS(int number);

**Parametre:** telefonnummer

**Returværdi:** ingen

**Beskrivelse:** sender sms til bruger via clickatell

#### RS232IF klasse

bool loginValid();

**Parametre:** ingen

**Returværdi:** true eller false

**Beskrivelse:** afventer login fra DE2 board

```
void aktiver(int adresse);
```

**Parametre:** adresse på enhed

**Returværdi:** ingen

**Beskrivelse:** beder om aktivering af enhed på adressen, ifølge protokol

```
void deaktiver(int adresse);
```

**Parametre:** adresse på enhed

**Returværdi:** ingen

**Beskrivelse:** beder om deaktiver af enhed på adressen, ifølge protokol

### BrugerUI klasse

```
void showMenu();
```

**Parametre:** ingen

**Returværdi:** ingen

**Beskrivelse:** skal styre hele brugerUI menuen.

```
bool printStatus();
```

**Parametre:** ingen

**Returværdi:** bool godkendt

**Beskrivelse:** hente status og navne fra hukommelse og udskrive dem på skærmen

## 4.4 Protokol

### 4.4.1 Seriel kommunikation

Kommunikationen mellem PC og CSS hovedenheden sker over seriel kommunikation på et RS232 interface.

Det fysiske setup for RS232-interfacet er: 9600 kbps, ingen paritet, 8 bits, 1 stop bit.

I tabel 4.3 beskrives de fælles informationer som gælder mellem computeren og CSS hovedenheden.

**Tabel 4.3.** Start og stop bytes for RS232 kommunikation

	ASCII	Hex
<b>STX</b>	'S' / 's'	0x53 / 0x73
<b>ETX</b>	'\r'	0x0D

Dataen formateres som vist i tabel 4.4. Alle frames er 7 byte.

**Tabel 4.4.** Data formatering for RS232 kommunikation

Byte	0	1	2..5	6
Indhold	STX	<Kommando>	<Data>	ETX

#### Blokken <Kommando>

Kun kommandoerne beskrevet i tabel 4.5 er gyldige. I tilfælde af at kommandoen ikke genkendes er der intet svar. Bemærk at det er muligt at bruge både store og små karakterer.

**Tabel 4.5.** Kommandoer for RS232 kommunikation

ASCII	HEX	Funktion
'A' / 'a'	0x41 / 0x61	Aktiver enhed
'D' / 'd'	0x44 / 0x64	Deaktiver enhed
'L' / 'l'	0x4C / 0x6C	Hent login status
'T' / 't'	0x54 / 0x74	Login korrekt
'F' / 'f'	0x46 / 0x66	Login forkert
'B' / 'b'	0x42 / 0x62	Lyd detekteret

**Blokken <Data>** Ved alle kommandoer undtaget Aktiver- og Deaktiverkommandoerne inderholder <Data> "9999"

For at bruge aktiver eller deaktiver kommandoerne er <Data> formateret som adressen. Denne adressering formateres som 4 byte, som hver består af ASCII karakterende '0' eller '1'. På den måde skriver man blot den adresse ind, som man har indstillet på sit X10 uddag. F.eks. "0100".

#### Eksempler:

"**SA0101\r**" Kommandoen aktiverer enheden med adresse "0101".

**"SL\r"** Kommandoen beder CSS hovedenheden om at returnerer login status.

CSS Hovedenheden vil returnerer et svar: **"ST9999\r"** for at brugeren er logget ind eller **"SF9999\r"** hvis brugeren ikke er logget ind. Bemærk at \r er ASCII karakteren for carriage return.

#### 4.4.2 X10 kommunikation

Kommunikationen mellem CSS Hovedenhed og X10 Udtagene sker over strømnettet via et X10 interface.

Den officielle X10 protokol bruges som udgangspunkt for denne arbitrerer X10 protokol. Afvigelserne fra den officielle X10 protokol ligger i hvilke gyldige kommandoer der er til rådighed. Kun kommandoerne i tabel 4.8 er gyldige. Se også tabel 4.6 for hvordan en data frame er bygget op. Bemærk at enheds adressen og kommando sendes i én pakke.

**Tabel 4.6.** Data formatering for X10 kommunikation

STX	<Kommando>	<Adresse>	ETX
-----	------------	-----------	-----

I det følgende differentieres der mellem almindelige binære mønstre og X10 formaterede bit mønstre. Den officielle X10 protokol beskriver at det binære 0 sendes som 01 og binært 1 sendes som 10. Se **INDSÆT REFFERENCE TIL X10 PROTOKOLLEN** for yderligere detaljer.

I tabel 4.7 beskrives de fælles informationer som gælder mellem CSS hovedenheden.

**Tabel 4.7.** Start og stop bytes for X10 kommunikation

	X10 kode
STX	1110
ETX	000000

#### Blokken <Kommando>

Alle kommandoer sendes to gange med tre 50 Hz perioder i mellem hver. Dette håndteres med ETX koden. I tilfælde af at kommandoen ikke genkendes er der intet svar.

**Tabel 4.8.** Kommandoer for X10 kommunikation

Binær	X10 kode	Funktion
00101	0101100110	Aktiver enhed
00111	0101101010	Deaktiver enhed

#### Blokken <Adresse>

Adresserne modtages fra PCen binært. Denne kode omsættes til X10 formatet og afsendes uden videre formatering. I tabel 4.9 vises nogle eksempler på adresser.

#### Eksempler

I tabel 4.10 er vist to eksempler som aktiverer og deaktiverer et X10 udtag. Mellemrummende i X10 koden er indsat for at kunne se blokkende og vil ikke eksisterer i praksis.

**Tabel 4.9.** Adresser formateret i X10 format

Binær	X10 kode
0001	01010110
0101	01100110
0111	01101010

**Tabel 4.10.** Adresser formateret i X10 format

Kommando	X10 kode
Tænd X10 udtag på adresse 0101	1110 0101100110 01100110 000000
Sluk X10 udtag på adresse 0011	1110 0101101010 01011010 000000

"**S0101A\r**" Kommandoen aktiverer X10 udtaget med adresse "0101".

"**s0101d\r**" Kommandoen deaktiverer X10 udtaget med adresse "0101".

Bemærk at \r er ASCII karakteren for carriage return.

# **Hardware design**

**5**

---

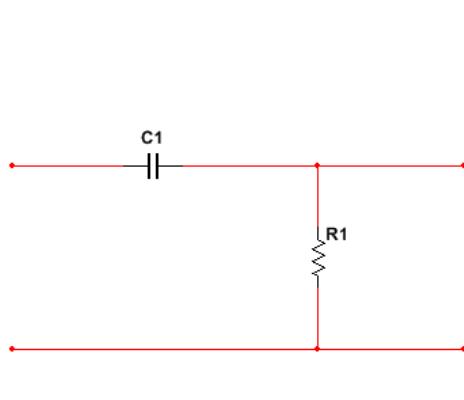
TODO::Beskrivelse af hardware design overordnet

## **5.1 Encoder**

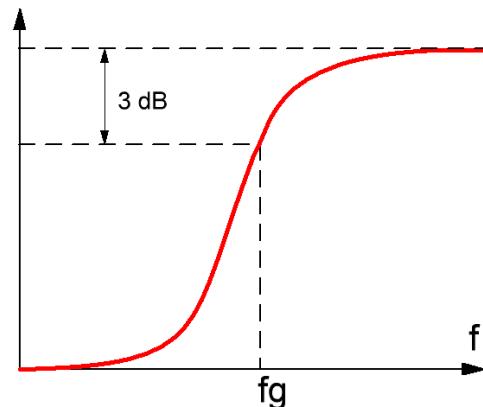
Encoderen, også omtalt som senderenhed, er den del af systemet der genererer X10 kommandoen og sender den ud på det eksisterende el-net. Et højpasfilter der lader 120 kHz burst passere mens det blokerer for nettets 50 Hz signal, udgør sammen med en zero crossing detector hele hardwaren for encoderen.

TODO:: billede af burst ud på nettet!!

### 5.1.1 Højpasfilter



**Figur 5.1.** Højpasfilter uden værdier



**Figur 5.2.** Kurvekarakteristik for højpas filter

For at sende X10 kommandoer ud på det eksisterende 50 Hz el-net er det nødvendigt at koble elektroniken direkte herpå og eftersom det elektronik ikke tåler de høje spændinger fra nettet, er det nødvendigt at blokere det signal, men stadig at kunne sende de 120 kHz ud. Dette løses med et højpasfilter.

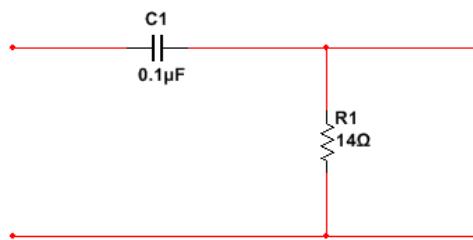
Den ønskede knækfrekvens skal ligge omkring de 120 kHz for at opnå mindst dæmpning herpå. Ved denne knækfrekvens skulle 50 Hz signalet være ubetydelig lille efter filteret.

Kondensatoren forudbestemmes for beregningerne til en værdi på 0,1 nF.

$$R_1 = \frac{1}{2 \cdot \pi \cdot f_c \cdot C} = \frac{1}{2 \cdot \pi \cdot 120000 \cdot 0,1 \cdot 10^{-6}} = 13,26\Omega \rightarrow R_1 > 13,26\Omega \quad (5.1)$$

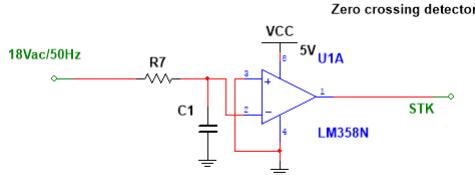
$R_1$  vælges da til  $14\Omega$

Med den beregnede modstands værdi er det endelige schematic færdigt som det ses på figur 5.3

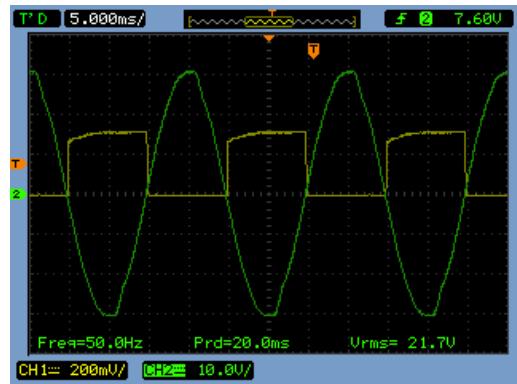


**Figur 5.3.** Højpasfilter med værdier

### 5.1.2 Zero Crossing Detector



**Figur 5.4.** Zero crossing detector uden værdier



**Figur 5.5.** Scope billede af Zero crossing detector, CH1(indgangssignal), CH2(Udgangssignal)

Zero crossing detectoren har til opgave at detektere nulgennemgang, det er et krav for at X10 protokollen kan virke. Der er placeret en Zero crossing detector både på Encoderen og Decoderen, da begge disse består af et STK-kit der kræver information om nulgennemgang. Opbygning kan ses på overstående figur 5.4, vi har anvendt en operationsforstærker af type LM358N, som toggler udgangssignalet ved hver nulgennemgang se figur 5.5. Operationsforstærkerens positive ben er koblet til stel for at lave et triggerniveau til 0 V.

Modstanden  $R_7$  sidder der bl.a. for at beskytte Zero crossing detectoren mod 18 VAC nettet, men sammen med kondensatoren udgør den også et lavpasfilter. Under implementeringen opdagede vi at der kom støj ind på Zero crossing detectoren, og dette problem løste lavpasfilteret.

Da vi ønsker at dæmpe 120 kHz signalet, designes lavpasfilteret ud fra en knækfrekvens på 1,0 kHz.

$$R = \frac{1}{2 \cdot \pi \cdot f_c \cdot C} \quad (5.2)$$

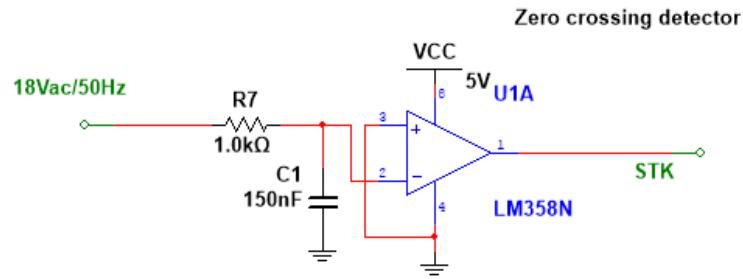
Der vælges en kondensator med en værdi på 150 nF. Med denne værdig kan modstanden i lavpasfilteret beregnes.

Lavpasfilterets modstand:

$$R_7 = \frac{1}{2 \cdot \pi \cdot 1000 \cdot 150 \cdot 10^{-9}} = 1061\Omega \quad (5.3)$$

$R_7$  vælges da til 1,0 kΩ

Med alle de beregnede komponentværdier kan det endelige schematic designes. Det endelige design er vist på figur 5.6



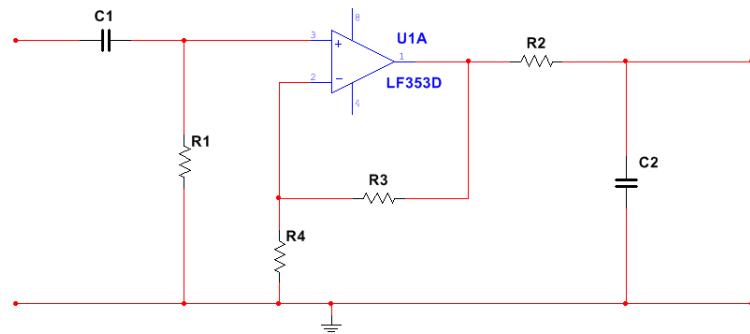
**Figur 5.6.** Zero crossing med værdier

## 5.2 Decoder

Decoderen som er den del i systemet der omdanner burst fra encoderen til X10 kommando er bygget op af et båndpasfilter, zero crossing detector og en envelope detector. I starten af kredsløbet sidder båndpasfilteret. Dette blokerer for 50 Hz nettet og forstærker 120 kHz signalet der kommer fra encoderen. Herefter ledes signalet gennem envelope detector som omdanner burstet til et TTL signal.

TODO: Billede af burst ind og TTL signal efter envelope..

### 5.2.1 Båndpasfilter



**Figur 5.7.** Båndpas uden værdier

Båndpasfilteret har til opgave at filtrere alle signaler med frekvenser over og under 120 kHz fra samtidig med at det forstærker signalet i båndpasset. Forstærkningen opnås ved at koble en ikke inverterende OpAmp med modstande, hvis størrelse afhænger af den ønskede forstærkning, mellem et højpasfilter og lavpasfilter som illustreret på figur 5.7.

Da vi ønsker at forstærke 120 kHz signalet, designes højpasfilteret således at knækfrekvensen udregnes til 110 kHz, og for lavpasfilteret beregnes en knækfrekvens på 130 kHz.

$$R = \frac{1}{2 \cdot \pi \cdot f_c \cdot C} \quad (5.4)$$

Fælles for begge filtre vælges en kondensator med en værdi på 1,0 nF. Med denne værdig kan de to modstande i henholdsvis lavpas- og højpasfilteret beregnes.

Højpasfilterets modstand:

$$R_1 = \frac{1}{2 \cdot \pi \cdot 110000 \cdot 1,0 \cdot 10^{-9}} = 1447\Omega \rightarrow R_1 > 1447\Omega \quad (5.5)$$

$R_1$  vælges da til 1,6 kΩ

Lavpasfilterets modstand:

$$R_2 = \frac{1}{2 \cdot \pi \cdot 130000 \cdot 1,0 \cdot 10^{-9}} = 1225\Omega \rightarrow R_2 < 1224\Omega \quad (5.6)$$

$R_2$  vælges da til 1,1 kΩ

Da det ikke kan undgås at 120 kHz signalet bliver dæmpet både gennem højpasfilteret og lavpasfilteret er det nødvendigt at forstærke signalet ved hjælp af en LF353 OpAmp. Ligningen for udregning af udgangsspændingen er som følgende:

$$V_{out} = \left(1 + \frac{R_3}{R_4}\right) \cdot V_{in} \quad (5.7)$$

Ved at vælge  $R_4$  til 10 kΩ kan forstærkningen nemt reguleres med  $R_3$ . Ved målinger på indgangssignalet før operationsforstærkeren kan vi fastslå indgangsamplituden til at være 8,0 V. Da udgangssignalet ledes gennem en diode der fjerner de negative halvperioder vil

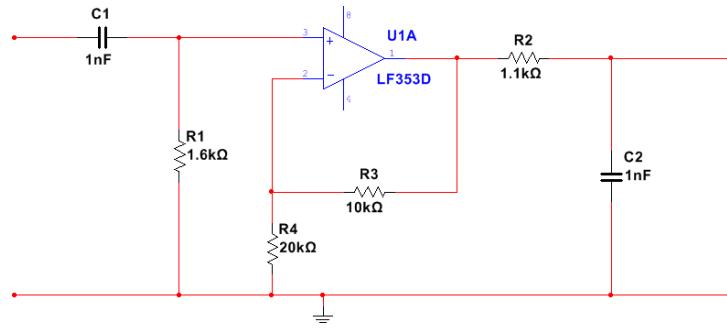
amplituden blive halveret gennem denne. Samtidig er der dæmpning i lavpasfilteret og der vil derfor være brug for en mere end en fordobling i forstærkeren.

$$R_4 = (\beta - 1) \cdot R_3 = (3 - 1) \cdot 10000 = 20000\Omega \quad (5.8)$$

$R_4$  er valgt til  $20000\Omega$  så signalet derved bliver forstærket 3 gange.

Der er som nævnt valgt en LF353 OpAmp. Denne er valgt ud fra at den har en bred båndbredte, en hurtig sætte tid og lav støj på indgangene.

Med alle de beregnede komponentværdier kan det endelige schematic designes. Det endelige design er vist på figur 5.8



**Figur 5.8.** Båndpas med værdier

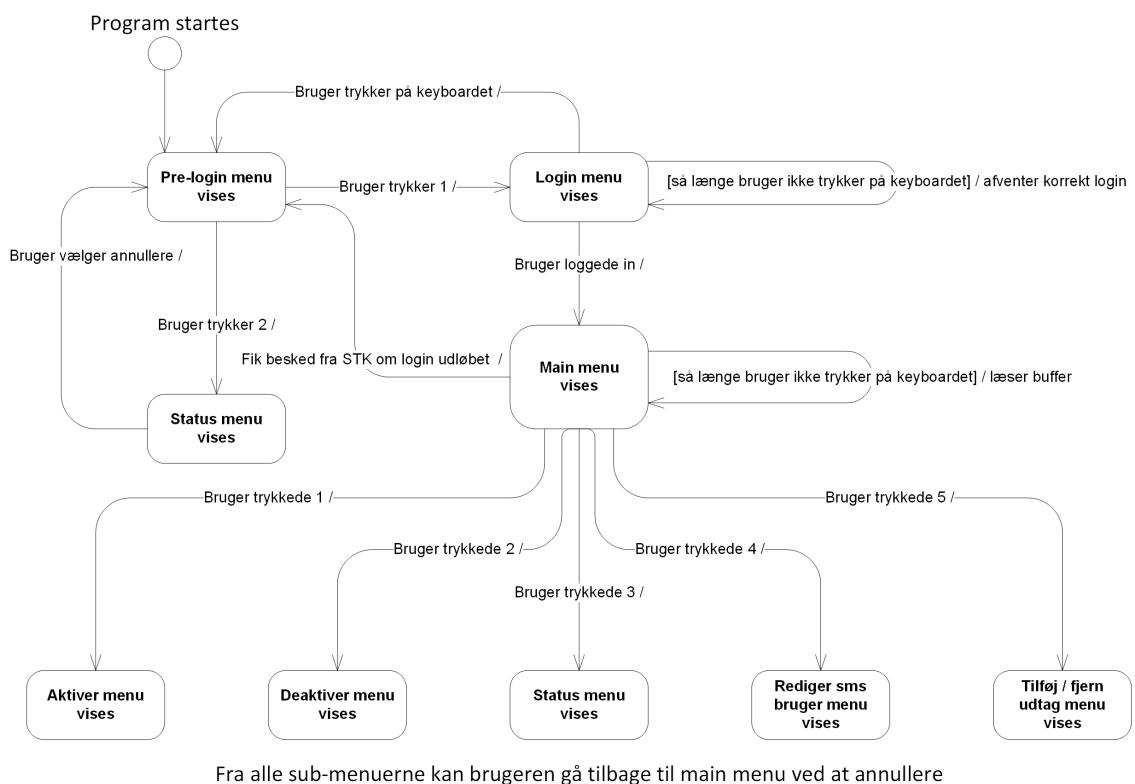
### 5.2.2 Envelope Detector



# Software design 6

---

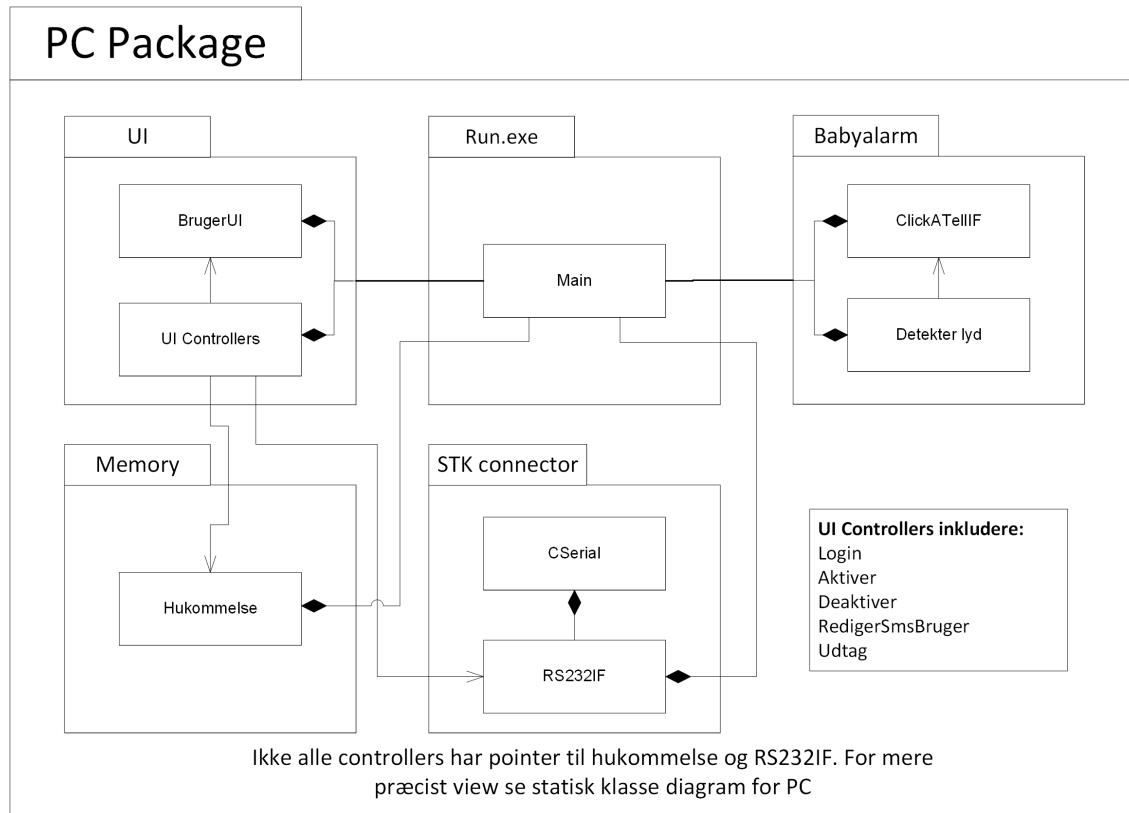
## 6.1 Logical View



*Figur 6.1.* State machine diagram over forløbet fra PC start til menuer.

Diagrammet ovenfor skal illustrere hvordan forløbet er fra PC opstart. Hvor man møder Pre-login menuen som kun giver en mulighed for at få vist login menu eller vis status menu. Efter der er logget ind vil den stå og føle på input bufferen, på den port hvor PC'en er forbundet med CSS hovedenheden. Det gør den for at en evt. babyalarm kan afbryde forløbet og blive sendt. Desuden kan CSS hovedenheden give besked om at der ikke længere er logget ind, hvilket sender brugeren til pre-login menuen igen.

Når brugeren så trykker på en tast og trykker enter vil han blive sendt til en af de 5 menuer. Dog stadig under forudsætning af at han valgte en værdi imellem 1-5 for den pågældende menu. Ved forkert tast sker der ingenting. Fra hver af de 5 sub-menuer har bruger mulighed for at annullere og komme tilbage til main menuen. Dette gør han ved at taste 27 og enter.



**Figur 6.2.** Logical view: PC package

Figuren ovenfor viser de forskellige packages man kan indlede PC klasserne i. Alle controllers som har UI pointers er smidt i UI pakken sammen med BrugerUI. STK Connector pakken indeholder RS232IF og så har den et CSerial objekt som er en klasse der har de 5 mest basale metoder til at sende og læse på en seriel port.

### 6.1.1 Klasse CircBuffer

Efter som X10 kommunikationen er meget langsom (50 bits/s) bruges en buffer til at holde på kommandoerne ind til de er klar til at blive afsendt. Dette er CircBuffer klassens opgave. Denne er udformet som en circulær buffer som kan holde 2 fulde kommandoer. Bemærk at i følge X10 protokollen skal alle kommandoer afsendes to gange. Så der er plads til 4 kommandoer, hvor to af dem altid er de samme.

Klassen er bygget til selv at holde styr på hvilket bit der skal sendes næste gang. Dette forenkler brugen væsentligt, da udtagning af data fra bufferen sker fra en interrupt service rutine (ISR). Denne rutine er beskrevet i detaljer senere.

Alle kommandoer termineres med '\0' karakteren.

### 6.1.2 Klasse X10IF

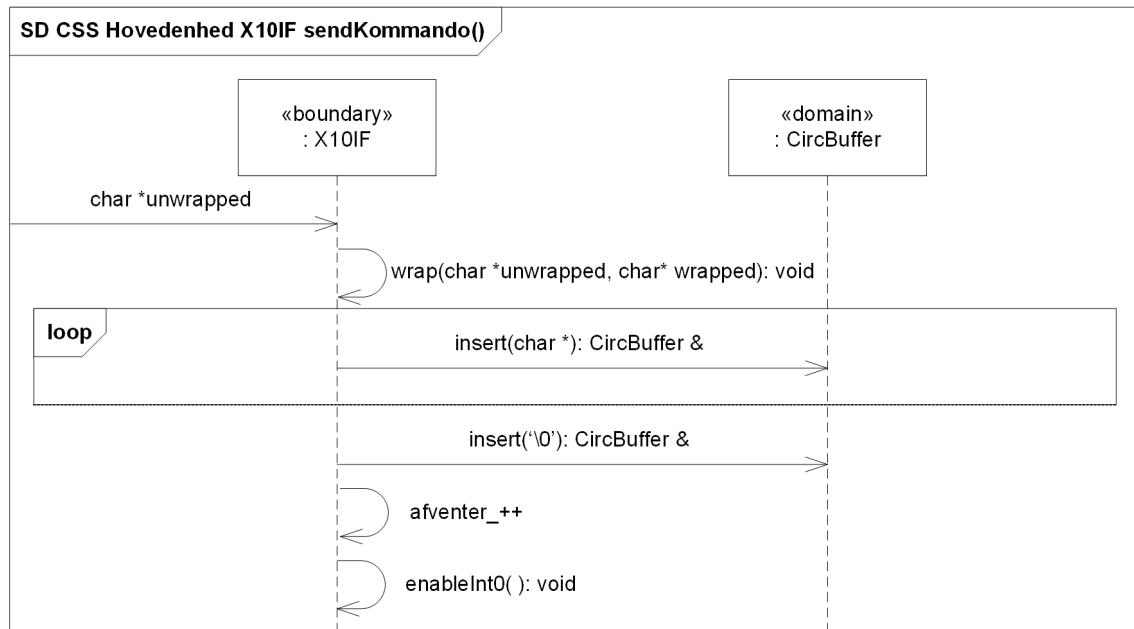
En af de kritiske og avancerede klasser er X10IF klasserne på hhv. CSS Hovedenheden og X10 Udtaget. I designfasen er der udviklet sekvensdiagrammer som beskriver nogle af metoderne og sammenspillet til andre klasser.

Funktionaliteten for metoden sendKommando() i X10IF klassen, på CSS hovedenheden, har som ansvar at afsende en fuld X10 kommando, iht. protokollen, ud fra en parameter formateret som vist i tabel 6.1.

**Tabel 6.1.** Parameter opbygning til sendKommando() metoden i X10IF

Byte	0	1..4
Data	Kommando	Adresse

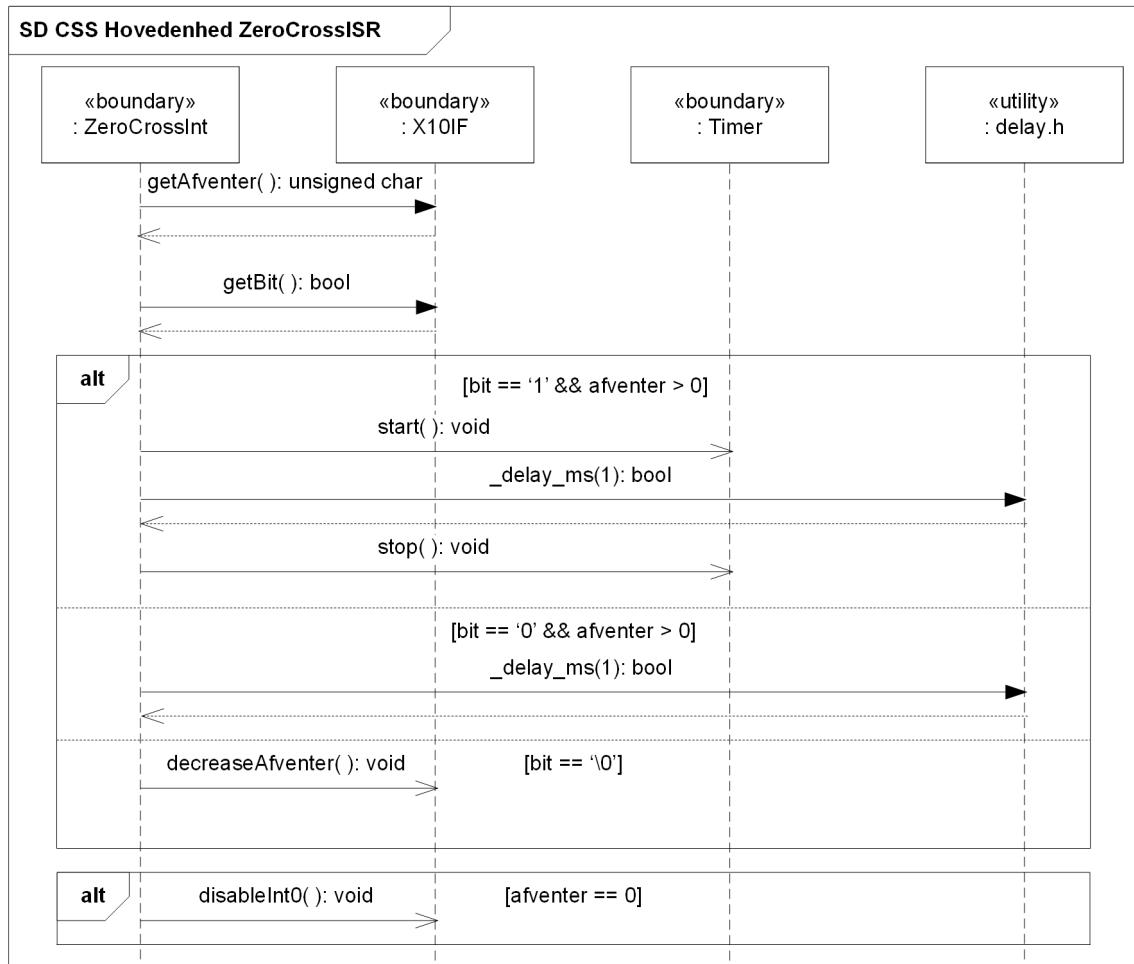
Metoden skal først omskrive den modtagende kommando til en X10 bit streng. Her efter indsættes alle bitsende i en buffer, hvorfra de afsendes når der detekteres et zerocross. Sekvensen er vist i figur 6.3.



**Figur 6.3.** Sekvensdiagram for metoden sendKommando() i X10IF klassen på CSS hovedenheden

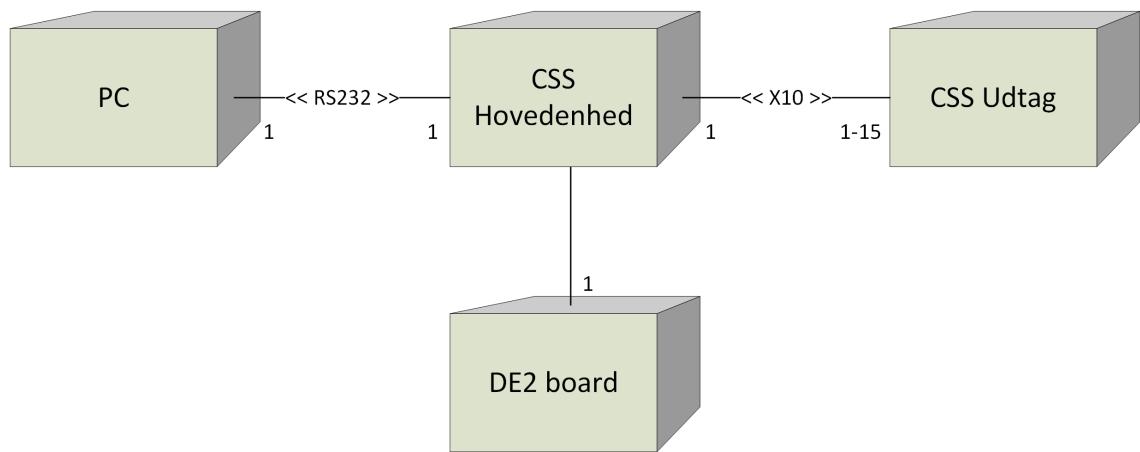
### 6.1.3 ZeroCrossInt funktion

I tilfælde af et interrupt signal på INT0 benet på CSS hovedenheden køres en bestemt funktion i microcontrolleren. Dette er kaldet en ISR. Forløbet for denne er beskrevet i sekvensdiagrammet på figur 6.4. Først kontrollerer den om der er nogle kommandoer i kø. Der efter henter den det næste bit der skal afsendes fra bufferen. Ud fra værdien bestemmer den om der skal tændes for 120 kHz frekvensen i timeren. Når en kommando er helt sendt nedskriver den køen og hvis køen er tom slår den interruptet fra.



**Figur 6.4.** Sekvensdiagram for INT0 ISR på CSS hovedenheden

## 6.2 Deployment View



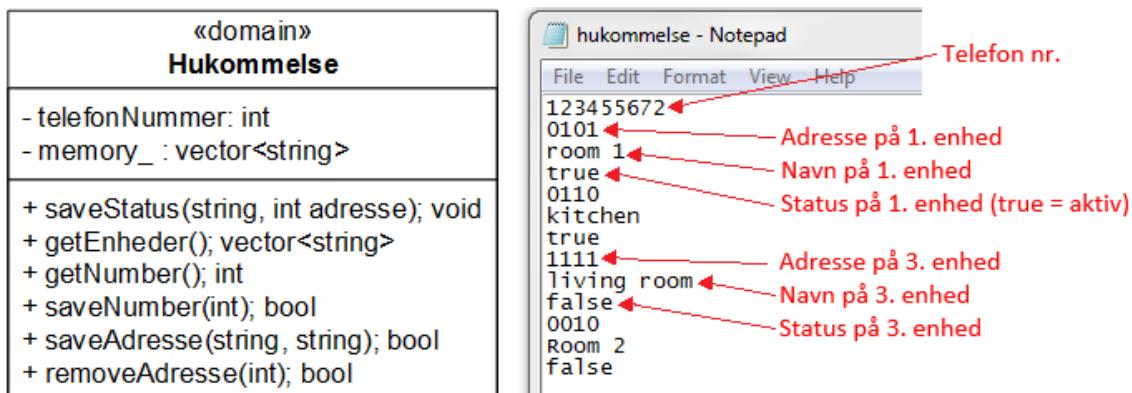
**Figur 6.5.** Deployment View

### 6.3 Data view

Vi valgte at vores system skal kunne klare at blive slukket og tændt uden at det ville miste alle data omkring vores udtags tændt/slukket status, adresser, navne og brugeren telefonnummer. Derfor besluttede vi at vi skulle have en klasse på PCen som kunne gemme og loade disse dataer. Det står klassen Hukommelse for.

Klassen hukommelse læser og redigere i en text fil hver gang der sker ændringer. Når den læser text filen skriver den hver linje ind på en plads i en vector som kan indeholde strings. Linje 1 i text filen kommer altså til at stå på plads nr 0 i vectoren osv.

Telefonnummeret står altid på den første linje i text filen og ligger altså altid på plads nr 0 i vectoren. Enhederne har 3 parametre som vi gerne vil gemme. Deres adresse, navn og status. De bliver gemt i den ordre, dvs at adressen på den første enhed ligger på plads nr 1 (vectornavn[1]) hvorefter navn står på nr 2 og status på nr 3.



*Figur 6.6.* Hukommelses header og udklip af tekst filen data gemmes i

# Modultest 7

---

## 7.1 Hardware Modultest

### 7.1.1 Encoder

Encoderen er testet vha et testprogram (Henvisning til burst.cpp). Testprogrammet sender et 120kHz signal ud i 1 ms, hver gang Zero cross signalet toggler. Herunder testes de enkelte blokke i encoderen.

#### Zero Crossing

Ud fra figur XX ses at Zero Crossing detectoren virker som ønsket. Hver gang 18 Vac/50Hz passerer 0, toggles Zero Cross signalet.

#### Højpasfilter

#### Output

## 7.2 Decoder

### 7.2.1 Zero Crossing

### 7.2.2 Båndpasfilter

### 7.2.3 Envelope detector

### 7.2.4 Output

## 7.3 SW Modultest

### 7.3.1 PC klasser

### 7.3.2 CSS hovedenhed klasser

Alle klasser i CSS hovedenhedspakken er testet. De enkelte tests er beskrevet i det følgende afsnit. Se det statiske klassediagram for at se sammenhængen mellem klasserne i figur 4.13.

#### Timer

Timer klassen er testet med programmet "Timer Test.cpp" som ligger i kildekoden til klassen. Programmet opretter et objekt af Timer typen og starter et endeløst loop som tænder timeren i 1 sekund og slukker den i 1 sekund.

#### Opstilling

En oscilloscope probe sættes på PD5 på et STK500 kit som har programmet kørende.

#### UART

UART klassen er testet med programmet "UART Test.cpp" som ligger i kildekoden til klassen. Programmet starter med at sende en streng over RS232 interfacet. Her efter afventer den at modtage en hel kommando. Når denne er modtaget sendes den retur.

#### Opstilling

Test programmet ligges på et STK500 kit. En computer forbindes med STK500 kittet over RS232 Spare porten (Sub-D). En jumper forbinder headeren RXD med PD0 og TXD med PD1. Start et terminalprogram op indstillet til 9600 buad, 8 databit, 1 stopbit og ingen paritet. Reset STK500 og programmet starter.

Først modtages strengen "CSS hovedenhed\r\n". Send igennem terminal her efter, én karakter af gangen: "A0101\r". Dette modtages igen som svar. Send igen, én karakter af gange: "D0011\r". Dette modtages igen som svar.

#### CircBuffer

CircBuffer klassen er testet med programmet "CircBuffer Test.cpp" som ligger i kildekoden til klassen.

#### Opstilling

Test programmet ligges på et STK500 kit. En computer forbindes med STK500 kittet over RS232 Spare porten (Sub-D). En jumper forbinder headeren RXD med PD0 og TXD med PD1. Start et terminalprogram op indstillet til 9600 buad, 8 databit, 1 stopbit og ingen paritet. Reset STK500 og programmet starter.

Det sender først de to index fra klassen som peger på den næste plads og den plads som skal aflæses næste gang adskilt af mellemrum. Denne forventes at være "0 0". Her efter indsættes en karakter ('A') og index tallende udskrives igen. Denne gang forventes det at NextIndex er steget, så "1 0". Karakteren læses så ud igen med get()-metoden og denne sendes efterfulgt af index tallende. Disse forventes nu at være "1 1".

Bufferen fyldes til sidste med 'B' 1.5 gange, altså overfyldes den. Den afsluttes med et 'C'. Her efter sendes alle værdier i bufferen retur indtil den rammer 'B'et. Der forventes at

komme 52 'B'er ud og et 'C'. Dette viser at det er en circulær buffer da de ældste værdi er overskrevet. Til sidst udskrives index tallende. Da de er castede fra ints til chars modtages "6 6" hvilket svare til 54.

## RS232IF

RS232IF klassen er testet med programmet "RS232IF Test.cpp" som ligger i kildekoden til klassen.

### Opstilling

Test programmet ligges på et STK500 kit. En computer forbindes med STK500 kittet over RS232 Spare porten (Sub-D). En jumper forbinder headeren RXD med PD0 og TXD med PD1. Et 8-ledet fladkabel sættes mellem LEDS og PORTB på STK500 kittet. Start et terminalprogram op indstillet til 9600 buad, 8 databit, 1 stopbit og ingen paritet. Reset STK500 og programmet starter.

Først afsender programmet de tre kommandoer som kan sendes til computeren. Hhv. avisering om babyalarm og svar på loginstatus. Forventet modtages "SB9999\r", "ST9999\r" og "SF9999\r". Her efter afventer programmet en fuld kommando. Hvis den stemmer overens med en af de mulige UC udskrives nummeret binaert på LEDerne på STK500 kittet og den sender den modtagende adresse retur. Testes med kommandoerne i tabel 8.1 og svar modtages. Bemærk at i Atmels terminal miljø skal karakterende sendes enkeltvis.

*Tabel 7.1.* Test kommandoer og svar for RS232IF modul test

Kommando ASCII	Kommando HEX	Svar
SA1010\r	53 41 31 30 31 30 0D	1010 LED1 lyser (UC2)
sd0011\r	73 64 30 30 31 31 0D	0011 LED1 og LED0 lyser (UC3)
sL9999\r	73 4C 39 39 39 39 0D	LED0 lyser (UC1)

## X10IF

X10IF klassen er testet med programmet "X10IF Test.cpp" som ligger i kildekoden til klassen.

### Opstilling

Test programmet ligges på et STK500 kit. En computer forbindes med STK500 kittet over RS232 Spare porten (Sub-D). En jumper forbinder headeren RXD med PD0 og TXD med PD1. Et 8-ledet fladkabel sættes mellem LEDS og PORTB på STK500 kittet. Jumper forbinder SW0 med PD2. Start et terminalprogram op indstillet til 9600 buad, 8 databit, 1 stopbit og ingen paritet. Reset STK500 og programmet starter.

Først sendes antallet af kommandoer i kø som ASCII tal. Der efter indsættes kommandoen aktiver i køen og antallet sendes igen. Dette gentages for deaktiver kommandoen. Der forventes udskrevet "0 1 2". Her efter tømmes bufferen og alle værdier sendes. Disse er formateret i X10 formatet og kontrolleres i hendhold til protokol beskrivelsen. Interrupt delen testes når LED7 lyser. Her er det muligt, ved tryk på SW0 at få alle dioderne til at lyse i 0,25 sekunder.

### ZeroCrossInt

ZeroCrossInt ISR er testet med programmet "ZeroCrossInt Test.cpp" som ligger i kildekoden til funktionen.

#### Opstilling

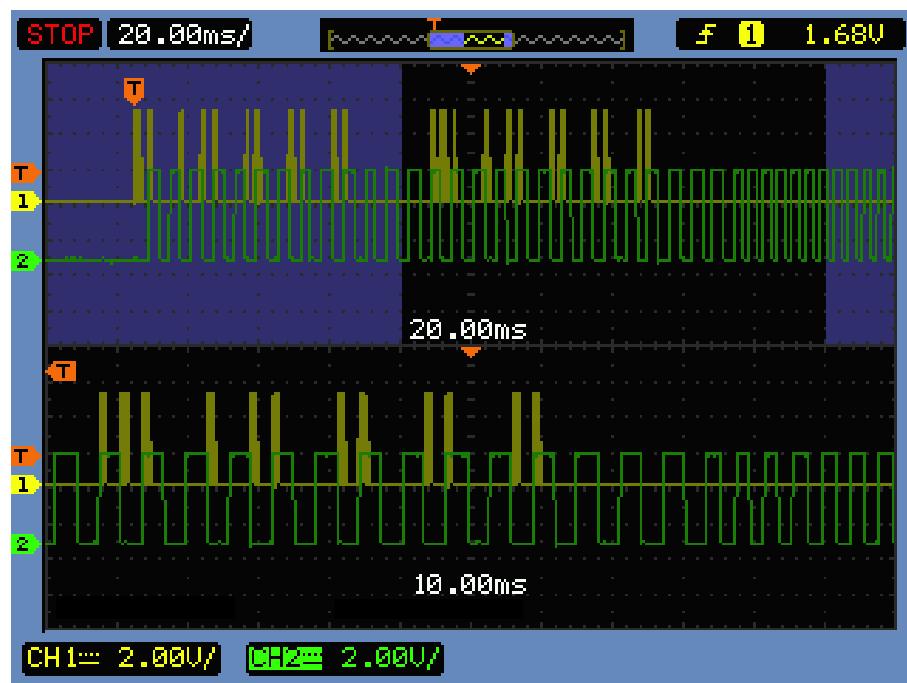
Test programmet ligges på et STK500 kit. Jumper forbinder PD2 og PB0. Et oscilloscope forbindes med to prober til PB2 og PD5. Reset STK500 og programmet starter.

Først lægges to kommandoer i kø, hhv. aktiver og deaktiver. Her efter genereres et firkantsignal på 250 Hz som føres ind på INT0 benet for at starte ISR funktionen.

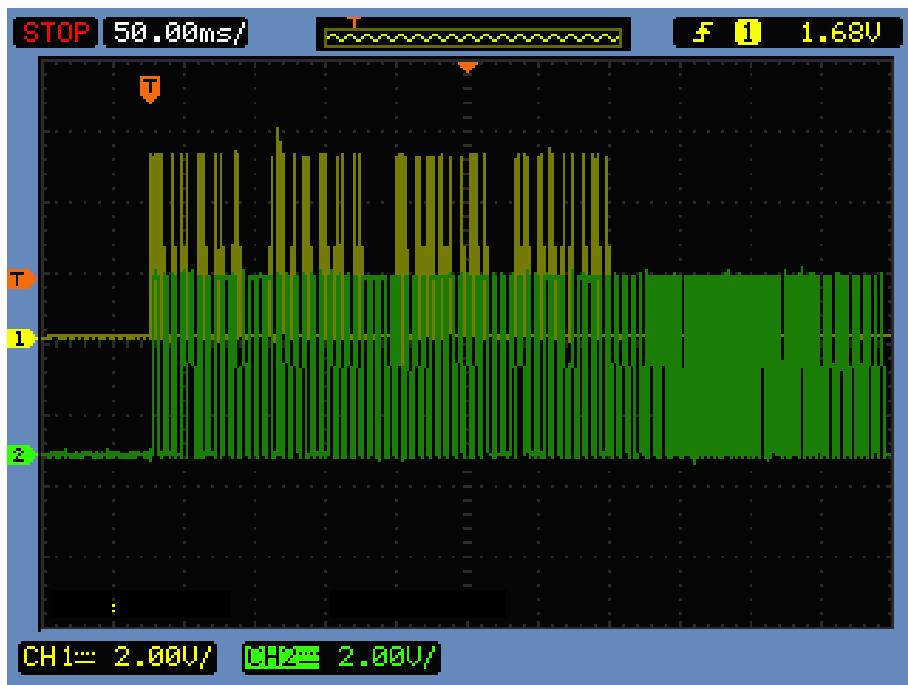
Med scopet kan så måles de rette bursts ud fra kommandoerne. På figur 8.1 kan man se de to kommandoer afsendt. Det gule signal er 120 kHz bursts svarende til X10 formaterede bits. Det grønne signal er det simulerede ZeroCross signal. I den øverste del kan man se aktiver kommandoen sendt to gange og i den nederste del er der zoomet ind på den sidste del. Dette stemmer overnes med protokol beskrivelsen.

På figur 8.2 kan man se både aktiver og deaktiver kommandoen. Bursts signalet skal være på 120 kHz.

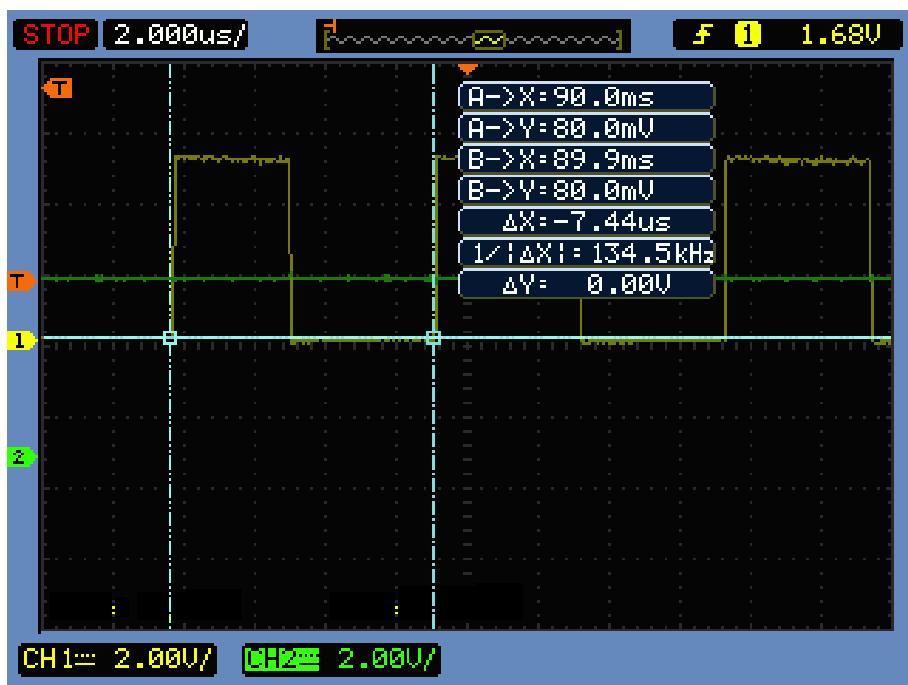
Figur 8.3 kan man se bursts signalet. Bemærk at frekvensen er lidt højere end forventet (134 kHz). Dette skyldes at den interne timer i STK500 kittet ikke kan komme 120 kHz nærmere. Reaktionstiden for programmet er målt i figur 8.4. Her kan det konstateres at fra der kommer et toggle på INT0 indgangen går der 32 us før burstet sendes afsted.



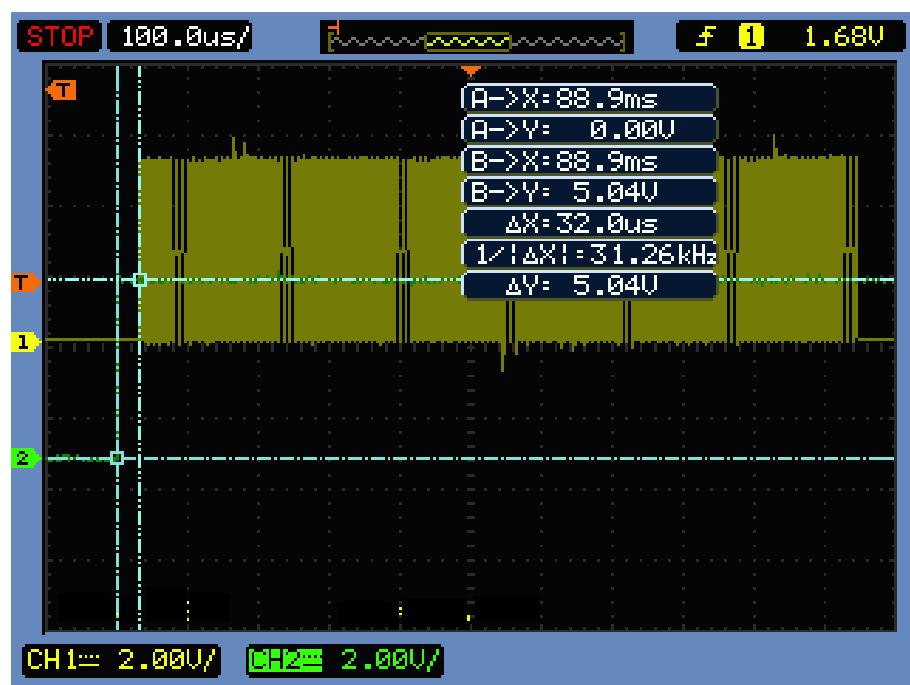
**Figur 7.1.** X10 Aktiver kommando. Oscilloscope måling, PB2 (Grøn) og PD5 (Gul)



**Figur 7.2.** X10 Aktiver og deaktiver kommandoer. Oscilloscope måling, PB2 (Grøn) og PD5 (Gul)



**Figur 7.3.** 120 kHz bursts. Oscilloscope måling, PB2 (Grøn) og PD5 (Gul)



Figur 7.4. Reaktionstid. Oscilloscope måling, PB2 (Grøn) og PD5 (Gul)

### 7.3.3 X10 Udtag klasser