

Indholdsfortegnelse

Kapitel 1	Ordliste	3
Kapitel 2 Kravspecifikation		
2.1	Indledning	5
2.2	Aktører	7
2.2.1	Bruger	7
2.2.2	Eksterne enheder	7
2.2.3	Barn	7
2.2.4	SMS modtager	7
2.2.5	DE2 Board	7
2.3	Usecases	8
2.3.1	UC1: Login	9
2.3.2	UC2: Aktiver	10
2.3.3	UC3: Deaktiver	11
2.3.4	UC4: Udlæs status	11
2.3.5	UC5: Detekter lyd	12
2.3.6	UC6: Rediger SMS-modtager	12
2.3.7	UC7: Startopsætning	13
2.3.8	UC8: Tilføj/fjern X10 udtag	13
2.4	Ikke-funktionelle krav	15
2.5	Begrænsninger	15
2.6	HMI(Human Machine Interface)	16
Kapitel 3 Forundersøgelse		
3.1	GSM	19
3.1.1	GSM-valg	20
3.2	Lås	21
3.2.1	Låsvalg	21
3.3	Babyalarm	22
3.3.1	Babyalarm sammenligning	22
Kapitel 4 System Arkitektur		
4.1	Domænemodel	23
4.2	Hardware	24
4.2.1	Hardware beskrivelse	24
4.2.2	BDD Hardware	24
4.2.3	BDD Hovedenhed	24
4.2.4	BDD Modtager	25
4.2.5	Plantegning over HW	25
4.2.6	IBD Hardware	26
4.2.7	IBD Hovedenhed og Modtager	26

4.2.8 Grænseflade	27
4.3 Software	29
4.3.1 Applikations model for PC	29
4.3.2 Klassebeskrivelse for PC	33
4.3.3 Applikations model for CSS hovedenhed	39
4.3.4 Klassediagram og beskrivelse for CSS hovednehed	41
4.3.5 Applications model for X10 modtager	53
4.3.6 Klassebeskrivels for X10 modtager	55
4.4 Protokol	56
4.4.1 Seriel kommunikation	56
4.4.2 X10 kommunikation	57
Kapitel 5 Hardware design	59
5.1 Encoder	59
5.1.1 Højpasfilter	60
5.1.2 Zero Crossing Detector	61
5.2 Decoder	62
5.2.1 Båndpasfilter	63
5.2.2 Envelope Detector	65
5.3 DE2-kodelås	65
Kapitel 6 Software design	67
6.1 Logical View	67
6.1.1 Klasse CircBuffer	68
6.1.2 Klasse X10IF	68
6.1.3 ZeroCrossInt funktion	69
6.2 Deployment View	71
6.3 Data view	72
Kapitel 7 Modultest	73
7.1 Hardware Modultest	73
7.1.1 Encoder	73
7.2 Decoder	74
7.2.1 Zero Crossing	75
7.2.2 Båndpasfilter	75
7.2.3 Envelope detector	76
7.2.4 Output	76
7.3 DE2-kodelås	76
7.4 SW Modultest	77
7.4.1 PC klasser	77
7.4.2 CSS hovedenhed klasser	78
7.4.3 X10 Udtag klasser	83

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	
v1.1	19-05-2014 Indledning + DE2 som aktør
v1.0	24-03-2014 Hele gruppen (efter 1. review)
v0.5	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

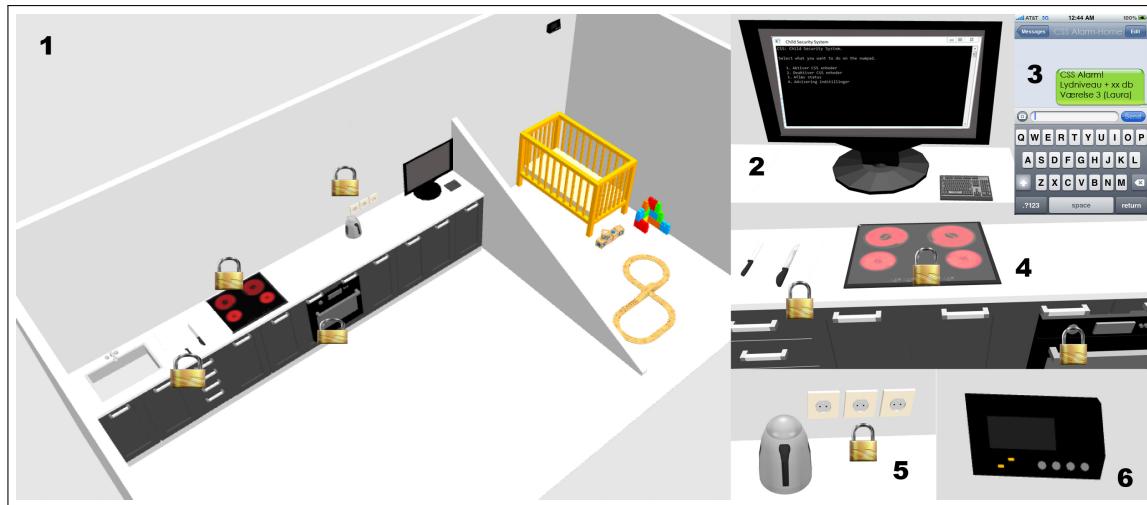
Den anden del af systemet er en babyalarm. Næsten alle mennesker i Danmark har deres mobiltelefon i nærheden hele tiden, så i stedet for at skulle have en babyalarm med rundt også, så kan man koble sin mobil til systemet og få besked når barnet giver lyd fra sig.

Dette ender ud i tre produkter:

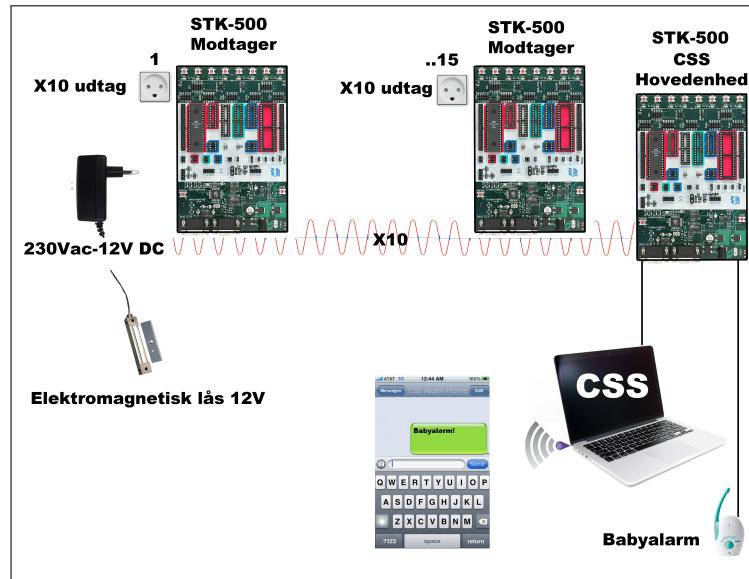
- Afbryder til valgt 230 Vac stikkontakt
 - Beskyttelse mod kogeplader og lignende
- Låsemekanisme til at låse skabe og skuffer
 - Aflæsning af skuffe med køkkenknive
- Babyalarm til lyddetektering
 - SMS-beskeder i stedet for en ekstra ”boks” i lommen

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.

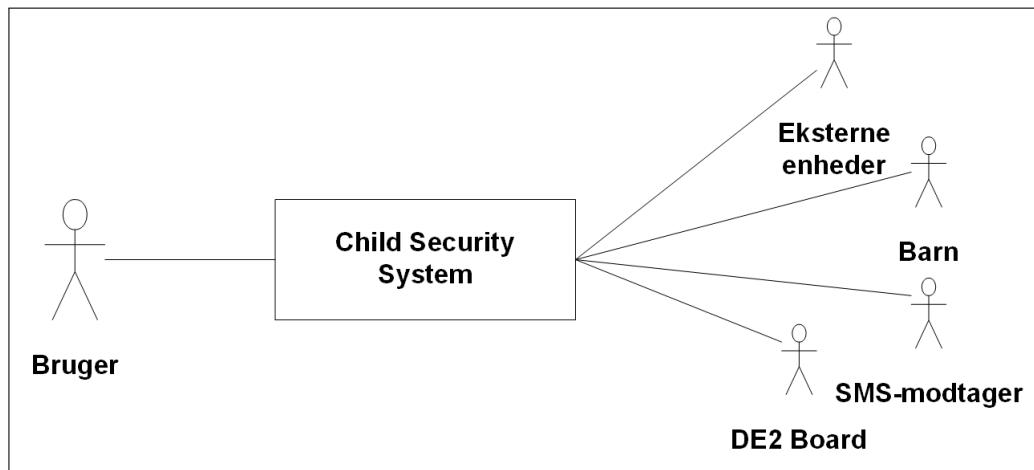
*Figur 2.1.* Installationsoversigt

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)
------------------	--

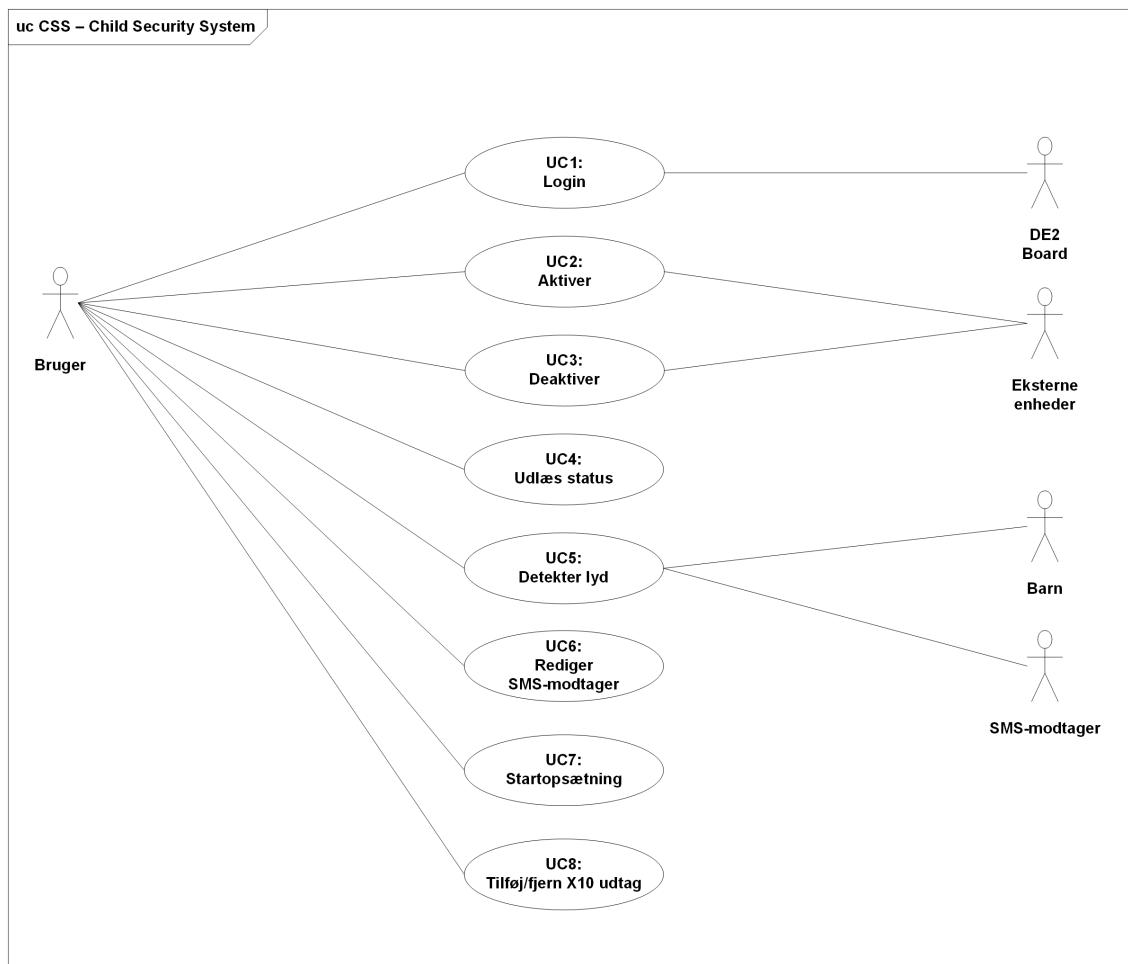
2.2.4 SMS modtager

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

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

2.3.2 UC2: Aktiver

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

2.3.3 UC3: Deaktiver

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

2.3.4 UC4: Udlæs status

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

2.3.5 UC5: Detekter lyd

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

2.3.6 UC6: Rediger SMS-modtager

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

¹Grunden 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

UC8: Tilføj/fjern X10 udtag	
Hovedforløb	<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>Tilføj valgt</p> <ul style="list-style-type: none"> a) Bruger indstiller addresseswitchen til en adresse på X10 udtaget b) Bruger indtaster den fire cifrede kombination som er indstillet på X10 udtaget efterfulgt af "Enter" <p>[Undtagelse 1b.a] Adressen er ikke unik</p> <p>[Undtagelse 1b.b] Adressen har ikke den rette længde</p> <ul style="list-style-type: none"> c) Programmet udskriver beskeden "Indtast navn" d) Bruger indtaster et selvvalgt navn for X10 udtaget efterfulgt af "enter" <p>[Undtagelse 1d.a] Navnet har ikke den rette længde</p> <ul style="list-style-type: none"> e) Bruger sætter X10 udtaget i det ønskede 230 Vac udtag <p>Fjern valgt</p> <ul style="list-style-type: none"> a) Den ønskede enhed markeres og der trykkes Fjern <p>2. Programmet returnerer til hovedskærmen</p>
Undtagelser	<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

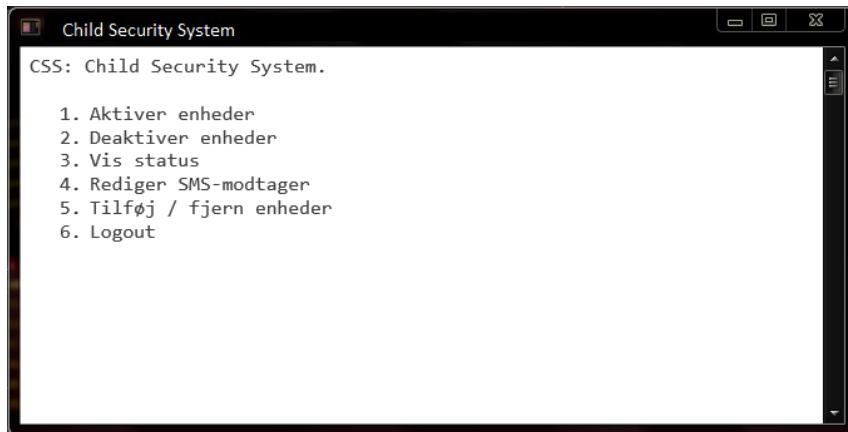
18. Lyddetektoren skal registrere lyde på over 68 dB
19. Der må maksimalt afsendes 1 SMS-besked pr. minut ved gentagende reaktion fra lyddetektoren
20. Låse enheder må maksimalt være 8x5x3 cm
21. Låse enhederne skal kunne holde 5 kilogram

2.5 Begrænsninger

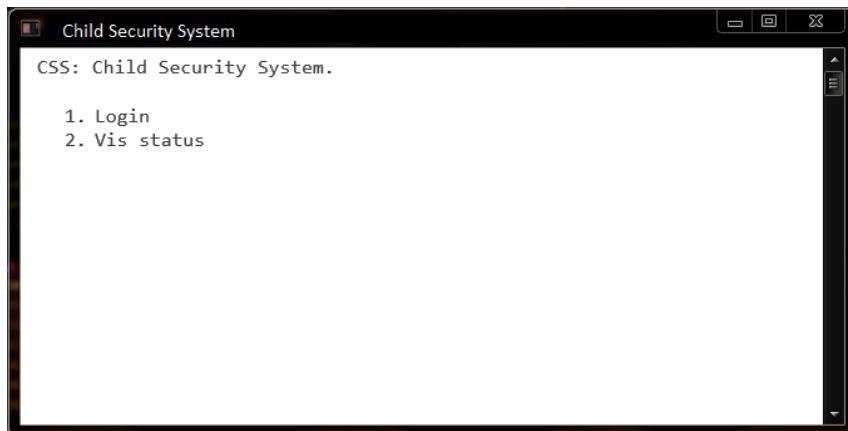
- Prototypen udføres i et 18 Vac testmiljø
- I stedet for magnetlåse til at simulere låsemekanismen bruges en lysindikator
- Prototypen udføres med et STK500 kit, hvorfor krav til dimensionerne frafalder

2.6 HMI(Human Machine Interface)

Billederne er inverteret for læsbarhedens skyld.



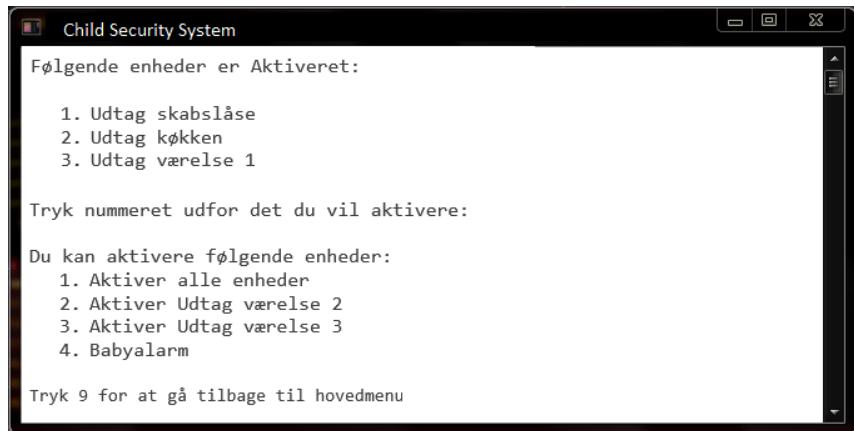
Figur 2.5. CSS Menu



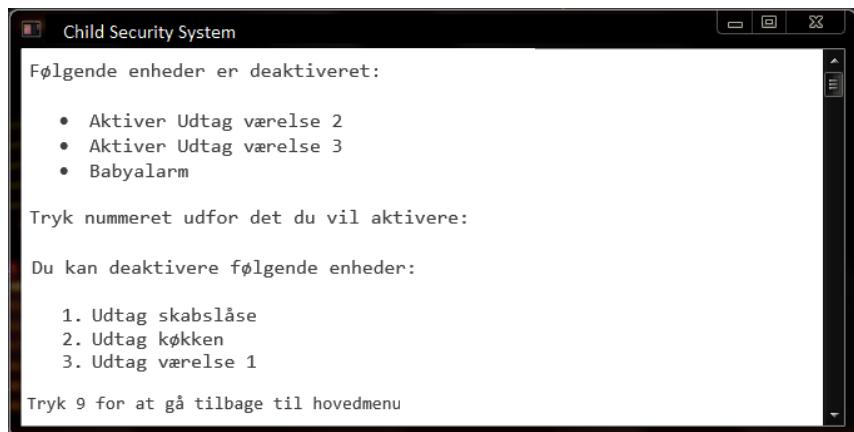
Figur 2.6. CSS Pre-Login



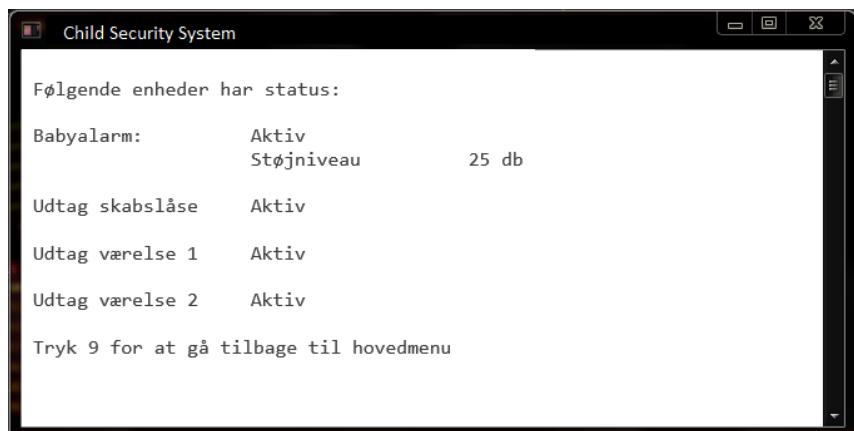
Figur 2.7. CSS Login



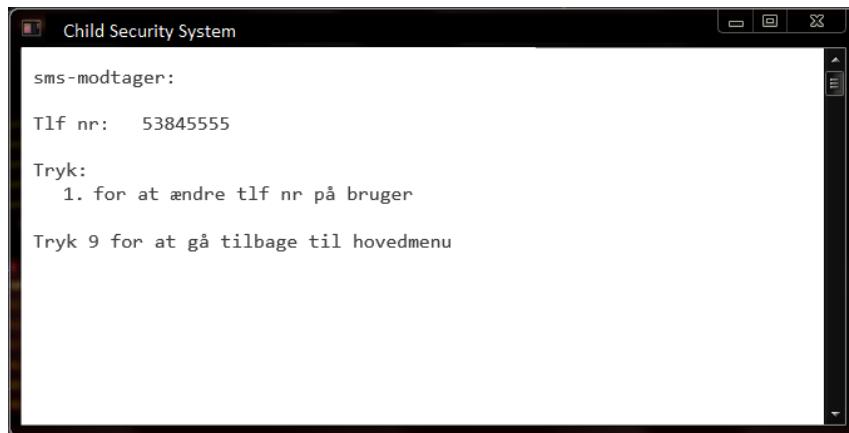
Figur 2.8. CSS Aktiver



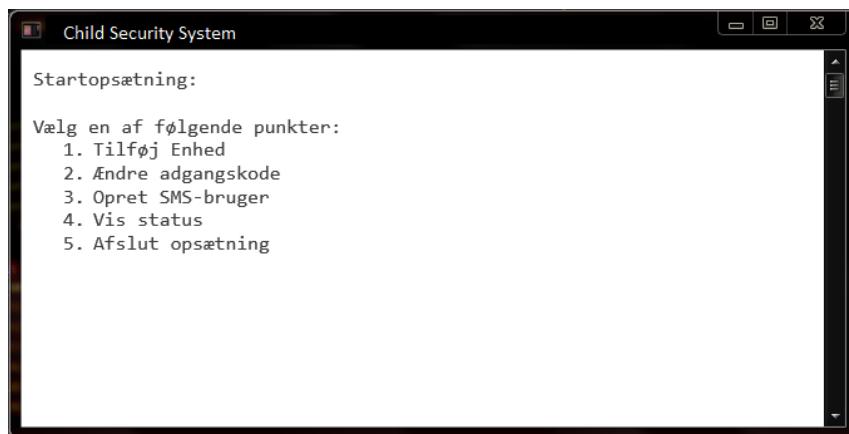
Figur 2.9. CSS Deaktivér



Figur 2.10. CSS Vis Status



Figur 2.11. CSS Advisering



Figur 2.12. CSS Startopsætning

Forundersøgelse 3

3.1 GSM

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

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



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



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

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



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



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

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



Navn	Supernova D7
Rækkevidde	600m
Lyd	Ukendt
Batteri	2 genopladelige batterier
Pris	999 kr
Stråling	Lav
Link	http://goo.gl/JFZcf5



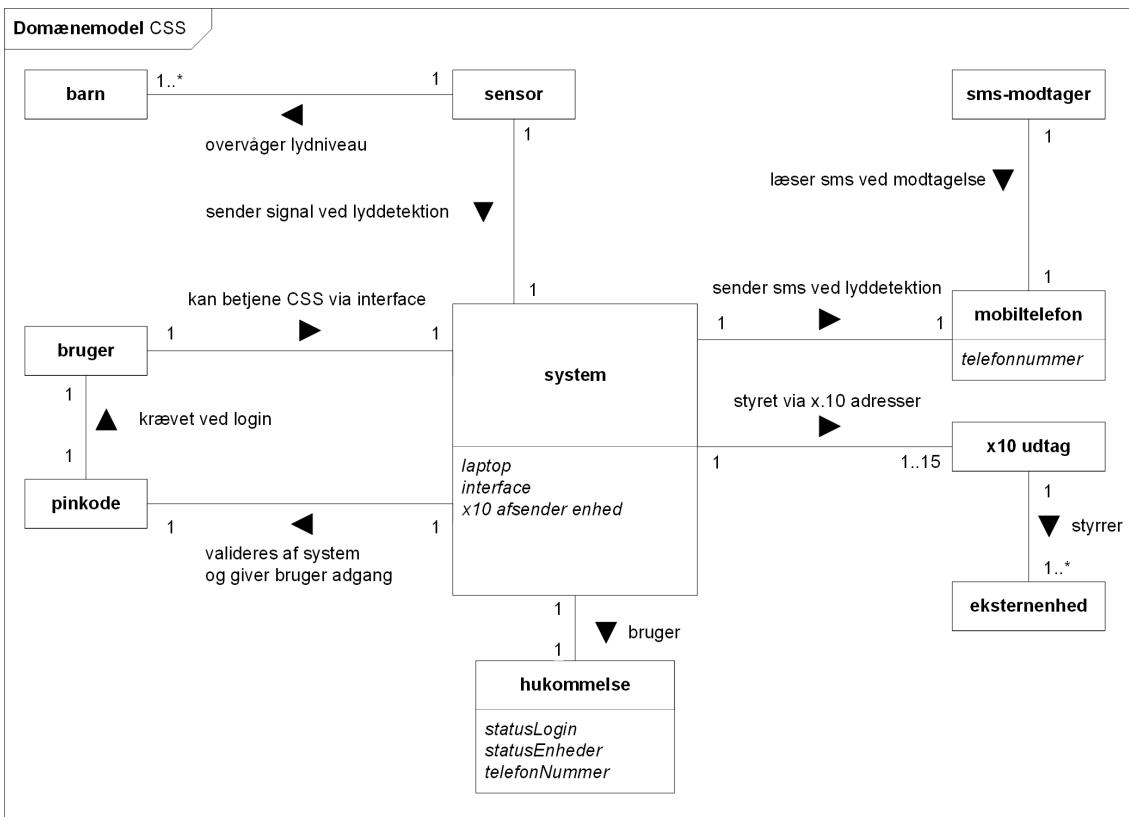
3.3.1 Babyalarm sammenligning

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



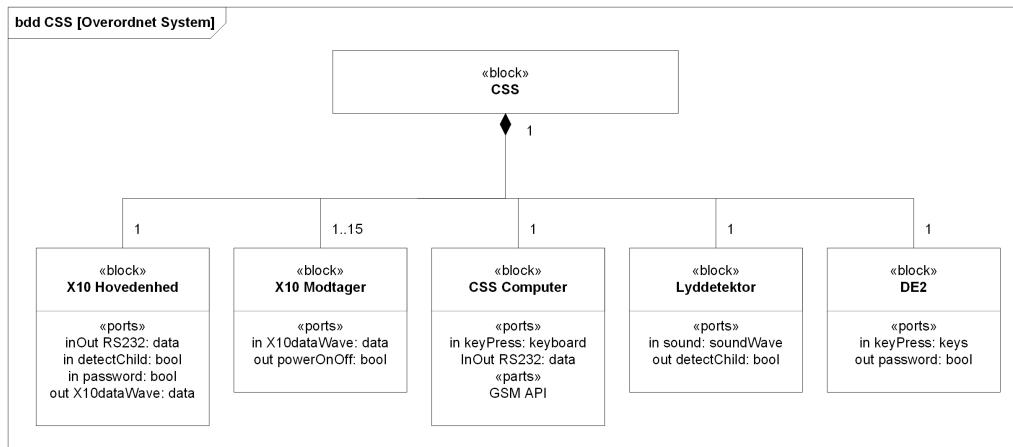
Figur 4.1. Domænemodel

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

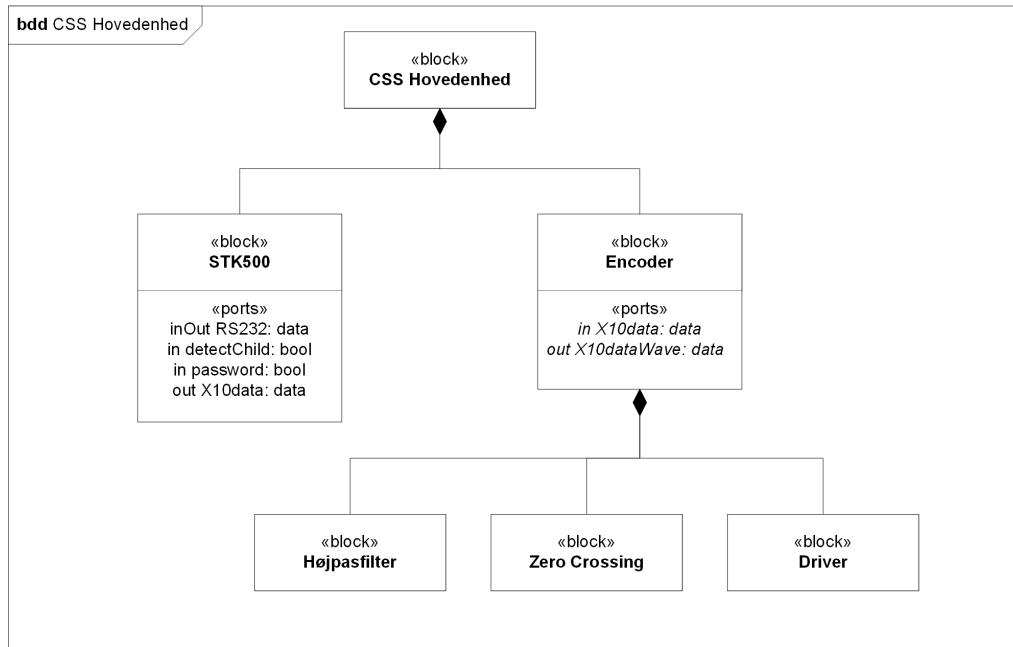
4.2.2 BDD Hardware



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

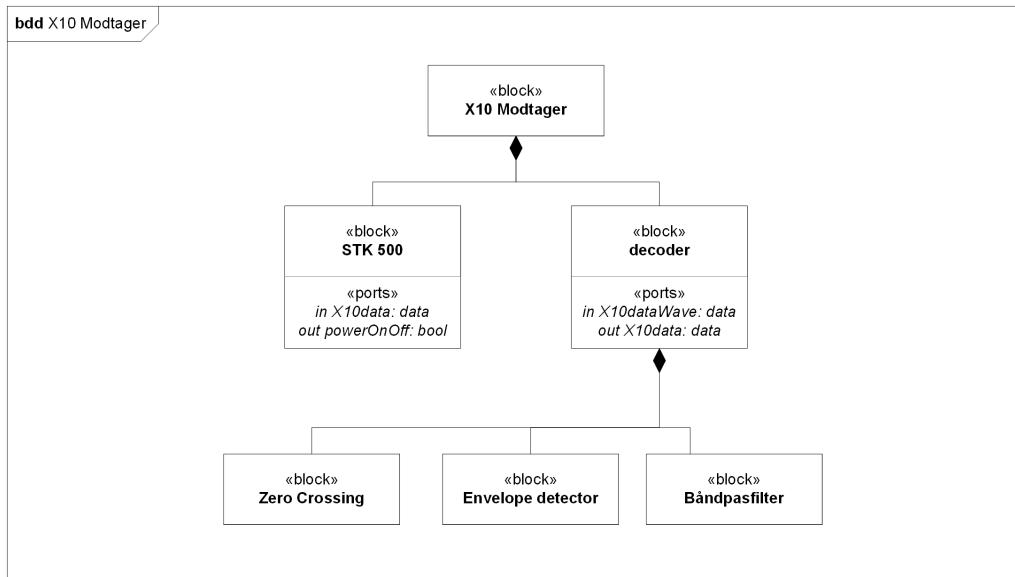


Figur 4.3. BDD Hovedenhed

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

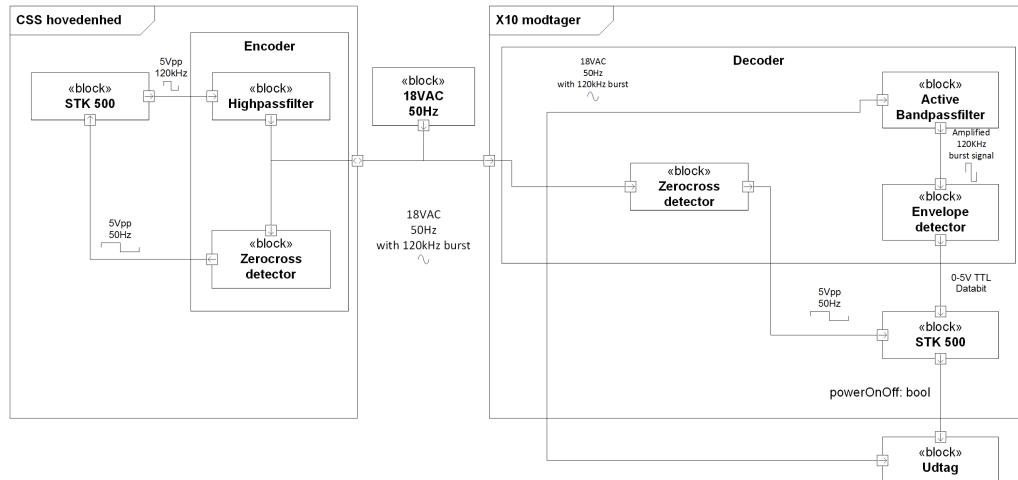
BDD diagrammet giver et overblik over hvad X10 modtageren består af. Vi ser en portbeskrivelse for STK kittet og decoderen.

4.2.4 BDD Modtager



Figur 4.4. BDD Modtager

4.2.5 Plantegning over HW



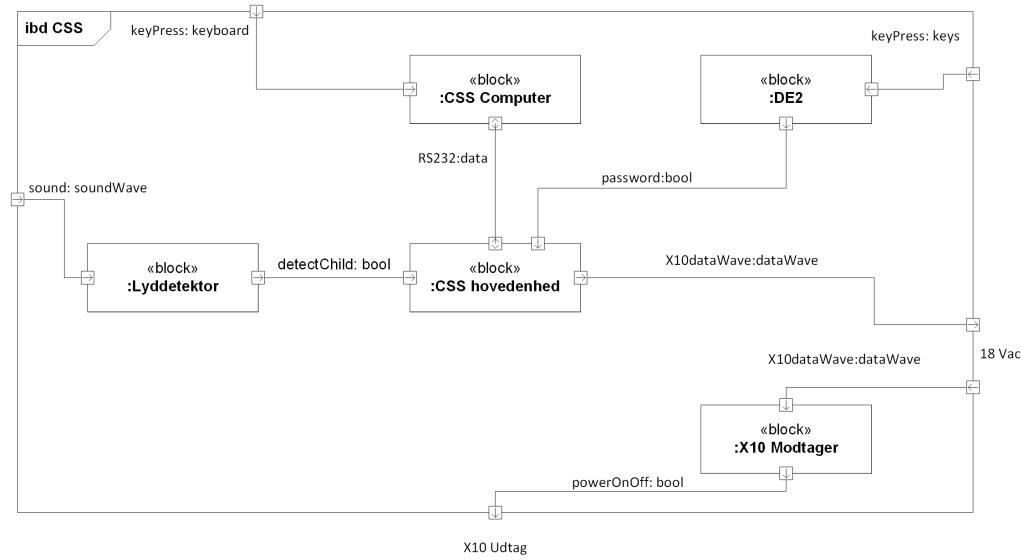
Figur 4.5. Plantegning over HW

Plantegningen over HW giver et overblik over hvordan CSS hovedenheden og X10 modtager er forbundet, samt hvilken type signaler der bliver sendt imellem dem.

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.

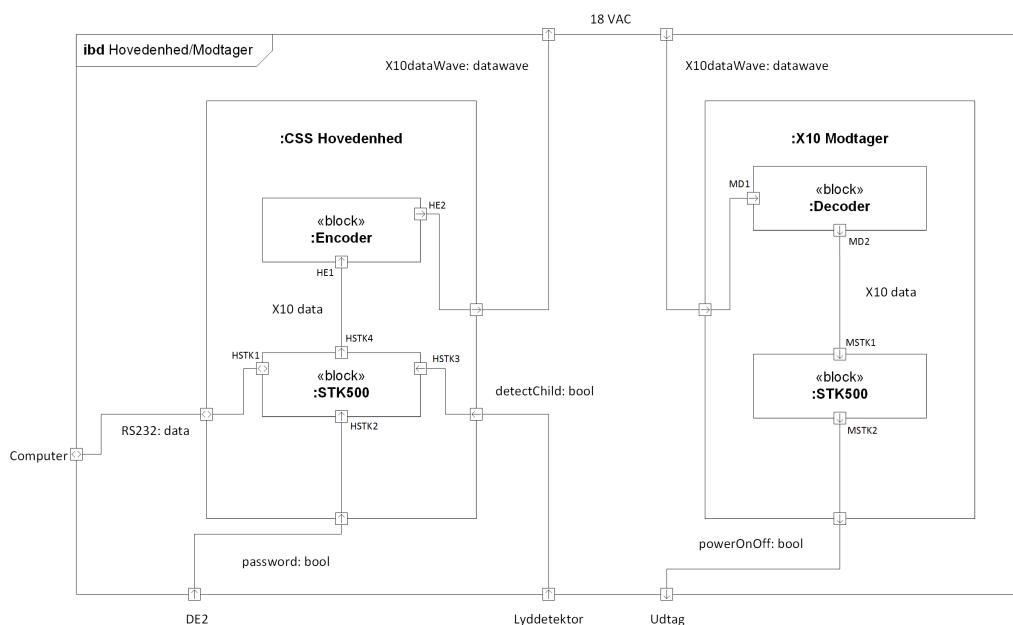
IBD diagrammet giver et internt overblik over hvordan vores X10 hovedenhed og X10 modtager er forbundet. Vi ser hvilke type signaler der bliver sendt imellem vores forskellige blokke.

4.2.6 IBD Hardware



Figur 4.6. IBD Hardware

4.2.7 IBD Hovedenhed og Modtager



Figur 4.7. IBD Hovedenhed og Modtager

4.2.8 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

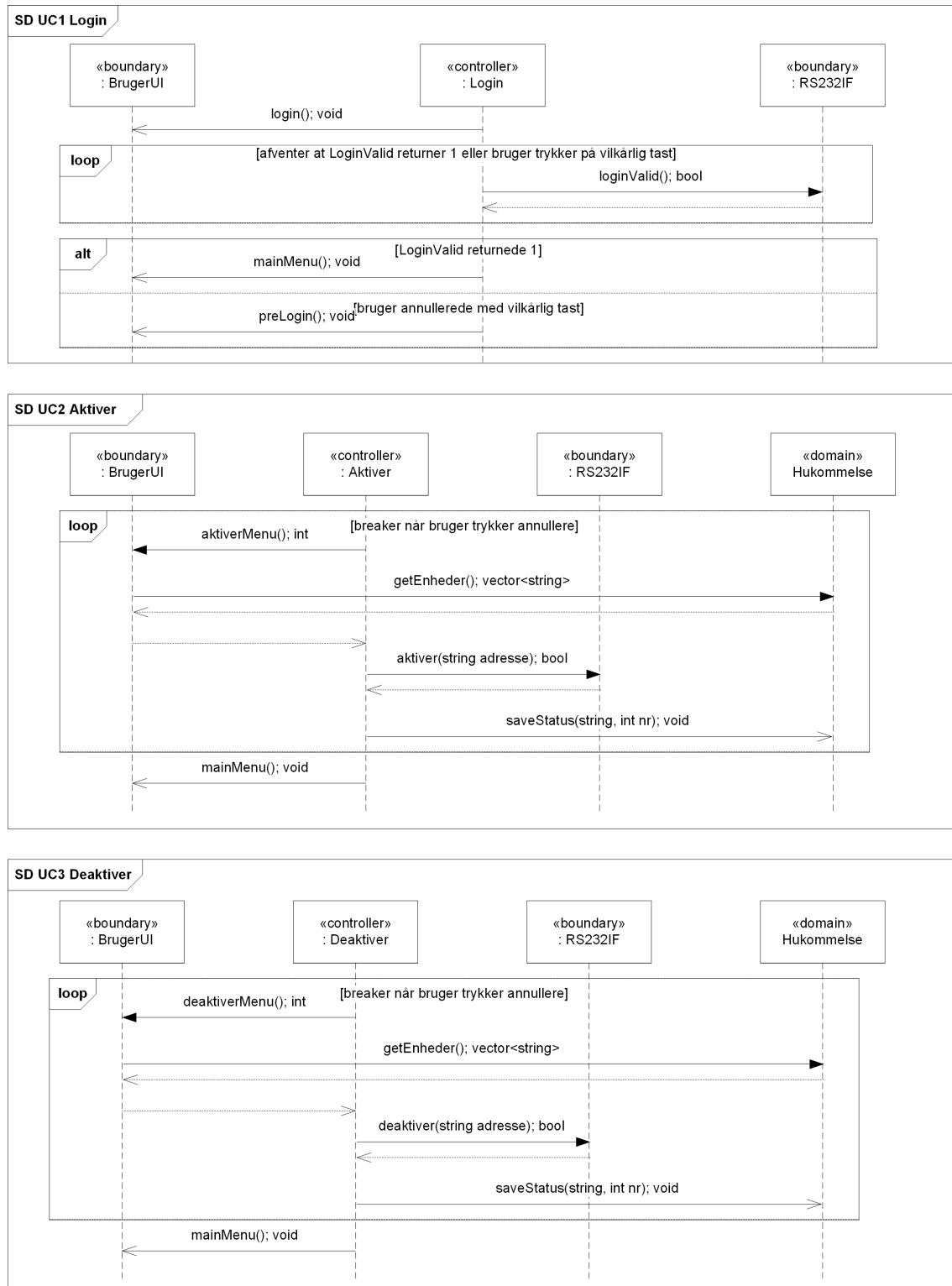
For at fuldende beskrivelsen af grænsefladen er der lavet en signaltabel som kan ses herunder. Hvert signal er beskrevet og tilknyttet en kort kommentar. Området et signal er defineret under er også beskrevet. Blok og terminal indgår også.

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



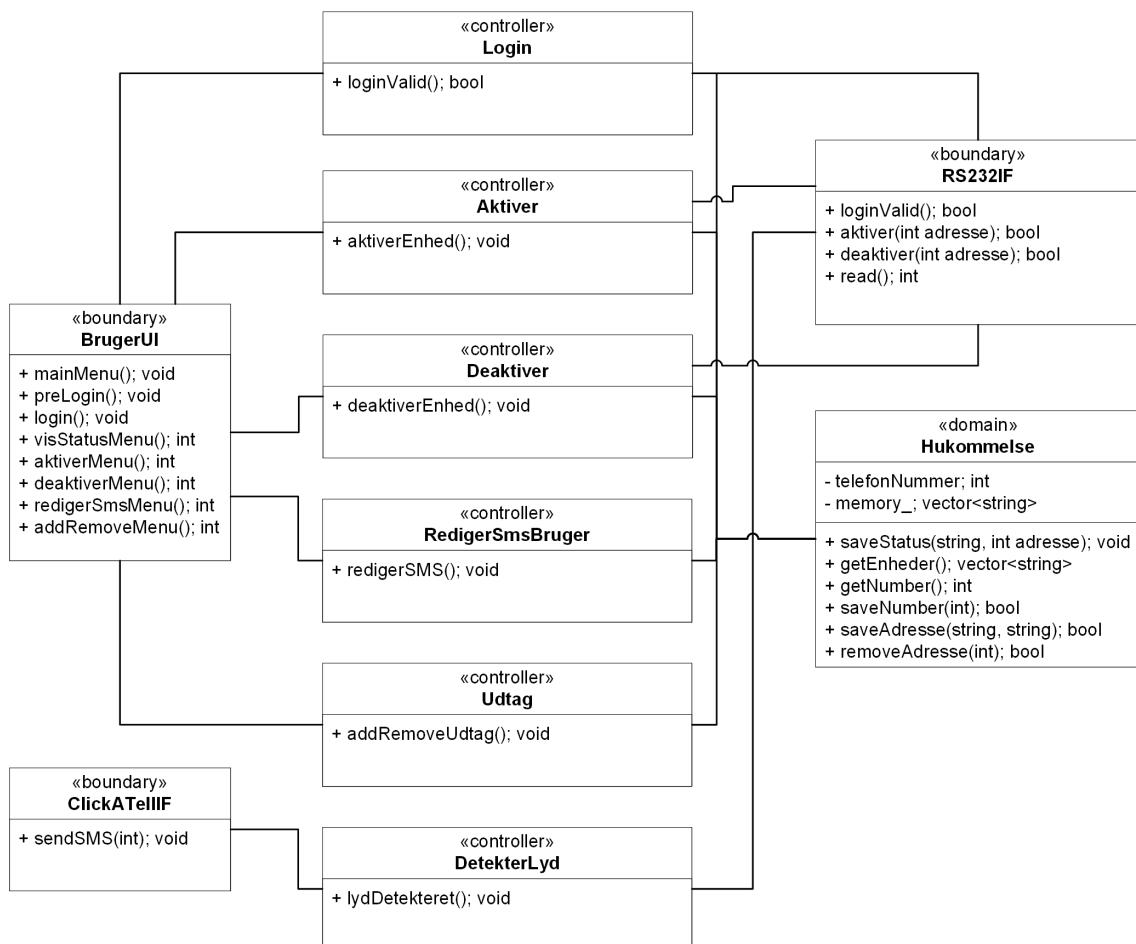
Figur 4.8. Use-case 1-3 sekvensdiagram for PC

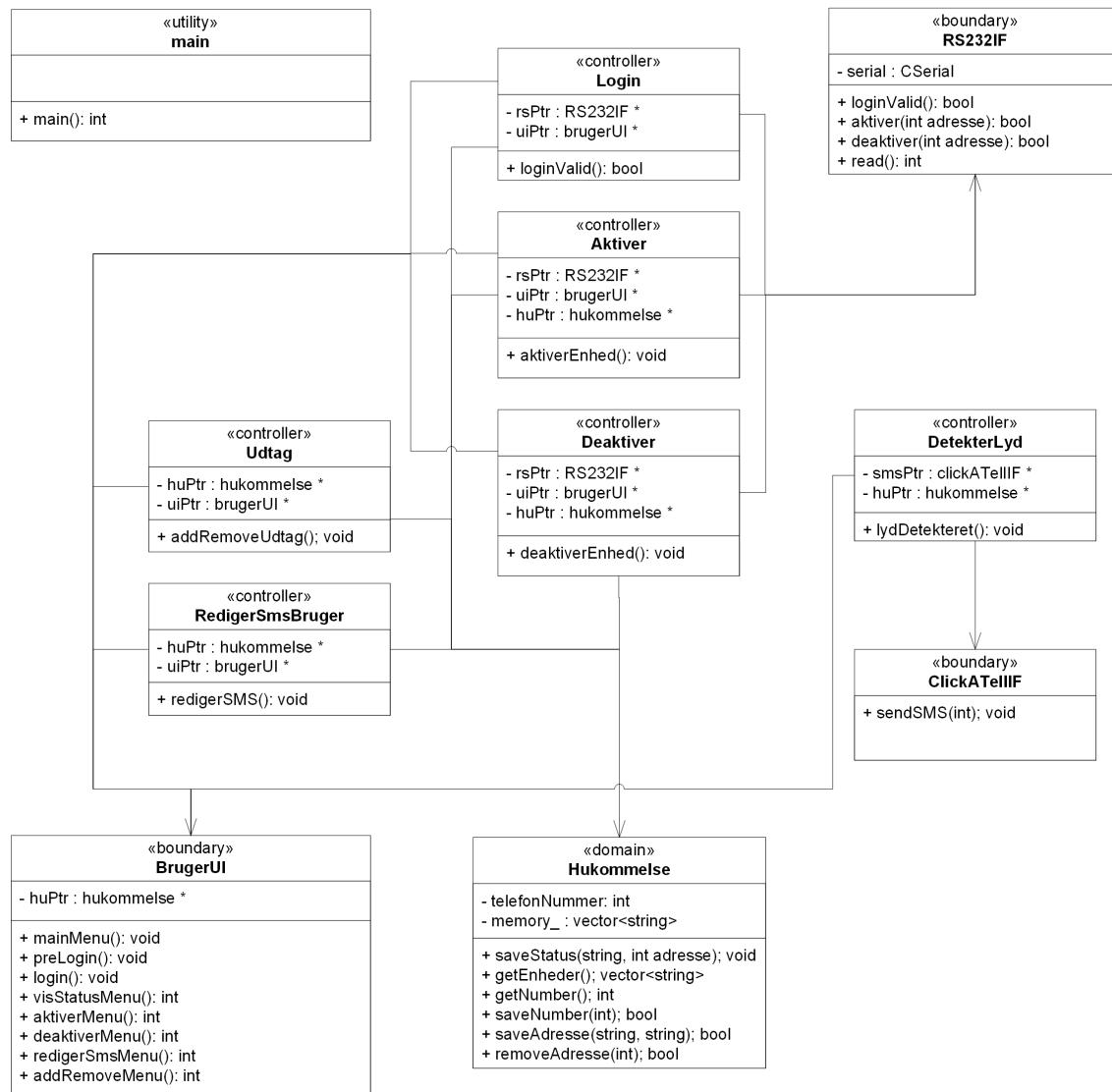


Figur 4.9. Use-case 5-8 sekvensdiagram for PC

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 4.10.** Klassediagram for PC



Figur 4.11. 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();
```

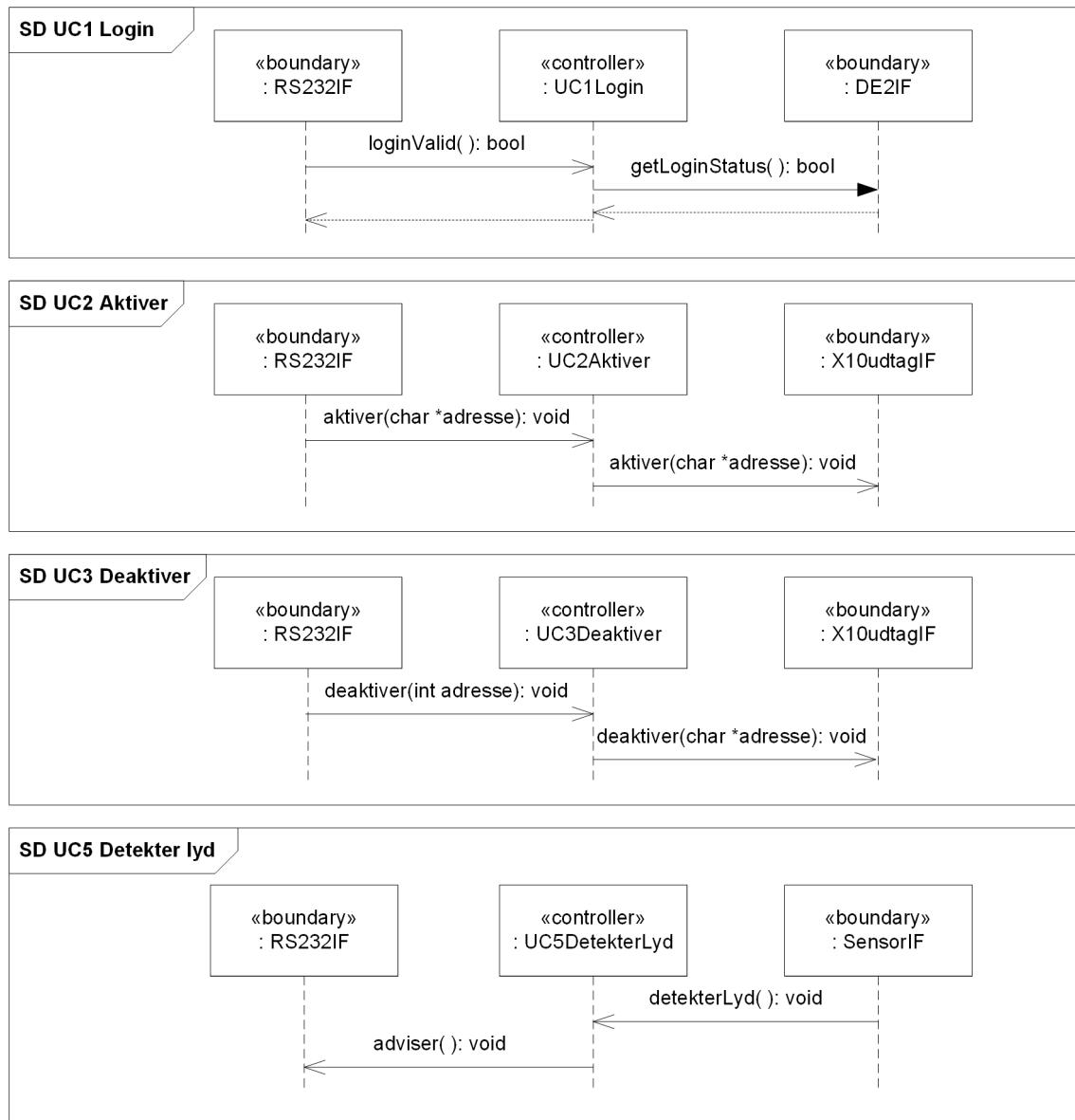
Parametre: ingen

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.

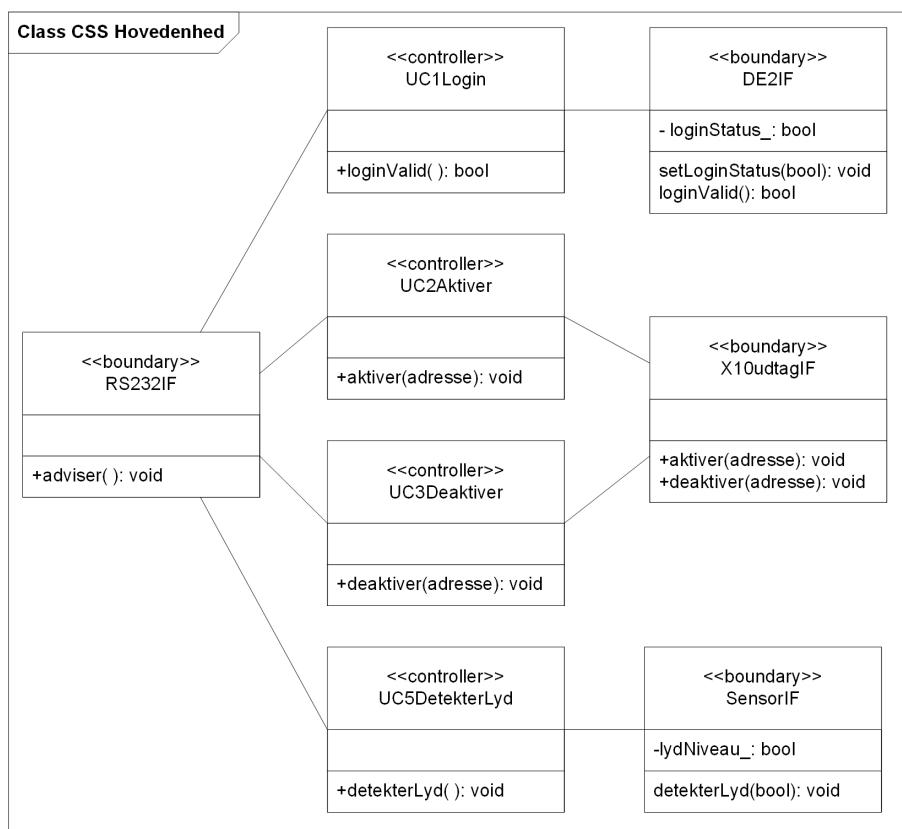
4.3.3 Applikations model for CSS hovedenhed

For CSS hovedenheden er udviklet en række diagrammer ud fra applikationsmodel metoden. Der er et sekvensdiagram på figur 4.12 for alle de aktuelle use-cases som beskriver systemets virkemåde. Ud fra dette er der lavet et klassediagram på figur 4.13 som dækker de forskellige use-cases med controller klasser og kommunikationen til PC via RS232 og X10 Udtag via X10.



Figur 4.12. Use-case sekvensdiagrammer for CSS hovedenhed

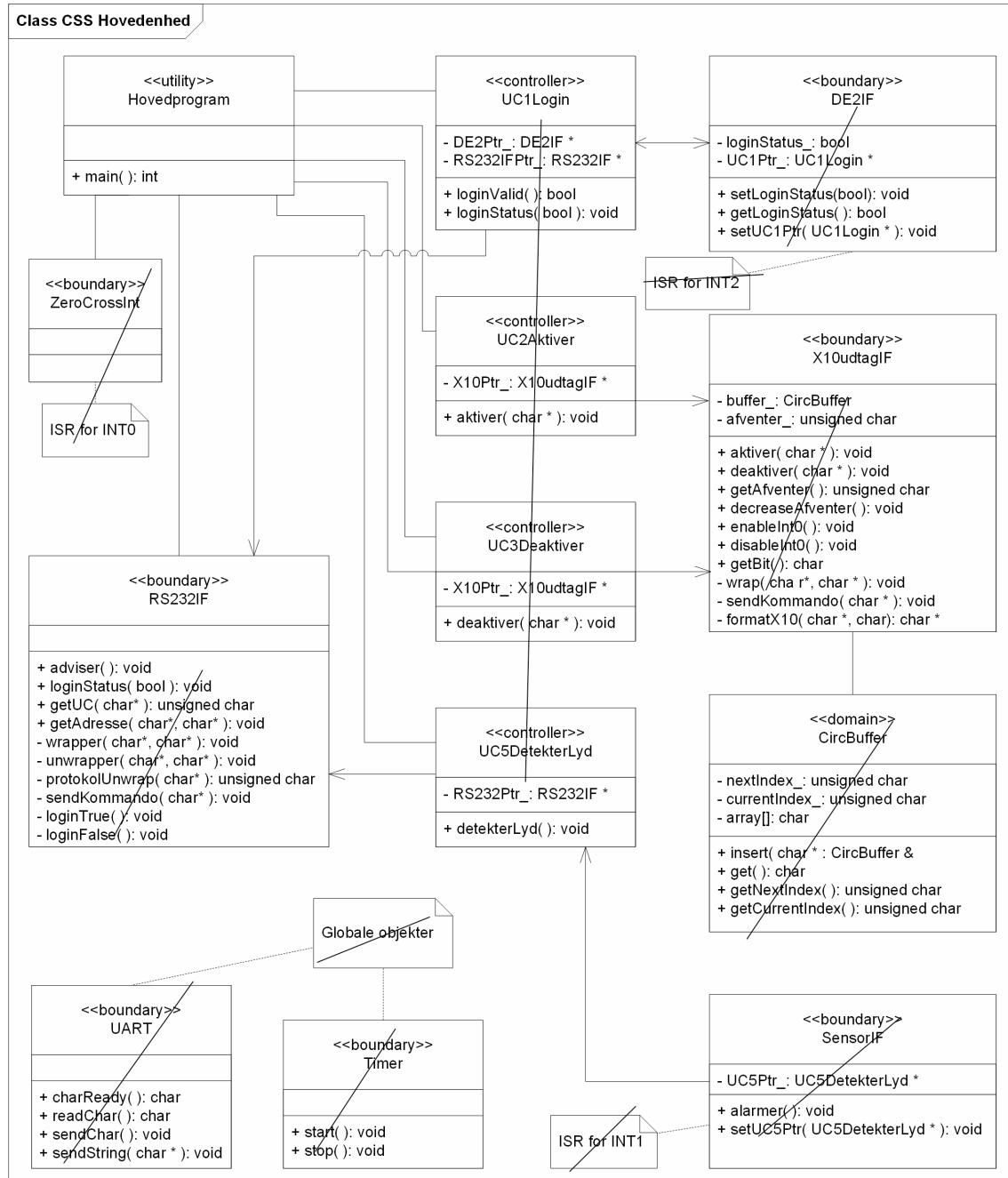
Klasse diagrammet er dog kun et udkast, da den endelige implementering vel medføre en række støtte funktionaliteter. Da det er udvikling til en microcontroller er der også det hardware mæssige aspekt som resulterer i en række global funktioner som kan tilgås fra hardwarenære funktioner. På figur 4.14 ses det endelige diagram.



Figur 4.13. Klassediagram for CSS hovedenhed

4.3.4 Klassediagram og beskrivelse for CSS hovednehed

Ud fra applikationsmodellens klasse diagram (Figur 4.13) og en dybere analyse af funktionalitete er der udledt et statisk klassediagram, se figur 4.14. Dette baserer sig på analyse og praktisk implementering.



Figur 4.14. Statisk klassediagram for CSS hovedenhed

Her følger klassebeskrivelser for alle klasser til CSS hovedenheden.

RS232IF

Ansvar: At varetage kommunikation mellem CSS hovedenhed og PC over RS232 protokol-

len.

Attributer: Ingen

Metoder:

void adviser();

Parametre: Ingen

Returværdi: Ingen

Beskrivelse: Sender kommando "SB9999\r" over RS232 forbindelsen

void loginStatus(bool status);

Parametre: Status for login, true = logget ind, false = logget ud

Returværdi: Ingen

Beskrivelse: Kalder hjælpemetoderne loginTrue() eller loginFalse() afhængig af status parameteren

unsigned char getUC(char * kommando);

Parametre: Pointer med plads til den modtagende RS232 kommando

Returværdi: Nummer på UC som tal

Beskrivelse: Afventer en fuld data pakke på RS232 forbindelsen. Gemmer den ud fra pointeren. Dekoder kommandoen iht. RS232 protokollen og returnerer UC nummeret

void getAdresse(char * kommando, char * adresse);

Parametre: Pointer til den modtagede kommando. Pointer med plads til adressen.

Returværdi: Ingen

Beskrivelse: Hvis kommandoen er formateret med STX og ETX fjernes disse og adressen skrives over i adresse pointeren

void wrapper(const char * kommando, char * wrapped);

Parametre: Pointer til rå kommando data uden STX og ETX. Pointer med plads til den fulde kommando inkl. STX og ETX

Returværdi: Ingen

Beskrivelse: Indsætter STX og ETX karaktere iht. RS232 protokollen

void unWrapper(const char * wrapped, char * kommando);

Parametre: Pointer til rå kommando data inkl. STX og ETX. Pointer med plads til kommandoen uden STX og ETX

Returværdi: Ingen

Beskrivelse: Fjerner STX og ETX karaktere iht. RS232 protokollen

unsigned char protokolUnwrap(char * kommando);

Parametre: Pointer til rå kommando data uden STX og ETX

Returværdi: Ingen

Beskrivelse: Returnerer et tal svarende til UC nummeret ud fra RS232 protokollen

```
void sendKommando( char * wrapped );
```

Parametre: Pointer til rå kommando data inkl. STX og ETX

Returværdi: Ingen

Beskrivelse: Sender hvert tegn i kommandoen, igennem det globale RS232UART objekt, ud på RS232 forbindelsen

```
void loginTrue();
```

Parametre: Ingen

Returværdi: Ingen

Beskrivelse: Sender kommando ”ST9999\r” over RS232 forbindelsen

```
void loginFalse();
```

Parametre: Ingen

Returværdi: Ingen

Beskrivelse: Sender kommando ”SF9999\r” over RS232 forbindelsen

UC1Login

Ansvar: At varetage UC1 Login forløbet.

Attributer:

- **DE2IF * DE2Ptr_**

Pointer til associeret DE2IF objekt

- **RS232IF * RS232IFPtr_**

Pointer til associeret RS232IF objekt

Metoder:

```
bool loginValid();
```

Parametre: Ingen

Returværdi: True = logget ind, False = logget ud

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

```
void loginStatus( bool status );
```

Parametre: True = logget ind, False = logget ud

Returværdi: Ingen

Beskrivelse: Kalder loginStatus() metoden i RS232IF objektet

UC2Aktiver

Ansvar: At varetage UC2 Aktiver forløbet.

Attributer:

- **X10IF * X10IFPtr_**

Pointer til associeret X10IF objekt

Metoder:

```
void aktiver( char * adresse );
```

Parametre: Pointer til adressen på enhed

Returværdi: Ingen

Beskrivelse: Kalder aktiver() metoden i X10IF objektet med den modtagede adresse

UC3Deaktiver

Ansvar: At varetage UC3 Deaktiver forløbet.

Attributer:

- **X10IF * X10IFPtr**

Pointer til associeret X10IF objekt

Metoder:

```
void deaktiver( char * adresse );
```

Parametre: Pointer til adressen på enhed

Returværdi: Ingen

Beskrivelse: Kalder deaktiver() metoden i X10IF objektet med den modtagede adresse

UC5DetekterLyd

Ansvar: At varetage UC5 Detekter Lyd.

Attributer:

- **RS232IF * RS232Ptr**

Pointer til associeret RS232IF objekt

Metoder:

```
void detekterLyd();
```

Parametre: Ingen

Returværdi: Ingen

Beskrivelse: Kalder adviser() metoden i RS232IF objektet

DE2IF

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

Dette er en global klasse da ISR for INT2 benet kalder setLoginStatus() metoden. Det globale objekt hedder DE2IFObj.

Attributer:

- **bool loginStatus**

True = Login bekræftet på DE2 board

False = Login ikke bekræftet på DE2 board

- **UC1Login * UC1Ptr**

Pointer til associeret UC1Login objekt

Metoder:

```
void setLoginStatus( bool status );
```

Parametre: True hvis logget ind, False hvis logget ud

Returværdi: Ingen

Beskrivelse: Sætter attribut loginStatus_ til aktuel status på DE2 board og kalder endten start()- eller stop()-metoden i det globale loginTimer objekt hvis status er hendholdsvis true eller false

```
bool getLoginStatus();
```

Parametre: Ingen

Returværdi: True hvis logget ind, False hvis logget ud

Beskrivelse: Returnerer attribut loginStatus_

```
void setUC1Ptr( UC1Login * Ptr );
```

Parametre: Pointer til associeret UC1Login objekt

Returværdi: Ingen

Beskrivelse: Sætter pointer attribut til Ptr

```
ISR( INT2_vect );
```

Parametre: Adresse til INT2 vectoren

Returværdi: Ingen

Beskrivelse: Kald setLoginStatus() metoden med parameter true

X10IF

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

Dette er en global klasse da ISR for INT0 tilgår buffer_ objektet. Det globale objekt hedder X10IFObj.

Attributer:

- **CircBuffer buffer_**

CircBuffer objekt til at holde kommandoer inden afsendelse

- **unsigned char afventer_**

Holder antallet af kommandoer der venter på at blive afsendt. Kun positive tal

Metoder:

```
void aktiver( char * adresse );
```

Parametre: Pointer til adresse på enhed der skal aktiveres

Returværdi: Ingen

Beskrivelse: Afsend kommandoen "SAXXXX\r", hvor XXXX er adressen på enheden der skal aktiveres, over X10 forbindelsen ved brug af sendKommando() metoden

```
void deaktiver( char * adresse );
```

Parametre: Pointer til adresse på enhed der skal deaktiveres

Returværdi: Ingen

Beskrivelse: Afsend kommandoen "SDXXXX\r", hvor XXXX er adressen på enheden der skal deaktiveres, over X10 forbindelsen ved brug af sendKommando() metoden

unsigned char getAfventer();

Parametre: Ingen

Returværdi: Ingen

Beskrivelse: Returner afventer_ attributen

void decreaseAfventer();

Parametre: Ingen

Returværdi: Ingen

Beskrivelse: Hvis afventer_ ikke er 0 trækkes 1 fra

void enableInt0();

Parametre: Ingen

Returværdi: Ingen

Beskrivelse: Aktiver interrupts for INT0 benet ved toggles og aktiver global interrupt flag

void disableInt0();

Parametre: Ingen

Returværdi: Ingen

Beskrivelse: Deaktiver interrupts for INT0 benet

char getBit();

Parametre: Ingen

Returværdi: Ingen

Beskrivelse: Hent næste karakter i buffer_ objektet med metoden get()

void wrap(char * unwrapped, char * wrapped);

Parametre: Pointer til rå kommando uden STX og ETX. Pointer med plads til X10 formateret kommando

Returværdi: Ingen

Beskrivelse: Omskriv kommando fra RS232 format ("SAXXXX\r") til X10 format inkl. STX og ETX iht. X10 protokollen

void sendKommando(char * unwrapped);

Parametre: Pointer til kommando i RS232 format ("SAXXXX\r") uden STX og ETX

Returværdi: Ingen

Beskrivelse: Omformater kommando til X10 format og indsæt i buffer_ objektet. Terminator bufferen med '\n'. Forøg afventer_ attributen og aktiver interrupts på INT0

```
char * formatX10( char * plads, char bit );
```

Parametre: Pointer med plads til X10 formateret bit (2 char plader). Char som skal formateres fra. Kun '0' og '1' gyldige.

Returværdi: Pointer til næste plads i X10 kommando

Beskrivelse: Formaterer bit om til X10 formaterede bits iht. X10 protokollen og returnerer *plads + 2

CircBuffer

Ansvar: At holde kommandoer som ligger i kø til afsendelse.

Attributer:

- **unsigned char nextIndex_**
Index til næste plads til skrivning
- **unsigned char currentIndex_**
Index til næste plads til læsning
- **char array[]**
Array til at holde bit karakterende

Metoder:

CircBuffer(); (Constructor)

Parametre: Ingen

Returværdi: Ingen

Beskrivelse: Initialiser attributter til 0. Fyld array med '\0'

```
CircBuffer & insert( char * bit );
```

Parametre: Pointer til bit der skal indsættes i bufferen

Returværdi: Reference til objektet (this)

Beskrivelse: Indsætter værdi i bufferen og ligger en til nextIndex_ attributten. Hvis denne er større end arrayets størrelse startes fra 0

```
char get( );
```

Parametre: Ingen

Returværdi: Udlæst bit fra arrayet

Beskrivelse: Udlæser bit fra arrayet. Lgger en til currentIndex_ attributten. Hvis denne er større end arrayets størrelse startes fra 0

```
unsigned char getNextIndex( );
```

Parametre: Ingen

Returværdi: Attributten nextIndex_

Beskrivelse: Returnerer attributten nextIndex_

```
unsigned char getCurrentIndex( );
```

Parametre: Ingen

Returværdi: Attributten currentIndex_

Beskrivelse: Returnerer attributten currentIndex_

SensorIF

Ansvar: At holde styr på aktuel lyd detektering og give besked over RS232 forbindelsen hvis lyd registreres.

Dette er en global klasse da ISR for INT1 tilgår alarmer() metoden. Det globale objekt hedder SensorIFObj.

Attributer:

- UC5DetekterLyd * UC5Ptr_

Pointer til associeret UC5 objekt

Metoder:

void alarmer();

Parametre: Ingen

Returværdi: Ingen

Beskrivelse: Kalder detekterLyd() metoden i UC5DetekterLyd objektet.

void setUC5Ptr(UC5DetekterLyd * Ptr);

Parametre: Pointer til associeret UC5DetekterLyd objekt

Returværdi: Ingen

Beskrivelse: Initialiser UC5Ptr_ til Ptr

ISR(INT1_vect);

Parametre: Adresse til INT1 vector

Returværdi: Ingen

Beskrivelse: Kald alarmer() metoden i det globale SensorIFObj objekt

ZeroCrossInt

Ansvar: At detekterer ZeroCross signalet og afsende X10 kommandoer ud fra dette.

Attributer: Ingen **Metoder:**

ZeroCrossInt(); (Constructor)

Parametre: Ingen

Returværdi: Ingen

Beskrivelse: Aktiver interrupts for INT0 og aktiver det globale interrupt flag

~ZeroCrossInt(); (Destructor)

Parametre: Ingen

Returværdi: Ingen

Beskrivelse: Deaktiver interrupts for INT0

ISR(INT0_vect);

Parametre: Adresse til INT0 vector

Returværdi: Ingen

Beskrivelse: Hvis der er data i køen på X10IFObj objektet udhentes et bit fra den. Bittet afsendes ved at tænde for det globale Timer objekt iht. X10 protokollen. Hvis kommando termineringen udtages fra bufferen tælles køen ned. Hvis køen er tom deaktiveres interruptet ved at kalde disableInt0() metoden i X10IFObj objektet.

UART

Ansvar: At interface med USART hardwaren på atMega32 processoren på STK500 kittet.

Dette er en global klasse og det globale objekt hedder RS232UART

Attributer:

- **char dataIn[]**

Global: Array til at holde modtagede karakterer

- **unsigned char volatile dataCount**

Global: Tæller til at registrerer mængden af modtagende karakterer

- **unsigned char volatile commandReady**

Global: Flag til registrering af fuldt modtaget kommando

Metoder:

UART(unsigned long BaudRate, unsigned char databit); (Constructor)

Parametre: Buadrate. Mellem 110 og 115200. Antallet af databits. Mellem 5 og 9.

Returværdi: Ingen

Beskrivelse: Sæt de hardwarenære registre til kommunikation med USART delen af at-Mega32 processoren samt modtage interrupts

~UART(); (Destructor)

Parametre: Ingen

Returværdi: Ingen

Beskrivelse: Deaktiver USART afsendelse og modtagelse samt modtage interrupt

char charReady();

Parametre: Ingen

Returværdi: >0 hvis data modtaget

Beskrivelse: Returner > 0 hvis der er modtaget data, ellers 0

char readChar();

Parametre: Ingen

Returværdi: Modtaget karakter

Beskrivelse: Afvent data modtagning og returner modtaget char

void sendChar(char tegn);

Parametre: Karakter til afsendelse

Returværdi: Ingen

Beskrivelse: Afvent afsendelsesbuffer klar og send karakter

```
void sendString( char * streng );
```

Parametre: Pointer til nul-termineret karakter streng

Returværdi: Ingen

Beskrivelse: Kald sendChar() metoden for hver karakter indtil nul-terminering

```
ISR( USART_RXC_vect );
```

Parametre: Adresse til USART modtagnings vector

Returværdi: Ingen

Beskrivelse: Læs modtaget data ind i dataIn[] arrayet. Registrer om en fuldt modtaget og termineret kommando er modtaget. Hvis det er tilfældet nulstilles dataCount og commandReady flaget sættes. Hvis der er modtaget flerer karakterer end kommandolængeden og de ikke har rigtige STX og ETX nulstilles bufferen. Hvis intet af ovenstående udføres, øges dataCount med 1

Timer

Ansvar: At genererer 120 kHz firkant signal med intern hardware timer.

Dette er en global klasse da ISR for INT0 kan tilgå start() og stop() metoderne. Det globale objekt hedder timer.

Attributer: Ingen **Metoder:**

Timer(); (Constructor)

Parametre: Ingen

Returværdi: Ingen

Beskrivelse: Sæt output port og indstil hardware registrer til timer1 på atMega32 processoren til at genererer 120 kHz firkant signal

```
~ZeroCrossInt( ); (Destructor)
```

Parametre: Ingen

Returværdi: Ingen

Beskrivelse: Afbryd hardware forbindelse til timeren og stop den. Sæt ouput ben lavt

```
void start( );
```

Parametre: Ingen

Returværdi: Ingen

Beskrivelse: Sæt hardware forbindelse til output ben og start timer1

```
void stop( );
```

Parametre: Ingen

Returværdi: Ingen

Beskrivelse: Afbryd hardware forbindelse til timeren og stop den. Sæt ouput ben lavt

LoginTimer

Ansvar: At holde styr på tiden siden sidste validerede login med den interne timer0 i atMega32 processoren.

Dette er en global klasse da ISR for timer0 overflows tilgår objektet. Det globale objekt hedder loginTimer.

Attributer:

- **unsigned int ovfCount_**
Tæller til at holde antallet af overflows

Metoder:

`Timer();` (Constructor)

Parametre: Ingen

Returværdi: Ingen

Beskrivelse: Initialiser ovfCount_ attributten til 0 og kald stop() metoden

`~ZeroCrossInt();` (Destructor)

Parametre: Ingen

Returværdi: Ingen

Beskrivelse: Kald stop() metoden

`void start();`

Parametre: Ingen

Returværdi: Ingen

Beskrivelse: Start timer0 til længst mulige tidstagning og aktiver overflow interrupts samt det globale interrupt flag

`void stop();`

Parametre: Ingen

Returværdi: Ingen

Beskrivelse: Stop timer0 og deaktiver overflow interrupts

`void inc();`

Parametre: Ingen

Returværdi: Ingen

Beskrivelse: Tæl ovfCount_ 1 op. Hvis den rammer overflow grænsen, som afgør tiden man er logget ind kaldes setLoginStatus() metoden i det globaleDE2IFObj objekt og ovfCount_ nulstilles

`ISR(TIMER0_OVF_vect);`

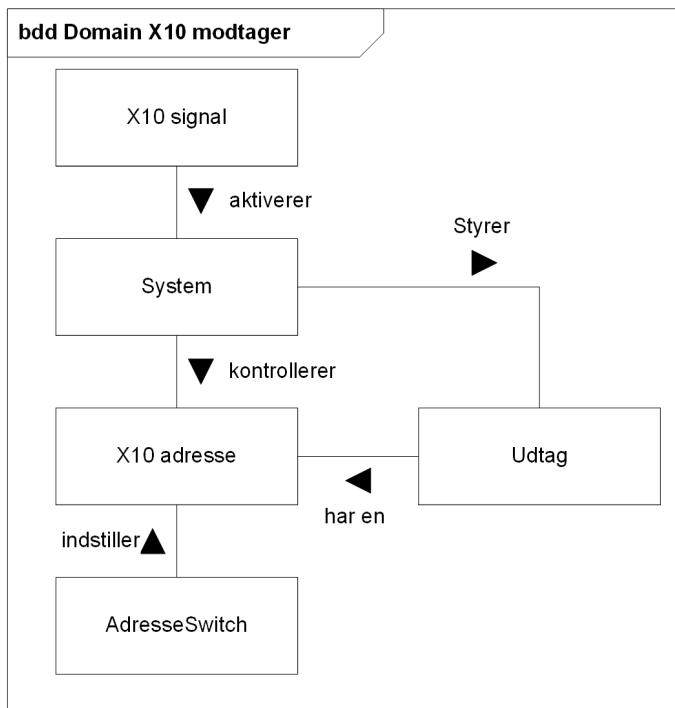
Parametre: Adresse til timer0 overflow vectoren

Returværdi: Ingen

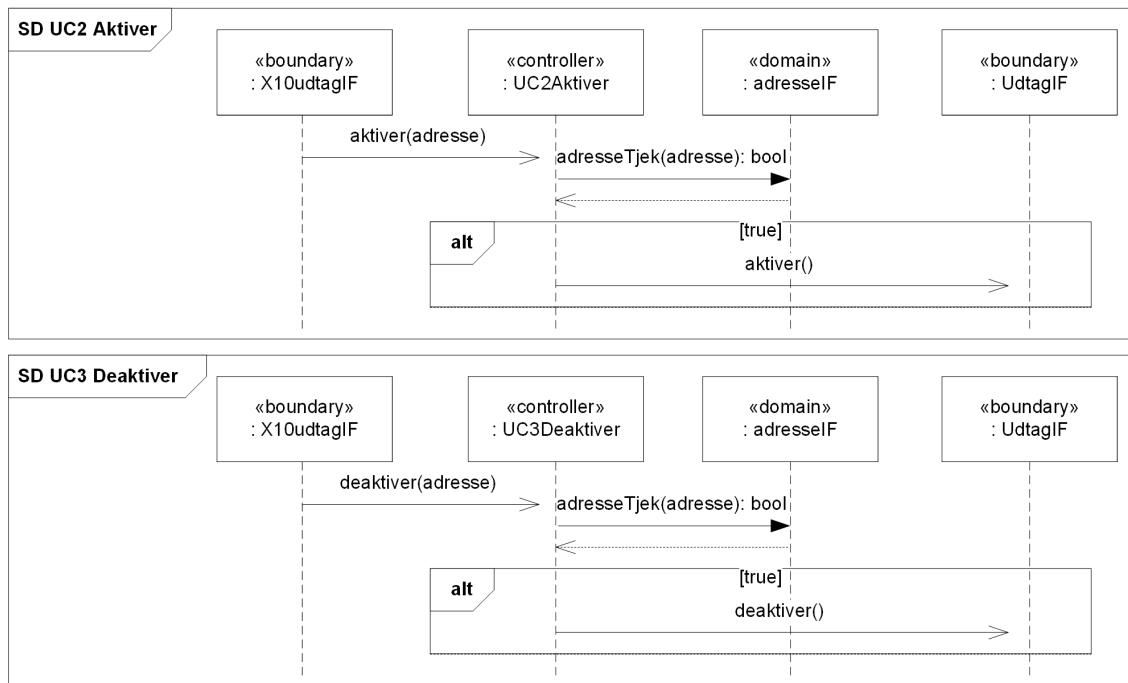
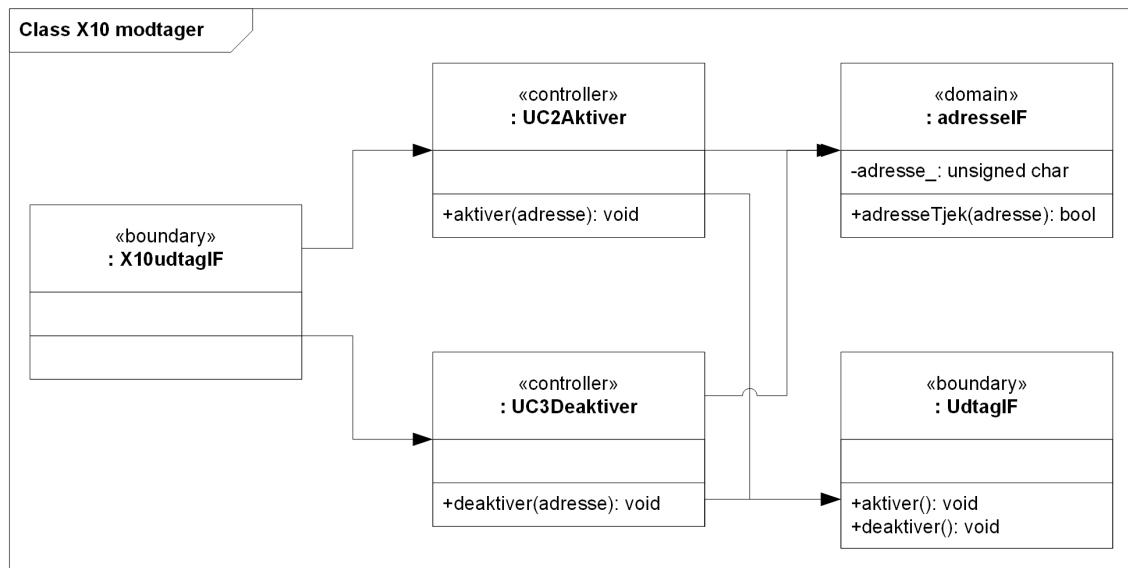
Beskrivelse: Kald inc() metoden i loginTimer objektet

4.3.5 Applications model for X10 modtager

For X10 modtageren er der udviklet en række diagrammer ud fra applikationsmodel metoden. På figur 4.15 er domænemodellen som resten af diagrammerne er udviklet ud fra. Der er et sekvensdiagram på figur 4.16 for alle de aktuelle use-cases som beskriver systemets virkemåde. Ud fra dette er der lavet et klassediagram på figur 4.17 som dækker de forskellige use-cases med controller klasser og kommunikationen til CSS hovedenheden via X10.



Figur 4.15. Domænemodel for X10 modtager

*Figur 4.16.* Use-case sekvensdiagrammer for X10 modtager*Figur 4.17.* Klassediagram for X10 modtager

4.3.6 Klassebeskrivels for X10 modtager

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

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
STX	'S' / 's'	0x53 / 0x73
ETX	'\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.

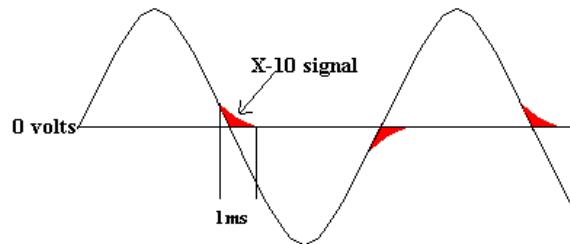
Hardware design

5

Generelt kan det overordnede hardware design beskrives som noget elektronik der sender information ud på nettet, som sendes og analyseres af noget elektronik i den anden ende. Disse vil blive beskrevet herunder som encoder og decoder.

5.1 Encoder

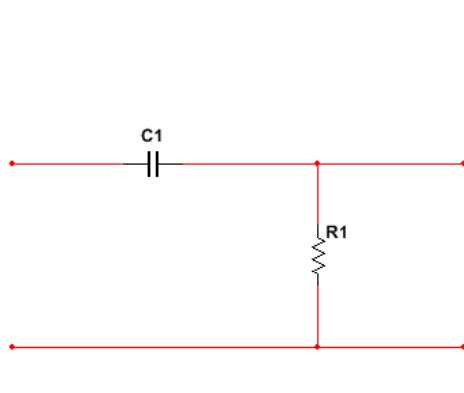
Encoderen, også omtalt som senderenhed, er den del af systemet der genererer X10 kommandoer og sender den ud på det eksisterende el-net. Et højpasfilter der lader 120 kHz burst passere mens det blokerer for nettets 50 Hz signal, udgør sammen med en zero crossing detector hele hardwaren for encoderen.



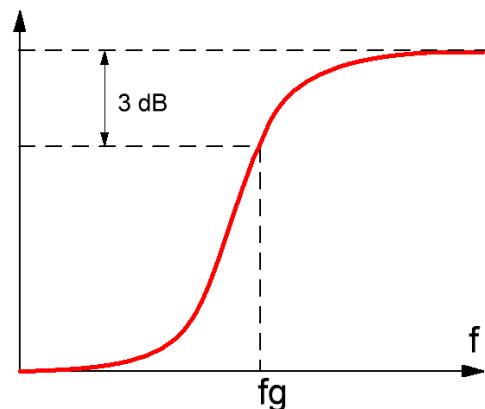
Figur 5.1. 120 kHz burst i zero crossing

Ideen med at sende burst ud på nettet i zero crossing kan ses illustreret på figur 5.1

5.1.1 Højpasfilter



Figur 5.2. Højpasfilter uden værdier



Figur 5.3. Kurvekarakteristik for højpasfilter

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.

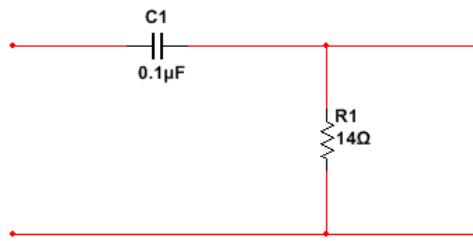
Den ønskede knækfrekvens skal ligge omkring de 120 kHz for at opnå mindst dæmpning herpå. Ved denne knækfrekvens skulle 50 Hz signalet være ubetydelig lille efter filteret.

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

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

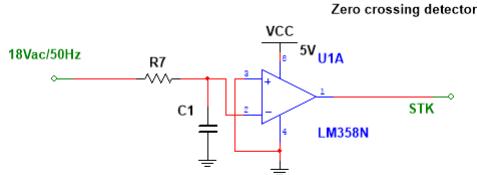
R_1 vælges da til 14Ω

Med den beregnede modstands værdi er det endelige schematic færdigt som det ses på figur 5.4

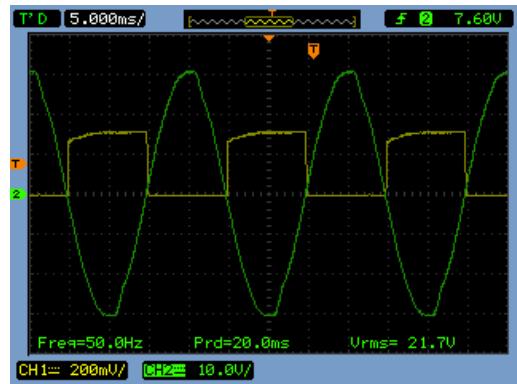


Figur 5.4. Højpasfilter med værdier

5.1.2 Zero Crossing Detector



Figur 5.5. Zero crossing detector uden værdier



Figur 5.6. 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 5.5, vi har anvendt en operationsforstærker af type LM358N, som toggler udgangssignalet ved hver nulgennemgang se figur 5.6. 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.

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

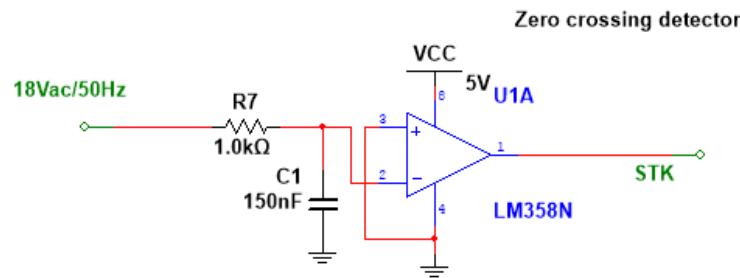
Der vælges en kondensator med en værdi på 150 nF. Med denne værdig kan modstanden i lavpasfilteret beregnes.

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 k Ω

Med alle de beregnede komponentværdier kan det endelige schematic designes. Det endelige design er vist på figur 5.7



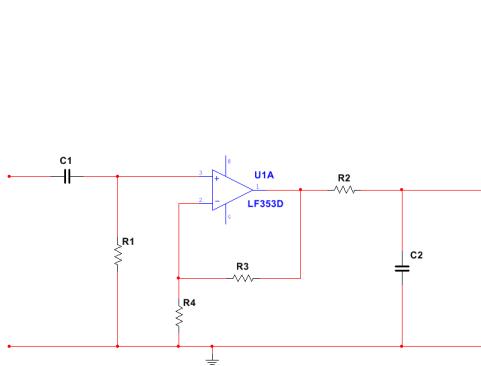
Figur 5.7. Zero crossing med værdier

5.2 Decoder

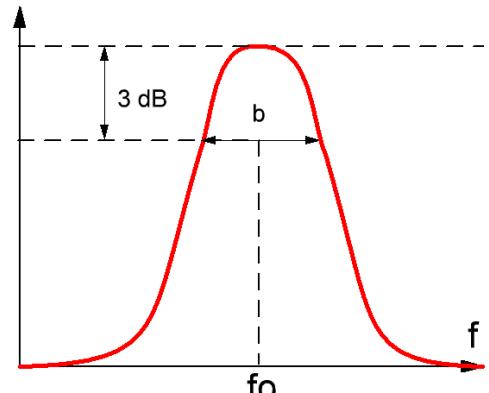
Decoderen, som er den del i systemet der omdanner burst fra encoderen til X10 kommando, er opbygget af et båndpasfilter, zero crossing detector og en envelope detector. I starten af kredsløbet sidder båndpasfilteret, og dette blokerer for 50 Hz nettet og forstærker 120 kHz signalet der kommer fra encoderen. 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 5.8. Båndpasfilter uden værdier



Figur 5.9. Kurvekarakteristik for båndpasfilter

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 5.8. Kurvekarakteristikken er illustreret på figur 5.9.

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 k Ω

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 k Ω

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:

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

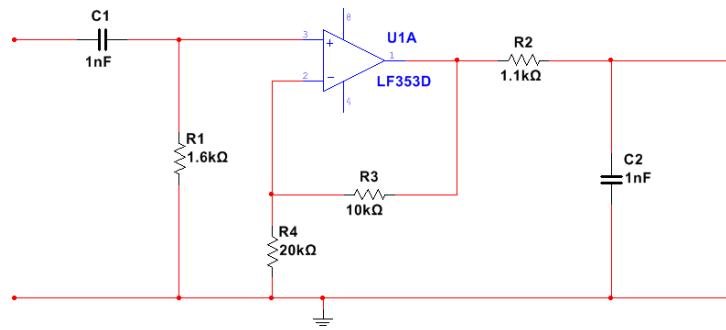
Ved at vælge R_4 til $10\text{ k}\Omega$ kan forstærkningen nemt reguleres med R_3 . Ved målinger på indgangssignalet før operationsforstærkeren kan vi fastslå indgangsamplituden til at være $8,0\text{ V}$. Da udgangssignalet ledes gennem en diode der fjerner de negative halvperioder vil amplituden blive 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 20000Ω så signalet derved bliver forstærket 3 gange.

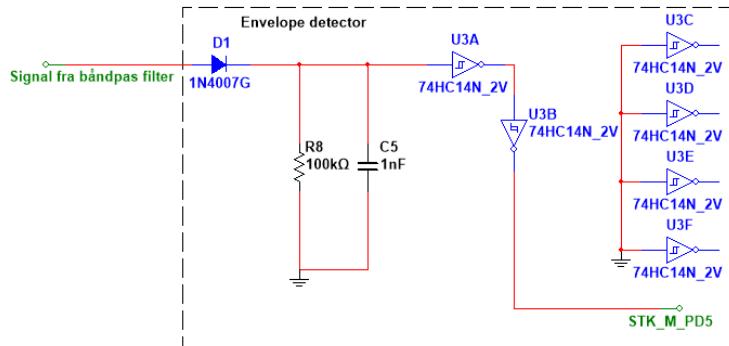
Der er som nævnt valgt en LF353 OpAmp. Denne er valgt ud fra at den har en bred båndbredde, en hurtig sætte tid og lav støj på indgangene.

Med alle de beregnede komponentværdier kan det endelige schematic designes. Det endelige design er vist på figur 5.10



Figur 5.10. Båndpas med værdier

5.2.2 Envelope Detector



Figur 5.11. Envelope detector

Envelope detectorens opgave er at udglatte burstsignalet fra båndpasfilteret og lave det om til et TTL (0-5V) signal som STK-kitten kan aflæse.

Den er opbygget af en diode, et RC-led og en schmitt trigger. Dioden har til opgave at sortere alle de negative halvperioder fra og kun sende de positive halvperioder til RC-leddet. Kondensatoren vil derfor kun opfange de positive halvperioder, og undgå at belaste det foranliggende kredsløb ved at aflade de negative halvperioder. Kondensatoren er med til at udglatte signalet da den ikke kan nå at aflades på en periode.

Modstanden R_8 er en afladningsmodstand, og den bestemmer hvor hurtigt kredsløbet skal aflades, jo højere modstand jo langsommere går det med at aflade. Vi har beregnet modstanden på følgende måde.

Tidskonstanten beregnes til.

$$\tau = 120000^{-1} Hz = 8,33 \cdot 10^{-6} s \quad (5.9)$$

Kondensatoren bestemmes til at være 1 nF, og modstand R_8 kan derfor beregnes ud fra denne formel.

$$\tau = R \cdot C \quad (5.10)$$

$$R = \frac{\tau}{C} = \frac{8,33 \cdot 10^{-6}}{1 \cdot 10^{-9}} = 83k\Omega \quad (5.11)$$

Modstanden R_8 er rundet op til en modstandstørrelse på 100kΩ, dette gør at vi er helt sikre på at kondensatoren ikke aflader for hurtigt.

Der er anvendt en schmitt trigger til at lave signalet om til et firkantet signal som STK-kitten kan aflæses. Grunden til at der er brugt 2 er fordi de er inverterende.

5.3 DE2-kodelås

DE2 Boardet bliver brugt som kodelås til CSS Hovedenheden. DE2 Boardet er programmeret i E2DSD Exercise 7 [HENVISNING TIL KODE I BILAG] som en

Statemachine. Der er dog blevet ændret i koden fra øvelsen, da der kun er brug for et enkelt bit til CSS Hovedenheden og ikke en konstant høj.

```
end if;
if present_state = s_unlocked then
    lock_counter <= lock_counter + 1;
else
    lock_counter <= "0000000000000000";
end if;
end if.
```

Figur 5.12. Lock Counter

Figur 5.12 viser en unsigned ved navn lock_counter, som ligger i state_reg processen. Den tæller op, når programmet befinner sig i s_unlocked staten.

```
when s_unlocked => -- UNLOCKED
next_err_cnt <= err_cnt;
if lock_counter < "1100001101010000" then -- 1ms
    next_state <= s_unlocked;
else
    next_err_cnt <= "0000";
    next_state <= s_idle;
end if;
```

Figur 5.13. s_unlocked

Figur 5.13 viser koden, der fortæller, hvor længe man skal befinde sig i s_unlocked staten, som sætter LOCK outputtet høj. I dette tilfælde er der valgt 1 ms. Efter 1 ms vil programmet nulstille error-counteren og gå til s_idle state - klar til nyt input.

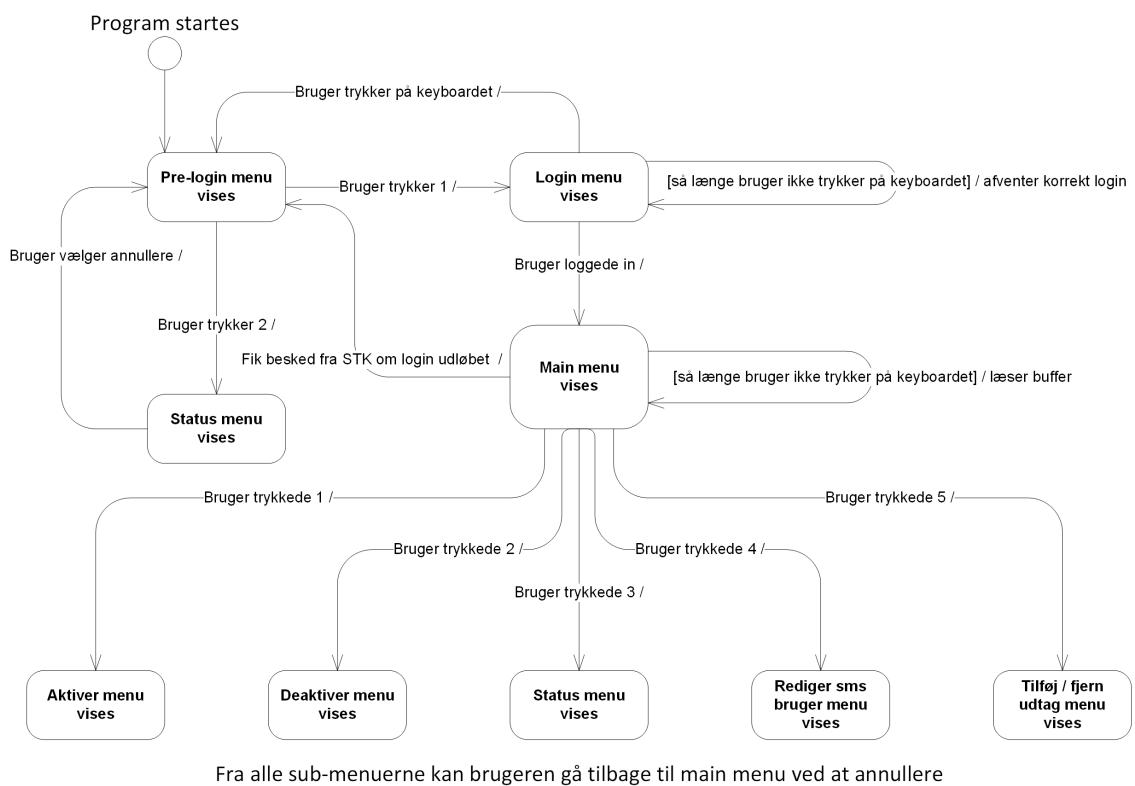
Outputtet er sat til GPIO1 JP2 D9 (PIN_M25), vist i Figur 5.14.



Figur 5.14. DE2-kodelås pinout

Software design 6

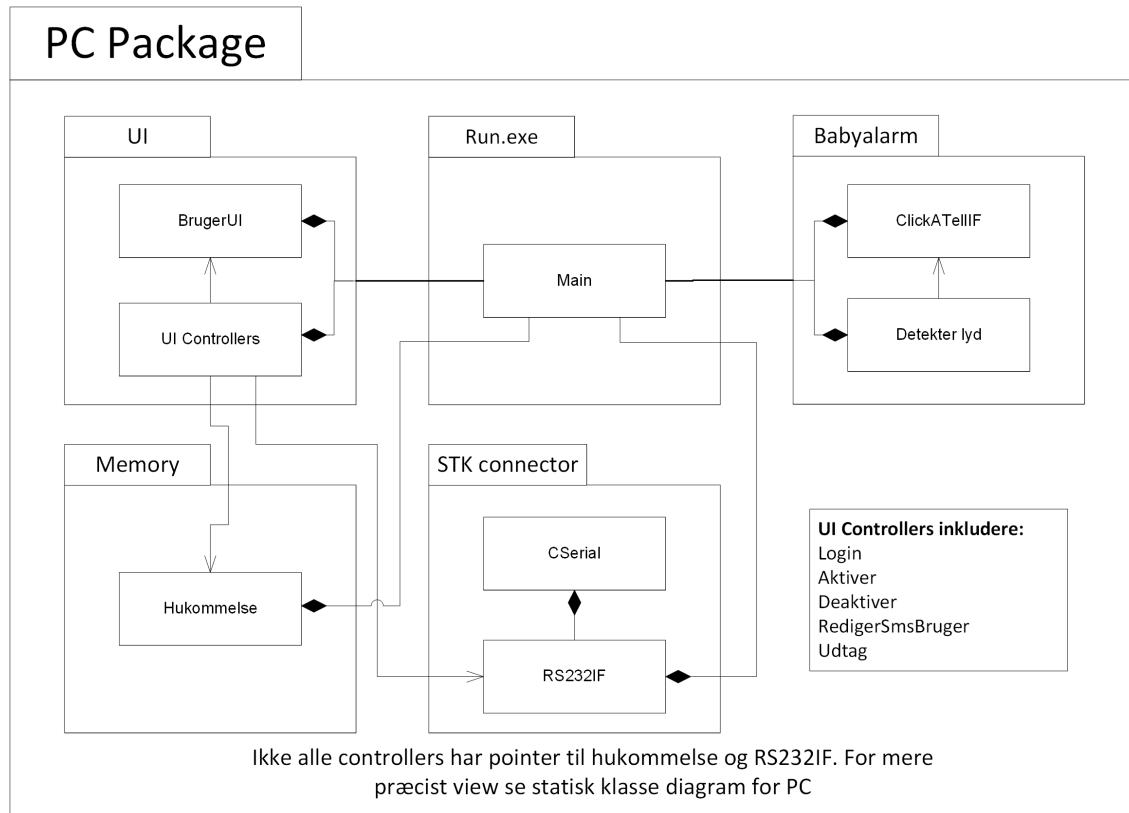
6.1 Logical View



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

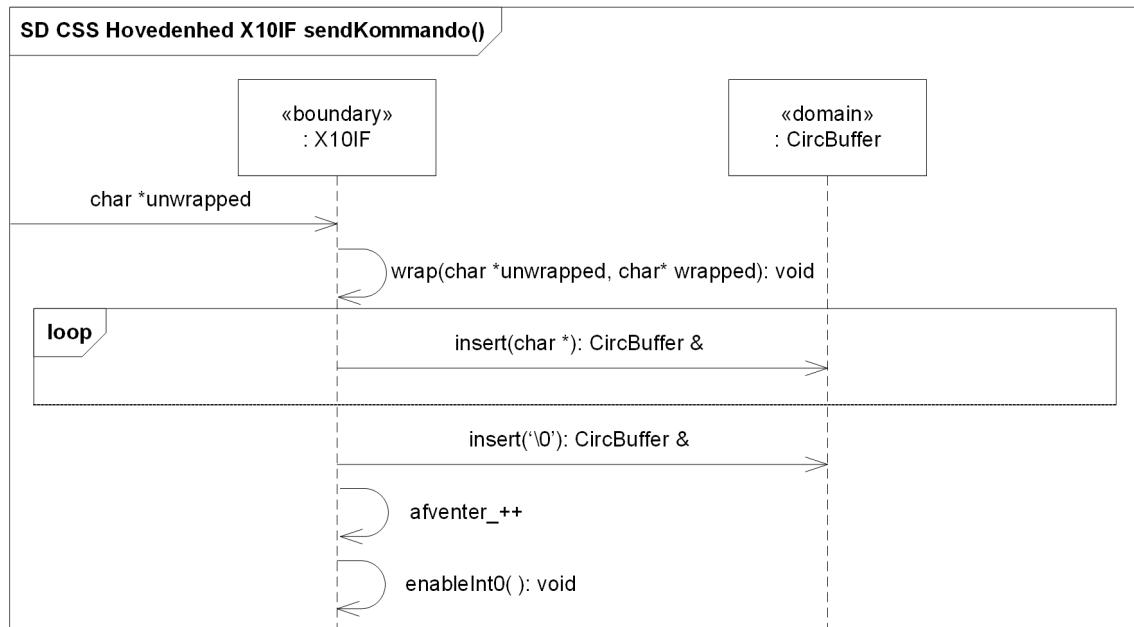
En af de kritiske og avancerede klasser er X10IF klasserne på hhv. CSS Hovedenheden og X10 Udtaget. I designfasen er der udviklet sekvensdiagrammer som beskriver nogle af metoderne og sammenspillet til andre klasser.

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

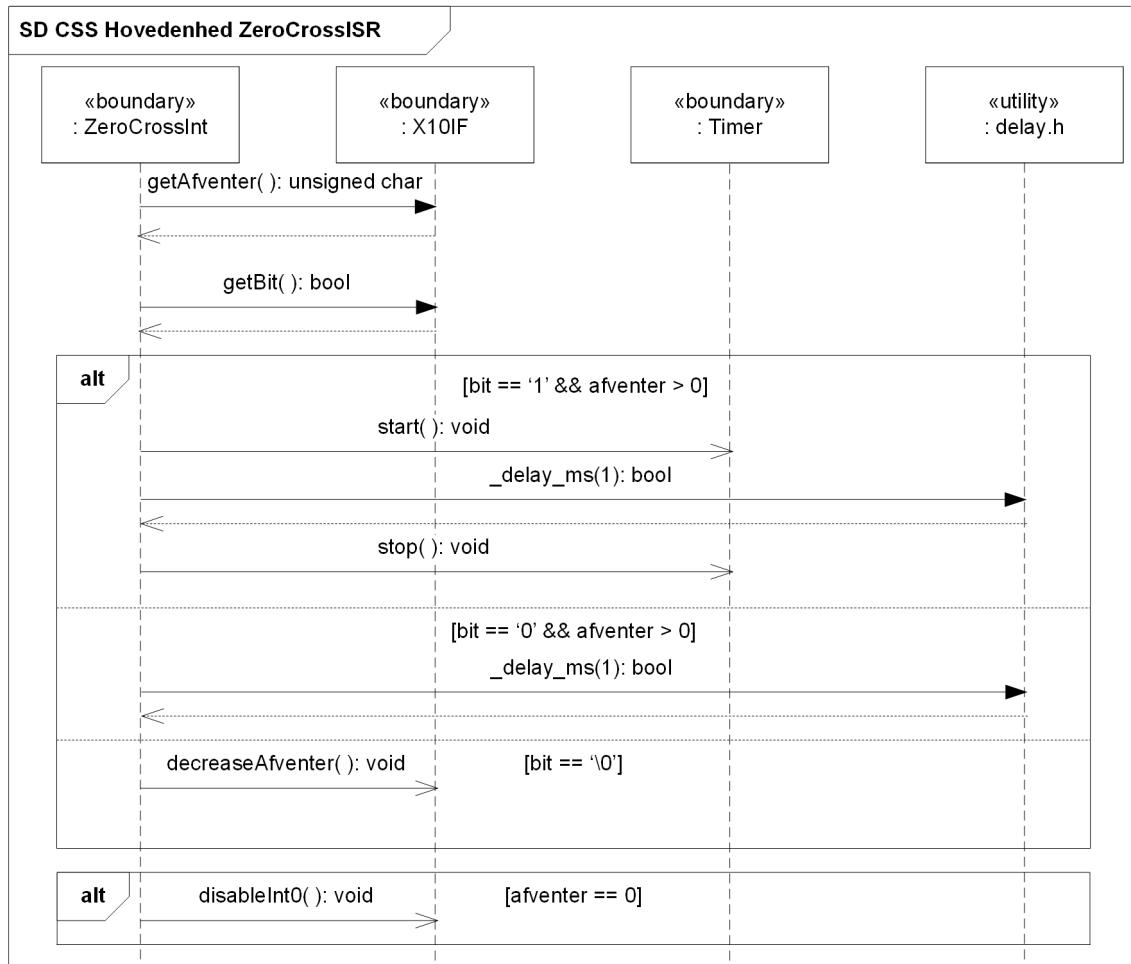
Metoden skal først omskrive den modtagende kommando til en X10 bit streng. Her efter indsættes alle bitsende i en buffer, hvorfra de afsendes når der detekteres et zerocross. Sekvensen er vist i figur 6.3.



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

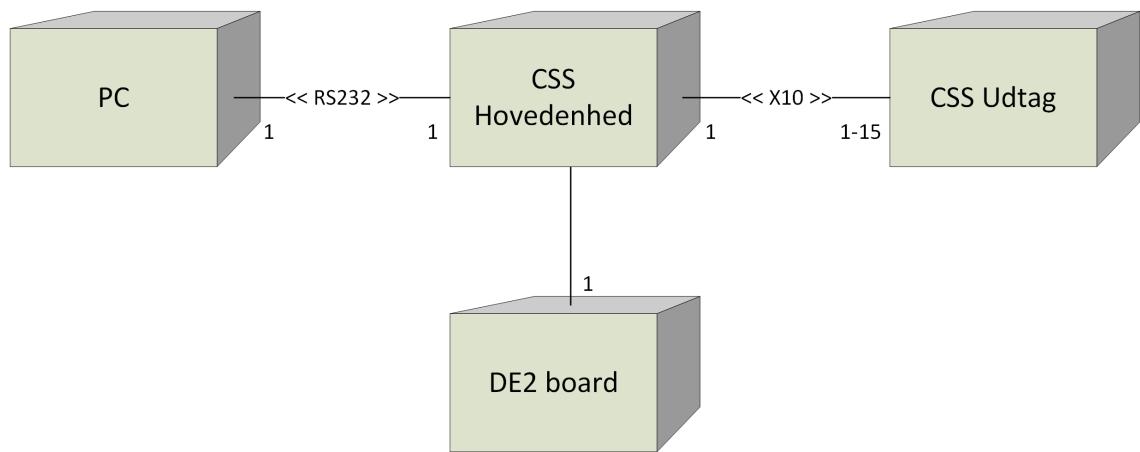
6.1.3 ZeroCrossInt funktion

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



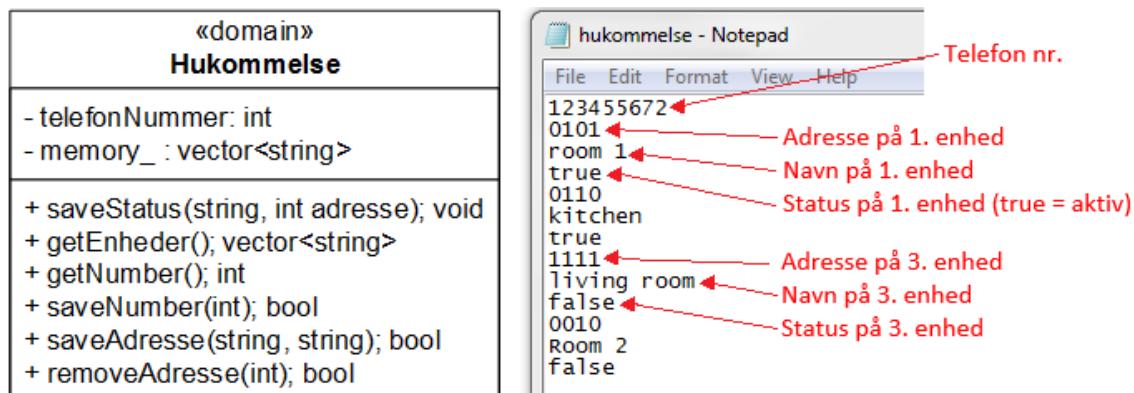
Figur 6.5. Deployment View

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

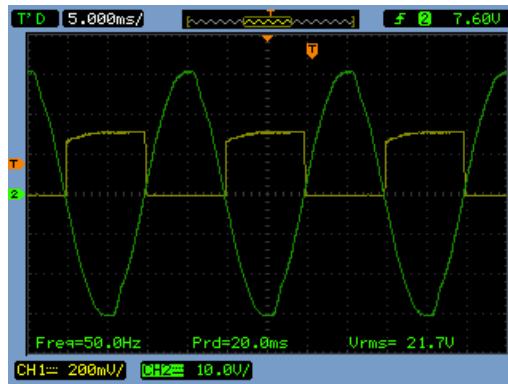
Modultest 7

7.1 Hardware Modultest

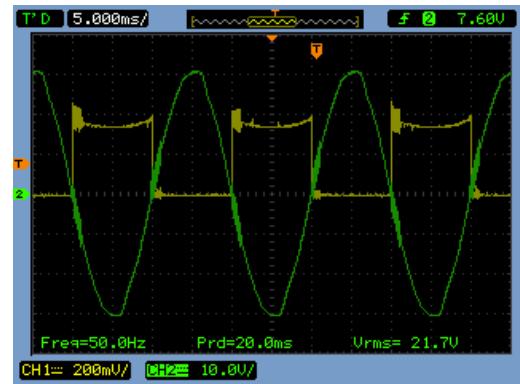
7.1.1 Encoder

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



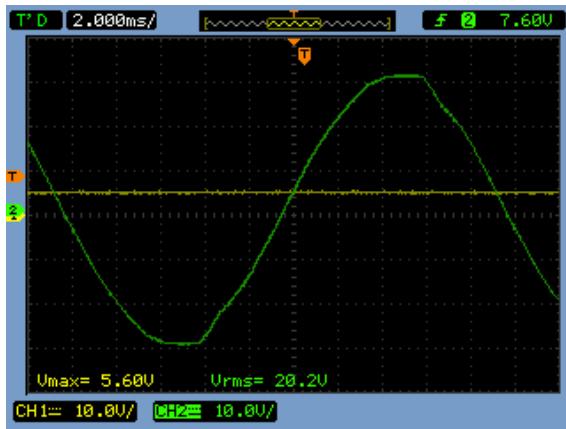
Figur 7.1. Zero Cross detector



Figur 7.2. Zero Cross detector med burst

Ud fra figur 7.1 ses at Zero Crossing detectoren virker som ønsket. Hver gang 18 Vac/50Hz passerer 0, toggles Zero Cross signalet. På figur 7.2 ses 18Vac/50Hz signalet med burst for hver zero cross. Heraf ses at det er netop ved zero cross at 120kHz burstet placeres på 50Hz signalet.

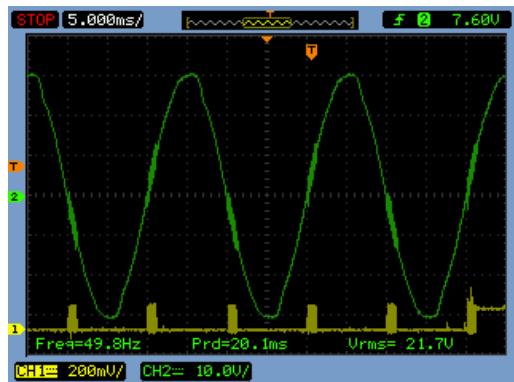
Højpasfilter



Figur 7.3. CH1:Efter højpasfilteret fra 18Vac/50Hz siden. CH2:18Vac/50Hz

Figur 7.3 viser begge sider af højpasfilteret. CH1 viser at 18Vac/50Hz signalet er filtreret fra. CH2 viser 18Vac/50Hz signalet fra forsyningsnettet.

Output



Figur 7.4. CH1: 120kHz burst. CH2: 18Vac/50Hz med 120kHz burst



Figur 7.5. Forstørring af 7.4. Burst på 1ms på 18Vac/50Hz

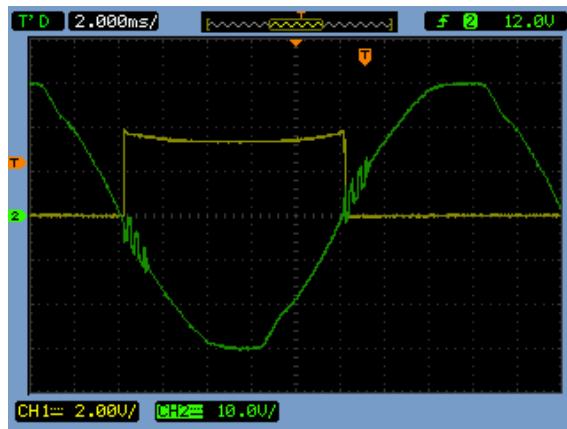
Herover ses figur 7.4. CH1 viser 120kHz burstet som sendes ud på det eksisterende forsyningsnet. CH2 viser outputet fra encoderen med 120kHz burstet. Det er dette signal der sendes ud på elnettet. Det er nu decoderens opgave at afkode dette vha X10 protokollen.

Til højre for, ses figur 7.5. Dette oscilloscop billede viser at burstet er på præcis 1ms.

7.2 Decoder

Decoderen er testet uden STK-500 kittet. Det vil sige der er testet uden software på modtager delen. Testprogrammet på hovedenheden er det samme som modultesten for encoderen. Et burst på 120kHz sendes ud for hvert zero cross.

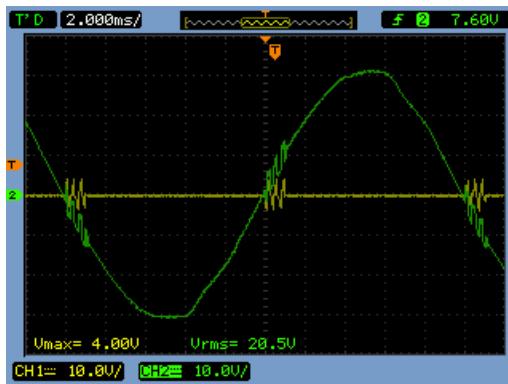
7.2.1 Zero Crossing



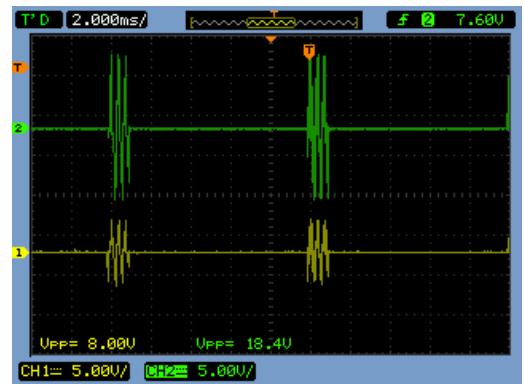
Figur 7.6. CH1: Toggle for hvert zero cross. CH2: 18Vac/50Hz med burst for hver zero cross

Herover ses figur 7.6. CH1 viser zero crossing detektorens firkantsignal der genereres af operationsforstærkeren når 18Vac/50Hz er negativ laver operationsforstærkeren en positiv firkant puls. CH2 viser 18Vac/50Hz signalet med 120kHz burst for hvert zero cross.

7.2.2 Båndpasfilter



Figur 7.7. Højpasfilterdelen af bandpasfilteret

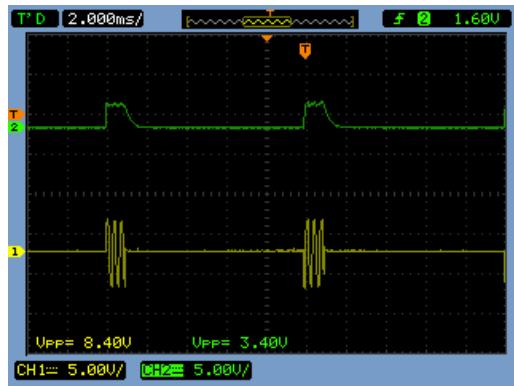
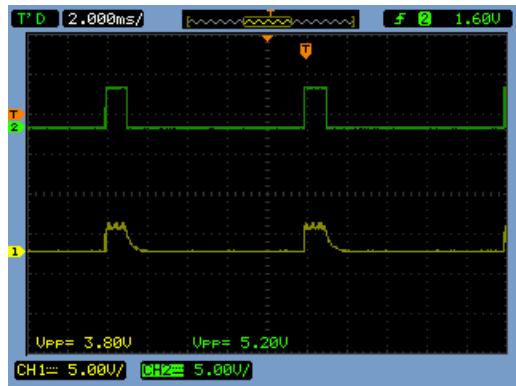


Figur 7.8. Forstærkningen i bandpasfilteret

Båndpasfilteret består af et højpasfilter, en operationsforstærker samt et lavpasfilter. Oscilloscop billede på figur 7.7 viser på CH1 viser højpasfilterets funktion. Højpasfilteret filtrerer de 18Vac/50Hz fra og lader kun 120kHz signalet passere. CH2 viser inputtet fra forsyningsnettet 18Vac/50Hz med 120kHz burst ved hvert zero cross.

Figur 7.8 viser forstærkningen i båndpasfilteret. Teoretisk (XXXX Henvisning) skal der være dobbelt forstærkning. I praksis ses der at signalet (CH1) forstærkes fra 8V til 18,4V (CH2) vha den inveterende operationsforstærker.

Det efterfølgende lavpasfilter vil igen formindste signalet. Dette ses i figur 7.9 på CH1

**Figur 7.9.** Ensretning dioden**Figur 7.10.** Envelope

7.2.3 Envelope detector

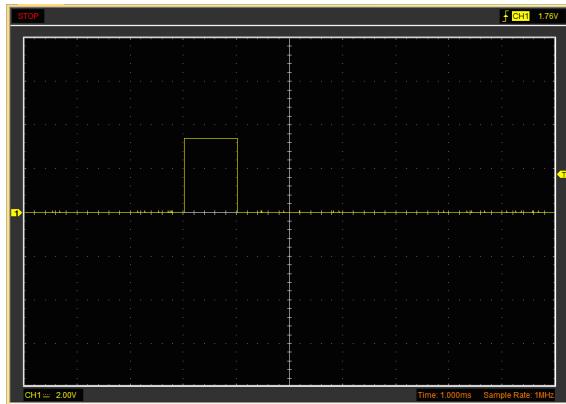
Diodens opgave er at ensrette signalet. Figur 7.9 viser signalet før (CH1) og efter (CH2) dioden. Toppene af 120kHz signalet sammesættes til en firkant puls. Denne firkant puls har et max på 3,4V. Figur 7.10 viser hvordan dette signal, signalet efter dioden (CH1), bliver rettet til et pænt firkantet TTL signal, som STK-500 kippet kan arbejde med.

7.2.4 Output

Outputtet af modtageren er et 18Vac/50Hz udtag. Dette udtag er enten aktiveret eller deaktiveret. Hvis udtaget ønskes aktivt skal softwaren fungere således at PD7 på modtagerens STK-500 kit sættes høj, herved aktiveret relæet og der er gennemgang for de 18Vac/50Hz fra forsyningsnettet. Figurene herover XX og XX viser at udtaget enten er aktiveret eller deaktiveret. Denne test er udført ved at aktivere relæet blot fra 5V forsyningen.

7.3 DE2-kodelås

DE2 1 ms bit målt med oscilloscope på DE2 GPIO1_D9 ift. stel. Med funktionen Trigger Single Sweep. Målingerne viser et 1 ms signal på 3.3Vpp, som ønsket.

**Figur 7.11.** 1 ms bit fra DE2-kodelås

7.4 SW Modultest

7.4.1 PC klasser

7.4.2 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 4.14.

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 forbinder 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 forbinder 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 binært på LEDerne på STK500 kittet og den sender den modtagende adresse retur. Testes med kommandoerne i tabel 7.1 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

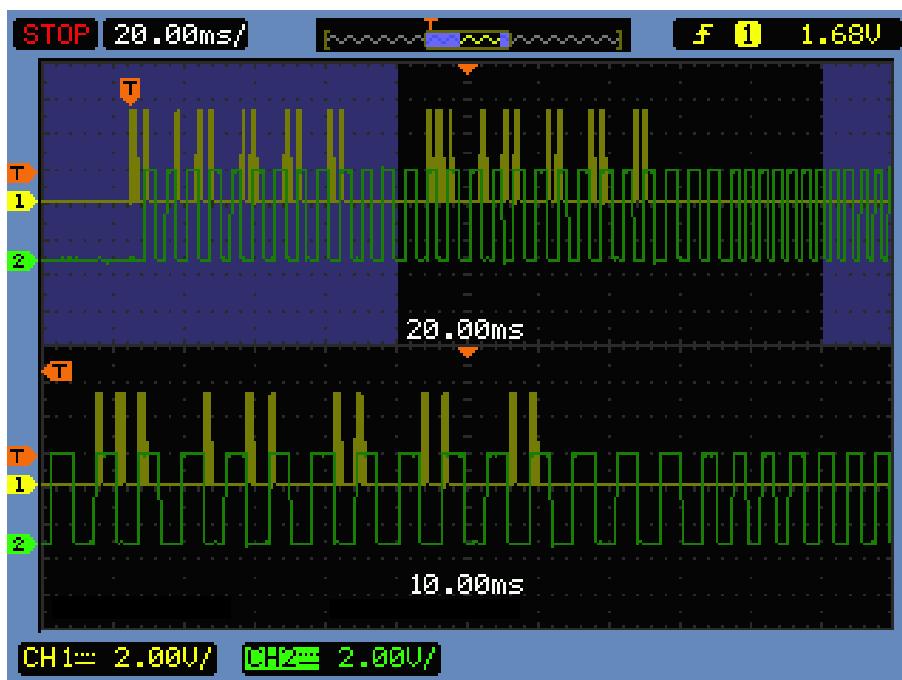
Test programmet ligges på et STK500 kit. Jumper forbinder PD2 og PB0. Et oscilloscope forbindes med to prober til PB2 og PD5. Reset STK500 og programmet starter.

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.

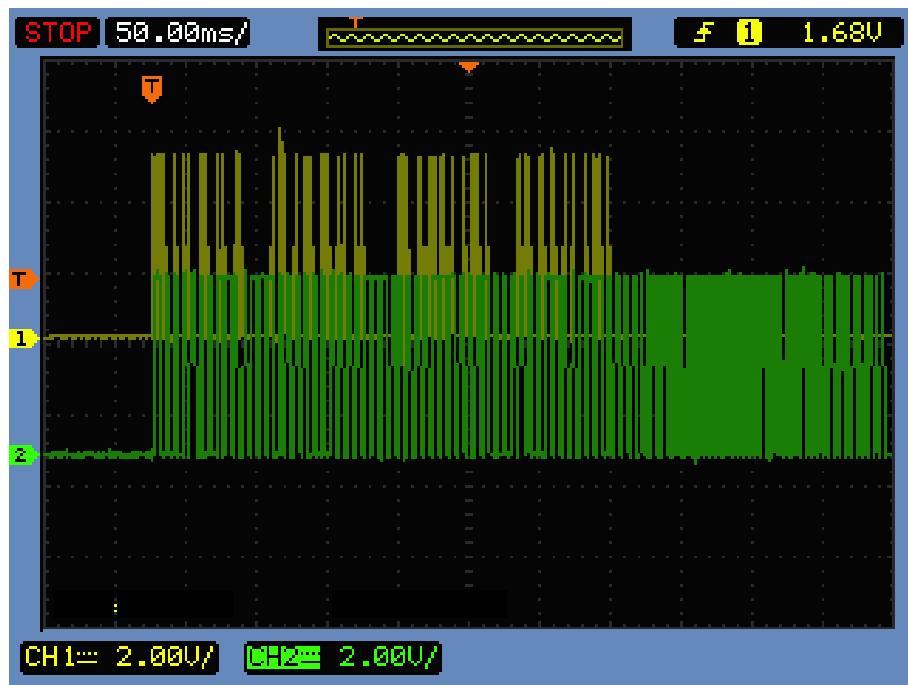
Med scopet kan så måles de rette bursts ud fra kommandoerne. På figur 7.12 kan man se de to kommandoer afsendt. Det gule signal er 120 kHz bursts svarende til X10 formaterede bits. Det grønne signal er det simulerede ZeroCross signal. I den øverste del kan man se aktiver kommandoen sendt to gange og i den nederste del er der zoomet ind på den sidste del. Dette stemmer overnes med protokol beskrivelsen.

På figur 7.13 kan man se både aktiver og deaktiver kommandoen. Bursts signalet skal være på 120 kHz.

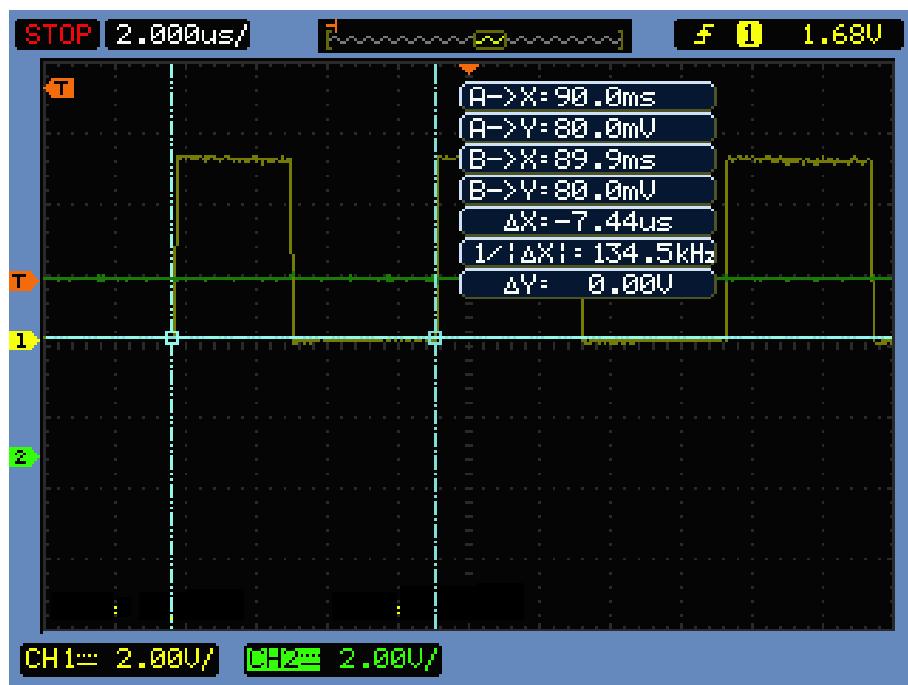
Figur 7.14 kan man se bursts signalet. Bemærk at frekvensen er lidt højere end forventet (134 kHz). Dette skyldes at den interne timer i STK500 kitter ikke kan komme 120 kHz nærmere. Reaktionstiden for programmet er målt i figur 7.15. Her kan det konstateres at fra der kommer et toggle på INT0 indgangen går der 32 us før burstet sendes afsted.



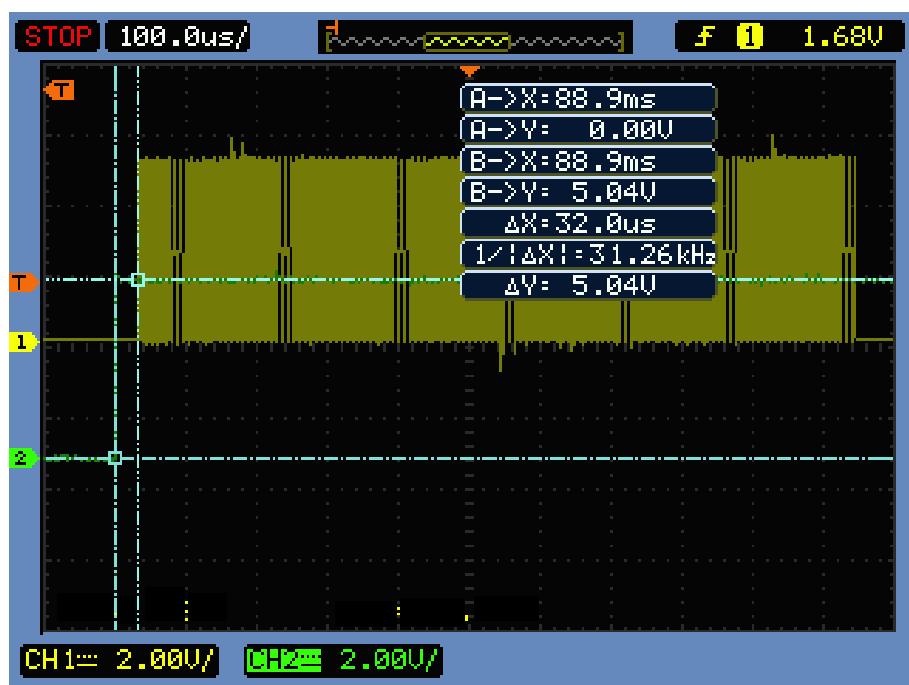
Figur 7.12. X10 Aktiver kommando. Oscilloscope måling, PB2 (Grøn) og PD5 (Gul)



Figur 7.13. X10 Aktiver og deaktiver kommandoer. Oscilloscope måling, PB2 (Grøn) og PD5 (Gul)



Figur 7.14. 120 kHz bursts. Oscilloscope måling, PB2 (Grøn) og PD5 (Gul)



Figur 7.15. Reaktionstid. Oscilloscope måling, PB2 (Grøn) og PD5 (Gul)

7.4.3 X10 Udtag klasser