

X.10 – sikkerhedssystem

Systemarkitektur

Gruppe nr.: 1

Semester projekt 2

Aarhus Universitet: Ingeniørhøjskolen Katrinebjerg

Vejleder: Lars Mandrup

Navn	Studienummer
Emil Munksø Jørgensen	201505049
Emil Stounberg	201509023
Joachim Nyholm Thomsen	201303815
Jacob L. F. Kurtzhals	201505507
Frederik Brasch	201505863
Mads Christian Rosendahl	201509250
Rasmus Kjær Pedersen	201310849
Morten Fogh Jensen	201409925

Versionshistorik

Version	Ændring
1.0	Dokument oprettet, indsat arkitektur af BDD og IBD.
1.1	Tekst indsat for BDD diagrammer
1.2	Indsat IBD'er med tekst og rettet figurnumre og tekster. Retter overskrifter og rækkefølge heraf. Indsat opsætning af dokument. Indsat SW-Arkitektur for use case 1 (uden tekst).
1.3	UC1+UC2 prosa skrevet.
1.4	Ændring i BDD og IBD Signaltabel Rettelse af applikationsmodeller Prosa skrevet til alt indsat
1.5	Rettelse efter Lars' kommentarer
1.6	Rettelser af diagrammer i alle applikationsmodeller, samt IBD og BDD for UserInterface.

Systemarkitektur

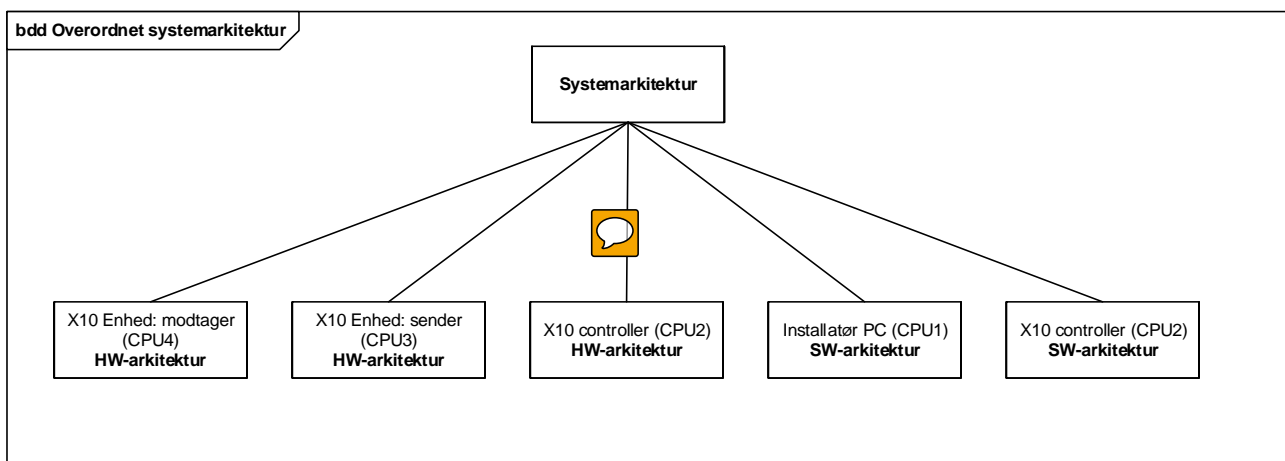
Dette dokument er til beskrivelse af systemarkitektur for sikkerhedssystemet, som formuleret i projektformuleringen og specificeret i kravspecifikationen. Systemet er et sikkerhedssystem der udnytter X.10 kommunikation til at **armere, desarmere og udløsning** af en alarm bestående af en sensor til detektion (udløsning) og en lampe som sættes til at blinke (som det alarmerende element i systemet). For en mere grundig beskrivelse af det overordnede system henvises til systembeskrivelsen i dokumentet "Kravspecifikation".

Formålet med dette dokument er:

- At identificere overordnede komponenter og delsystemer, samt fastlæggelse af deres grænseflader.
- At identificere interne såvel som eksterne komponenter, der avnedes i projektet.
- At identificere arbejdsopgaver for projektet design- og implementeringsfase.

Overordnet Systemarkitektur

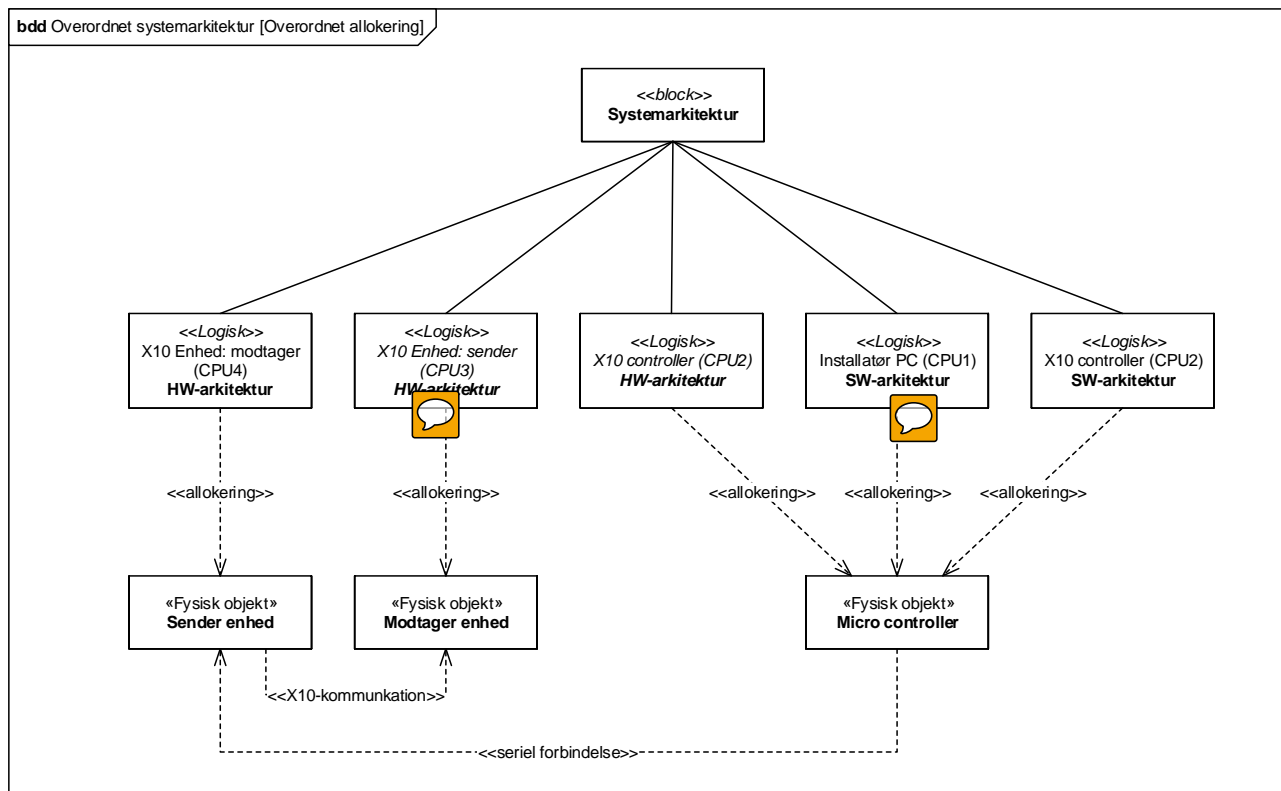
Det overordnede system er beskrevet på Figur 1. Systemet består af i alt 4 CPU'er, hvor én har både en HW-arkitektur og en SW-arkitektur. X10-controllere (CPU2) samt en sender placeres et centralt sted hvor der er mulighed for en aktør handling, som værende armering og desarmering af sikkerhedssystemet. X10-enheder (CPU 3 og 4) placeres respektive tæt på eller inkorporeret i sensor/lampe. Installatør PC (CPU1) skal forbindes til systemet ved konfiguration og intern test af systemet.



Figur 1: Overordnet systemarkitektur, beskriver systemets forskellige CPU'er

Overordnet allokeringsdiagram for systemet

På Figur 2 ses et overordnet allokeringsdiagram. Det ses at den logiske funktionalitet af X10 controllere og installatør PC's software aspekter er allokeret på en microcontroller og bliver styret derfra. X10 enheder logiske funktion er allokeret på henholdsvis en sender enhed og en modtager enhed, som har interne komponenter der bygger denne allokering. Denne interne allokering er beskrevet senere i systemarkitekturen. Kommunikation mellem sender og modtager enhed er ved X10-kommunikation. Der er også en forbindelse mellem microcontrolleren og sender enheden, som blot er serial kommunikation.



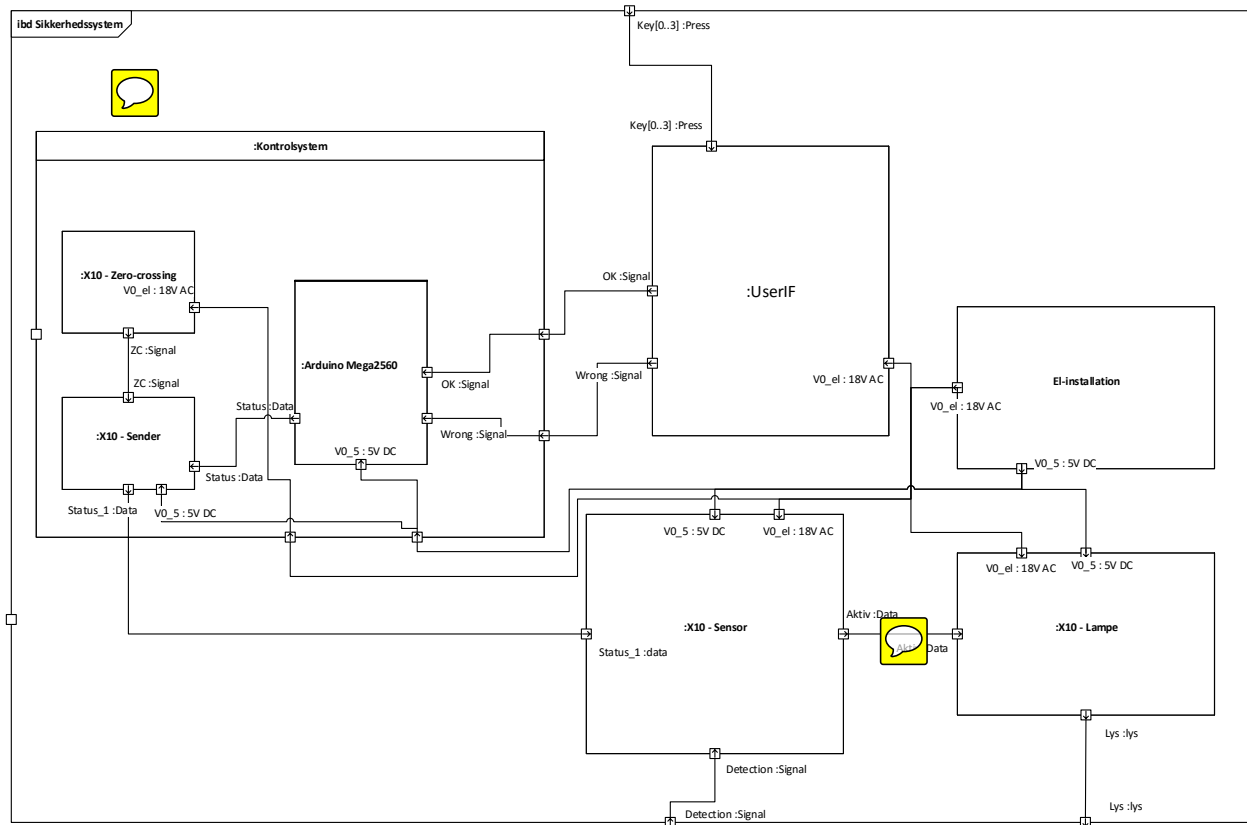
Figur 2: Allokerings BDD for overordnet system.

På Figur 3 ses et generelt BDD for sikkerhedssystemet. Vores system består af en **sensor og lampe** som er de to enheder der skal sende/modtage signaler fra en micro controller om at aktivere eller ej baseret på **X10 – Lampe** og **X10 – Sensor** (som er beskrevet et niveau dybere på Figur 7, Figur 9 og Figur 11). **Kontrolsystem**, fungerer som et mellemlid til kommunikationen mellem **UserIF**, **X10 – lampe** og **X10 – Sensor**. **Zero-crossing** signal er et værktøj til at sende og modtage databits fra micro-controllere over X10. **El-Installation** blokken er en fællesblok til strøm og stel. Det er også her X10 kommunikationen vil blive gennemført.



Side 5 af 35

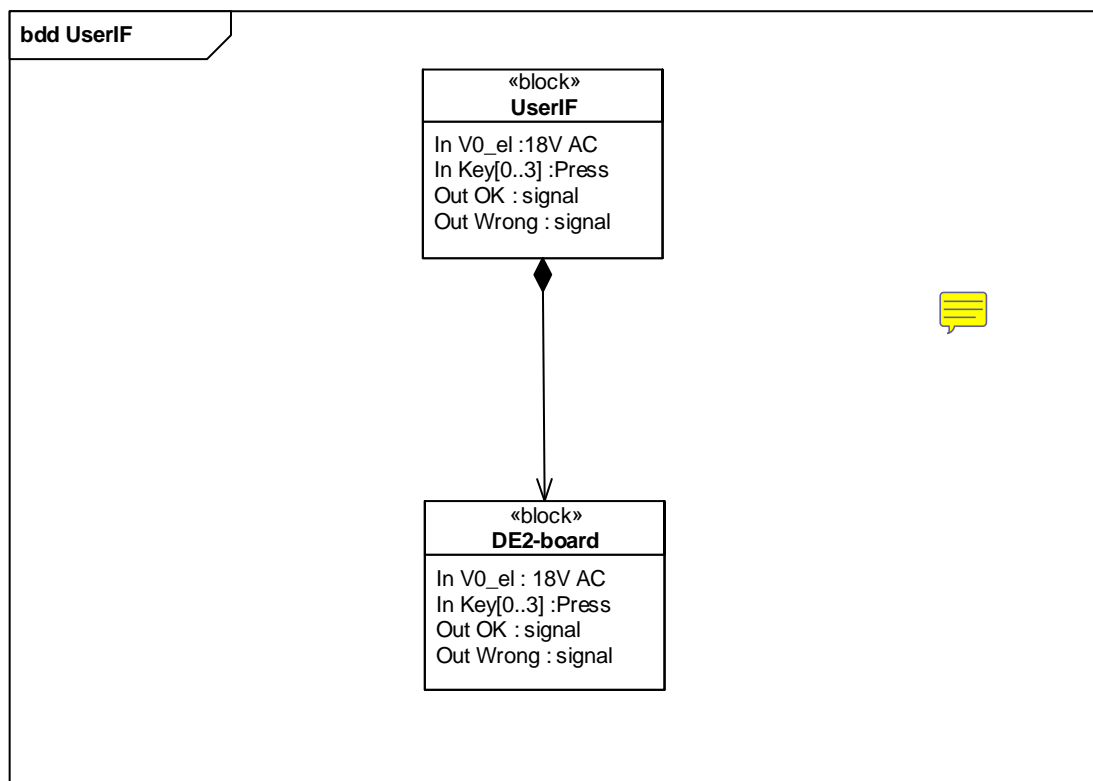
På Figur 4 ses det interne block diagram for det overordnede system. Her ses det logiske flow mellem de interne blokke. **EI-installation** leverer strøm til alle blokke i systemet, enten 18VAC eller 5VDC. **Kontrolsystem** har to inputs, Key :Press og USB :String, der bruges til henholdsvis, armering og desarmering af systemet, og adgangskode til administrator bruger. **UserIF** kommunikerer med **Kontrolsystem** med to signaler, OK og Wrong (betegnelser på om kode fra DE2 board er korrekt eller ej). **Kontrolsystem** sender et kommunikationssignal ud til **X10 sensor**, der tolker om beskeden indeholder om den skal armere eller desarmere. Yderligere sender sensoren baseret på detektion eller ej om **lampen** skal gå i aktiv eller inaktiv tilstand (om alarmen udløses eller ej).



Figur 4: IBD for overordnet system.

Kontrolsystem

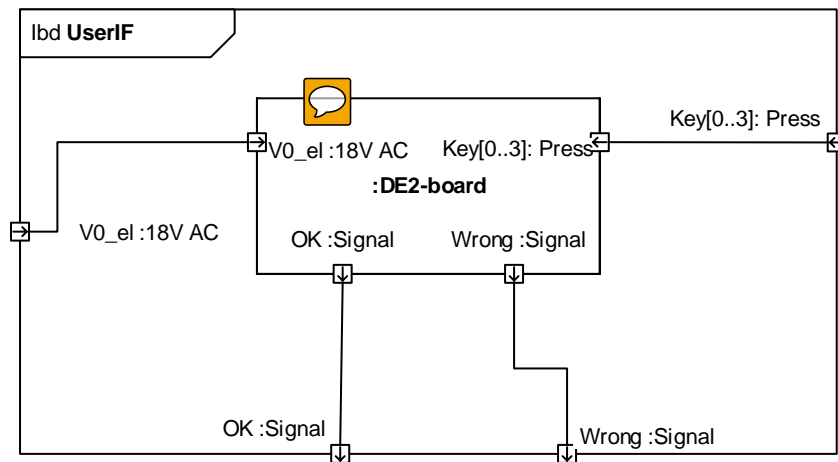
På Figur 5 ses BDD for **UserIF** blokken som ses på Figur 3. **Kontrolsystem** består af en Installatør PC som benyttes til at bl.a. at sætte adresser på X10 enheder op. Et DE2 board som modtager key press fra brugeren og derved sender (baseret på om koden er korrekt eller ej) to signaler ud til Arduino for **Kontrolsystem**, som beskrevet på Figur 4.



Figur 5: BDD for kontrolsystem

Ved DE2-board har brugeren mulighed for indtastning af en kodelås ved KEY[0..3], og den indtastede kode skal valideres af internt i DE2 boardet, men informerer kontrolsystems Arduino om koden er korrekt eller ej.

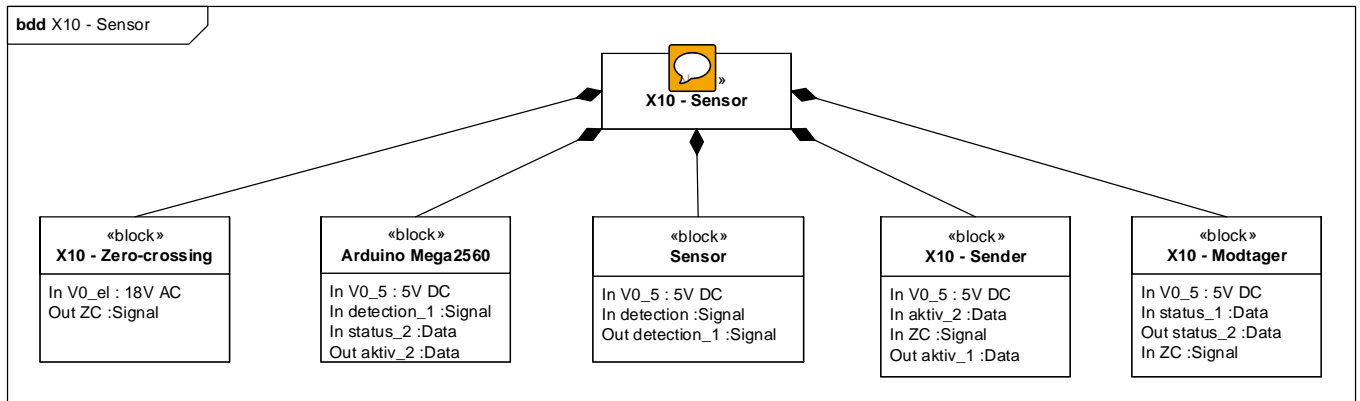
På Figur 6 ses IBD for **UserIF**, der består af et **DE2-board**. **DE2-boardet** bruges som *kodelås* for sikkerhedssystemet. **DE2-board** modtager en kode og validerer koden internt, hvorefter den sender hvorledes det indtastet er korrekt eller forkert.



Figur 6: IBD for Kontrolsystem.

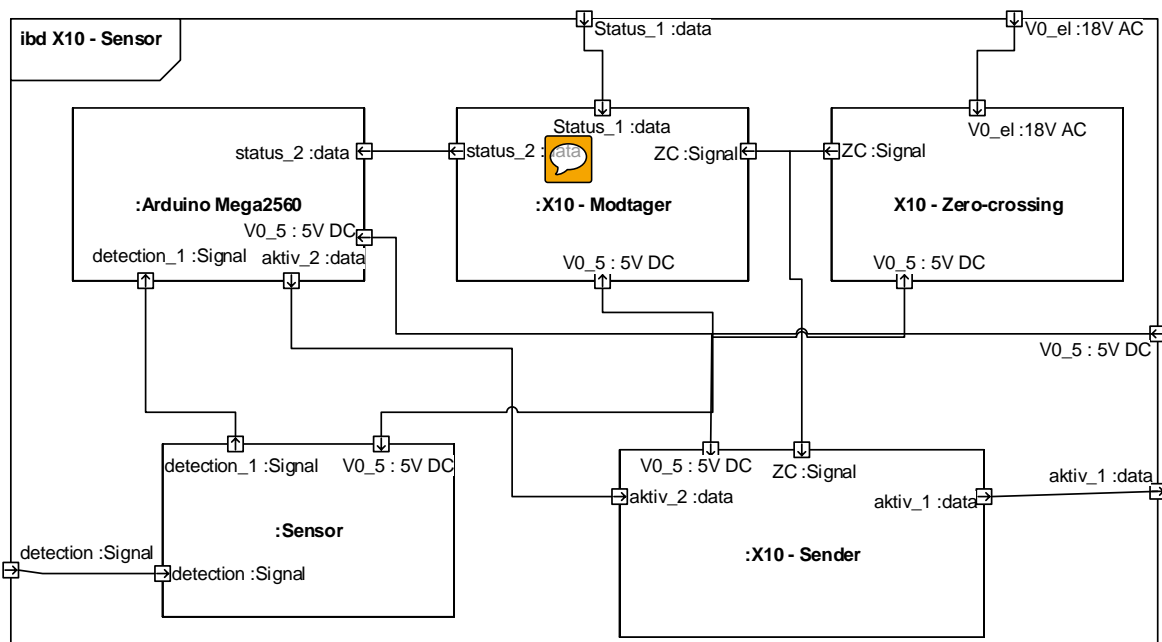
Sensor X10

På Figur 7 ses BDD for **X10 sensormodulet**, dette modul består af en **Arduino Mega2560**, der står for at modtage signaler fra **sensoren** og som sender signal videre til **kontrolsystemet**. Til dette bruger den **X10 sender og modtager**, samt en **Zero-crossing detector** til at detektere når der bliver sendt data til modulet.



Figur 7: BDD for X10 – Sensormodul.

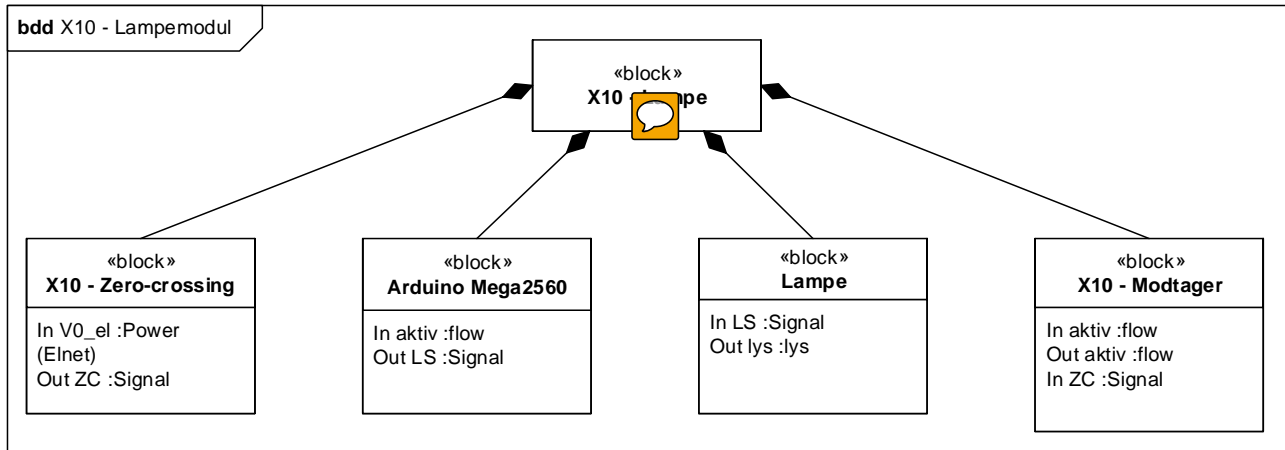
På Figur 8 ses IBD'et for X10 **sensormodulet**. Det ses her hvorledes de interne signaler er forbundet, samt hvilke signaler der bliver sendt ind i og ud af modulet. Se Tabel 1 for signalbeskrivelser.



Figur 8: IBD for X10 – Sensormodul.

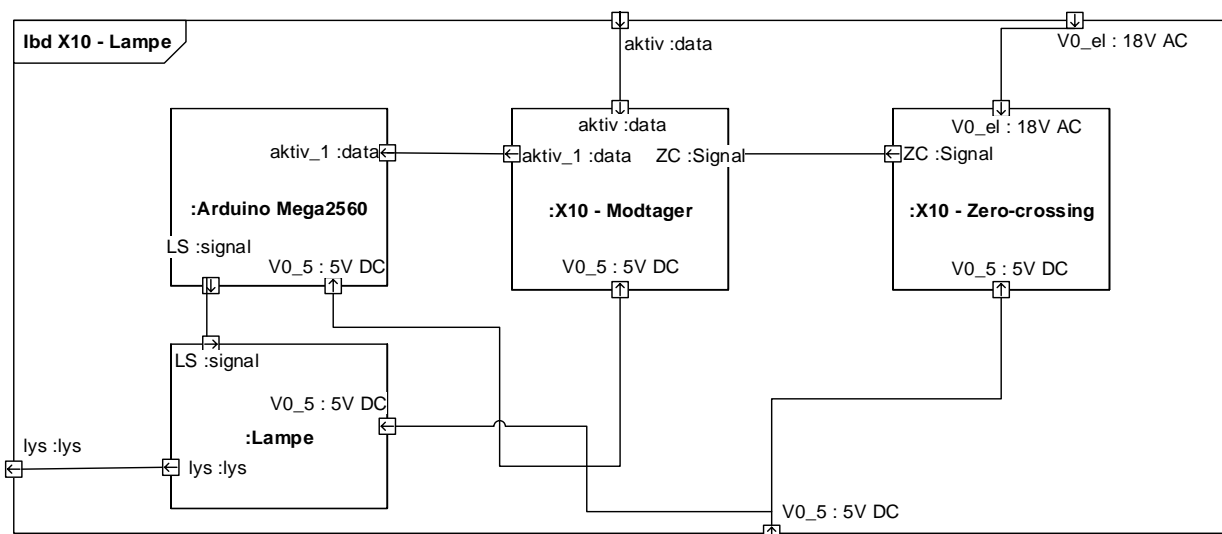
Lampe X10

På Figur 9 ses BDD for **X10 lampemodul**. Dette modul består af samme dele som **X10 sensormodul**, bortset fra, at **lampemodul** ikke består af en **X10 sender**, da denne kun har til formål at modtage besked om at tænde og slukke for sin **lampe**.



Figur 9: BDD for X10 – Lampemodul.

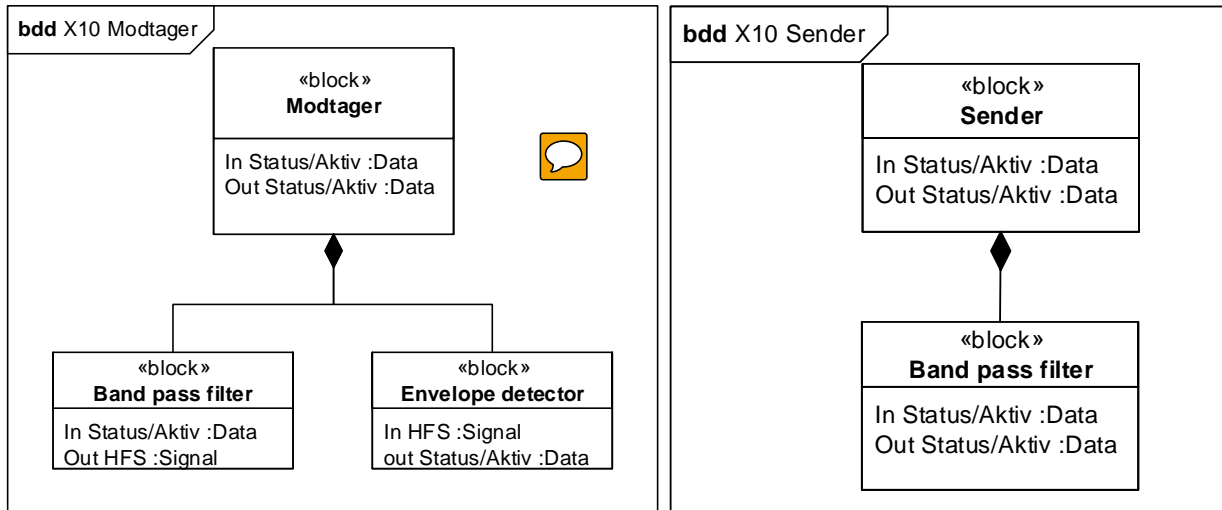
På Figur 10 ses IBD for X10 lampemodulet.



Figur 10: IBD for X10 – Lampemodul.

Sender og modtager

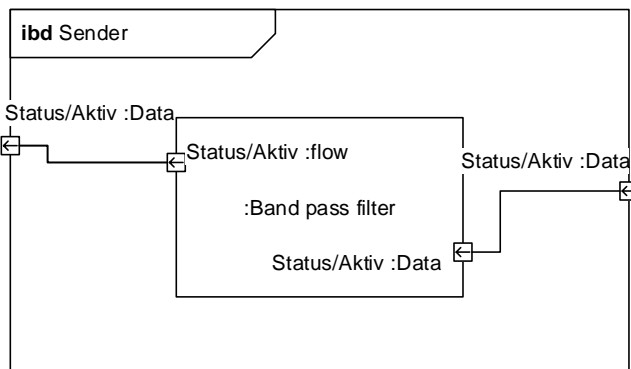
På Figur 11 ses både et BDD for **Modtager** og et BDD for **Sender**. Begge enheder består af **Band pass filter** som sorterer frekvenser således vi kan aflæse beskeder over el-nettet. **Envelope detector** modtager et høj frekvens fra **Band pass filter** og gør det læsbart for en microcontroller.



Figur 11: Til venstre ses BDD for modtager. Til højre BDD for sender.

Sender

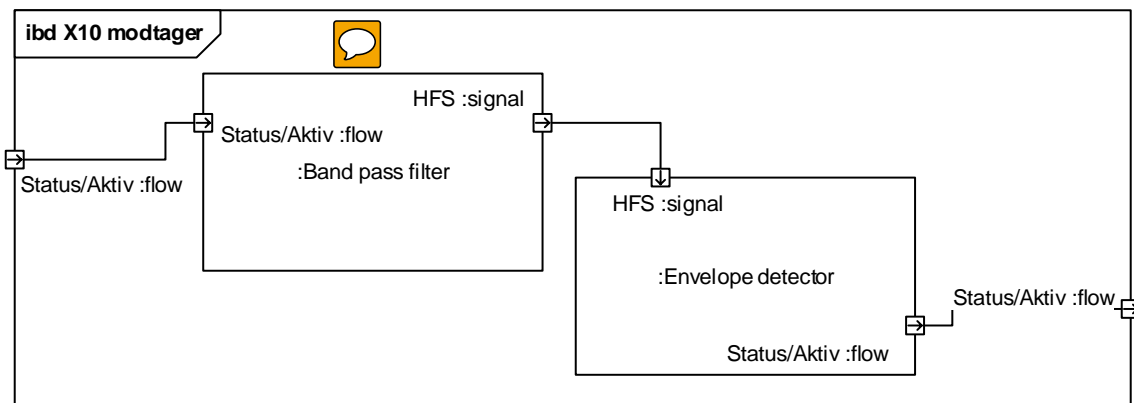
På Figur 12 ses X10 **sender** består af et **Band pass filter**, som kun sender det signal, der skal anvendes over 18VAC nettet.



Figur 12: IBD for X.10 sender.

Modtager

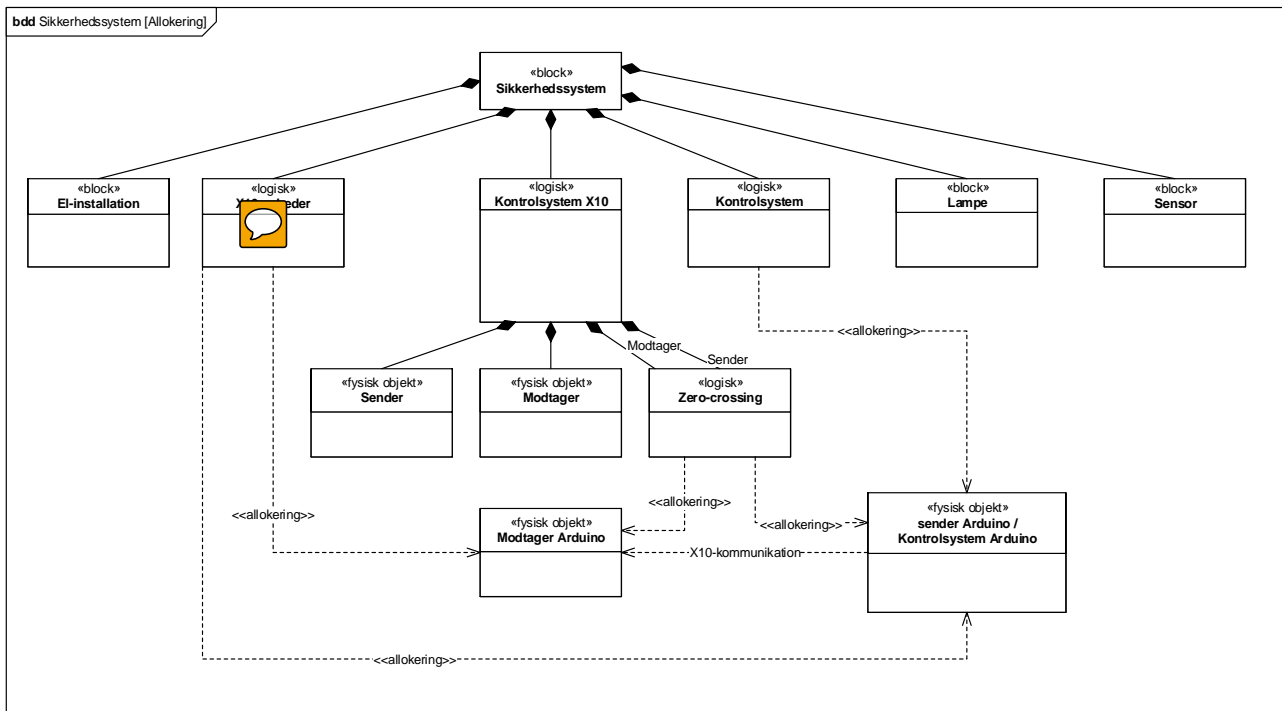
På Figur 13 ses X10 **modtager**, der består af et **Band pass filter** og en **Envelope detector**, som filtrerer de uønskede signaler, og gør data læsbar for næste block.



Figur 13: IBD for X.10 modtager.

Allokering af logisk funktionalitet i sikkerhedssystemet



På figur 14 ses den ovenstående interne allokering mellem sikkerhedssystemets delsystemet. Diagrammet viser allokering af forskellige delsystemer til Arduino, og kommunikationen mellem Arduino.



Figur 14: Allokering for sikkerhedssystem.


Signalbeskrivelser

Følgende beskriver hvilke typer signaler som er mellem de forskellige hardwareblokke.

Signal-navn	Funktionsbeskrivelse	Område 	Port 1 (out)	Port 2 (in)	Kommentar
Status_1	data	bits	X10 - sender	X10 - modtager	Armering desarmering
detection	signal	signal	Omgivelser	Sensor	Sensor registrer bevægelse
aktiv	data	bits	X10-sender	X10 -modtager	
lys	lys	0V – 5V	lampe	Omgivelser	Lamperne blinker
Key[0..3]	Press	bits	Aktør	DE2-board	Koden til armering eller desarmering af alarm
USB	String	bits	Installatør PC	Kontrolsystem	Opsætnings PC
OK	signal	0V – 5V	DE2-board	Arduino mega2560	Key[0..3] OK
Wrong	signal	0V – 5V	DE2-board	Arduino mega2560	Key[0..3] Wrong
V0_5	5V DC	5V ± 0,1V	El-installation	X10 - sender X10 - modtager Sensor Lampe Arduino mega2560 Kontrolsystem	Forsyningsspænding
V0_el	18V AC	18V ± 0,1V	El-installation	X10 - Zero-crossing	
					
status	data	bits	Arduino mega2560	X10-sender	
ZC	signal	0V – 5V	X10 - Zero-crossing	X10-sender	
aktiv_1	data	bits	X10 - modtager	Arduino mega2560	
LS	signal	0V – 5V	Arduino mega2560	Lampe	
Status_2	data	bits	X10 - Modtager	Arduino mega2560	
aktiv_2	data	bits	Arduino mega2560	X10 - sender	
detection_1	signal	0V – 5V	Sensor	Arduino mega2560	
Konfig	signal	0V – 5V	Ude fra	Installatør PC	

Tabel 1: Signalbeskrivelser for BDD'er og IBD'er.

Tabel 2 er en signal beskrivelse af X10 – protokol kommunikation. Disse signaler er gældende for al kommunikation mellem en X10 sender og en X10 modtager. Som det kan ses på tabellen, består en komplet besked af et enable bit, en besked string, og en slutbit.

NAVN	FUNKTION	Antal bits
Enable	Klargøring af signal 0 – Ikke klar 1 – Klar 	2
Besked	Kommunikation mellem arduino gennem elnet. 1111 – Amering 0011 – Desarmering 1100 - Aktivering	4
Slutbit	Stop besked 1 – Ved stop 0 – Ikke stop	2

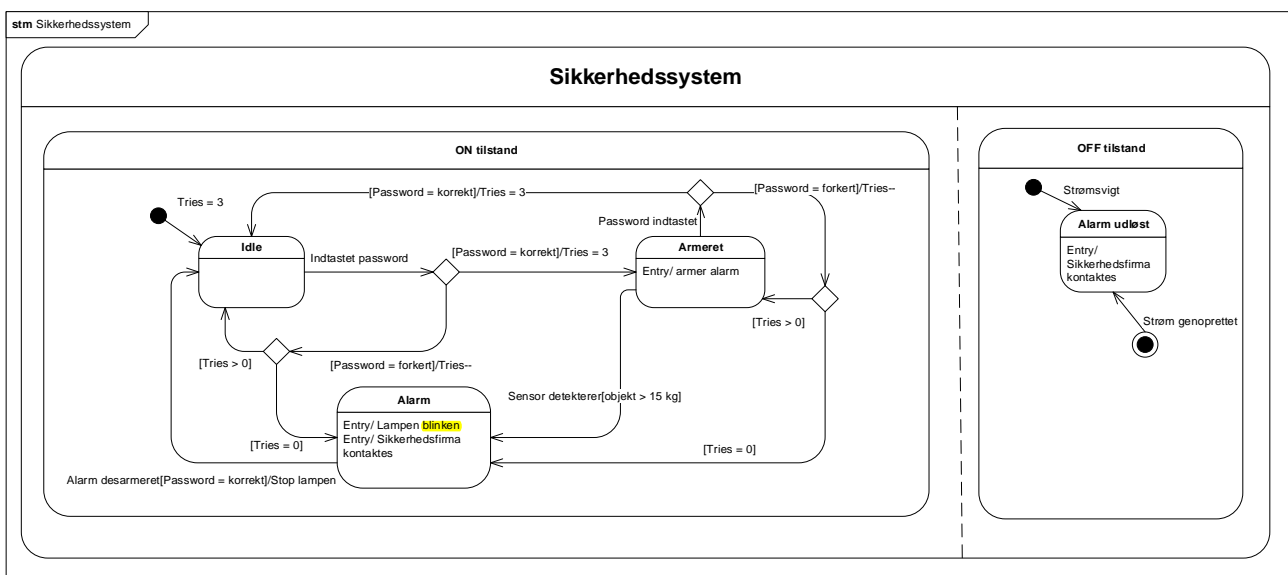
Tabel 2: Tabel over opbygningen af kommandoer, med startbits, besked, og slutbits længde.

Softwarearkitektur

Dette afsnit beskriver softwarearkitektur for sikkerhedssystemet, som formuleret i projektformuleringen og specificeret i kravspecifikationen. Afsnittet giver indblik i hvordan software er specificeret gennem grænseflader, klasseidentifikation, metodeidentifikation og kontrolklasseidentifikation.


STM

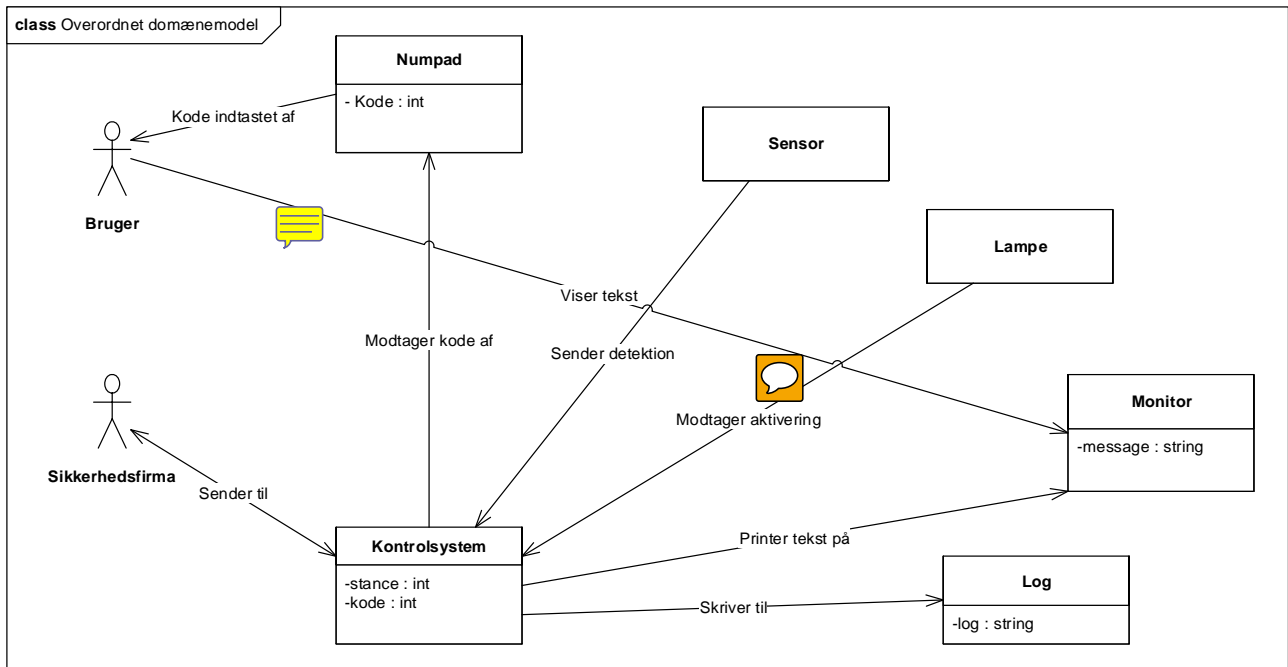
På figur 15 ses en state maskine for den overordnede funktionalitet i systemet. Her kan det ses at systemet kan skifte mellem et idle stadie (som også er desarmeret tilstand), armeret stadie og et udløst stadie. Derudover kan systemet enten være i ON tilstand, som er når systemet er aktiv (strøm til systemet) eller hvis der kommer strømsvigt (ingen strøm til systemet) vil det gå i OFF tilstand og udløse alarmen.



Figur 15: State maskine for sikkerhedssystem.

Domæneanalyse

På Figur 16 ses en overordnet domæneanalyse for sikkerhedssystemet.  er ikke indsat boundary, domæne, eller controller klasser, da dette diagram blot skal symbolisere sammenhængen mellem systemets forskellige aktører og blokke. I diagrammet bliver UC3 i kombination med UC1, og viser et scenarie hvor en bruger armerer, og alarmen bliver udløst og aktiverer lamper samt alarmerer sikkerhedsfirma.



Figur 16: Domæne model for overordnet system.

Software er designet så klasserne specifikke opgaver bortset fra kontrolsystem. Formålet med dette er at give hver klasse højere selvstændighed og mindske koblingen mellem dem. Der defineres klasser ud fra grænseflade- og domæneanalyse. Grænseflader består af de problemer der opstår når der sker kommunikation mellem to dele af systemet. Domæneproblemer er de resterende problemer systemet skal løse, dette er ofte eksterne forbindelser eller håndteringen af parametre og information. Ud fra domæneanalysen er følgende problemer defineret:

- Grænseflader
 1. Brugergrænseflade til brugeren.
 2. Grænseflade til bevægelses sensor.
 3. Grænseflade til lampe(r).
 4. Grænseflade til installatør pc.
 5. Grænseflade til vagtfirma.
- Domæner
 1. Håndtering af begivenheder til log.
 2. Håndtering af de specifikke parametre for armering, desarmering og udløsning.
 3. Håndtering af tilsluttede enheder.

Klasseidentifikation

Hver applikationsmodel har til opgave at vise klasserne, som er involveret i den pågældende use case funktionalitet. Denne funktionalitet håndteres af den respektive kontrol klasse. Grænseflade- og domæneklasserne er bestemt ud fra grænseflade- og domæneproblemerne beskrevet i sidste afsnit. Controller klasser er identificeret ud fra hvilken klasse der håndterer og modtager flest metoder og parametre. Der er identificeret følgende klasser blandt alle applikationsmodeller:

Boundary klasser:

- 1) **Numpad:**
Er switches på DE2-board. Grænseflade for brugeren.
- 2) **Monitor:**
DE2-board monitor. Grænseflade for brugeren.
- 3) **Sensor:**
Består af en bevægelses sensor og X10. Grænseflade for kontrolsystemet.
- 4) **Lampe:**
Består af en lampe og en X10. Grænseflade for kontrolsystemet.

Domæne klasser:

- 1) **Log:**
Håndterer og gemmer tilstandsskift, dato og timestamps for eventuelle tilstandsskift for sikkerhedssystemet. Samme log som bliver skrevet til i alle UC's.

Controller:

- 1) **Kontrolsystem:**
Består af DE2-board og Kontrolsystem X10. Kontrollerer UC1-3.
- 1) **Installatør PC:**
Består af en PC til konfiguration af Kontrolsystemet. Kontrollerer UC4.

Applikationsmodeller

I dette afsnit vil hver applikationsmodel pr. use case blive gennemgået ved først klasseidentifikation, efterfulgt af domæneidentifikation, metodeidentifikation og til sidst kontrolklasseidentifikation. Applikationsmodeller for hver af de 4 use cases fra kravspecifikationen er lavet på basis af sidst læste afsnit.

Klasseidentifikation – Her laves klassediagrammer for hver use case, der beskriver sammenhængen mellem boundary, controller og domæne klasser. Til hver klasse er der også tilføjet metoder og attributter, som kan identificeres i de respektive sekvensdiagrammer i afsnittet ”metodeidentifikation”.

Domæneidentifikation – Denne del af applikationsmodellerne bliver relationen mellem klasserne yderligere defineret. Relationerne viser bl.a. hvilke slags forbindelser der er mellem klasserne, og hvilken retning signalerne har. Det beskrives også her hvilken slags relation der er mellem klasserne. Dette er også første skridt til at identificere metoder til sekvens diagrammer.

Metodeklasseidentifikation – Der er lavet sekvens diagrammer på baggrund af de fundne klassediagrammer. Sekvensdiagrammerne bruges til identifikation af de metoder og attributter, som fuldfører use cases funktionalitet. I sekvensdiagrammer er både hovedscenarie og undtagelser beskrevet. Undtagelser er beskrevet ved alt diagrammer.

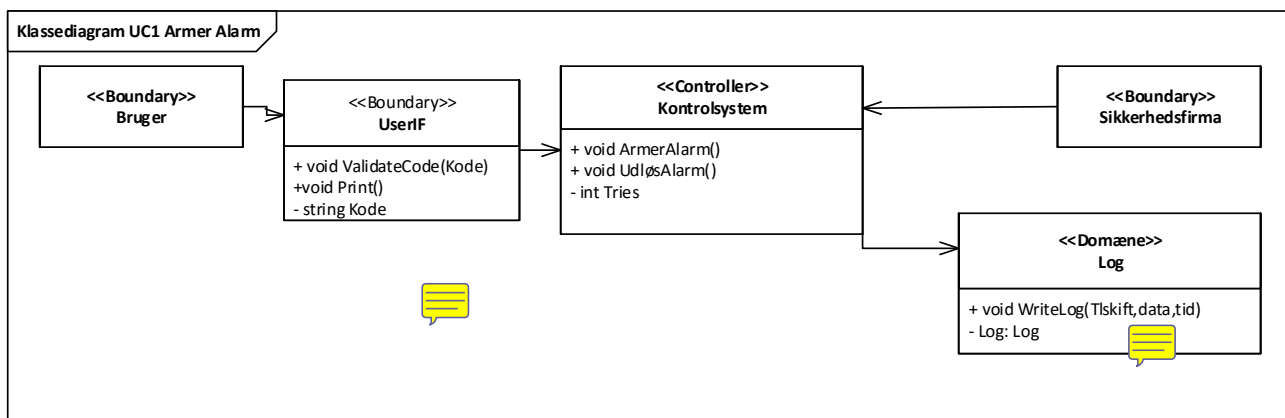
Kontrolklasseidentifikation – For at give yderligere indblik i hvordan kontrol klassen varetager opgaver, beskrives softwaren som et emachine. Formålet med dette er at vise systemets tilstande der forekommer igennem use casens forløb.

Use Case 1: Armér alarm

Klasseidentifikation

Figur 17 viser klassediagrammet for UC1, dette klassediagram er benyttet til at identificere nødvendige klasser til use case 1.

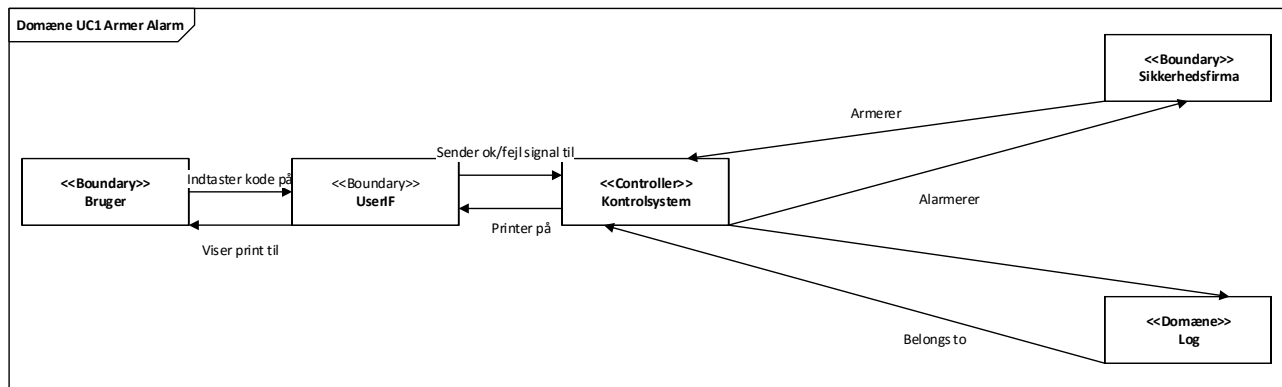
- 1) Kontrolsystem – Kontrollerer sikkerhedssystemer.
- 2) Log – Tilhører kontrolsystemet, og håndterer en datalog for sikkerhedssystemet ændringer af tilstande, udløsninger og konfiguration.
- 3) Monitor – Grænseflade for brugeren, viser evt. information fra kontrolsystemet.
- 4) Taster – Grænseflade for brugeren, til indtastning af kode.



Figur 17: På billedet ses klassediagram for UC1.

Domæneidentifikation

På Figur 18 ses domæne modellen for use case 1.



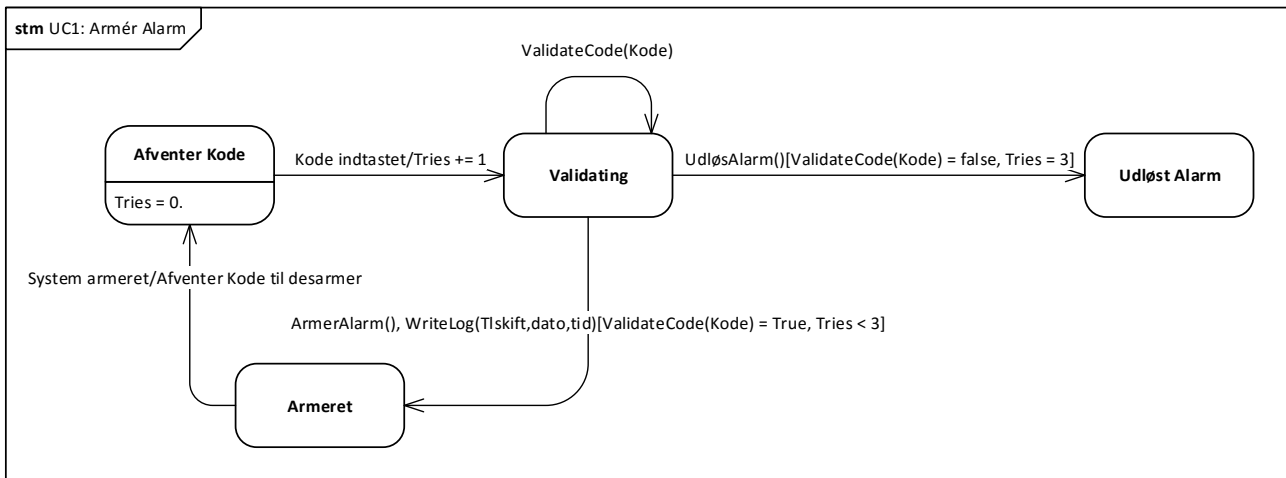
Figur 18: Viser domæne modellen for UC1, og kommunikation mellem forskellige klasser.

Figur 19 viser SD for UC1. På sekvens diagrammet ses den sekventielle gennemgang af UC1. Dette SD er blevet brugt til at identificere de nødvendige attributter, funktioner og stadier.



Kontrolklasseidentifikation

På figur 20 ses det, at systemet vil returnere til afventer kode stadiet efter at være blevet armeret. Dette er fordi stadiet går igen i UC2, da den afventer samme kode for at blive desarmeret.




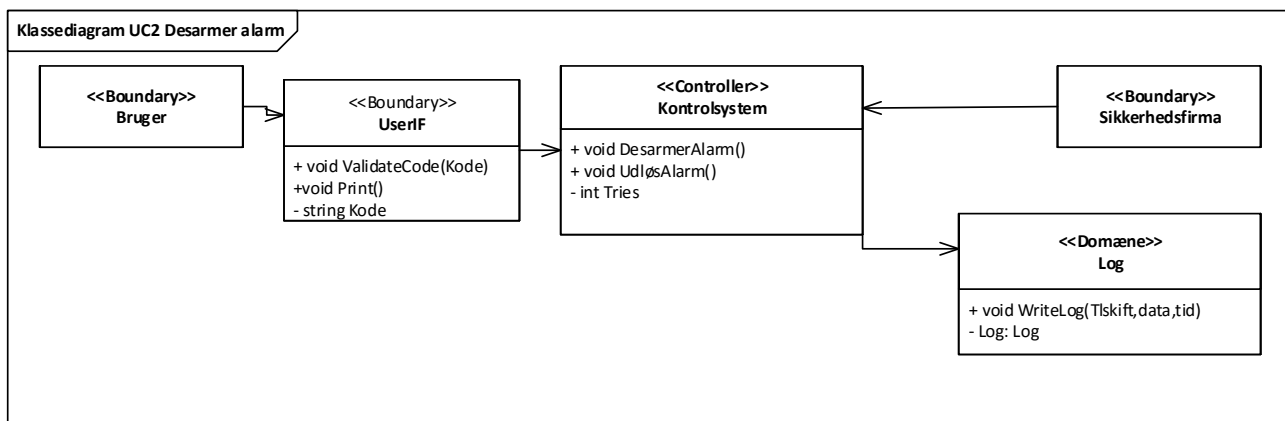
Figur 20: Billedet viser STM for UC1.

Use Case 2: Desarmér alarm

Klasseidentifikation

Figur 21 viser klassediagrammet for UC2, dette klassediagram er benyttet til at identificere nødvendige klasser til use case 2.

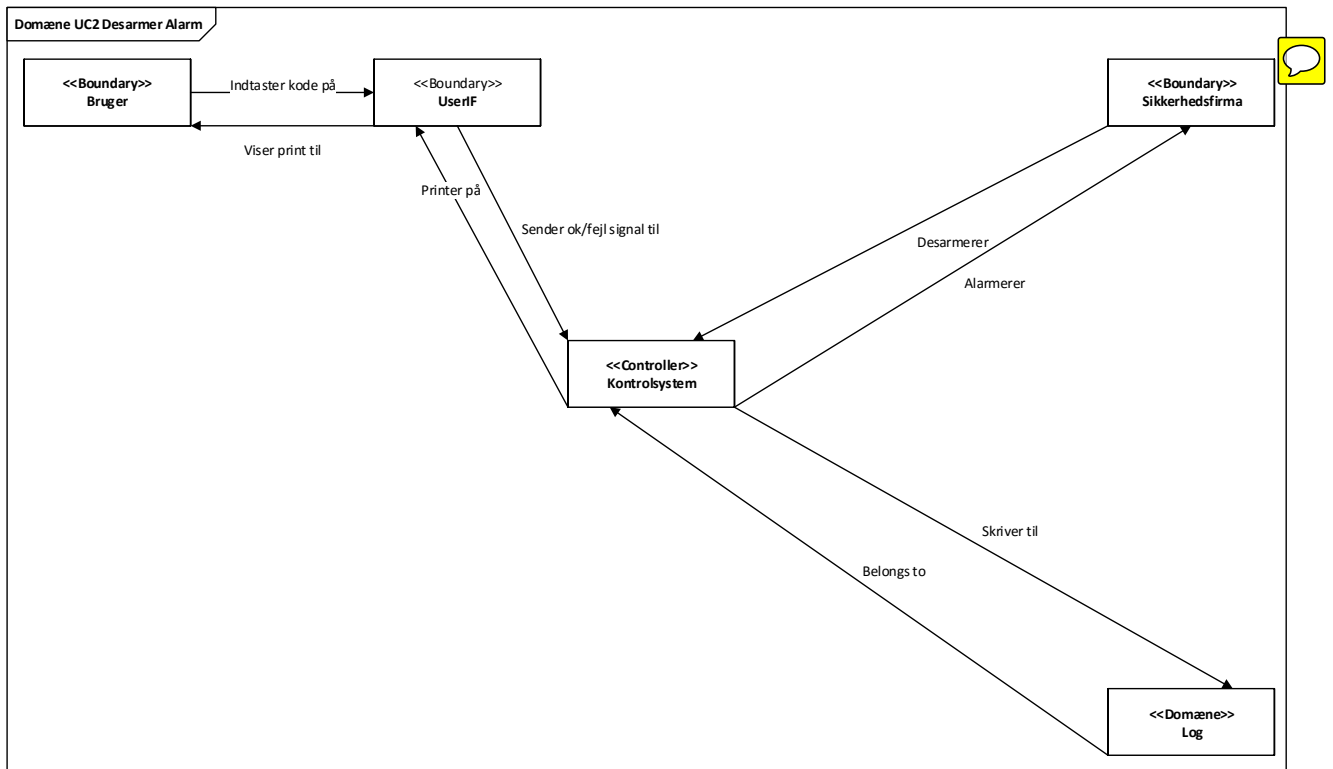
- 1) Kontrolsystem – Kontrollerer sikkerhedssystemer.
- 2) Log – Tilhører kontrolsystemet, og håndterer en datalog for sikkerhedssystemet ændringer af tilstande, udløsninger og konfiguration.
- 3) Monitor – Grænseflade for brugeren, viser evt. information fra kontrolsystemet.
- 4)  Numpad – Grænseflade for brugeren, til indtastning af kode.



Figur 21: På billedet ses klassediagrammet for UC2.

Domæneidentifikation

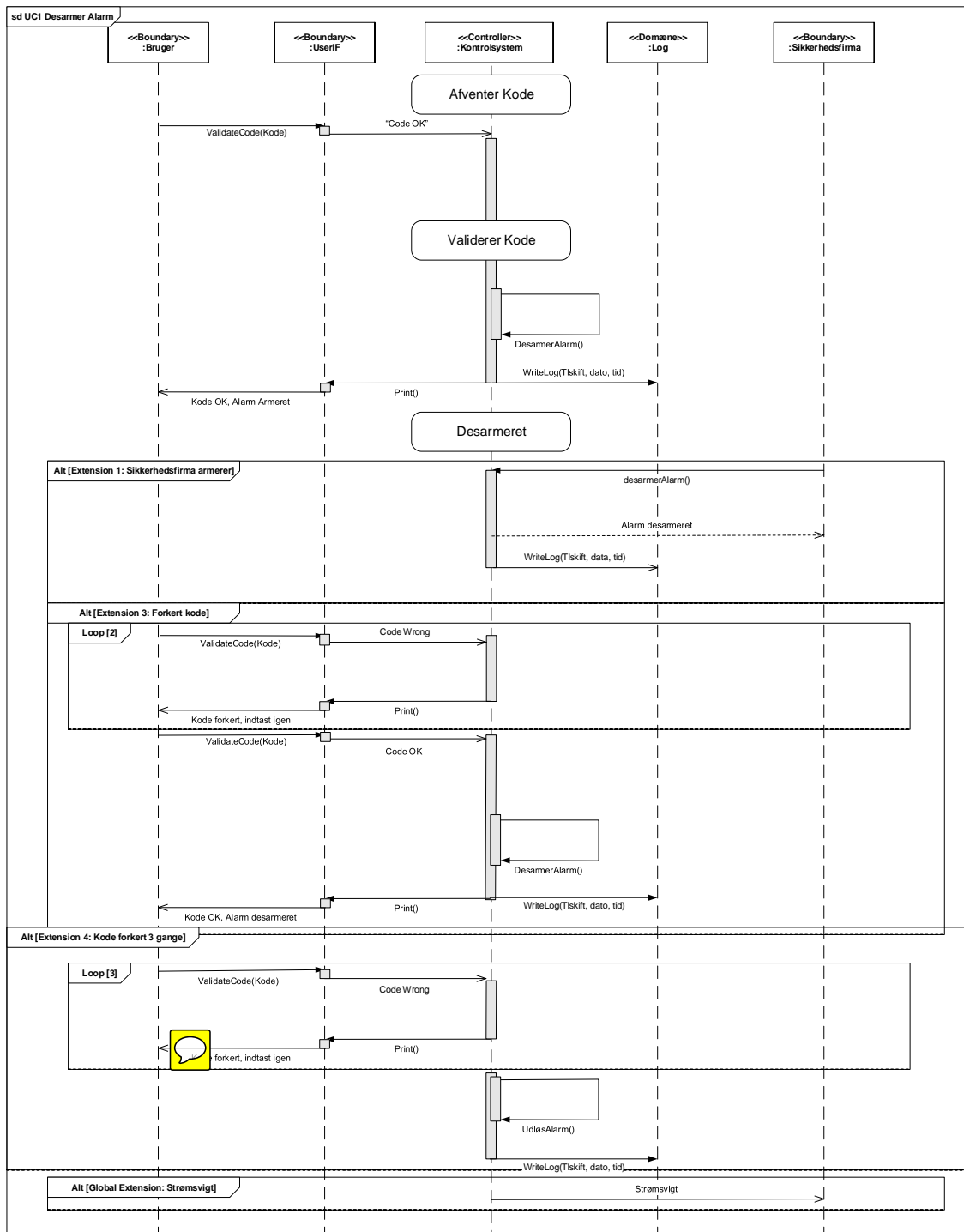
På Figur 22 ses domænemodel for use case 2 "Desarmer alarm".



Figur 22: Viser domæne model for UC2, og kommunikation mellem klasserne i use casen.

Metodeidentifikation

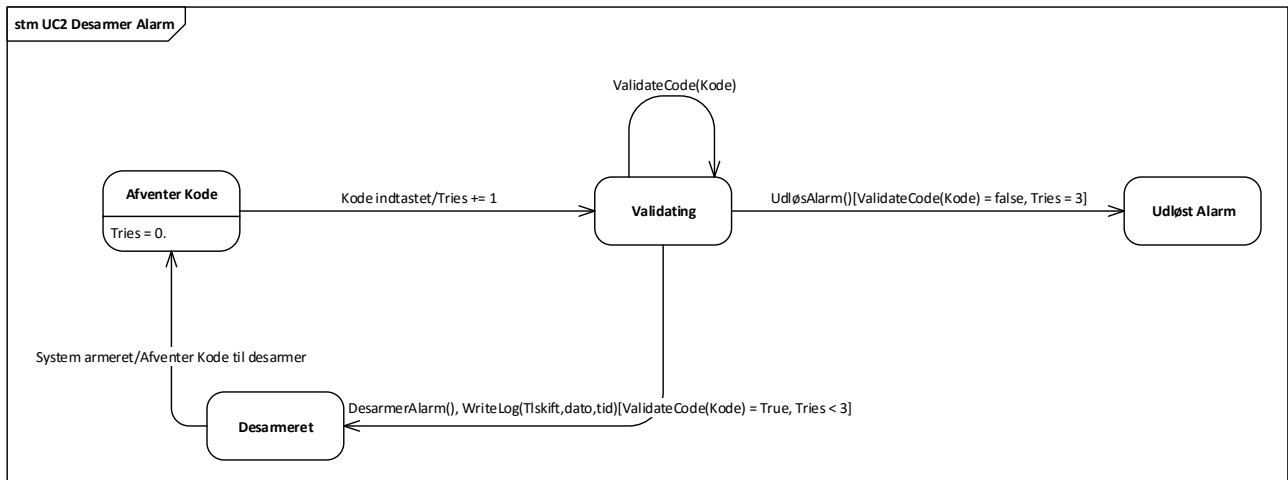
Figur 23 viser SD for UC2. På sekvens diagrammet ses den sekventielle gennemgang af UC2. Dette SD er blevet brugt til at identificere de nødvendige attributter, funktioner og stadier i use casen.



Figur 23: På billedet ses SD for UC2.

Kontrolklasseidentifikation

På figur 24 ses det, at systemet vil returnere til afventer kode stadiet efter at være blevet desarmeret. Dette er fordi stadiet går igen i UC2, da den afventer samme kode for at blive armeret.



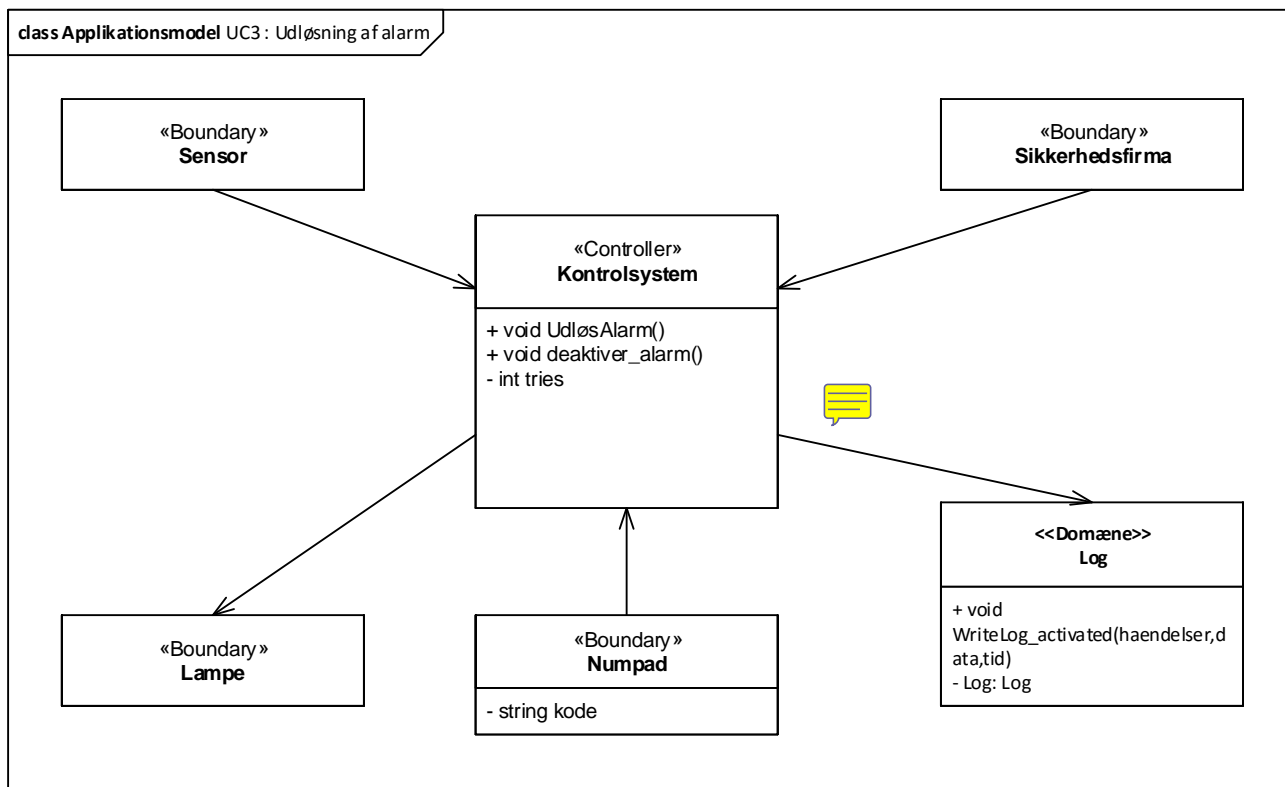
Figur 24: Billedet viser STM for UC2.

Use case 3: Udløsning af alarm

Klasseidentifikation

På figur 25 ses klassediagram for use case 3, "Udløsning af alarm". Nedenfor beskrives kort klassernes ansvar for use case 3:

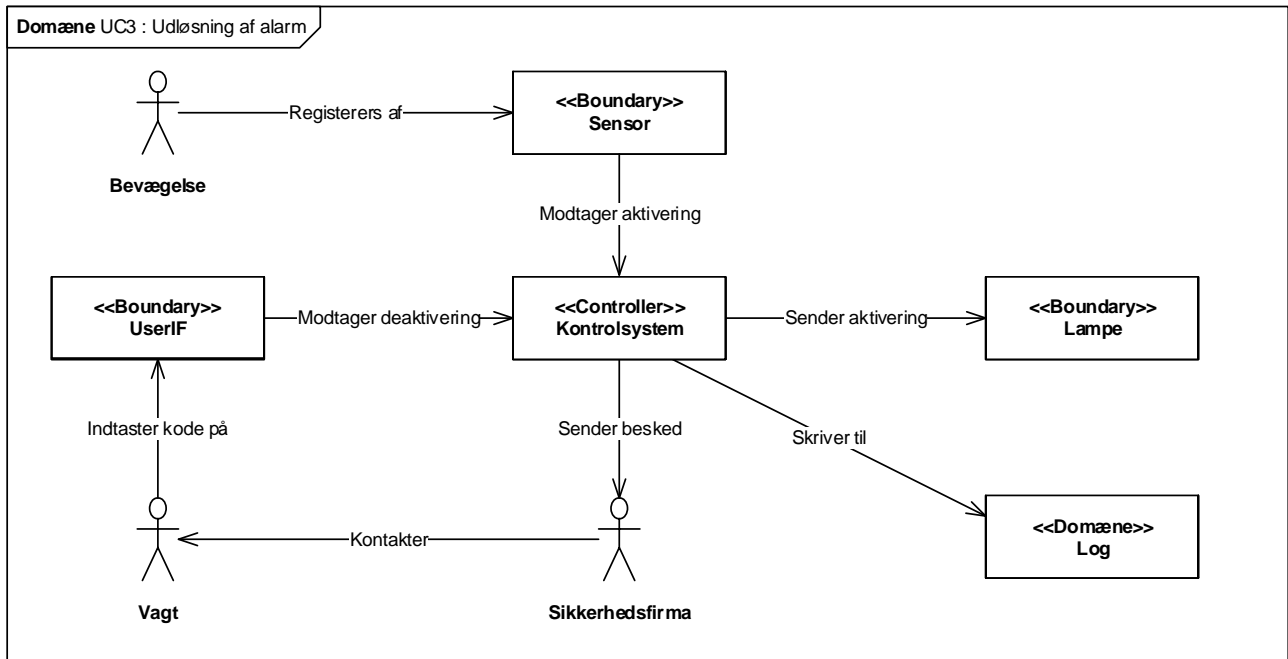
- 1) Lampe – Har til opgave at modtage aktivering fra controller klassen. Modtager deaktivering af controller klassen.
- 2) Sensor – Udsender detektion til **controllersystemet**. Modtager deaktivering fra controller klassen
- 3) Numpad – Interface for indtastning af deaktiveringskode
- 4) Log – Håndterer en data log for sikkerhedssystemet ændringer af tilstande, udløsninger og konfiguration



Figur 25: Klassediagram for use case 3

Domæneidentifikation

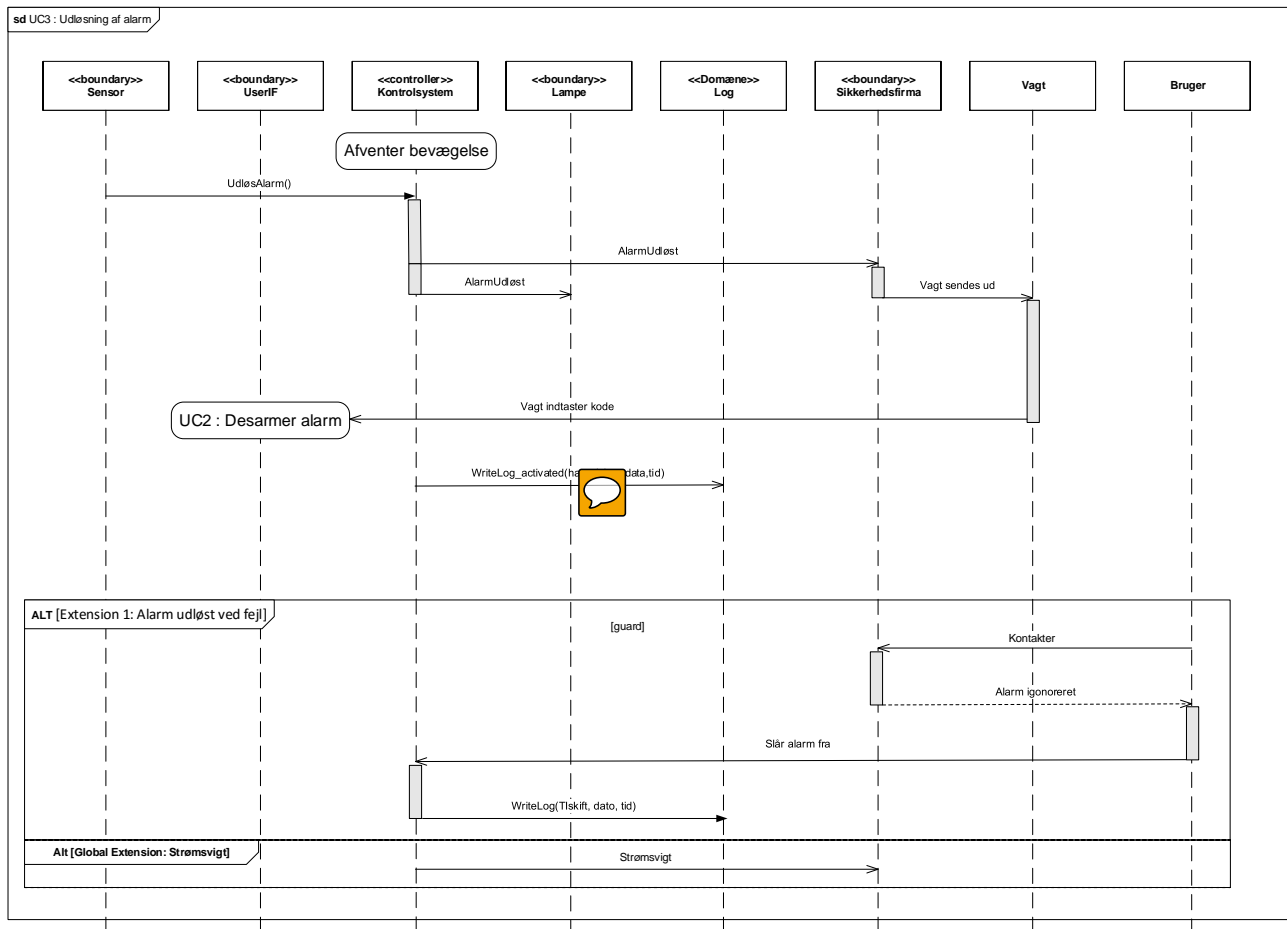
På figur 26 ses domæne model for use case 3 "Udløsning af alarm".



Figur 26 :Domænemodel for use case 3

Metodeidentifikation

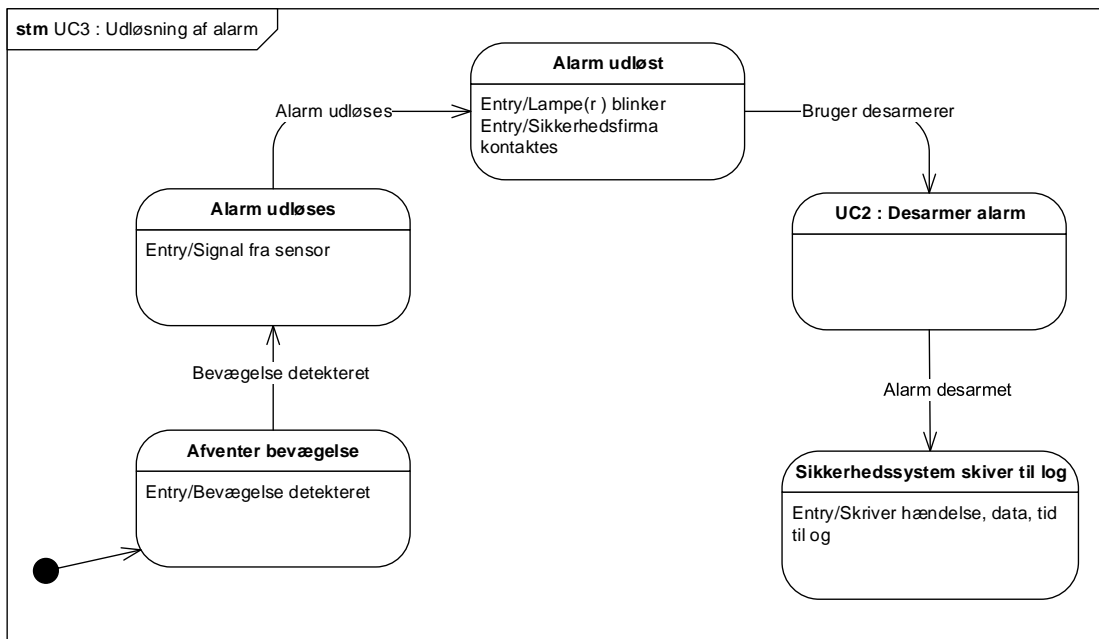
På figur 27 ses sekvens diagram for use case 3.



Figur 27: Sekvens diagram for use case 3

Kontrolklasseidentifikation

På Figur 28 ses det at systemet vil afvente detektion fra sensor, hvorefter alarmen kan blive udløst. Dette vil resultere i at systemet går i udløst tilstand, hvorefter use case 2 "Desarmer alarm" kan blive kørt for at desarmere den udløste alarm. Herefter går systemet i desarmeret tilstand og skriver til log.



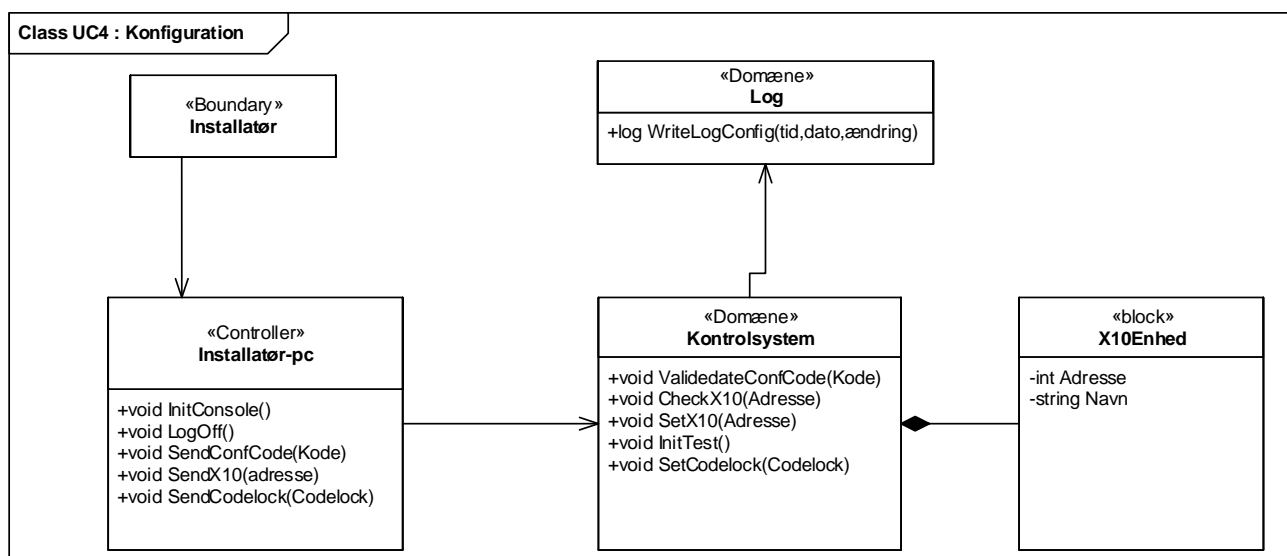
Figur 28: State machine for use case 3

Use case 4: Konfiguration/Test af system

Klasseidentifikation


På figur 29 ses klassediagrammet for use case 4 “Konfiguration/Test af system”. Nedenfor beskrives kort klassernes ansvar for use case 4:

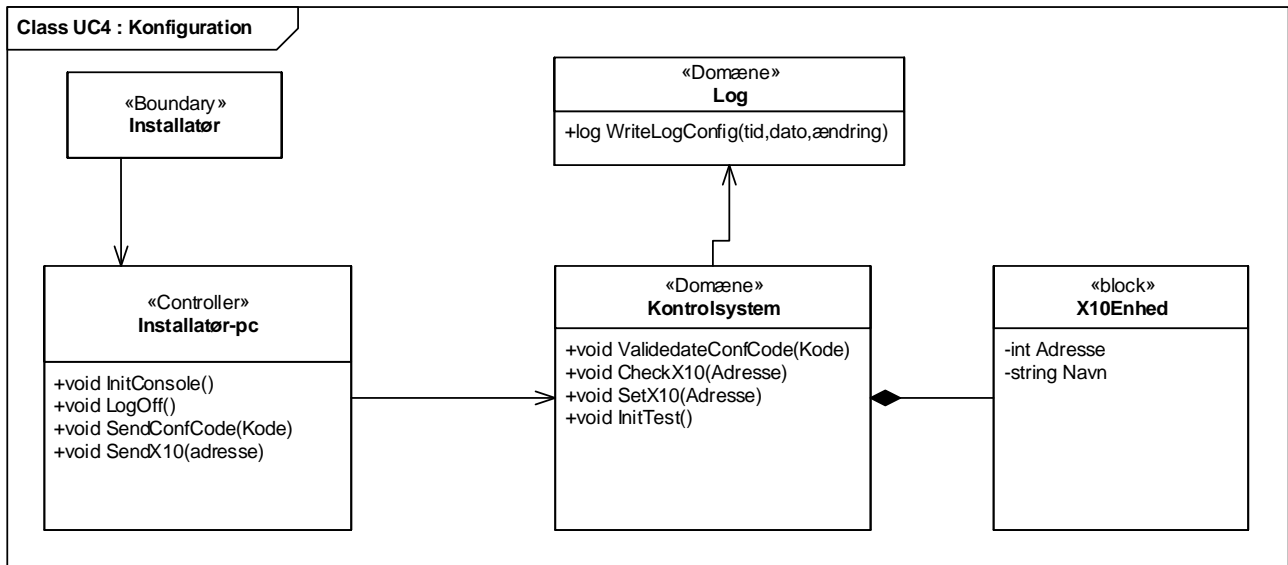
- 1) Installatør-pc – Her sker interaktionen mellem installatør og denne controller klasse. Det er herfra al konfiguration og system test sker fra
- 2) Kontrolsystem – Denne klasse modtager konfigurationsbeskeder eller test beskeder og korrigerer disse ud korrekt til X10 enheder.
- 3) Log – Håndterer en data log for sikkerhedssystemet ændringer af tilstande, udløsninger og konfiguration



Figur 29: Klassediagram for use case 4

Domæneidentifikation

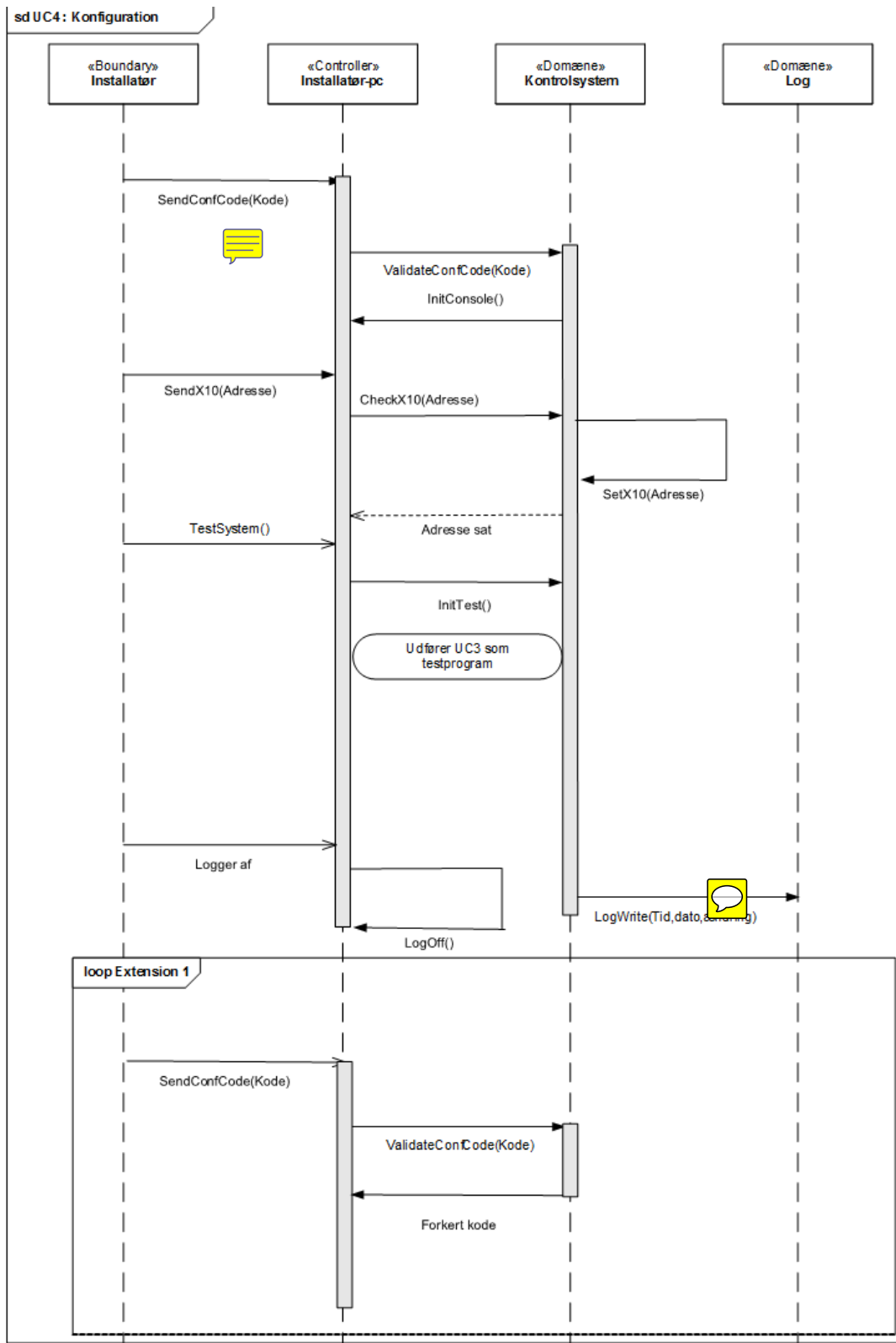
På figure 30. **Fejl! Henvisningskil**  **ikke fundet.** ses domænemodel for use case 4 "Konfiguration/test af system".



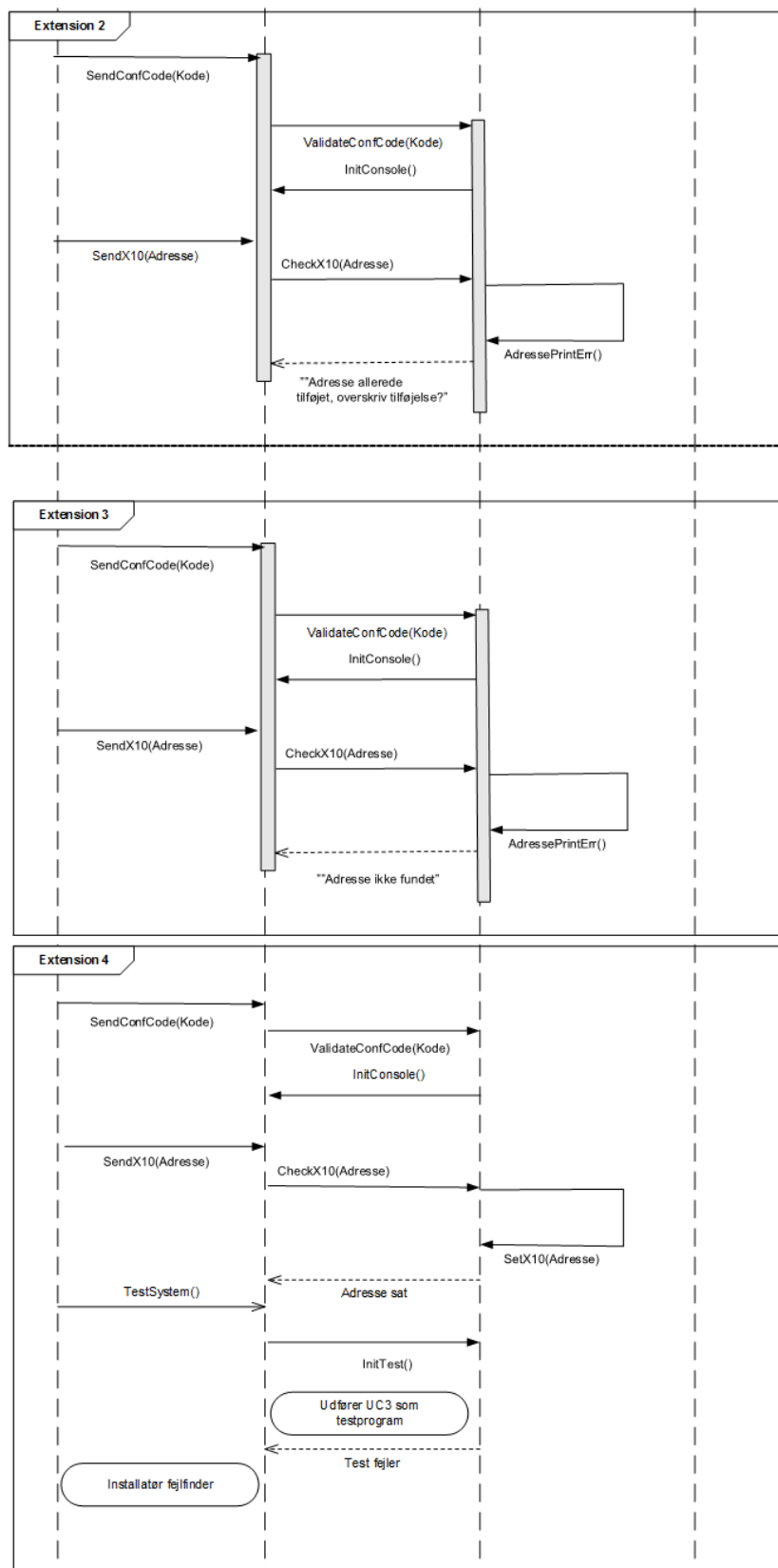
Figur 30: Domænemodel for use case 4

Metodeidentifikation

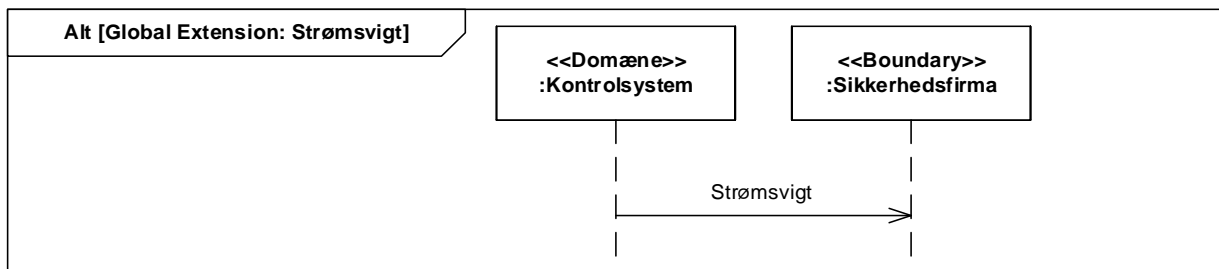
På figur 31, figur 32 og figur 33 ses sekvensdiagrammerne for use case 4. Dette sekvensdiagram er fordelt ud på tre forskellige diagrammer.



Figur 31: Sekvens diagram for use case 4, del 1



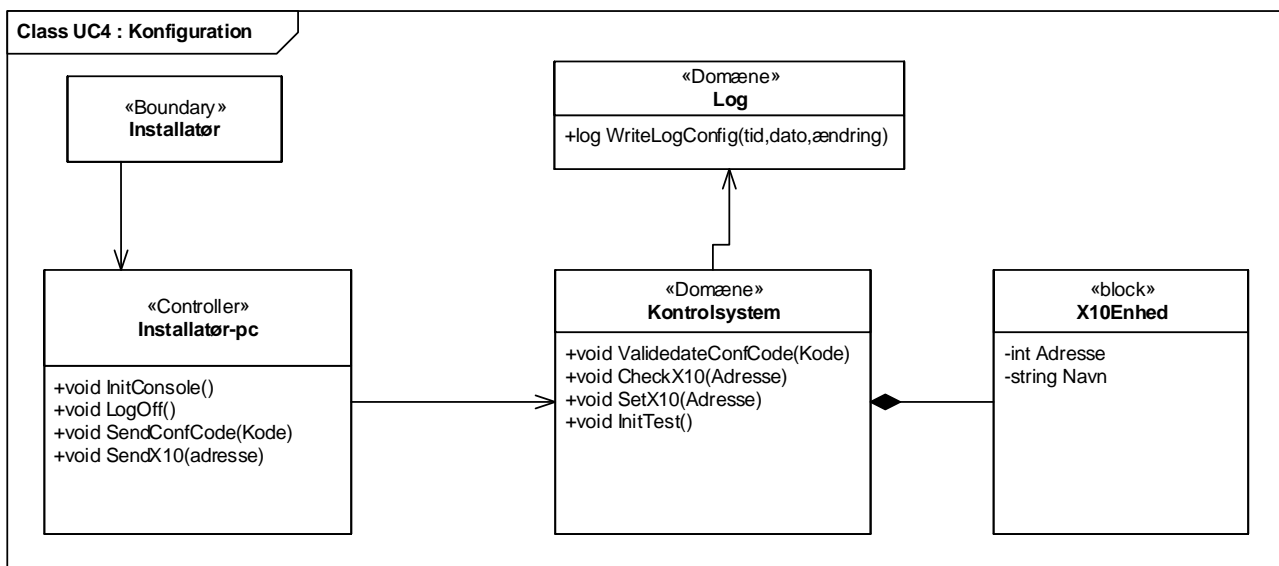
Figur 32: Sekvens diagram for use case 4, del 2



Figur 33: Sekvens diagram for use case 4, del 3

Kontrolklasseidentifikation

På Figur 34 ses state machine for use case 4. Her vil systemet afvente et login, hvorefter brugeren kommer ind til en konsol med valgfri mulighed for enten configuration eller test. Her vil brugeren vælge mellem at konfigurere X10 adresser, konfigurere kodelås og at teste systemet. Det kan ikke ses på diagrammet, men en forudsætning for at systemet kan testes skal systemet være konfigureret. Dette kan læses mere om i de respektive use cases.



Figur 34: State machine for use case 4