AARHUS UNIVERSITET
AU ENGINEERING

Home Protection

Semesterprojektgruppe 5

Projektrapport

Studienummer:	***Deltagere:***
201905020	Arne Jan Broeders
201705103	Andreas Stavning Erslev
201709351	Christian Bach Johansen
201905236	Thomas Stenholt Laursen
201905424	Mads Nørregaard Morratz
201906146	Frederik Thomsen
201807198	Shynthavi Prithviraj

Vejleder

Gunvor Elisabeth Kirkelund

Afleveringsdato

15-06-2020

Antal sider: 57

Antal tegn med mellemrum: 71728

1. Abstract

In this project, work has been done on the development of a break-in prevention system linked to the use of an X10 protocol. The purpose of this report will be to illustrate and explain the uses of the X10 protocol in conjunction with related hardware and software modules. The aim of the system is to prevent theft by automating a gradually regulated lamp as well as a switch that can control electrical appliances, to imitate the presence of activity in the home.

Software and hardware modules have been developed for the system, and the report will contain in-depth descriptions of software protocols as well as a review of hardware modules and their structure. The interaction between hardware and software is mainly controlled through a set of microcontrollers that communicate via transmitter -and receiver modules across the powerline. This allows a signal to be sent from the system computer which can control two electrically powered units.

The first device, Switch, allows switching on and off for a connected device, while the second module, Dimmer, can vary the duty-cycle of the output. Their primary functions are used in conjunction with time settings, thereby mimicking human use of the connected devices.

The system also allows you to adjust when the appliances are switched on as well as the brightness of the lamps. To access this feature, input of a password is required through the associated DE2-Board, which communicates with the primary microcontroller via UART. All system functions are accessible from a UI implemented through the system computer.

The project will not deal with the actual realization of the product, but rather work with a conceptual preparation and review of the system and its structure. The conceptual system is based on simulations and testing of software on microcontrollers and FPGA, without interaction with hardware modules, focusing on compatibility with C++ and VHDL.

2. Resumé

I dette projekt er der blevet arbejdet med udviklingen af et sikkerhedsudvidende system knyttet til benyttelsen af en X10 protokol. Denne rapport har til formål at anskueliggøre og redegøre anvendelsen af X10 protokollen i samspil med hardware og software moduler. Systemet har til formål at forebygge tyveri ved automatisering af en gradvist styret lampe samt en switch der kan styre eldrevne apparater, så der afbilledes tilstedeværelse af aktivitet i hjemmet.

Til systemet er der udviklet SW og HW-moduler, i rapporten vil være indeholdt dybdegående beskrivelser af softwareprotokoller samt gennemgang af hardware moduler og deres struktur. Sammenspillet mellem hardware og software styres hovedsageligt gennem et sæt mikrokontrollere, der kommunikerer vha. Sender -og modtager-moduler over lysnettet. Herved kan et signal sendes fra systemets computer, der kan styre to eldrevne enheder.

Den første enhed, Switch, giver mulighed for at tænde og slukke for et tilslutte apparat, mens det andet modul, Dimmer, kan variere spændingen der gives. Deres primære funktioner benyttes i sammenvirke med tidsindstillinger, hvorved menneskelig anvendelse af tilsluttede apparater kan efterlignes.

Systemet giver ligeså mulighed for selv at indstille hvornår apparaterne ønskes tændt, samt hvilken lysstyrke lamperne skal lyse med. For at tilgå denne funktion kræves kodeord verificeret gennem tilhørende DE2-Board, der kommunikerer med den primære mikrokontroller via UART. Alle systemets funktioner er tilgængelige fra en UI implementeret gennem systemets computer.

Projektet vil ikke omhandle en aktuel realisering af produktet, men rettere arbejde med en konceptuel fremstilling og gennemgang af systemet, og dets opbygning. Det konceptuelle system er baseret på simuleringer og test af software på mikrokontroller og FPGA uden sammenspil af hardware moduler, med fokus på kompatibilitet med C++ og VHDL.

Indholdsfortegnelse

3. Forord

Følgende rapport omhandler et projektarbejde som er blevet udviklet af projektgruppe 5 bestående af Elektronik- og IKT-studerende på 2. semester på Ingeniørhøjskolen, Aarhus Universitetet. Rapporten er udarbejdet af Elektronikingeniørstuderende: Arne Jan Broeders, Frederik Thomsen og Thomas Stenholt Laursen, og Softwareingeniør-studerende: Andreas Stavning Erslev, Christian Bach Johansen, Mads Nørregaard Morratz samt Shynthavi Prithviraj.

Projektet er blevet vejledt af Gunvor Elisabeth Kirkelund som er lektor på Ingeniørhøjskolen, Aarhus Universitet. Projektet har foregået i perioden 09-02-2020 til og med afleveringsfristen 15-06-2020. Projektet dokumenteres i selve rapporten samt i tilhørende bilag indeholdende filer, hvoraf disse indeholder yderligere dybdegående detaljer.

3.1 Arbejdsfordeling

Ansvarsområde	**Ansvarshavende**	
	Primær	**Sekundær**
Software Arkitektur	Andreas, Christian, Mads, Shynthavi	
SW Design: Arduino_sender	Andreas, Christian	
SW Design: Arduino_receiver_switch	Andreas	Christian
SW Design: Arduino_receiver_dimmer	Christian	Andreas
SW Design: DE2-Board	Mads, Shynthavi	
Hardware Arkitektur	Arne, Thomas, Frederik	
HW Design: Simulink	Thomas, Arne	
HW Design: Zero Crossing	Arne, Thomas	
HW Design: Carrier Generator	Thomas	Arne
HW Design: Carrier Detector	Arne	Thomas
HW Design: Switch/Dimmer	Frederik	

Tabel 1

3.2 Ordforklaring

Forkortelse/Ord	**Betydning**
X10	Protokol for kommunikation mellem elektroniske enheder
ac	Aktør-Kontekst
BDD	Block Definition Diagram
IBD	Internal Block Diagram
uc	Use-case
sd	Sekvens Diagram
cd	Klasse Diagram
SA	Sender-Arduino
MA	Modtager-Arduino
Mikrocontroller/Arduino	Arduino Mega2560 (ATmega2560 chip)
LE	Logic Elements
DE-2 Board	Altera DE-2 Development and Education Board Cyclone II – FPGA EP2C35F672C6N (35000 LEs)
HW	Hardware
SW	Software
UI	User Interface
UML	Unified Modeling Language
SysML	Systems Modeling Language

Tabel 2

4. Indledning

Indbrud er noget som mange mennesker gerne vil undgå. Det er tvivlsomt at nogen mennesker ønsker oplevelsen af et gennemrodet hjem. Personlige materielle goder kan gå tabt, hvor nogle ikke kan erstattes. Desuden kan indbrud være en ubehagelig psykisk oplevelse[1], idet fremmede mennesker har været i ens private bolig. Ifølge en undersøgelse bliver 95 ud af 100 indbrud aldrig opklaret[2]. Fra perioden 2011-2018 blev der begået omkring 555.000 indbrud i de danske hjem, men kun 29.000 af disse politianmeldelser førte til en dom eller bøde[3].

Der findes mange produkter på markedet, det kan f.eks. være *Miraca ST-5*[4] som er en pakkeløsning bestående af bl.a. en indbrudsalarm samt en fjernbetjening til at slukke/tænde for alarmen. Et andet produkt er *Netgears Arlo* kamera[5] som er indbygget med bevægelsescensorer, som giver en notifikation på din telefon hvis der er en tyv i dit hus. Fælles for de to førnævnte produkter er dog, at de først reagerer når uheldet er ude. Det er svært at gøre noget ved indbrud når det først er sket, derfor har projektarbejdet omhandlet forebyggelsen af indbrud.

Home Protection er et produkt der kan skabe en illusion af at man er hjemme, selvom man i virkeligheden ikke er. Produktet gør det muligt at bestemme valgfrie tidspunkter hvorpå lamper, TV samt andre elektroniske apparater skal tændes selvom man ikke er hjemme. Dette betyder at tyven kan afskrækkes når han/hun undersøger mulighederne for indbrud på en bolig og bliver mødt af tændte lamper samt en støj fra fjernsynet[6].

[1] Ref[i1]

[2] Ref[i2]

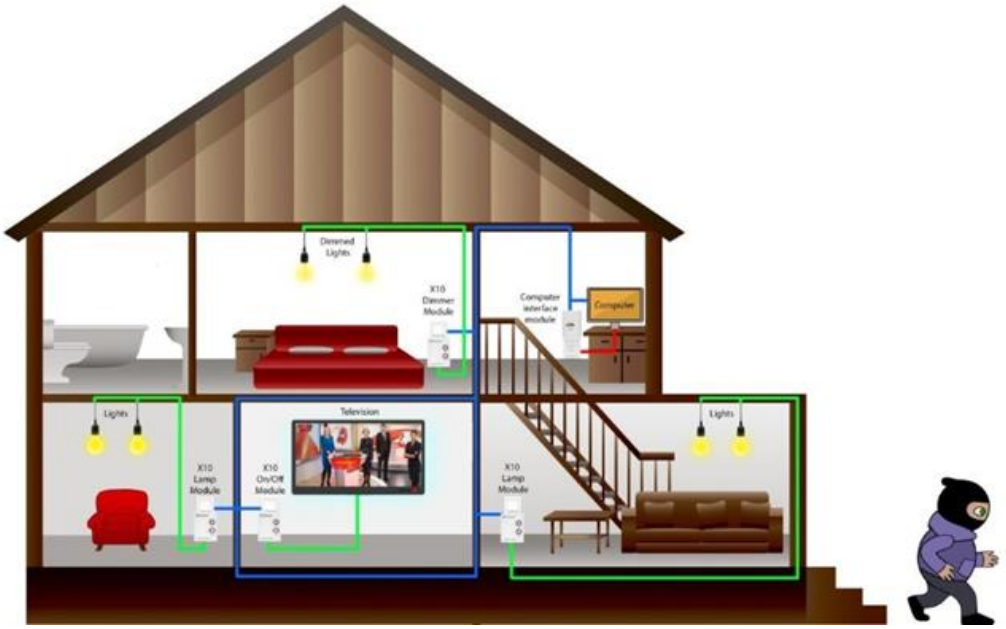
[3] Ref[i2]

[4] Ref[i3]

[5] Ref[i4]

[6] Ref[i5]

4.1 Projektbeskrivelse



Figur 1: Home Protection Illustration

For helt at undgå at tyven kommer i nærheden af privatboligen er produktet *Home Protection* valgt at blive udviklet, en illustration af produktet ses på Figur 1. Det er et indbrudsforebyggende produkt som kan medvirke til, at tyven ikke bryder ind og politiet dermed aldrig behøver at indblandes. Home Protection bygger på 'Powerline Communication' og herunder X10 kommunikation. Hermed muliggøres det at tænde elektroniske apparater i huset på en computerapplikation samt at bestemme tidspunkter hvorpå førnævnte apparater skal tændes/slukkes. Programmet på computeren vil tilbyde 2 forskellige modes.

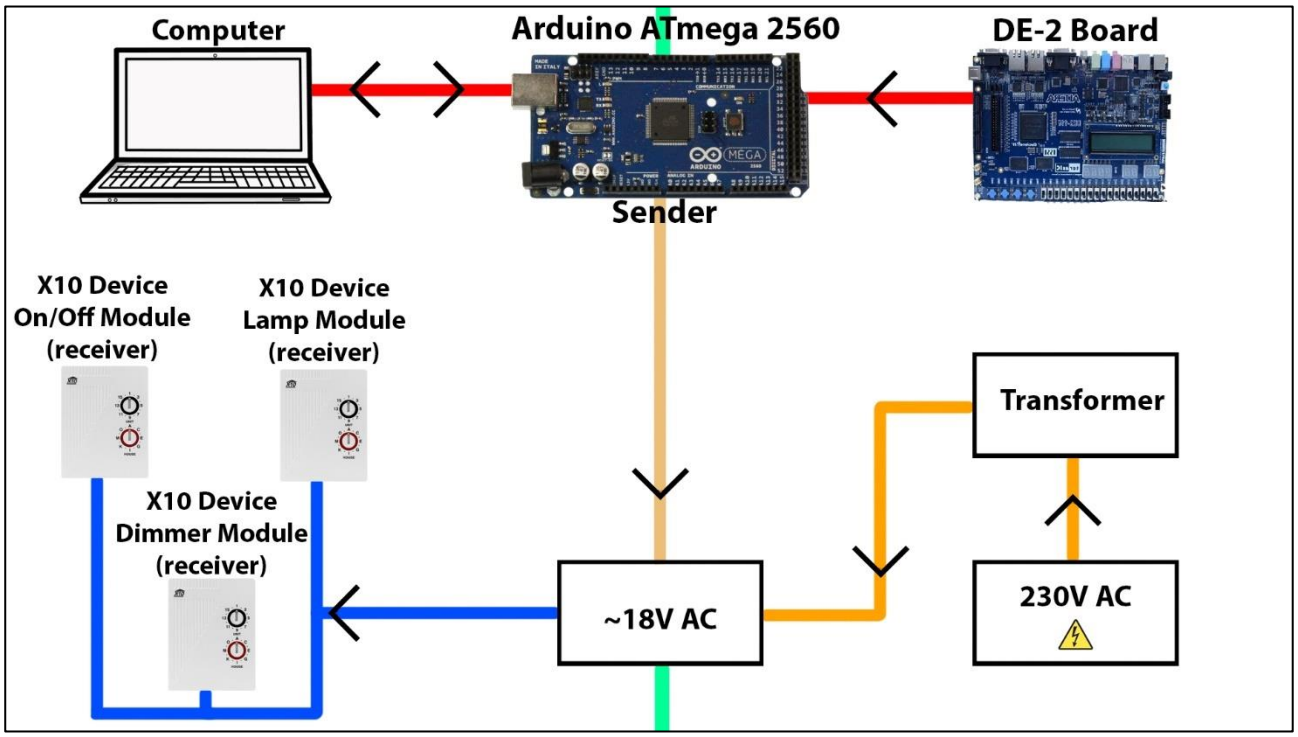
Mode 1 er et automatiseret system der er præ-programmeret til at tænde og slukke lamper og evt. andre tilsluttede X10 moduler i hjemmet til en planlagt tid. Mode 2 er en brugerdefineret mode, det vil sige, at brugeren herved selv kan indstille hvornår de forskellige X10 moduler skal aktiveres og deaktiveres.

Til prototypen af produktet vil der blive forsøgt på at lave 3 fungerende X10 moduler. Et modul der styrer en lampe, som blot kan tænde og slukke lampen. Det andet modul styrer også en lampe, men denne lampe har dog en ekstra funktion, nemlig at den også kan dæmpe lyset. Det tredje modul vil være et tænd/sluk modul. Sidstnævnte modul er tiltænkt at tilsluttes et fjernsyn, en radio eller et helt tredje apparat.

For at sikre at der ikke ændres i de bruger-definerede indstillinger vil der være en tilsluttet en aktiv lås vha. et DE2-board. For at ændre indstillingerne skal brugeren altså først indtaste en given kode på DE2-boardet.

Produktet vil i første omgang ikke have en GUI, eller være tilgængelig fra telefonen gennem en app. Derudover vil produktet heller ikke være indbygget med en sensor til at bestemme hvornår mode 1. skal tænde og slukke, afhængig af lysintensiteten udenfor.

Systemet består af en computer, der ved seriel USB-forbindelse er tilsluttet en X10 sender. Computerapplikationen vil have en tekstbaseret terminal til at styre systemet. Senderen består af en Arduino og et elektrisk system, der er tilkoblet et 18V AC 50Hz elnet. Derudover består systemet af forskellige X10 modtager-moduler. Modtageren afkoder de digitale signaler sendt over lysnettet af senderen. Systemet er skitseret på Figur 2.



Figur 2: System sammensætning

5. Projektafgrænsning

Projektet *'Home Protection'* er blevet lavet under Corona-pandemien i 2020. Dette har gjort at projektet står overfor mange afgrænsninger. Corona-krisen har primært betydet at projektet er blevet tæt på at være rent teoretisk, og derfor har det ikke været muligt at lave nogle reelle hardwareelementer til implementering. Sammenspillet mellem hardware og software er derfor rent konceptuelt. Projektgruppen har valgt at lave praktiske use-cases, der derfor heller ikke er overkomplicerede. Projektet er derfor også blevet udført med fokus på, at alle 3 use-cases i de funktionelle krav bliver udført og konceptuelt implementeret. For de funktionelle krav har gruppen opstillet en række krav, ud fra MoSCoW principperne. Herunder at systemet skal kunne indstilles til to forskellige modes, hvor den ene skal kunne indstilles med brugerdefinerede indstillinger. Systemet burde have tre fuldt funktionelle moduler, en lampe, en lampe med en Dimmer og en Switch der kan tænde og slukke for et vilkårligt eldrevet apparat.
Systemet kunne have en brugervenlig GUI, med fokus på at brugeren nemt kan indstille produktet.
Systemet vil ikke være blevet fuldt testet da det ikke er muligt at bygge et fuldendt produkt, grundet COVID-19. Disse krav er blevet stillet da det er de minimale krav der opstilles før produktet, kan blive anerkendt som en prototype.

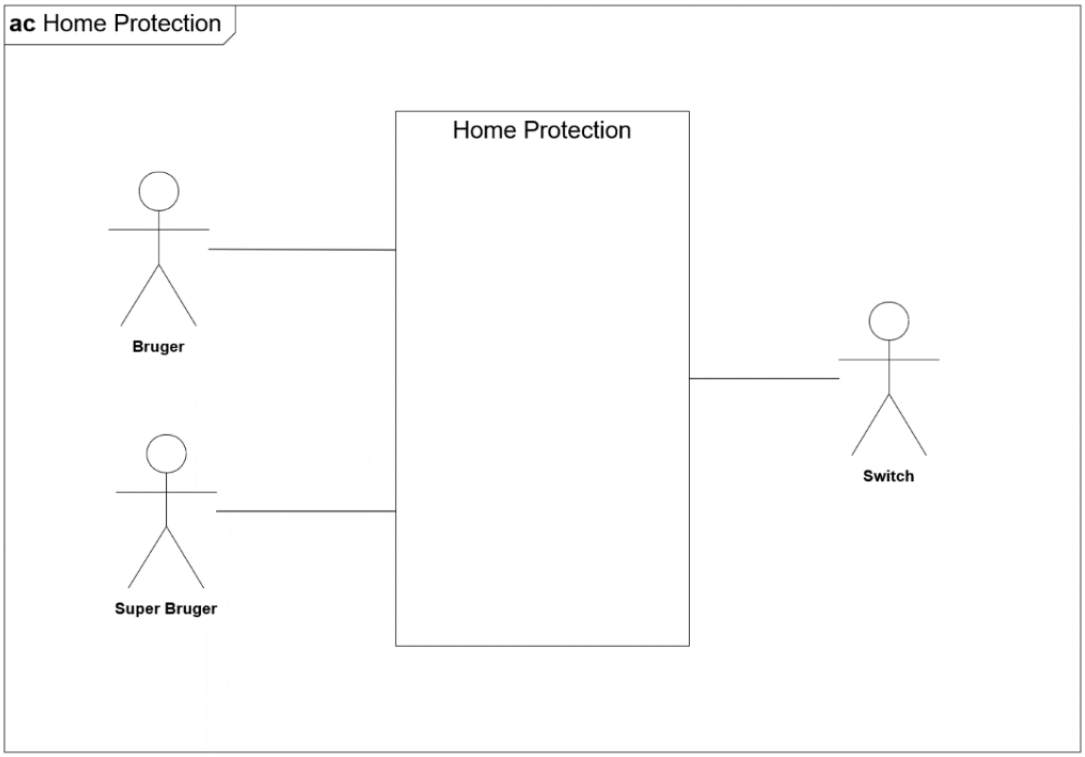
6. Kravsspecifikation

Ud fra emnet 'Home Automation' og en analyse af projektafgrænsningen er der blevet lavet funktionelle krav til produktet 'Home Protection'. Nedenstående afsnit forklarer det samlede system i individuelle detaljer, herunder er der benyttet værktøjer fra kurset *Indledende System Engineering*.

6.1 Aktør Beskrivelse

Systemet er beskrevet ved hjælp af et aktør-kontekst-diagram, dette ses på Figur 3. I diagrammet ses *Bruger*, *Super Bruger* og *Switch*. Bruger, som er en primær aktør, har mulighed for at starte og slukke systemet. Bruger vil kunne benytte en computer med programmet for *'Home Protection'*, hvor mulighederne for start og sluk af et prædefineret- eller brugerdefineret- program er tilgængelige. Super Bruger, som også er en primær aktør, har samme muligheder som Bruger samt udvidede muligheder. Super Bruger besidder en kode der gør at det er muligt at indstille og ændre det førnævnte brugerdefinerede program. *Switch* er en sekundær aktør. Dette er en X10 modtagerenhed i 'Home Protection' systemet og er i stand til at tænde eller slukke en enhed.

De vigtigste funktionelle krav for produktet 'Home Protection' er at kunne aktivere de forskellige 'modes' i systemet. Opfyldes disse krav ikke vil det ikke være muligt at automatisere systemet og dermed vil den ønskede funktionalitet ikke opnås, nemlig at opnå beskyttelse imod indtrængen af ubudne gæster. De forskellige use cases ses i nedenstående UC Diagram, Figur 4. Ønskes yderligere detaljer om diagrammet samt en detaljeret gennemgang af hver enkelt use case, henvises der til kravsspecifikationsdokumentet: Kravsspecifikation og Accepttest, afsnit KA1.4[7].



Figur 3: Aktør-kontekst Diagram

[7] Ref[K1]

Herunder følger en kort beskrivelse af brugeren.

Navn:	Bruger
Alternativ reference:	Husejer
Type:	Primær
Beskrivelse	Brugeren kan interagere med systemet igennem computerens terminal. Brugeren kan vælge et forud lavet program eller brugerdefineret program.

Herunder følger en kort beskrivelse af Super Brugeren.

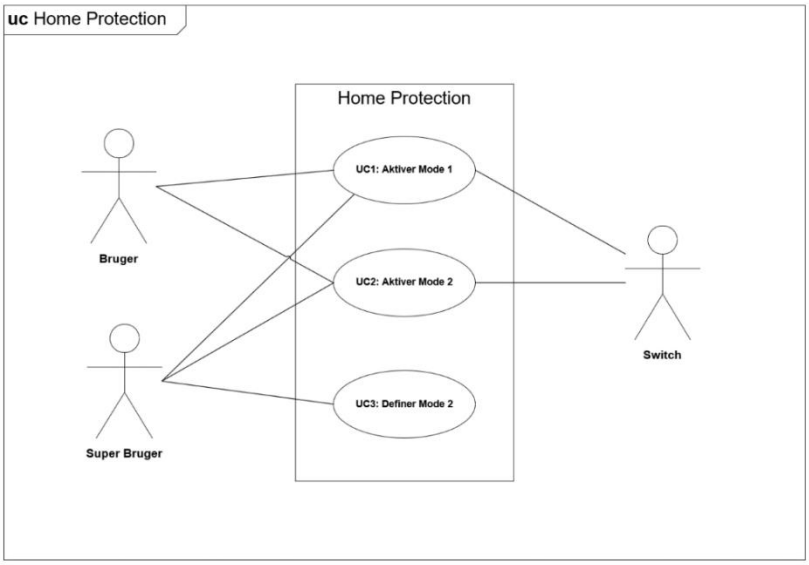
Navn:	Super Bruger
Alternativ reference:	Administrator
Type:	Primær
Beskrivelse	Super bruger kan interagere med systemet igennem computerens terminal. Super Bruger kan vælge et forud lavet program eller et brugerdefineret program. Super bruger kan også indstille det brugerdefineret program.

Herunder følger en kort beskrivelse af Switch'en.

Navn:	Switch
Alternativ reference:	
Type:	Sekundær
Beskrivelse	Er i stand til at tænde og slukke forskellige elektroniske apparater i huset. (En lampe eller et TV)

6.1.1 Use Case Diagram

På Figur 4 ses Use Case Diagrammet for home simulation systemet. Der er tre Use Case's. Brugeren kan initiere enten Mode 1 eller Mode 2.



Figur 4: Usecase Diagram

6.1.2 Use Case Beskrivelse

UC1 Aktiver Mode 1:

Brugeren aktiverer mode 1. Herefter sender X10 senderen programmet i Mode 1, der vil tænde og slukke for forskellige X10 modtager moduler. Mode 1 er en prædefineret mode som i dette tilfælde er en demo-mode. Dette vil sige at der er en præprogrammeret default mode. I mode 1 vil lampe og switch tændes når mode 1 aktiveres og slukkes efter 1 time.

UC2 Aktiver Mode 2:

Brugeren aktiverer mode 2. Herefter sender X10 senderen det bruger definerede program i mode 2, der vil tænde og slukke for forskellige X10, modtager moduler. Mode 2 kan enten være indstillet af Super Brugeren fra tidligere, og hvis dette ikke er tilfældet, vil en aktivering af mode 2 resultere i en aktivering af default Mode 1.

UC3 Definer Mode 2:

Super Brugeren vælger 'Definer Mode 2', herefter bliver Super Brugeren bedt om at indtaste en kode. Når den korrekte kode er indtastet, kan mode 2 defineres. Her kan brugeren selv vælge hvilke X10 moduler der skal tænde og slukke, samt hvornår de skal tænde og slukke.

6.2 Ikke funktionelle krav

Systemet '*Home Protection'* skal leve op til nogle krav som der er blevet stillet til projektet. Disse krav er lavet ved hjælp af FURPS+, i tabel 3 har er der blevet lavet en tabel over de ikke funktionelle krav som der stilles til systemet samt deres prioritering vha. MoSCoW metoden. Hvert krav har et serienummer

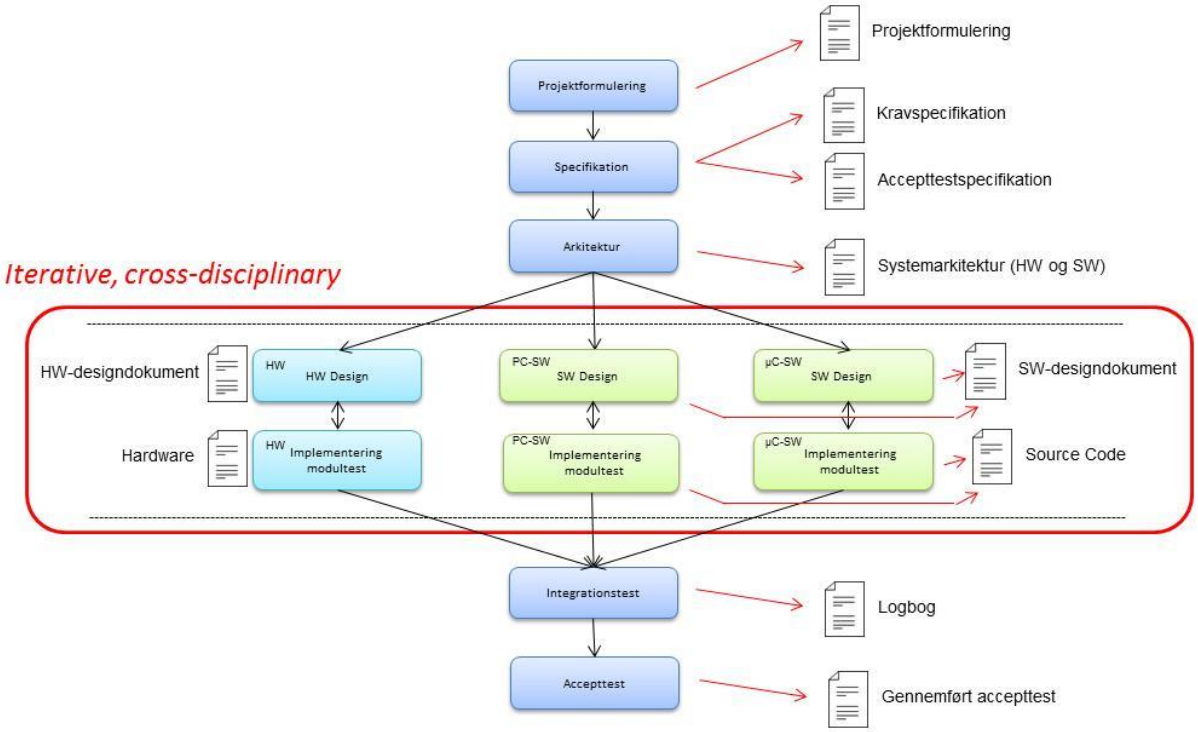
som der kan henvises til, i tabel 3 kan der ses under hvilket FURPS+ krav, kravet er stillet samt dets prioritering.

Serienummer	Ikke-funktionelt krav	FURPS+	MoSCoW
REQ 2.1U	Skal kunne låse administrative indstillinger op efter 1 korrekt indtastning af kodeord	Usability	Must
REQ 2.2U	Skal kunne spærre adgang i 5 minutter til administrative indstillinger efter kodeord er tastet forkert 3 gange i streg	Usability	Must
REQ 2.3U	Skal kunne bevare kodeord indtil ønsket udskiftning	Usability	Must
REQ 2.4U	Kunne være muligt at ændre koden på DE2-Boardet	Usability	Could
REQ 2.5U	Kunne have en GUI.	Usability	Could
REQ 2.5U	Vil ikke have adgang til internettet	Usability	Will not
REQ 2.1R	Bør have en driftssikkerhed på 99% (± 0.5%)	Reliability	Should
REQ 2.2R	Bør kunne låse op for administrative indstillinger 99% (±0.5%) af gangene hvor kodeord indtastet korrekt	Reliability	Should
REQ 2.3R	Bør gå 400 timer mellem fejl i samlet system (± 1 time)	Reliability	Should
REQ 2.4R	Vil ikke kunne have flere end 1 mode aktivt	Reliability	Will not
REQ 2.1P	Bør tage mindre end 10ms at sende signaler gennem el-nettet (± 0.1 ms)	Performance	Should
REQ 2.2P	X10-enheden + HW skal have dimensionerne 10cmx20cmx30cm(±2cm)	Performance	Must
REQ 2.3P	Det samlede system må maksimalt veje 10kg(±1kg)	Performance	Must
REQ 2.4P	SW responstid på input skal maksimalt være på 1 sekund (± 0.1 s)	Performance	Should
REQ 2.5P	Kunne have mere end 2 modes	Performance	Could
REQ 2.6P	Kunne fungere på flere platforme	Performance	Could
REQ 2.7P	Kunne have flere eller mindre end 3 moduller	Performance	Could
REQ 2.1S	Kunne være muligt at tilføje flere enheder	Supportability	Could
REQ 2.2S	Bør kunne tilsluttes 99% (± 0.5 %) af el-net med type 18, 50Hz	Supportability	Should
REQ 2.3S	Mode 2 vil ikke kunne værre indstillet på mere end 1 måde ad gangen	Supportability	Will not

Tabel 3

7. Metode

For at lave dette projekt, har gruppen brugt ASE modellen (se Figur 5). ASE-modellen er en semi-iterativ udviklingsmodel, som er drevet af use cases. Use cases bliver brugt til at beskrive hvordan diverse aktører bruger produktet.



Figur 5: ASE modellen illustreret[8]

Gennem projektets forskellige faser er der blevet lavet forskellige UML og SysML diagrammer. SysML er blevet brugt til at lave detaljerede HW og SW-beskrivelser. UML-diagrammerne der er blevet lavet er bla. klassediagrammer, der giver et overblik over SW-klasserne.

Under specifikationsfasen er der blevet lavet forskellige SysML diagrammer, herunder Aktør-kontekst diagrammet og Use case diagrammet. Til at beskrive funktionaliteten af HW benyttes BDD'er og IBD'er. Til at beskrive SW funktionaliteten benyttes SD'er, STM og CD'er.

Yderligere forklaringer på metoderne brugt, kan findes i Process Rapporten[9]

[8] Ref[G1]
[9] Ref[P1]

7.1 Udviklingsværktøj

Projektets udviklingsproces har taget udgangspunkt i ASE-modellen, også kaldt Semesterprojektmodellen. Projektbeskrivelsen blev udarbejdet som det første, efterfulgt af kravsspecifikation, accepttestspecifikation, systemarkitektur og design. Under design delen af projektet blev der arbejdet iterativt med hardware og software. Der blev altså lavet forbedringer løbende, indtil det ønskede mål blev opnået. I systemarkitekturen- og design-fasen er der blevet anvendt SysML. Grundet corona-udbruddet var der ikke mulighed for at opbygge kredsløbet, da laboratoriet var midlertidig lukket og der dermed ikke var adgang til hardware-komponenterne. Kredsløbet blev derfor bygget i Simulink og Multisim. For at opbygge kredsløbet, blev der lavet BDD og IBD, hvilket giver et overblik over de enkelte dele systemet bestod af. I softwaredelen er der blevet anvendt klasse- og sekvensdiagrammer, der er med til at beskrive de forskellige Use Cases.

Igennem forløbet er der blevet holdt 2 reviews, hvor grupperne har indsendt deres materiale og kommenterer på hinandens arbejde. Det første møde blev afholdt oppe på universitet, mens det andet møde blev holdt over Zoom. For at aftale møderne sendte repræsentanter fra hver gruppe mails til hinanden. Det nye indblik for de afholdte review møder hjalp med at opdage fejl og mangler i materialet. Her kunne der tages inspiration fra hinandens materiale og opbygninger. Det var også med til at skabe deadlines til tidsplanen, eftersom gruppen skal være klar til at kunne fremvise noget til hver review møde. Møderne gav et indblik i hvordan de andre grupper har tacklet nogle af deres problemer og gav inspiration til nogle rettelser og tilføjelser.

7.2 Proces

7.2.1 Gruppesammensætning

Projektgruppen består af 4 IKT-studerende og 3 E-studerende. Det var ikke alle der havde kendskab til hinanden, og der blev derfor lavet en introduktionsrunde den første dag med navn og Insights farver. Generelt forekom farverne blå og grøn oftere, mens der var mangel på den rødlige farve. På trods af at personer med rød farveenergi er de målbevidste og handlekraftige, og kan være vigtige i en projektgruppe, kunne dette ikke mærkes, da flere i gruppen kunne træde til og styre gruppen, hvis det var nødvendigt. Heraf blev der også diskuteret om hvilke kompetencer de forskellige gruppemedlemmer havde. I fællesskab blev der herefter udarbejdet en samarbejdsaftale[10], vedlagt i bilag, om fremmøde, arbejdsindsats, ambitionsniveau og sanktioner.

7.2.2 Tidsplan

Gennem projektforløbet er der blevet lavet en tidsplan der løbende er blevet opdateret efter hver uge. Her er der blevet indsat forskellige tidsfrister hvor gruppen skulle være færdig med eks. Arkitektur- og design-fasen, gruppen har målrettet arbejdet på at nå disse mål, der dog også ændrede sig lidt grundet COVID-19.

For at se tidsplanen henvises der til bilaget tidsplan[11] i projektmappen

[10] Ref[S1]
[11] Ref[T1]

7.2.3 Arbejdsfordelingen

Arbejdsfordelingen af projektet blev diskuteret som noget af det første i arbejdsprocessen. Gruppemedlemmer fik mulighed for i grove træk at vælge sig ind på hvilket aspekt af projektet de helst ville fokusere på. Forståeligt havde gruppemedlemmerne fra IKT-studieretningen et generelt ønske om at arbejde med software, og ligeså med E-studieretningen og hardware.

Dette var også til fordel for gruppen, idet de to studieretningers skemaer ikke overlappede, denne opdeling gav derfor bedre mulighed for at arbejde på skolen.

Der blev afholdt gruppemøde hver gang efter projektarbejdet, hvor hardware -og softwareafdelingerne opdaterede hinanden ift. hvordan arbejdet havde skredet frem, om der var opstået problemer under dagens arbejde, hvilke planer der var for arbejdet til næste arbejdsgang. Til hvert gruppemøde er knyttet en logbog[12], der kan findes under bilag.

I retrospekt viste opdelingen af projektarbejdet sig specielt fordelagtigt. Pga. Corona-epidemien, og efterfølgende karantæne, blev gruppearbejdet tvunget online. I normalt gruppearbejde er det sundt både at have arbejdet både en smule med HW og SW af systemet. Dette blev dog en stor udfordring, da det er langt svære at holde overblik over de forskellige dele, specielt mellem HW og SW, af systemet når den eneste kontakt foregår gennem skærmdeling, og man ikke blot kan gå om på den anden side af arbejdsbordet.

At gruppen blev delt op i mindre del-grupper der enten kun fokuserede på software eller hardware viste sig at hjælpe arbejdsprocessen fremad. Med rimelige intervaller blev der så foretaget review mellem del-grupperne, for at sikre at HW og SW-design stemte overens.

7.2.4 Online projekt

For at holde styr på aftaler og møder blev der brugt en gruppe på Facebook til at påminde gruppemedlemmerne, hvornår der var møder. Dette gjorde det nemmere for gruppemedlemmerne at informere omkring sygdom og eventuelle andre mangler og problemer. Trello.com er en hjemmeside, hvor der kan oprettes projekter, hvor arbejdsopgaver kan uddelegeres samt udforme en tidsplan. Trello er blevet benyttet til at holde styr på de forskellige arbejdsopgaver som de forskellige gruppemedlemmer har, samt hvor langt de er nået.

Til at holde møderne er der blevet brugt to programmer, henholdsvis Discord og Zoom. Discord er et gratis program, som tillader op til 50 personer at snakke sammen og dele deres skærm. Zoom er et program givet af skolen, som blev primært brugt til at holde vejledermøder. Til at opbevare dokumenter, billeder og andre filer, blev Microsoft OneDrive brugt.

Onedrive er et program som kobler sig sammen med alle Microsoft programmer, som gør det muligt for adskillige personer at skrive i samme dokument. Det skaber også et online drev, som alle gruppemedlemmer har adgang til. I dette drev kan filer gemmes og deles, hvilket giver nemmere adgang, så længe man har internet tilgængeligt.

[12] Ref[L1]

8. Systemarkitektur

I nedenstående afsnit vil arkitekturen for systemet blive beskrevet. Arkitekturen er udviklet i nogle af de første stadier af projekt. Formålet er at skabe et overblik over hele systemet, hvad det skal indeholde og hvordan det hænger sammen. Der laves altså en beskrivelse, af systemet og dets enkelte moduler og deres funktion samt kommunikation. Dette overblik gør projektet mere håndgribeligt og simplificere designdelen af projektet. I systemarkitekturen er der benyttet systembeskrivelser med SysML- og UML-diagrammer.

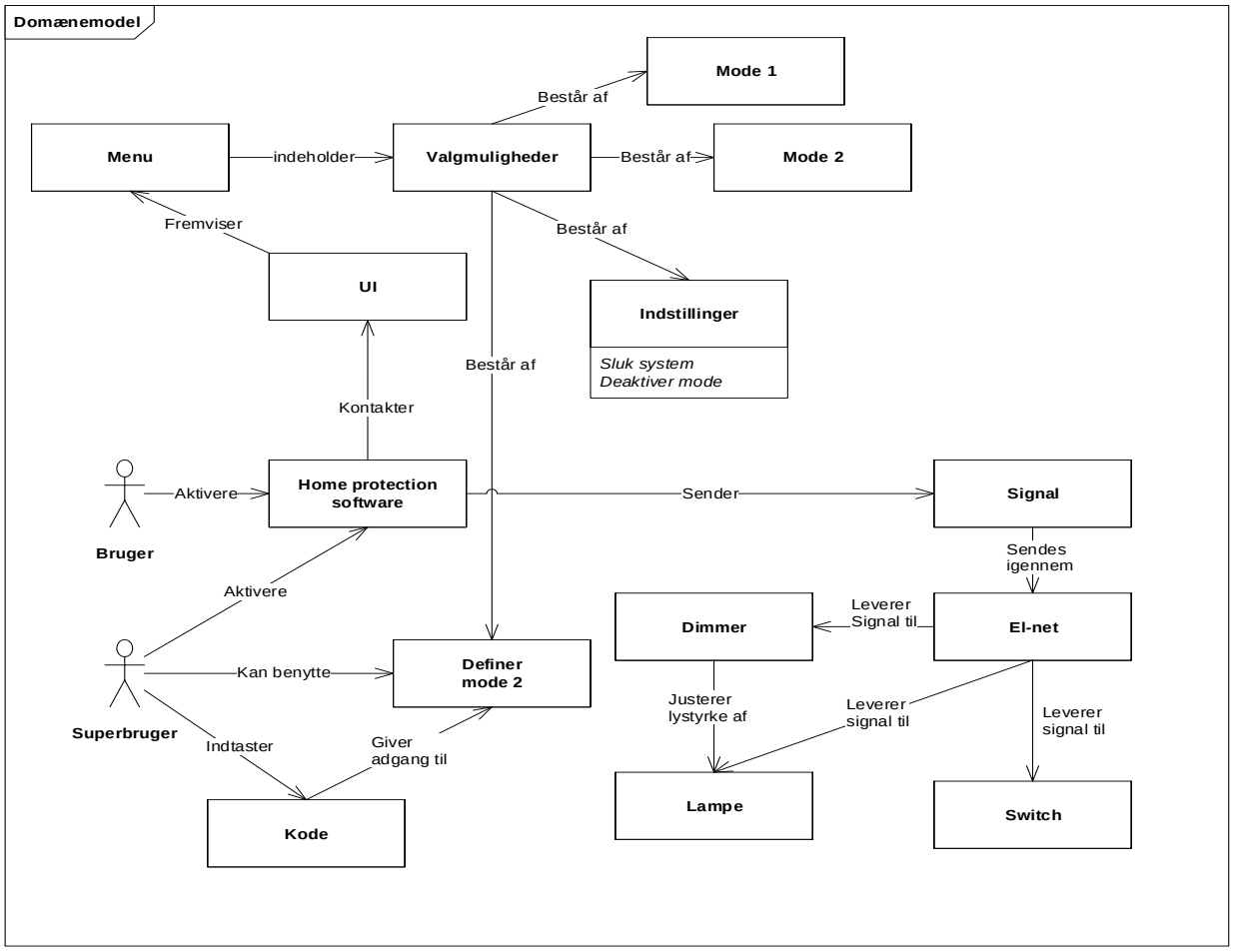
8.1 Software

I følgende afsnit vil softwarearkitekturen for systemet blive beskrevet. Afsnittet indledes med en domænemodel efterfulgt af klassediagrammer og sekvensdiagrammer, der opbygges ud fra Use Cases, hvilket kan findes i dokumentet Kravspecifikation[13].

Domænemodellen viser systemets grundlæggende sammenhæng. Programmet "Home Protection software" kan aktiveres af de to aktører, Bruger og Super Bruger. Programmet har så mulighed for at fremvise en UI, eller sende et signal, ud fra inputs i UI'en. Her kan der vælges "Mode 1", "Mode 2" eller "Definer Mode 2". Det er kun Super Bruger der har adgang til 'Definer Mode 2', idet kun Super Bruger er i besiddelse af adgangskoden. Begge brugere har desuden adgang til indstillinger, som tillader aktøren enten at stoppe en Mode der er aktiv, eller at lukke selve programmet.

Signalet der sendes af Home Protection modtages af X.10 senderen, der kommunikerer med X.10 modtagerne "Dimmer" og "Switch" gennem el-nettet. De ønskede indstillinger aktiveres så ud fra det modtagne signal.

[13] Ref[K1]



Figur 6: Domæne model Home Protection

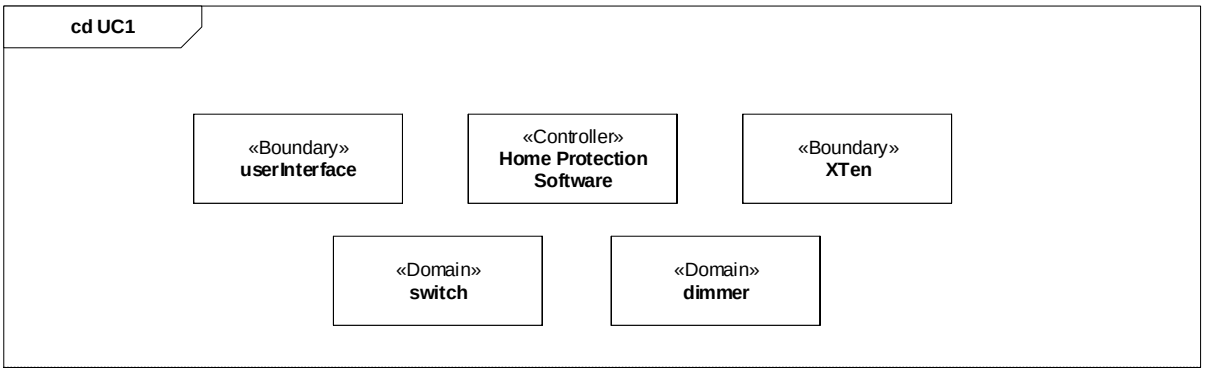
8.1.1 Applikationsmodel

Den overordnede software arkitektur for Home Protection systemet gennem sekvensdiagrammer og klassediagrammer vil blive uddybet. Softwareblokkene angiver her forventede softwareklasser og funktioner i.e. vil ikke være repræsentative for den endelige kode. Der henvises til Kap. 9 for indsigt i den implementerede kode.

Use Case 1 og 2:

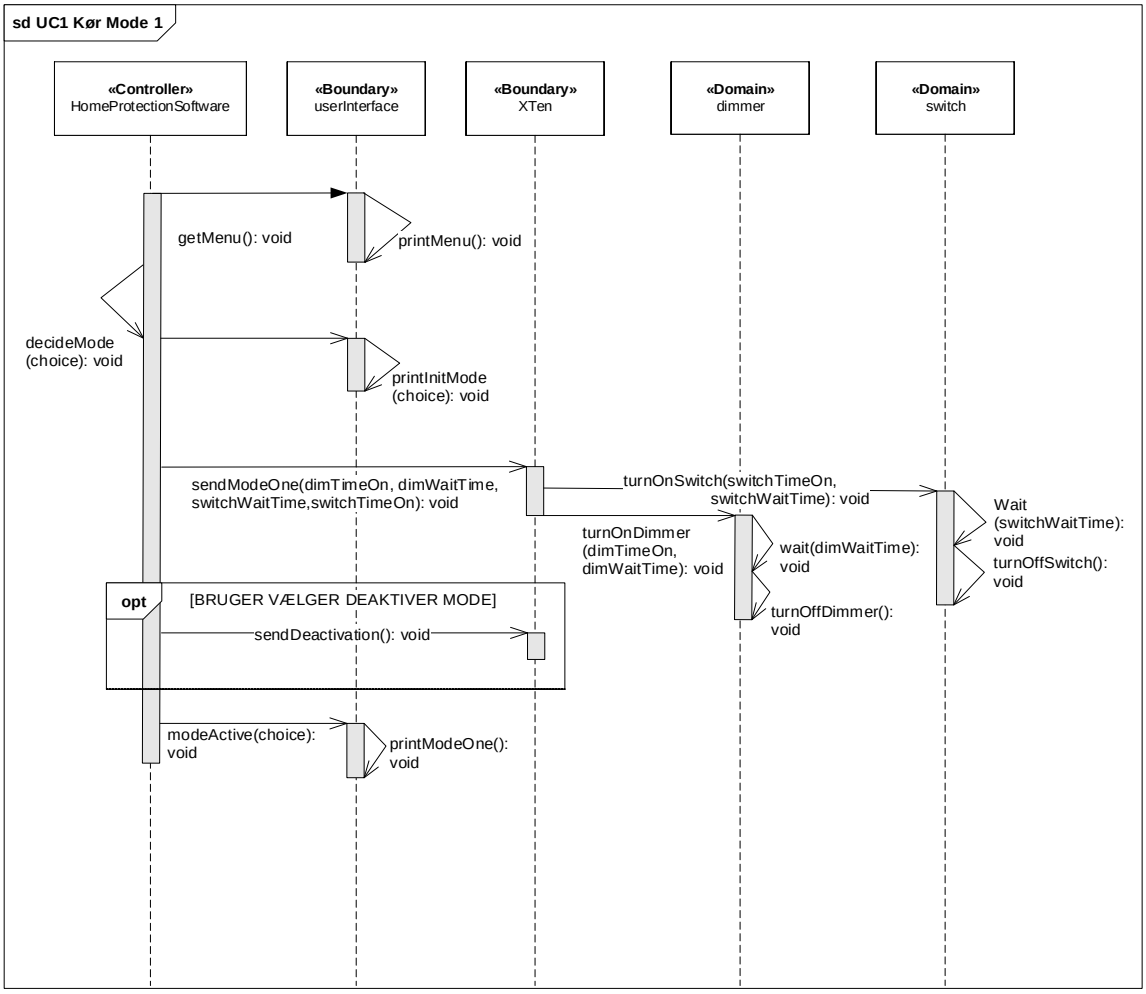
Ved Use Case 1 ønskes der at vælge ”Mode 1”, der skal kunne bestemme hvornår Dimmer og Switch skal tændes og slukkes. Mode 1 har prædefineret indstillinger.

Som det ses på Figur 7 består Use Case 1 af en controller klasse (Home Protection Service), 2 boundary klasse (userInterface, XTen) og 2 domæne klasser (Switch, Dimmer).



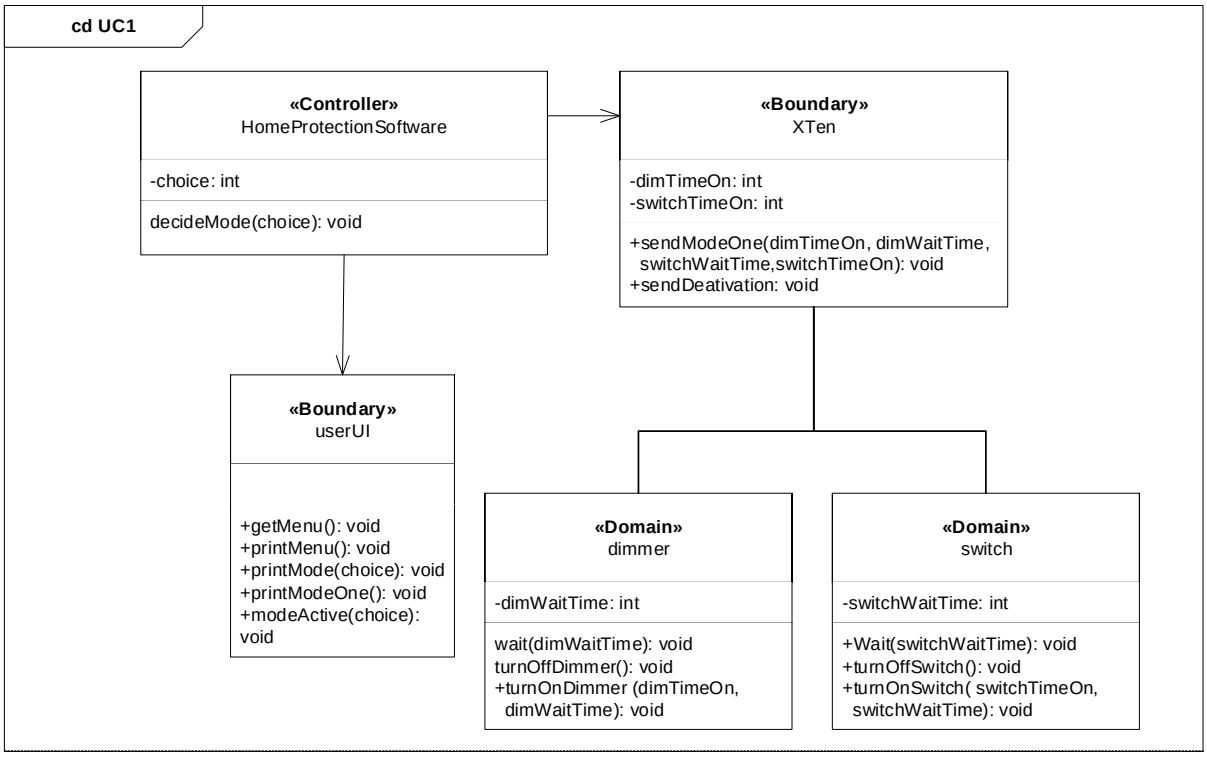
Figur 7 – Klasse diagram UC1

På Figur 8, ses det, hvorledes processen for Use Case 1 udfolder sig. Systemet viser først en menu på UI'en, hvor der herefter skal tages et valg fra brugeren, hvor der gøres brug af en "switch" sætning. Systemet fortæller brugeren at systemet initialiseres og videregiver tænd og sluk tidspunkt videre til XTen, der indstiller Dimmer og Switch. Hvis brugeren bruger menu muligheden "Deaktiver Mode", vil systemet deaktivere Dimmer og Switch, altså, ikke give dem noget tænd og sluk tidspunkt. Til sidst, vil systemet udskrive en succes besked om aktivering eller deaktivering ved UI'en.



Figur 8 - Sekvensdiagram Use Case 1

På Figur 9 ses et udvidet CD. SD forbinder diverse klasser med funktioner. Hertil udnyttes der denne viden, til at opsætte de individuelle funktioner, til den klasse der indeholder den. Yderligere ses det, hvor de diverse variabler holder til.



Figur 9 – Klassediagram for UC1 udvidet

Use case 2 er, på papiret, identisk med Use Case 1, derfor slås de to sammen. Forskellen ligger i, at mode 2, fra Use Case 2, kan modificeres af Use Case 3. Dog ses det ikke i udførslen af Use Casen, da der ikke ændres på mode 2. Mode 2 har desuden de samme værdier som Mode 1 som default.

Software arkitektur for Use Case 3 kan findes i software Dokumentation, afsnit SWA1[14] i projektmappen.

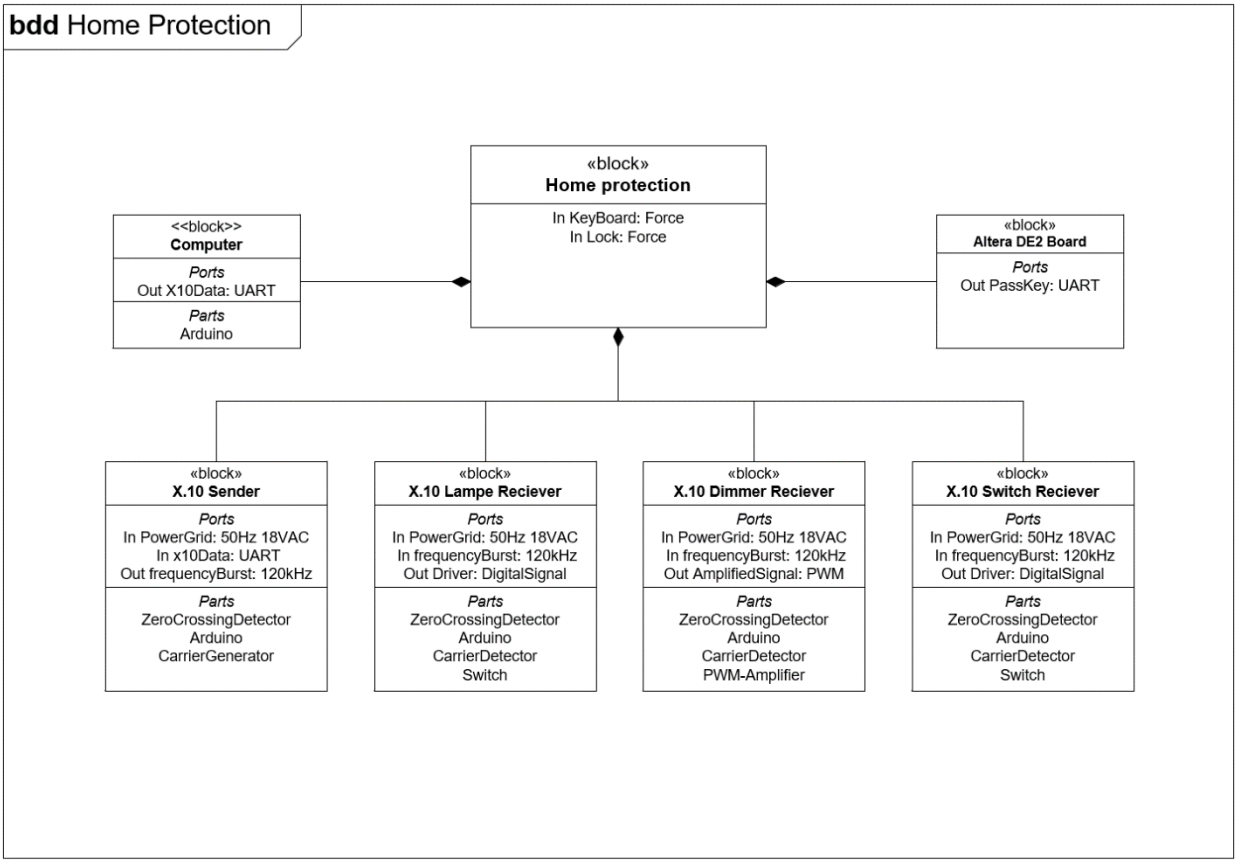
[14] Ref[D1]

8.2 Hardware

Projektet bygger på konceptet "Power Line Communication" og hermed principperne for 'Home Automation' hvor kommunikationsteknikken X10 benyttes. Denne teknologi muliggør en kommunikation, over et lukket 18V AC 50Hz net mellem elektroniske enheder. Som en del af den indledende litteratur til projektet er applikationsnoten (AN236[15]) blevet udleveret og dernæst studeret af projektdeltagerne. Fra litteraturen er der blevet erfaret nogle nødvendige enheder for at få systemet til at fungere. Disse enheder består blandt andet af en Carrier Generator, Carrier Detektor samt en Zero Crossing Detector.

Efterfølgende, er der lavet egen research online omkring de forskellige enheder og hvordan de opbygges. Her er der fundet information omkring emner som Zero Crossing[16], filtrer[17], Envelope Detection[18] . Hermed er der opnået yderligere viden om hvordan systemet skal kommunikere og hvad de forskellige blokke er ansvarlige for. Derudover, er der blevet stillet krav om at der skal være en enkelt computer samt et DE2-board som en fungerende kodelås. Med den indsamlede viden har det været muligt at udvikle et BDD, se Figur 10.

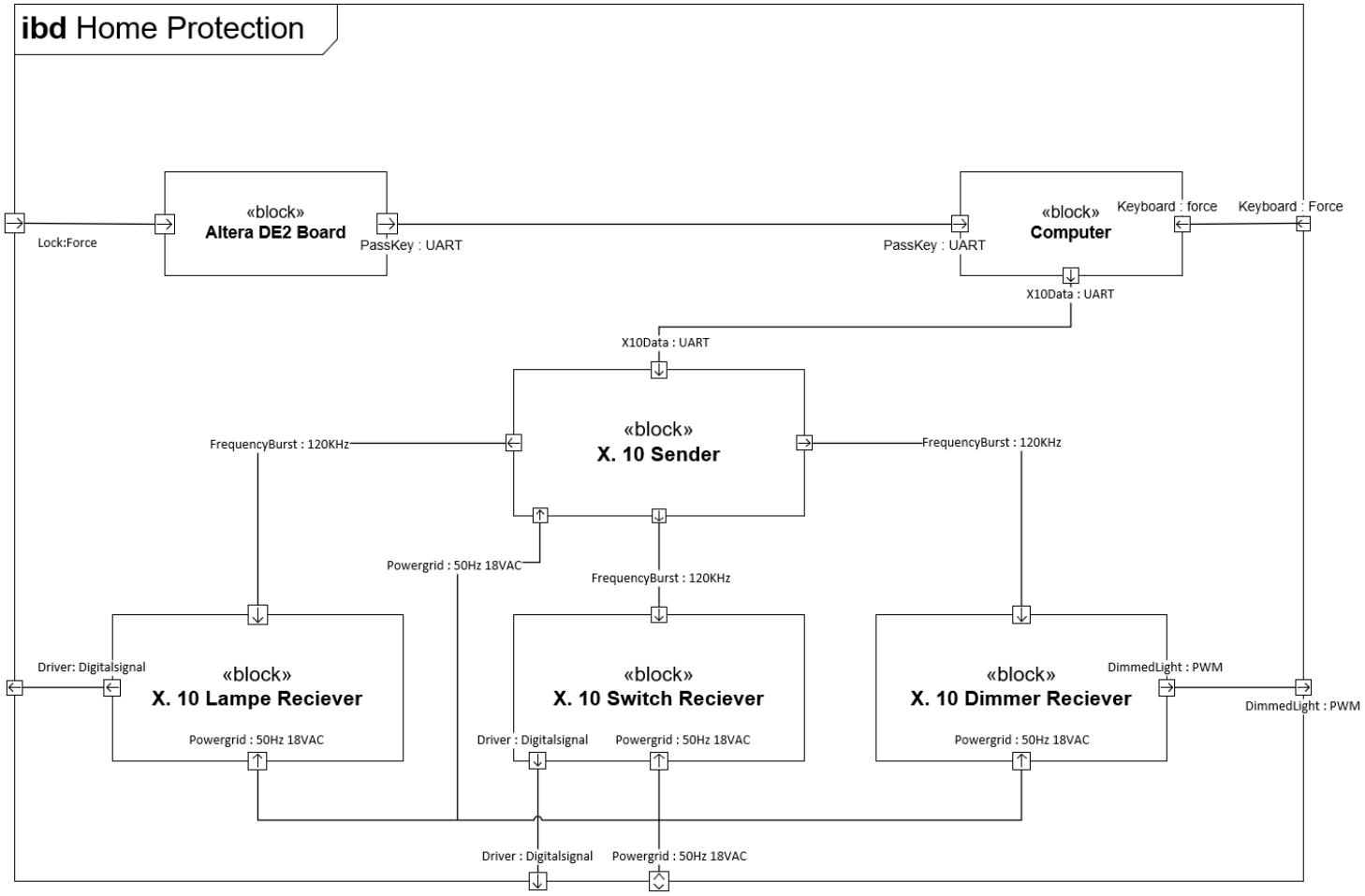
Det udviklede BDD viser både *parts* og *ports. Parts* henviser til de dele som systemet består af, f.eks. en Carrier Generator, Carrier Detektor og en Zero Crossing Detector. *Ports* henviser til de forskellige indgange og udgange som systemet har. De enkelte blokkes interne relationer bliver yderligere formidlet i et IBD (se Figur 12) som er udviklet ud fra nedenstående BDD.



Figur 10: Home Protection BDD

[15] Ref [G2]
[16] Ref [G3]
[17] Ref [G4]
[18] Ref [G5]

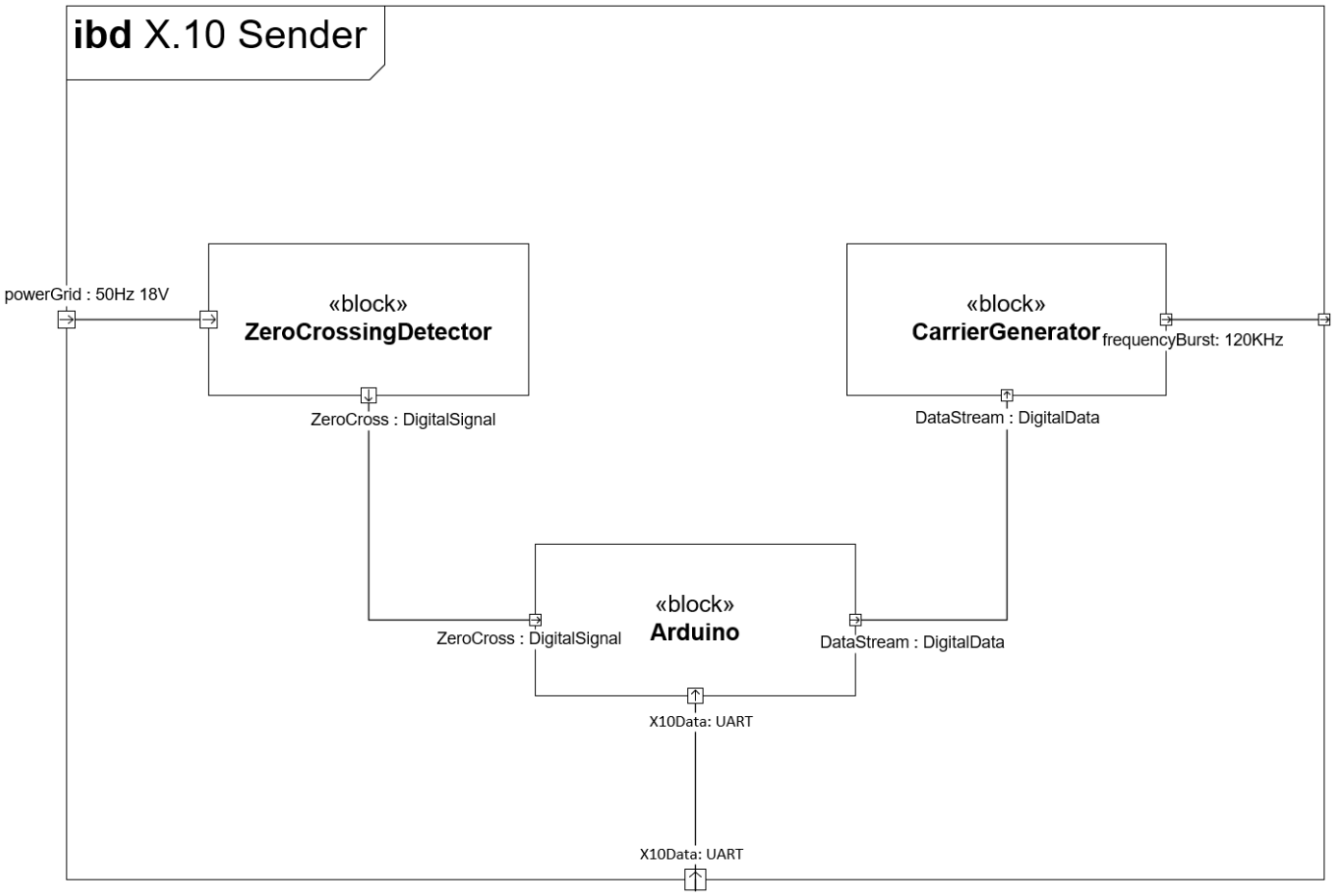
Det overordnede IBD over systemet ses på Figur 12. Her illustreres de enkelte blokkes interne kommunikation og dermed deres sammenhæng. Der skal specielt lægges mærke til at Powergrid og FrequencyBurst er blevet valgt som to separate signaler. I realiteten vil begge signalerne gå på lysnettet og dermed på samme signal, men der blevet valgt at adskille dem. Dette er blevet gjort for overskueligheden, og dermed, gøre det klart hvilke dele af systemet der tager imod og sender de to forskellige former for signaler.



Figur 12: Home Protection IBD

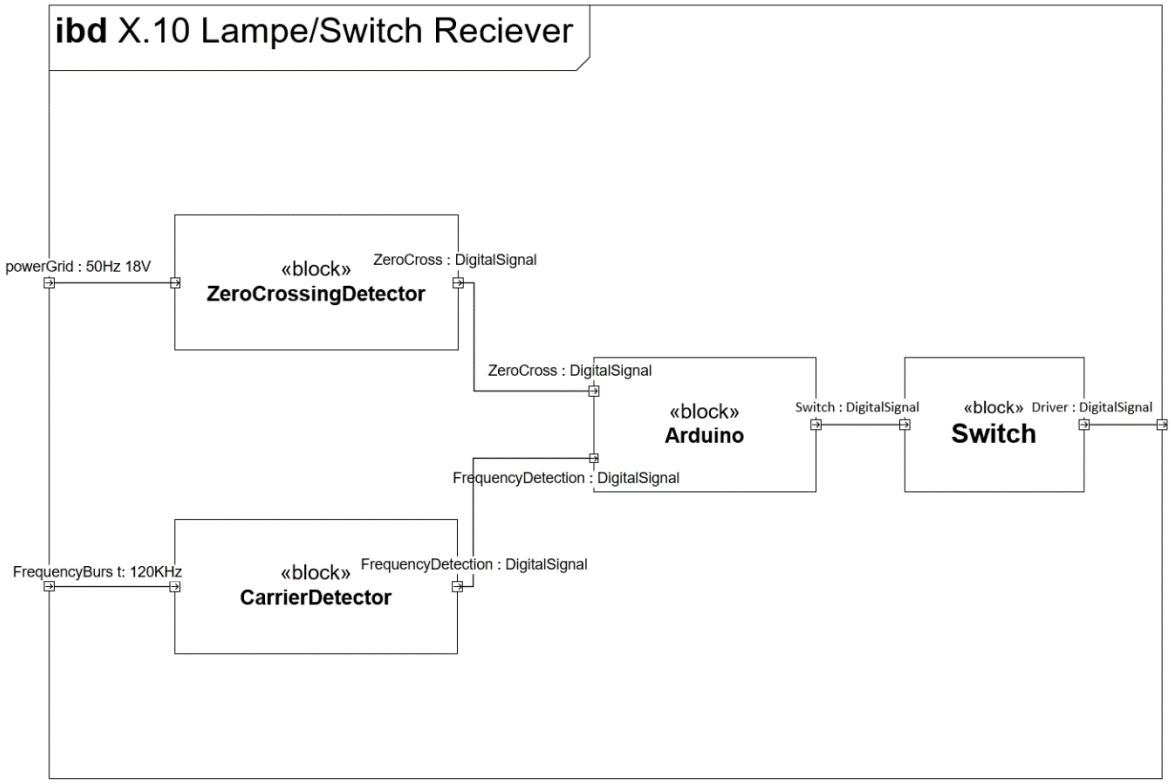
8.2.1 Intern IBD

I de interne IBD'er for X.10 Sender (Figur 13), X. 10 Lampe/Switch Reciever(Figur 14) og X.10 Dimmer Reciever(Figur 15) kan der ses hvilke signaler der går ind på hvilke parts. De nedenstående figurer viser de interne IBD'er for de forskellige X.10 blokke, deres parts og deres interne kommunikation, her er det vigtigt at lægge mærke til powerGrid og frequencyBurst for at se hvordan de forskellige blokke bruger de to signaler til at kommunikere. For yderligere information om de forskellige blokke samt deres signaler, se Block og Signal Beskrivelse i projektmappen.[19]

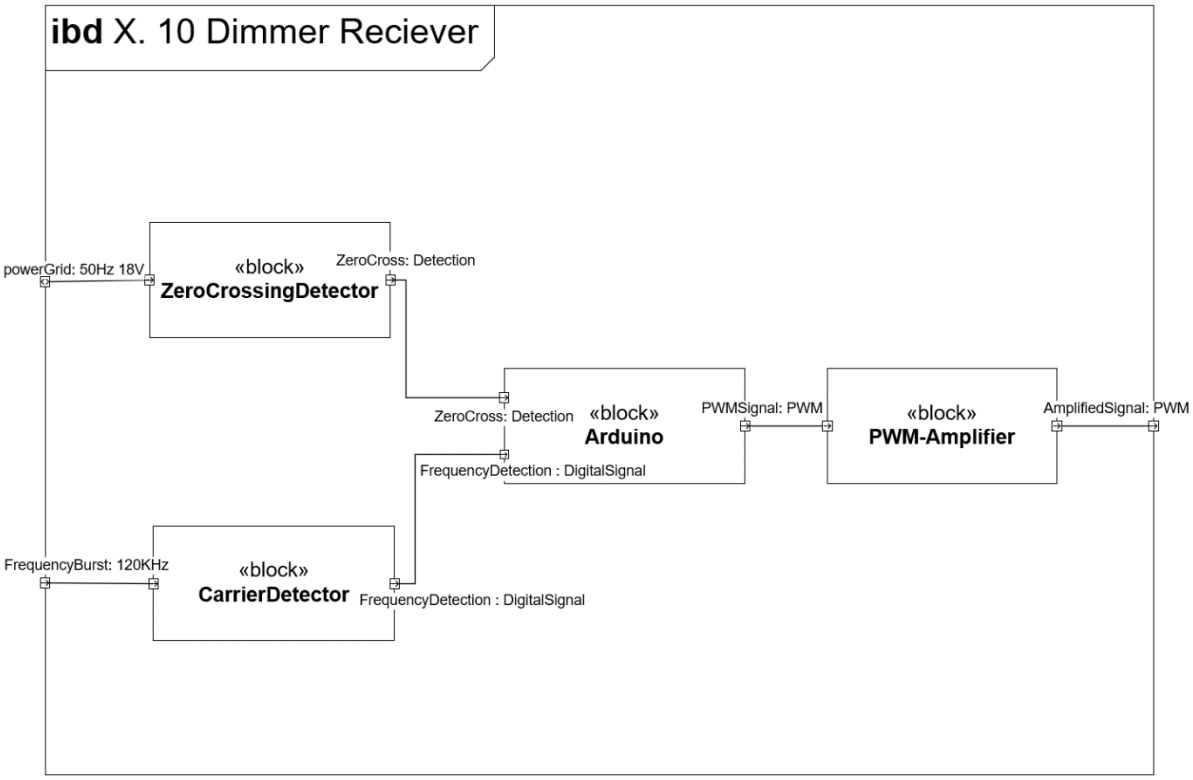


Figur 13: IBD sender modul

[19] Ref[BS1]



Figur 14: Lampe/Switch modtager modul



Figur 15: Dimmer modtager modul

9. Design

9.1 Software

I dette afsnit vil sofware designet af Home Protection blive uddybet, med beskrivelser af klasser tilgængelige under Software Dokumentation, afsnit SWA2[20] i projektmappen Modulbeskrivelser for alle funktioner kan findes i Software Dokumentation, afsnit SWA3[21] i projektmappen.

Store dele af softwaren er skrevet i C++, med enkelte funktioner importeret fra Arduino IDE-softwaren og brug af VHDL til DE2-boardet. Desuden benyttes VHDL funktionalitet med DE2-Board. Dette valg af sprog kom, da C++ er sproget der er bedst erfaring med. Lignende benyttes Atmel Studio til at skrive koden, idet det giver bedre kompatibilitet med Arduino. Indkluderingen af Arduino IDE skyldes primært behovet for direkte kommunikation mellem computer og SA, hvoraf Arduino IDE giver adgang til en række funktioner designet til dette formål.

Yderligere har der også været behov for at benytte Arduino IDE-funktioner, i tilfælde hvor C++ funktionerne ikke har været tilstrækkelige. Arduino IDE har givet muligheden for at hente softwaren direkte ned på systemets Arduino enheder, så information gemt i variable og arrays kan opdateres kontinuert, fremfor at gemme dem på systemets computer. Dette gør at systemet kan holde en aktiveret mode kørende på trods af om den tilsluttede computer er tændt eller slukket.

Software designet af systemet består af fire dele:

- *Arduino_sender*: Sender Arduinoen skal have software, der kan læse et input fra tastaturet på systemets computer, og sende det passende signal videre til. Det er også denne del af softwaren, der står for at printe UI'en gennem PuTTY.
- *Arduino_receiver_dimmer*: Denne software del skal modtage instrukser til dimmeren, og kunne sende PWM-signaler ud fra dem.
- *Arduino_receiver_switch*: Denne software del skal modtage instrukser til switch'en, og kunne sende et HIGH-LOW signal ud fra dem.
- *code_lock_UART:* Skal modtage kode af Super Bruger, vurdere om det indtastede kode er korrekt og sende et HIGH eller LOW signal tilbage til SA. Hvis koden indtastes forkert 3 gange bliver systemet låst.

Til kommunikation mellem modulerne benyttes to forskellige kommunikationsformater. Der benyttes for DE2-Boardet en UART seriel, der tillader kommunikation med SA.

For at muliggøre kommunikationen mellem SA og MA over elnettet benyttes der manchester-kode. Protokollen er baseret på binær, og er derfor velegnet til brug sammen med X10, da der derfor kun er behov for HIGH-LOW signaler.

Baseret på en clock, her et zeroCrossing signal, sendes enten HIGH efterfulgt af LOW for 1, eller LOW efterfulgt af HIGH for 0. Derved kan information overføres over elnettet af X10. Manchester-kode er også brugbart, idet aflæsning altid sker på en rising eller falling edge, som gør det nemt for Arduino modulerne at sende og modtage koden.

[20] Ref[D1]
[21] Ref[D1]

9.1.1 Arduino sender

SA er sat op til at kunne modtage signaler fra en computer ved hjælp af Serial.read() funktionen, der bliver tilgængelig ved hjælp af en sketch, der er lavet i Arduino IDE, som aktiveres ved kommandoen Serial.begin(9600), hvor 9600 er den serielle BAUD rate. Arduino IDE er et program/sprog, der kommunikerer direkte til Arduinoen. Alle kommandoer der bruges via Arduino IDE, tilgås via biblioteket <Arduino.h>, der kommer fra førnævnte sketch.

Appen PuTTy gør det muligt at skabe en UI, der kan sende de serielle forbindelser, der transmitteres mellem computer og Arduino. PuTTy er en terminalemulator, der kan skabe en seriel forbindelse, med en bestemt BAUD rate, hvorved der kan kommunikeres direkte med Arduinoen, via en desktop app. En illustrering af UI'en kan findes i software Dokumentation, afsnit SWA5[22] i projektmappen.

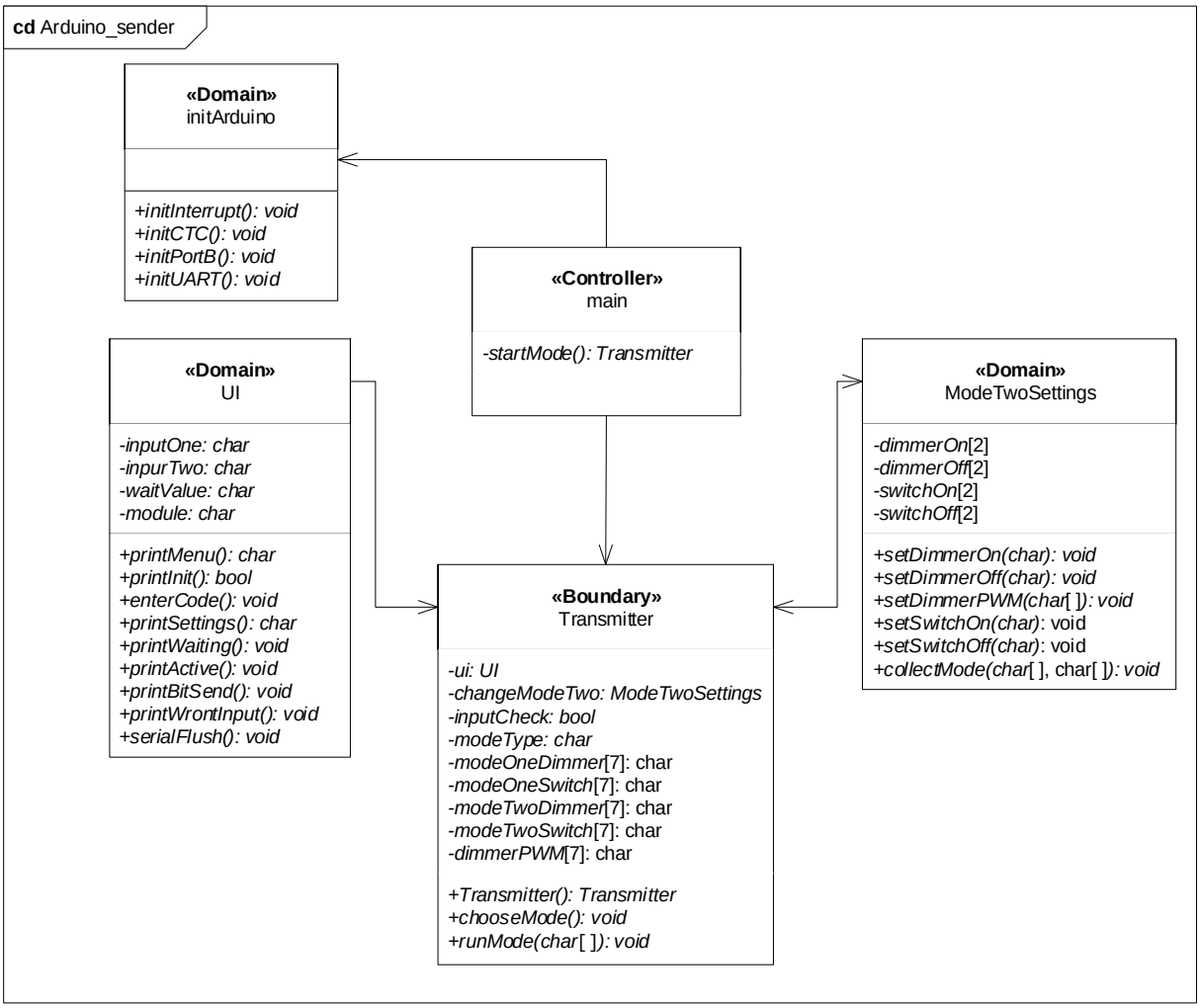
Arduino IDE giver adgang til Serial.print(), Serial.println(), Serial.read(), Serial.available() og Serial.write(). Serial.print()/Serial.println() bruges til at videregive information fra Arduino til PuTTy/computeren som tekst. Serial.write() bruges til at skrive kommandoer til PuTTy/computeren. I koden bruges Serial.write(12) til at rydde konsollen. Serial.read() læser brugerens input og Serial.avilable() sikre, at programmet er klar til nyt input.

Ved brug af Arduino IDE og PuTTy, skabes et nemt og overskueligt interface, som brugeren kan benytte til at kommunikere med SA. Ved hjælp af PuTTy, gøres det yderligere muligt at lave en desktop app. På denne måde, er det nemt og enkelt, at åbne systemet, og udfører den ønskede handling.

En smart feature ved PuTTy og Arduino IDE, er at funktionaliteten af SA ikke kræver yderligere information fra brugeren/computeren end at sættes i gang. Programmet kan altså lukkes på computeren, mens arduinoen stadig udføre sit arbejde.

Selve koden er blevet skrevet i C++, hvilket ses ved brug af klasser og objekter. Klasserne er blevet opdelt, så de diverse funktioner SA skal kunne udføre, består af dedikerede klasser, der udføre det givende stykke arbejde. Se klassediagrammet på Figur 16 for det fulde overblik over klasserne.

[22] Ref[D1]



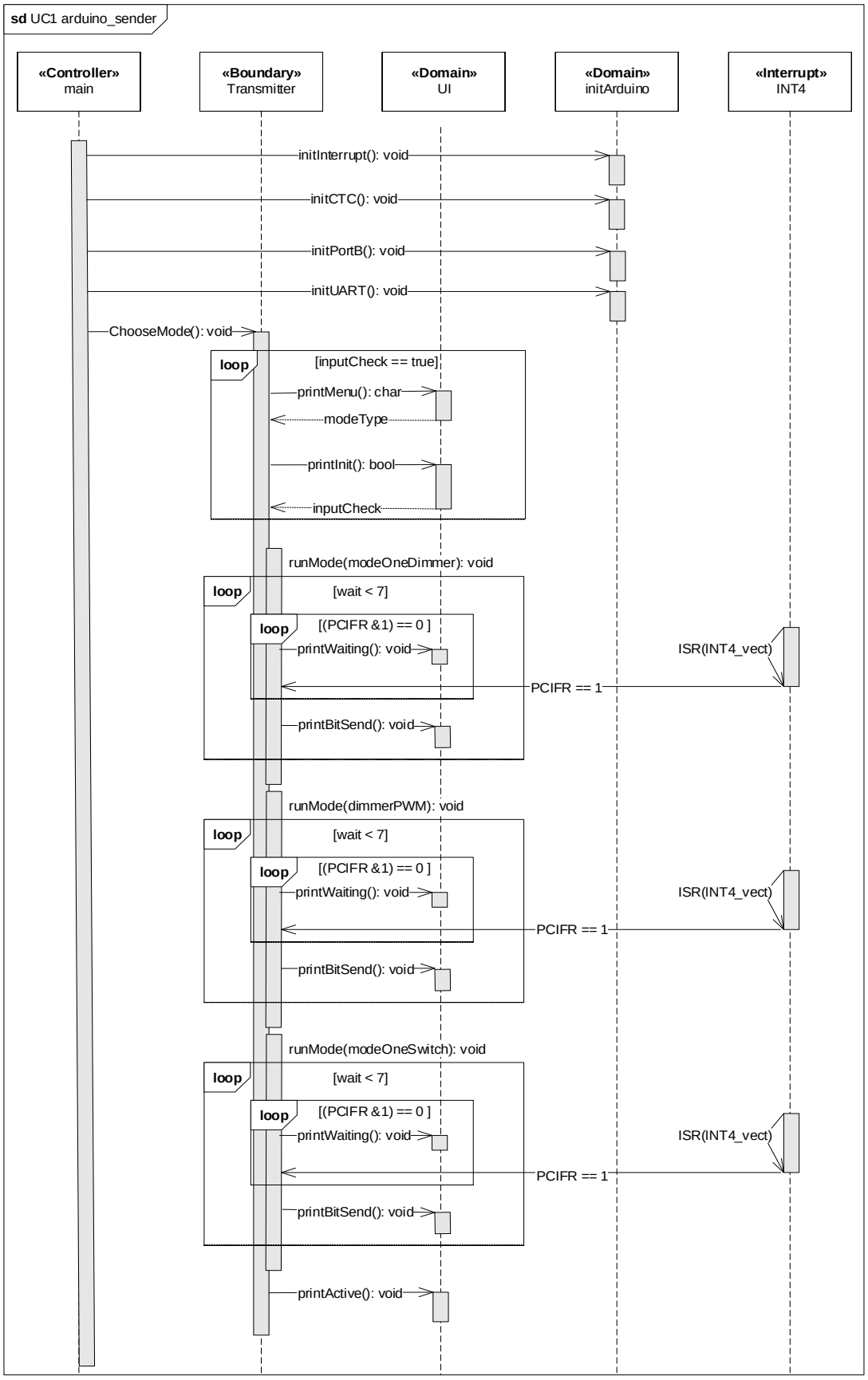
Figur 16: CD Arduino_sender

Arbejdet er opdelt i tre klasser, med en enkelt utility klasse:

- Transmitter, der er i stand til at videresende et array, der består af binære værdier, altså, 0 (LOW) eller 1 (HIGH). Dette gøres ved start-, slut- og specifikations bit. Der er tre arrays, et der indeholder dimmerens tænd/sluk, et der indeholder et PWM-signal der bestemmer lysets intensitet og et der indeholder switchens tænd/sluk. MA kan kende forskel på hvilket array der tilhører Dimmer og Switch, ved hjælp af førnævnte start-, slut- og specifikations bit. Alle tre arrays sendes via interrupts på enten rising og faling edge, der bestemmes ud fra den givende værdi af arrayet der videresendes.

- UI, der interagerer med Bruger, ved visning af information fra Arduinoen og modtage information fra brugeren. Her bruges PuTTy, der giver en seriel kommunikationsvej mellem Arduino og bruger. Arduinoens interne værdier vil blive ændret af brugeren, ved hjælp af serial.read() funktionen og kommunikere værdier tilbage til PuTTy via Serial.print()/Serial.println().

- changeModeTwo der bruges til ændring af Mode 2. Denne bestemmer hvilke værdier der skal indsættes på given array plads for Mode 2, der gør at Mode 2 vil få nye start/sluk og intensitets værdier.



Figur 17: SD uc1 sender

Med udgangspunkt i UC1 er der opstillet et sekvensdiagram for SA klasser og funktioner. Programmet indledes ved at kører fire initialiseringsfunktioner, hvorefter den primære funktion, chooseMode(), køres.

Her benyttes først funktionen printMenu(), derudover til at printe UI menu i konsollen og returnerer et input fra keyboardet, til at sætte variablen modeType. Funktionen printInit() printer en besked i UI afhængig af inputtet ift. den tidligere menu, og returnere en bool til variablen inputCheck, der benyttes til at sikre et korrekt input.

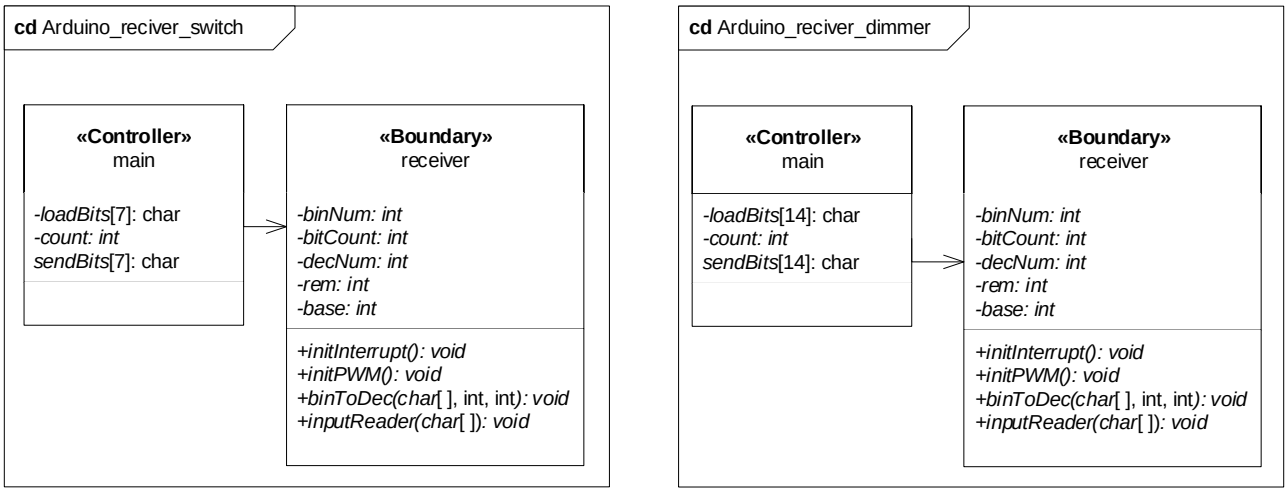
modeType benyttes til at indstille en switch-funktion, hvorfra mode 1 vælges. Herfra køres runMode funktionen så tre gange. runMode kan tage et char array som input, som så sendes som manchester-kode som output ved 120kHz bursts. Hver mode, 1 og 2, har arrays der indeholder bits, som kan aflæses af MA.

Funktionen gør brug af tidligere initialiserede interrupts, og alt afhængig af om det bit der skal sendes er et '0' eller '1', indstilles interrupt til at ske ved enten rising eller falling edge på de eksterne interrupt ben. Efter interruptet er indstillet venter funktionen på at et interrupt forekommer, hvorefter et for-loop sikre alle bits i arrayet bliver sendt.

Interruptet indstiller registret OCR3A, så der sendes et CTC signal på 120kHz som output, hvorefter registret EICRB indstilles til low-input for at sikre der ikke sker unødige interrupts. Delay() funktionen giver os den ønskede længde på burstet, og global interrupt bliver så disabled, for at gøre klar til at sende næste bit.

9.1.2 Arduino modtager

AM'erne modtager informationen sendt fra SA. Hvorved den kunne tænde og slukke for en dimmer, switch på de korrekte tidspunkter og indstille intensiteten af dimmeren. Koden der bliver downloadet ned på de to MA'er vil være næsten identisk. Den eneste forskel bliver, at dimmer modtageren også skal modtage information om styrken på det PWM signal, der skal styre styrken af dimmeren. Det ses derfor at arrayet loadBits har længden 7 for arduino_receiver_switch, men er forlænget til 14 for arduino_receiver_dimmer.

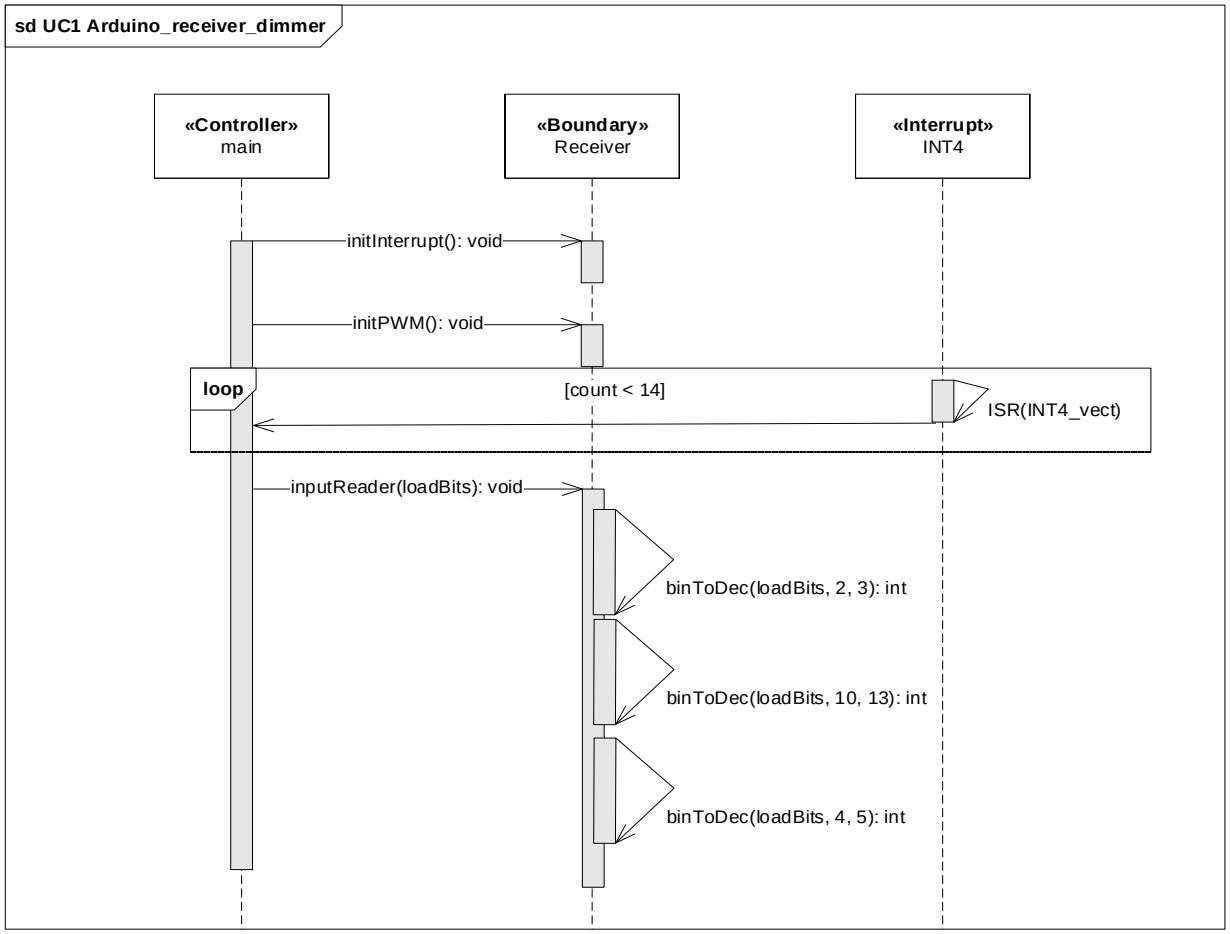


Figur 18: CD for Switch og Dimmer

Med udgangspunkt i UC1 er der sammensat sekvens diagrammer for Dimmer Arduinoen. MA vil være forbundet til en envelope der sender henholdsvist HIGH-LOW ud fra signalet sendt fra SA. Der er også forbindelse til et zeroCrossing-modul, der giver Arduinoen input om hvornår der sker zeroCrossing.

Det initialiserede interrupt(INT4) foretager et interrupt hver gang den modtager en rising edge fra zeroCrossing-modulet. digitalRead() funktionen benyttes så til at aflæse hvorvidt der modtages HIGH eller LOW fra envelopen, hvoraf der så kan afkodes machester-koden, hvorefter det binære tal læses ind i et array, loadBits[].

Variablen count benyttes her til at sikre, at kun den ønskede længde af manchester kode læses ind i arrayet, hvorefter interrupt bliver disabled. Idet dimmer Arduinoen skal modtage en større mængde information, kører den så længe [count < 14]. Funktionen inputReader() benyttes til at læse informationen fra arrayet, og tjekke for start bit, for at sikre at det er de rigtige værdier der læses og behandles. Utility funktionen binToDec() læser en vilkårlig længde af arrayet fra binært, og returnerer den tilhørende decimalværdi.



Figur 19: SD UC1 arduino_receiver_dimmer

Returnværdien benyttes så til at sætte enten ventetiden for at modulet skal tænde, slukke, eller i dimmerens tilfælde, hvor stærkt PWM-outputtet skal være.

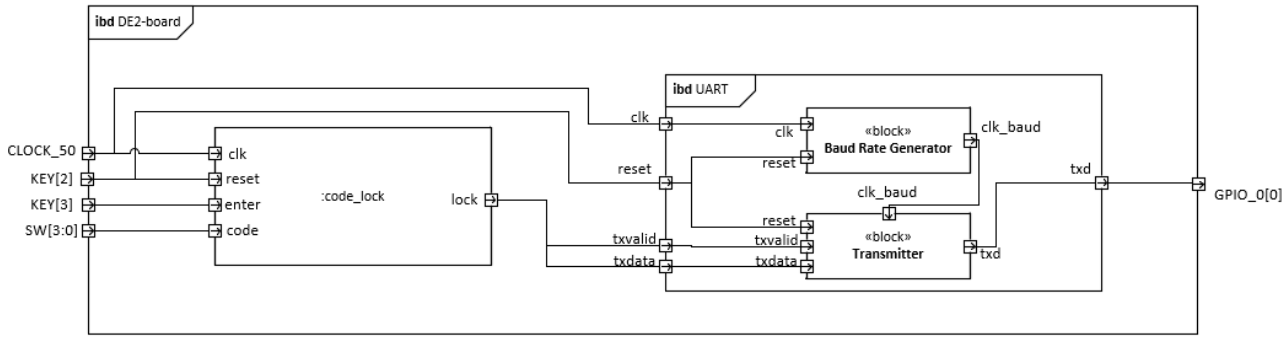
Valget om at først at lægge det modtagne binære tal ind i et array, og først oversætte til decimal efter interruptet er foretaget med belæg. Softwarestrukturen er lavet så udskiftning af kode ville være enkel, og at dele kunne genbruges senere. Ved at benytte den valgte struktur er det nemt at ændre koden hvis der ønskes at indlæse arrayet på en anden måde, dette bygger på *ISE* principperne om lav-kobling.

9.1.3 DE2-board

Systemet er forbundet til et Altera DE2-board, som anvendes i forbindelse med Use Case 3, hvor Super Bruger indstiller mode 2. Når Super Bruger vælger "Definer Mode 2", beder systemet om at få en kode fra Super Bruger.

På Figur 20 er der vist et IBD over hvordan DE2-boardet er programmeret, herunder hvilke inputs Super Bruger kan benytte til indtastning af kodesekvensen og hvilke outputs, der viser bl.a. den pin Arduinoen skal forbindes til for at oprette kommunikationen mellem DE2-Board og Arduino. Derudover vises det at softwaren består af en code_lock, en baudrate generator og en UART-transmitter.

Da der er arbejdet med UART- og code_lock-opgaver i DSD-kurset, er disse benyttet som inspiration til opbygning af koden. Koden skrives i VHDL, da det er det sprog, der er blevet anvendt tidligere, når der arbejdes med FPGA-boardet. Koden og beskrivelsen kan findes i software Dokumentation, afsnit SWA6[23] i projektmappen.



Figur 20 - IBD for DE2-board

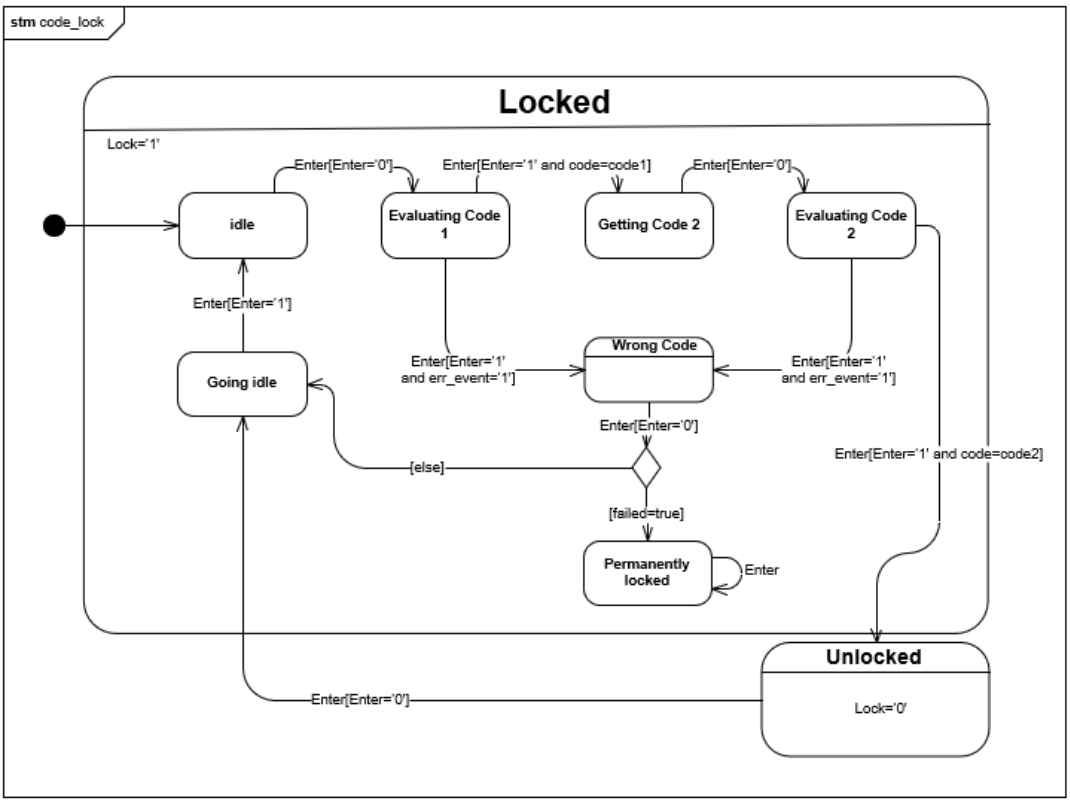
Navn	I/O	Beskrivelse
CLOCK_50	Input	50 MHz clock input
KEY[2]	Input	Knap til reset
KEY[3]	Input	Knap til enter
SW[3:0]	Input	Switches til indtastning af kode
clk	Input	50 MHz clock input den får fra CLOCK_50
reset	Input	Asynkron reset
enter	Input	Enter
code	Input	Kode
lock	Output	Sender 0 eller 1 ud afhængig af om koden er korrekt eller ej
txd	Output	Data bit som sendes ud via GPIO_0[0]
txvalid	Input	txdata bliver sendt til txd, når txvalid=1
clk_baud	Output	Signal fra baud rate generator
txdata	Input	UART data ouput
GPIO_0[0]	Output	Pin på DE2-board, der skal forbindes til Arduino

[23] Ref[D1]

Den første del, code_lock, sørger for kodelåsen. Kodelåsen får input fra Super Bruger, der udfører kodesekvensen, via SW[3:0] efterfulgt af KEY[3]. Herefter vil code_lock foretage en validering af den indtastede kode, for at se om denne er korrekt eller ej, hvis koden er korrekt, sendes der logisk '0' ud gennem output-signalet "lock", ellers sendes defaultværdien '1' hvis koden er forkert. Værdien fra output-signalet "lock", vil herefter gå ind som input i UART'en gennem txdata. UART-controlleren er den, der giver adgang til at sende data fra DE2-boardet til Arduinoen.

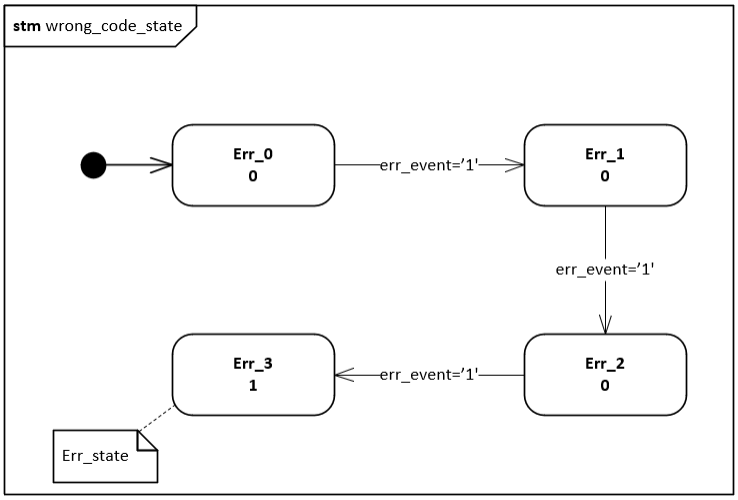
Typisk indeholder UART en receiver, transmitter og baud rate generator. Dog er der intet behov for receiveren i dette tilfælde, da DE2-boardet kun skal sende et signal til Arduinoen, og ikke modtage et signal fra den. UART´en er derfor udelukkende bygget af en transmitter og en baud rate generator, som det ses på Figur 20. Transmitteren sender det data, den får fra code_lock ud på pin GPIO_0(0) på DE2-boardet gennem txd. Herfra kan det nu forbindes til Arduinoen via en ledning.

På DE2-boardet benyttes 4 switches til at indtaste koderne "1111" og "0000". Herefter skal programmet på DE2-boardet kunne foretage en validering af koden. Nedenstående state machine diagram, beskriver code_lock systemets opførsel.



Figur 21 - STM for code_lock

Hvis kodesekvensen indtastes rigtigt, skifter programmet tilstand til "unlocked", hvilket betyder at kodelåsen er deaktiveret. Indtastes koden forkert, vil programmet forblive i "locked" state og fortsætte med at sende HIGH-signaler '1' til Arduinoen.



Figur 22 - STM for wrong_code_state

Desuden, hvis koden indtastes forkert tre gange, låses DE2-boardet permanent, og herefter skal Super Bruger trykke på reset knappen KEY(2), for at kunne prøve igen. Når kodelåsen låses op og programmet står i "unlocked" state, sender DE2-boardet LOW-signal '0' over til Arduinoen igennem UART_transmitteren. Herefter får Super Bruger adgang til at kunne ændre i indstillingerne der omtales i UC 3.

Udførte tests samt opstillingen af DE2-boardet kan findes i software Dokumentation, afsnit SWA7[24] i projektmappen.

9.2 Hardware

9.2.1 Simulink

Grundet de specielle omstændigheder der har fundet sted i projektets forløb med COVID-19 har en realisering af hardwaren, samt integrationen mellem software og hardware, ikke været muligt. Det har derfor været nødvendigt at kigge på alternative metoder at designe hardwaren samt produktets funktionalitet. Der er derfor lagt en del tid i egen oplæring af simuleringsprogrammet *Simulink*, som er en del af *MATLAB*. Simulink er et grafisk værktøj som muliggør modellering, analyse og simulering af systemer. Programmet giver stor frihed til at opbygge sit system og teste om det vil virke.

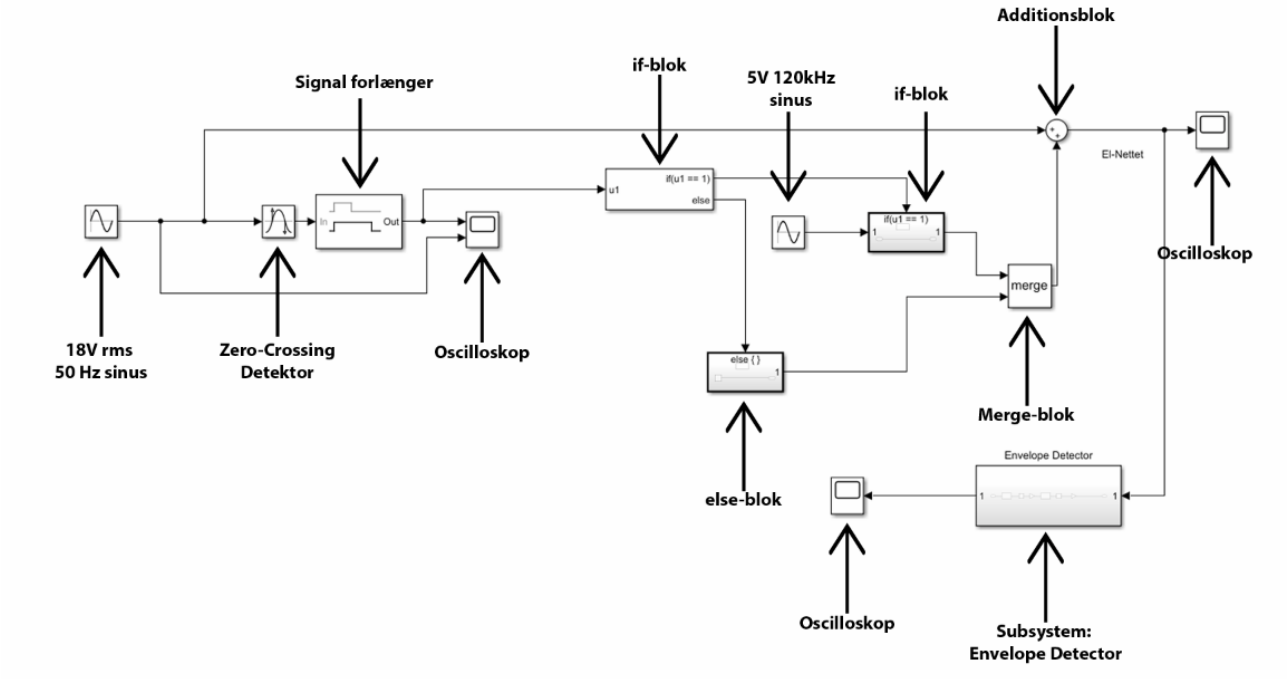
Der benyttes blokke i programmet, hvor værdier og virkemåde kan modificeres på egen hånd. De forskellige blokke kobles med en enkelt streg IN/OUT. Simulink er et værktøj, der giver mange muligheder for modellering og tests, grundet konfigurations indstillinger samt tilgængelige blokke.

Værktøjet er blevet benyttet i design-delens startfase. Her er hele X10 systemet opbygget og der kigges på virkemåden overordnet, samt sammenspillet mellem de enkelte blokke. Det overordnede system kan ses på Figur 23.

Systemet er bygget op af en Zero-Crossing blok, se nærmere på Hardware Dokumentation, afsnit HW1 [25] for information omkring zero-crossing blokken og oscilloskopet.

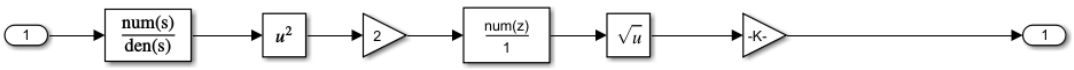
[24] Ref[D1]
[25] Ref[D2]

Ud fra den overordnede viden om systemet, vides det at mikrocontrolleren aflæser disse zero crosses, og derefter kan mikrocontrolleren på baggrund af dette, sende data i form af et 120kHz signal ud på lysnettet på det korrekte tidspunkt. For at lave dette koncept i Simulink er det blevet lavet en if-blok. I alt sin enkelthed vil if-blokken sørge for at sende et 120-kHz sinus signal ud og addere det med det oprindelige sinus-signal, hvis der er Zero-crossing. Hvis der ikke er Zero-Crossing, så sker der intet med det oprindelige signal. Se Hardware Dokumentation, afsnit HW1[26]



Figur 23: Simulink opsætning

Efter at der er blevet sendt data ud på lysnettet skal det være muligt at opfange disse data med et modtager-modul. Princippet bag modtager modulet er at det udsendte data skal opfanges af en Envelope Detector, der er blevet implementeret i et subsystem. Indholdet af subsystemet kan ses på Figur 24.



Figur 24: Subsystem: Envelope detector

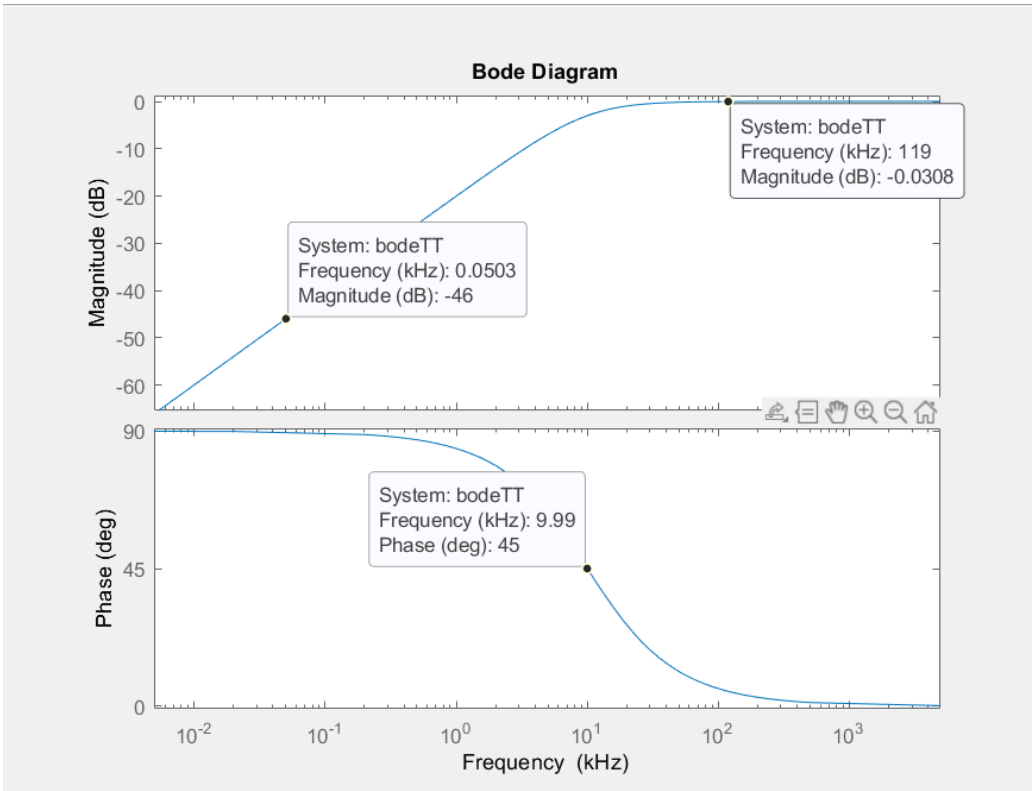
I Envelope Detectoren ses først et analogt højpas-filter. Højpas-filteret er lavet ved at vælge et filter blok højpas-filteret bliver opstillet ved nedenstående overføringsfunktion. Se udregning i Hardware dokumentation afsnit HW1[27]

$$G(s) = \frac{1.592 \cdot 10^3}{\frac{1}{s \cdot 10 \cdot 10^{-9}} + 1.592 \cdot 10^3} = 1 - \frac{6281}{s + 6281}$$

[26] Ref[D2]
[27] Ref[D2]

Ud fra standard bodeplot[28] vides at α er lig med knækfrekvensen, standard bodeplottet stemmer overens med det bode plot der er blevet genereret af Simulink.

Bodeplottet på Figur 25 viser, at der ved 50 HZ ca. dæmpes med 56 DB, ved 120 kHz dæmpes der ikke. Samt at knækfrekvens er ved 10 kHz. Se Hardware Dokumentation.pdf afsnit HW1[29] for mere information omkring implementering af Envelope-Detektoren.



Figur 25: Bodeplot for højpas-filter

Ved at se på det samlede kredsløb kan der antages at det konceptuelle kredsløb burde virke. Altså er der blevet påvist at de 3 moduler, Zero-Crossing, Sender, og Modtager kan kommunikere med hinanden. Dette vil dog kun være muligt hvis de forskellige moduler bliver designet således at de forskellige signaler der bliver sendt ud på kredsløbet, overholder de samme principper som der er blevet påvist på simulink. Samt at signaler bliver bearbejdet på samme måde som de forskellige blokke i simulink bearbejder signalerne.

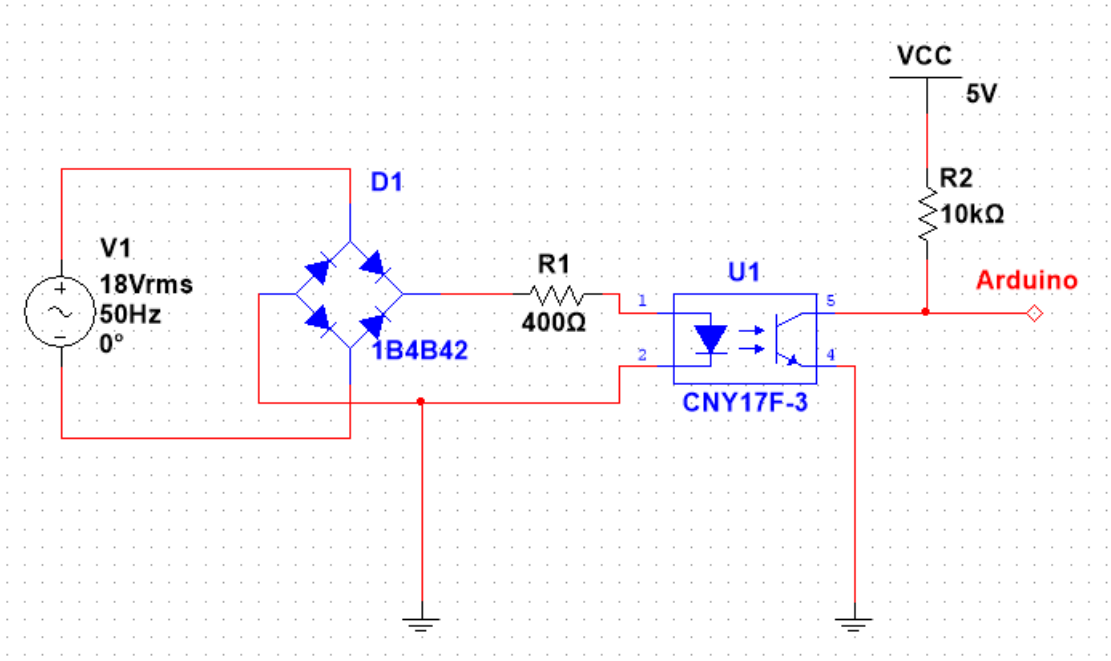
Simulink redskabet er blevet brugt som en tilføjelse til projektet da det ikke har været muligt at bygge et produkt. For at lære simulink redskabet at kende har en del af projektgruppen brugt en stor mængde tid på at eksperimentere med programmet samt fulgt et online kursus[30], for at få en basis forståelse af

[28] Ref[G6]
[29] Ref[D2]
[30] Ref[G7]

programmet. Herefter har der med en smule vejledning været muligt at danne basale kredsløb samt basis forståelse af programmet til fremtidigt brug.

9.2.2 ZeroCrossing

For at kunne kommunikere over lysnettet er det nødvendigt at vide hvornår der er Zero Crossing. Dvs. når 18 V AC-signalet på 50 Hz krydser 0 V. For at signalere til Arduinoen hvornår der er Zero Crossing opbygges kredsløbet således at der er et output på 5 V, altså et logisk 1, når der er Zero Crossing, og 0 V når der ikke er.



Figur 26: Zero crossing design

Zero Crossing modulet (se Figur 26) er lavet ud fra en online undersøgelse[31] , her blev der hurtigt fundet ud af at der skulle bruges en optokobleren og en diodebro. Ved behandling af det harmoniske signal på 50Hz ønskes det at de negative halvperioder vendes til positive, og det er her diodebrokobleren benyttes. Diodebroen 1B4B42[32] er blevet benyttet. Der forklares yderligere om kredsløbet i Hardware Dokumentation, afsnit HW2[33].

Optokobleren benyttes så Arduinoen beskyttes fra kredsløbet, da den ikke kan klare 18 V 50 Hz kredsløbet. Optokobleren CNY17F-3[34] benytter sig af en lysdiode, når lysdioden lyser, løber der en strøm igennem fototransistoren. Herved løber de 5 V fra forsyningen direkte til ground. Når sinuskurven nærmer sig 0 vil lysdioden stoppe med at lyse og spændingsforsyningen kan løbe ind til Arduinoen, der sættes en 10k Ω modstand foran for at sikre at der ikke løber en for stor strøm ind i Arduinoen.

[31] Ref[G3]
[32] Ref[G8]
[33] Ref[D2]
[34] Ref [G9]

$$\frac{5V}{10k\Omega} = 0.5\ mA$$

Arduinoens i/o pins kan ikke håndtere mere end 40 mA[35], herved sikres at boardet ikke ødelægges.

Ud fra optokobleren datablad er der blevet bestemt at der er en max-current på 60 mA og et spændingsfald på 1.65 V. Herved bestemmes for modstanden således at max-current ikke bliver overskredet.

Ved et 18 V RMS-kredsløb bestemmes peak amplituden.

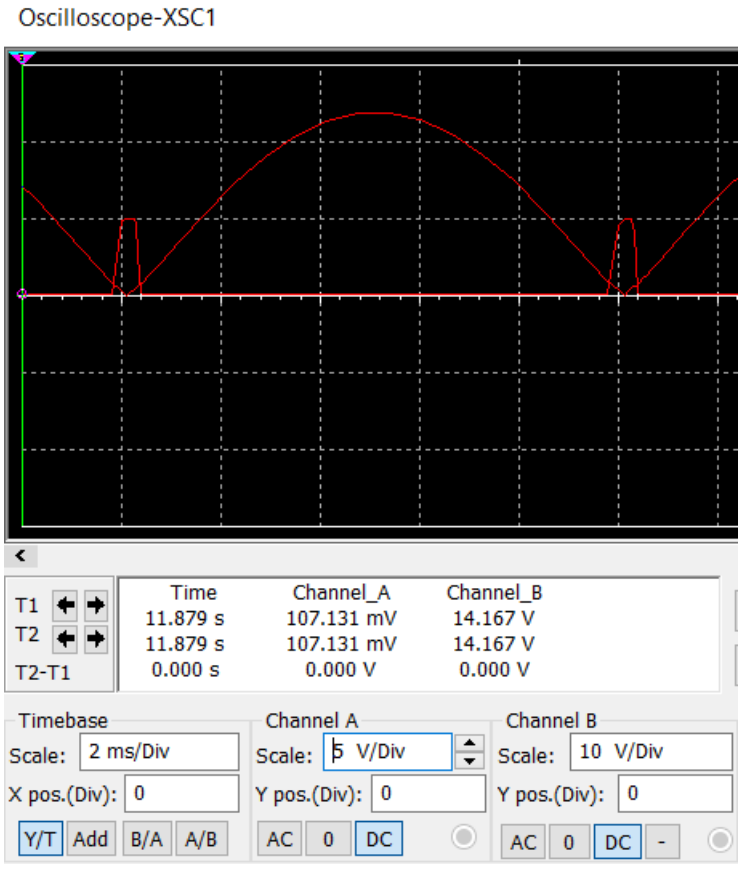
$$18V \cdot \sqrt{2} = 25{,}456\ V$$

Herefter bestemmes for modstanden.

$$\frac{25{,}456V - 1{,}65V}{60mA} = 396{,}77\Omega$$

For modstanden bliver altså 400 Ω.

På Figur 27 ses der hvordan zero-crossing signalet ville komme til at se ud.



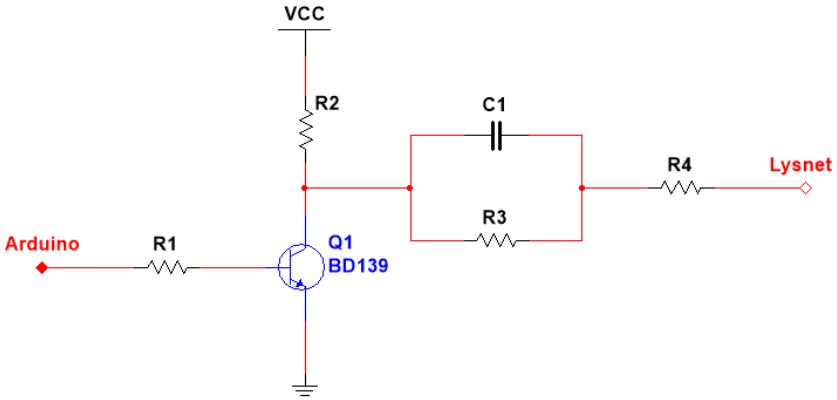
Figur 27: Zero crossing

[35] Ref [G10]

9.2.3 Carrier Generator

Carrier generator er et elektronisk kredsløb som får et input fra Arduinoen, som er en del af Sender-Modulet, og giver outputtet til lysnettet. Denne hardwaredel gør det muligt at sende 120kHz bursts ud på lysnettet, hvoraf disse bursts senere opfanges af Carrier detektor. Carrier Generatoren er bl.a. opbygget af en transistor og en kondensator.

Transistoren skal virke som en switch. Dette betyder at når Arduinoen sender et 120kHz burst ind på basebenet på transistoren, så vil denne 'åbne sig' og lade de 120kHz passere ud på elnettet. I kredsløbet er der også et afkoblingsled bestående af en kondensator og en resistor. Dette led har en beskyttende faktor samt at det muliggør at de førnævnte bursts kan passere ud på lysnettet. Det er ikke normal procedure at tilkoble et svagstrømskredsløb direkte på lysnettet, men nødvendigt i dette tilfælde. For mere information om dette anbefales det at se Hardware Dokumentation afsnit HW3[36]. Nedenstående viser opbygning af Carrier generatoren på Figur 28.



Figur 28: Carrier Generator

På transistorens collector-ben sidder en pull-up resistor, R2, som har til formål at forhindre en kortslutning til emitter-benets stel når transistoren åbner sig.
Formodstanden, R1, til base-benet skal udregnes således at transistoren er i mætning. Lykkedes dette ikke, vil transistoren ikke 'åbne sig' når der bliver udsendt 120kHz fra Arduinoen. Udregning af basebenets resistor, R1, er fundet på følgende måde:

$$R_B := \frac{V_{in} - V_{BE}}{I_B} = 888.889\ \Omega$$

Der udregnes en modstandsværdi på ca. 888Ω. Ud fra Standard Decade Value Table[37] vælges en resistor størrelse på 900Ω med 5% tolerance.

Afkoblingsleddet i kredsløbet, nærmere bestemt kondensatoren, bliver der kigget yderligere på. I applikationsnoten[38] er dette tilkendegivet som et højpasfilter. Der er altså to kilder der påvirker kredsløbet, henholdsvis lysnettet på 18V AC 50Hz og Arduinoens 120kHz 0-5V firkant signal. Det er altså muligt at opstille Thevenin-ækvivalenter for kredsløbet, hvor kredsløbet kan analyseres vha.

[36] Ref[D2]
[37] Ref[G11]
[38] Ref[G2]

superposition. Der kan så opstilles overføringsfunktioner som standardfunktioner hvoraf der kan skabes bodeplots, som kan fortælle påvirkning mellem de to kredsløb, og om det er forsvarligt at koble svagstrømskredsløbet direkte på lysnettet.

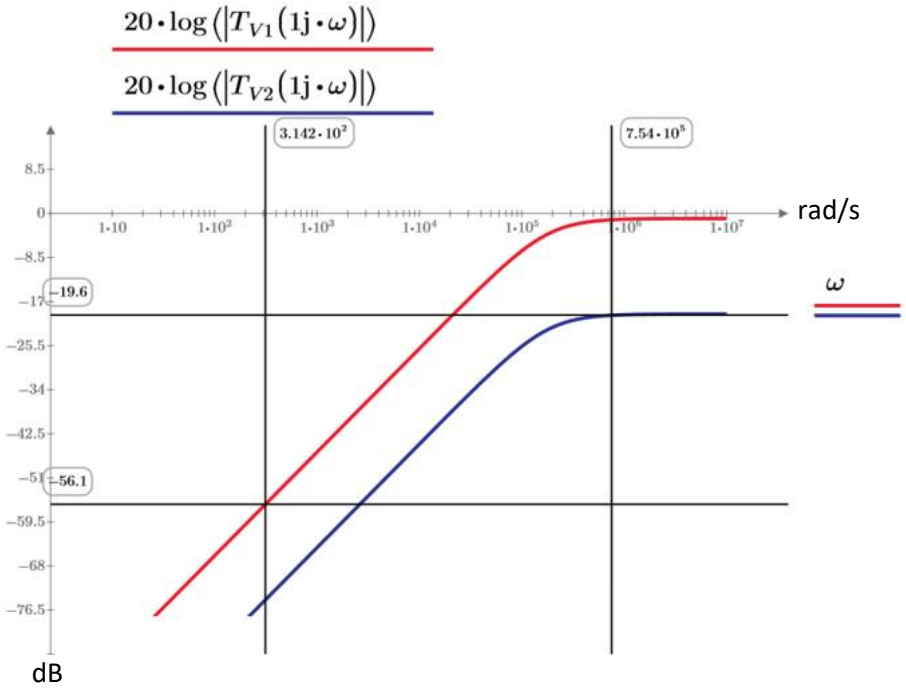
Der undersøges hvordan lysnettet påvirker elektronikken, hvor Arduino-kilden slukkes ved superposition. Hertil fås følgende overføringsfunktion.

$$T_{V1}(s)=\frac{V_{out}}{V_{in}}=\frac{\frac{1}{(R2+R4)\cdot C}}{s+\frac{1}{(R2+R4)\cdot C}}\cdot\frac{s}{\frac{1}{R2\cdot C}}$$

Dernæst undersøges hvordan Arduino 120kHz påvirker lysnettet. Ved superposition slukkes lysnet-kilden og følgende der fås følgende overføringsfunktion.

$$T_{V2}(s)=\frac{V_{out}}{V_{in}}=\frac{\frac{1}{(R2+R4)\cdot C}}{s+\frac{1}{(R2+R4)\cdot C}}\cdot\frac{s}{\frac{1}{R4\cdot C}}$$

Udfra disse er der blevet udformet bodeplots.



Figur 29: Bodeplot for Carrier Detektor

Hvor de vertikale cursors er frekvensen i rad/s og horisontale cursors er dæmpning i dB, begge udregninger vises på nedenstående.

$$120000\ Hz\cdot 2\cdot\pi=753982.237\ \frac{rad}{s}$$

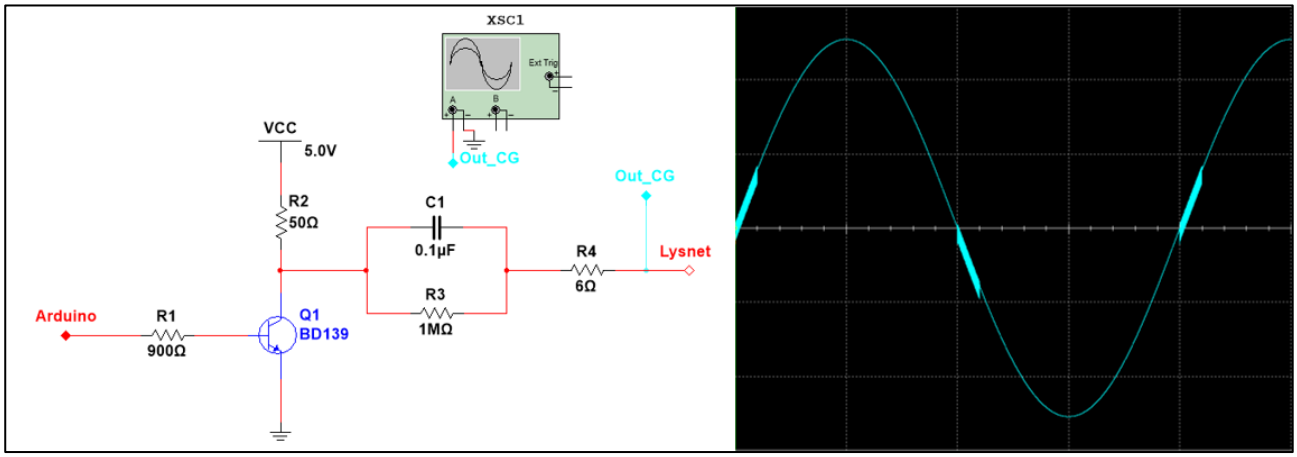
$$50\ Hz\cdot 2\cdot\pi=314.159\ \frac{rad}{s}$$

$$20\cdot\log\left(\left|T_{V1}(1j\cdot 50\cdot 2\cdot\pi)\right|\right)=-56.078$$

$$20\cdot\log\left(\left|T_{V2}(1j\cdot 120000\cdot 2\cdot\pi)\right|\right)=-19.638$$

Det kan konkluderes at lysnet-delen bliver dæmpet markant og er ikke i nærheden af pasbåndet, men må derimod siges at være i stopbåndet. Ud fra beregningerne kan det konkluderes at, der er en lille dæmpning på Arduino-120kHz, dog vil det siges at kunne accepteres da det stadig passerer og kan forstærkes hvis nødvendigt. Som også set på bodeplottet, må det siges at Arduino-120kHz er ved, eller næsten i, pasbåndet og undgår stopbåndet.
Samlet set kan det siges at koblingen mellem lysnet og et svagstrømskredsløb er muligt, selvom det ikke er normalt procedure.

Med inspiration fra appliaktionsnoten[39] samt egne udregninger er det der bestemt værdier for alle komponenter i kredsløbet. Der er bestemt R1 således transistoren er i mætning, så den ønskede funktionalitet opnås. R2, C1 og R3 er alle analyseret og testet endegyldig er R4 en modstand der kommer af transformatoren, som er nødvendigt for at konvertere 230V AC fra stikkontakten til 18V AC som der bliver benyttet i dette projektarbejde. Endnu engang anbefales det at se Hardware Dokumentation, afsnit HW3[40] for alle beregninger samt analyse. På Figur 30, er et billede af Carrier Generator med endelige komponentværdier, samt et oscilloskop billedet af output, som giver et billede af funktionaliteten af Carrier Generatoren.

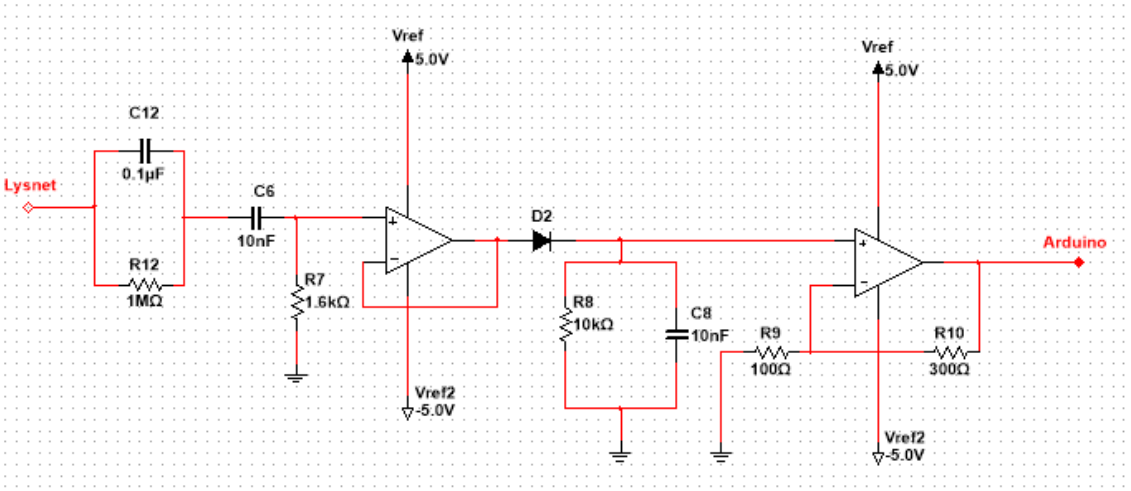


Figur 30: Endelige Carrier Generator

[39] Ref[G2]
[40] Ref[D2]

9.2.4 Carrier detektor

Carrier Detektoren er blevet bygget på baggrund af applikationsnoten[41]. Detektor Modulet (se figur 31) bliver lavet vha. en række forskellige filtrer og operations-forstærkere.



Figur 31: Endelige Carrier Detektor

Det første filter der ses i kredsløbet, er magen til det der er blevet lavet i sender kredsløbet. Som det også er forklaret Hardware Dokumentation afsnit HW3[42], er dette sat på for at beskytte svagstrømskredsløbet når det kobles direkte på lysnettet, mere herom i førnævnte afsnit. Dernæst kommer der et aktivt-højpasfilter. Her der er blevet lavet nogen undersøgelser[43] om filtrer. Det nævnte filter er magen til det der er blevet skrevet i Hardware Dokumentation afsnit HW1[44]. Den sidste del af modtageren laves vha. en Envelope Detektor, som er blevet undersøgt online[45], herefter sættes en operations forstærker på til at forstærke signalet til den rigtige spænding, 5V, denne operations forstærker har den egenskab at den 'klipper' envelope signalet. Herved fås et pænere 5V signal, der langsomt aflades, i stedet for et ujævnt signal, der potentielt ikke nåede en høj nok udgangs spænding. Operationsforstærkeren efter høj pas-filteret sørger for at filtret og envelope-detektoren ikke har indflydelse på hinandens impedanser, og derved ikke forstyrrer input signalet på de 120 kHz.

Høj-pas filteret bliver designet således at der er en knæk-frekvens på 10 kHz. Udregningen af dette filter er blevet beskrevet i Hardware Dokumentation, afsnit HW4[46]. Overføringsfunktionen er givet ved:

$$G(s) = \frac{1.592 \cdot 10^3}{\frac{1}{s \cdot 10 \cdot 10^{-9}} + 1.592 \cdot 10^3} = 1 - \frac{6281}{s + 6281}$$

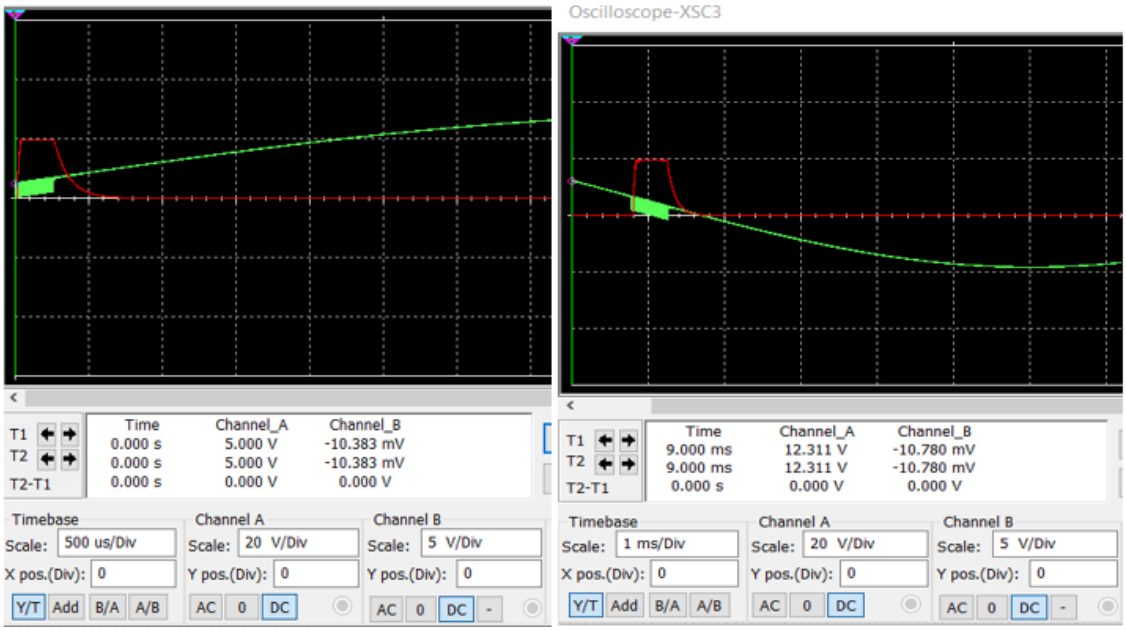
Herved vides at bode plottet vil se ud som bodeplottet der blev lavet i Simulink. Her dæmpes der med 46 DB ved 50 Hz. På denne måde kommer 50 Hz signalet ikke igennem, hvorimod 120kHz signalet vil

[41] Ref[G2]
[42] Ref[D2]
[43] Ref[G4]

[44] Ref[D4]
[45] Ref[G5]
[46] Ref[D2]

løbe uhindret igennem. Igen henvises til Hardware Dokumentation, afsnit HW3[47] for test og undersøgelse af højpas-filteret.

Efter dioden på figur 31 ses Envelope Detektoren, der består af et parallelt RC-kredsløb. Her er der blevet valgt værdier magen til dem fra applikationsnoten. Envelope-detektoren, lader 10 nF kondensatoren (C8) op, når der kommer et sinus-signal. Når sinus-signalet nærmer sig 0, vil kondensatoren blive afladt, da kondensatoren sørger for at der ikke kommer bratte ændringer i spænding. Pga. dioden, vil sinus-signalet dog ensrette sig. Sidst i modtager kredsløbet ses en operations forstærker der vil forstærke signalet så det kan aflæses af mikrokontrolleren. Til sidst fås output signal som på figur 32.



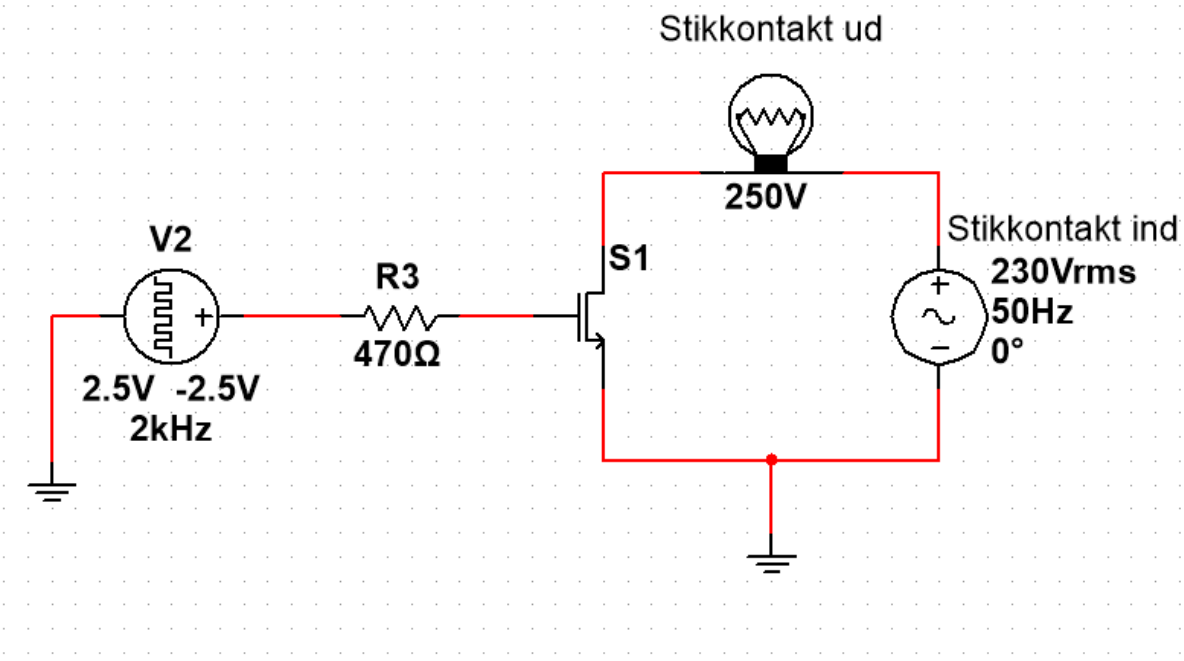
Figur 32: Oscilloskop billede af envelope-detektor ved opad- og nedadgående-sinuskurve

For mere information omkring envelope-detektoren samt test af envelope-detektoren og udregning af forstærkning henvises til Hardware Dokumentation afsnit HW4[48].

[47] Ref[D2]
[48] Ref[D2]

9.2.5 Switch/Dimmer

Switchen/Dimmeren har, som andre hardwaredele i projektet, taget inspiration fra applikationsnoten[49]. Switchen/Dimmeren er lavet ved hjælp af en transistor, som skaber et udgangssignal fra lampen der er halvt så stor som inputtet fra elnettet. Måden den virker på, er at DC-signalet bliver sendt ind på transistoren, som omdanner DC-signalet til et AC-signal der bestemmer hvornår lampen skal være tændt. I teorien skulle lampen være slukket når der ikke bliver tilført et DC-signal og omvendt når der bliver tilført et signal. Dette skaber så en mulighed for at tænde og slukke lampen så hurtigt at ens øjne opfanger det som et svagere lys. Kredsløbet til denne switch/dimmer er vist på Figur 333, som viser sammenhængen mellem PWM-signalet og elnettet. Elnettet består af en stikkontakt ud og en stikkontakt ind, som repræsenterer henholdsvis lampen og elnettet.



Figur 33: Switch/Dimmer

Det er dog svært at sige om dette kredsløb ville virke i virkeligheden, men eftersom der ikke har været mulighed for at teste det, var det nødvendigt at gå med det design som gav det mest præcise resultat i MultiSim. Dette design var designet med transistoren, sådan at PWM-signalet tændte og slukkede for lampen alt efter om det var højt eller lavt. For mere information om switchen/dimmeren henvises der til Hardware Dokumentation afsnit HW5[50].

[49] Ref[G2]
[50] Ref[D2]

9.2.6 Tolerance og produktion

En overvejelse som er blevet gjort i design fasen er komponenternes tolerance. Tolerancen på de enkelte komponenter er forholdsvis vigtigt at have med i sine overvejelser når produktet skal designes. Ydermere, er det også relevant til en fremtidig produktion.

Uden adgang til et laboratorie er pålideligheden af kredsløbet usikkert. Derfor er der i stedet blevet simuleret i de to ydre grænser af komponenternes tolerance. Modstandene der ville blive brugt, har en tolerance på 5%. Kondensatorerne ville blive brugt havde en tolerance på 10%. For at sikre sig at de enkelte moduler stadig sender tilfredsstillende signaler er der for nogen af modulerne blevet lavet en undersøgelse i komponenternes ydre grænser. For mere information henvises til Hardware Dokumenation, afsnit HW5[51] hvor der er beskrivelser af udregninger og simuleringer.

På baggrund af simuleringerne i dokumentationen kan der konkluderes at de forskellige moduler ville virke i de ydre grænser af tolerancerne for de forskellige komponenter. Herved kan der sikres at produktet virker cost-efficient, de hardwarekomponenter der er blevet brugt ville have en høj tolerance på henholdsvis 5- og 10%. Ved at undersøge om modulet stadigvæk virker efter hensigt kan der derved spares penge ved komponenternes kvalitet.

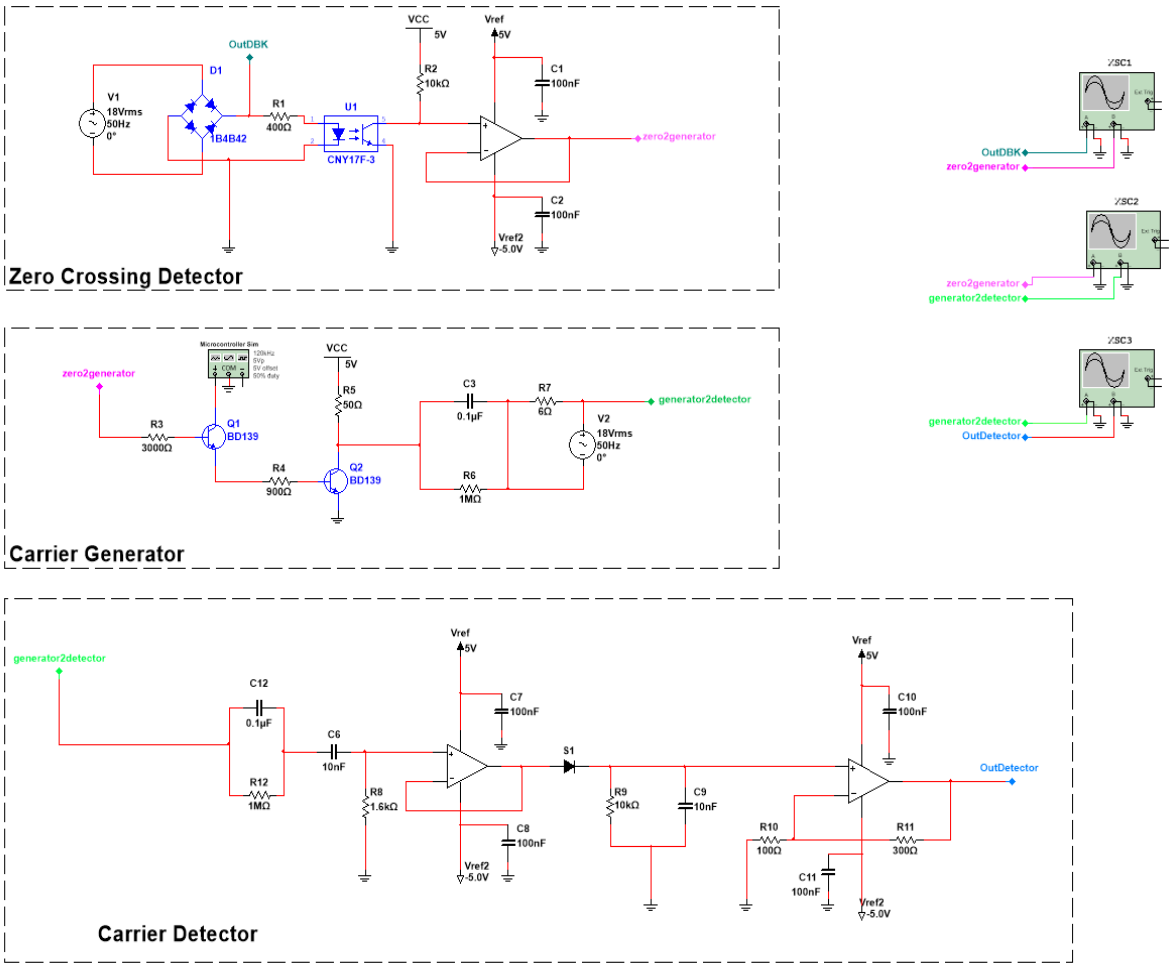
[51] Ref[D2]

10. Test

I dette afsnit vil der blive lavet en kort gennemgang af hvordan de forskellige HW og SW-moduler og klasser er blevet testet. Da det ikke har været muligt at teste de implementerede dele, er testene primært blevet simuleret for at se om de rigtige signaler bliver registreret og bearbejdet.

Da det meste af produktet er rent konceptuelt, kan der derfor heller ikke med sikkerhed siges at de forskellige HW og SW-moduler virker, selv om simuleringen påviser at det burde kunne lade sig gøre. Simuleringerne der normalt bliver brugt som et mellem led, for at se om selve signal-behandlingerne er plausible, bliver hermed brugt som et hint, for at påvise hvordan implementeringen af modulerne ville se ud.

10.1 Modultest hardware



Figur 34: Samlet HW-kredsløb til test

For at lave modul test af de forskellige hardwaremoduler er der blevet valgt at forbinde de forskellige moduler, som der ses på Figur 344. Her kan der ses at de forskellige dele der skal kommunikere, er blevet slået sammen, da det ikke har været muligt at bygge kredsløbet.

Ved Carrier Generatoren er der blevet valgt at påsætte en transistor og en funktionsgenerator før kredsløbet. Dette skal fungere som om Arduinoen sender 120kHz bursts ud. Normalt ville der vha. software bestemmes hvornår der skulle sendes bursts ud, men dette er ikke muligt i en simulering. I stedet er transistoren blevet sat ind, der her virker som en switch.

Når der er Zero-Crossing bliver transistoren mættet og funktionsgeneratorens output på 120 kHz løber ind i sender kredsløbet. For at adskille de forskellige kredsløb og holde styr på impedanserne er der blevet sat operations forstærkere imellem kredsløbene. På hver operations forstærker er der blevet indsat afkoblings kondensatorer på 100 nF, for at minimere det støj der potentielt havde været der når kredsløbet blev bygget.

Herved tages de nødvendige forholdsregler der normalt var blevet taget under opbygning af det reelle kredsløb. Ud fra test af det samlede kredsløb kan der konkluderes at de forskellige hardwaremoduler kan kommunikere med hinanden under konceptuelle forhold. Når der rent faktisk skal kommunikeres over et lukket AC-kredsløb, er det svært at sige hvordan kredsløbet ville reagere, da det kun er blevet simuleret. Der er dog blevet taget højde for de forskellige impedanser, i kredsløbet så forventningen er at der kunne blive bygget et fungerende kredsløb.

For at se udførsel af test henvises til Hardware Dokumentation, afsnit HW7[52].

10.3 Modultest software

Hvert SW-modul er blevet testet parallelt med deres udvikling. Pga. De specielle omstændigheder rapporten er skrevet under har mulighederne for at teste SW været let begrænsede. Det har f.eks ikke været muligt at teste hvorvidt MA faktisk kan aflæse Manchester kode gennem elnettet. Alle funktioner på arduinoen er blevet testet via simuleringer.

I stedet er der for hvert modul først blevet lavet en SW-simulering, hvor terminal vinduet og Arduino shieldet er blev benyttet til at teste programmets afvikling og logik. Terminalvinduet er blevet brugt til at se, om programmet reagere på brugerens input. Arduino shield er blevet brugt til direkte kontakt til Arduinoen, ved brug af interrupts og LED'er.

Efter tilfredsstillende resultater, benyttes en Analog Discovery til at simulere et signal fra envelope -og ZeroCrossing-HW, samt til at aflæse indlejret Manchester-kode.

Testkode for foretaget modultest kan findes i software Dokumentation, afsnit SWA4[53] i projektmappen.

For DE2-boardet er der brugt simuleringsværktøjet i Quartus, hvor man kan opstille en situation for implementeringen og herefter se om de enkelte dele i koden fungerede som ønsket. Da disse resultater kørte som ønsket, blev DE2-boardet herefter koblet til Analog Discovery, så der kunne testes om DE2-boardet sendte det korrekte signal ud. For at opnå dette, er der blevet benyttet Protokol-funktionen i Waveforms, der tillader test af UART-kommunikation. For at se opstilling og resultater af test henvises der til dokumentationsbilaget Software Dokumentation – SWA7[54].

[52] Ref[D2]
[53] Ref[D1]
[54] Ref[D1]

10.4 Integrationstest

Grundet karantænen har en faktisk integrationstest været vanskelig, da der ikke har været mulighed for at konstruere HW-moduler til at teste SW af på. Derfor er der blevet nødsaget til kun at foretage modultest og simuleringer af systemet.

En faktisk integrationstest ville indeholde test af hvert HW-modul med tilegnet SW installeret. Hvert modul ville først blive testet individuelt fra resten af systemet, for at minimere området hvori evt. fejl kunne opstå i. To moduler sættes så sammen, for at sikre der ikke opstår nogen fejl ved koblingen. Dette gentages for alle moduler.

Til sidst samles alle modulerne, og der foretages teste af systemets funktioner. Fejl i systemet ved dette stadie kan evt. føre til at moduler fraskilles systemet og testes separat igen, for at bestemme om fejlen skabes af modulet selv, eller af kommunikation med et andet modul.

10.5 Accepttest

Her testes kravspecifikationerne for systemet. Det noteres her at grundet karantænen har nogen test vist sig ikke at være mulige pga. mangel på HW, så der ikke kan findes fyldestgørende resultater for alle test.

For hver Use Case er der opstillet tilhørende Accepttest, der skal verificeres for at sikre at systemet fungerer som forventet. For hver enkelt test er der noteret handling, forventet resultat, faktisk resultat og vurdering under kapitel 11. Resultater.

11. Resultater

11.1 Use Case 1:

Use Case under test:		Aktiver mode 1		
Scenarie:		Hovedscenarie		
Prækondition:		Systemet er funktionelt og tilgængeligt		
Postkondition:		Mode 1 er aktiveret		
Nr.	**Handling**	**Forventet resultat**	**Faktisk resultat**	**Vurdering (OK/FAIL)**
1.	Brugeren opstarter ”Home protection” softwaren på sin computer	Programmet opstartes	Der åbnes et konsolvindue med start menu	OK
2.	Brugeren vælger ”Vælg mode 1”.	Programmet udskriver ’mode 1 initieres’ Vent Programmet udskriver ’Mode 1 aktiveret”	Programmet udskriver ’mode 1 initieres’ Programmet udskriver ”Waiting” og ”Bit send” Programmet udskriver ’Mode 1 aktiveret	OK (Med forbehold)

Use Case under test:		Aktiver mode 1		
Scenarie:		Deaktivere mode 1		
Prækondition:		Systemet er funktionelt og tilgængeligt.		
Postkondition:		Mode 1 er deaktiveret		
Nr.	**Handling**	**Forventet resultat**	**Faktisk resultat**	**Vurdering (OK/FAIL)**
1.	Brugeren opstarter ”Home Protection” softwaren på sin computer	Programmet opstartes	Der åbnes et konsolvindue med start menu	OK
2.	Brugeren vælger ”Deaktiver Home Protection”.	Programmet fremviser besked ”Home Protection deaktiveret”	Programmet udskriver ’Home Protection is deactivating’ Programmet udskriver ”Waiting” og ”Bit send” Programmet udskriver ‘Home protection is deactivated’ Start menu fremvises	OK (Med forbehold)

11.2 Use Case 2:

Use Case under test:		Aktiver mode 2		
Scenarie:		Hovedscenarie		
Prækondition:		Systemet er funktionelt og tilgængeligt		
Postkondition:		Mode 2 er aktiveret		
Nr.	**Handling**	**Forventet resultat**	**Faktisk resultat**	**Vurdering (OK/FAIL)**
1.	Brugeren opstarter ”Home Protection” software på sin computer	Programmet opstartes	Der åbnes et konsolvindue med start menu	OK
2.	Brugeren vælger ”Vælg mode 2”	Programmet udskriver ’Mode 2 initieres’ Vent Programmet udskriver ’Mode 2 aktiveret’	Programmet udskriver ’Mode 2 initieres’ Programmet udskriver ”Waiting” og ”Bit send”	OK (Med forbehold)

			Programmet udskriver 'Mode 2 aktiveret'	

Use Case under test:		Aktiver mode 2		
Scenarie:		Deaktivere mode 2		
Prækondition:		Systemet er funktionelt og tilgængeligt.		
Postkondition:		Mode 2 er deaktiveret		
Nr.	**Handling**	**Forventet resultat**	**Faktisk resultat**	**Vurdering (OK/FAIL)**
1.	Brugeren opstarter "Home Protection" software på sin computer	Programmet opstartes	Der åbnes et konsolvindue med start menu	OK
2.	Brugeren vælger "Deaktiver Home Protection"	Programmet fremviser besked "Home Protection deaktiveret"	Programmet udskriver 'Home Protection is deactivating' Programmet udskriver "Waiting" og "Bit send" Programmet udskriver 'Home protection is deactivated' Start menu fremvises	OK (Med forbehold)

11.3 Use Case 3:

Use Case under test:		Indstil mode 2		
Scenarie:		Indstilling af mode 2		
Prækondition:		Systemet er funktionsdygtig og tilgængeligt		
Postkondition:		Mode 2 er indstillet efter ønske		
Nr.	**Handling**	**Forventet resultat**	**Faktisk resultat**	**Vurdering (OK/FAIL)**
1.	Brugeren opstarter "Home Protection" software på sin computer	Programmet opstartes	Der åbnes et konsolvindue med start menu	OK
2.	Brugeren vælger 'Indstil mode 2'	Programmet udskriver "Indtast kode på DE2-Board: "	Programmet udskriver "Enter code on DE2-board"	OK (Med forbehold)
3.	Brugeren indtaster koden på DE2-Board	Programmet udskriver	Programmet udskriver	OK (Med forbehold)

		menuen 'Indstillinger til mode 2'	"Change your settings for mode two"	
4.	Brugeren vælger hvilket modul der skal tændes og hvor lang tid der skal gå før det tændes.	Programmet udskriver "Modul tændes: __:__"	Programmet spørger om hvad tid dimmer skal tænde, dimmer skal slukke, dimmer intensitet, switch skal tænde, switch skal slukke	FAIL
5.	Brugeren vælger hvilket modul der skal slukkes og hvor lang tid der skal gå før det slukkes.	Programmet udskriver "Modul slukkes: __:__ " Programmet udskriver "Indstillinger gemt i mode 2"	Programmet udskriver 'Mode 2 is now changed'	FAIL

Use Case under test:		Indstil mode 2		
Scenarie:		Deaktiverer mode 2		
Prækondition:		Systemet er funktionelt og tilgængeligt.		
Postkondition:		Mode 2 er deaktiveret		
Nr.	**Handling**	**Forventet resultat**	**Faktisk resultat**	**Vurdering (OK/FAIL)**
1.	Brugeren opstarter "Home Protection" software på sin computer	Programmet opstartes	Der åbnes et konsolvindue med start menu	OK
2.	Brugeren vælger " Deaktiver Home Protection"	Programmet fremviser besked "Home Protection deaktiveret"	Programmet udskriver 'Home Protection is deactivating' Programmet udskriver "Waiting" og "Bit send" Programmet udskriver 'Home protection is deactivated'	OK (Med forbehold)

Use Case under test:		Indstil mode 2		
Scenarie:		Brugeren taster forkert kode 3 gange		
Prækondition:		Systemet er funktionelt og tilgængeligt.		
Postkondition:		Programmet låses, indtil der trykkes reset		
Nr.	**Handling**	**Forventet resultat**	**Faktisk Resultat**	**Vurdering (OK/FAIL)**
1.	Brugeren opstarter ”Home Protection” software på sin computer	Programmet opstartes	Der åbnes et konsolvindue med start menu	OK
2.	Brugeren vælger ’Indstil mode 2’	Programmet udskriver ”Indtast kode på DE2-Board ”	Programmet udskriver ”Enter code on DE2-board”	OK (Med forbehold)
3.	Brugeren indtaster koden forkert 3 gange på DE2-Board	Programmet fremviser besked ”Programmet er låst”	Programmet låses ikke.	FAIL

*”OK (Med forbehold)” betyder, at det forventet resultat stemmer overens med det faktiske resultat, dog, er der forskel på teksten programmet fremviser via UI.

11.4 Ikke-funktionelle krav

Grundet karantænen har der ikke været fuld mulighed for at teste alle ikke-funktionelle krav. Der er derfor en række af kravene der ikke er blevet godkendt på baggrund af at de ikke kunne testes. F.eks har det ikke været muligt at teste REQ 2.1P, da der ikke kan implementere SW på HW. Ligeledes er REQ 2.3P ikke testet, idet det ikke har været muligt at realisere HW.

Af de dele der kunne testes, ses det, at ikke alle de ikke-funktionelle krav er opfyldt. REQ 2.3U er afhængig af muligheden for at kunne skifte kodeord, hvilket aldrig blev implementeret. Den implementerede struktur for DE2-Boardets SW gør også, at REQ 2.2U ikke vil være opfyldt.

Dog kan bl.a. REQ 2.1U og REQ 2.4P godkendes, da der sagtens har kunnet teste separate dele af softwaren på Arduino. Der er også en række af krav som ikke er målbare, idet de er uden for tidshorisont. For at se den fulde tabel af de ikke-funktionelle krav og deres accept test henvises til Kravsspecifikation og Accepttest dokumentet[55], afsnit KA3.2 i projektmappen.

[55] Ref[K1]

12. Diskussion

Som det ses ud fra resultaterne af accepttesten, er store dele af funktionaliteten ikke opnået. En del af disse nedsættelser skyldes effekten af karantænen, som har afskåret os fra alle former for integrationstest.

Lignende er der features, som er blevet udeladt fra det endelige produkt grundet tidsmæssige begrænsninger. Grundelementerne der skal sørge for at de forskellige Arduinoer kan kommunikere med hinanden blev prioriteret højest, og mindre essentielle funktioner som f.eks. løsning af systemet efter tre forkerte kodeord, blev en eftertanke igennem udviklingen.

Systemet viste sig at være struktureret anderledes end forventet, og implementeringen af disse funktioner ville drastisk have nedsat arbejdet der kunne lægges i andre funktioner. Herved er nogle af funktionerne og implementeringer blevet lavet anderledes end de oprindeligt blev udtænkt i projektets begyndelsesfase. Systemet overholder de resultater nedsat i problemformuleringen, der vha. simuleringer er blevet underbygget, at systemet ville kunne indstilles til to forskellige modes, hvor den ene kan indstilles af brugeren.

Derudover understøtter simuleringerne at der ville være tre funktionelle moduler. Disse moduler er opbygget af delmoduler, der ud fra test og simuleringer viser sig at fungere separat. Dette er det tætteste vi kommer på at kunne teste hele systemet, hvilket betyder, ud fra vores kendskab til systemet, at systemet teoretisk set burde virke. Dette er blevet underbygget af den konceptuelle fremgangsmåde der er blevet brugt igennem hele projektet.

13. Konklusion

I projektforløbet er produktet 'Home Protection' forsøgt udviklet. Produktet tilbyder et indbrudsforbyggende system hvortil det besidder forskellige funktionaliteter. Heriblandt er Mode 1, som er et præ-indstillet program, i stand til at afbillede tilstedeværelse af personer i huset. Mode 2 har samme funktionalitet, men er i stedet et bruger-defineret program.

Det kan konkluderes, at det ikke var muligt at lave en funktionel prototype af produktet, da det i realiteten ikke kunne bygges eller testes. Hertil kan det dog konkluderes, at alt software og hardware er blevet testet så vidt muligt, ved hjælp af simuleringer og redskaber, som var tilgængelig hjemme hos projektmedlemmerne. Vha. simuleringerne kan der påpeges, at et funktionelt produkt kunne blive udviklet. Dette er dog med et forbehold på en ren teoretisk tilgang, da en implementering mellem software og hardware ikke har været muligt og en eventuel fejlfinding ikke kunne fremføres.

Det kan konkluderes, at selvom produktet er udviklet primært teoretisk, at der i projektarbejdet taget højde for en eventuel fremtidig realisering. Det kan desuden konkluderes, at en eventuel realisering af Home Protection ville være økonomisk fordelagtig, da der er blevet påvist at produktet kan bygges ud fra billige komponenter.

Projektarbejdet er opbygget omkring ASE-modellen og er blevet gennemført derudfra. Projektarbejdet er blevet kørt med faste ugentlige møder, både projektgruppen alene og med vejleder. Projektgruppen har haft en faglig udvikling, hvor de individuelle medlemmer har haft hvert deres ansvarsområde at fokusere på.

Den sociale udvikling har ikke været en stor faktor i dette projektforløb. Dette betyder, at kommunikation mellem gruppemedlemmerne er blevet påvirket, da man ikke har siddet sammen om et bord og arbejdet fælles på projektet og kunnet spørge hinanden til råds. Grundet den unikke situation har det været nødvendigt at ty til andre midler i form af online arbejde. Derfor kan det konkluderes, at gruppen har opnået en stor læring i at arbejde online og dermed læring i koordination og planlæggelse af projektarbejde på en alternativ måde.

Projektarbejdet har været anderledes end hvad projektgruppen forventede, og derfor har der været en tilpasningsperiode, hvor det har været nødvendigt at omvende, hvordan arbejdet skulle foregå. Grundet dette, så er der opnået en stor udvikling i alternativ kommunikation og samarbejde.

14. Fremtidigt arbejde

Da projektet er blevet påvirket af Corona krisen, er der nogle funktioner der er blevet sorteret fra. Der er blevet prioriteret, at systemet, ud fra simuleringer og test, skulle kunne fungere lavpraktisk. Fremtidigt arbejde på projektet vil være fokuseret på udvidelse af programmets brugervenlighed.

Dette skulle være i forbindelse med WI-FI, en mobil app og en simpel GUI. Ved WI-FI signalet, vil brugeren kunne tilgå programmet nemmere og bruge eksterne enheder til kontrol over programmet, som for eksempel ved brug af en mobil app, dette vil betyde at brugeren nemt kan tilgå sit 'Home Protection' system udefra, herved kan hjemmets sikkerhed øges når som helst.

Programmet skulle da have en brugervenlig GUI, der skulle kunne arbejde tydeligt sammen mellem computerappen og mobilappen. En yderligere funktion til programmet ville være en logbog, der skulle kunne fortælle brugeren og udvikleren om systemets funktionalitet, drift og eventuelle fejl. Herved kan der holdes styr på systemet, f.eks. logbog over ændringer af mode 2.

En anden brugbar udvidelse til UI'en kunne være en mulighed for at fremvise nuværende indstilling for mode 2. Programmet kunne yderligere udvikles til at have adgang til flere moduler, i.e. flere lyskilder.

15. Referenceliste

Ref [i1]: Rebensdorf, Jens - "**Det er aldrig »bare« et indbrud**" Berlingske Artikel:
https://www.berlingske.dk/samfund/det-er-aldrig-bare-et-indbrud

Ref [i2]: ritzau – "**95 ud af 100 indbrud bliver aldrig opklaret**" Politikken Artikel:
https://politiken.dk/indland/art7190594/95-ud-af-100-indbrud-bliver-aldrig-opklaret

Ref [i3]: TESTSEKTIONEN.DK – EKSPERTBLOGGER - "**Tyverialarm Test – Den bedste sikkerhed for hele familien**":
https://testsektionen.dk/tyverialarm-test/

Ref [i4]: Arlo (Netgear) website:
https://www.arlo.com/en-us/default.aspx

Ref [i5]: TrygFonden og Det Kriminalpræventive Råd (DKR) – "**HVAD VIRKER? VIDEN OM INDBRUD OG INDBRUDSFOREBYGGELSE I PRIVATE HJEM I DANMARK**":
https://www.dkr.dk/media/9834/hvad-virker-viden-om-indbrud.pdf

Ref [D1]: \Dokumentation\SW\Software Dokumentation.pdf

Ref [D2]: \Dokumentation\HW\Hardware Dokumentation.pdf

Ref[K1]: \Dokumentation\Kravsspecifikationer og Accepttest\ Kravsspecifikation og Accepttest.PDF

Ref[P1]: \Dokumentation\Bilag\Process Rapport.PDF

Ref[S1]: \Dokumentation\Bilag\Samarbejdsaftale.PDF

Ref[T1]: \Dokumentation\Bilag\Tidsplan

Ref[L1]: \Dokumentation\Bilag\Logbog

Ref[BS1]: \Dokumentation\HW\HW\Arikitektur\Block og signal beskrivelse.PDF

Ref [G1]: \Dokumentation\Bilag\PDF-filer\Development Processes.PDF

Ref [G2]: \Dokumentation\Bilag\PDF-filer\AN236_ApplicationNote.PDf

Ref [G3]: Loflin, Lewis – "**Improved AC Zero Crossing Detectors for Arduino**" (Zero Crossing inspiration):
http://www.bristolwatch.com/ele2/zcnew.htm

Ref [G4]: ElectronicsTutorials – "**Passive High Pass Filter**"
https://www.electronics-tutorials.ws/filter/filter_3.html

Ref [G5]: Lesurf, Jim "**The Envelope Detetector**":

https://www.st-andrews.ac.uk/~www_pa/Scots_Guide/RadCom/part9/page2.html

Ref [G6]: \Dokumentation\Bilag\PDF-filer\Standard Bodeplot.PDF

Ref [G7]: Dr. Ryan Ahmed - "**Simulink Tutorial**":

https://www.youtube.com/watch?v=vxzR3W2BcRk

Ref [G8]: \Dokumentation\Bilag\PDF-filer\1B4B2.PDF

Ref [G9]: \Dokumentation\Bilag\PDF-filer\CNY17F.PDF

Ref [G10]: \Dokumentation\Bilag\PDF-filer\Arduinomega2560.PDF

Ref [G11]: \Dokumentation\Bilag\PDF-filer\Standard_ decade_value_table.PDF