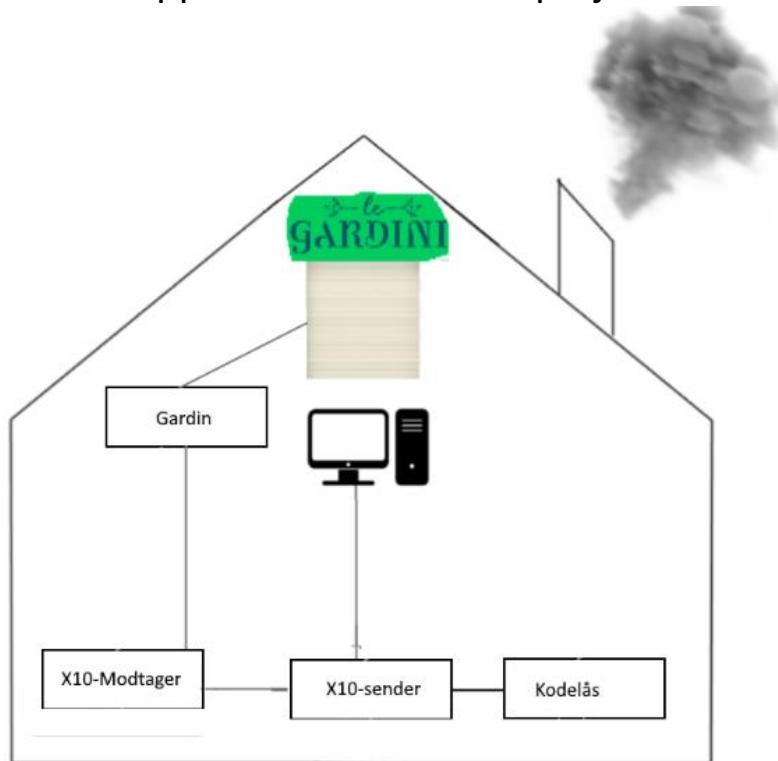


Dokumentation

Gruppe 3: PRJ2 semester projekt



Navn:	StudieNr:	Studie:
Anton Bjerg Brodkorb	201810587	E
Jacob Frost Hamann	201809587	IKT
Mads Damborg Rübner	201810590	E
Morten Yde Sloth Christensen	201810585	E
Morten Overgaard Kristensen	201510558	IKT
Rune Eriksen	201810578	E
Alexander V. Steen	201710792	IKT

Ordforklaring

Der vil i denne rapport forekomme forkortelser brugt for at undgå gentagelser for forklaring af protokoller, ord mm. Derfor er der her udarbejdet en ordforklaring hvori de gængse forkortelser vil blive præsenteret med en forklaring for disse.

Ord:	Forklaring:
TFS-LG	Tyveri forebyggelses system – Le Gardini
Le Gardini	Det afgrænsede produkt, som er vores gardin
X10	Der er her tale om den protokol der understøtter muligheden for at sende data på tværs af enheder over et elnet
X10-sender	Sender modulet der modtager kommandoer fra arduinoen og sender det ud på elnettet når der er et zerocross.
X10-modtager	Modtager modulet der modtager og omkoder signalet sendt på elnettet, til et digitalt højt eller lavt signal.
Kodelås	Der refereres her til et lille udsnit af knapperne på et Altera DE2 board
GUI	Brugergrænseflade som er gemt på PC
Computer	Også kaldet PC. Det er her på vores GUI befinder sig
Mikrocontroller	Arduino Mega 2560
Analog discovery	Måleinstrument og funktions generator, brugt ved hardware måling
Zero cross	Det punkt hvor en given funktion krydser x-aksen. I elektronikken er det oftest det punkt hvor spændingen er 0.

Arbejdsfordeling

Vi har i gruppen valgt at dele os op i mindre delgrupper. Disse grupper er bestemt til et møde i gruppen, hvor alle medlemmer har fået mulighed for at vælge hvad de gerne vil arbejde med. På følgende tabel fremgår arbejdsfordelingen

	Morten Yde Christensen	Alexander Steen	Morten Kristensen	Anton Brodkorb	Jacob Hamann	Rune Eriksen	Mads Rübner
X10 sender (hardware)	S			P		P	S
X10 modtager (hardware)	P			S		S	P
X10 sender (software)					P		
X10 modtager (software)			P				
GUI					P		
Kodelås			P		P		
Gardin		P					
Test:							
Modultest (hardware)	P			P		P	P
Modultest (software)			P		P		
Accepttest	P		P	P	P	P	P
Andet:							
Hardware Arkitektur	P			P		P	P
Software Arkitektur			P		P		
Hardware Design	P			P		P	P

Software Design			P		P		
Hardware Implementation	P			P		P	P
Software Implementation			P		P		
Integrations test	P		P	P	P	P	P
Accepttest	P		P	P	P	P	P
Diskussion				P		P	
Konklusion	P			S			
	Morten Yde Christensen	Alexander Steen	Morten Kristensen	Anton Brodkorb	Jacob Hamann	Rune Eriksen	Mads Rübner

Tabel 1 - Tabel over arbejdsfordeling - P står for Primær, og S står for sekundær

Resumé

I 2. semesterprojektet er formålet at lave et produkt der forebygger tyveri og indbrud. Et af kravene til projektet er, at produktet skal kommunikere via en X10 enhed, over elnettet. Desuden skal produktet også indeholde en kodelås og en computer. Til at opfylde kravene om et tyveriforbyggelsessystem, laves et gardin der kan indstilles til at rulle op og ned på bestemte intervaller, der skal simulere aktivitet i hjemmet. Systemet består af en kodelås, en computer, et X10-kredsløb, og 2 Mega2560 mikrocontrollere, til at sende og modtage funktioner over el-nettet. Systemet vil ikke køre over det rigtige el-net, men et simuleret el-net.

Til styring af gardinet, skal der både være manuel styring, og automatisk styring. For at indstille automatisk styring, kræves det at superbrugeren logger ind på kodelåsen. Manuel styring skal altid være aktiveret, og her skal blot laves 2 knapper i vores GUI, der ruller gardinet op og ned.

Til at strukturere projektet bruges viden fra faget ISE, og der bruges blandt andet vandfaltsmodellen, MoSCoW, SysML og UML.

Produktet testes ved en accepttest, som er udarbejdet ud fra krav der er opstillet af gruppen.

Abstract

The purpose of this 2nd term project is to create an anti-theft system, which is to automatically open and close curtains in the house on a set timer, to simulate activity at home. One of the requirements of the project is that the final product must communicate through the power-grid by using an x10 protocol. It will also be possible to manually open and close the curtains through the graphical user interface (GUI). The product must also have a code-lock and a computer connected. To unlock the automatic curtain control, the super-user must log in through the code-lock. The manual control will always be accessible, even if the super-user is not logged in. The system also consists of a X10 circuit, and 2 mega2560 microcontrollers, with the purpose of sending and receiving commands through the power grid.

To create the product, the ASE-model has been used. This includes the V-model, Waterfall-model, Sys-ML, and UML-diagrams. For system-limitations, the MoSCoW model has been utilized.

The final product is tested by creating an “accepttest”, that has been developed by the group

Indhold

Ordforklaring.....	2
Arbejdsfordeling.....	3
Resumé.....	5
Abstract.....	5
Indhold	6
1.0 Indledning	11
1.1 Motivation og kontekst	11
1.2 Projektformulering	11
1.3 Afgrænsning	12
2.0 Kravspecifikation.....	12
2.1 System Beskrivelse	13
2.2 GUI/brugergrænseflade	14
2.3 Funktionelle krav	16
2.3.1 Aktør beskrivelse	16
2.3.2 Use Case Diagram	18
2.3.3 Use Case beskrivelse.....	18
2.3.4 Fully Dressed Use Case beskrivelser.....	19
Use case 1: Login	19
Use case 2: Aktiver autostyring.....	20
Use case 3: Deaktiver autostyring.....	21
Use case 4: Rul op / ned	21
2.4 MoSCoW.....	22
3.0 Ikke-funktionelle krav – F(URPS).....	23
3.1 Motor.....	23
3.2 Kodelås / DE2.....	23
3.3 X-10.....	23
4.0 Accepttestspezifikation.....	24
4.1 Use case 1: Login	24

4.2 Use case 2: Aktiver autostyring.....	25
4.3 Use case 3: Deaktiver autostyring.....	26
4.4 Use case 4: Rul op/ned	26
4.5 Accepttest ikke-funktionelle krav.....	27
5.0 Metode og proces.....	30
5.1 Vandfaldsmodellen.....	30
5.2 V-modellen	31
5.3 Semesterprojektsmodellen	31
5.4 SysML.....	32
5.5 UML	32
6.0 Analyse	33
6.1 Kodelås	33
6.2 X10 Sender.....	33
6.3 Zero Cross Detector.....	33
6.4 X10 Modtager.....	34
6.5 Gardin	34
7.0 Arkitektur	34
7.1 Overordnet system arkitektur.....	35
7.2 Protokoller.....	35
7.2.1 X.10.....	36
En variation af X10.....	36
7.2.2 UART	37
7.3 Hardware arkitektur.....	38
7.3.1 IBD.....	38
7.3.2 System sekvens diagram.....	39
7.4 Blok beskrivelse	40
7.5 Signalbeskrivelse	42
7.6 Software arkitektur	44
7.6.1 Applikationsmodeller	44
7.6.2 Applikationsmodel for X10 Sender.....	45

7.6.3 Applikationsmodel X.10 Modtager.....	47
7.6.4 Applikationsmodel for Kodelås.....	48
7.6.5 Applikationsmodel for GUI	50
8.0 Hardware Design, Implementation og Modultest.....	52
8.1 X10 sender kredsløb.....	52
8.1.1 Højpas filter	53
8.1.1.1 Højpas filter Hardwaredesign.....	53
Analyse af sender kredsløb med applikationsnote værdier	53
Analyse af sender kredsløb med valgte værdier	57
8.1.1.2 Højpas filter Implementering	59
8.1.1.3 Højpas filter Modultest	60
8.1.2 120kHz generator	61
8.1.2.1 120kHz generator Hardwaredesign	62
8.1.2.2 120kHz generator Implementering.....	63
8.1.2.3 120kHz generator Modultest	64
8.1.3 Samlet sender kredsløb.....	65
8.2 Zero cross detektor	66
8.2.1 Zero Cross hardware design	66
8.2.2 Zero Cross Implementering	70
8.2.3 Zero Cross Modultest	70
8.3 X10 modtager kredsløb	72
8.3.1 Højpasfilter	74
8.3.1.1 Højpasfilter Hardware design.....	74
8.3.1.2 Højpasfilter Implementering	81
8.3.1.3 Højpasfilter Modultest	81
8.3.2 Operationsforstærker.....	84
8.3.2.1 Operationsforstærker Hardware design	84
8.3.2.2 Operationsforstærker Implementering.....	86
8.3.2.3 Operationsforstærker Modultest.....	86
8.3.3 Envelope Detector.....	87

8.3.3.1 Envelope Detector Hardware design	88
8.3.3.2 Envelope Detector Implementering.....	89
8.3.3.3 Envelope Detector Modultest.....	90
8.3.4 Schmitt-trigger.....	92
8.3.4.1 Schmitt-trigger Hardware Design.....	92
8.3.4.2 Schmitt-trigger Implementering	93
8.3.4.3 Schmitt-trigger Modultest.....	94
8.3.5 Samlet modtager kredsløb	95
8.4 Resultater hardware modultest	96
8.4.1 Opstilling hardwaretest.....	97
8.5 Samlet kredsløb	98
9.0 Software Design, implementering og modultest.....	100
9.1 Sender.....	100
9.1.1 Design	100
9.1.2 Implementering.....	101
9.1.3 Modultest	109
Modultest af forbindelse mellem GUI og sender.....	109
Modultest af korrekt indlæsning af bitstrenge i signal array	110
Modultest af bursts ved zerocrosses	111
9.2 Modtager.....	113
9.2.1 Design	113
9.2.2 Implementering.....	115
9.2.3 Modultest	120
9.3 GUI	124
9.3.1 Design	124
9.3.2 Implementering.....	126
9.3.3 Modultest	134
Modultest af kodelås til GUI.....	134
8.4 Kodelås	135
9.4 Resultater software modultest	137

10.0 Integrationstest.....	139
11.0 resultater.....	142
11.1 Accepttestresultater.....	142
11.1.1 Use case 1: Login	142
11.1.2 Use case 2: Aktiver autostyring.....	143
11.1.3 Use case 3: Deaktiver autostyring.....	143
11.1.4 Use case 4: Rul op / ned	144
11.1.6 Accepttest ikke-funktionelle krav.....	145
12.0 Diskussion	148
Projektproblematikker	148
Hardwareproblematikker	148
Softwareproblematikker	148
Procesproblematikker	149
13.0 Konklusion.....	149
13.1 Fremtidigt arbejde.....	150
Referenceliste	151



1.0 Indledning

1.1 Motivation og kontekst

Det samlede antal af ejendomsforbrydelser i Danmark er faldende, fra 115.000 til 73.000 indbrud årligt, det er et fald på 38% over 10 år.¹

Vi vil gerne gøre vores for at opretholde den nedadgående trend.

En ting der kan hjælpe med at afskrække indbrudstyve, er aktivitet i huset. Det kan f.eks. være lys, der tænder og slukker, samt gardiner der ruller op og ruller ned i bestemte tidsrum.

Derfor vil vi gerne implementere et system, hvor lys og gardiner kan styres automatisk, uden at man selv behøver at være hjemme.

Vores system skal bidrage til at sænke hyppigheden af indbrud, ved at afskrække indbrudstyve fordi de tror der er nogen hjemme. Desuden skal systemet også kunne bruges i dagligdagen, til nemmere at kunne styre hjemmets funktioner.

Vores vision for projektet er et home security system.

Idéen er at automatisere mange af hjemmets funktioner, så man automatisk kan tænde og slukke lyset, og ved hjælp af en motor, automatisk kan rulle gardiner op og ned.

Det skal hjælpe med at simulere aktivitet i hjemmet, for afskrække indbrudstyve.

Denne rapport omhandler en prototype af systemet, og ikke det fulde produkt. Der vil være idéer til at udvide systemet senere i rapporten

1.2 Projektformulering

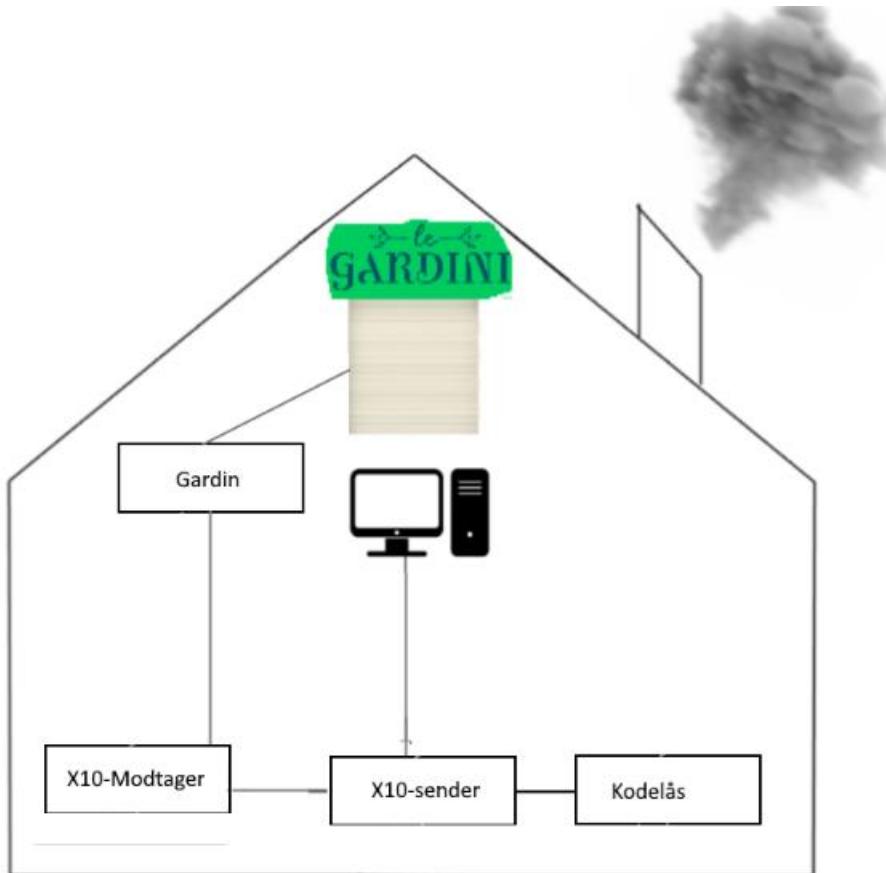
Målet for projektet er at lave en prototype til et system der kan rulle gardiner i hjemmet op og ned på en indstillet timer, eller manuelt. Systemet fungerer ved at være sat op med en computer, der skal fungere som Graphical User interface (GUI), hvor brugeren enten kan vælge manuel styring eller autostyring. Computeren er koblet op til X10 kredsløbet, som gennem el-nettet skal styre gardinerne. Desuden skal GUI'en låses ved med en kodelås, så autostyring ikke er tilgængelig for den almindelige bruger. Kodelåsen ligger på et DE2-board, hvor en 4 cifret kode skal indtastes, for at låse den fulde GUI op.

¹<https://www.dst.dk/da/Statistik/nyt/NytHtml?cid=27998&fbclid=lwAR19ZocHGnW967xyCi0Ac4QCuYVPKmY5K00JF6FMk1MK48qV1qfQ-pdeBP4>

1.3 Afgrænsning

I dette projekt vil der være fokus på et afgrænset produkt. Vi vil kun fokusere på styringen af et gardin, som skal kunne indstilles til at rulle op og rulle ned ved hjælp af en motor, i et bestemt tidsinterval. Dette system skal kunne indstilles fra en computer, hvor disse indstillinger og autostyringen kan låses op for med en kodelås. Oplysninger om aktiverings tidspunkter skal sendes igennem elnettet ved hjælp af en variant af X10 protokollen, så der ikke er brug for andre ledninger end dem der normalt er i hjemmet. Alt dette skal kunne indstilles ved hjælp af en brugergrænseflade på computeren.

2.0 Kravspecifikation



Figur 1 - System diagram



2.1 System Beskrivelse

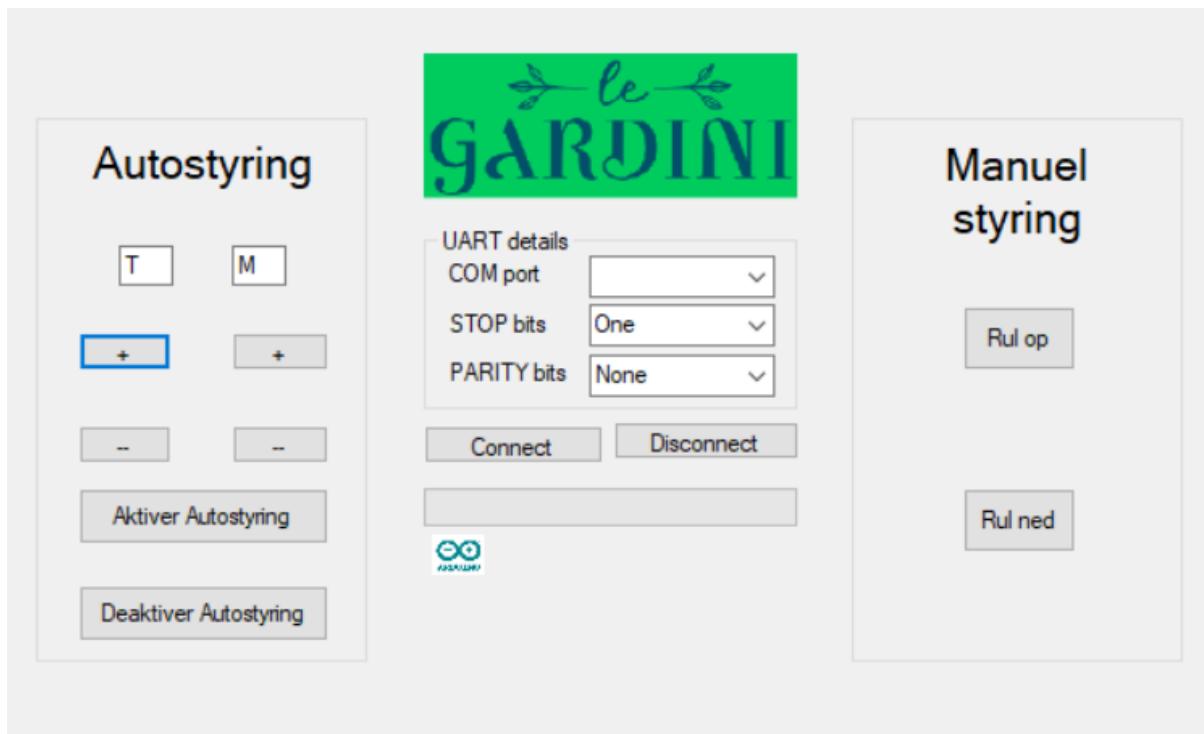
Systemet vil først og fremmest være opbygget omkring en X10 modtager, samt X10 sender. Derudover bruger vi også 2 mikrocontroller til at udføre en variant af X10 protokollen, så vi kan kommunikere over elnettet.

Selve X10 protokollen skal bruges til at kunne styre et rullegardin vha. en stepmotor. Gardinet skal kunne indstilles i et tidsinterval igennem en brugergrænseflade på en computer. Her vil der være mulighed for at indstille gardinet i timer og minutter, samt at styre systemet manuelt. Disse indstillinger vil blive gemt på computeren.

Systemets indstillingsmuligheder for autostyring låses op igennem en ekstern kodelås. Her skal kunden indtaste en 4 cifret kode for at låse op for brugergrænsefladen på computeren. Systemet vil have to typer brugere, Superbruger og bruger. Superbrugeren vil fungere som administrator i husholdningen og vil som den eneste kunne indstille på den automatiske intervalstyring. Brugeren vil kun kunne styre systemet manuelt, men vil ikke kunne tilgå automatisk intervalstyring.

2.2 GUI/brugergrænseflade

Nedst  ende ses brugergr  nsefladerne for henholdsvis GUI og kodel  sen.



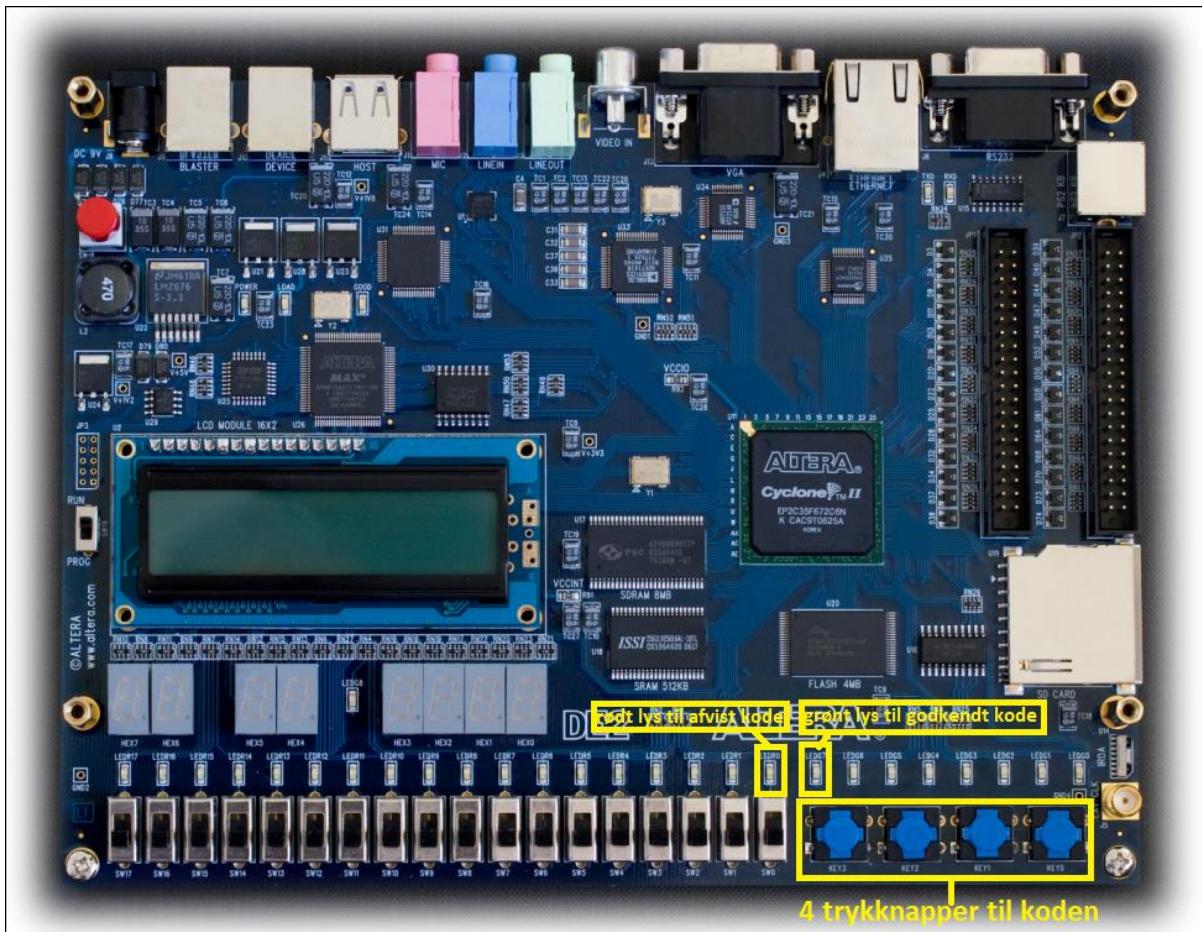
Figur 2 -GUI/ Brugergr  nseflade p   computer

P   figur 2 ses GUI-designet som vil fungere som brugerens kontrolpanel for burger og superbruger til systemet.

GUI'en skal i midten have en forbindelses menu til x10-senderen, s   bruger kan oprette forbindelse mellem GUI og resten af systemet. Bruger skal selv v  lge com-port, som programmet selv registrerer, og trykke connect. Hvis forbindelse er oprettet, vil en gr   progressbar under menuen blive gr  n.

Til h  jre skal der v  re en menu til manuel styring, de vil kunne rulle gardinet op eller ned, s   fremt gardinet ikke allerede er p   det   nskede stadie.

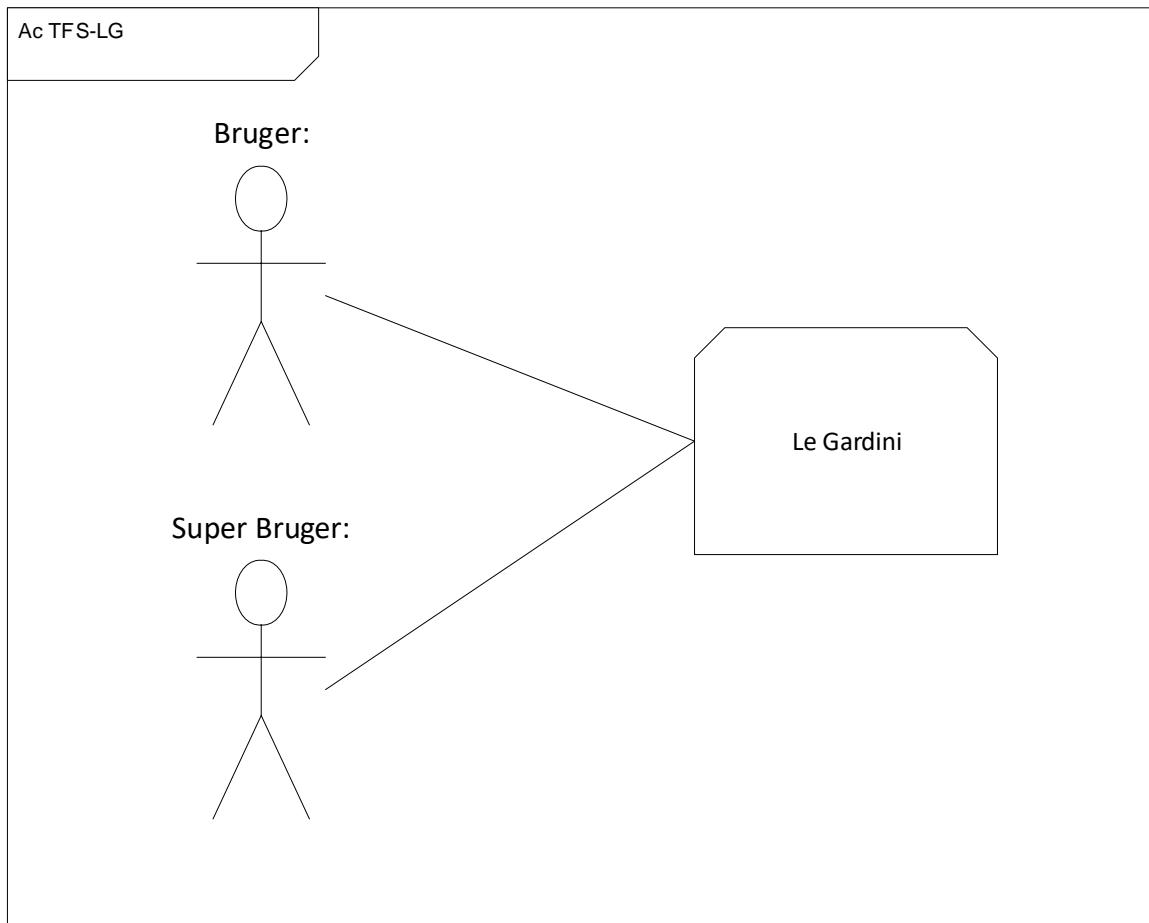
Til venstre skal der v  re en menu til autostyring der forekommer utilg  ngeligt uden korrekt indtastet kode fra den eksterne kodel  s. N  r den er l  st op, skal der kunne indstilles et tidsinterval i timer og minutter, og der skal v  lge knapper til at aktivere og deaktivere timeren. Den indtastede tid vil fremg   af "t" og "m" modulerne, og n  r timeren igangs  ttes vil de t  lle ned. N  r "t" rammer 0 samtidig med at "m" er 0, vil den via UART sende et signal for enten at rulle op eller ned til senderen, s  fremt bruger har oprettet forbindelse via forbindelsesmenuen.



Figur 3 - Brugergrænseflade af kodelåsen på DE2-board beskrevet med 4 trykknapper og grønt- og rødt-lys.

Kodelåsen skal have en hardcoded kode. Hvis koden godkendes vil en GPIO pin på boardet, sende et signal viderer til senderen.

2.3 Funktionelle krav



Figur 4 - Aktør kontekst diagram

2.3.1 Aktør beskrivelse

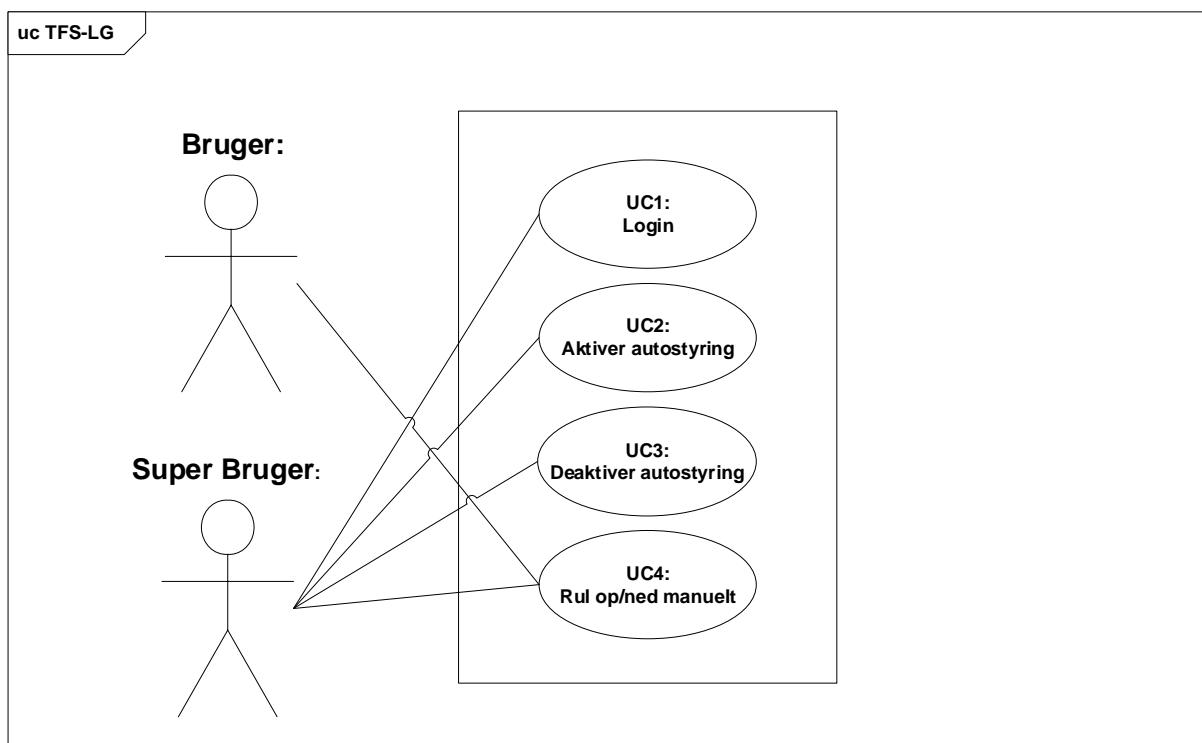
Aktør navn	Bruger
Type	Primær
Beskrivelse	Brugeren er alle der interagerer med systemet. Alle brugere har mulighed for at interagere med systemet, ved at rulle op og ned for gardinet uden at have fuldstændig kontrol over systemet.

Tabel 2 - Aktør beskrivelse for bruger

Aktør navn	Superbruger
Type	Primær
Beskrivelse	<p>Superbruger er en bruger der er i besiddelse af koden til kodelåsen.</p> <p>Denne brugertype har den fuldkomme kontrol over det samlede system. Det vil sige at ud over muligheden for op og nedrulning af gardiner, så har denne brugertype også adgang til aktivering og deaktivering af det automatisk system, samt at indstille det automatiske system.</p>

Tabel 3 - Aktør beskrivelse for superbruger

2.3.2 Use Case Diagram



Figur 5 - Use case diagram

2.3.3 Use Case beskrivelse

UC1 Login

Denne use case har til formål at superbrugeren kan indtaste en kode på kodelåsen, og derved låse op for aktivering og deaktivering af autostyring med superbruger privilegier.

UC2 Aktiver autostyring

Denne use case har til formål at superbrugeren skal kunne aktivere autostyring af systemet, ved at indstille det tidsinterval hvor gardinet skal rulle op og ned.

UC3 Deaktiver autostyring

Denne use case har til formål at superbruger skal kunne deaktivere systemet med en knap i brugergrænsefladen, så autostyringen ikke længere er aktiveret.

UC4 Rul op/ned

Denne use case har til formål at sørge for brugeren kan rulle gardinet op og ned uden for autosyrings rutinen. Dette kan gøres i et brugerinterface, uden superbruger privileger

2.3.4 Fully Dressed Use Case beskrivelser

Use case 1: Login

Navn:	Login
Version	1.0
Sidste opdatering	07-11-2019
Initiering	Brugeren skriver sin kode på kodelåsen.
Aktører	Bruger
Samtidige forekomster	Ingen
Prækondition	Brugeren er ikke logget ind.
Postkondition	Bruger er logget ind, og har nu Superbruger privilegier.
Hovedscenarie	<ol style="list-style-type: none"> 1. Brugeren skriver sin 4-cifret kode ind på kodelåsen. 2. System verifierer koden. [Udvidelse 1a: Forkert kode] 3. Grøn lampe stopper med at lyse 4. Systemet giver adgang til brugergrænsefladens timer indstillinger.
Udvidelser	<p>[Udvidelse 1a: Forkert kode]</p> <ol style="list-style-type: none"> 1. Rød lampe lyser.

	<ol style="list-style-type: none"> 2. Tidsintervals indstillingerne forbliver låst på brugergrænsefladen 3. Use case afsluttes
--	--

Tabel 4 - Use Case 1

Use case 2: Aktiver autostyring

Navn:	Aktiver autostyring
Version	1.4
Sidste opdatering	07-11-2019
Initiering	Superbrugeren benytter GUI'en
Aktører	Superbruger
Samtidige forekomster	Ingen
Prækondition	Superbrugeren har logget ind korrekt, og autostyring er deaktiveret.
Postkondition	Autostyringen er aktiveret, og ruller op/ned samt genstarter timer når intervallet er nået.
Hovedscenarie	<ol style="list-style-type: none"> 1. Superbrugeren indstiller tidsinterval for ændring af tilstand (Gardinet ruller ned hvis start tilstanden er "oppe", og ruller op hvis start tilstanden er "nede") 2. Brugeren trykker på knappen "aktiver" 3. Autostyring er aktiveret fra tidspunktet hvor knappen "aktiver" bliver trykket. 4. Når tidsintervallet er slut, rulles gardinet op/ned. Tidsintervallet starter forfra.

Tabel 5 - Use case 2

Use case 3: Deaktiver autostyring

Navn:	Deaktiver autostyring
Version	1.4
Sidste opdatering	7-11-2019
Initiering	Superbrugeren benytter GUI'en
Aktører	Superbruger
Samtidige forekomster	Ingen
Prækondition	Superbrugeren er logget ind korrekt, og autostyringen er aktiveret.
postkondition	Autostyringen er deaktiveret
Hovedscenarie	<ol style="list-style-type: none"> 1. Superbrugeren trykker på deaktiver autostyring 2. Autostyringen er deaktiveret. 3. Tidsintervallet nulstilles.

Tabel 6 - Use case 3

Use case 4: Rul op / ned

Navn:	Rul op/ned manuelt
Version	1.4
Sidste opdatering	7-11-2019
Initiering	Bruger eller superbruger trykker på "Rul op" eller "Rul ned"
Aktører	Bruger, Superbruger
Samtidige forekomster	Ingen
Prækondition	Brugerne eller superbrugeren har brugerfladen tilgængelig
postkondition	Rullegardinet ruller op eller ned
Hovedscenarie	<ol style="list-style-type: none"> 1. Bruger/superbruger åbner op for brugerfladen 2. Gennem brugerfladen vælger brugeren/superbrugeren at rulle gardinet op 3. Gardinet ruller op
Alternativ	<ol style="list-style-type: none"> 1. Bruger/superbrugeren åbner op for brugerfladen

	<ol style="list-style-type: none">2. Gennem brugerfladen vælger bruger/superbruger en at rulle gardinet ned3. Gardinet ruller ned
--	--

Tabel 7 - Use case 4

2.4 MoSCoW

Must have:

- Systemet **skal** have en brugergrænseflade, f.eks til tidsindstillinger.
- Systemet **skal** kunne deaktiveres og aktiveres ved en knap i brugergrænsefladen.
- Superbrugerens **skal** kunne indstille tidsintervaller for at rulle op og ned.
- Systemets autostyring **skal** låses op/i ved brug af en kodelås.

Should have:

- Systemet **bør** have mulighed for indstilling af længde af gardin.
- Systemet **bør** kunne styre mere end ét gardin

Could have:

- Systemet **kan** have mulighed for indstilling af hastighed.

Won't have:

- Systemet **vil ikke** kunne styre yderligere komponenter i huset.

3.0 Ikke-funktionelle krav – F(URPS)

Usability:

- U1: Der medfølger en brugermanual til systemet
- U2: Det skal være muligt at finde samtlige GUI funktioner i selvsamme manual

Reliability:

- R1: Systemet skal kunne udføre UC3 mindst 15 gange inden der skal skiftes dele.
- R2: Systemet skal kunne gennemføre UC1 mindst 15 gange inden der skal skiftes dele

Performance:

- P1: Systemet skal kunne rulle helt op og helt ned, til en afstand af 1 centimeter fra vinduets top eller bund.
- P2: Gardinet skal kunne rulle op med en hastighed på 7 cm / s

Supportability:

- S1: Systemet skal kunne repareres på maksimalt 1 time (MTTR).
- S2: Systemets fysiske dele bør kunne udskiftes på 30 minutter.

3.1 Motor

- M1: Gardinets motor opererer ved en gennemsnitseffekt på 10W

3.2 Kodelås / DE2

- K1: Ved indtastning af forkert kode 3 gange skal en rød LED lyse. (LEDR(13))
- K2: Ved rigtig kode skal en grøn LED stoppe med at lyse. (LEDG(0))
- K3: På kodelåsen skal KEY(0 ... 3) bruges til indtastning af koden.
- K4: Koden til kodelåsen skal være "1111".

3.3 X-10

- X1: Det simulerede elnet som X-10 opererer på skal have en Vrms på $18V \pm 4V$ AC $50Hz \pm 1 Hz$.

Ovenstående ikke funktionelle krav er henvendt til et færdigt produkt, og ikke den prototype vi laver. Derfor vil vi på prototypen altså ikke leve op til samtlige ikke funktionelle krav. Dette indikeres med ”**Testes ikke**” i ”accepttest ikke-funktionelle krav”.

4.0 Accepttestspezifikation

4.1 Use case 1: Login

Use case under test:	Login		
Scenarie:	Hovedscenarie		
Samtidige forekomster	Ingen		
Prækondition	Brugeren er ikke logget ind		
Postkondition	Brugeren er logget ind, og har nu Superbruger privilegier		
<hr/>			
Handling:	Forventet observation/resultat	Faktisk observation/resultat	Vurdering (OK / Fail)
Brugeren skriver sin 4-cifret kode ind på kodelåsen	Grøn lampe lyser i 5 sekunder, og brugergrænsefladen låser op for indstillinger for autostyring.		

Tabel 8 - Accepttest Use case 1

Use case under test:	Login
Scenarie:	Udvidelse 1a
Samtidige forekomster	Ingen
Prækondition	Brugeren er ikke logget ind
Postkondition	Brugeren er ikke logget ind
<hr/>	

Handling:	Forventet observation/resultat	Faktisk observation/resultat	Vurdering (OK / Fail)
Brugeren skriver en forkert kode ind på kodelåsen	Rød lampe lyser i 5 sekunder, og tidsinterval indstillingerne forbliver låst.		

Tabel 9 - Accepttest Use case 1. Udvigelse 1a

4.2 Use case 2: Aktiver autostyring

Use case under test:	Aktiver autostyring		
Scenarie:	Hovedscenarie		
Samtidige forekomster	Ingen		
Prækondition:	Brugeren har logget ind korrekt, og autostyring er deaktivert.		
Postkondition:	Systemet er aktiveret, og kører automatisk		
Handling:	Forventet observation/resultat	Faktisk observation/resultat	Vurdering (OK / Fail)
Superbrugeren indstiller intervallet for ændring af tilstand (rul op hvis gardinet er nede og rul ned hvis gardinet er oppe)	Autostyringen ændrer tilstand i det korrekte interval.		

Tabel 10 - Accepttest Use case 2

4.3 Use case 3: Deaktiver autostyring

Use case under test:	Deaktiver autostyring		
Scenarie:	Hovedscenarie		
Samtidige forekomster	ingen		
Prækondition	Brugeren har logget ind korrekt, og systemet er aktiveret		
Postkondition	Systemet er deaktiveret		
Handling:	Forventet observation/resultat	Faktisk observation/resultat	Vurdering (OK / Fail)
Superbrugeren trykker på "deaktiver autostyring" på brugergrænsefladen.	Autostyringen er deaktiveret		

Tabel 11 - Accepttest Use case 3

4.4 Use case 4: Rul op/ned

Use case under test:	Rul op/ned		
Scenarie:	Hovedscenarie		
Samtidige forekomster	Ingen		
Prækondition	Brugeren eller superbrugeren har adgang til brugerfladen		
Postkondition	Gardinet ruller op eller ned		
Handling:	Forventet observation/resultat	Faktisk observation/resultat	Vurdering (OK / Fail)
Bruger/superbruger en åbner op for brugergrænsefladen	Brugergrænsefladen åbner på computeren.		

Bruger/superbruger en trykker på knappen "rul op"	Gardinetrullen op		
Alternativ:			
Bruger/superbruger en åbner op for brugergrænsefladen	Brugergrænsefladen åbner på computeren.		
Bruger/superbruger en trykker på "rul ned" knappen	Gardinetrullen ned		

Tabel 12 - Accepttest Use case 4

4.5 Accepttest ikke-funktionelle krav

Udstyr til test af krav:

- Multimeter (bruges til at måle spænding)
- Målebånd (bruges til at måle længder)
- Stopur (bruges til tidtagning)

No.	Krav	Test / udførelse	Faktisk observation / resultat	Vurdering (OK/FAIL)
U1	Der medfølger en brugermanual til systemet	Der er en brugermanual	Testes ikke	

U2	Det skal være muligt at finde samtlige GUI funktioner i selvsamme manual	GUI funktioner findes i manualen	Testes ikke	
R1	Systemet skal kunne udføre UC4 mindst 15 gange inden der skal skiftes dele	UC4 udføres 15 gange		
R2	Systemet skal kunne gennemføre UC1 mindst 15 gange inden der skal skiftes dele	UC1 udføres 15 gange		
P1	Systemet skal kunne rulle helt op og helt ned, til en afstand af 1 centimeter fra vinduets top eller bund	UC4 udføres og afstanden til top / bund måles med målebånd	Testes ikke, idet Alexander ikke længere er en del af gruppen.	
P2	Gardinet skal kunne rulle op med en hastighed på 7 cm / s	UC4 udføres og der tages tid med stopur. Herefter beregnes hastigheden	Testes ikke, idet Alexander ikke længere er en del af gruppen.	
S1	Systemet skal kunne repareres på maksimalt 1 time (MTTR)	Med stopur tages der tid på hvor længe det tager at reparere systemet		
S2	Systemets fysiske dele bør kunne udskiftes på 30 minutter	Der tages tid med stopur og fysiske dele udskiftes		

M1	Gardinets motor opererer ved en gennemsnitseffekt på $100W \pm 5W$	Motorens gennemsnitseffekt måles med multimeter	Testes ikke, idet Alexander ikke længere er en del af gruppen.	
M2	Motoren tager 100 steps pr. 1 meter gradin ± 2 centimeter	Motoren indstilles til at køre 100 steps og herefter måles gardinets længde med målebånd	Testes ikke, idet Alexander ikke længere er en del af gruppen.	
K1	Ved forkert kode skal en rød LED lyse. (LEDR(13))	Forkert kode indtastes		
K2	Ved rigtig kode skal en grøn LED stoppe med at lyse. (LEDG(0))	Korrekt kode indtastes		
K3	På kodelåsen (DE2) skal KEY(0...3) bruges til indtastning af koden	Koden til kodelåsen indtastes på de 4 taster		
K4	Koden til kodelåsen skal være "1111"	Koden "1111" indtastes på kodelåsen og der låses op		
X1	Det simulerede elnet som X-10 opererer på skal have en Vrms på $18V \pm 4V$	Spændingen over det simulerede elnet måles med multimeter		

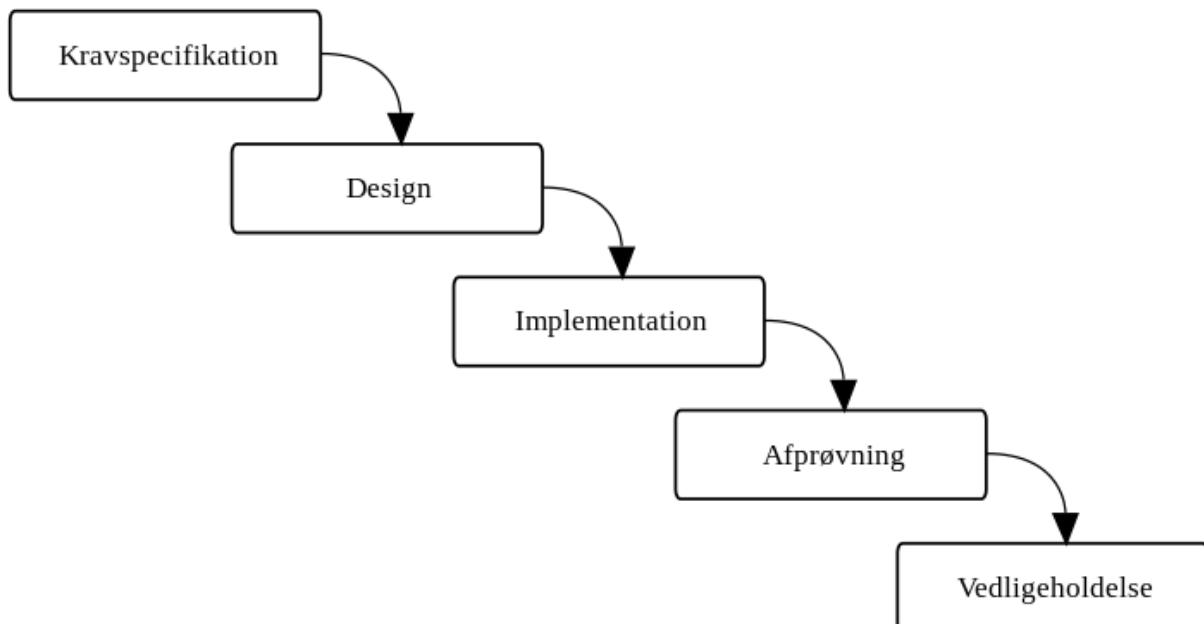
Tabel 13 - Accepttest af ikke-funktionelle krav

5.0 Metode og proces

I dette afsnit vil metoderne som bruges i projektet blive kort forklaret. Formålet med disse metoder er at overskueliggøre og fastsætte arbejdsprocessen for udarbejdelsen af produktet.

5.1 Vandfaldsmodellen

Vandfaldsmodellen er en metode til gennemførelse af et projekt, hvor man først påbegynder næste opgave når foregående opgave er færdig. Det er altså strengt sekventielt.

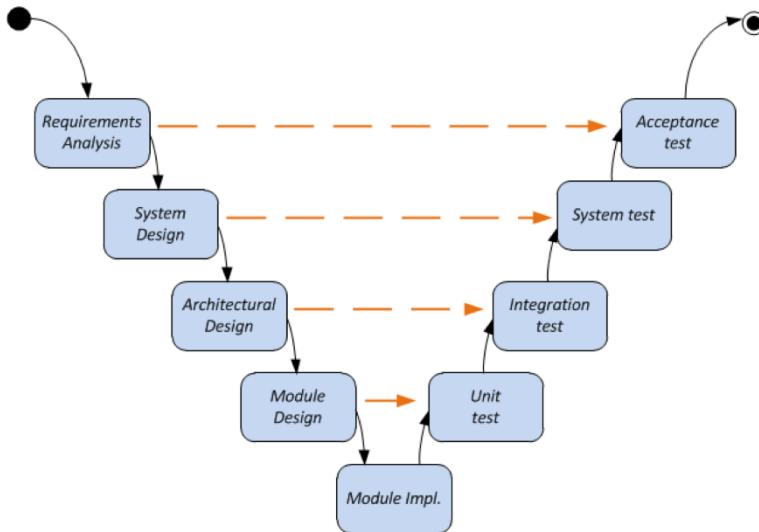


Figur 6 Vandfaldsmodellen

Vandfaldsmodellen bliver brugt til udarbejdelse af dele af vores projekt, da det er et mindre projekt. I større projekter, hvor kravene vil blive ændret løbende, der ville man ikke kunne gøre brug af vandfaldsmodellen i samme omfang. Da vi selv sætter kravene til vores projekt, så er det mere brugbart med denne sekventielle model.

5.2 V-modellen

V-modellen anses som værende en viderebygning af vandfaltsmodellen, hvori man i hvert skridt af udviklingsfasen også har en test associeret med.



Figur 7 V-modellen

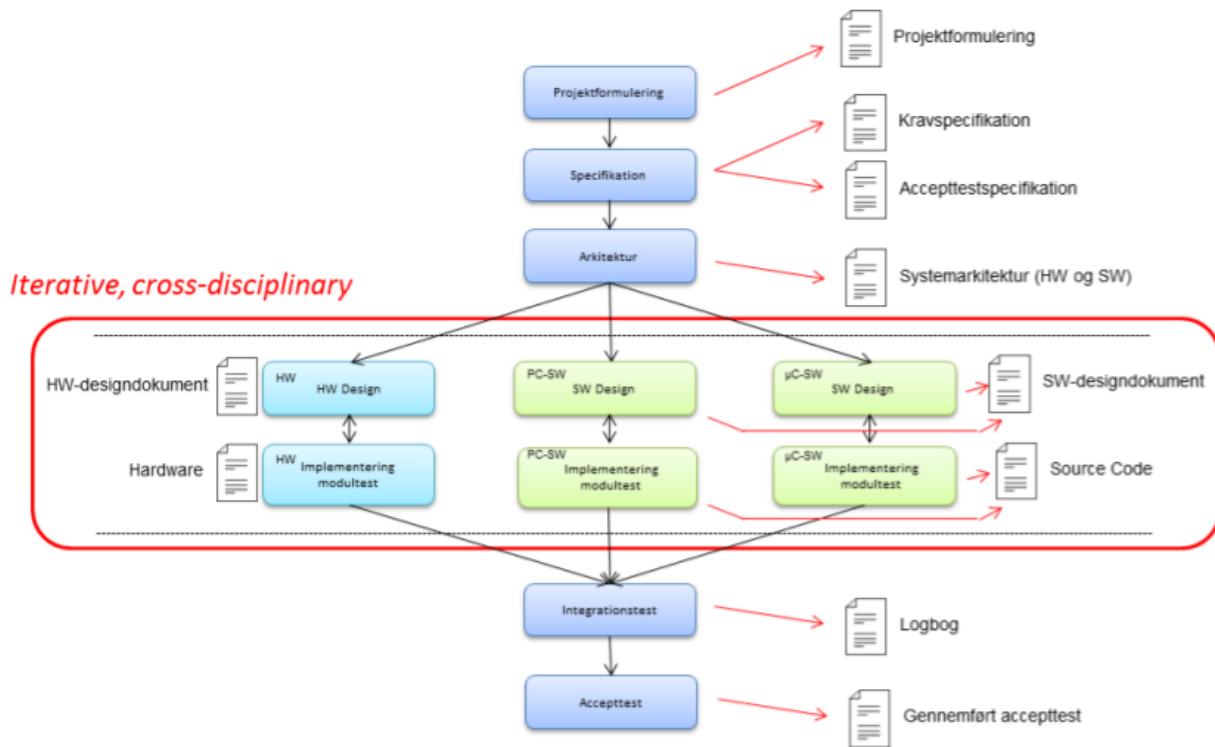
Man kan altså se på ovenstående figur 7 hvordan man har en test for hver fase i udviklingen. Man tester altså løbende, før man endeligt laver en afsluttende accepttest.

5.3 Semesterprojektsmodellen

I semesterprojektsmodellen for 2. semester er udviklingsprocessen opdelt i bestemte faser og blokke som vist i semesterprojektsmodellen herunder. Formålet med arbejdsformen beskrevet i modellen er at specificere et mål eller et færdigt produkt for hver fase og til slut have et færdigtestet og dokumenteret produkt eller system.

De første faser udarbejdes fælles i gruppen, og har til formål at opdele projektet i blokke, så man derefter kan designe, udvikle og teste de individuelle blokke parallelt. Udførelsen af udviklingen og testen af blokkene bør foregå iterativt så man efter hver iteration kan beslutte hvad der skal indgå i de færdige produkt/system. Derved får man efterhånden et større og større system så der gradvist kan føres integrationstests internt i gruppen. Det er i dette projekt valgfrit om vi vælger at dokumentere modul- og integrationstest.

Ved enhedstests testes de enkelte HW- og SW-blokke mens ved integrationstest testes systemet i flere trin hvor der gradvist tilføjes dele af systemet så man til sidst har et færdigtestet system der er klar til aflevering/lancering.



Figur 8 - Semesterprojektsmodellen

5.4 SysML

SysML (System modeling language) er et visuelt "sprog", som i dette projekt er blevet brugt til at lave diagrammer. F.eks. i hardwarearkitekturen, er der lavet BDD (block definition diagram) og IBD (internal block diagram), som tilsammen giver en god visualisering og forståelse for opbygning af systemets hardwaredele.

5.5 UML

UML (Unified Modelling language) består ligeledes også af diagrammer. I dette projekt er det hovedsageligt blevet brugt til at beskrive software, gennem SD (Sekvensdiagrammer) og Klassediagrammer. Desuden bruges domænemodellen til at vise det endelige system. I domænemodellen ses sammenspillet mellem hardwarens enkelte dele.

I projektet er der opstillet sekvensdiagrammer for hver use case, der viser hvilke enkelte use case' forskellige funktioner, og beskriver rækkefølgen use casen bliver udført i.

6.0 Analyse

Dette afsnit vil indeholde en analyse af de enkelte enheder i det overordnede system. Der vil blive reflekteret over mulig komponent valg, og mulige løsninger.

6.1 Kodelås

I DSD har vi lavet en kodelås på DE2-boardet, som også skal bruges i projektet.

Implementeringen af kodelåsen er skrevet i VHDL. Kodelåsen er lavet ved brug af finite state maskiner, og fungerer ved at den skifter imellem de forskellige "states"

Kodelåsens funktion er at sikre, at kun personer med den korrekte kode kan indstille tidsintervallerne og deaktivere og aktivere autostyringen.

I vores projekt vil alle knapper i autostyringsmodulet på brugergrænsefladen være inaktive, mens kodelåsen er låst.

6.2 X10 Sender

Senderens formål er at sende et 120 kHz signal ud på elnettet. Kredsløbet dæmper elnettets signal, og lader signalet fra mikrocontrolleren passere forbi. Dette sker ved hjælp af et højpasfilter og et transistor-kredsløb. Mikrocontrolleren sender sine funktioner igennem senderkredsløbet ved hjælp af 1 ms bursts, der bliver modtaget i modtagerkredsløbet. Dette tolkes i modtagerkoden og behandles derefter.

For at opbygge sender-kredsløbet, tager vi udgangspunkt i applikationsnoten, som vi har fået udleveret i forbindelse med projektet. I applikationsnoten er der givet nogle værdier for en lignende applikation, men for 120VAC ved 60Hz, og derfor skal der beregnes nye værdier. Der vil ikke overvejes andre løsninger end denne.

6.3 Zero Cross Detector

Zero Cross Detectoren er koblet på el-nettet, og har til formål at registrere, hvornår de 18V på el-nettet krydser 0 volt. Ved et "zero cross" bliver det behandlet af kredsløbet til at være et digitalt højt signal. Dette signal sendes til vores sender-controller der påbegynder en interrupt-rutine som sender den ønskede bit-streng, hvis brugeren har valgt en ønsket kommando.

For at forsvarligt kunne arbejde med elnettet, skal de 18V isoleres fra den del af kredsløbet der lever et 5V logisk højt signal. Her er tanken at der kan bruges en optokobler, hvis formål er at afkoble højspænding fra lavspænding, da der kan bruges en schmitt-trigger til at lave det logiske høje signal, som ikke kan håndtere 18V.

Modtager-controlleren skal også kobles til et zero cross kredsløb, da det kun skal "lytte" til kommandoer når der kommer et zero cross.

Der bygges derfor 2 identiske Zero Cross kredsløb, der begge skal lytte på elnettet, og give besked videre når det krydser 0 volt.

6.4 X10 Modtager

X10 modtagerens funktion er at modtage data fra X10 senderen gennem elnettet.

X10 modtagerens funktion er at modtage data og funktioner fra mikrocontrolleren gennem elnettet, og behandle disse data så en mikrocontroller kan læse dem, og sende dem videre til vores stepmotor som styrer vores gardin.

Kredsløbet kan bygges op af et højpas filter med et forstærkerled, et led til at glatte signalet ud, og en schmitt-trigger til at omdanne signalet til et 5V logisk højt signal som kan læses fra en Mikrocontroller.

6.5 Gardin

Til styring af gardin bruger vi en stepmotor, der styres af vores X10 modtager.

Motoren har 4 inputs, der kobles op til 4 outputs på mikrocontrolleren. Derudover skal motoren også have 12 volts VCC.

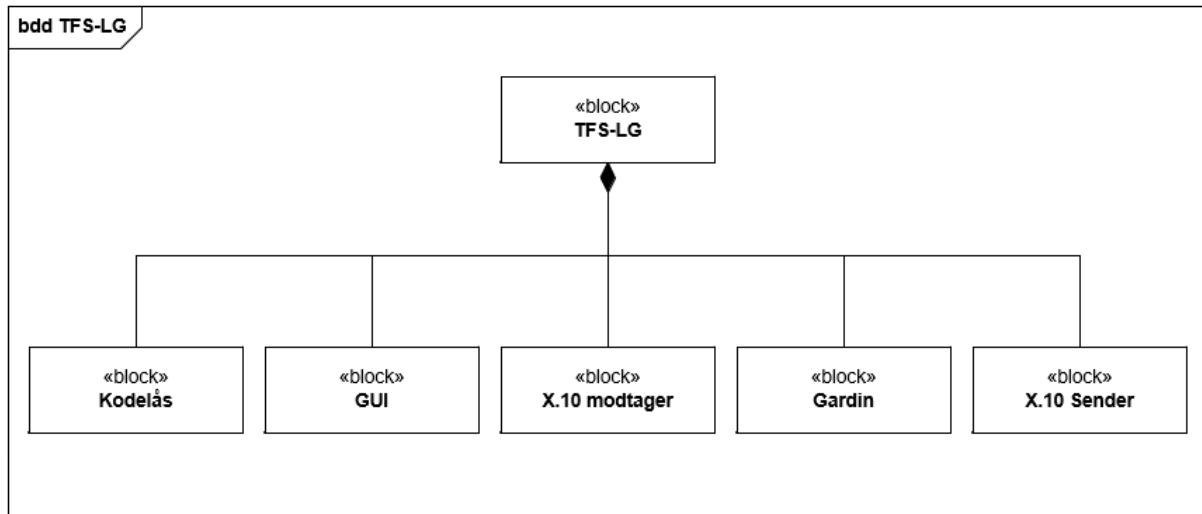
Man kan koble 5-12 volt på, men da vi skal bruge den til at trække et gardin, vælger vi 12 volt for den største ydeevne

7.0 Arkitektur

Dette afsnit omhandler systemarkitekturen for både software og hardware for TFS-LG. Herigennem vil det være muligt at se modeller over det overordnede system, samt udpegsling af hardware-, og software-arkitekturen.

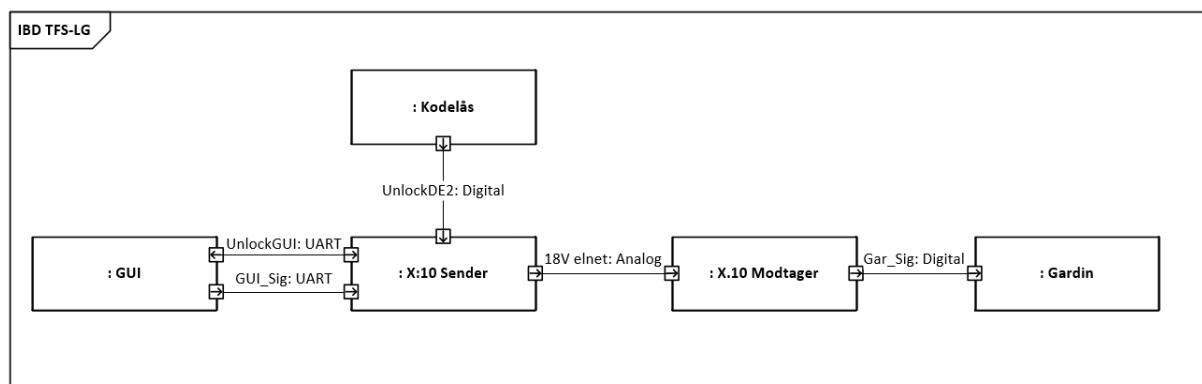
7.1 Overordnet system arkitektur

På figur 9 ses BDD'et for hardwaren af vores overordnede system. BDD'et giver et godt overblik over hvilken hardware der skal udvikles og hvordan de forskellige blokke



Figur 9 - BDD for overordnet system

Vores system er inddelt i 5 delsystemer. Et delsystem er defineret ved, at det er et system med sin egen CPU og dertilhørende software. På figur 10 ses IBD'et for disse 5 delsystemer, hvoraf der fremgår flow af information mellem delsystemerne.



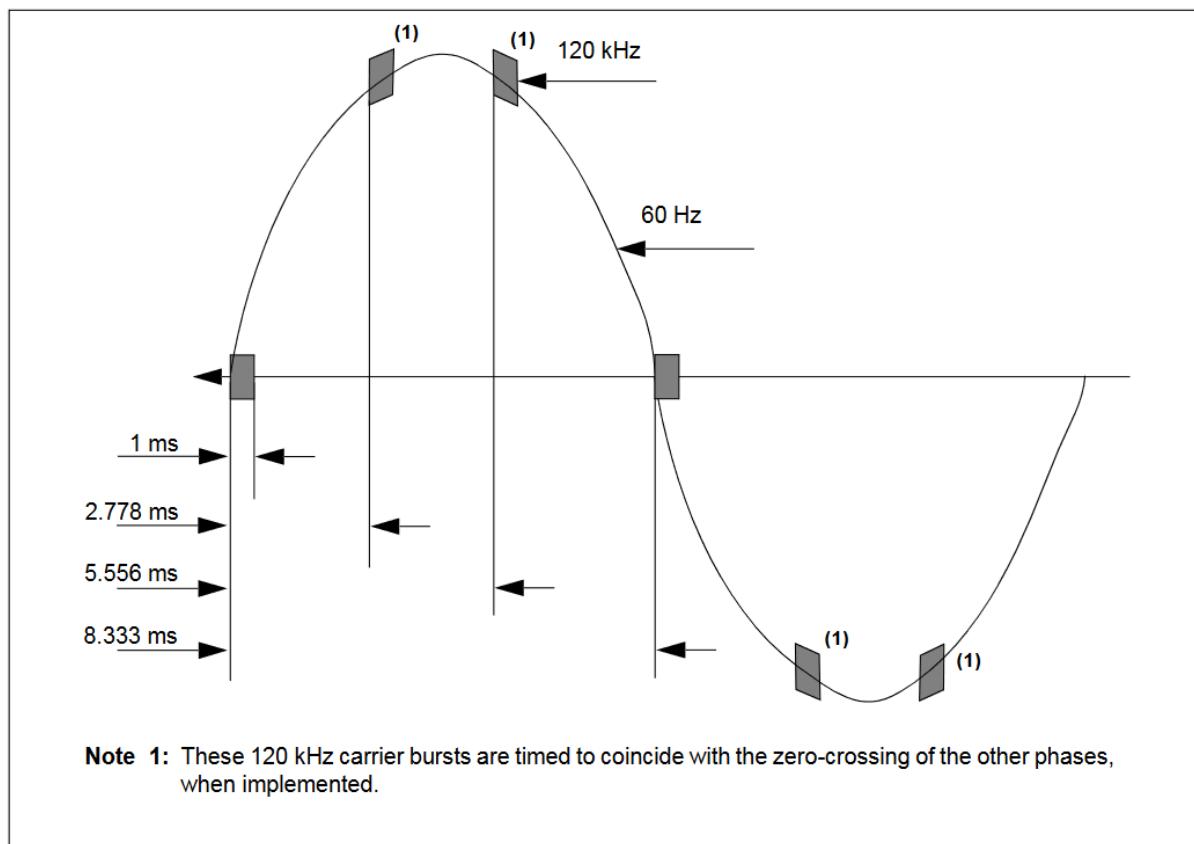
Figur 10 - IBD for overordnet system

7.2 Protokoller

I dette afsnit vil vi kort forklare de protokoller der bruges i vores projekt. Dette vil henholdsvis være X10 og UART.

7.2.1 X10

X10 er en protokol for kommunikation mellem elektroniske enheder. Protokollen er designet til at operere på et 120VAC elnet, hvorpå den kontrollerer de påsatte X10 enheder. X10 fungerer ved at tjekke elnettet for zero-crossings og enten sende et logisk "1" eller et logisk "0" for hvert af disse. Et logisk "1" er repræsenteret ved et 1 ms 120 kHz burst efter hvert zero-crossing. Et logisk "0" er derimod repræsenteret ved manglen på et 120 kHz burst ved en zero-crossing. En illustration af dette fremgår af figur 11, hvortil det skal noteres, at der i vores tilfælde kun tages højde for den fase, som ligger ved et reelt zero-cross.



Figur 11 - 120 kHz burst

En variation af X10

Det at der kun tages højde for én fase og ikke alle el-nettets 3 faser gør, at vi arbejder med vores egen variation af X-10 protokollen, som tager udgangspunkt i applikationsnoten². Da systemet kun benytter sig af én fase, så sender vi også kun 2 bursts per periode. Såfremt

https://blackboard.au.dk/bbcswebdav/pid-2290252-dt-content-rid-6994102_1/courses/BB-Cou-UUVA-86257/AN236_ApplicationNote.pdf

X10 protokollen var blevet brugt til fulde havde man gjort brug af alle elnettets 3 faser, hvilket ville medføre at man kunne sende 6 bursts per periode.

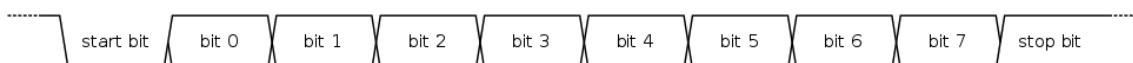
I dette system benyttes X10 protokollen heller ikke som beskrevet i applikationsnoten, i forhold til bit strømmen. X10 protokollen beskriver at man først skal sende adressen 2 gange og dernæst command 2 gang med kun stopbits imellem. I dette system sendes adressen og command kun én gang hver. Systemets rækkefølge ender derfor med at hedde Startbit – husadresse – unit adresse – suffix – stopbit – startbit – husadresse – function command – suffix – stopbit, hvilket udgør en samlet bitstrøm på 56 bit.

7.2.2 UART

For at kunne sende information fra GUI'en til den mikrocontroller der fungerer som x10 sender, benyttes der en UART-forbindelse.

Helt grundlæggende er en UART-forbindelse en asynkron seriell forbindelse (Universal Asynchronous receiver-transmitter), der sender en streng af bits der består af et start bit, 5-8 data bits, og en eller to stop bits (se figur 12). Dertil er der også mulighed for en parity bit, der fortæller om hvorvidt der er ligevægt mellem høje og lave bits i den bitstreng den tilhører (ikke afbilledet).

Både afsender og modtager af et UART-signal skal være "enige" om disse fornævnte kriterier, samt kommunikationens baudrate. Baudraten er defineret som bits pr sekund, og er altså hastigheden med hvilken kommunikationen sendes. Har modtager og afsender ikke samme baudrate, vil forbindelse ikke fungere.



Figur 12 - eksempel på en UART-bit streng med 8 databits.

I vores system er der en seriell kommunikation fra GUI'en og ned til x10 senderen der bestemmer hvornår og hvilken command senderen transmitterer ud på elnettet.

Derudover læser x10 senderen også på en pin forbundet med kodelåsen om autostyringen er låst op. Denne information sender den som et UART-signal til GUI'en der herefter låser op for autostyringsmenuen.

Begge disse forbindelser kører med 1 startbit, 8 databits, 1 stopbit og ingen paritybits, samt en baudrate på 9600 bits pr sekund.

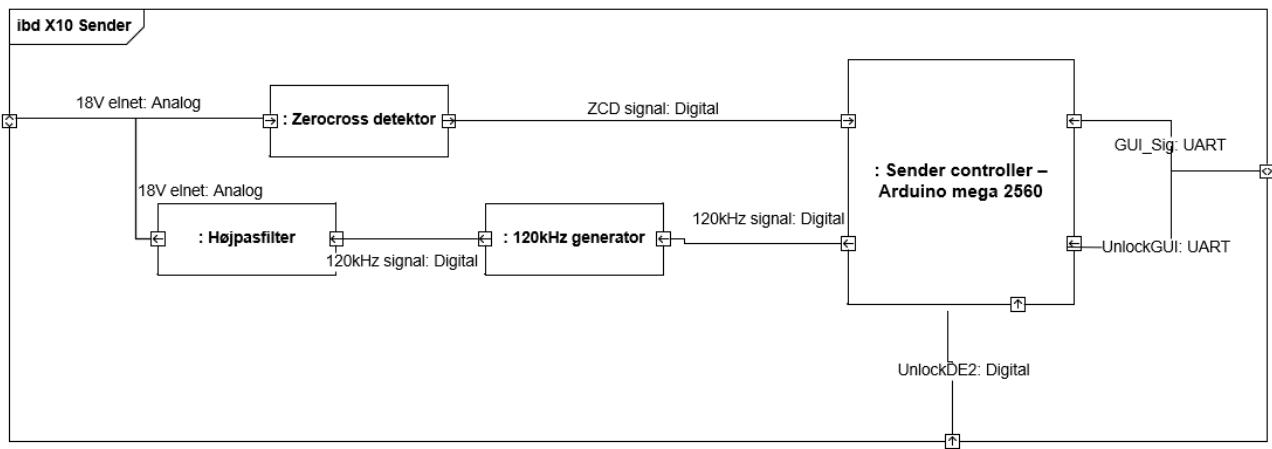
7.3 Hardware arkitektur

I dette afsnit vil der blive arbejdet med arkitekturen for de hardwareelementer der kommer til at indgå i dette projekt. Her vil der blive lavet IBD og BDD for hardwaren, samt en blok og signal beskrivelse for hele systemet.

7.3.1 IBD

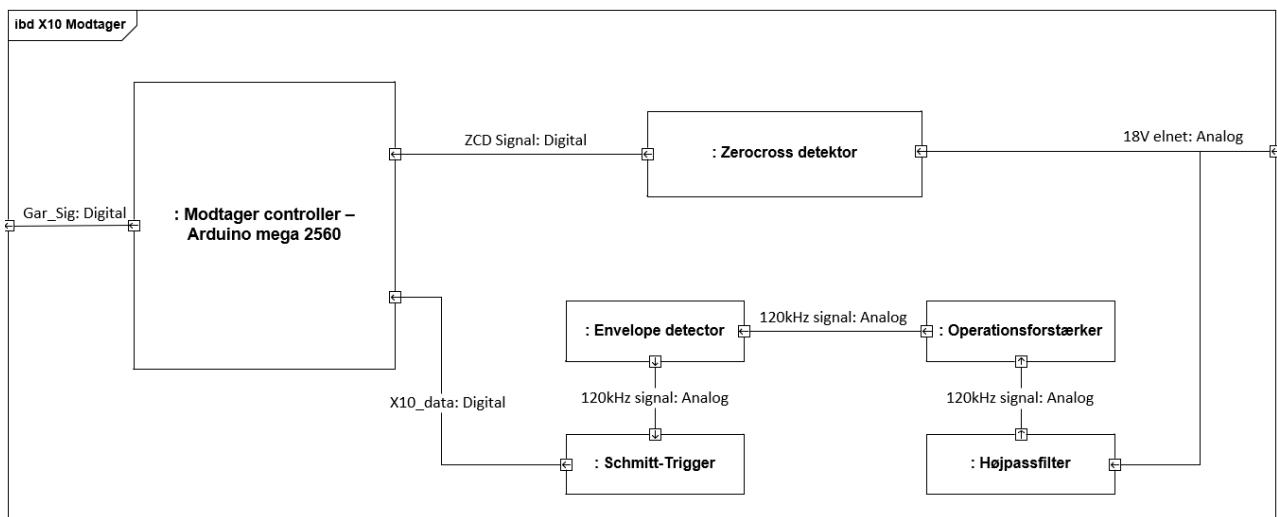
X10 sender-modulet består af en Zero Cross detektor, et højpasfilter og en 120kHz generator. Desuden består det også af en sender controller, som er en Arduino mega 2560 Mikrocontroller. De interne signaler kan ses på nedenstående block diagram i figur 13

IBD for X10 Sender modul



Figur 13 - IBD for X10 Sender

Modtager modulet består af en Zero Cross Detektor, en schmitt-trigger, en envelope detector, Operationsforstærker, og et højpasfilter. Desuden består det også af endnu en Mikrocontroller, som i dette tilfælde fungerer som modtager controller. Nedenstående block diagram viser modtager modulets indre blokke
IBD for X10 Modtager modul

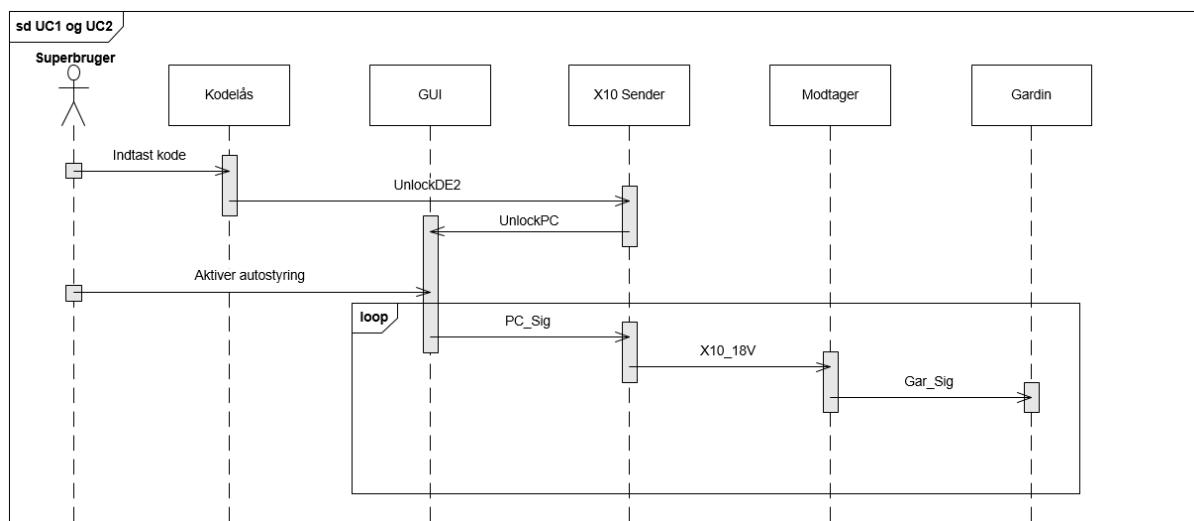


Figur 14 - IBD for modtager

7.3.2 System sekvens diagram

I dette afsnit vises systemets sekvensdiagram. Diagrammet tager udgangspunkt i use case 1 og 2.

System sekvensdiagram for UC1 og UC2



Figur 15 - System sekvens diagram for UC1 og UC2



7.4 Blok beskrivelse

Bloknavn	Funktionsbeskrivelse	Signalnavn	Signaltyppe	Port specifikation
X10 Sender	Bestemmer den data der skal sendes over elnettet og har desuden ansvaret for at sende data med et 120kHz signal ud på elnettet. For CMOS logik: Høj når > 3.5V Lav når < 1.5V	ardPower	VIN: DC	0-12V Single Supply Voltage, Tolerance ± 6V, Max 1A
		ardGround	GND: DC	Arduino ground
		UnlockDE2	In: Digital	0-5V ± 500mV, Max 40 mA
		UnlockGUI	Out: UART	0-5V ± 500mV, Max 40 mA
		GUI_Sig	In: UART	0-5V ± 500mV, Max 40 mA
		120kHz Signal	Out: Digital	Firkant signal på 120kHz ± 2kHz, 50% duty cycle.
		ZCD Signal	In: Digital	5V ± 500mV, Max 40mA
X10 Modtager	Modtager signalet fra elnettet. Har ansvaret for at filtrere lavfrekvenssignalet fra, så kun 120kHz signalet er tilbage. Dette signal skal yderligere behandles, så data'en kan læses som et logisk "1" eller "0".	ardPower	VIN: DC	0-12V Single Supply Voltage, Tolerance ± 6V, Max 1A
		ardGround	GND: DC	Arduino ground
		X10_data	In: Digital	0-5V ± 500mV, Max 40 mA
		Gar_Sig	Out: Digital	0-5V ± 500mV, Max 40 mA
		ZCD Signal	In: Digital	5V ± 500mV, Max 40mA
Kodelås		Trigger	Force	Knapper



	Kodelåsen fungerer som en aktivering/deaktivering af GUI'en	DE2Power	VIN:DC	Single Supply Voltage Tolerance 0-9V ±0.2V Max 1,3 A
Gardin	Indeholder steppermotor og motordriver.	Gar_sig	In:Digital	0-5V ± 500mV, Max 40 mA
		MotorPower	VIN: DC	0-12V Single Supply Voltage, Tollerance ± 6V, Max 200mA
		MotorGND	GND:DC	Arduino ground
GUI	Disse signaler dækker over typen af in og outputs som gui'en enten sender eller behandler. GUI_sig dækker over on og off-commands der sendes til x10-sender blocken. UnlockGUI er et uart signal der enten fremgår som høj(1) eller lav(0) og bestemmer om autostyring er låst op eller ej.	UnlockGUI	In:Uart	Baud-rate: 9600 Parity bits: none Stop bits: 1 Data bits: 8 Portnavn: Com1-8
		GUI_sig	Out:Uart	Baud-rate: 9600 Parity bits: none Stop bits: 1 Data bits: 8 Portnavn: Com1-8

Tabel 14 - Blokbeskrivelse



7.5 Signalbeskrivelse

Signal-navn	Funktion	Område	Port 1	Kommentar
UnlockDE2: Digital	Sender et signal til sender-controlleren om gyldig kode er indtastet	0-5V ± 500mV Max 40mA	DE2 GPIO 1 ben 0	Sender et logisk "1" til sender-controlleren ved korrekt kode.
UnlockGUI: UART	Sender et signal fra sender-controlleren til PC om at låse op for GUI'en.	0-5V ± 500mV Max 40mA	USB/UART COM PORT 3	Baud-rate: 9600 Parity bits: none Stop bits: 1 Data bits: 8 Portnavn: Com1-8
GUI_sig: UART	UART-kommunikation til X10 senderen.	0-5V ± 500mV Max 40mA	USB/UART COM PORT 3	Baud-rate: 9600 Parity bits: none Stop bits: 1 Data bits: 8 Portnavn: Com1-8
18V elnet: Analog	Fungerer som kommunikationsnet for X10 protokollen.	18V RMS ± 4V Max 0.5A	N/A	Det fælles elnet som der sendes data over.
120kHz Signal: Digital	Et PWM-signal med 50% duty cycle.	0-5V ± 500mV Max 40mA 120kHz ± 2kHz	PE3/OC3A	Bliver genereret af X10 sender blocken, hver gang der er zerocross.
ZCD Signal: Digital	Sender et signal til sender-/modtager-controlleren om hvorvidt der er et zero cross.	0-5V ± 500mV Max 40mA	PDO/INT0	Sender 5V ved et logisk "1" for hvert zerocross detekteret på elnettet.



Gar_Sig:	PWM-signal til motor controller	0-5V± 500mV Max 40mA	PB2/PCINT2 og PB3/PCINT3	PB2 modtager 5V ved Off (rul op) PB3 modtager 5V ved on (rul ned)
-----------------	---------------------------------	-------------------------------	--------------------------------	--

Tabel 15 - Signalbeskrivelse

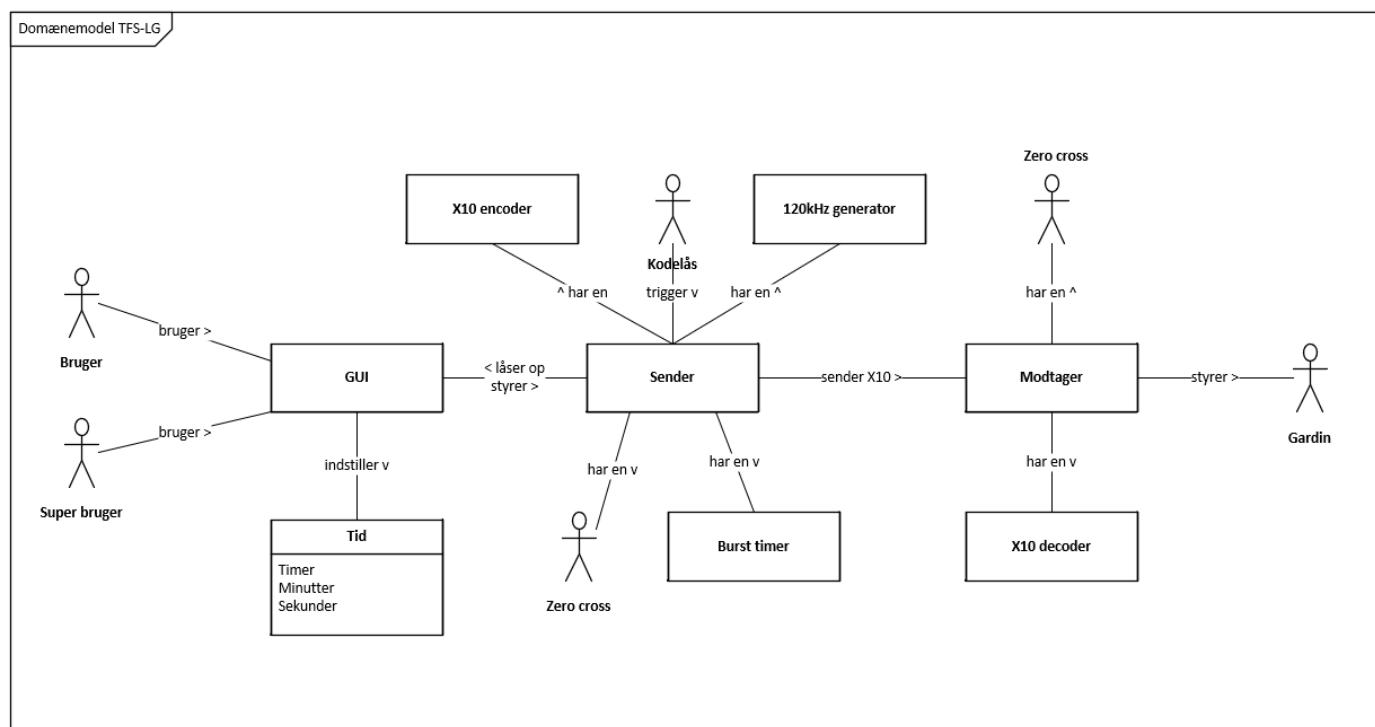
7.6 Software arkitektur

I dette afsnit vil der blive arbejdet med en domænemodel for det overordnede system. Ud fra denne model, vil der også blive arbejdet med klassediagrammer, sekvensdiagrammer, samt state machines for de førnævnte use cases.

7.6.1 Applikationsmodeller

I dette afsnit vil der blive lavet applikationsmodeller for alle softwarepakker. Her beskrives de forskellige softwarepakkers kald til hinanden igennem klassediagrammer og sekvensdiagrammer. Derudover vil det være synligt hvordan de arbejder henover hardwaren.

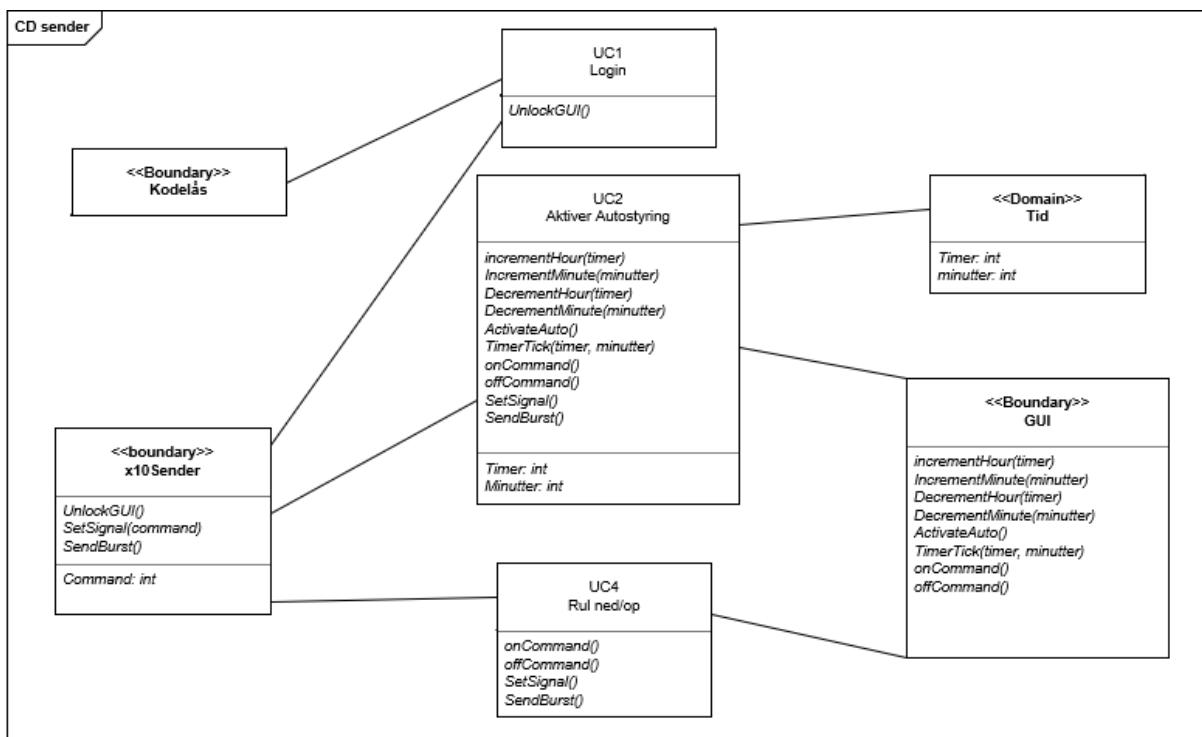
Heraf fremgår domænemodel for det overordnede system. Domænemodellen giver et grundlæggende overblik over systemets interne blokke samt kommunikationen mellem blokkene. Domænemodellen kan ses på figur 16



Figur 16 - Domænemodel for overordnet system

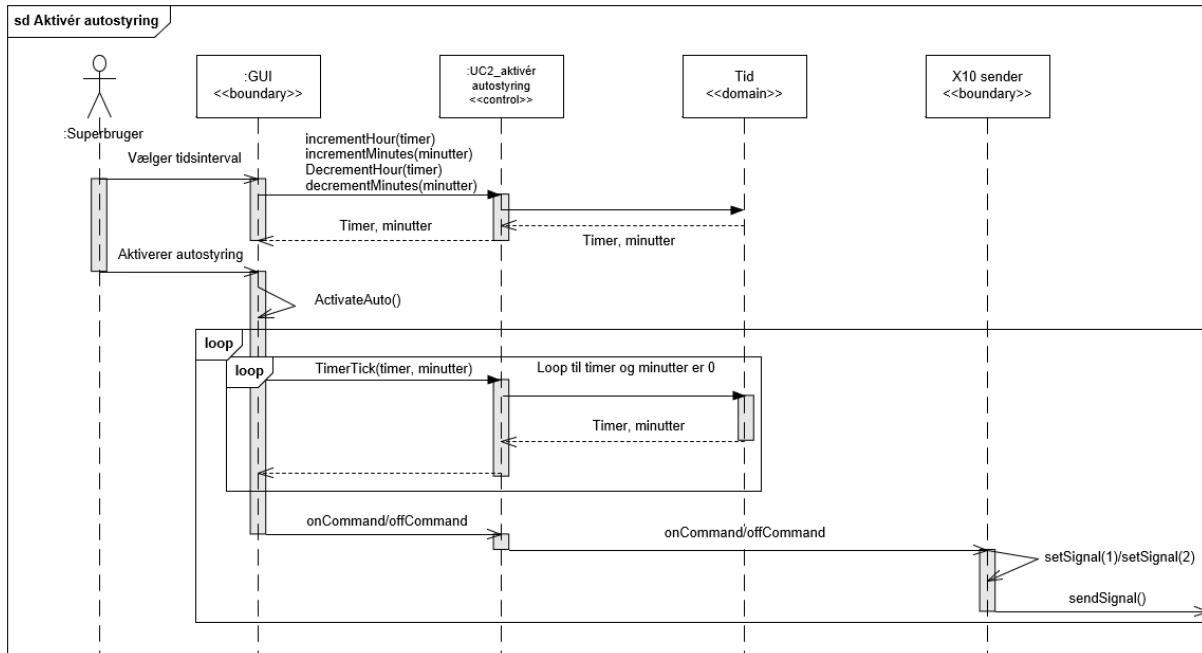
7.6.2 Applikationsmodel for X10 Sender

Applikationsmodellen for X10 senderen indeholder et klasse-diagram der beskriver funktioner og blokke der benyttes i de forskellige use cases hvori senderen indgår. Derudover er der sekvensmodeller for modtagerens softwaredele. Som kontrolklasser i senderen indgår alle Use cases, idet de alle bruger senderen.

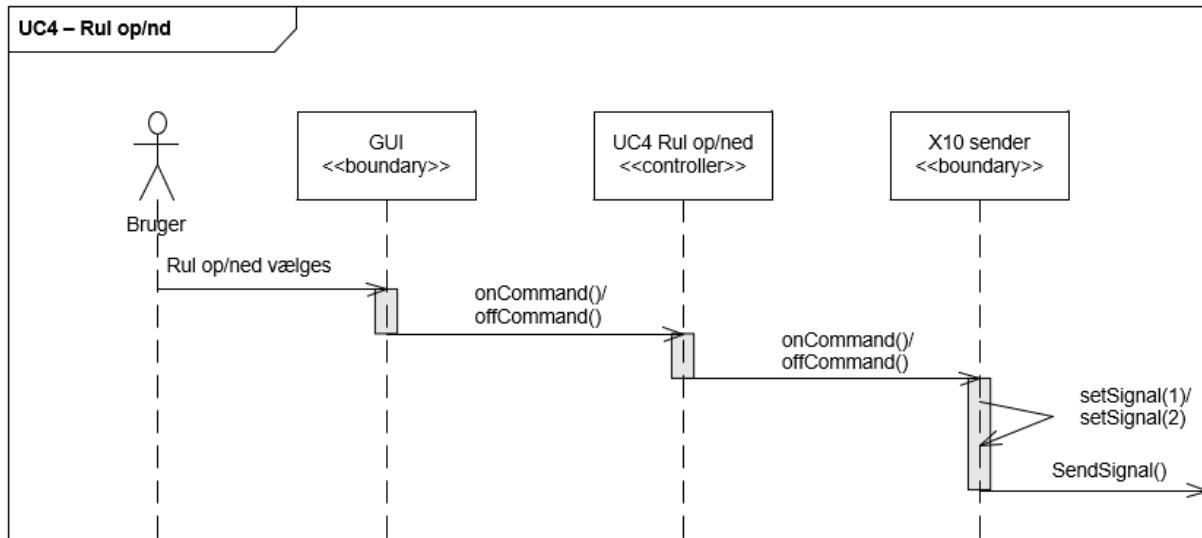


Figur 17 - Klasse diagram for X10 sender.

Hver block har sine egne funktioner i sig. Usecasene har de nødvendige funktioner fra dens tilhørende blokke til at udføre usecasen.



Figur 18 - Sekvensdiagram af aktivér autostyring (UC2)

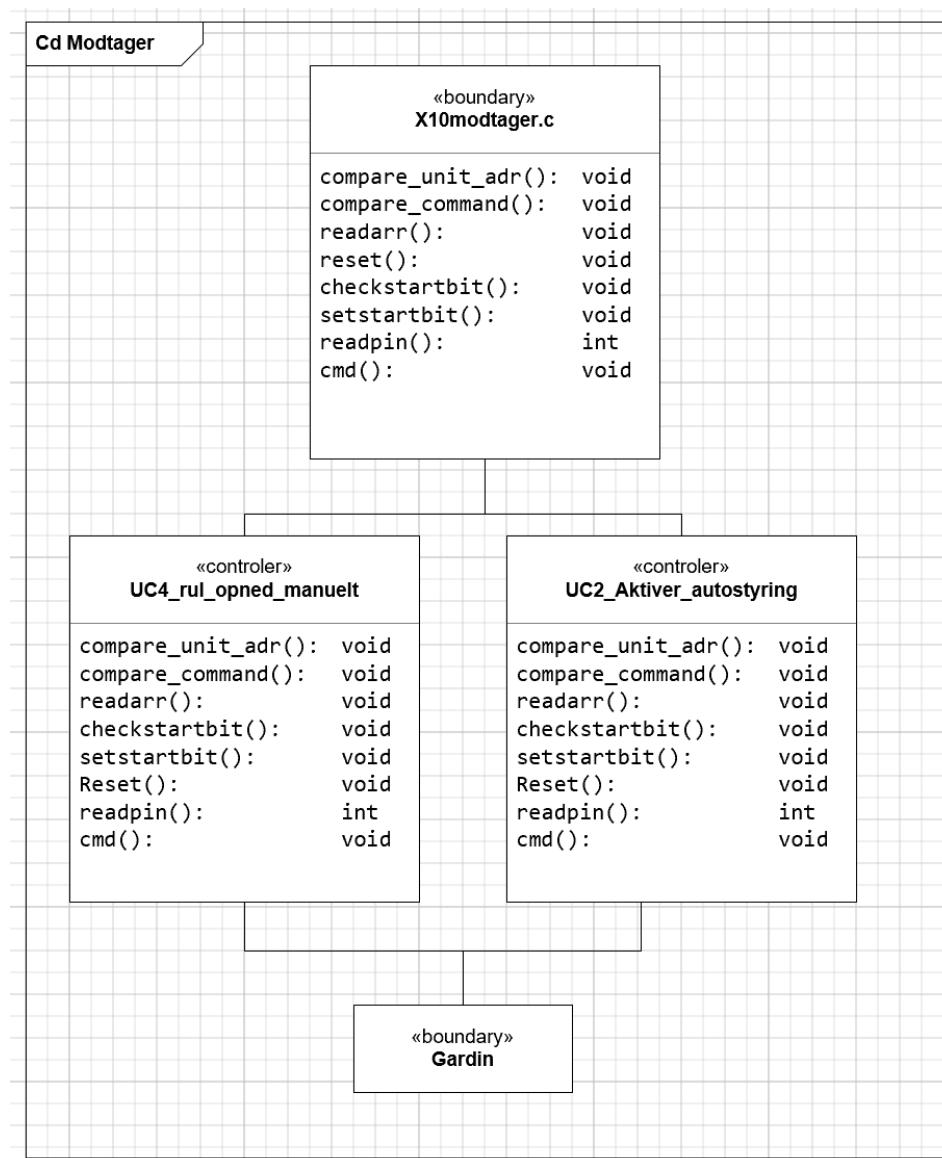


Figur 19 - sekvensdiagram af rul op/rul ned manuelt (UC4)

7.6.3 Applikationsmodel X10 Modtager

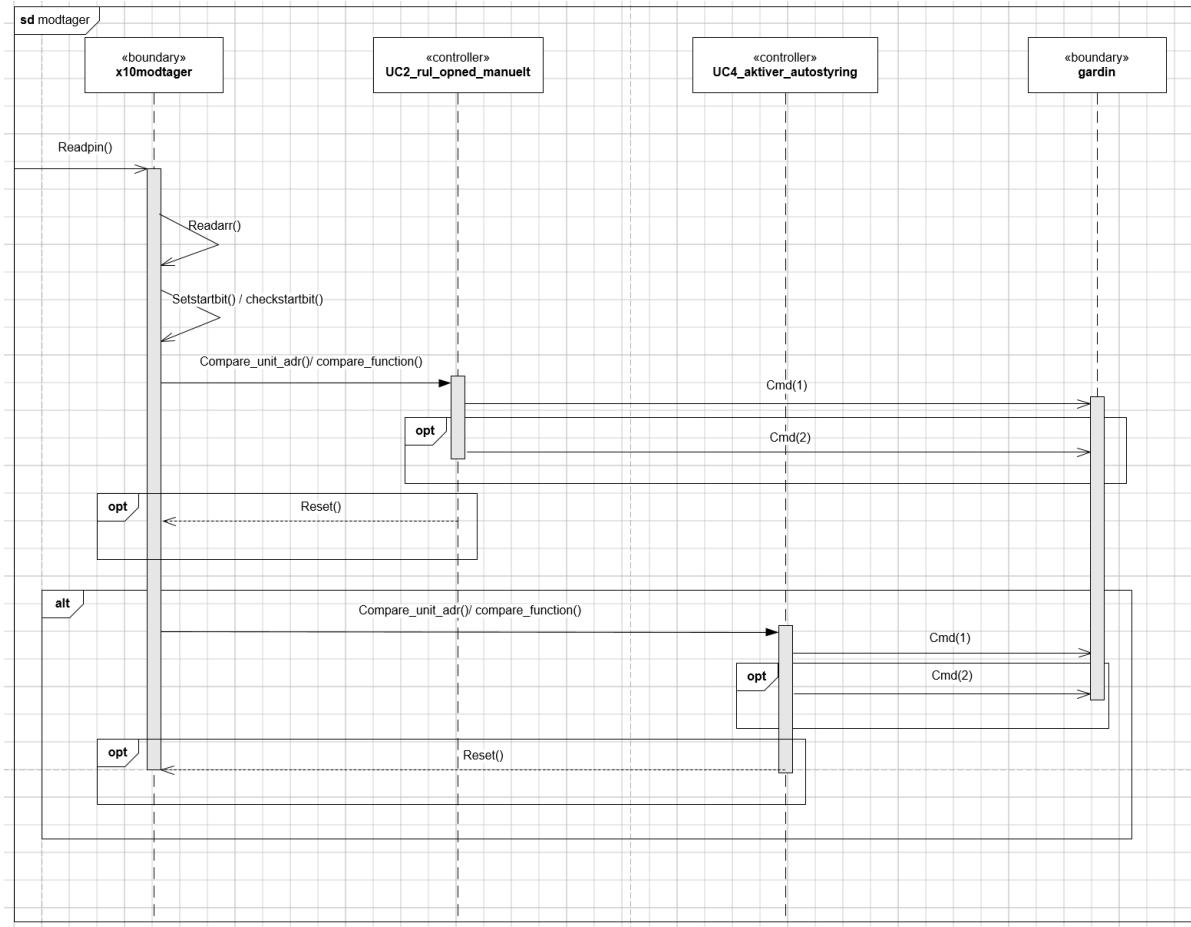
Applikationsmodellen for X10 modtageren indeholder et klassediagram samt en sekvensmodel for modtagerens softwaredele. Som kontrolklasser i modtageren indgår UC2 og UC4, idet det udelukkende er disse use cases der tilgår modtageren.

I applikationsmodellen er der tilføjet 'gardin' som boundary, da det var hensigten at vi skulle styre et gardin. Dog var det Alexander, der desværre stoppede på studiet, der skulle stå for motor og gardinstyringen, hvorfor det altså aldrig blev aktuelt med styring af et gardin. Den er dog stadig tilføjet her. I praksis styrer systemet 2 led'er.



Figur 19 Klassediagram for X10 modtager

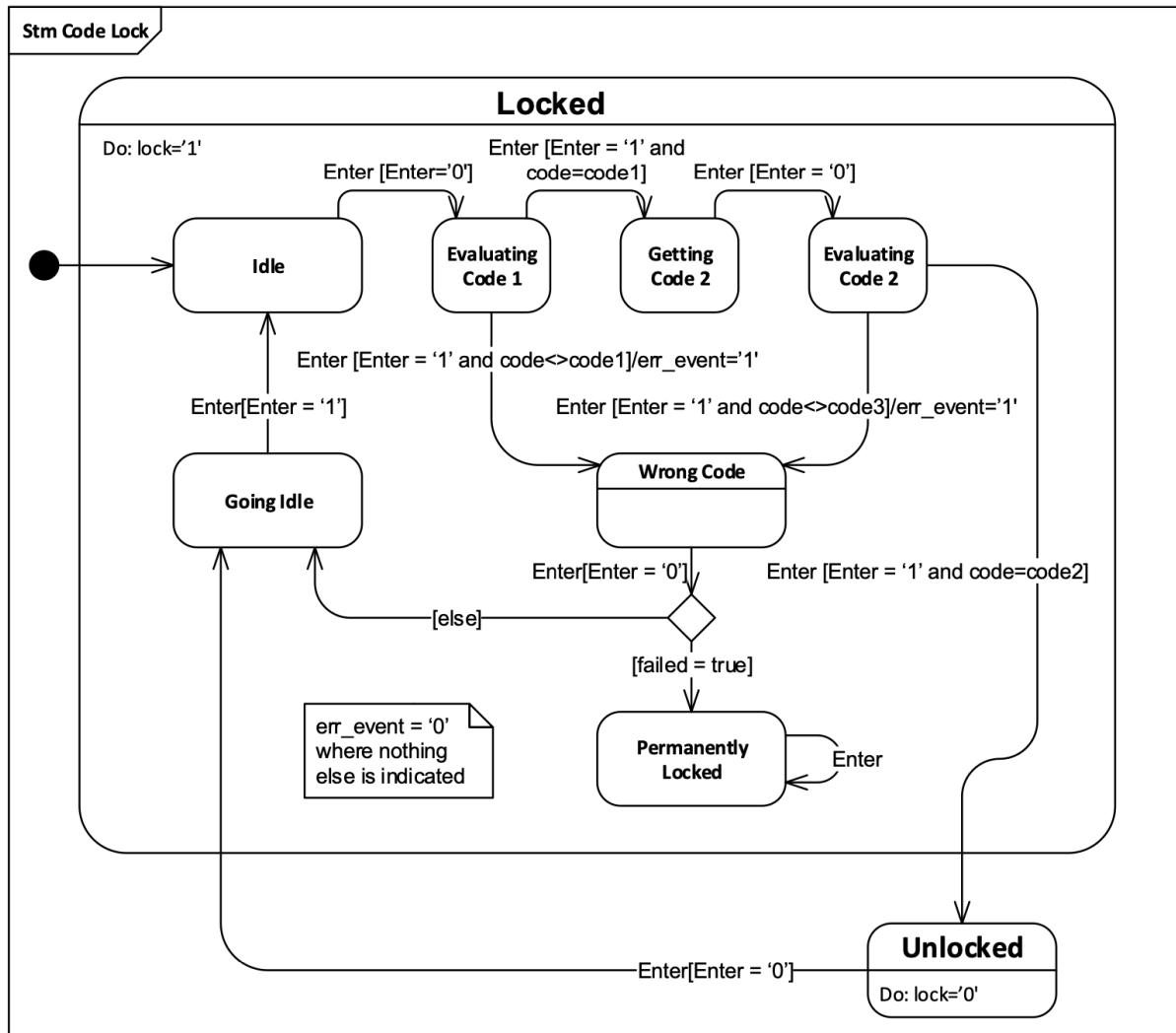
På baggrund af klassediagrammet er der udviklet et sekvensdiagram.



Figur 20 Sekvensdiagram X10 modtager

7.6.4 Applikationsmodel for Kodelås

Applikationsmodellen for kodelåsen indeholder en state machine der viser hvordan evaluering af koden foregår, og hvordan den derefter kan gå i unlocked mode eller i locked mode. State machinen er taget fra opgaveformuleringen til DSD Exercise 7, og fremgår også af vores bilag.

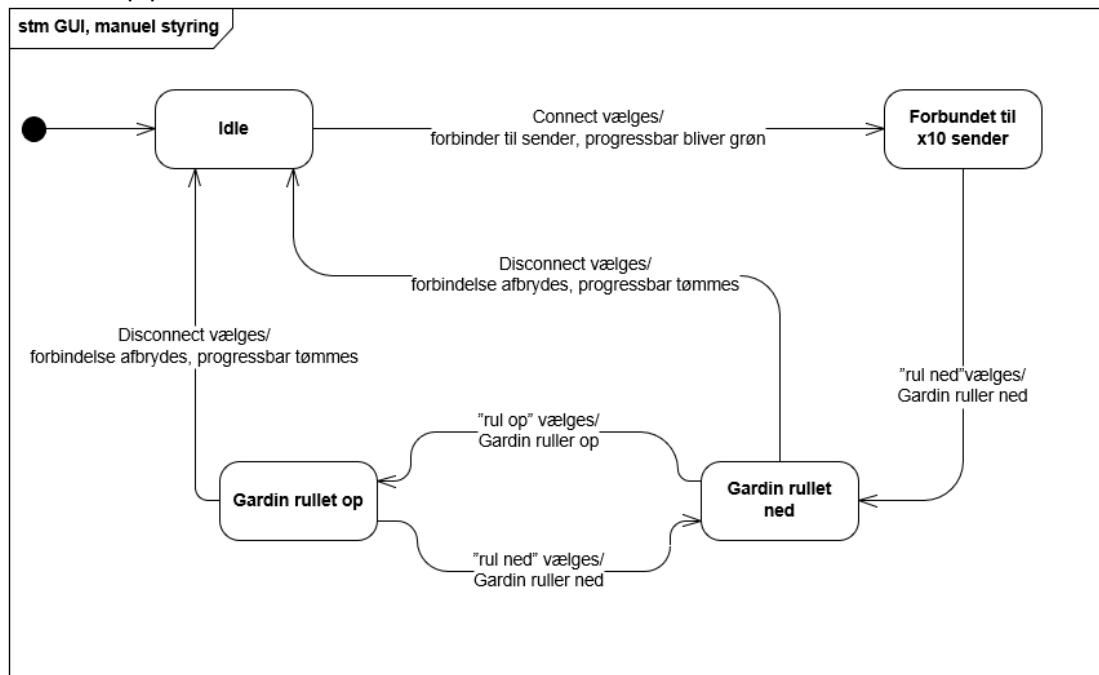


3

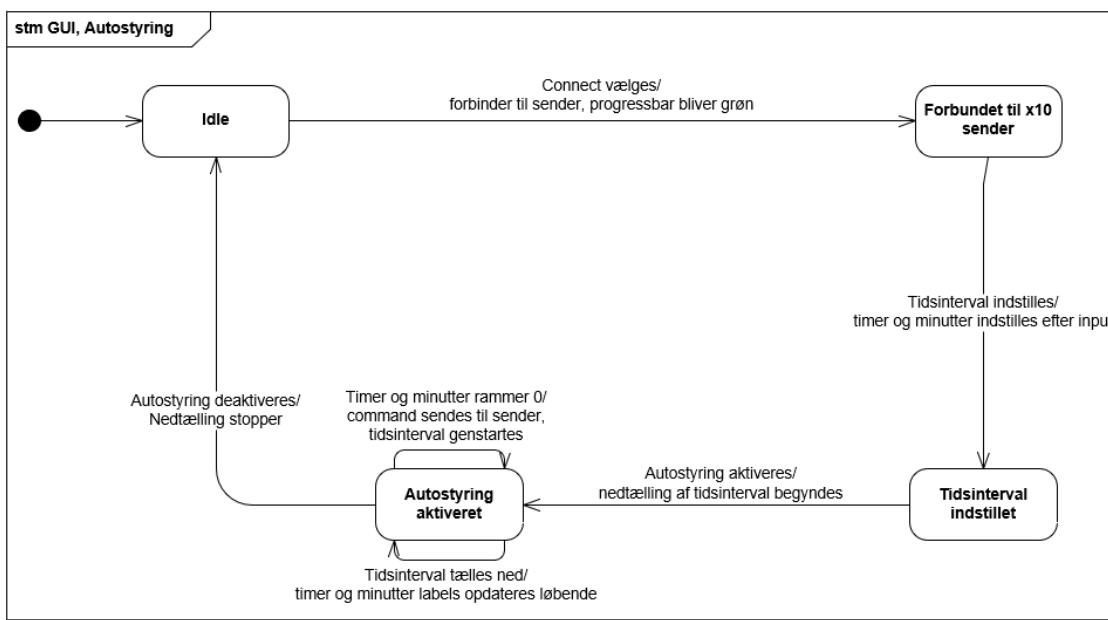
Figur 22 State machine diagram over kodelås

³ https://blackboard.au.dk/bbcswebdav/pid-2299742-dt-content-rid-7021817_1/courses/BB-Cou-UUVA-86254/DSD/Exercises/dsd_exe_7.pdf

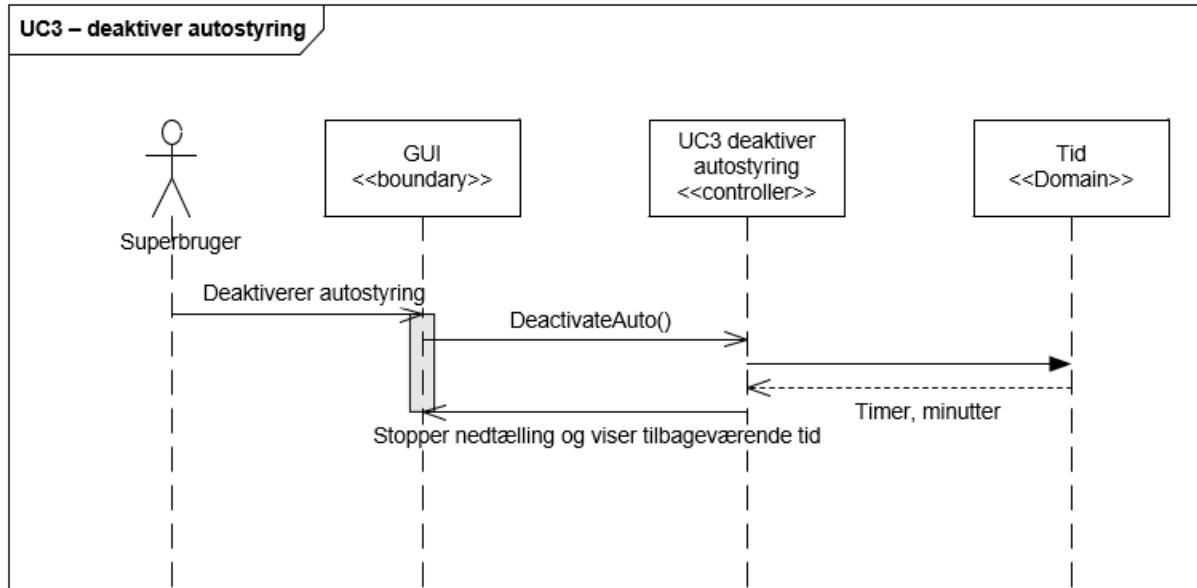
7.6.5 Applikationsmodel for GUI



Figur 23 - Statemachine for GUI når bruger benytter sig af manuel styring



Figur 24 - Statemachine for GUI når der benyttes autostyring



Figur 25 - sekvensdiagram for UC3 - deaktiver autostyring

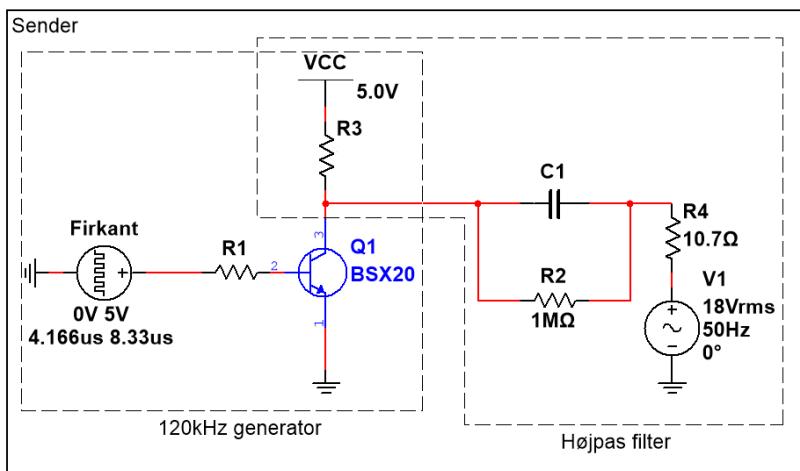
8.0 Hardware Design, Implementation og Modultest

I følgende afsnit vil der blive arbejdet med hardware design, implementation og modultest.

Modultesten er udført ved at teste de enkelte dele en efter en i serie, så det er muligt at følge signalet gennem hele kredsløbet. Dette giver et godt overblik over den samlede hardware test.

8.1 X10 sender kredsløb

På figur 26 ses det færdige sender kredsløb. Diagrammet er delt op i de dele som senderkredsløbet består af, en 120kHz generator, som hovedsageligt består af en transistor som bliver brugt som switch, og et højpas filter for at filtrere alt lavfrekvens fra. Signalet "firkant" er det PWM-signal der kommer fra Sender-controlleren, og R4 er den indre modstand i 18V elnet transformeren.



Figur 26 - Færdige sender kredsløb design, uden værdier for RA(R3), C(C1) og R1

Reference	Værdi	Antal
R1	250Ω	1
R2	1MΩ	1
R3	27Ω	1
C1	0.1μF	1
Q1	BSX20	1

Tabel 16 - Stykliste for senderkredsløb

8.1.1 Højpas filter

Højpasfilteret har ansvaret for at filtrere de 18V 50Hz fra, så 120kHz generatorens transistor og R3, ikke bliver overbelastet.

Højpas filteret består af R3, C1 og R4.

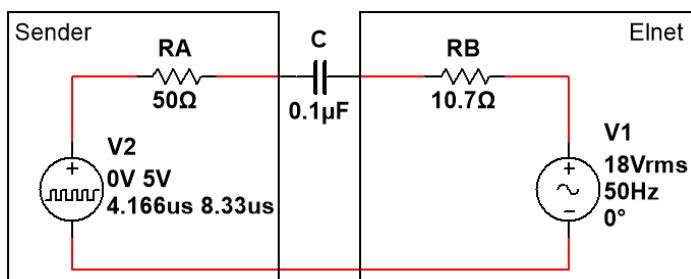
R4 har en bestemt værdi, som kan udregnes ved at måle på spændingen over transformeren uden og med belastning. Derudover er der i applikationsnoten, givet nogle forslag til værdier for sender kredsløbets modstande og kondensator.

Der vil i første omgang blive taget udgangspunkt i disse applikationsnote værdier, for at kunne analysere hvordan de påvirker kredsløbets output, og derefter vil der blive beregnet nogle værdier som er bedre egnet til vores formål.

8.1.1.1 Højpas filter Hardwaredesign

Analyse af sender kredsløb med applikationsnote værdier

For at bestemme værdierne for RA, det som svarer til R3 i det kredsløb på figur 27 og C som svarer til C1, skal der først opstilles to Thevenin-ækvivalent kredsløb:



Figur 27 - Sender og elnet system med applikationsnote værdier.

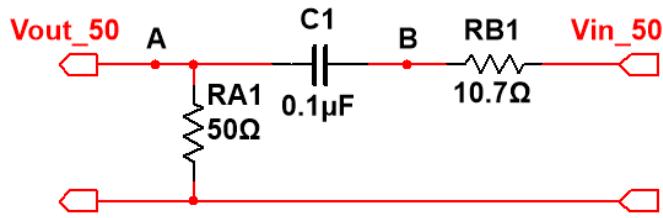
Der tages udgangspunkt i de værdier som er givet i applikationsnoten, hvor RA er 50Ω , C er $0.1\mu F$. RB er den indre modstand i 18V transformeren, udregnet til $\sim 10.7\Omega$:

$$AC_{load} := 21.62 \text{ V} \quad AC_{open} := 21.83 \text{ V} \quad R_{load} := 1100 \Omega$$

$$R_{int} := \left(\frac{AC_{open}}{AC_{load}} - 1 \right) \cdot R_{load} = 10.685 \Omega$$

Da der er to kilder, skal der bruges superposition til at lave de 2 kredsløb.

Først analyseres påvirkningen fra elnettet i punktet A:



Figur 28 – Sender kredsløb, set fra elnettet

Der skal opstilles en overføringsfunktion for hvordan elnettet påvirker punktet A. Der er her tale om en spændingsdeler, og overføringsfunktionen er derfor:

$$G1(s) := \frac{RA}{\frac{1}{C \cdot s} + RB + RA}$$

Hvis man omskriver det, så det står på standard form:

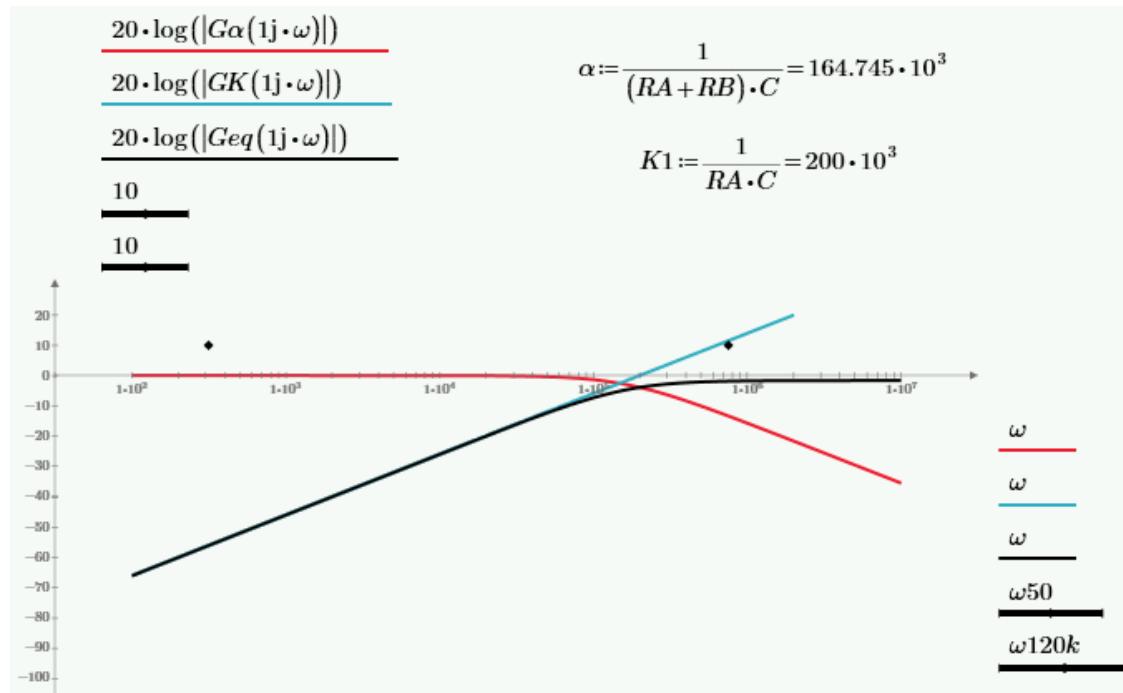
$$G1(s) := \frac{\alpha}{s + \alpha} \cdot \frac{s}{K1} = \frac{\frac{1}{(RA+RB) \cdot C}}{s + \frac{1}{(RA+RB) \cdot C}} \cdot \frac{s}{\frac{1}{RA \cdot C}}$$

Hvor α og $K1$ er:

$$\alpha := \frac{1}{(RA+RB) \cdot C} = 164.745 \cdot 10^3 \quad K1 := \frac{1}{RA \cdot C} = 200 \cdot 10^3$$

Ved at definere de reelle værdier, kan man grafisk vise hvordan forskellige frekvenser bliver påvirket i punktet A.

Hvis man plotter påvirkningen med et bodeplot:



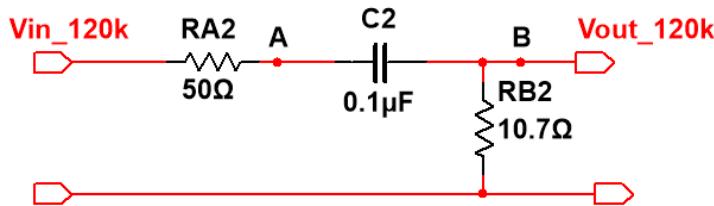
Figur 29 - Bodeplot for frekvenspåvirkningen i punktet A. Applikationsnote værdier, set fra elnettet.

Beregnet dæmpning

Ved :	Amplitudepåvirkning, i dB :	Antal gange dæmpning:
50 Hz	$20 \cdot \log((Geq(\omega_{50} \cdot 1j))) = -56.078$	$ Geq(\omega_{50} \cdot 1j) = 0.0016$
120 kHz	$20 \cdot \log((Geq(\omega_{120k} \cdot 1j))) = -1.887$	$ Geq(\omega_{120k} \cdot 1j) = 0.8047$

Her er det tydeligt at der er tale om et højpas filter. Filteret får effektivt filtreret de 18V, 50Hz fra, og lader de 120kHz passere. Da dette bodeplot er en udregning for hvordan sender kredsløbet bliver påvirket af elnettet ved forskellige frekvenser, er det mest interessant at kigge på hvor meget filteret dæmper de 50Hz, i dette tilfælde er det med 56 decibel. Det kan også ses at filtret også har en påvirkning på max amplituden et signal kan have efter filteret.

Derefter skal påvirkningen i punktet B, set fra elnettet også analyseres. Dette er interessant da der ønskes en lav dæmpning af vores 120kHz burst signal.



Figur 30 - Påvirkningen af punktet B, set fra 120kHz generatoren

Overføringsfunktionen for hvordan de 120kHz påvirker elnettet i punktet B, er næsten den samme som for punktet A, her er det også en spændingsdeler, dog hvor man er interesseret i spændingsfaldet over RB:

$$G2(s) := \frac{RB}{\frac{1}{C \cdot s} + RB + RA}$$

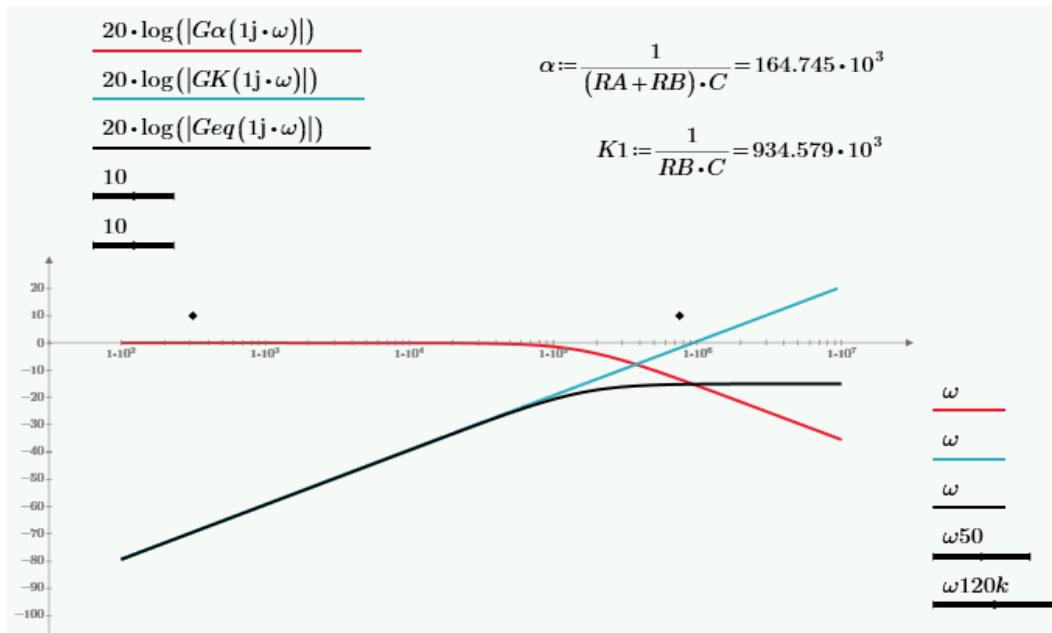
Hvis man omskriver det til standardform:

$$G2(s) := \frac{\alpha}{s + \alpha} \cdot \frac{s}{K2} = \frac{\frac{1}{(RA+RB) \cdot C}}{s + \frac{1}{(RA+RB) \cdot C}} \cdot \frac{s}{RB \cdot C}$$

Hvor α og $K2$ er:

$$\alpha := \frac{1}{(RA+RB) \cdot C} = 164.745 \cdot 10^3 \quad K2 := \frac{1}{RB \cdot C} = 934.579 \cdot 10^3$$

Med de samme definerede værdier, som ovenfor, kan amplitudepåvirkningen beregnes, og plottes:



Figur 31 - Bodeplot for applikationsnote værdier, set fra elnettet.

Beregnet dæmpning

Ved :	Amplitudepåvirkning, i dB :	Gange dæmpning af spænding:
<u>50 Hz</u>	$20 \cdot \log((Geq(\omega_{50} \cdot 1j))) = -69.469$	$ Geq(\omega_{50} \cdot 1j) = 0.0003$
<u>120 kHz</u>	$20 \cdot \log((Geq(\omega_{120k} \cdot 1j))) = -15.279$	$ Geq(\omega_{120k} \cdot 1j) = 0.1722$

Som det kan bemærkes på Bode plottet, så er de 120kHz dæmpet væsentligt grundet filterets påvirkning på max amplituden. Dette 120kHz signal kan i modtagerkredsløbet forstærkes hvis det er nødvendigt, men med en mindre dæmpning af de 120kHz, kan man gøre brug af en mindre forstærkning.

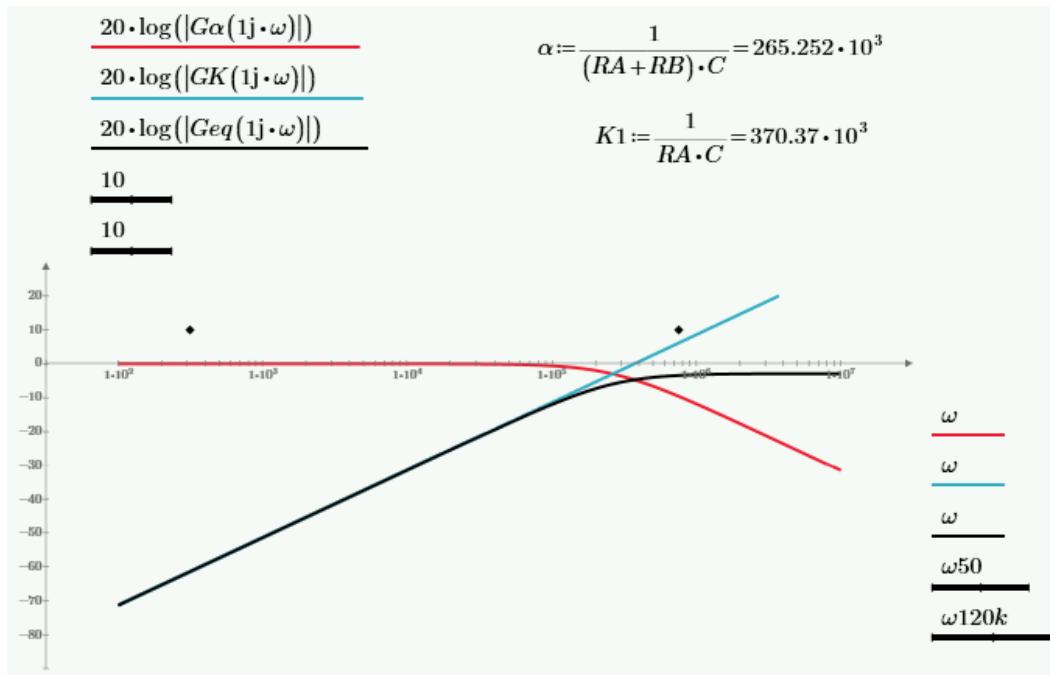
Analyse af sender kredsløb med valgte værdier

Ved brug af udregnede komponentværdier, kan vi mindske dæmpningen på de 120kHz. Den maksimale amplitudepåvirkning på signalerne, er bestemt af hvor meget højere K1/K2 er end alpha. Derfor, hvis K værdien kan mindske, samtidigt med at alpha værdien forhøjes, så de ligger tættere på hinanden, bliver dæmpningen mindre. Dog fordi at begge led af overføringsfunktionen afhænger af størrelsen på C, vil der altid være den samme gange afstand imellem alpha og K, derfor er det værdien af RA der bliver ændret.

Ved at bestemme C til at være det samme som applikationsnoten, og bestemme RA til at være tættere på RB, vil alpha blive en højere værdi, og derfor ikke dæmpe 120kHz signalet ligeså meget.

En modstand for RA(R3) på **27Ω** er blevet valgt, da det er tættere på den indre modstand for elnettet, RB(R4), på de **10,7Ω**.

Påvirkningen i punktet A vil endnu engang blive set på, men denne gang med den udregnede værdi for RA:



Figur 32 - Bodeplot for påvirkningen i punkt A, udregnet værdi.

Beregnet dæmpning

Ved :

Amplitudepåvirkning, i dB :

Gange dæmpning af spænding:

50 Hz

$$20 \cdot \log(|G1(\omega_{50} \cdot 1j)|) = -61.43$$

$$|G1(\omega_{50} \cdot 1j)| = 0.0008$$

120 kHz

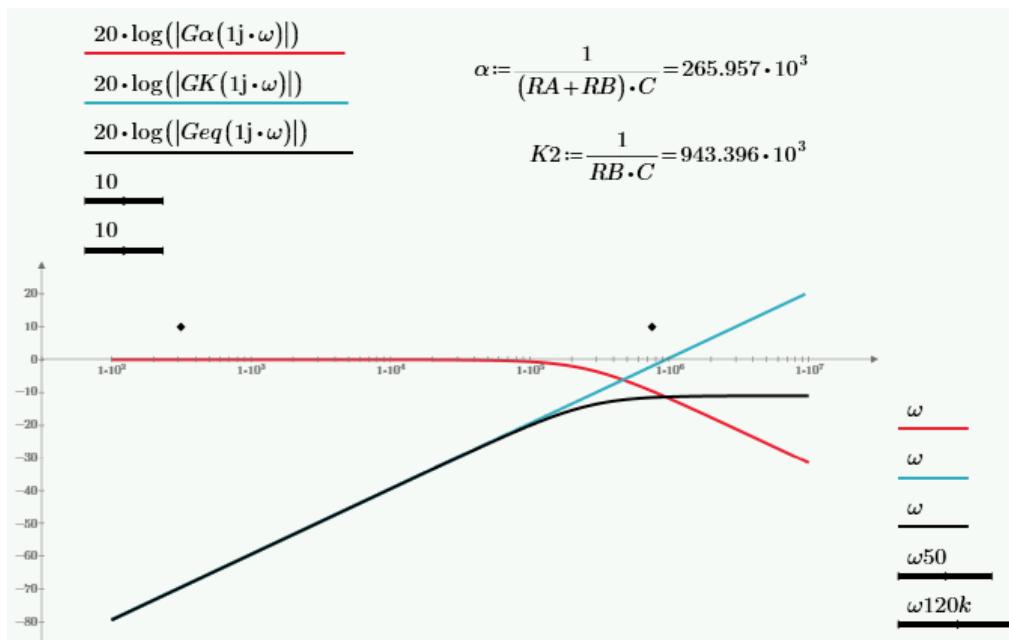
$$20 \cdot \log(|G1(\omega_{120k} \cdot 1j)|) = -3.406$$

$$|G1(\omega_{120k} \cdot 1j)| = 0.6756$$

Som det kan ses, så bliver de 50Hz dæmpet med ~61 decibel, dette svare til at der efter højpas filteret burde være en maksimumamplitude på:

$$30 \text{ V} \cdot 0.0008 = 24 \text{ mV}$$

Herefter skal påvirkningen i punktet B også beregnes med den nye værdi for RA:



Figur 33 - Bodeplot for de nye valgte værdier, set fra elnettet.

Beregnet dæmpning

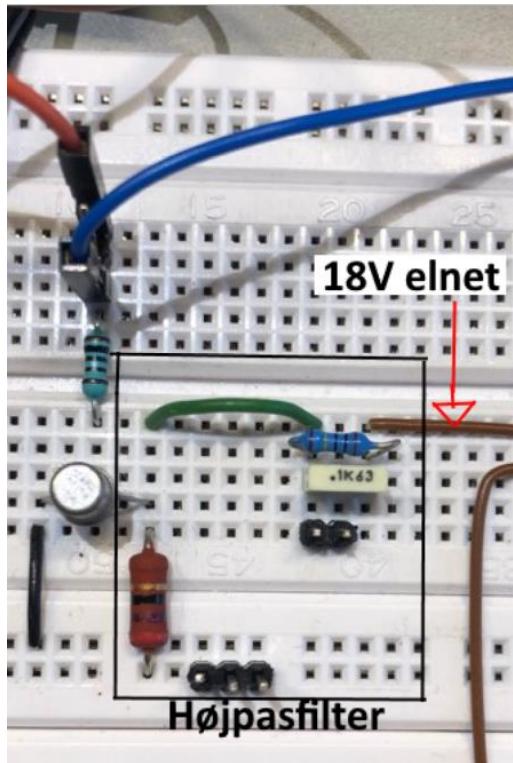
Ved:	Amplitudepåvirkning, i dB:	Antal gange dæmpning:
50 Hz	$20 \cdot \log((Geq(\omega_{50} \cdot 1j))) = -69.469$	$ Geq(\omega_{50} \cdot 1j) = 0.0003$
120 kHz	$20 \cdot \log((Geq(\omega_{120k} \cdot 1j))) = -11.446$	$ Geq(\omega_{120k} \cdot 1j) = 0.2677$

Med de beregnede værdier for højpas filterets C(C1), og RA(R3), opnås der en væsentligt lavere dæmpning af de 120kHz, mens de 18V 50Hz ikke påvirker sender kredsløbet væsentligt. Dette giver et signal med en højere amplitude når det skal behandles af modtager kredsløbet.

Udregningen for dæmpningen af de 120kHz set fra elnettet, er i bedstefald et estimat, og er kun brugt til at estimere en mere passende værdi for RA. Grunden til dette er, kun når transistoren er slukket, er der en modstand på 27Ω . Når transistoren tænder, og tillader en direkte forbindelse til stel, vil amplitudekarakteristikken ændre sig, da modstanden går mod 0Ω .

8.1.1.2 Højpas filter Implementering

Som det ses af vores implementering på figur 34, kobler vi vores $1M\Omega$ modstand parallelt med vores kondensator. Dette gøres for at aflade kondensatoren idet systemet bliver slukket.

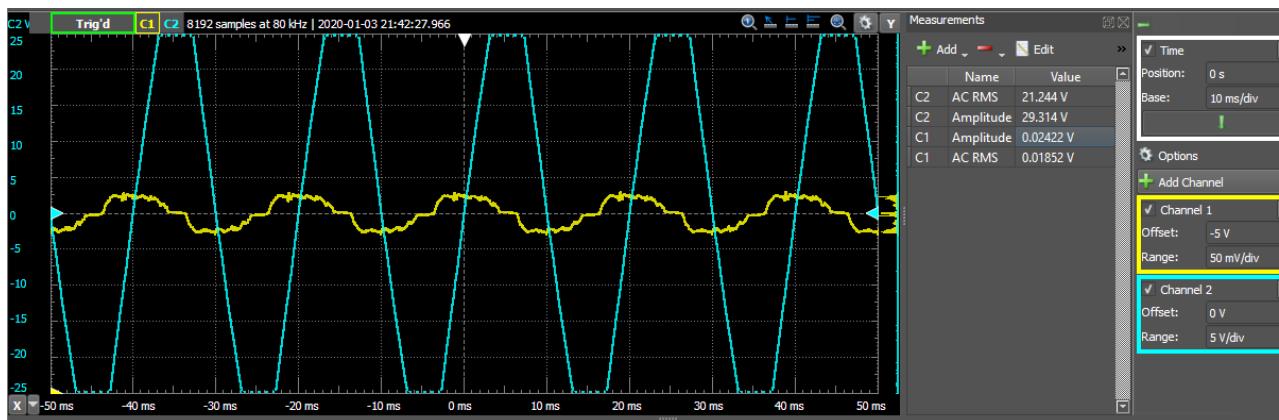


Figur 34 - Sender kredsløbets højpas filter

På figur 34 ses implementering af højpas filteret i sender kredsløbet, dog uden den indre modstand fra elnettet. Da RA(R3)'s modstand er meget lav, på de 27Ω , betyder det at der med de 5V koblet, løber 0,185A igennem den, det svare r til 0,925W, og det kan de almindelige modstande ikke holde til. Der er i implementeringen derfor brugt en lidt mere hårdføre modstand, som kan klare op mod 2W.

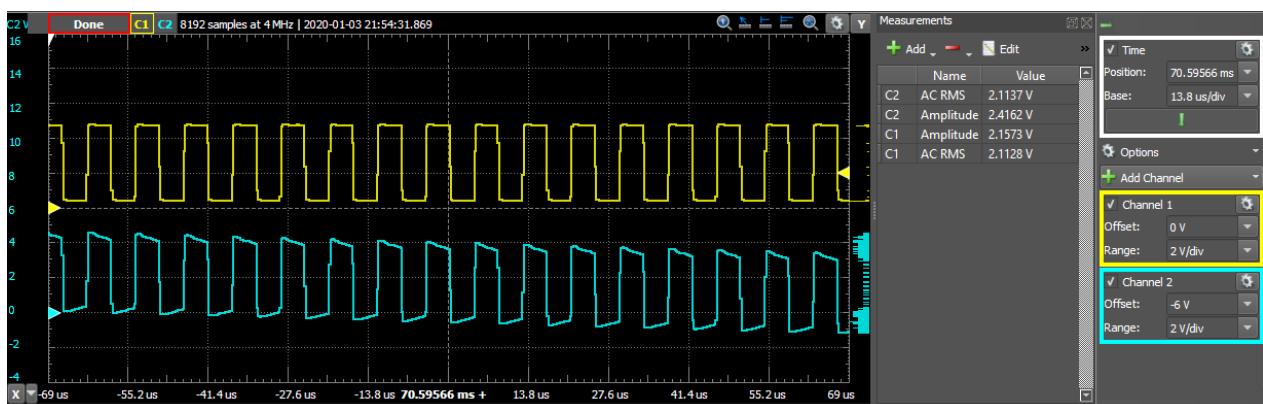
8.1.1.3 Højpas filter Modultest

Højpas filteret har ansvaret for at filtrere de 50Hz væk, og lade de 120kHz passere. Modultesten for højpas filteret går derfor ud på at måle hvad der er tilbage af de 18V, 50Hz efter højpas filteret.



Figur 35 – 18V 50Hz påvirkning. Blå: Inden højpas filter, Gul: Efter højpas filter. Range Blå: 5V / div, Range Gul: 50mV / div, -5V offset. Base: 10ms / div.

Det kan måles at der efter højpas filteret er en amplitude på de 50Hz, på 0,02422V, eller 24mV. Dette stemmer overens med beregningerne fra højpas filterets overføringsfunktion, som efter beregningerne skulle dæmpe 50Hz med -61,4 decibel, eller 0,0008 gange



Figur 36 - 120kHz signalet. Blå: Efter højpas filteret oven i de 50Hz. Gul: Inden højpas filteret. Range: 5V / div for begge. Blå: offset -6V

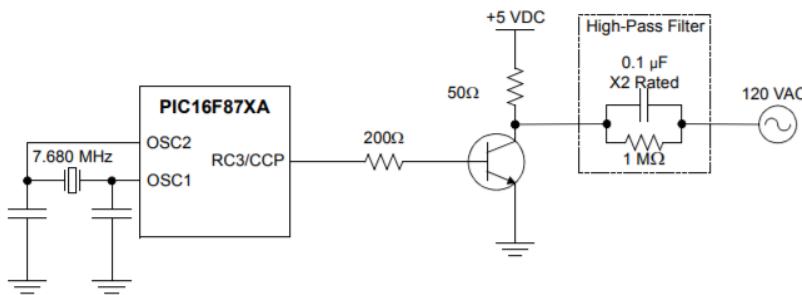
Da der som nævnt i højpas filterets designel, er beregningen af påvirkningen på de 120kHz et estimat. Det kan ses på figur 36 at de 120kHz ikke er dæmpet væsentligt.

8.1.2 120kHz generator

120kHz generator ledet har ansvaret for at generere det signal som bliver sendt videre ud på elnettet. Dette led består af en transistor, hvor der er koblet 5V til collector benet, og en Arduino til basebenet. Mikrocontrolleren har ansvaret for at generere et 120kHz PWM-signal som bruges til at tænde og slukke transistoren, og derved generere det 120kHz signal der

sendes på elnettet. Grunden til at man gør det på denne måde, er for at undgå at overbelaste mikrocontrolleren og transistoren.

Der tages udgangspunkt i det kredsløb givet i applikationsnoten som er blevet udleveret i sammenhæng med projektet, figur 37:



Figur 37 - 120kHz generator givet i applikationsnote

8.1.2.1 120kHz generator Hardwaredesign

Selve 120kHz generator ledet fungere ved at der er koblet 5V til collector benet, og stel til emitter benet. Ved at koble et firkantsignal med en givet frekvens, 120kHz i dette tilfælde, på base benet, tænder og slukker man for transistoren, og collector benet kommer derved til at være 5V når transistoren er slukket, og 0V når transistoren er tændt, frekvensen på skiftet kommer til at være bestemt af frekvensen på det firkantsignal der bliver sendt ind på base benet af mikrocontrolleren.

Da collector modstanden, R3, allerede er blevet udregnet i højpas filter delen, er det eneste komponent der mangler beregning, R1, som er base modstanden.

Da mikrocontrolleren maksimalt kan give 40mA, er base modstandens værdi bestemt ud fra at der skal løbe omkring 20mA ind i basebenet, ved 5V:

$$\frac{5 \text{ V}}{0.02 \text{ A}} = 250 \Omega$$

Dette er nok til at tænde transistoren.

Da modstandsværdien for collector benet, kun er 27Ω vil det sige at der løber 185mA ind i transistoren:

$$\frac{5 \text{ V}}{27 \Omega} = 0.185 \text{ A}$$

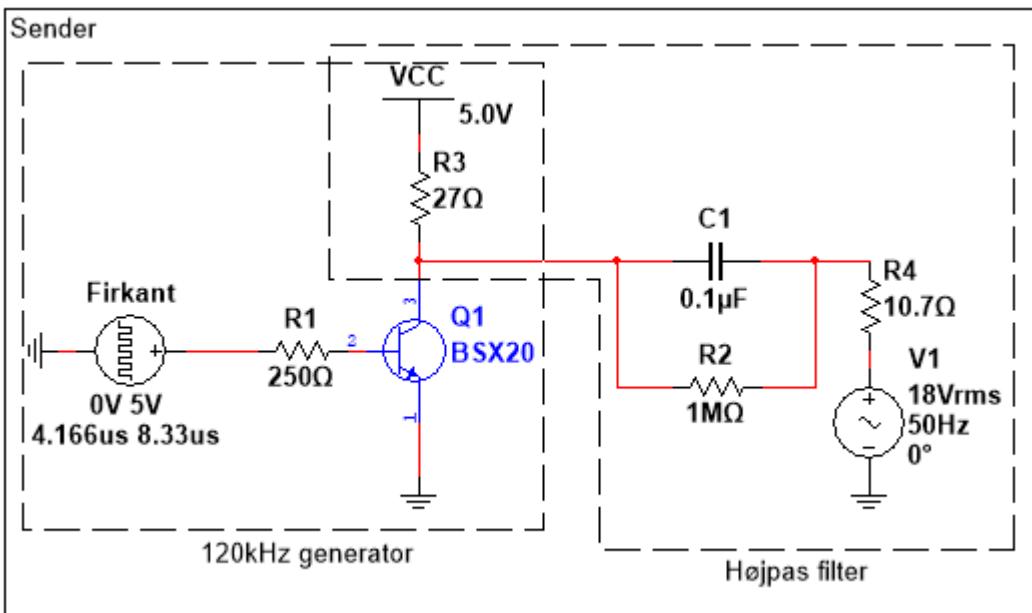
Dette har betydning for den transistor som vælges, da ikke alle transistorer kan klare så meget strøm igennem sig. Derfor er valget af transistor faldet på BSX20. Denne transistor kan klare maksimalt 0,5A peak, meget mere end hvad der løber igennem den.

Collector Peak Current ($t = 10\mu s$)	I_{CM}	0.5	A
--	----------	-----	---

Figur 38 - Datablad for BSX20, absolutte maksimum ratings sektionen

Udover at BSX20'eren kan klare 0,5A peak, er den ifølge databladet også velegnet til højfrekvente skift, hvilket passer godt ind i vores brug af den.

Der kan argumenteres for at 185mA på collector benet, og 20mA på base benet er meget strøm at sende ind i transistoren, bare for at tænde og slukke den. Med højere modstande vil man kunne spare strøm.



Figur 39 - Sender kredsløb med beregnede værdier.

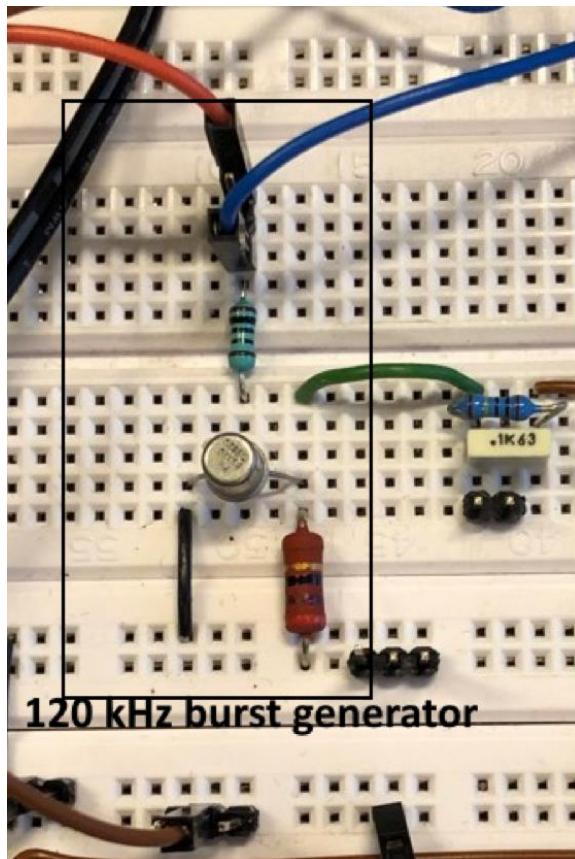
R4 er en del af strømforsyningen, og er derfor ikke reelt et komponent som indgår i sender kredsløbet

8.1.2.2 120kHz generator Implementering

Implementeringen af 120kHz generatoren har haft lidt problemer undervejs, i form af forskellige transistorer kan have meget forskellige duty-cycles. BSX20'eren som er den transistor der er blevet valgt i denne implementering outputter et meget pænt firkantsignal, med en duty cycle på omkring de 50%.

Et tidligere valg af transistor har været en BD139, denne transistor kan godt holde til den mængde strøm der bliver sendt igennem den, men de efterfølgende målinger af

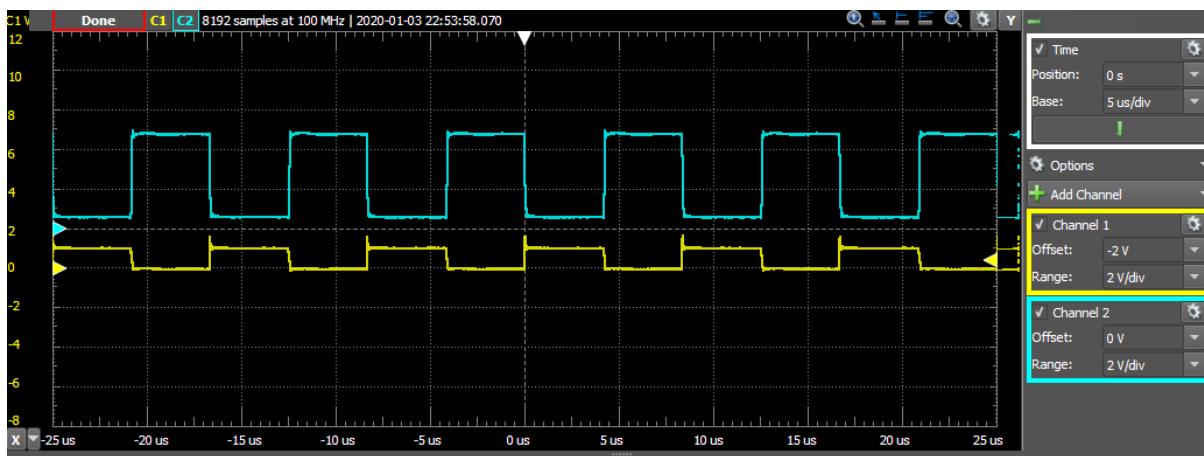
udgangssignalet viste at den producerede et firkant-signal med en meget lav duty cycle, på kun omkring 15% til 20%. Med en lav duty cycle får modtager kredsløbet svært ved at holde signalet over de 3,5V som schmitt-triggeren skal bruge.



Figur 40 - Sender kredsløbets 120 kHz burst generator

8.1.2.3 120kHz generator Modultest

Først skal 120kHz generatoren testes. Dette er gjort med en frekvens generator, som sender et 120kHz firkant signal ind på base benet af transistoren for at tænde og slukke den. For at se det signal der bliver generet, er der blevet målt på collector benet af transistoren.

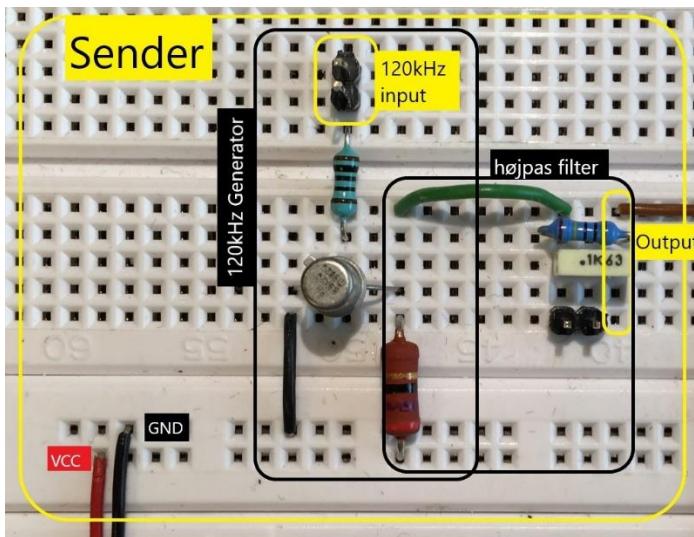


Figur 41 – 120kHz generator. BLÅ: efter transistoren, 2V / div, 0V offset. GUL: inden transistoren, 2V / div, -2V offset. Base: 5us / div

Figur 41, som forventet, er signalets frekvens helt magen til inputtets frekvens, dog inverteret og med en smule støj. Når transistoren er slukket, når det gule signal er lavt, så er der 5V på collector benet, blå signal, og når transistoren er tændt, er der en direkte forbindelse til stel, og derfor er signalet 0V, dog vil der altid være lidt spænding afsat i transistoren, så der er reelt $\sim 0.6V$ i stedet for 0V.

8.1.3 Samlet sender kredsløb

Det samlede senderkredsløb, her inddelt i de beskrevne komponenter



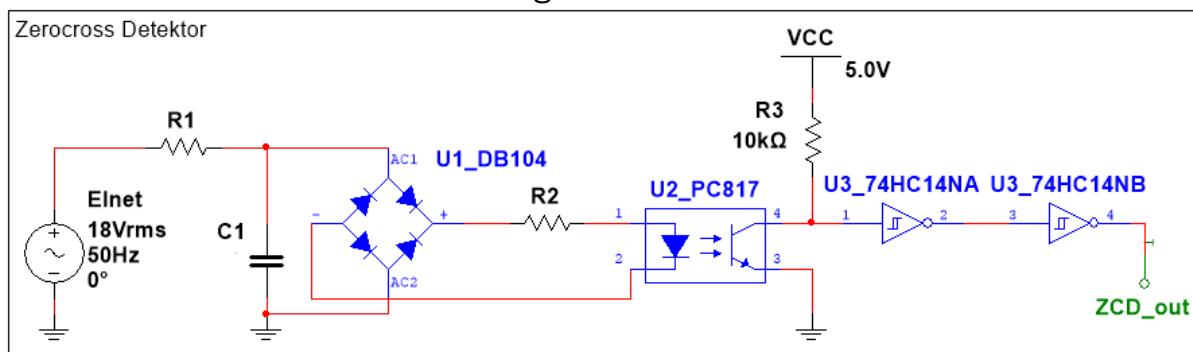
Figur 42 - Samlet senderkreds, dog uden zerocross

Den brune ledning er elnettets ledning, og er koblet videre til modtager kredsløbet og to zero cross kredsløb.

8.2 Zero cross detektor

Zero cross kredsløbet indgår i både modtager, og sender kredsløbet. Zero cross detektoren er det kredsløb der skal fortælle sender- og modtager-controlleren, hvornår vores 18V elnet krydser 0 volt. Denne funktion er vigtig, da det er den der laver den fælles "clock" som X10 signalet, sender- og modtagercontrolleren er synkroniseret efter, så de ved hvornår der skal sendes og lyttes efter signaler.

8.2.1 Zero Cross hardware design



Figur 43 - Zerocross Detektor kredsløb uden værdier

Da R3 er en pull-up modstand skal dens værdi være tilstrækkelig høj, her er $10\text{k}\Omega$ typisk brugt.

Zerocross detektoren i systemet består hovedsageligt af 4 dele.

1. Et lavpas filter som har ansvaret for at filtrerer 120kHz signalet fra, og lade 50Hz passere, med en cut-off frekvens på $\sim 1200\text{Hz}$
2. En DB104, diode bro IC som skal ensrette AC signalet, så der kun er den positive spændingsfald tilbage.
3. En PC817, optokobler IC, som består af en LED og en lysfølsom NPN-transistor.
4. En 74AHC14N, hex inverting schmitt trigger. Denne IC har ansvaret for at behandle det analoge signal som optokobleren sender ud, og gøre det digitalt, så vores mikrokontrollere kan aflæse signalet som enten logisk "1" hvis der er et zerocross eller "0" hvis der ikke er.

R1 og C1 udgør et lavpas filter som har til formål at fjerne 5V, 120kHz signalet fra vores 18V, 50Hz elnet signal, dette gøres fordi 120kHz signalet ikke skal bruges i dette kredsløb og er derfor støj, som potentielt kan påvirke kredsløbet uforudsigligt.

■ Electro-optical Characteristics

($T_a = 25^\circ\text{C}$)

Parameter	Symbol	Conditions	MIN.	TYP.	MAX.	Unit
Forward voltage	V_F	$I_F = 20\text{mA}$	-	1.2	1.4	V

Figur 44 - Datablad udklip for PC817 optokobler

Ifølge databladet for optokobleren, figur 44, er der et spændingsfald over dioden på $\sim 1,2V$, og der ønskes at der løber en strøm på $20mA$, derfor skal dette passe med den totale modstand i den del af kredsløbet. Optokobleren adskiller dens ben 1 og 2, fra ben 3 og 4 elektrisk, da den tænder transistoren mellem ben 3 og 4, med en LED.

■Electrical Characteristics ($T_a=25^\circ C$ Unless otherwise specified)

PARAMETER	SYMBOL	UNIT	TEST CONDITIONS	DB101	DB102	DB103	DB104	DB105	DB106	DB107
Maximum instantaneous forward voltage drop per diode	V_F	V	$I_{FM}=0.5A$				1.05			

Figur 45 - Datablad udklip for DB104 diodebro

I databladet for vores diodebro, figur 45, kan det aflæses at der er et spændingsfald over én af dioderne på $\sim 1,05V$. Spændingsfaldet over en af dioderne skal bruges til udregningen for den total modstand som skal være i kredsløbet.

Den reelle RMS-værdi for de $18V$, $50Hz$ er blevet målt til omkring $21V$ RMS, $30V$ peak, og der ønskes en spænding efter diodebroen på omkring de $5V$.

Derved kommer beregningen for modstandene til at se således ud:

$$AC_{rms} := 21 \text{ V} \quad V_F\text{Diode} := 1.05 \text{ V} \quad V_F\text{LED} := 1.2 \text{ V}$$

$$R_{total} := \frac{AC_{rms} - V_F\text{Diode} - V_F\text{LED} - 5 \text{ V}}{0.02 \text{ A}} = 687.5 \Omega$$

Det er bestemt at der skal være 2 modstande i kredsløbet inden optokobleren, hvor en af dem er brugt til lavpas filteret, og den anden er imellem diodebroen og optokobleren.

R1 bliver bestemt til at være 500Ω og R2 til 180Ω .

For at lave et lavpas filter skal der først vælges en komponentværdi for R1, den er bestemt, og vælges en knækfrekvens. I dette tilfælde er R1 bestemt til at være 500Ω , og knækfrekvensen ønskes at være omkring $1200Hz$, 2 dekader under $120kHz$, da det er væsentligt højere end $50Hz$, og tilpas langt fra $120kHz$.

Da lavpas filteret i kredsløbet, ikke bare er et RC lavpas filter, men et RC lavpas filter med en parallel modstand, som er vores belastning.

Overføringsfunktionen kommer derfor til at se således ud:

$$G_{lp}(s) := \frac{\left(\frac{1}{R2} + C1 \cdot s\right)^{-1}}{R1 + \left(\frac{1}{R2} + C1 \cdot s\right)^{-1}} \xrightarrow{\text{simplify}} \frac{R2}{R1 + R2 + s \cdot C1 \cdot R1 \cdot R2}$$

En simpel spændingsdeler. Hvis man arrangerer det så overføringsfunktionen står på standard form, ser den sådan ud:

$$Glp(s) := \frac{1}{C1 \cdot R1} \cdot \frac{1}{s + \frac{R1 + R2}{R1 \cdot R2 \cdot C1}}$$

Hvor knækfrekvensen, og udregningen af C1 er givet ved:

$$R1 := 500 \quad R2 := 180$$

$$\frac{R1 + R2}{C1 \cdot R1 \cdot R2} = 1200 \xrightarrow{\text{solve}, C1} \frac{17}{5400000 \cdot \pi} = 1.002 \cdot 10^{-6}$$

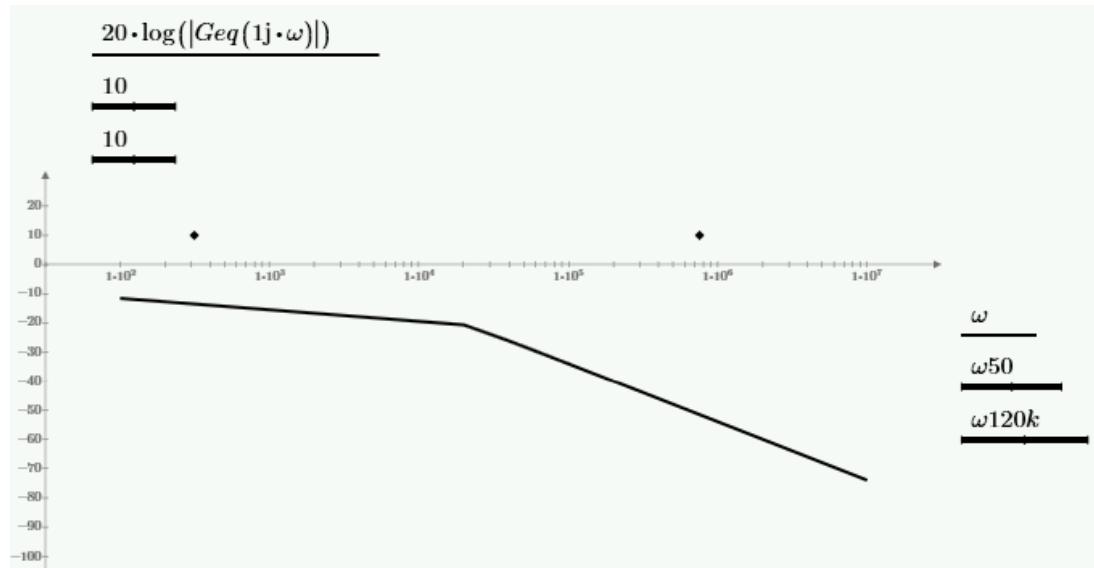
Da det den kondensatorværdi der er tættest på, er $1\mu F$, bliver den brugt.

Dette giver os en reel cut-off frekvens med de nye værdier på:

$$R1 := 500 \quad R2 := 180 \quad C1 := 1 \cdot 10^{-6}$$

$$\frac{R1 + R2}{C1 \cdot R1 \cdot R2} = 1.203 \cdot 10^3$$

Knækfrekvensen er beregnet til at være omkring de 1203Hz, meget tæt på den ønskede frekvens. Bodeplot for filteret:



Den beregnede dæmpning på vores forskellige signaler, efter de har været igennem filtret, er derfor:

Beregnet Dæmpning

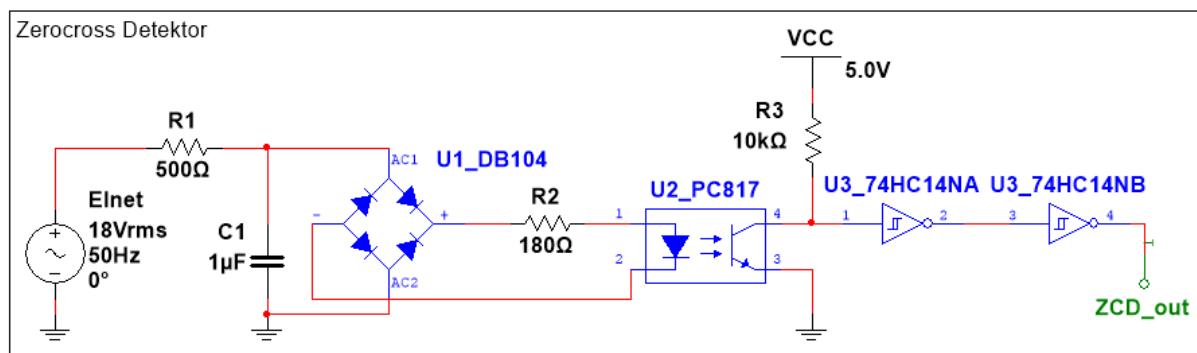
Ved :	Amplitudepåvirkning, i dB :	Antal gange dæmpning :
<u>50 Hz</u>	$20 \cdot \log((G_{eq}(\omega_{50} \cdot 1j))) = -11.552$	$ G_{eq}(\omega_{50} \cdot 1j) = 0.2645$
<u>120 kHz</u>	$20 \cdot \log((G_{eq}(\omega_{120k} \cdot 1j))) = -51.527$	$ G_{eq}(\omega_{120k} \cdot 1j) = 0.0027$

Som det kan ses på udregningen ovenfor, så forsvinder vores 120kHz signal nærmest fuldstændigt. Dog er de 50Hz også dæmpet betydeligt, da belastningen på lavpas filteret, gør at den maksimale amplitude ender med at være på -11,5dB.

Dæmpningen af de 30V peak fra elnettet burde derfor være:

$$30 \text{ V} \cdot 0.2645 = 7.935 \text{ V}$$

Med de indsatte værdier ser zerocross detektorens kredsløb således ud:

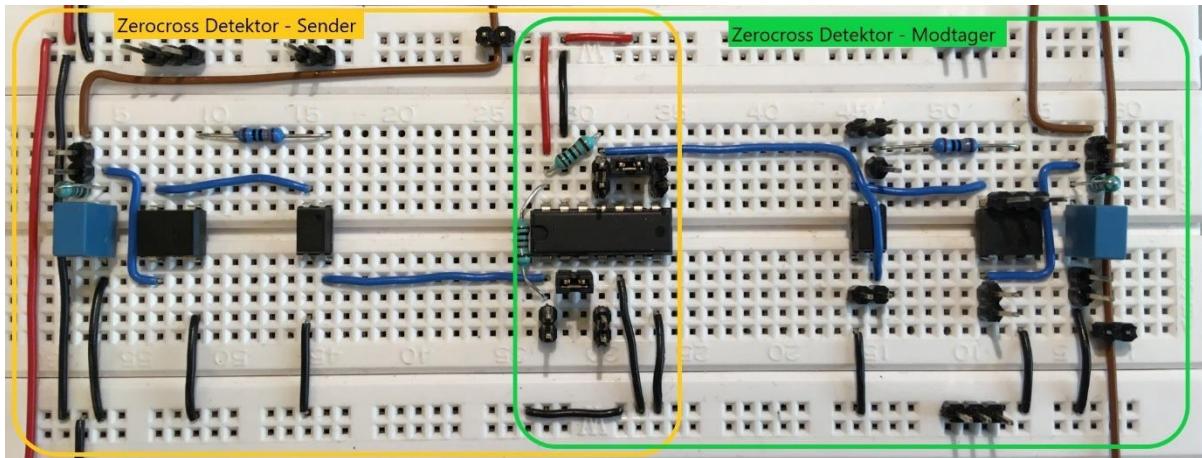


Figur 46 - Diagram over Zerocross Detektor kredsløbet

Reference	Værdi	Antal
R1	500Ω	1
R2	180Ω	1
R3	10kΩ	1
C1	1μF	1
U1	DB104	1
U2	PC817A	1
U3	74AHC14N	1

Tabel 17 - Stykliste til Zerocross Detektor kredsløb

8.2.2 Zero Cross Implementering



Figur 47 - Zerocross Detektor realiseret på fumlebræt

Implementeringen af Zero cross detektoren er lavet ud fra figur 46. Vores fysiske implementering har 2 spejlvendte Zero cross Detektors, på hver deres side af boardet. Den ene er lavet for modtager-kredsløbet, og den anden er lavet for sender-kredsløbet. Vi bruger samme inverterende Schmitt trigger til begge kredsløb, og kører begge kredsløb igennem den 2 gange for at ende med et ikke-inverteret signal.

8.2.3 Zero Cross Modultest

Zerocross detektor modultesten er testet ved brug af Analog Discovery 2 og 18V elnettet. For at måle strømstyrke er der blevet brugt et multimeter.

Det er blevet beregnet i design delen, at der efter lavpas filteret burde være en spænding på omkring 8V grundet dæmpningen fra lavpas filteret.

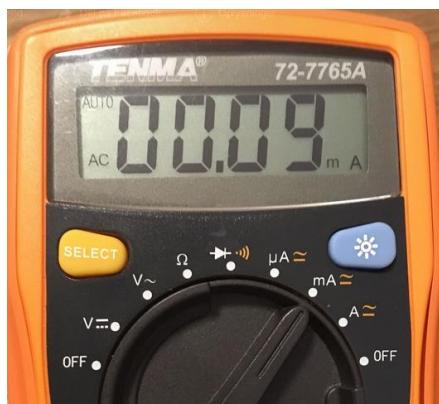


Figur 48 - måling før og efter lavpas filter. Gul: før filteret, Blå: efter filteret. Range: 5V / div.

Dog som det kan ses på figur 47, er der efter filteret omkring 10V, og ikke de beregnede 8V.

Dette kan skyldes upræcise komponentværdier for modstandene R1 og R2, eller den indre modstand i de anvendte dioder, hvilket vil føre til en upræcis beregning af overføringsfunktionen og den tilhørende dæmpning. Hertil er der også påvirkning fra de filtre som indgår i sender og modtager kredsløbene.

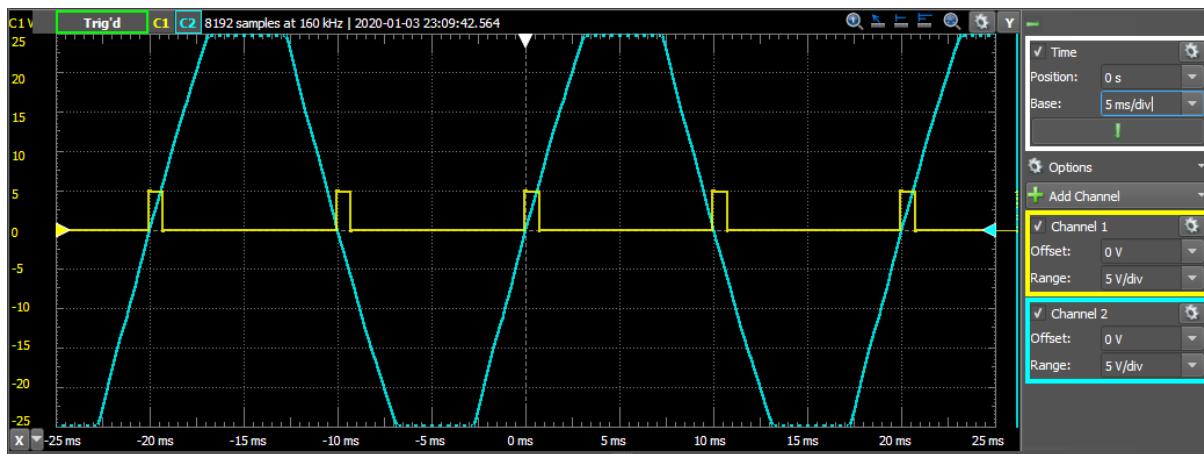
I design delen er der blevet brugt en beregning for de modstande der skal indgå i kredsløbet, for at opnå en strømstyrke på 20mA. Ved en måling med et multimeter er strømstyrken der løber igennem optokobleren målt til at være 9mA, figur 49.



Figur 49 - Multimeter måling af strømstyrken der løber igennem optokobleren

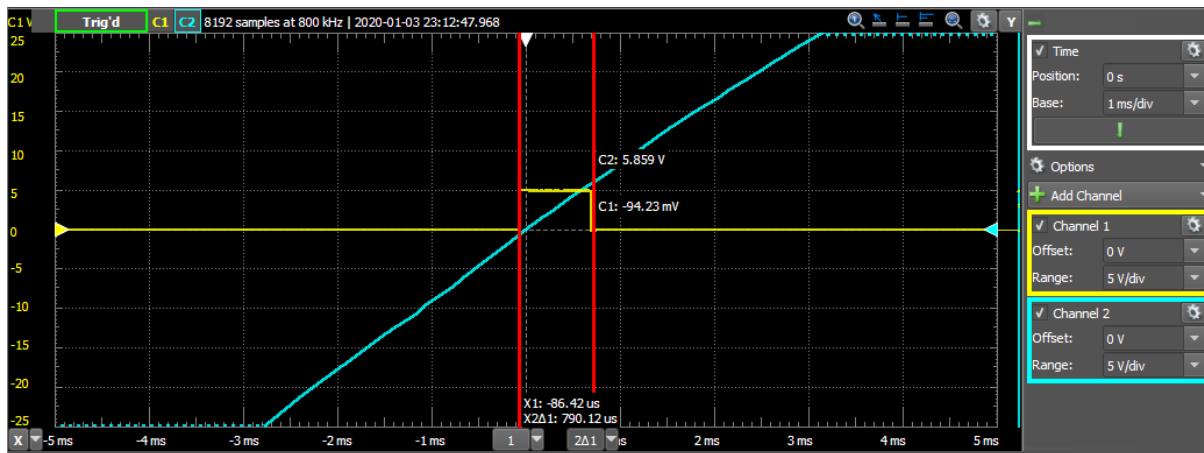
Denne måling er kun halvdelen af den beregnede strømstyrke, hvilket der igen viser at den totale modstand i kredsløbet er væsentligt højere end den beregnede. Udgangen på optokobleren viser dog at de 9mA stadig er nok strøm til at tænde den interne transistor. Dette vil for transistoren betyde at der går længere tid mellem at den slukker, til den tænder igen.

Da hovedmålet med zero cross detektoren er at der skal gives et 5V højt signal på udgangen, i et bestemt tidsrum, når 18V elnettet krydser nul-volts-punktet, er dette også en måling der er relevant at have med.



Figur 50 - Blå: Elnettet, Gul: Zerocross output. Range: 5V / div for begge

På figur 50 kan det ses at vores zero cross detektor fungerer efter hensigten. Den laver et 5V højt signal, når der er et zero cross.

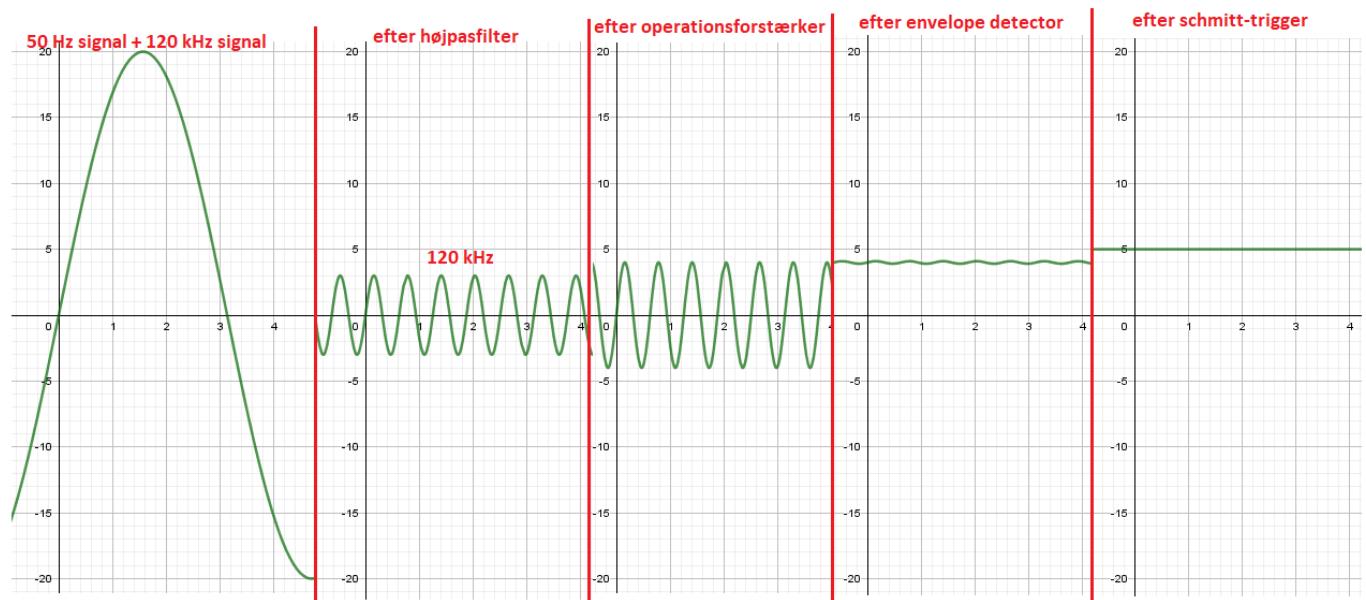


Figur 51 - Måling af tiden for et zerocross output. Blå: elnettet, Gul: Zerocross outputtet. Range: 5V / div for begge

Figur 51. Det høje signal genereret af zerocross detektoren, vare i $790\mu s$, hvilket er længe nok for at en mikrocontroller kan nå at aflæse det, og sende / modtage en kommando.

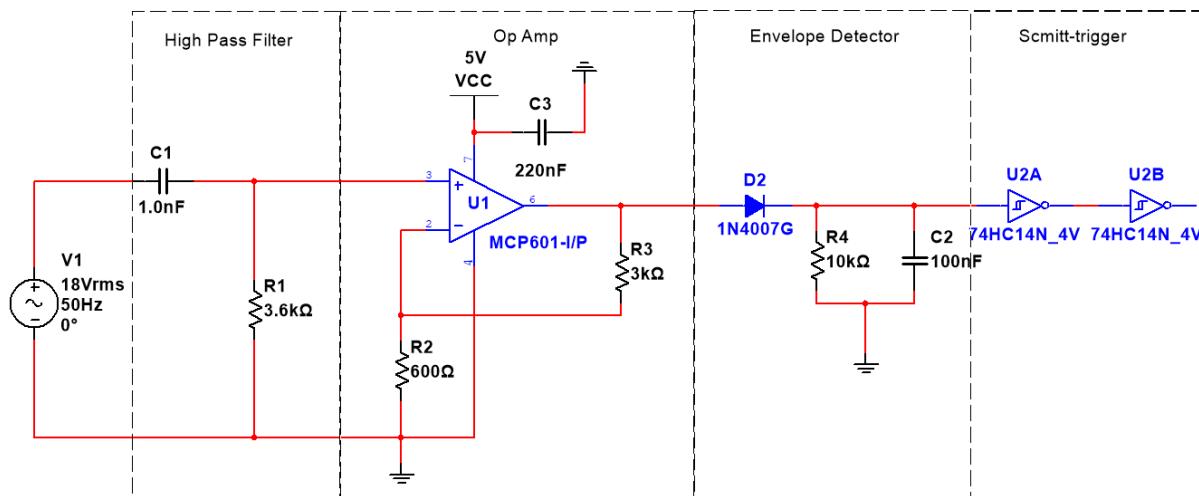
8.3 X10 modtager kredsløb

I dette afsnit vil vi uddybe designet, implementeringen og modultesten af hardwaren for modtager kredsløbet. Modtagerkredsløbets hardware har til funktion at filtrere senderkredsløbets 120 kHz signal fra elnettets 50 Hz signal og forstærke dette signal, således at Arduinoen og derunder softwaren, får et signal med den rigtige frekvens og spænding. For at gøre dette implementeres der et højpasfilter, en operationsforstærker, en envelope detector og 2 inverterende schmitt-triggere. En illustration af hardwarens funktion ses på følgende figur 52.



Figur 52 - signalertyper igennem kredsløbet

På figur 53 ses det samlede modtagerkredsløb. Her ses 4 markerede punkter, heriblandt højpas filter, operationsforstærker, envelope detector og 2 Scmitt-triggere. Disse vil blive udspesificeret i de følgende afsnit.



Figur 53 - kredsløbs design for modtagerkredsløb

Reference	Værdi	Antal
R1	3.6 kΩ	1
R2	600 Ω	1

R3	3 kΩ	1
R4	10 kΩ	1
C1	1 nF	1
C2	100 nF	1
C3	220 nF	1
U1	MCP601- I/P	1
U2	74HC14N	2
D2	IN4007G	1

Tabel 18 - stykliste for modtagerkredsløb

8.3.1 Højpasfilter

Højpasfilteret har ansvaret for at filtrere, altså at dæmpe, de 18V 50Hz fra de 120 kHz. Det vil sige, at vi tillader de 120 kHz at passere igennem, på sammen tid med at vi beskytter kredsløbet fra de 18V.

8.3.1.1 Højpasfilter Hardware design

For modtager kredsløbet skal vi bruge et højpasfilter for at sortere i signalerne fra senderen. Først skal vi have beregnet vores cutoff frekvens. Denne kan findes ud fra følgende formel:

120kHz filtreres fra 50 Hz. Til at gøre dette bygges et simpelt RC-kredsløb:

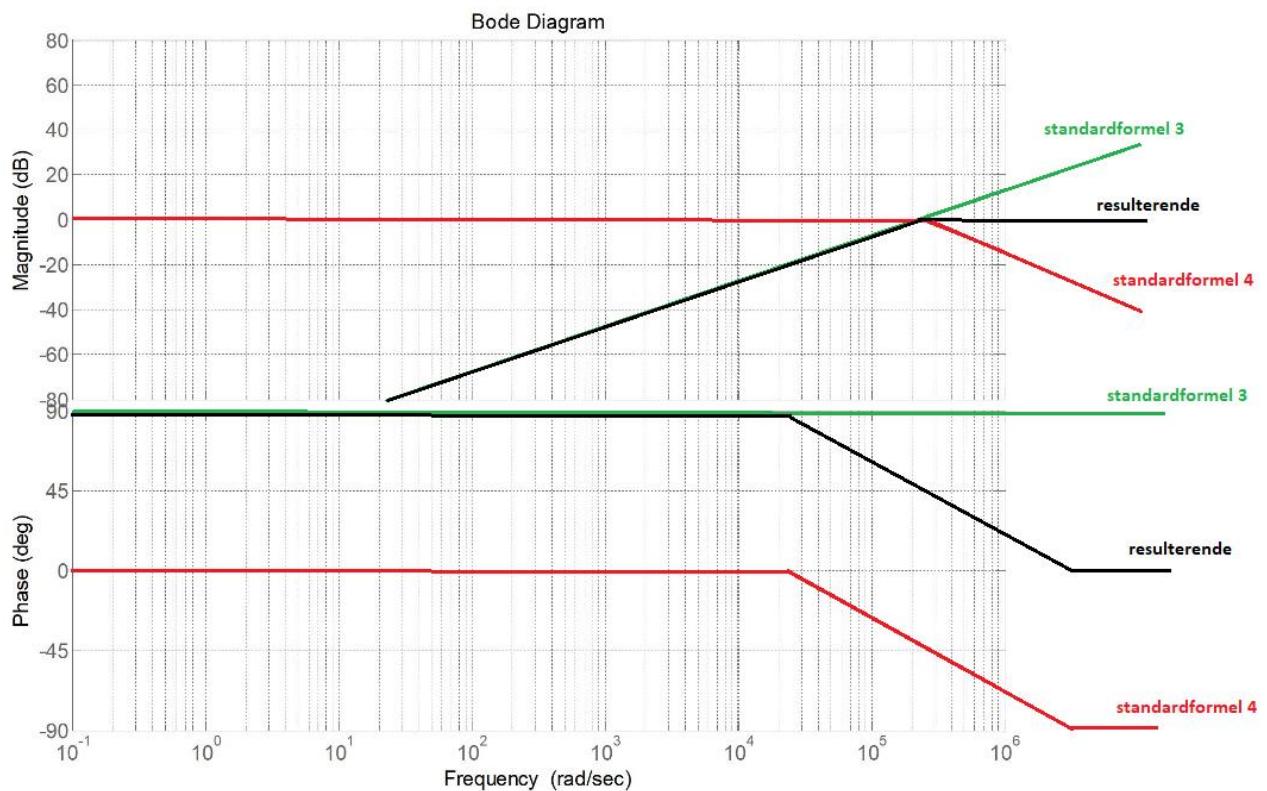
For at mindske dæmpningen af vores 120 kHz signal og samtidig sikre en markant dæmpning af de 50 Hz, så har vi valgt at tage en cutoff frekvens beregnet ud fra en frekvens på 60 kHz:

$$\frac{60 \cdot 10^3}{\sqrt{2}} = 4.243 \cdot 10^4 \text{ Hz}$$

$$\omega_c := \frac{60 \cdot 10^3 \cdot 2 \cdot \pi}{\sqrt{2}} = 2.666 \cdot 10^5 \frac{\text{rad}}{\text{s}}$$

Her ses det, at vores beregnede cutoff frekvens er på 42.43 kHz.

For at kunne beregne vores komponenter i højpas-filteret bruger vi standardformlerne 3 og 4 fra bilag 1. Som udgangspunkt tegnes det asymptote bodeplot for højpasfilteret i hånden:



Figur 54 - Bodeplot for højpasfilter

Herefter regnes der videre med overføringsfunktioner:

Formlerne ser således ud:

Der gælder i følge standardformlerne, at $\alpha = \omega_c$, samt at $K = \omega_c$.

$$\alpha := \omega_c = 2.666 \cdot 10^5$$

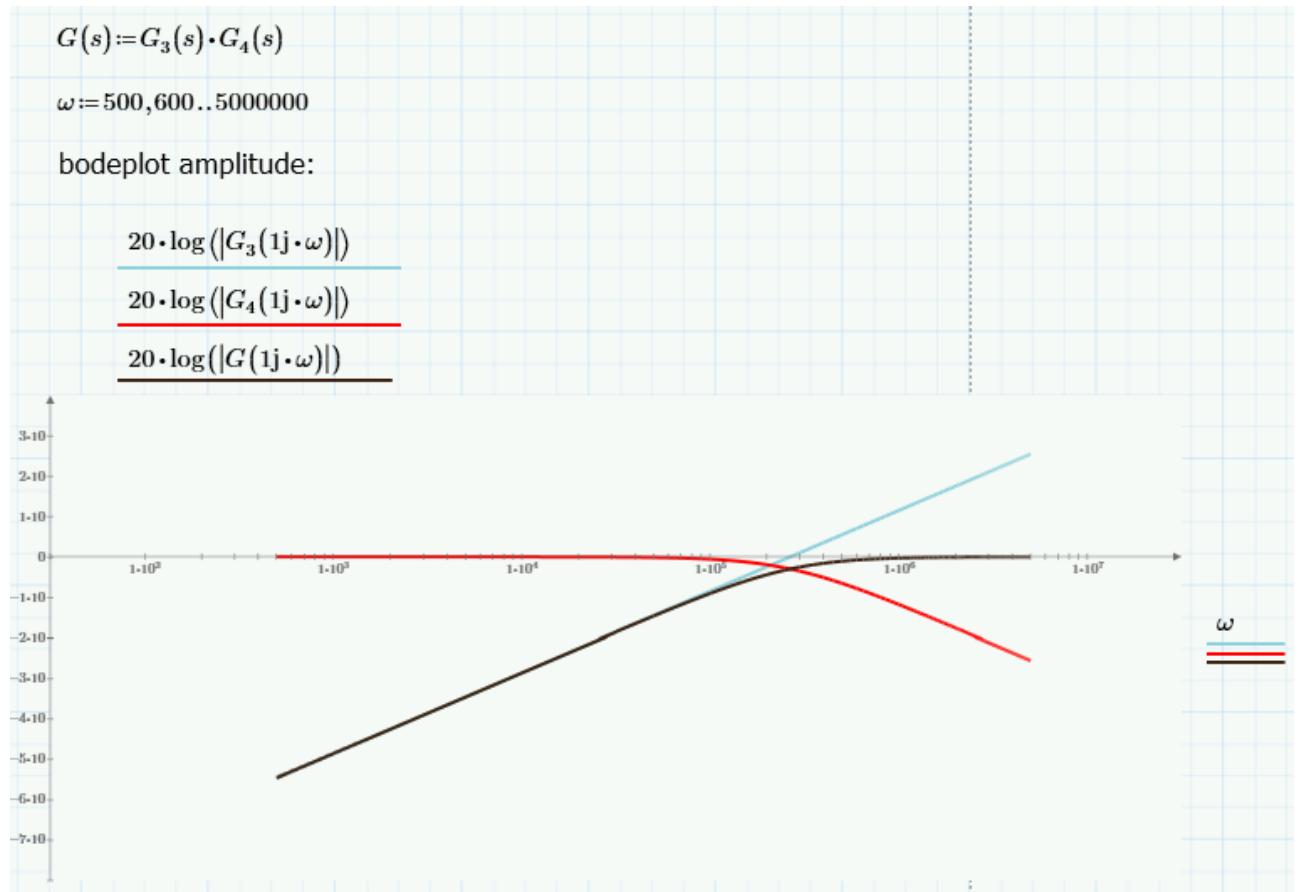
$$K := \alpha$$

$$G_3(s) := \frac{s}{K}$$

+

$$G_4(s) := \frac{\alpha}{s + \alpha}$$

Ud fra overstående overføringsfunktion kan vi nu lave et bode plot for højpasfilteret, samt et faseplot.



Figur 55 - Bode plot for højpas filter

bodeplot fase:



Figur 56 – Fase plot for højpas filter.

Disse stemmer meget godt overens med det tegnede asymptote bodeplot.

Nu opskrives overføringsfunktionen med komponenter. Som udgangspunkt bruges en simpel spændingsdelerformel til at opskrive overføringsfunktionen, hvorefter s isoleres:

$$T(s) := \frac{R}{\frac{1}{C \cdot s} + R} = \frac{R \cdot C \cdot s}{1 + R \cdot C \cdot s} = \frac{R \cdot C \cdot s}{R \cdot C + \frac{1}{s}} = \frac{\frac{R \cdot C \cdot s}{R \cdot C}}{\frac{1}{R \cdot C} + \frac{s}{R \cdot C}} = \frac{s}{\frac{1}{R \cdot C} + s}$$

Overføringsfunktionen med komponenter sammenlignes med overføringsfunktionen på standardform:

$$K := \alpha$$

$$\frac{s}{K} \cdot \frac{\alpha}{s + \alpha} \rightarrow \frac{s}{\alpha + s}$$

Vi ser således at:

$$\alpha = \frac{1}{R \cdot C}$$

Der vælges nu en værdi for kondensatoren, hvorefter modstanden beregnes. Der vælges en kondensatorværdi på 1 nF:

$$C := 1 \cdot 10^{-9} \text{ F}$$

$$R := \frac{1}{R \cdot C} = 2.666 \cdot 10^5 \xrightarrow{\text{solve}, R} 3750.9377344336084021 \Omega$$

Hernæst beregnes dæmpningen af de to 120 kHz og 50 Hz signaler:

Dæmpning af 120 kHz:

$$20 \cdot \log(|G(1j \cdot 120 \cdot 10^3 \cdot 2 \cdot \pi)|) = -0.512 \text{ dB}$$

$$|G(1j \cdot 120 \cdot 10^3 \cdot 2 \cdot \pi)| = 0.94281$$

Dæmpning af 50 Hz:

$$20 \cdot \log(|G(1j \cdot 50 \cdot 2 \cdot \pi)|) = -58.573 \text{ dB}$$

$$|G(1j \cdot 50 \cdot 2 \cdot \pi)| = 0.00118$$

Beregninger med valgte værdier

Med udgangspunkt i den beregnede modstand på 3.75 kΩ, så blev der valgt en modstand på 3.6 kΩ. Denne modstand blev med multimeter målt til 3.568 kΩ:

Vi har således følgende værdier:

$$C := 1 \cdot 10^{-9} \text{ F}$$

$$R := 3568 \Omega$$

Med den nye modstand beregnes knækfrekvensen:

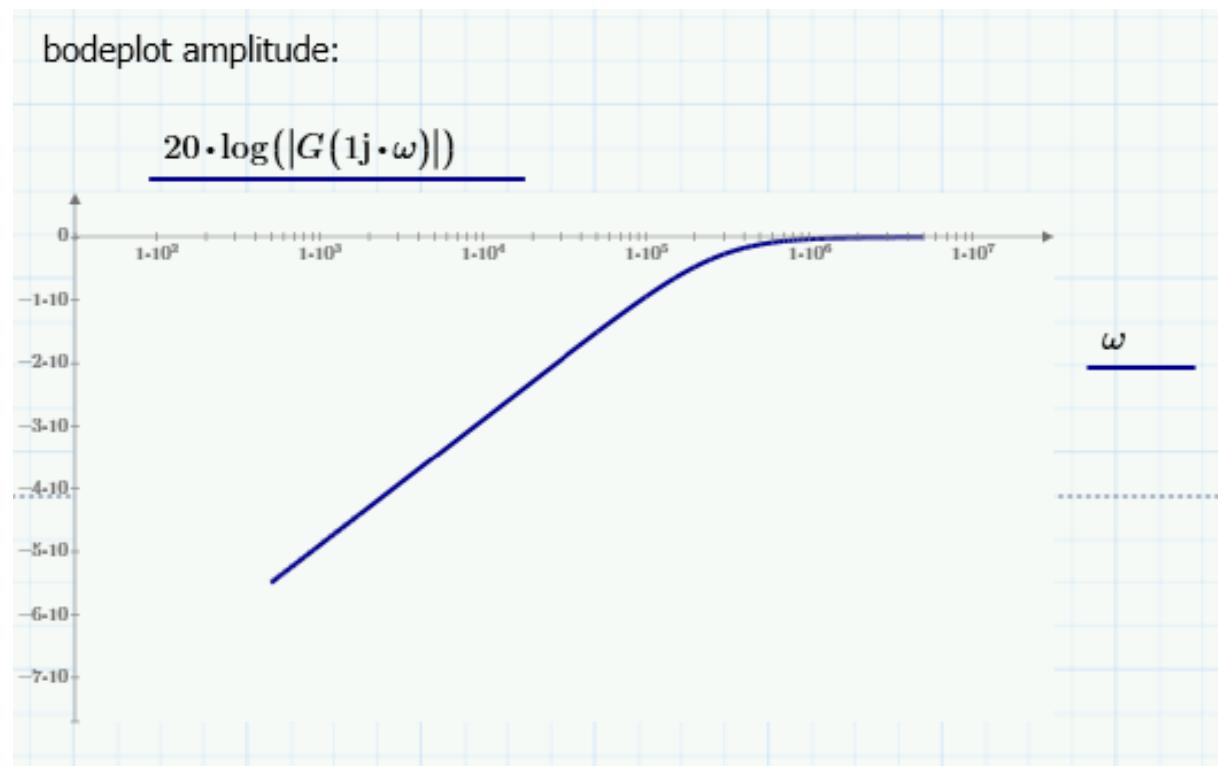
$$\alpha := \frac{1}{R \cdot C} \quad K := \alpha \quad \omega := 500,600..5000000$$

$$G(s) := \frac{s}{K} \cdot \frac{\alpha}{s + \alpha}$$

Der vides fra tidligere, at $\alpha = \omega_c$

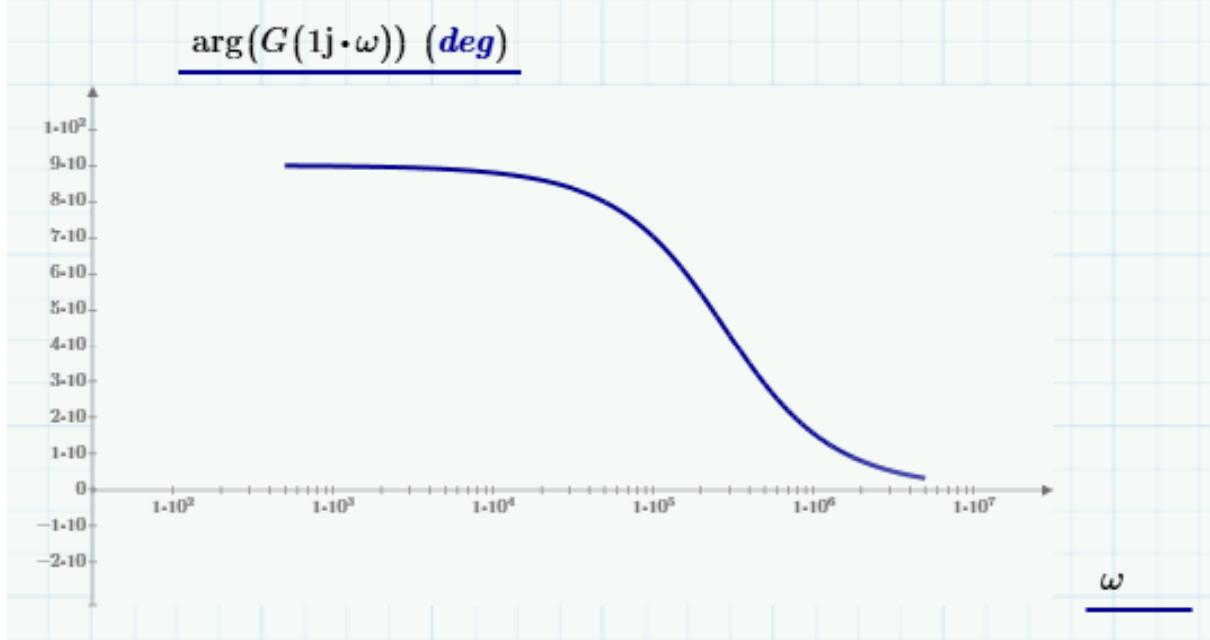
$$\omega_c := \alpha = 2.803 \cdot 10^5 \frac{\text{rad}}{\text{s}}$$

Med overføringsfunktionen defineret, tegnes bode plot og bergnes dæmpning:



Figur 57 - Bodeplot for højpasfilter med reelle værdier

bode plot fase:



Figur 58 - Faseplot for højpasfilter med reelle værdier

dæmpning af 120 kHz:

$$20 \cdot \log(|G(1j \cdot 120 \cdot 10^3 \cdot 2 \cdot \pi)|) = -0.562 \text{ dB}$$

$$|G(1j \cdot 120 \cdot 10^3 \cdot 2 \cdot \pi)| = 0.93734$$

dæmpning af 5V signal med 120 kHz:

$$5 \text{ V} \cdot 0.93734 = 4.687 \text{ V}$$

dæmpning af 50 Hz:

$$20 \cdot \log(|G(1j \cdot 50 \cdot 2 \cdot \pi)|) = -59.009 \text{ dB}$$

$$|G(1j \cdot 50 \cdot 2 \cdot \pi)| = 0.00112$$

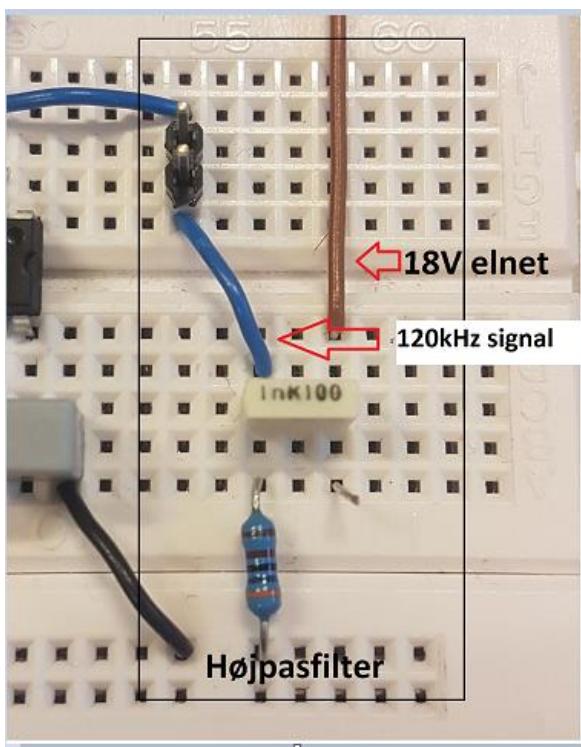
dæmpning af 50 Hz elnet med en amplitude på omkring 30V:

$$30 \text{ V} \cdot 0.00112 = 33.6 \text{ mV}$$

Vi har således vores forventede værdier for dæmpningen af elnettet og vores 5V signal.

8.3.1.2 Højpasfilter Implementering

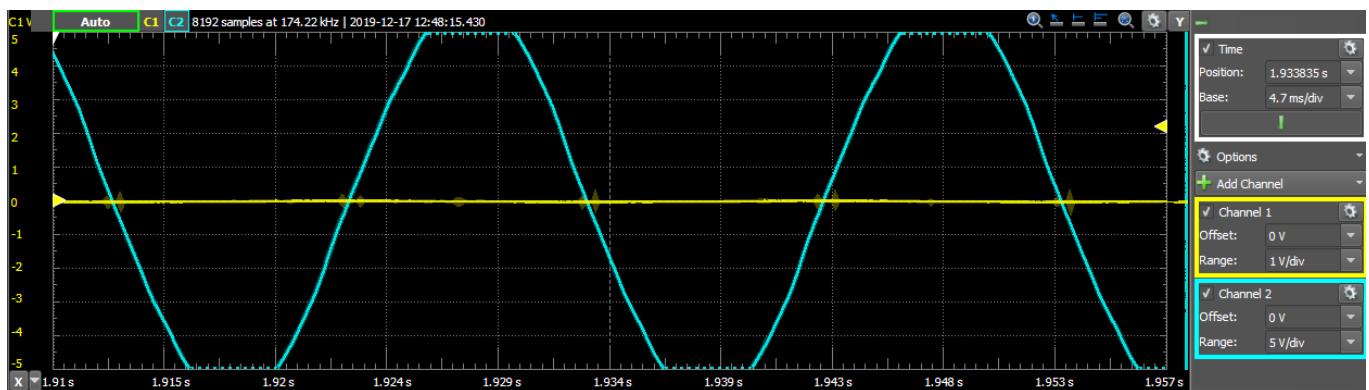
Ved implementeringen af vores højpasfilter ses det, at vi har placeret vores beregnede modstand på 3600 ohm, i serie med vores valgte kondensator på en 1 nF. Her kan det yderligere ses at vi har et 18V elnet input og 120 kHz signal output.



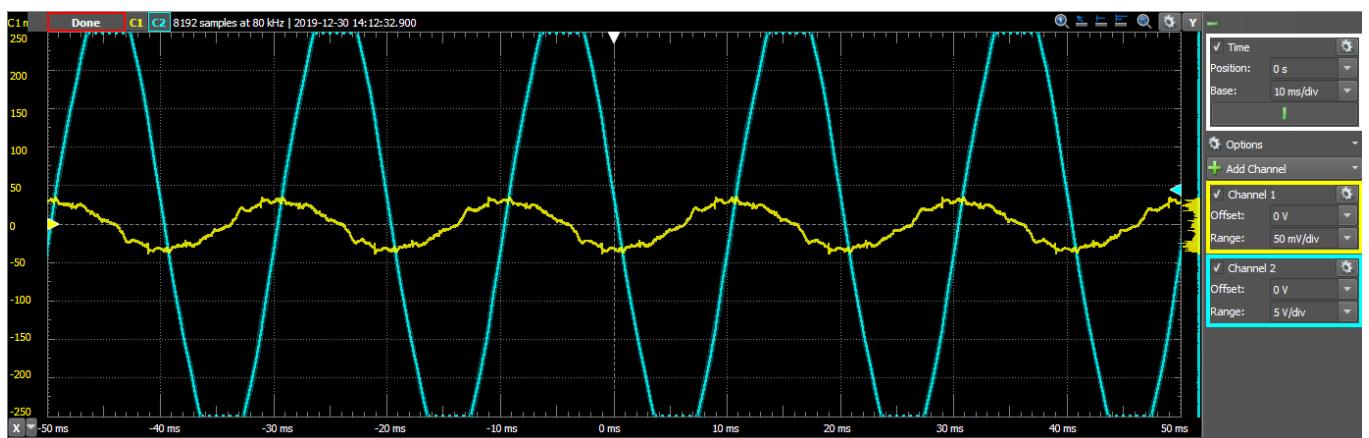
Figur 59 - Implementering af højpasfilter

8.3.1.3 Højpasfilter Modultest

I første omgang testes højpasfilteret med 18V elnettet som det eneste input. Resultatet af dette ses på figur 60, hvor vi har vores blå input, 18V 50 Hz elnet, og vores gule output. Som det fremgår af figuren, så dæmpes de 18V signalet markant.



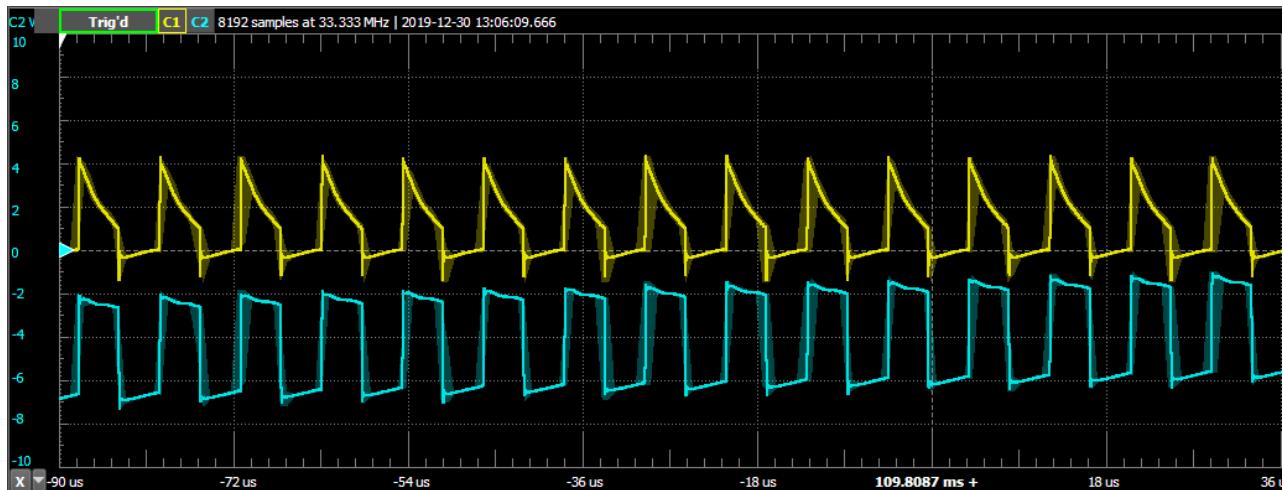
Figur 60 - test af højpasfilteret med 18V 50 Hz som input (channel 2, blå) og output (channel 1, gul). Channel 2 viser 5V pr. div. og channel 1 viser 1V pr. div.



Figur 61 - test af højpasfilteret med 18V 50 Hz som input (channel 2, blå) og output (channel 1, gul). Channel 2 viser 5V pr. div. og channel 1 viser 50 mV pr. div.

På figur 61 er der zoomet ind på outputtet fra figur 60. Her aflæses outputtet af højpasfilteret til at være omkring 40 mV, når kun elnettet er tilsluttet. Dette afviger noget fra den beregnede dæmpning på 33,6 mV, hvilket kan skyldes at amplituden på elnettet ikke er præcist 30V og eventuelt tolerancen på kondensatoren.

Hernæst testes højpasfilteret med 18V elnettet påført de 5V 120 kHz. Som det fremgår af figur 62, så har vi vores gule output, som er vores signal efter det har passeret højpasfilteret. Det blå signal viser 120 kHz 5V signalet der ligger på elnettets 18V 50 Hz. Det fremgår af figuren, at 5V 120 kHz signalet passerer højpasfilteret uden væsentlig dæmpning modsat forrige figur, hvor de 18V 50 Hz blev væsentlig dæmpet.



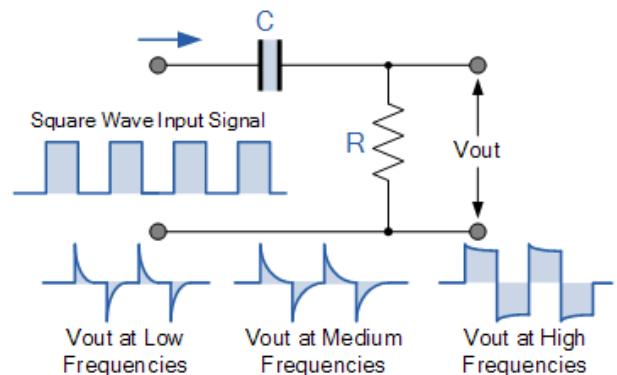
Figur 62 - test af højpasfilteret med 18V påført 120 kHz 5V signal som input (blå). Det gule signal er outputtet.

Der er flere ting at notere ifht. signalet efter højpasfilteret.

Figur 62 illustrerer meget godt, at signalet ikke længere ligner et firkantignal efter filteret. Dette skyldes, at der ved rising edge på et firkantignal er en uendelig stor frekvens⁴, hvor kondensatoren oplades med det samme, hvorefter frekvensen er nul, udelukkende DC, hvilket forårsager at kondensatoren aflader indtil falling edge. Ved falling edge er der en uendelig stor negativ frekvens, som tvinger signalet ned til nul og kondensatoren aflades således på et splitsekund.

Den kluge læser havde umiddelbart forventet, at vores 120 kHz signal efter højpasfilteret ville være spejlet omkring 0 volt i en større grad end vi ser på figur 62, som det er illustreret på figur 63⁵. Dette skyldes, at vores single-supply operationsforstærker, som signalet går over i efter højpasfilteret "dræber" de negative spændinger. På figur 64 ses signalet uden resten af modtagerkredsløbet tilkoblet, hvorpå man ser, at signalet spejles omkring OV.

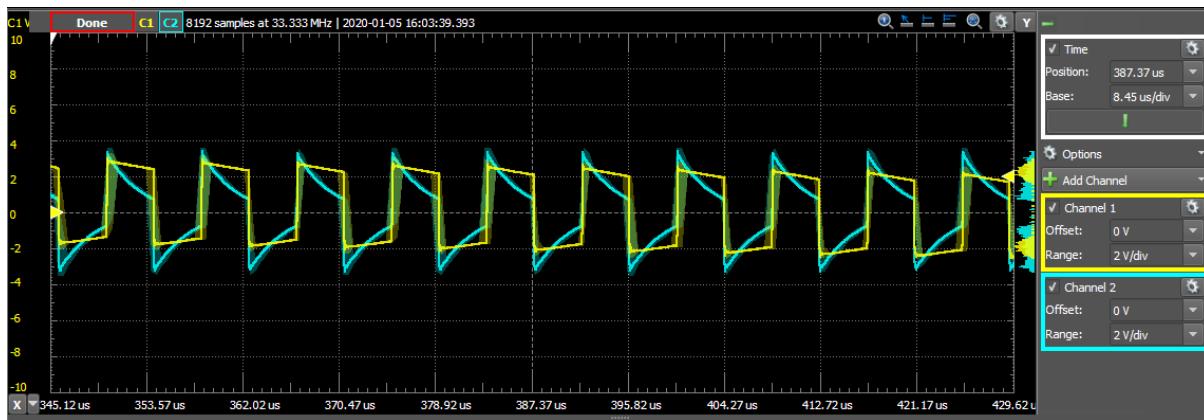
Man kunne med fordel have tilkoblet en diode til ground efter højpasfilteret, således ville der være vished om hvordan og hvorledes de negative spændinger stoppes, samt undgå at ødelægge operationsforstærkeren.



Figur 63 - illustration af forventede output med et firkantignal som input

⁴ https://en.wikipedia.org/wiki/Square_wave

⁵ https://www.electronics-tutorials.ws/filter/filter_3.html



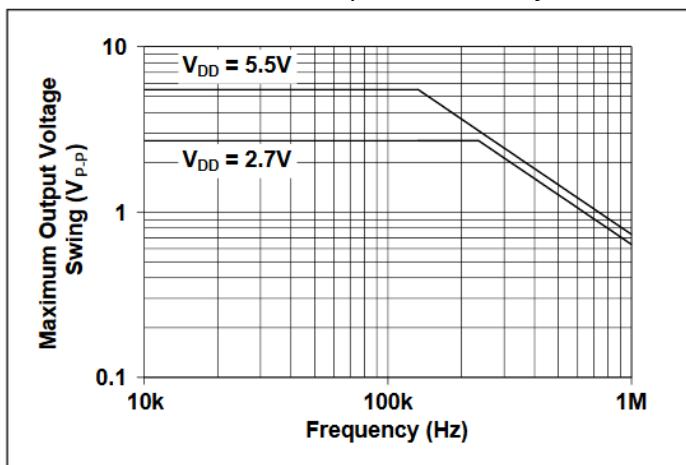
Figur 64 - test af højpasfilter uden resterende modtagerkredsløb tilkoblet. Gul = input, blå = output.

8.3.2 Operationsforstærker

Efter højpasfilteret vil vi have dæmpet senderens 5V 120 kHz signal så meget, at det ikke er garanteret, at et højt signal kommer op over Schmitt-triggerens upper threshold spænding. For at garantere dette implementeres en operationsforstærker efter højpasfilteret.

8.3.2.1 Operationsforstærker Hardware design

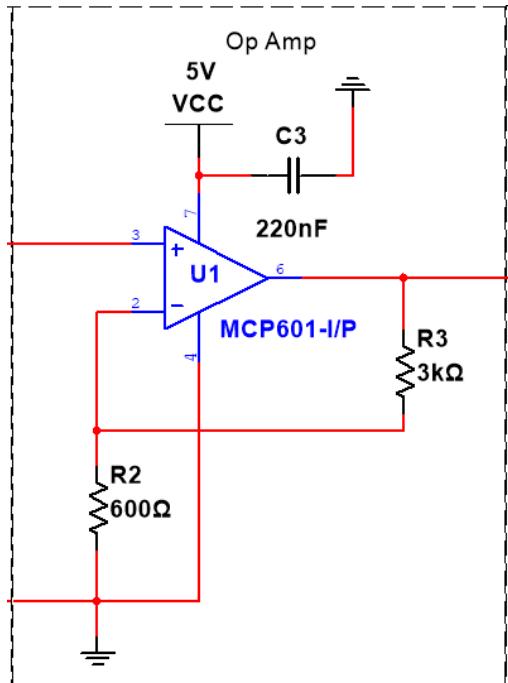
Som operationsforstærker er der valgt en MCP601⁶, som kan fungere både med single og dual supply. Der er i kredsløbet valgt at forsyne forstærkeren med single supply, hvilket betyder at den ikke er egnet til at forstærke negative spændinger. Yderligere gælder der for forstærkeren, at den kan operere ved høje frekvenser, dette ses af figur 65.



Figur 65 - output og frekvens kurvediagram. Se referencelistens nr. 8.

⁶ <http://ww1.microchip.com/downloads/en/DeviceDoc/21314g.pdf>

Der er valgt at designe operationsforstærkeren ikke inverterende. På nedestående figur ses designet for operationsforstærkeren inklusiv komponenter, som beregnes i den efterfølgende del af designet.



Figur 66 - Operations forstærker for modtager

Der er valgt et højt gain på 6, idet vi ved, at outputtet på en rail to rail operationsforstærker ikke kan blive større end forsyningsspændingen og der er således ingen bekymring om, at forstærke for meget.

På det nedenstående ses beregningen af modstanden R3, efter vi har valgt R2 til at være 600Ω og et gain på 6:

$$R_2 := 600$$

Da det er en ikke inverterende operationsforstærker, bruger vi følgende formel:

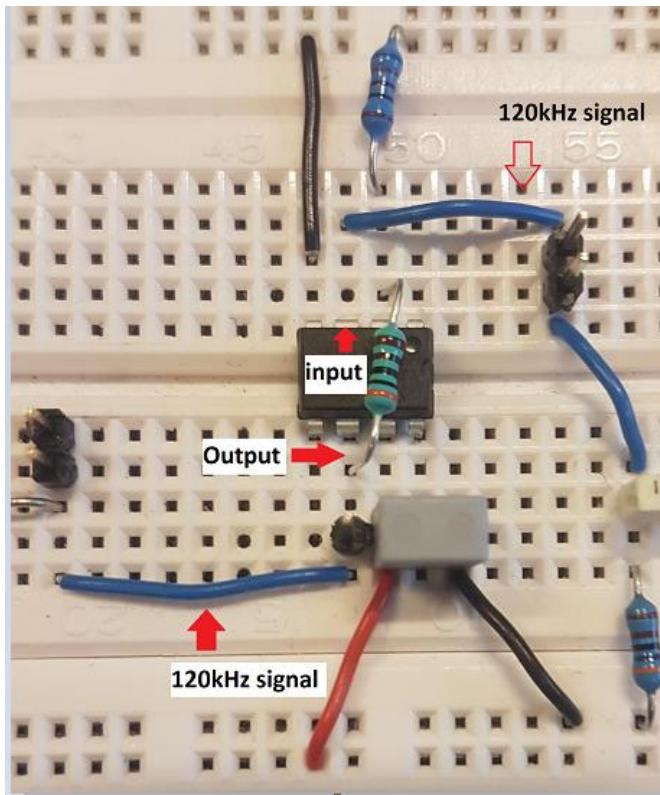
$$A := 1 + \frac{R_3}{R_2} = 6 \xrightarrow{\text{solve } R_3} 3000$$

+

Yderligere indgår der en afkoblingskondensator fra Vcc til ground, som tager eventuelt støj.

8.3.2.2 Operationsforstærker Implementering

Implementeringen af operationsforstærkeren fremgår af figur 68, hvor vi har operationsforstærkeren placeret på vores fumlebræt. Her får vi et 120kHz signal ind på ben 3, hvilket er vores input. Her noteres det, at vi har placeret en 3000 ohm feedback modstand fra vores output ben 6 til ben 2 for således at opnå det ønskede gain på 6.

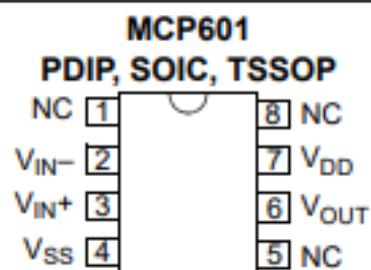


Figur 68 - Implementering af operationsforstærker

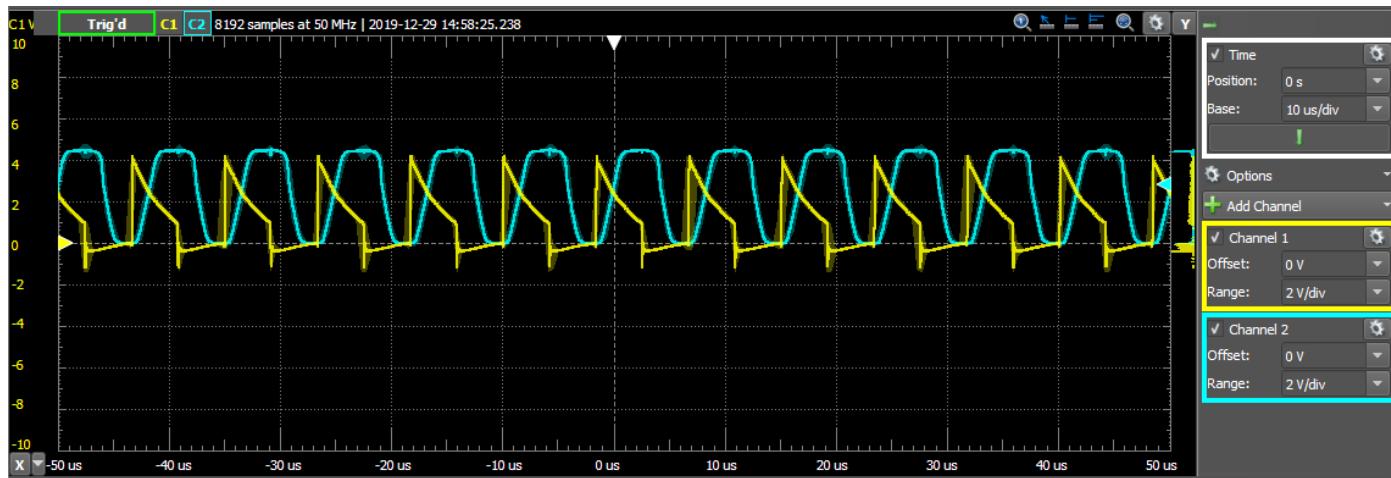
Det kan ses at vi på ben 7 af operationsforstærkeren har 5V Vcc. Derudover er ben 4 trukket i ground og vi har afkoblingskondensatoren der er koblet fra Vcc til ground.

8.3.2.3 Operationsforstærker Modultest

Outputtet fra højpasfilteret, figur 62, går højnæst over i operationsforstærkeren. Som det fremgår af figur 69, så forstærkes inputtet, det gule signal, som ønsket og vi får således et blødere output, det blå signal, som peaker i længere tid og det minder således nærmere om et sinussignal. Vi har erfaret, at dette blødere signal giver et pænere output på envelope detectoren, idet kondensatoren ikke oplades øjeblikket, hvilket resulterede i spikes ved opladning og afladning.



Figur 67 - pin configuration MCP601

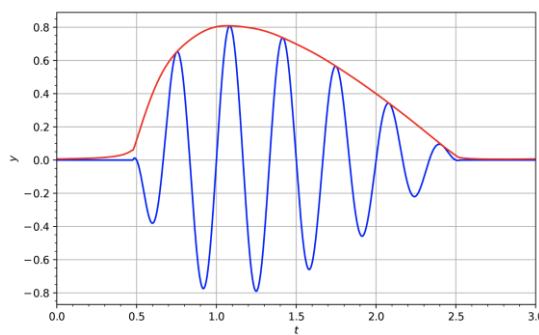


Figur 69 - test af operationsforstærker. Gul = 120 kHz input og blå = output.

Under modultesten af operationsforstærkeren erfarede vi, at operationsforstærkeren modsat forventningen, ikke kunne forstærke op til de 5V, men nærmere op omkring 4,7V. Dette betyder at der ikke forstærkes markant over signalets peak, men den resterende del af signalets top forstærkes op til de 4,7V.

8.3.3 Envelope Detector

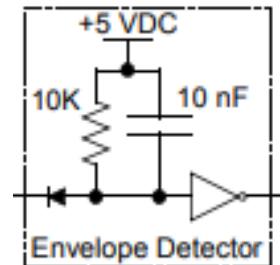
Det signal som vores Schmitt-trigger skal aflæse, kommer i form af et 1 ms 5V burst signal, som svinger 120 tusinde gange i sekundet. Så for at sikre os, at vores Schmitt-trigger kan nå at opfange og omdanne dette burst signal til et højt 5V signal, implementeres en envelope detector, som har til funktion af udglatte de positive spændinger, således at vi står tilbage med et tilnærmelsesvis fladt signal. Et eksempel på funktionen af en envelope detector ses på figur 70.



Figur 70 - eksempel på envelope detector funktion

8.3.3.1 Envelope Detector Hardware design

Designet af envelope detectoren tager udgangspunkt i applikationsnoten⁷, figur 71. Der er valgt at bruge en 100 nF kondensator i stedet for applikationsnotens 10 nF. Yderligere er der valgt at bruge notens modstand på 10 kΩ samt en diode vendt i strømmens retning. Dioden er vendt i strømmens retning, idet der ønskes at fjerne eventuelle negative ladninger og kun tillade de positive at passere. Herefter oplagres ladningen i kondensatoren, som langsomt aflader gennem modstanden når signalet falder, hvilket resulterer i en udglattet kurve. Vores samlede envelope detector ses på figur 72.



Figur 71 - Envelope detector
applikationsnoten



Figur 72 - Envelope detector

Det relevante for komponentværdierne i envelope detectoren er, at afladningstiden på kondensatoren er kort nok til, at der kan aflades mellem hvert zero cross, men samtidig lang nok til, at der ikke aflades mellem toppene på vores 120 kHz burst signal. Med komponentværdierne fra applikationsnoten på 10kΩ og 10 nF, oplevede vi, at der var et markant fald i spænding mellem toppene på 120 kHz signalet. Der blev af denne grund valgt den førnævnte kondensator på 100 nF, hvilket resulterede i et væsentligt pænere signal.

Til at beregne afladetiden aflæses amplituden efter envelope detectorens diode på figur 74⁸ til at være 4,0V, yderligere aflæses threshold spændingen for et højt output efter én

⁷ Referenceliste nr. 2

⁸ Side 92

schmitt-trigger på figur 77⁹, hvilket svarer til et lavt output efter to. Disse bruges i de følgende beregninger for afladetid:

Afladetid

Der er valgt modstanden:

$$R := 10 \cdot 10^3 \Omega$$

og kondensatoren:

$$C := 100 \cdot 10^{-9} F$$

Spænding der ønskes afladet til:

$$U_C := 2.2159 V$$

aflader fra:

$$U_{start} := 4.0 V +$$

Beregning af afladetid:

$$U_C = U_{start} \cdot e^{\frac{-t}{R \cdot C}} \xrightarrow{\text{solve}, t} 0.00059063571960686839233 s$$

tid mellem zerocross:

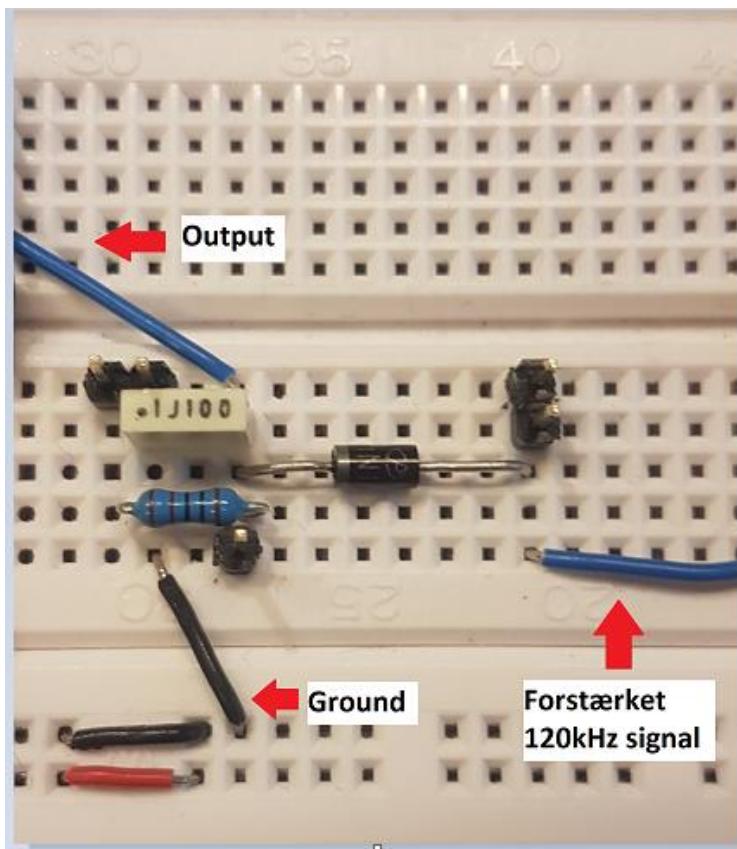
$$f_{60Hz} := \frac{1}{100} = 0.01 s$$

Der kan heraf ses, at vi har en afladetid på omkring 0,6 ms hvilket er væsentligt kortere end de 10 ms der går mellem hvert zero cross.

8.3.3.2 Envelope Detector Implementering

I implementeringen af envelope detectoren, kan det ses at vi modtager det forstærkede 120 kHz signal fra operationsforstærkeren. Her har vi placeret vores diode, så vi kun får den positive spænding igennem. Herefter kommer det over i envelope detectoren, som er bestående af en modstand og en kondensator sat parallelt.

⁹ Side 94

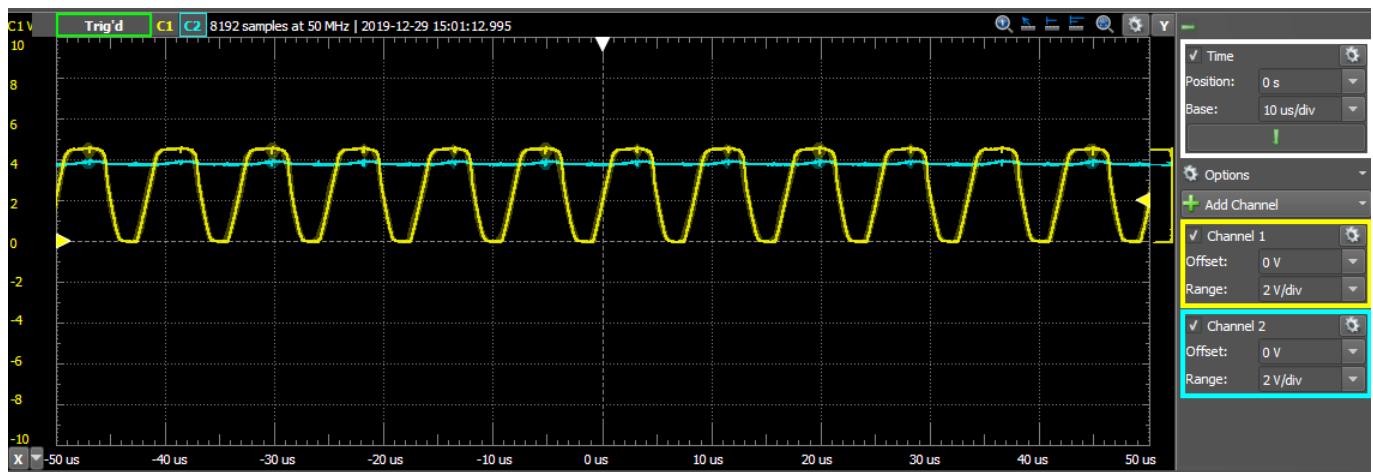


Figur 73 - Implementering af Envelope detector

Her kan det ses på figur 73 at envelope detectoren blandt andet er trukket i jord, samt at vi har et output til schmitt-triggeren.

8.3.3.3 Envelope Detector Modultest

Envelope detectoren er testet ved, at lade outputtet fra operationsforstærkeren gå over i envelope detectoren. Resultatet af modultesten ses på figur 74.



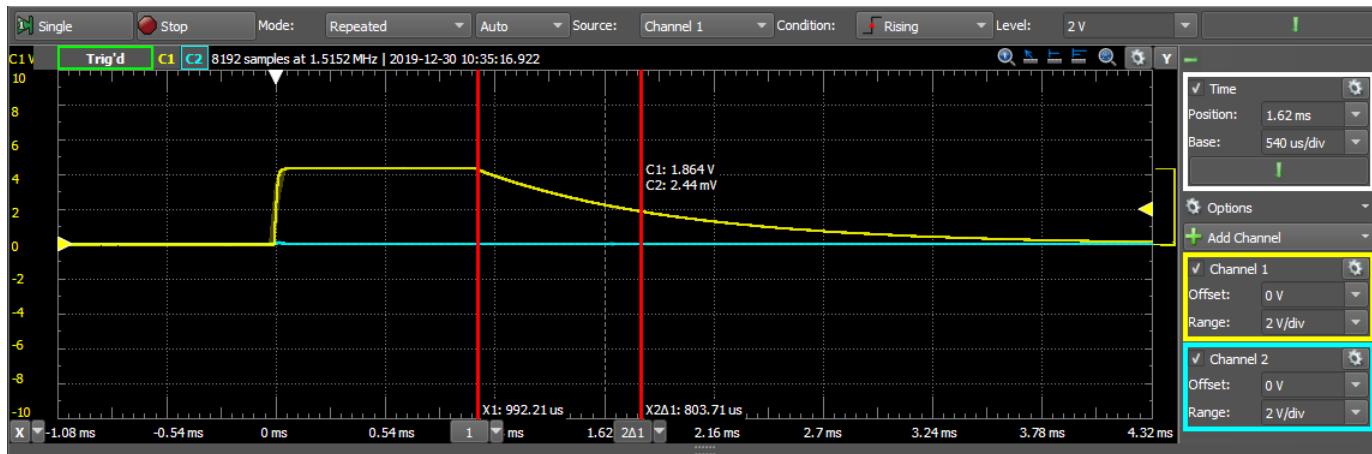
Figur 74

På figuren kan det ses, at vores gule input fra operationsforstærkeren, som ønsket bliver glattet ud efter passage af envelope detectoren. Heraf fremgår det dog ikke, hvorvidt envelope detectoren kan aflade mellem zero cross. For at teste hvorvidt der kan aflades mellem zero cross, er der lavet en separat måling, hvor der med Analog Discovery's wavegen er påført signalet på figur 75 direkte på envelope detectoren.



Figur 75 - Wavegen indstilling for test af afladetid

Resultatet for målingen af afladetiden fremgår af figur 75. På figuren aflæses afladetiden fra 4V til 1.86V til at være ca. 0.8 ms. Denne tid er væsentligt kortere end de 10 ms der er mellem hver zero cross. 1.86V er valgt, idet det er en spænding der er lavere end den målte threshold spænding for et højt output på den inverterende schmitt trigger, som var på 2.21V (figur 77).



Figur 76 - afladning af envelope detectorens kondensator

8.3.4 Schmitt-trigger

Vi er interesseret i at få et tydeligt signal, som enten er 0V eller 5V, således at modtagerens mikrocontroller ikke er i tvivl om den modtager et højt eller et lavt signal ved hvert zero-cross. Dette kan man få ved at implementere en Schmitt-trigger i sit kredsløb.

En Schmitt-trigger er en form for logiks input type¹⁰, som giver en bestemt spænding på outputtet, så længe inputtet forbliver inden for Schmitt-triggerens øvre/nedre spændingstærskel, også kaldet upper/lower threshold spænding.

8.3.4.1 Schmitt-trigger Hardware Design

Schmitt-triggeren bruges i vores kredsløb til at gøre udgangssignalet på envelope detectoren mere veldefineret. Her er der valgt en SN74AHC14N¹¹, som er en inverterende Schmitt-trigger.

Da Schmitt-triggeren er af den inverterende slags, så vælges der at invertere signalet 2 gange, hvilket garanterer, at vores udgangssignal er 0 eller 5 volt.

For at sikre os, at vi sender et stort nok signal fra envelope detectoren til Schmitt-triggeren, har vi manuelt målt hvor stor threshold spændingen skal være før én Schmitt-trigger går høj/lav. Dette kan ses på figur 77.

¹⁰ <https://howtomechatronics.com/how-it-works/electrical-engineering/schmitt-trigger/>

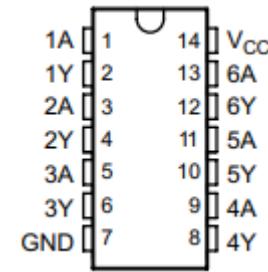
¹¹ Referencelisten nr. 3



Figur 77 - Threshold spænding for schmitt-trigger

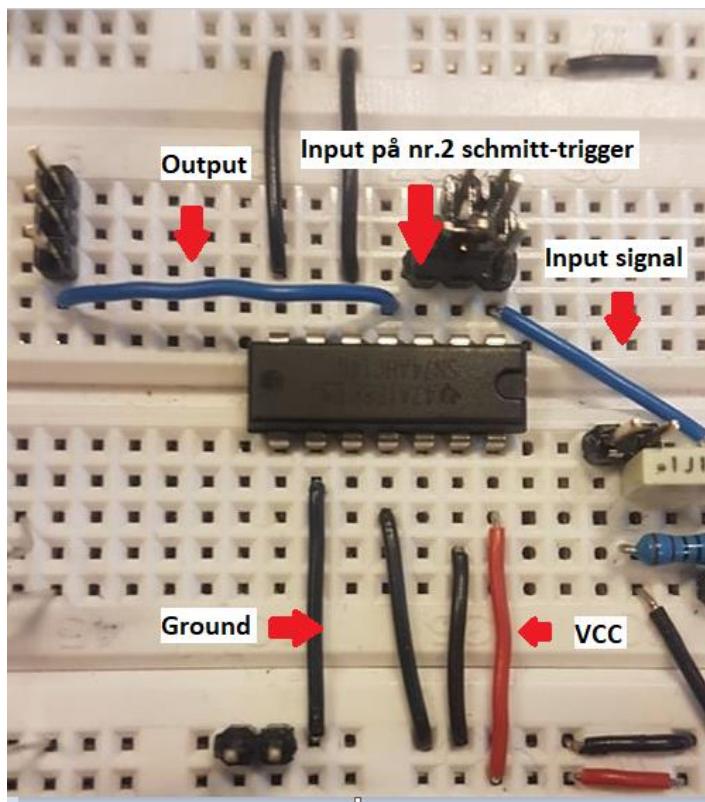
8.3.4.2 Schmitt-trigger Implementering

I Implementeringen af Schmitt-triggeren kan det ses på figur 79, at inputtet fra envelope detectoren går ind på ben 1 af Schmitt-triggeren¹², figur 78. Her noteres det, at vi trækker signalet igennem 2 Schmitt-triggere. Således sikres et højt output ved et højt input og omvendt et lavt output ved et lavt input. Til sidst kommer outputtet ud på Schmitt-triggerens ben 4, efter passage af de to inverterende Schmitt-triggere.



Figur 78 - pin configuration
SN74HC14

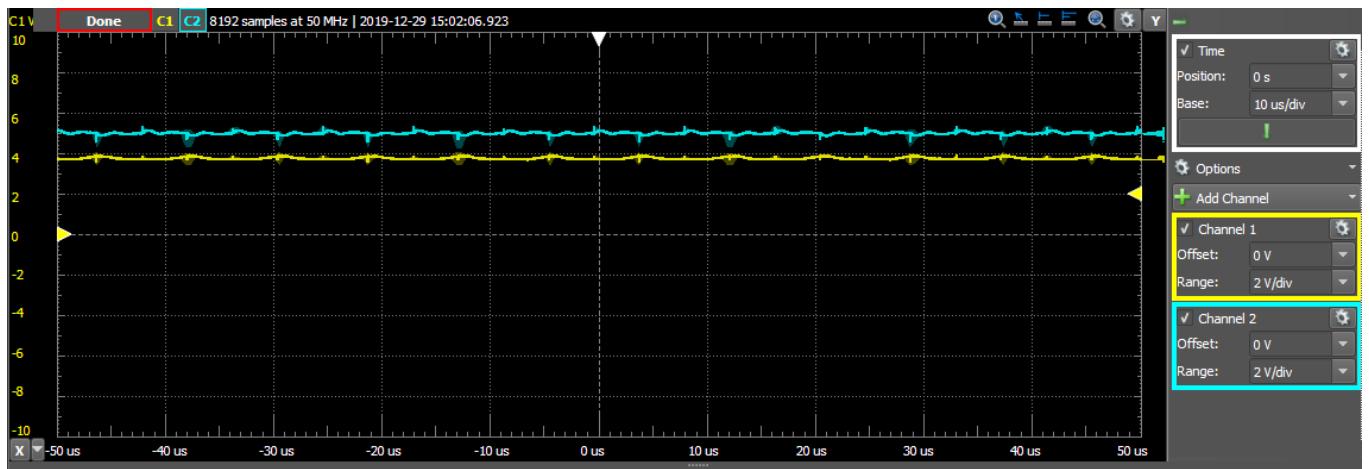
¹² Referencelisten nr. 3



Figur 79 - Implementering af Schmitt-trigger

8.3.4.3 Schmitt-trigger Modultest

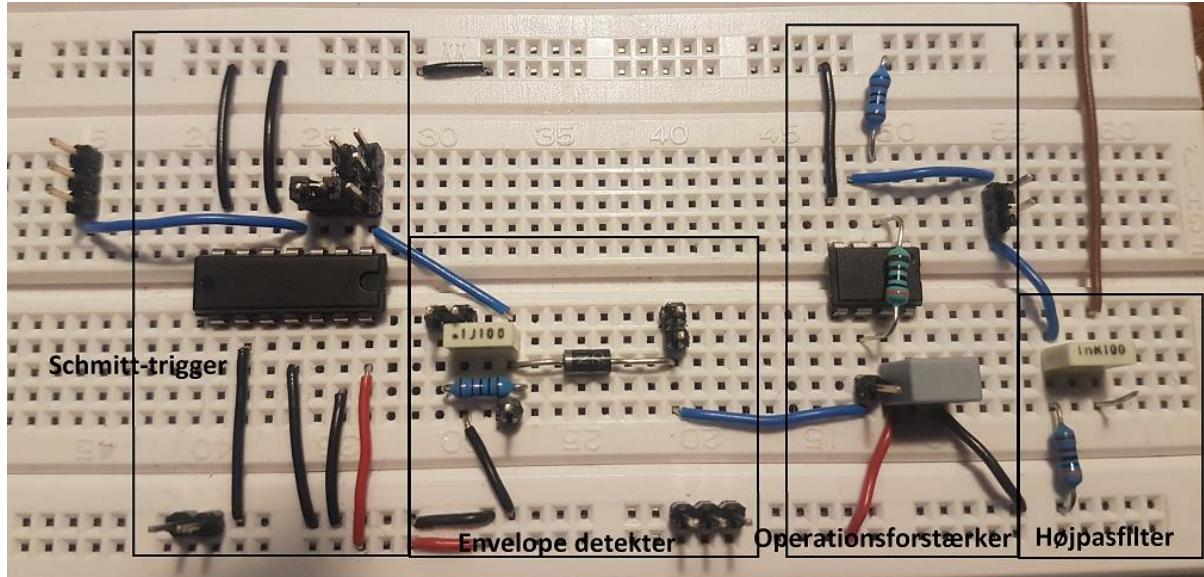
Testen af Schmitt-triggeren, er udført ved at lade outputtet fra envelope detectoren gå ind på Schmitt-triggeren. Resultatet af dette fremgår af figur 80, hvoraf det kan ses, at vi får et output signal der ligger på 5V. Af figuren kan det yderligere ses, at der er lidt svingninger på outputtet. Disse svingninger er små og umiddelbart ubetydelige, hvilket betyder at der ikke er gjort noget for at bekæmpe dem, men en afkoblingskondensator på forsyningssbenet kunne eventuelt hjælpe.



Figur 80 – før og efter schmitt-trigger

8.3.5 Samlet modtager kredsløb

Nedenstående implementering er designet ud fra figur 53¹³. Her noteres det, at der er fælles GND og VCC for alle hardware moduler. Kigger man på figur 81 ses de blokke som danner grundlag for vores modtager kredsløb. Det implementerede højpasfilter bliver koblet på elnettet (brun ledning) og dæmper her elnettets 18V signal inden det passere videre igennem kredsløbet.



Figur 81 - Modtager kredsløb realiseret på fumlebræt

¹³ Side 54



8.4 Resultater hardware modultest

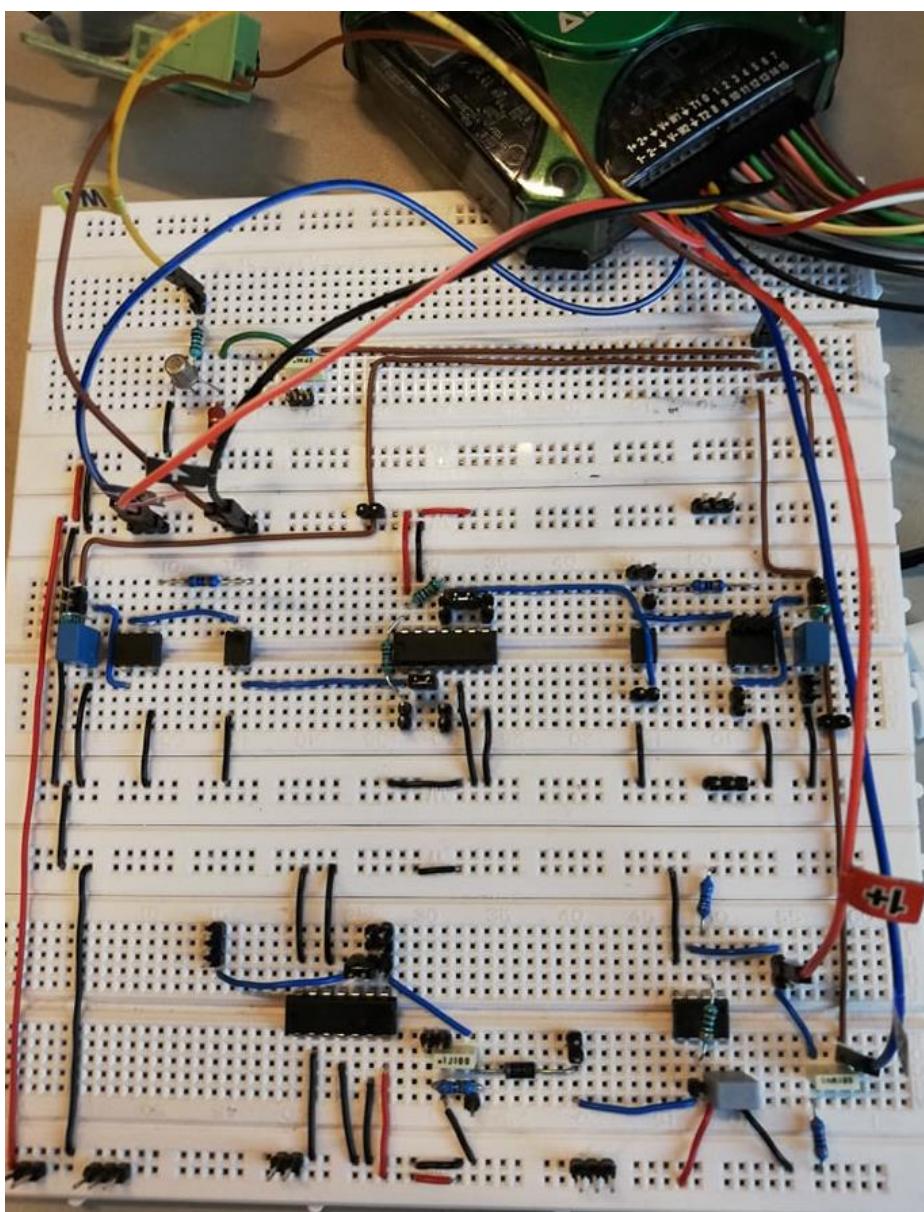
Kort oversigt over de forskellige moduler og testen af dem, samt de observerede resultater.

Enhed:	Beskrivelse af test:	Observeret resultat:
X10 sender hardware	Ved test af X10 sender sættes et oscilloskop på udgangen af senderkredsløbet. Der sendes et 120kHz signal ind, og 5 volt VCC	Ud fra vores Waveforms Scope, ses det at 120kHz signalet blandes med 18V AC signalet
Zero-Cross detector	Ved test af Zero-Cross detectoren måler vi på det simulerede elnet signal, og på zero-cross output.	Når der registreres 0 V på elnettet, sender Zero-Cross detektoren et signal på ca 0,7 millisekunder, som forventet
X10 modtager hardware	For at teste X10 modtager kredsløbet, tilkobles det senderkredsløbet, hvorpå der konstant sendes 5V 120 kHz ud på 18V elnettet.	På udgangen af X10 modtageren får vi vores forventede 5V

Tabel 18 - Modultest

8.4.1 Opstilling hardwaretest

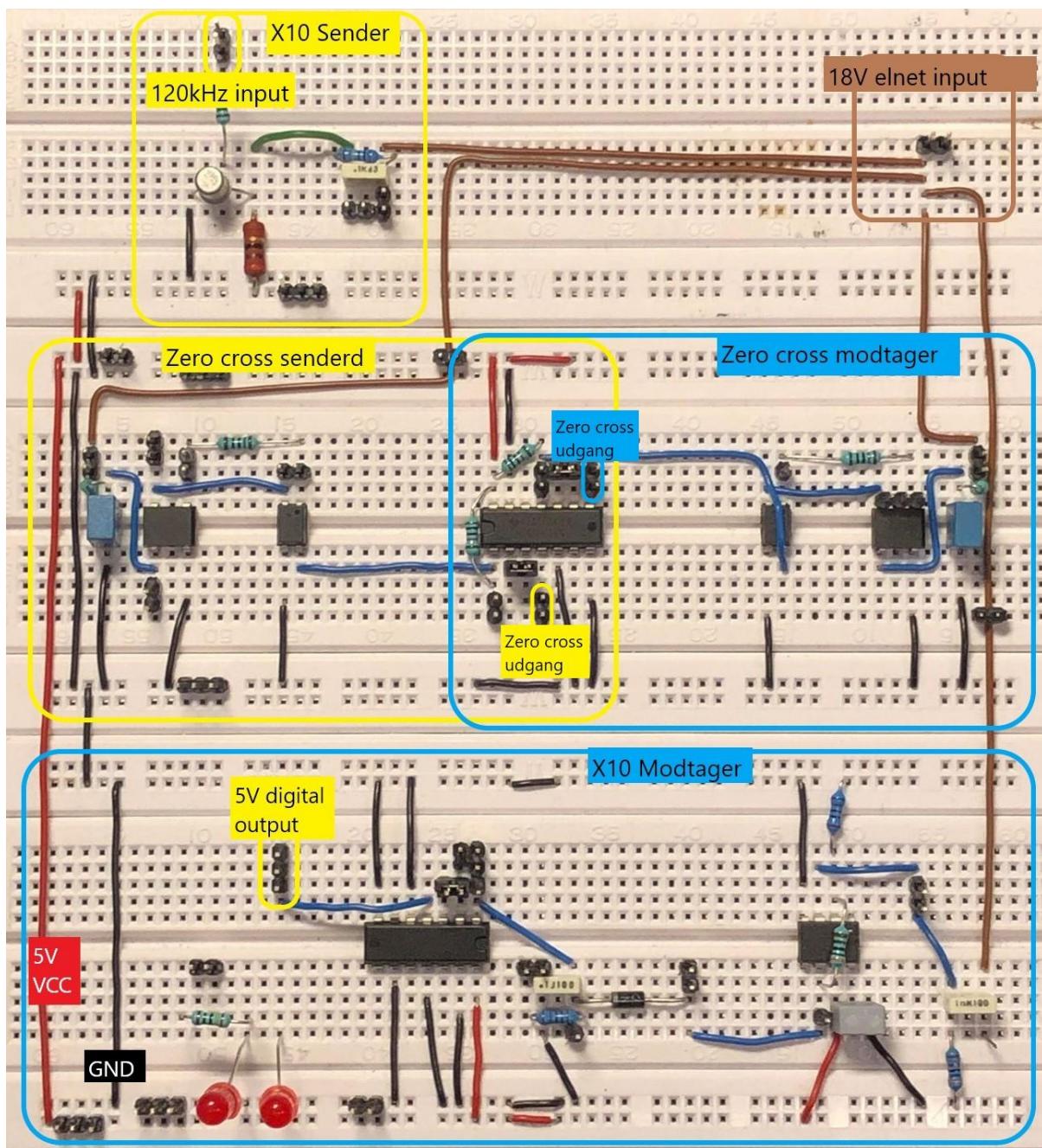
Som det fremgår af nedenstående figur 81, så har vi brugt følgende opstilling til at teste samtlige hardware moduler, eventuelle afvigelser er noteret. De **brune ledninger** er 18V elnettet, den **gule ledning** er wavegen 120kHz firkantet 0-5V, sort er Analog ground og **blå/orange** er Analog scope channel 1 og 2.



Figur 82 - samlet opsætning af hardware - Opstilling af kredsløb

8.5 Samlet kredsløb

Nedenfor fremgår et billede af det samlede hardwarekredsløb.





Figur 83 - Samlet kredsløb

9.0 Software Design, implementering og modultest

9.1 Sender

9.1.1 Design

Funktionsbeskrivelser af funktioner brugt i Sender-blokken. Funktionerne initX10 og initUART er ikke inkluderet i klasse-diagrammet fra tidligere, da de blot klargør hardwaren.

Void initX10(void)

Enabler global interrupts.

Port B, D og E bliver sat til indgange.

Enabler interrupt 0 og sætter den til rising edge.

Timer 3 enables og sættes til fastPWM mode.

OCR3A og ICR3 indstilles så man opnår et pwm signal på 120khz på 50% duty cycle.

Void initUART(unsigned long BaudRate, unsigned char DataBit, unsigned char Parity, unsigned char RX_int)

Initierer uart og indstiller baudrate, databits, parity, og RX interrupt efter parameter inputs.

Derudover sættes der 1 stopbit.

Void setSignal(int command)

Benytter sig af globale arrays for startbits, unit og house adresser, samt commands, suffixes og stopbits.

Funktionen opretter et signal array på 56 pladser.

Herefter indsættes startbits, unit og house adresser, samt commands, suffixes og stopbits.

Der kan vælges mellem 2 commandarrays som her afhænger af parameteren.

Void sendBurst(void)

OC3A's pins på arduinoen benytter sig af Port E som tidligere blev sat som indgang.

I sendBurst sættes port E som udgang og man sender derved et burst med tidligere bestemt frekvens og duty cycle. Dette sendes i 1 ms og stoppes derefter ved at port E bliver sat til indgang igen.

Void sendSignal(void)

Denne funktion går igennem det tidligere indstillede signal-array sekventielt. Her benyttes en integer-variabel som index. Står der et 0 på den givne plads venter den 1 ms og indexet inkrementeres. Er der et 1 kaldes funktionen sendBurst og indexet inkrementeres. Overskridt indexet længden på arrayet sættes index til 0 og variablen enableZeroCross sættes til 0.

Void UnlockGui(void)

Hvis port b, pin 0 er høj skal der sendes et højt signal til GUI'en via UART.

ISR(USART0_RX_vect)

Når senderen modtager et UARTsignal fra GUI'en afhentes signalet i UDR0 registret. Den kan enten modtaget en integer af 1 eller 2.

Er signalet 1, kaldes funktionen setSignal() med 1 som parameter. Er signalet 2 gøres det samme med 2 som parameter.

Herved er signal-arrayet klar til at blive sendt når der er zerocrosses.

Derudover sættes en variabel enableZeroCross til 1.

ISR(INT0_vect)

Interrupt 0 pinnen på arduinoen er forbundet med udgangen fra en zerocrossdetektor.

Hvis enableZeroCross er sat til 1 i UART interrupt rutinen skal funktionen sendSignal kaldes.

9.1.2 Implementering

Her dokumenteres implementering af de funktioner der blev beskrevet i design afsnittet.

Void initX10(void)

```
void initX10(){
    sei(); //enable interrupt
    DDRD = 0x00; //sætter port d til input
    DDRE = 0x00; //sætter port e til input
    DDRB = 0x00; //sætter port b til input. modtager signal fra kodelås
    EICRA = 0b00000011; //interrupt 0 sat til rising edge
    EIMSK = 0b00000001; //interrupt 0 enabled
    TCCR3A = 0b11110010; //timer 3 sættes til fastpwm mode med ICR3 som top. Sætter OCR3A til compare match. clear på BOTTOM (invering mode).
    TCCR3B = 0b00011001; //timer 3 sættes til fastpwm mode med ICR3 som top. prescaler sættes til 1 (ingen prescaling).
    OCR3A = 133/2;
    ICR3 = 133;
}
```

Figur 84 - Implementering af funktioner



Void initUART(unsigned long BaudRate, unsigned char DataBit, unsigned char Parity, unsigned char RX_int)

```
void InitUART(unsigned long BaudRate, unsigned char DataBit, unsigned char Parity, unsigned char RX_Int)
{
    if ((BaudRate >= 110) && (BaudRate <= 115200) && (DataBit >=5) && (DataBit <= 8))
    {
        // "Normal" clock, no multiprocessor mode (= default)
        UCSR0A = 0b00100000;
        // No interrupts enabled
        // Receiver enabled
        // Transmitter enabled
        // No 9 bit operation
        UCSR0B = 0b00011000;
        // Enable RX interrupt (if required by parameter)
        if (RX_Int)
            UCSR0B |= (1<<7);
        // Asynchronous operation, 1 stop bit
        // Bit 2 and bit 1 controls the number of data bits
        UCSR0C = (DataBit-5)<<1;
        // Set Baud Rate according to the parameter BaudRate:
        // Adding (8*Baudrate) ensures correct rounding (up/down)
        UBRR0 = (F_CPU+(8*BaudRate))/(16*BaudRate) - 1;
        if (Parity == 'E') {
            UCSR0C |= 0b00100000;
        } else if (Parity == 'O') {
            UCSR0C |= 0b00110000;
        } else {
            UCSR0C |= 0b00000000;
        }
    }
}
```

Figur 85

Void setSignal(int command)

```
//arrays
//Start
const int startBit[4] = {1,1,1,0};
const int startLength = 4; //startBit længden er 4 bit lang.
//stop
const int stopBit[6] = {0,0,0,0,0,0};
const int stopLength = 6; //stopBit er 6 bit lang
//x10 commands fra A-2 datablad
const int onCmd[8] = {0,1,0,1,1,0,0,1};
const int offCmd[8] = {0,1,0,1,1,0,1,0};
int cmd = 0;
const int cmdLength = 8; // commands er 8 bit lange uden suffix
//unit addresser
const int address1[8] = {1,0,1,0,1,0,0,1};
const int addressLength = 8;
//huse - houses
const int houseA[8] = {0,1,1,0,1,0,0,1};
const int houseLength = 8;
//suffix
const int addressSuffix[2] = {0,1};
const int commandSuffix[2] = {1,0};
const int suffixLength = 2;
//signals
int signal[56] = {0};
int signalLength = 56;
int counter=0;
```

Figur 86

Her ses de hardcodede bit strenge i deres respektive arrays. Nederst ses den erklæringen af det tomme signal array.

```
void setSignal(int command){
    //counter index nulstilles.
    for(int i=0;i<signalLength;i++){
        signal[i]=0;
        counter = 0;
    }
    //sætter bits ind i arrayet der skal sendes.
    //startbits
    for(int i=0;i<startLength;i++){
        signal[counter]=startBit[i];
        counter++;
    } // 4 bits
    //house adresse
    for(int i=0; i<houseLength; i++){
        signal[counter] = houseA[i];
        counter++;
    } //4+8=12
    //unit adresse
    for(int i=0; i<addressLength; i++){
        signal[counter]=address1[i];
        counter++;
    } //12+8=20
    //adresse suffix
    for (int i=0;i<suffixLength;i++){
        signal[counter]=addressSuffix[i];
        counter++;
    }
} //20+2=22
```

Figur 87



```
//stopbit
for(int i=0;i<stopLength;i++){
    signal[counter]=stopBit[i];
    counter++;
}//22+6=28
//sætter commands
//startbits
for(int i=0;i<startLength;i++){
    signal[counter]=startBit[i];
    counter++;
}//28+4=32
//house adresse
for(int i=0; i<houseLength; i++){
    signal[counter] = houseA[i];
    counter++;
}//32+8=40
```

Figur 88

```
//command setup
switch(command){
    case 1:
        for(int i=0; i<cmdLength; i++){
            signal[counter]=onCmd[i];
            counter++;
        }
        break;
    case 2:
        for(int i=0; i<cmdLength; i++){
            signal[counter]=offCmd[i];
            counter++;
        }
        break;
}//40+8=48
//command suffix
for (int i=0;i<suffixLength;i++){
    signal[counter]=commandSuffix[i];
    counter++;
}//48+2=50
//stopbit
for(int i=0;i<stopLength;i++){
    signal[counter]=stopBit[i];
    counter++;
}//50+6=56
}
```

Figur 89

Her indsættes alle de hardcodede bitstrenge ind i i arrayet. Variablen counter fungerer som en global indexvariabel der uafhængigt af For-loopsne går igennem arrayets pladser.
Se kommentarer i koden for uddybelse.

Void sendBurst(void)

```
void sendBurst(){
    //Port E bliver sat til udgang, og et burst med før indstillet frekvens og duty cycle sendes
    DDRE = 0xff;
    //delay et milisekund
    _delay_ms(1);
    //Port E bliver sat til indgang igen, og burstet slukkes.
    DDRE=0x00;
}
```

Figur 90

Void sendSignal(void)

```
void sendSignal(){
    if(signal[sendIndex]==0){
        _delay_ms(1);
        sendIndex++;
    }
    else if(signal[sendIndex]==1){
        sendBurst();
        sendIndex++;
    }
    if(sendIndex==signalLength){
        sendIndex=0;
        enableZeroCross=0;
    }
}
```

Figur 91

Her går funktionen array signalet igennem vha. sendIndex. Når den møder et 0 venter den et milisekund og går videre. Når den møder et 1 sender den et burst og går videre. Alle pladser i arrayet er gået igennem sættes sendIndex og enableZeroCross til 0.

Void UnlockGui(void)

```
void unlockGUI(){
    if((PORTB&(1<<0))==0){
        while((UCSR0A&(1<<5))==0)
        {}
        UDR0 = 0xff;
    }
}
```

Figur 92

Hvis pin 0 ved PORTB er høj sendes et højt signal via UART 0.

ISR(USART0_RX_vect)

```
ISR(USART0_RX_vect){ //interrupt-rutine når der kommer signal fra uart.
    GUI_sig = UDR0;
    switch(GUI_sig){
        case 1: setSignal(1);
        break;
        case 2: setSignal(2);
        break;
        default: setSignal(1);
    }
    enableZeroCross=1;
}
```

Figur 93

Interruptrutine ved UART-signal. Hvis der modtages et 1-tal vælges command 1. Ligeledes vælges der command 2 hvis interruptrutinen modtager et 2-tal.

ISR(INT0_vect)

```
ISR(INT0_vect){ //interrupt-rutine ved zerocross fra elnettet.
    if(enableZeroCross==1){
        sendSignal();
    }
}
```

Figur 94

enableZeroCross er høj efter UART interruptrutinen, og INT0's interrupt pin modtager et højt signal, sendes signalet med den command der blev specificeret i den forkaldte interruptrutine.

Hvis enableZeroCross er høj efter UART interruptrutinen, og INT0's interrupt pin modtager et højt signal, sendes signalet med den command der blev specificeret i UART interruptrutinen og setSignal funktionen

Int main(void)

```
int main(void)
{
    InitUART(9600, 8, 0, 1);
    initX10();
    while(1)
    {
        unlockGUI();
    }
}
```

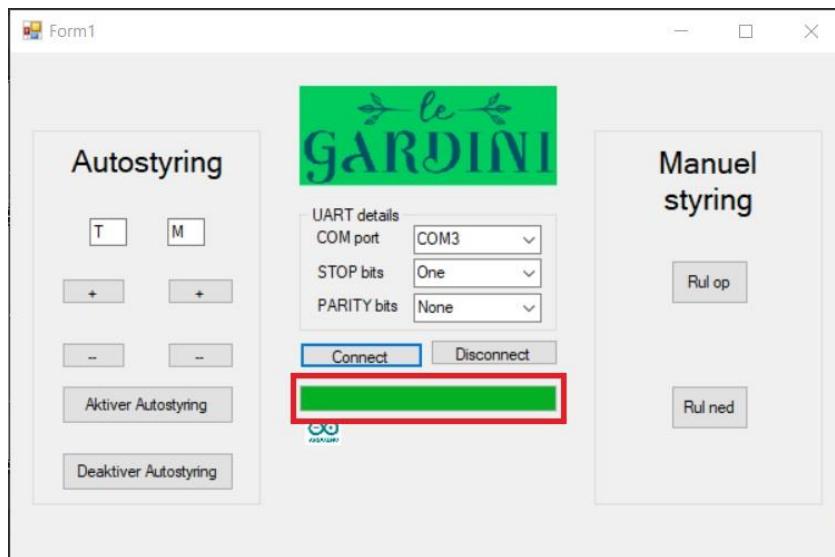
Figur 95

Her blot initiering, samt unlockGUI funktionen i et while-loop.

9.1.3 Modultest

Vi vil under modultesten se om senderen kan forbindes til GUI. sende bursts ved UART og zerocross interrupts samt en test af setSignal funktionen for at sikre os at bitstrenge bliver loadet korrekt.

Modultest af forbindelse mellem GUI og sender



Figur 96 - dokumentation af modultest for gui og sender forbindelse

Indbygget på GUI'en er en progressbar der fortæller om der er forbindelse mellem senderen og GUI'en. Dette er dokumenteret på figur 83.

Modultest af korrekt indlæsning af bitstrenge i signal array

Vi har her kopieret setSignal funktionen fra senderkoden, og indført printf funktioner fra signal-arrayet undervejs.

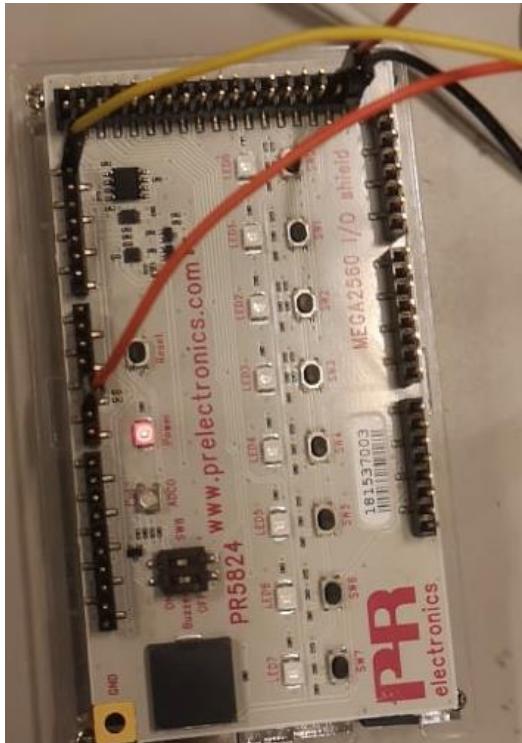
Hvis bitstrengeene er loadet korrekt, bør de print fra testkoden stemme overens med de hardkodede bitstrenge.

```
Startbits: 1110
House adresse: 01101001
Unit adresse: 10101001
Suffix: 01
Stopbits: 000000
Startbits: 1110
House adresse: 01101001
Command1: 01011001
Command2: 01011010
Suffix: 10
Stopbit: 000000
```

Figur 97

Her ses resultatet af testkoden. De printede koder er identiske med de hardkodede bitstrenge, og testen er hermed succesfuld.

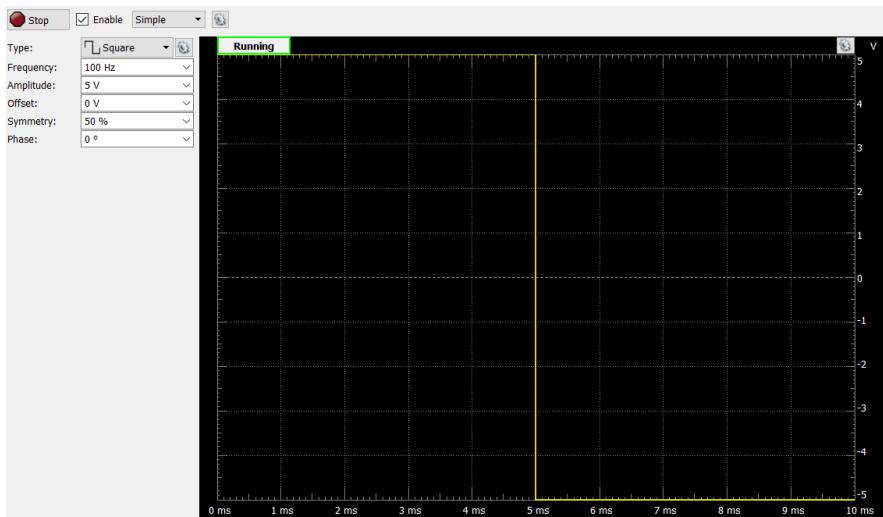
Modultest af bursts ved zerocrosses



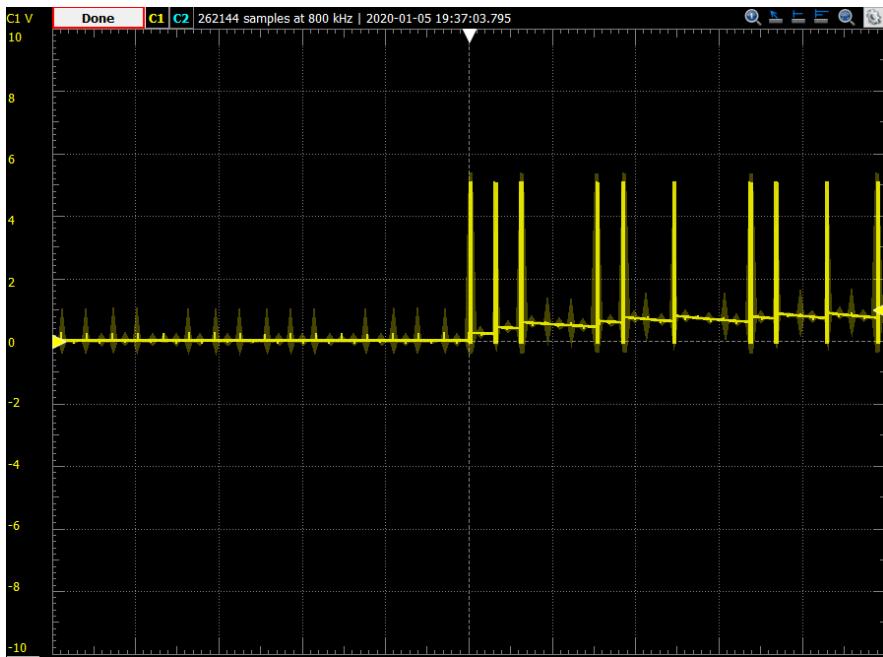
Figur 98

På sender arduino har vi ført en funktionsgenerator med et firkant signal på 100hz (gul ledning) på INT0's interruptben.

På sender arduino har vi ført en funktionsgenerator med et firkantsignal på 100hz (gul ledning) på INT0's interruptpin. Herefter måler vi mellem timer 3's output pin og ground. Hvis vi sender en bitstreng fra GUI'en og senderen bør vi kunne se den på et scope.



Figur 99 - 100hz firkant signal fra 0-5v

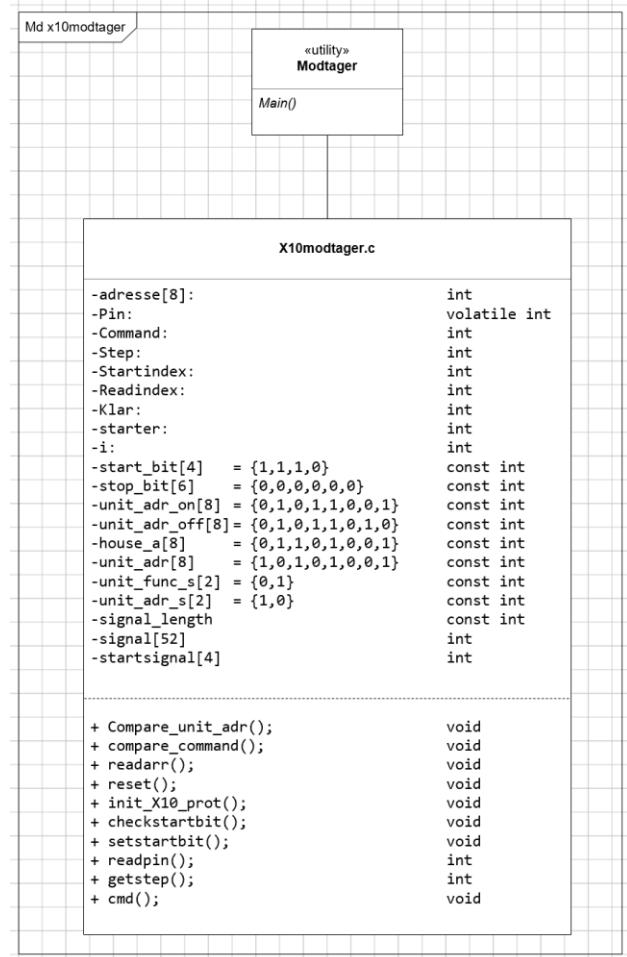


Figur 100 - måling på udgangen

På figur 100 kan vi se at vi har kunne måle en bitstreng. Vi kan udfra målingen Aflæse startbitstrengen. Den ses her som 3 høje og en lav(1110). Testen er hermed succesfuld.

9.2 Modtager

9.2.1 Design



Figur 101 Moduldiagram for X10 modtager

Void reset(); bliver brugt til lave et reset af programmet såfremt der en fejl i den bitstrøm der er modtaget fra senderen. Såfremt der er en fejl, så sættes signal[] arrayet til 0.

Void init_x10_prot(); er en funktion der bliver brugt til at initiere systemet og micro controlleren. Den sætter Port B som output og Port D som input på vores micro controller. Derudover sætter den også interrupt til rising edge og initierer det på PDO

Int getstep(); er en funktion der bliver kaldt i vores main() for at bestemme hvor langt vi er. Den returnerer en integer kaldet step. Step bliver talt op løbende i koden, hvilket gør at programmet kan afgøre hvor den er.

Int readpin(); Denne læser på PD1 ved hver zerocrossing og returnerer 1 hvis den får et high, ellers returneres 0.

Void setstartbit(); denne bruger ovennævnte readpin(); funktion og værdierne sættes ind i arrayet startsignal[] indtil den har kørt readpin(); 4 gange og vi dermed har modtaget 4 bits til vores array. Funktionen bliver først kørt efter den har registreret en high på PD1 én gang.

Void checkstartbit(); denne bruges til at tjekke om hvorvidt det vi har modtaget i setstartbit() er korrekt i forhold til de allerede definerede startbit {1,1,1,0}. Såfremt den har modtaget det korrekte signal, så inkrementerer den int step, ellers laver den et reset().

Void readarr(); er en funktion der nu kalder readpin(); til den har modtaget 52 bits. Disse placeres i arrayet signal[]. Readarr() bliver kun kaldt såfremt startbit er godkendt og step == 1.

Void compare_unit_adr(); er en funktion der sammenligner den hardcodede unit adresse med bit 8 til og med 15 i signal[] arrayet. Såfremt denne gennemføres, så inkrementerer den step funktionen og programmet kan derefter gennemføres.

Void compare_command(); denne function gør det samme som compare_unit_adr, blot med hvor den sammenligner bit 36 til og med 43 i signal[] arrayet. Denne inkrementerer ligeledes step, hvorfor den så kan forsætte til sidste funktion, hvilket er nedenstående cmd().

Void cmd(); er en funktion der er blevet lavet til at kunne teste systemet med, hvor den vil tænde en LED alt afhængigt af om command er 1 eller 2. Default for denne er at sætte alt til 0.

9.2.2 Implementering

```
x10modtager.h ➔ X main.c x10modtager.c*
➔ x10modtager.h ➔ C:\Users\mortenovergaard\Documents\

void compare_unit_adr();
void compare_command();
void readarr();
void reset();
void init_X10_prot();
void checkstartbit();
void setstartbit();
int readpin();
int getstep();
void cmd();
```

Figur 102 x10modtager.h

```
x10modtager.h      main.c      x10modtager.c*  ↗ X
→ readarr.if.if.signal
└─/*
   * CFile2.c
   *
   * Created: 12/18/2019 12:59:42 PM
   * Author: mortenovergaard
   */
#define F_CPU 16000000
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include "x10modtager.h"

// Valg af house
const int house_a[8] = {0,1,1,0,1,0,0,1};

// Start og stop bit
const int start_bit[4] = {1,1,1,0};
const int stop_bit[6] = {0,0,0,0,0,0};

// funktioner
const int unit_adr_on[8] = {0,1,0,1,1,0,0,1};
const int unit_adr_off[8] = {0,1,0,1,1,0,1,0};

// suffixes
const int unit_func_s[2] = {0,1};
const int unit_adr_s[2] = {1,0};

// Unit addresser
const int unit_adr[8] = {1,0,1,0,1,0,0,1};

// Signal
int signal[52] = {};
int startsignal[4] = {};
const int signal_length = 52;
```

Figur 103 Del 1 x10modtager.c



The screenshot shows a code editor with three tabs at the top: "x10modtager.h", "main.c", and "x10modtager.c*". The "x10modtager.c*" tab is active. The code in the editor is as follows:

```
// Adresse

int adresse[8] = {0};

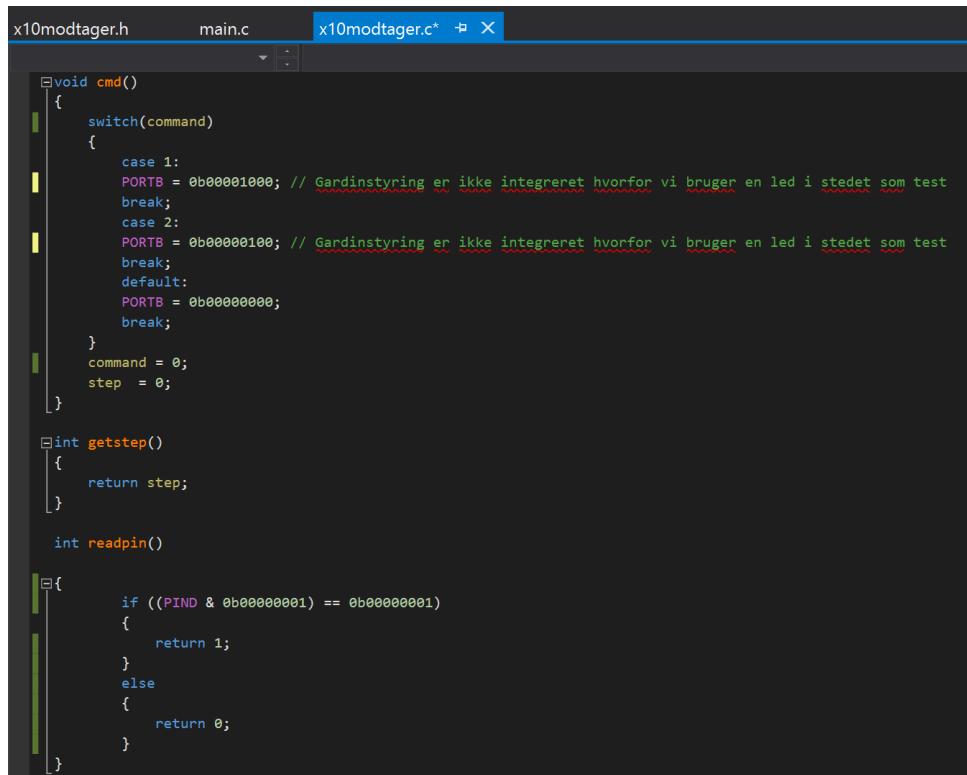
volatile int pin;

int command = 0;
int step = 0;
int startindex = 0;
int readindex = 0;
int klar = 0;
int starter = 0;
int i = 0;

void init_X10_prot()
{
    sei();
    DDRD = 0x00;
    DDRB = 0xFF;
    PORTB = 0x00;
    PORTD = 0x00;
    EICRA = 0b00000011;
    EIMSK = 0b00000001;
}

void reset()
{
    for (int i = 0; i < signal_length; i++)
    {
        signal[i] = 0;
    }
}
```

Figur 104 Del 2 af x10modtager.c



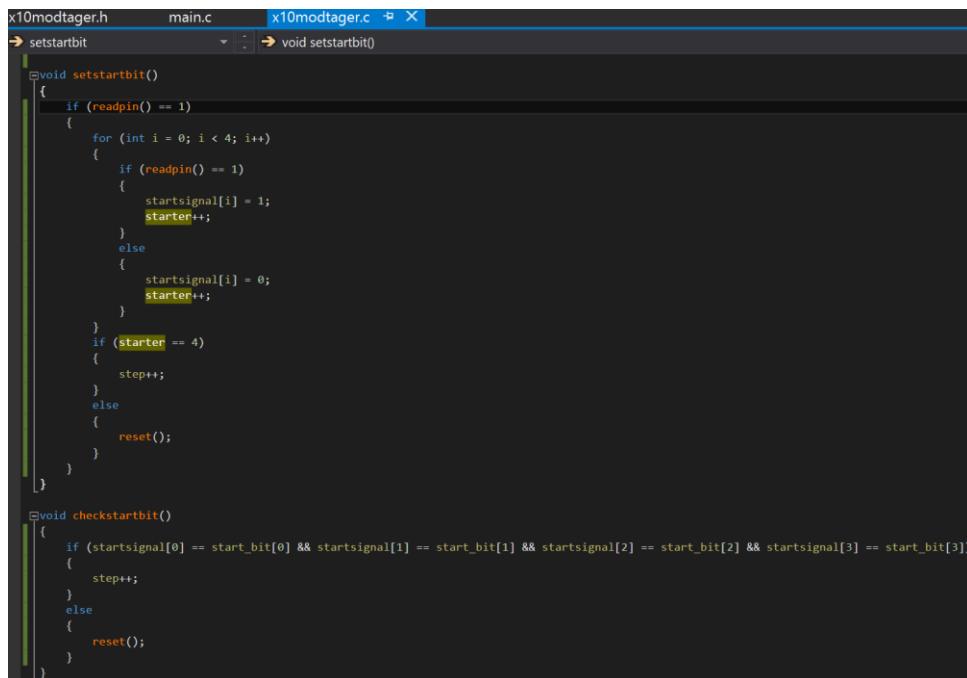
```
x10modtager.h      main.c      x10modtager.c* ✘ X
void cmd()
{
    switch(command)
    {
        case 1:
            PORTB = 0b00001000; // Gardinstyring er ikke integreret hvorfor vi bruger en led i stedet som test
            break;
        case 2:
            PORTB = 0b00000100; // Gardinstyring er ikke integreret hvorfor vi bruger en led i stedet som test
            break;
        default:
            PORTB = 0b00000000;
            break;
    }
    command = 0;
    step = 0;
}

int getstep()
{
    return step;
}

int readpin()

{
    if ((PIND & 0b00000001) == 0b00000001)
    {
        return 1;
    }
    else
    {
        return 0;
    }
}
```

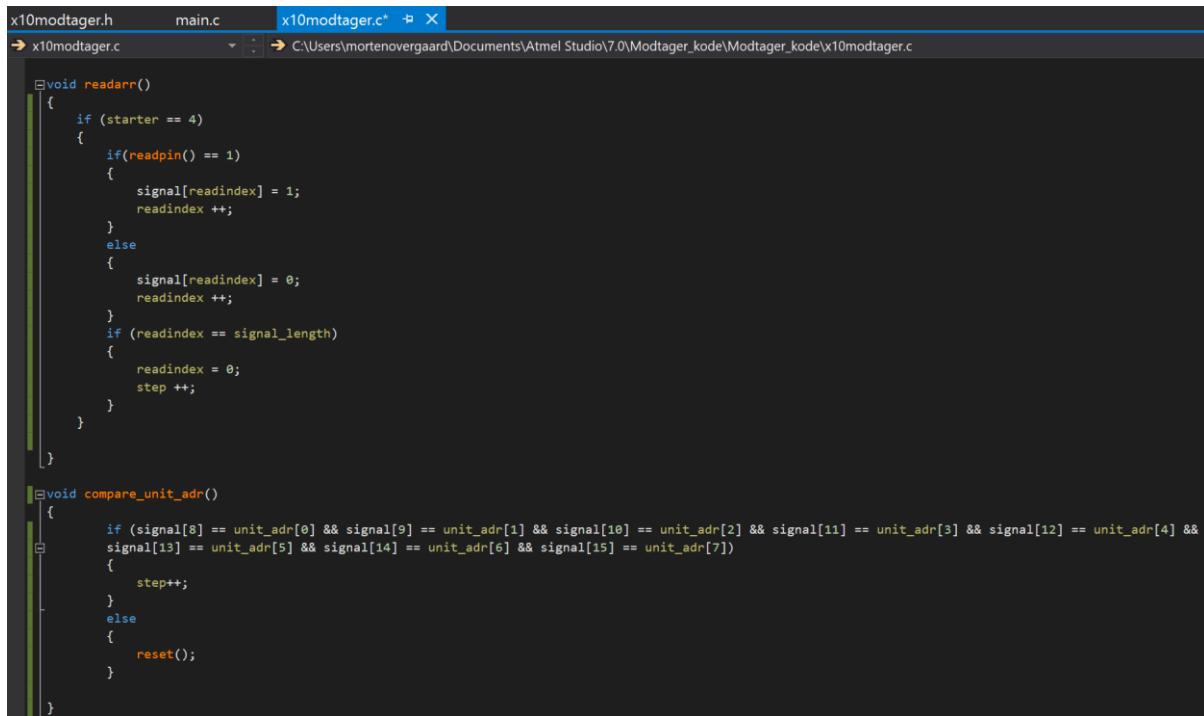
Figur 105 Del3 x10modtager.c



```
x10modtager.h      main.c      x10modtager.c* ✘ X
void setstartbit() ➔ void setstartbit()
{
    if (readpin() == 1)
    {
        for (int i = 0; i < 4; i++)
        {
            if (readpin() == 1)
            {
                startsignal[i] = 1;
                starter++;
            }
            else
            {
                startsignal[i] = 0;
                starter++;
            }
        }
        if (starter == 4)
        {
            step++;
        }
        else
        {
            reset();
        }
    }
}

void checkstartbit()
{
    if (startsignal[0] == start_bit[0] && startsignal[1] == start_bit[1] && startsignal[2] == start_bit[2] && startsignal[3] == start_bit[3])
    {
        step++;
    }
    else
    {
        reset();
    }
}
```

Figur 106 Del 4 x10modtager.c



```

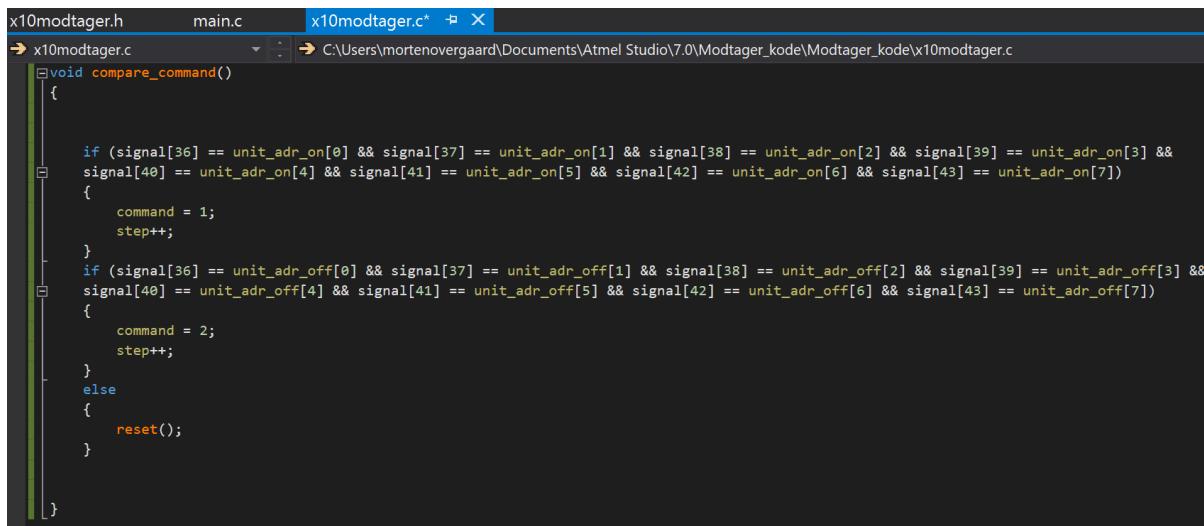
x10modtager.h      main.c      x10modtager.c*  ✘ X
➔ x10modtager.c
C:\Users\mortenovergaard\Documents\Atmel Studio\7.0\Modtager_kode\Modtager_kode\x10modtager.c

void readarr()
{
    if (starter == 4)
    {
        if(readpin() == 1)
        {
            signal[readindex] = 1;
            readindex++;
        }
        else
        {
            signal[readindex] = 0;
            readindex++;
        }
        if (readindex == signal_length)
        {
            readindex = 0;
            step++;
        }
    }
}

void compare_unit_addr()
{
    if (signal[8] == unit_adr[0] && signal[9] == unit_adr[1] && signal[10] == unit_adr[2] && signal[11] == unit_adr[3] && signal[12] == unit_adr[4] && signal[13] == unit_adr[5] && signal[14] == unit_adr[6] && signal[15] == unit_adr[7])
    {
        step++;
    }
    else
    {
        reset();
    }
}

```

Figur 107 Del 5 x10modtager.c



```

x10modtager.h      main.c      x10modtager.c*  ✘ X
➔ x10modtager.c
C:\Users\mortenovergaard\Documents\Atmel Studio\7.0\Modtager_kode\Modtager_kode\x10modtager.c

void compare_command()
{
    if (signal[36] == unit_adr_on[0] && signal[37] == unit_adr_on[1] && signal[38] == unit_adr_on[2] && signal[39] == unit_adr_on[3] && signal[40] == unit_adr_on[4] && signal[41] == unit_adr_on[5] && signal[42] == unit_adr_on[6] && signal[43] == unit_adr_on[7])
    {
        command = 1;
        step++;
    }
    if (signal[36] == unit_adr_off[0] && signal[37] == unit_adr_off[1] && signal[38] == unit_adr_off[2] && signal[39] == unit_adr_off[3] && signal[40] == unit_adr_off[4] && signal[41] == unit_adr_off[5] && signal[42] == unit_adr_off[6] && signal[43] == unit_adr_off[7])
    {
        command = 2;
        step++;
    }
    else
    {
        reset();
    }
}

```

Figur 108 Del 6 x10modtager.c



```
#define F_CPU 16000000
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>

#include "x10modtager.h"

ISR(INT0_vect)
{
    readpin();
    readarr();
}

int main(void)
{
    init_X10_prot();
    while (1)
    {
        if (getstep() == 0)
        {
            setstartbit();
        }
        else if (getstep() == 1)
        {
            checkstartbit();
        }

        else if (getstep() == 2)
        {
            readarr();
        }
        else if (getstep() == 3)
        {
            compare_unit_adr();
        }
        else if (getstep() == 4)
        {
            compare_command();
        }
        else if (getstep() == 5)
        {
            cmd();
        }
    }
}
```

Figur 109 Main for x10modtager

9.2.3 Modultest

For at teste X10 modtageren uafhængigt af hardwaren er der lavet en version hvor koden til X10 modtageren benytter brugerinput i stedet for at læse på den pin der modtager bursts fra senderen.

```
#include "modtager.h"

int main(void)
{
    init_X10_prot();
    while(1)
        if (getstep() == 0)
        {
            startinput();

        }
        else if (getstep() == 1)
        {
            readarr();
        }

        else if (getstep() == 2)
        {
            compare_unit_addr();
        }

        else if (getstep() == 3)
        {
            compare_command();
        }

        else if (getstep() == 4)
        {
            cmd();
        }
    return 0;
}
```

Figur 110 Main for testprogrammet. Her er interrupt rutinen naturligvis udeladt.

I vores main kan det ses at der bruges en integer der hedder step til at køre igennem programmet. Step sikrer at programmet skrider frem i korrekt rækkefølge.



Som det kan ses startes der ud med startinput() hvilket fremgår herunder.

```
Input 1 for burst eller 0 for intet burst
1
1
1
0
startbit godkendt, indtast nu husadresse
step =1
    input 1 for burst eller 0 for intet burst
```

Figur 111 Her ses input af startbit og godkendelse heraf.

Disse startbits tjekkes også for i det rigtige program, og her kan det ses at step på korrekt vis inkrementeres, og dermed kan forsætte programmet ved at køre vores readarr(). I testprogrammet håndteres sammenligningen i samme funktion, hvor det i Atmel programmet håndteres i 2 funktioner.

Figur 112 Herover ses indtastningen af arrayet i funktionen `readarr()`, derudover så udføres begge `compare`-funktioner og `cmd()`.

Som det kan ses, så indlæses arrayet på samme måde i både testprogram og rigtige program. Den største forskel er umiddelbart at man tjekker vores compare arrays i en forløkke og stiller det op med de hardcodede X10 commands i Atmel programmet. I testprogrammet er der ikke lavet en for løkke, men man sammenligner i stedet med de enkelte dele i arrayet som det fremgår på figur 98113 herunder. Derudover håndteres startbits i 2 funktioner i Atmel programmet da disse skal bekræftes før resten af programmet køres. I testprogrammet udføres dette i en enkelt funktion.

```

if (signal[36] == unit_adr_on[0] && signal[37] == unit_adr_on[1] && signal[38] == unit_adr_on[2] && signal[39] == unit_adr_on[3] &&
    signal[40] == unit_adr_on[4] && signal[41] == unit_adr_on[5] && signal[42] == unit_adr_on[6] && signal[43] == unit_adr_on[7])
{
    command = 1;
    step++;
}
if (signal[36] == unit_adr_off[0] && signal[37] == unit_adr_off[1] && signal[38] == unit_adr_off[2] && signal[39] == unit_adr_off[3] &&
    signal[40] == unit_adr_off[4] && signal[41] == unit_adr_off[5] && signal[42] == unit_adr_off[6] && signal[43] == unit_adr_off[7])
{
    command = 2;
    step++;
}

```

Figur 113 Her ses en del af compare command funktionen.

```

1  #include <iostream>
2  #include "Modtager.h"
3  using namespace std;
4
5  const int house_a[8] = { 0,1,1,0,1,0,0,1 };
6
7  // Start og stop bit
8
9  const int start_bit[4] = { 1,1,1,0 };
10 const int stop_bit[6] = { 0,0,0,0,0,0 };
11
12 // funktioner
13
14 const int unit_adr_on[8] = { 0,1,0,1,1,0,0,1 };
15 const int unit_adr_off[8] = { 0,1,0,1,1,0,1,0 };
16
17 // Unit addresser
18
19 const int adr_1[8] = { 1,0,1,0,1,0,0,1 };
20
21
22 // test arrays
23 int starttester[4] = { 0,0,0,0 };
24 int signal[52];

```

Figur 114 Herover ses de hardcodede X10 segmenter vi sammenligner med

Programmet genstartes grundet fejl i input
Input 1 for burst eller 0 for intet burst

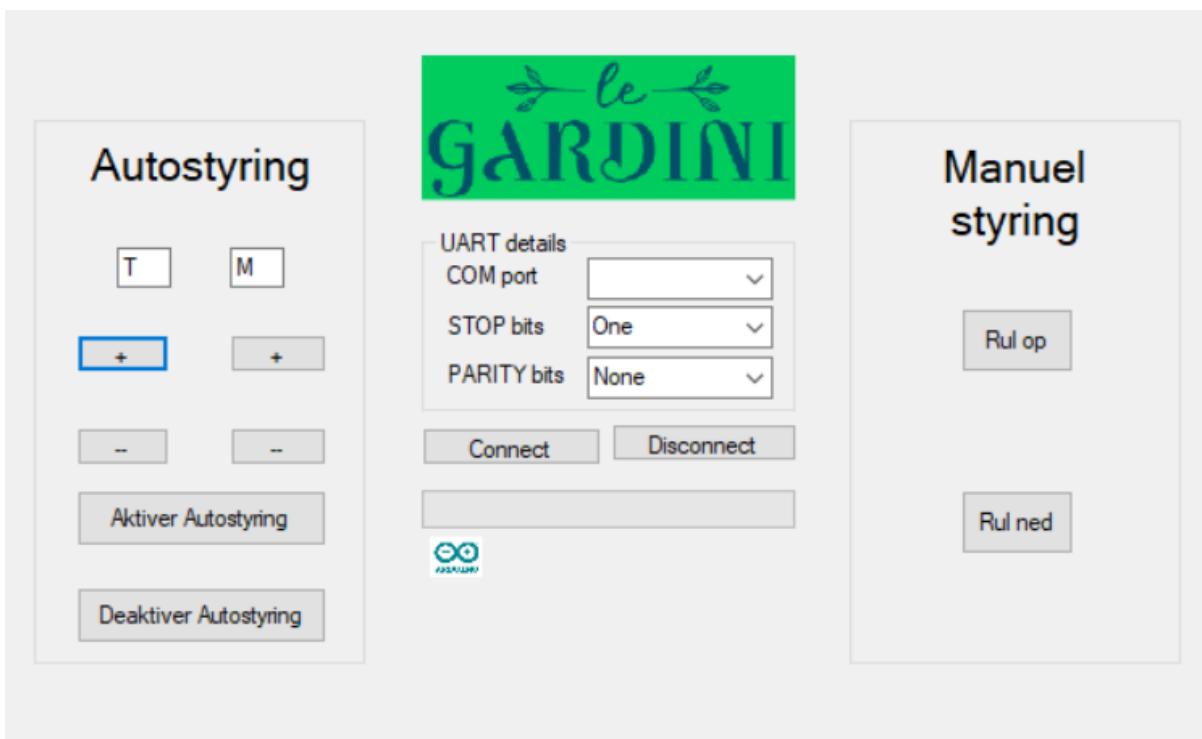
Figur 115 Et billede af fejlinput i signal arrayet

Det kan derfor konkluderes at koden virker før interrupt rutine og indlæsning på pin er indført. Det havde naturligvis været mere ideelt hvis der var brugt en Analog Discovery til at imitere zerocross signalet på interrupt pinnen, så der kunne testes mere nøjagtigt. Dog blev dette ikke aktuelt da det var at foretrække at brugerinput kom igennem en terminal.

9.3 GUI

9.3.1 Design

Den GUI bruger benytter til at interagere med systemet implementeres i C# via visual studios drag n drop design system.



Figur 116 - GUI

Designet i C# foregår ikke som funktioner, men som events. Herunder er beskrivelser af de eventhandlers der knytter sig til knapperne på GUI'en som set herover.

Form1_Load

Skal initiere tilkoblede COM-porte så de kan vælges under COM port dropdown menuen.
Derudover forhåndsindstilles timer og minutter displaysne til henholdsvis "T" og "M".
Der skal låses for autostyringens knapper.

btnHourIncrement_Click

skal kunne inkrementere timer displayet med én hver gang der trykkes. Hvis timer displayet er på 23 skal den ikke kunne inkrementeres yderligere.

btnHourDecrement_Click

skal kunne dekrementere timer displayet med én hver gang der trykkes. Hvis timer displayet er på 0 skal den ikke kunne dekrementeres yderligere.

btnMinuteDecrement_Click

skal kunne dekrementere minut-displayet med en hver gang der trykkes. Hvis minut-displayet går under 0 og timer displayet er over nul, skal timer displayet dekrementeres med én og minutter sættes til 59.

btnMinuteIncrement_Click

skal kunne inkrementere minut-displayet med én hver gang der trykkes. Hvis minut-displayet går over 59 og timer-displayet er under 23 skal timer displayet inkrementeres og minut-displayet sættes til 0.

btnActivateAuto_Click

påbegynder en timer med de brugerindtastede timer og minutter-værdier, og låser increment og decrement knapperne for timer og minutter mens timeren er i gang.

btnDeactivateAuto_Click

stopper timeren og låser op for increment og decrement knapperne for timer og minutter.

timer1_Tick

ticker minut-værdien ned hvert minut (under tests af timeren af dette sat til hvert sekund). Hvis minutter rammer 0 og timer er over 0 dekrementeres timer og minutter sættes til 59. Er både minutter og timer 0 sendes en command. Er det første command der bliver sendt eller onCommand var den sidste command til at blive sendt, sendes en onCommand der ruller gardinet ned. Er onCommand den sidste der blev sendt, sendes offCommand der ruller den op.

btnConnect_Click

skal oprette forbindelse til arduinoen senderkoden ligger på. Sætter baudrate til 9600. 8 databits. Benytter sig af brugerinput ved com-port, samt paritybits og stopbits. Disse skal have standardinput men kan ændres af brugeren.

herefter åbnes en UART-forbindelse til arduinoen. Er denne succesfuld fyldes progressbaren. Her åbnes også op for at GUI'en kan modtage uart-signaler.

btnDisconnect_Click

UART-forbindelsen afbrydes og progressbaren tømmes.

btnManuelOp_Click

Skal sende en offCommand for at rulle op. Er offCommand den sidste der blev kaldt, skal den ikke gøre noget

btnManuelNed_Click

Skal sende en onCommand for at rulle ned. Er dette den først command der sendes eller var onCommand den sidste command der blev kaldt, skal den ikke gøre noget

serialPort1_DataReceived

skal modtage data fra arduinoen gennem en UART-forbindelse. Dataen konverteres til en integer og sendes til showData evenhandleren

showData

Hvis data variablen er over 0 skal der låses op for autostyringens knapper

9.3.2 Implementering

Form1_Load

```
private void Form1_Load(object sender, EventArgs e)
{
    timerLbl.Text = "T";
    minuteLbl.Text = "M";
    string[] ports = SerialPort.GetPortNames();
    cboxCOMport.Items.AddRange(ports);
    btnHourDecrement.Enabled = false;
    btnMinuteDecrement.Enabled = false;
    btnHourIncrement.Enabled = false;
    btnMinuteIncrement.Enabled = false;
    btnActivateAuto.Enabled = false;
    btnDeactivateAuto.Enabled = false;
}
```

Figur 117



btnHourIncrement_Click

```
private void btnHourIncrement_Click(object sender, EventArgs e)
{
    if (timer == 23)
    {
        timer = 00;
    }
    else
    {
        timer++;
    }

    timerLbl.Text = timer.ToString();
}
```

Figur 118

btnHourDecrement_Click

```
private void btnHourDecrement_Click(object sender, EventArgs e)
{
    if(timer == 00)
    {
        timer = 23;
    }
    else
    {
        timer--;
    }

    timerLbl.Text = timer.ToString();
}
```

Figur 119



btnMinuteDecrement_Click

```
private void btnMinuteDecrement_Click(object sender, EventArgs e)
{
    if (minutter == 00)
    {
        if (timer > 00){
            timer--;
            minutter = 59;
        }
        else
        {
            minutter = 00;
        }
    }
    else
    {
        minutter--;
    }
    minutelbl.Text = minutter.ToString();
    timerlbl.Text = timer.ToString();
}
```

Figur 120

btnMinuteIncrement_Click

```
private void btnMinuteIncrement_Click(object sender, EventArgs e)
{
    if (timer==23)
    {
        timer = 23;
        minutter = 59;
    }
    else if(minutter>=59)
    {
        timer++;
        minutter = 0;
    }
    else
    {
        minutter++;
    }
    minutelbl.Text = minutter.ToString();
    timerlbl.Text = timer.ToString();
}
```

Figur 121

btnActivateAuto_Click

```
private void btnActivateAuto_Click(object sender, EventArgs e)
{
    timerTimer = timer;
    timerMinutter = minutter;
    timer1.Enabled = true;
    timer1.Start();
    btnHourDecrement.Enabled = false;
    btnMinuteDecrement.Enabled = false;
    btnHourIncrement.Enabled = false;
    btnMinuteIncrement.Enabled = false;
}
```

Figur 122



btnDeactivateAuto_Click

```
private void btnDeactivateAuto_Click(object sender, EventArgs e)
{
    timer1.Stop();
    btnHourDecrement.Enabled = true;
    btnMinuteDecrement.Enabled = true;
    btnHourIncrement.Enabled = true;
    btnMinuteIncrement.Enabled = true;
}
```

Figur 123

timer1_Tick

```
private void timer1_Tick(object sender, EventArgs e)
{
    timerMinutter--;
    if (timerMinutter < 0)
    {
        if (timerTimer > 0)
        {
            timerTimer--;
            timerMinutter = 59;
        }
    }
    if (timerTimer == 0 && timerMinutter < 0)
    {
        timerTimer = timer;
        timerMinutter = minutter;
        if (serialPort1.IsOpen)
        {
            switch (output_counter)
            {
                case 0:
                    serialPort1.Write(onCommand, 0, 1);
                    Debug.WriteLine(onCommand);
                    output_counter = 1;
                    break;
                case 1:
                    serialPort1.Write(offCommand, 0, 1);
                    Debug.WriteLine(offCommand);
                    output_counter = 0;
                    break;
            }
        }
        minutelbl.Text = timerMinutter.ToString();
        timerlbl.Text = timerTimer.ToString();
    }
}
```

Figur 124

btnConnect_Click

```
private void btnConnect_Click(object sender, EventArgs e)
{
    try
    {
        serialPort1.PortName = cboxCOMport.Text;
        serialPort1.BaudRate = 9600;
        serialPort1.DataBits = 8;
        serialPort1.Parity = (Parity)Enum.Parse(typeof(Parity), cboxPARITY.Text);
        serialPort1.StopBits = (StopBits)Enum.Parse(typeof(StopBits), cboxSTOP.Text);

        serialPort1.Open();

        connectionProgress.Value = 100;

        serialPort1.DataReceived += new SerialDataReceivedEventHandler(serialPort1_DataReceived);
    }
    catch (Exception err)
    {
        MessageBox.Show(err.Message, "Error, fill out all options", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

Figur 125

btnDisconnect_Click

```
private void btnDisconnect_Click(object sender, EventArgs e)
{
    serialPort1.Close();
    connectionProgress.Value = 0;
}
```

Figur 126

btnManuelOp_Click

```
private void btnManuelOp_Click(object sender, EventArgs e)
{
    if (output_counter == 1)
    {
        serialPort1.Write(onCommand, 0, 1);
        output_counter = 0;
    }
}
```

Figur 127

btnManuelNed_Click



```
private void btnManuelNed_Click(object sender, EventArgs e)
{
    if (output_counter == 0)
    {
        serialPort1.WriteLine(onCommand, 0, 1);
        output_counter = 1;
    }
}
```

Figur 128

serialPort1_DataReceived

```
private void serialPort1_DataReceived(object sender, SerialDataReceivedEventArgs e)
{
    data = serialPort1.ReadByte();
    Debug.WriteLine(data);
    this.Invoke(new EventHandler(showData));
    btnHourDecrement.Enabled = true;
    btnMinuteDecrement.Enabled = true;
    btnHourIncrement.Enabled = true;
    btnMinuteIncrement.Enabled = true;
    btnActivateAuto.Enabled = true;
    btnDeactivateAuto.Enabled = true;
    Debug.WriteLine(data);
}
```

Figur 129

showData

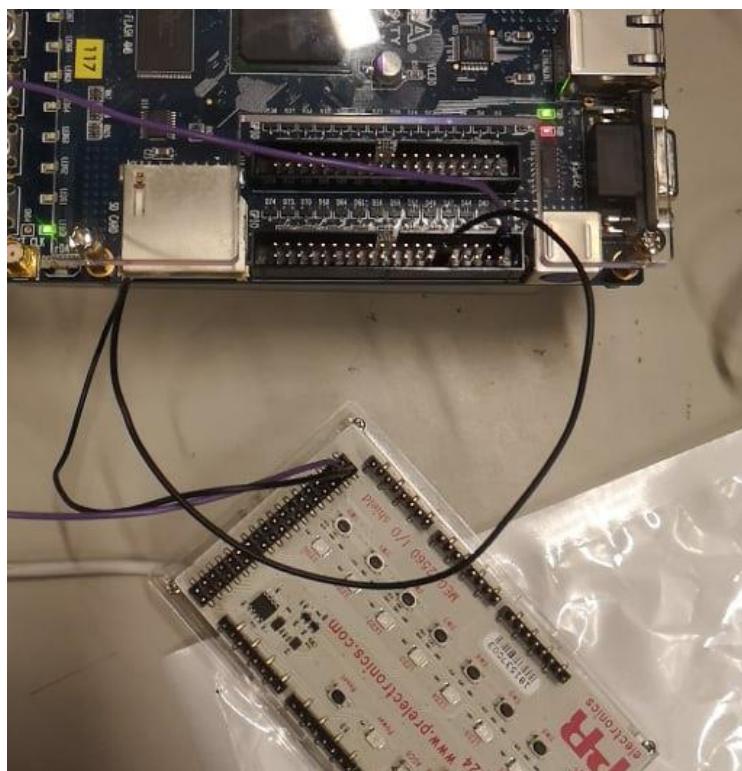
```
private void showData(object sender, EventArgs e)
{
    if (data>0)
    {
        btnHourDecrement.Enabled = true;
        btnMinuteDecrement.Enabled = true;
        btnHourIncrement.Enabled = true;
        btnMinuteIncrement.Enabled = true;
        btnActivateAuto.Enabled = true;
        btnDeactivateAuto.Enabled = true;
        Debug.WriteLine(data);
    }
}
```

Figur 130

9.3.3 Modultest

Viv i modultesten teste om kodelåsen kan låse op for GUI'ens autostyring. Derudover er GUI'ens egenskab til at forbinde til arduinoens og om den kan sende en bitstreng testet under senderens modultest.

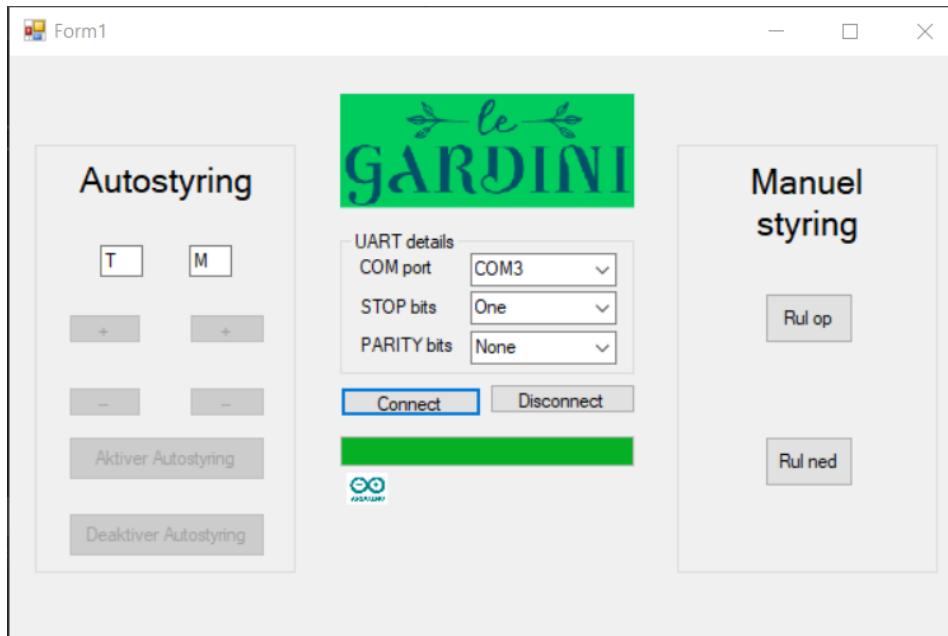
Modultest af kodelås til GUI



Figur 131

Her er et GPIO-ben fra kodelåsen forbundet til PORTb's pin 0 og common ground imellem dem.

Når arduinoen læser et højt signal på denne pin, vil den sende et højt signal til GUI'en og låse op for GUI'ens autostyring.



Figur 132

På GUI'en kan vi her se at der ikke bliver låst op.

Hvis vi manuelt sætter PORTB til at være høj vha. kode, kan vi se at autostyringen bliver låst op.

Efter en måling på den GPIO-pin fra DE2-boardet mäter vi kun 3.2 volt, hvilket ligger i den lave ende af ved en arduino ville tolke som et højt signal. Dette kan være et muligt svar på hvorfor det ikke virker.

9.4 Kodelås

For softwaren til kodelåsen er der taget udgangspunkt i Journal 4, opgave 7 lavet i faget DSD. Dertil har vi lavet nogle få ændringer.

```
-- port mapping code_lock_wc
UUT1: entity code_lock_wc port map
  (clk => CLOCK_50,
  reset => KEY(1),
  code => SW (3 downto 0),
  enter => KEY(2),
  lock => LEDG(0),
  GPIO1 => GPIO_1,
  tester1 => LEDR(17),
  tester2 => LEDR(16),
  tester3 => LEDR(15),
  tester4 => LEDR(14),
  tester5 => LEDR(13),
  tester6 => LEDR(12)
);

end;
```

Figur 133 GPIO01 forbindes med GPIO_1, så benene fungerer som output

Her ses port mapping, hvor GPIO01 er tilføjet og sættes til GPIO_1, hvilket er en række pins på DE2-boardet.

```
case present_state is
when idle =>

when eva_code_1 =>
  tester1 <= '1';

when get_code_2 =>
  tester2 <= '1';

when eva_code_2 =>
  tester3 <= '1';

when unlocked =>
  lock <= '0';
  GPIO1 <= "1";

when wrong_codes =>
  err_event <= '1'; -- eksempel på implementering af effect

when permanently_locked =>
  tester5 <= '1';

when going_idle =>
  tester4 <= '1';

when others =>
  tester1 <= '0';
  tester2 <= '0';
  tester3 <= '0';
  tester4 <= '0';
  tester5 <= '0';
  lock <= '1';
  reset_wc <= '0';

end case;
end process;
```

Figur 134 GPIO01 sættes til 1, eller high, ved unlock

Her kan det ses at vi sætter GPIO1 som "1", eller high, når kodelåsen er unlocked. Dette gør at vi kan læse en high på den designerede pin. Dette overføres så gennem en microcontroller til GUI'en.

9.4 Resultater software modultest

Kodelås	<p>Kodelåsen er testet visuelt, ved at indtaste både korrekt og forkert kode, og dernæst observere om kodelåsen låser op, eller forbliver låst. Derudover måles der spænd ind på GPIO1_0 benet.</p>	<p>Kodelåsen fungerer som forventet. Dette ses ved at der skiftes til "Unlock" stadiet, når de 2 korrekte koder er indtastet. Ved forkert kode kommer man tilbage til "Idle", og ved 3 forkerte forsøg kommer man i "Permanently locked". Når koden er korrekt tastet ind kan vi ligeledes måle en spænding på 3.3 volt på GPIO1_0</p>
GUI	<p>GUI'ens forbindelse med arduinoen er testet visuelt ved at se på progressbaren på GUI'en.</p> <p>På samme måde er kodelåsens forbindelse til GUI'en testet, ved at se om autostyringen låses op ved korrekt kode.</p> <p>Slutvis er dens evne til at sende en command til senderen testet.</p>	<p>GUI'en lavede succesfulde test ved forbindelse til arduino, samt dens evne til at sende commands.</p> <p>Det var derimod ikke ikke muligt at låse GUI'ens autostyring op vha. kodelåsen på DE2-board.</p>
X10 sender software	<p>Senderen er testet ved visuelt at se om den kan forbindes til GUI'en, om de hardcodede bitstrenge loades i signal-arrayet ordentligt og om den sender</p>	<p>Senderen kunne forbindes succesfuldt til GUI'en, signal-arrayet loadede de hardcodede bitstrenge korrekt, og den sendste bursts korrekt ved zero-crosses. Det var muligt</p>

	bursts korrekt ved zero-crosses.	at afkodede startbits og en del af en bitstreng.
X10 modtager software	Modtager softwaren er testet ved at lave et program der er en næsten tro kopi af koden, blot hvor der bruges brugerinput i stedet for at man modtager bursts. Der tager her ikke højde for zero crossings, men udelukkende bursts, hvorfor der altså er forskelle på testprogram og Atmel program.	Testprogrammet fungerer efter hensigten før interrupt tilføjes og man skal læse på et ben. Med blot brugerinputs lykkedes det at komme succesfuldt igennem programmet, og både CMD(1) og CMD(2) virker. Den melder også fejl såfremt inputs er forkerte, hvilket medfører reset af programmet.

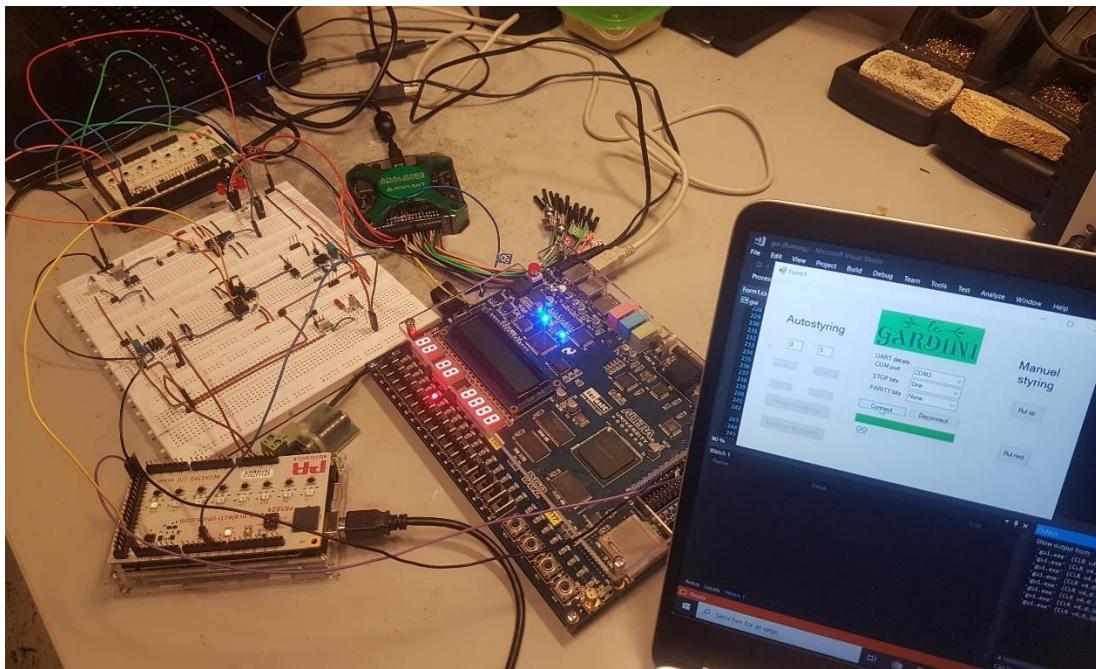
10.0 Integrationstest

I følgende afsnit bliver hvert modul testet sammen i en serie af opstillinger for at give indtryk af hvad der fungerede og hvad der forårsagede problemer. For at give et bedre overblik er opstillingerne indsat i en tabel.

Opstilling	Beskrivelse af opstilling	Testresultater
1. opstilling	Her testes GUI sammen med sender-softwaren. Her simuleres et zero cross ved brug af Analog Discovery.	Det ses når der måles på udgangen af mega2560 mikrocontrolleren, at den ønskede sender-kode sendes når autostyring startes på GUI'en. Dog så vi at senderen sender den samme kodestreng hver gang, hvor den ellers skulle have vekslet imellem kodestreng for rul op og rul ned
2. opstilling	Her testes kodelåsen sammen med GUI'en. Kodelåsen kobles på senderens mega2560 mikrocontroller der er koblet til computeren med GUI	Kodelåsen og GUI'en spillede ikke sammen. Hver del virker for sig, men arduinoen genkendte ikke signalet fra kodelåsen. Kodelåsen vil ikke medvirke i resten af integrationen
3. opstilling	Zero cross testes sammen med sendersoftwaren. Mikrocontrolleren tilsluttes Vcc og ground på zero cross kredsløbet. Elnettet tilkobles zero cross kredsløbet. Mikrocontrollerens interrupt 0 pin tilkobles zero crossens outputus.	Det ses at sender-koden og zero cross fra elnettet spiller sammen. Der sendes kun burst ved hver zero cross og den ønskede bit-streng ses på Mikrocontrollerens output.

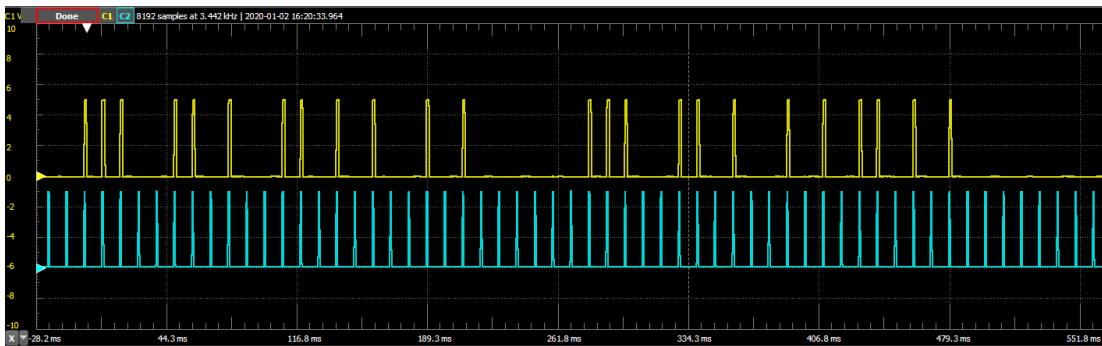
4. opstilling	Herefter sammenkobles senderkredsløbet, sendersoftwaren og zero cross. Der måles på 18V elnettet med Analog Discovery.	Ved måling på elnettet ses det, hvordan en bitstreng bliver sendt igennem, når timeren for autostyring rammer 0.
5. opstilling	Modtager kredsløbet tilkobles forrige opstilling. Der måles med Analog Discovery på udgangen af kredsløbet	Det ses på udgangen af modtager-kredsløbet, at elnettets 50Hz signal er filteret fra og et burst ses som en 5V streng i stedet for at svinge mellem 0-5V. En hel X10 kommando aflæses og godkendes. Se figur 136
6. opstilling	Til sidst tilkobles modtagerens Mikrocontroller og dertilhørende modtagersoftware på modtager kredsløbet. Modtagerens output tilkobles to LED'er, som bruges i stedet for gardinkredsløbet.	LED'erne tændes ikke. Der konkluderes at modtager softwaren ikke kan aflæse/forstå X10 kommandoerne.

Nedstående på figur 104135 kan den samlede integrations af alle moduler ses. Som nævnt overstående, er alle moduler ikke integreret med succes. Dog kan billedet bidrage til at give et visuelt indblik i hvordan integrationen har fundet sted.



Figur 135 - Samlet integration

Nedstående på figur 136 ses resultatet af opstilling 5. Her blev der sendt et vellykket bitstreng.



Figur 136 - Rigtig bitstrenge. opstilling 5.

11.0 resultater

Som det fremgår af integrationstesten, så er der en række moduler der ikke spiller sammen. Af denne grund har det ikke været muligt at gennemføre samtlige accepttests.

11.1 Accepttestresultater

Testes med vejleder

11.1.1 Use case 1: Login

Use case under test:	Login		
Scenarie:	Hovedscenarie		
Samtidige forekomster	Ingen		
Prækondition	Superbruger er ikke logget ind		
Postkondition	Superbruger er logget ind, og har nu Superbruger privilegier		
Handling:	Forventet observation/resultat	Faktisk observation/resultat	Vurdering (OK / Fail)
Superbrugerens skriver sin 4-cifret kode ind på kodelåsen	Grøn lampe stopper med at lyse, og brugergrænsefladen låser op for indstillinger for autostyring.	Grøn lampe stopper med at lyse på kodelåsen, men brugergrænsefladens indstillinger for autostyring forbliver låst	Fail

Tabel 19 - Accepttest Use case 1

Use case under test:	Login
Scenarie:	Udvidelse 1a
Samtidige forekomster	Ingen
Prækondition	Superbruger er ikke logget ind
Postkondition	Superbruger er ikke logget ind

Handling:	Forventet observation/resultat	Faktisk observation/resultat	Vurdering (OK / Fail)
Superbrugeren skriver en forkert kode ind på kodelåsen	Rød lampe lyser i 5 sekunder, og tidsintervalindstillingerne forbliver låst.	Ikke testet, da kodelås og GUI ikke snakker sammen.	Fail

Tabel 20 - Accepttest Use case 1. Udvidelse 1a

11.1.2 Use case 2: Aktiver autostyring

Use case under test:	Aktiver autostyring
Scenarie:	Hovedscenarie
Samtidige forekomster	Ingen
Prækondition:	Superbrugeren har logget ind korrekt, og systemet er deaktiveret.
Postkondition:	Systemet er aktiveret, og kører automatisk

Handling:	Forventet observation/resultat	Faktisk observation/resultat	Vurdering (OK / Fail)
Superbrugeren indstiller intervallet for ændring af tilstand (rul op hvis gardinet er nede og rul ned hvis gardinet er oppe)	Autostyringen ændrer tilstand i det korrekte interval.	Nedtællingen når 0, men autostyring ændrer ikke tilstand.	Fail

Tabel 21 - Accepttest Use case 2

11.1.3 Use case 3: Deaktiver autostyring

Use case under test:	Deaktiver autostyring
Scenarie:	Hovedscenarie
Samtidige forekomster	ingen

Prækondition	Superbrugeren har logget ind korrekt, og systemet er aktiveret		
Postkondition	Systemet er deaktiveret		
Handling:	Forventet observation/resultat	Faktisk observation/resultat	Vurdering (OK / Fail)
Superbrugeren trykker på ”deaktiver autostyring” på brugergrænsefladen.	Autostyringen er deaktiveret	Timeren stopper med at tælle ned	OK

Tabel 22 - Accepttest Use case 3

11.1.4 Use case 4: Rul op / ned

Use case under test:	Rul op/ned		
Scenarie:	Hovedscenarie		
Samtidige forekomster	Ingen		
Prækondition	Brugeren eller superbrugeren har adgang til brugerfladen		
Postkondition	Gardinet ruller op eller ned		
Handling:	Forventet observation/resultat	Faktisk observation/resultat	Vurdering (OK / Fail)
Bruger/superbrugeren åbner op for brugergrænsefladen	Brugergrænsefladen åbner på computeren.	Brugergrænsefladen åbner på computeren	OK
Bruger/superbrugeren trykker på knappen ”rul op”	Gardinet ruller op	Uden fungerende modtager-controller kode er denne UC ikke opfyldt	Fail
Alternativ:			

Bruger/superbrugeren åbner op for brugergrænsefladen	Brugergrænsefladen åbner på computeren.	Brugergrænsefladen åbner på computeren	OK
Bruger/superbrugeren trykker på "rul ned" knappen	Gardinet ruller ned	Uden fungerende modtager-controller kode er denne UC ikke opfyldt	Fail

Tabel 23 - Accepttest Use case 4

11.1.6 Accepttest ikke-funktionelle krav

Udstyr til test af krav:

- Multimeter (bruges til at måle spænding)
- Målebånd (bruges til at måle længder)
- Stopur (bruges til tidtagning)

No.	Krav	Test / udførelse	Faktisk observation / resultat	Vurdering (OK/FAIL)
U1	Der medfølger en brugermanual til systemet	Der er en brugermanual	Testes ikke	
U2	Det skal være muligt at finde samtlige GUI funktioner i selvsamme manual	GUI funktioner findes i manualen	Testes ikke	
R1	Systemet skal kunne udføre UC4 mindst 15 gange	UC4 udføres 15 gange	Testes ikke	

	inden der skal skiftes dele			
R2	Systemet skal kunne gennemføre UC1 mindst 15 gange inden der skal skiftes dele	UC1 udføres 15 gange	Testes ikke	
P1	Systemet skal kunne rulle helt op og helt ned, til en afstand af 1 centimeter fra vinduets top eller bund	UC4 udføres og afstanden til top / bund måles med målebånd	Testes ikke, idet Alexander ikke længere er en del af gruppen.	
P2	Gardinet skal kunne rulle op med en hastighed på 7 cm / s	UC4 udføres og der tages tid med stopur. Herefter beregnes hastigheden	Testes ikke, idet Alexander ikke længere er en del af gruppen.	
S1	Systemet skal kunne repareres på maksimalt 1 time (MTTR)	Med stopur tages der tid på hvor længe det tager at reparere systemet	Testes ikke	
S2	Systemets fysiske dele bør kunne udskiftes på 30 minutter	Der tages tid med stopur og fysiske dele udskiftes	Testes ikke	
M1	Gardinets motor opererer ved en gennemsnitseffekt på $100W \pm 5W$	Motorens gennemsnitseffekt måles med multimeter	Testes ikke, idet Alexander ikke længere er en del af gruppen.	
M2	Motoren tager 100 steps pr. 1 meter gradin ± 2 centimeter	Motoren indstilles til at køre 100 steps og herefter måles gardinets længde med målebånd	Testes ikke, idet Alexander ikke længere er en del af gruppen.	

K1	Ved forkert kode skal en rød LED lyse. (LEDR(13))	Forkert kode indtastes	Rød LED lys ved indtast af forkert kode	OK
K2	Ved rigtig kode skal en grøn LED stoppe med at lyse. (LEDG(0))	Korrekt kode indtastes	Grøn LED stopper med at lyse ved indtast af rigtig kode	OK
K3	På kodelåsen (DE2) skal KEY(0...3) bruges til indtastning af koden	Koden til kodelåsen indtastes på de 4 taster	Kodelåsen låser op efter indtastning af koden på de 4 taster	OK
K4	Koden til kodelåsen skal være "1111"	Koden "1111" indtastes på kodelåsen og der låses op	Der låses op med den indtastede kode	OK
X1	Det simulerede elnet som X-10 opererer på skal have en Vrms på $18V \pm 4V$	Spændingen over det simulerede elnet måles med multimeter	Der måles en spænding på 21,37V	OK

Tabel 24 - Accepttest af ikke-funktionelle krav

12.0 Diskussion

I dette afsnit vil vores resultater og problematikker blive diskuteret, og der vil også gives en samlet vurdering af de opnåede resultater med relation til projektets problemformulering

Projektproblematikker

Da vores use cases i høj grad er baseret på et funktionelt produkt med et gardin, og da ham der stod for gardinstyring, droppede ud, er vores accepttest ikke blevet godkendt i sidste ende. Dog har vi lavet systemets hardware funktionelt op til det punkt hvor gardinet kommer på, dog er der væsentlige problemer med modtager-softwaren, og der er problemer med at integrerer kodelåsen i sender og GUI softwaren. I stedet for at styre et gardin, har vi koblet 2 LED'er på, én til hver kommando, for at realisere produktet. Vi har derfor ikke helt nået det mål vi havde håbet på, da vi begyndte på projektet, men har alligevel fået lavet et produkt der kan kommunikere over det simulerede elnettet.

Hardwareproblematikker

I dette projekt har hardwaredelen skabt en del problemer i forhold til de filtre der er blevet brugt, og den operationsforstærker der er blevet brugt.

En af problemerne med filtrene, er at de påvirker hinanden når de er koblet sammen via elnettet, det påvirker knækfrekvensen og den dæmpning der opleves i pasbåndet. For vores hardware betyder det at der er nogle afvigelser i forhold til beregnede værdier, og målte.

Det problem der har været med operationsforstærkeren, har været først at finde den rigtige, som kan klare de frekvenser vi bruger den til, og derefter designe valget om hvorvidt den skal +5V og -5V eller kun +5V og stel, samt hvor stort "gain" den skal have. Dette design valg er vigtigt da den modtager negative spændinger, og da de valg har nogle fordele og ulemper i forhold til hvordan udgangssignalet ender med at være.

Softwareproblematikker

I udførslen af projektet er softwaren blevet skrevet i C, hvilket indledningsvis virkede som en god idé. I UART-forbindelsen mellem GUI og sender kode troede vi først det var muligt at sende tal eller strenge, og fordi C sproget tolker "1" som "true" og "0" som false, var ideen at C ville være et nemmere sprog at skrive både sender og modtager kode i.

Dette resulterede i at vi bla. ikke kunne skrive objektorienteret kode, og klassediagrammer blot er brugt for at skabe systematik i c-koden, da vi jo af gode grunde ikke kan have klasser. Vi fandt sidenhen ud af at vores antagelse omkring UART-forbindelsen nemt kunne arbejdes udenom, og C++ og objektorienteret programmering med klasser klart havde været nemmere at arbejde med.

Derudover har det været meget problematisk at teste modtagersoftwaren før i slutfasen, da tests er taget meget sekventielt i rækkefølgen det kommer i, i systemet. Det betyder at er er blevet brugt meget tid på at få sendersoftware til at sende det korrekte, og først derefter

har der været mulighed for at teste modtager software i systemet. Derfor er der udviklet et testprogram, men dette blev først påbegyndt sent i processen da der var forhåbning om at resten af systemet ville virke tidligere.

Under integrationstesten mellem GUI'en, senderen og sender-modtager hardware kunne vi se at senderen sendte den samme bitstreng hver gang, i stedet for først at sende en rul-ned-command og derefter en rul-op-command. Vi regner med at det skyldes forvirring ifht. Signalet mellem GUI og senderkode. Vi formåede at senderen fortolker alle signaler modtaget som "true" og derfor bare sender den første kode, og aldrig skifter. Grundet tidspres har vi ikke kunne undersøge det nærmere.

Det har heller ikke været muligt at låse autostyringen på GUI'en op vha. kodelåsen.

Da vi prøvede at lade de pins senderen læser på gå høj vha. kode, kunne vi godt låse op for autostyringen.

efterfølgende har vi kunne måle et output på 3.2 volt fra kodelåsen når der blev indtastet korrekt kode. Dette kan ligge i den lave ende, eller under, hvad en arduino tolker som et højt signal, og det er muligt at dette er problemet. Vi har ikke været i stand til at løse problemet.

Procesproblematikker

Der har i gruppen været en enighed om, at det har været svært at skulle beskrive et system med SysML diagrammer, som man ikke vidst hvordan der egentlig skulle opbygges og de problemer der ville komme. Dette har gjort at vi har lavet nogle diagrammer ud fra en ide om hvordan de forskellige blokke skulle opbygges, som i løbet af projektet er blevet lavet meget om, eftersom vi har fundet nogle andre løsninger end vi først havde forventet. Det har også betydet at vi har afviget fra vores planlagte tidsplan.

13.0 Konklusion

I dette afsnit vil der blive konkluderet på problemformuleringen. Her vil vi komme ind på forskellige punkter, heriblandt hvad der er lykkedes og hvad der ikke gjorde. Derudover vil der også blive konkluderet på den overordnede proces i løbet af projektet.

Der er blevet implementeret en færdig hardware prototype, bestående af X10 sender, X10 modtager og en zero cross detektor. Med fungerende sender software bliver kommandoen sendt perfekt gennem elnettet og behandlet af de dele der indgår i hardwaren. Det betyder også at der har været en fungerende prototype af sender software, men ikke for modtager softwaren. Derudover er der blevet implementeret en GUI til styring af selve X10 protokollen.

Grundet at én af vores gruppemedlemmerne er droppet ud, har vi ikke fået implementeret

et funktionelt gardin med stepper-motor, dog er diverse diagrammer og tekster stadig at finde i vores bilag.

Softwaren til henholdsvis sender og modtager er blevet skrevet i C kode og der er her udarbejdet funktioner til udførelsen og X10 protokollen.

Vores delsystemer kommunikerer som sagt igennem X10 protokollen. Her er der Hardcoded en *House address, unit address, Prefix, suffix og stop bits*. Disse bliver igennem X10 protokollen brugt til at sende forskellige commands til vores givne modtager modul, Heriblandt at rulle op og ned.

Det kan konkluderes at vi i løbet af vores projektarbejde har arbejdet ud fra ASE-udviklingsmodellen. Her er der som udgangspunkt arbejdet på en iterativ måde, for at bedst muligt kunne forbedre vores system. Derudover har vi som gruppe også draget nytte af vandfalds og V-modellen og disses arbejdstilgang.

I forhold til tidsplanen blev vi en del udfordret i slutningen af projektforløbet, da vi skulle lave integrationstest. Her blev softwarens implementering forsinket, hvilket gjorde at vi ikke havde så lang tid til at fejlfinde under integrationstesten af både hardware og software og derfor måtte udskyde det til efter julen.

Se proces beskrivelsen for de individuelle konklusioner¹⁴

13.1 Fremtidigt arbejde

I dette afsnit vil der kort blive beskrevet mulighederne i et fremtidigt arbejde.

I et fremtidigt arbejde ville man først og fremmest kunne færdiggøre vores system *Le gardini*. Dernæst ville det næste logiske step være at udvide systemet til samtlige gardiner i en husstand. Desuden kunne der udvikles endnu mere på systemet, så det ville være muligt at tilkoble andre tyveriforebyggelses systemer, såsom alarmer og lystilkobling.

¹⁴ Bilag 1, Proces beskrivelse

Referenceliste

1. Danmark Statistik:

<https://www.dst.dk/da/Statistik/nyt/NytHtml?cid=27998&fbclid=IwAR19ZocHGnW967xyCi0Ac4QCuYVPKmY5K00JF6FMk1MK48qV1qfQ-pdeBP4>

2. Application Note:

https://blackboard.au.dk/bbcswebdav/pid-2290252-dt-content-rid-6994102_1/courses/BB-Cou-UUVA-86257/AN236_ApplicationNote.pdf

3. Datasheet SN74AHC14:

<http://www.ti.com/lit/ds/symlink/sn74ahc14.pdf>

4. Datasheet db104 diodebridge:

<https://www.mouser.dk/datasheet/2/345/db101s-1012s-17410.pdf>

5. Datasheet PC817 optocoupler:

<https://www.farnell.com/datasheets/73758.pdf>

6. Datasheet BSX20 transistor

http://www.farnell.com/datasheets/1680009.pdf?_ga=2.195992940.1396614530.1537263236-1466402061.1525071283

7. Datasheet MCP601 Op-Amp:

<http://ww1.microchip.com/downloads/en/DeviceDoc/21314g.pdf>

8. Datasheet Arduino Mega 2560 (lang version):

https://ww1.microchip.com/downloads/en/devicedoc/atmel-2549-8-bit-avr-microcontroller-atmega640-1280-1281-2560-2561_datasheet.pdf

9. Envelope Detector, Wikipedia:

https://en.wikipedia.org/wiki/Envelope_detector

10. Arduino Mega 2560 datasheet (kort version)

<https://www.robotshop.com/media/files/pdf/arduinomega2560datasheet.pdf>

Bilag

Bilag er vedlagt rapporten og har som udgangspunkt samme navn som på følgende liste:

1. Proces Beskrivelse
2. Vejledning til færdiggørelse af projekt 2