1 Systemarkitektur

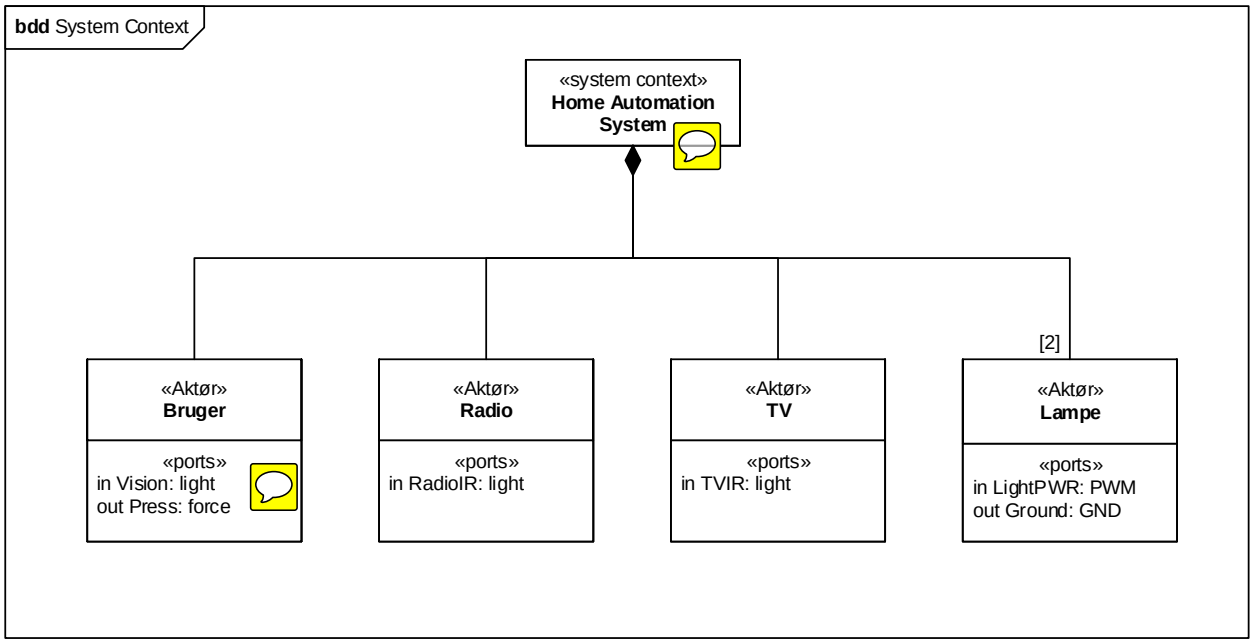
1.1 Version

Dato	Version	Initialer	Ændring
29. oktober	1	LS	Første udkast af dokumentet

1.2 SysML diagrammer

1.2.1 BDD for system kontekst

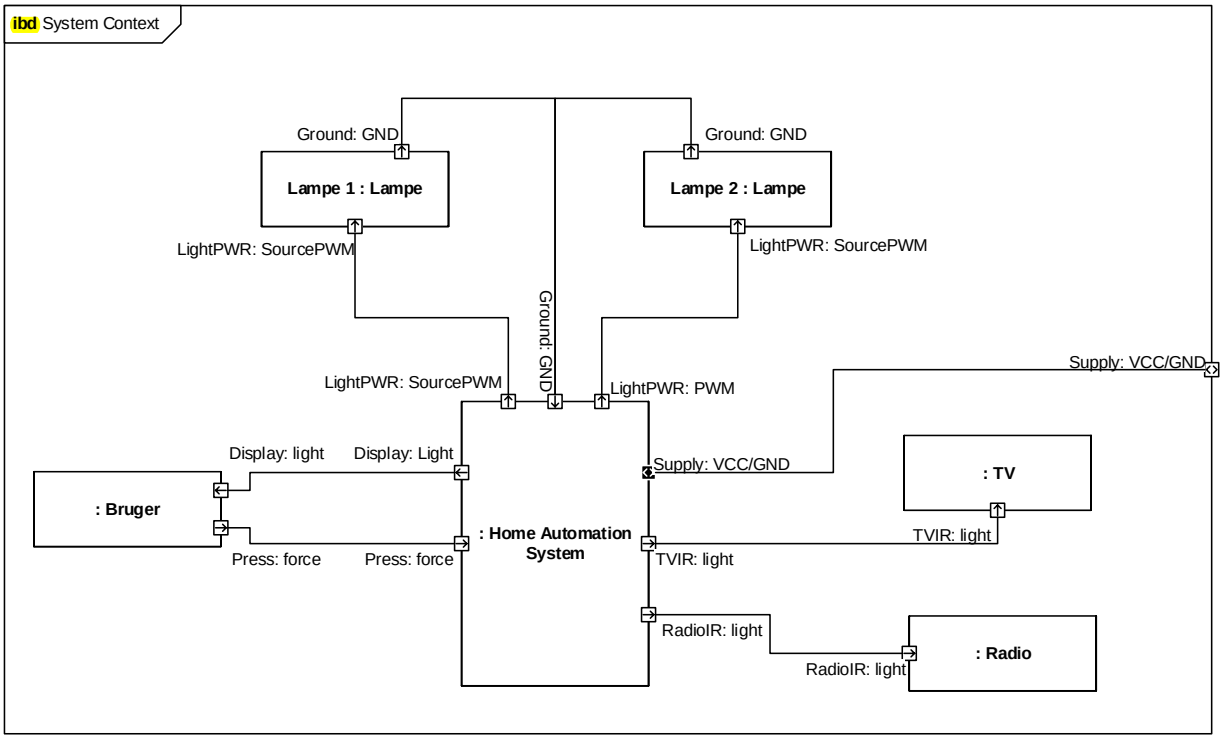
I Figur 1 vises konteksten for systemet, som består af de elektriske enheder samt brugeren af systemet. Yderligere vises porte på aktørerne, som agerer med systemet. Bruger blokken beskriver den person der interagerer med systemet. Radio blokken er en radio med mulighed for styring via en IR fjernbetjening, TV blokken er tilsvarende. De to lampe blokke er to $12V$ lyskilder, som kan dimmes via pulsbreddemodulation.



Figur 1: BDD diagram for system konteksten

1.2.2 IDB for system kontekst

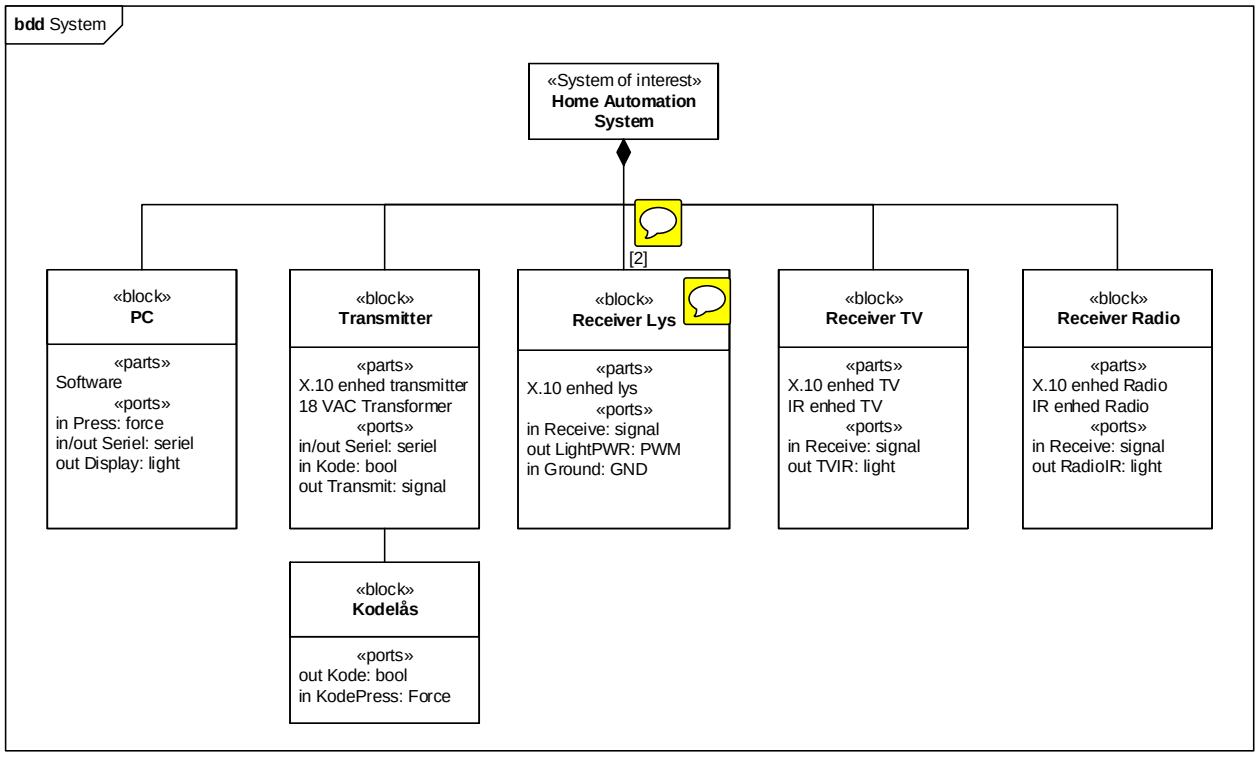
I Figur 2 vises systemets eksterne forbindelser til de øvrige blokke omkring systemet.



Figur 2: IBD diagram for system konteksten

1.2.3 BDD for systemet

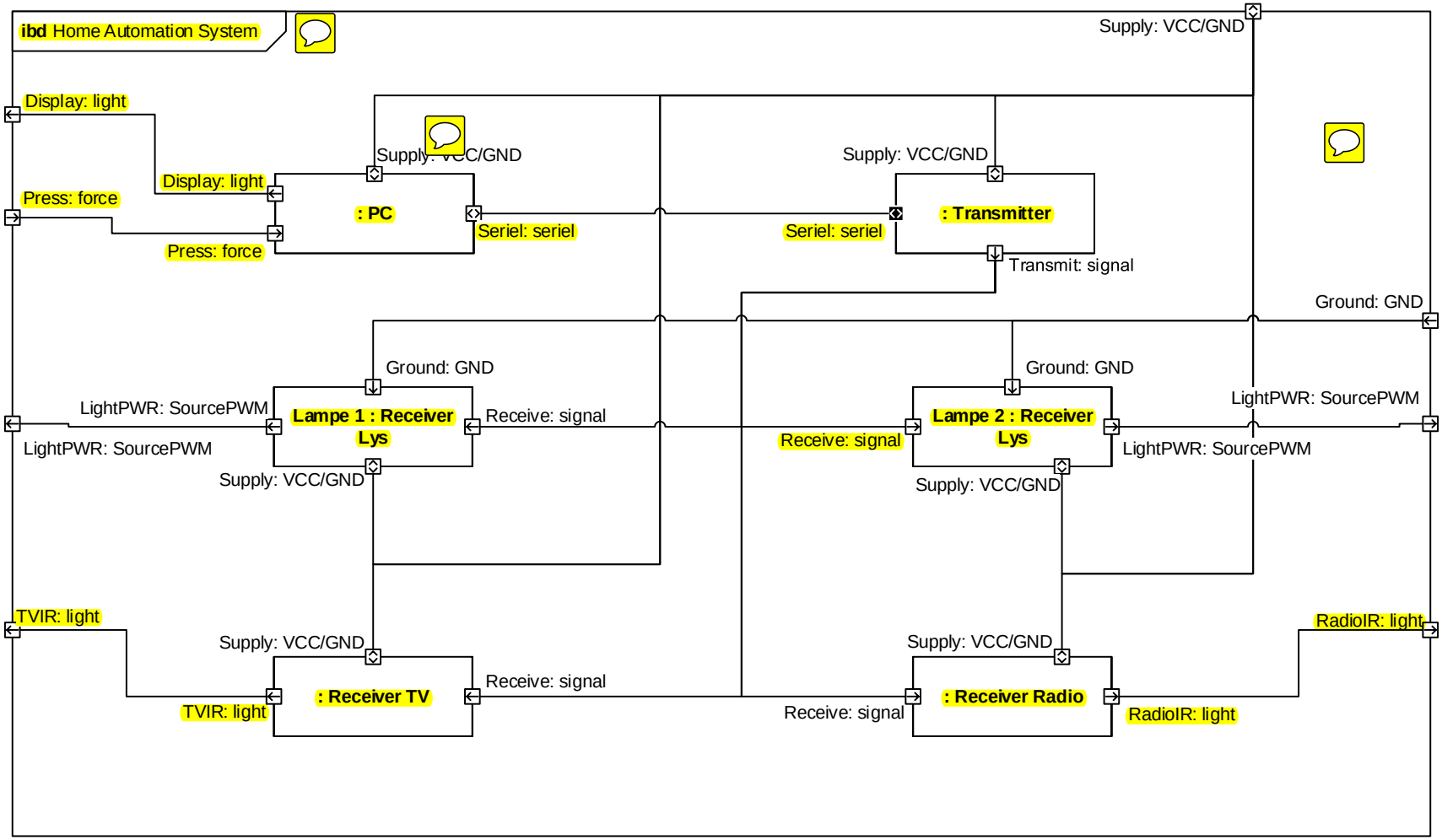
I Figur 3 vises et BDD over systemet i sin helhed. Diagrammet viser overordnede blokke i systemet, samt deres ports og parts.
PC blokken er brugerens grænseflade til systemet. Transmitterblokken modtager information fra PC softwaren, såfremt kodelåsen er aktiveret, og sender kommandoer til Receiver-blokkene.



Figur 3: BDD diagram for systemet

1.2.4 IBD for systemet

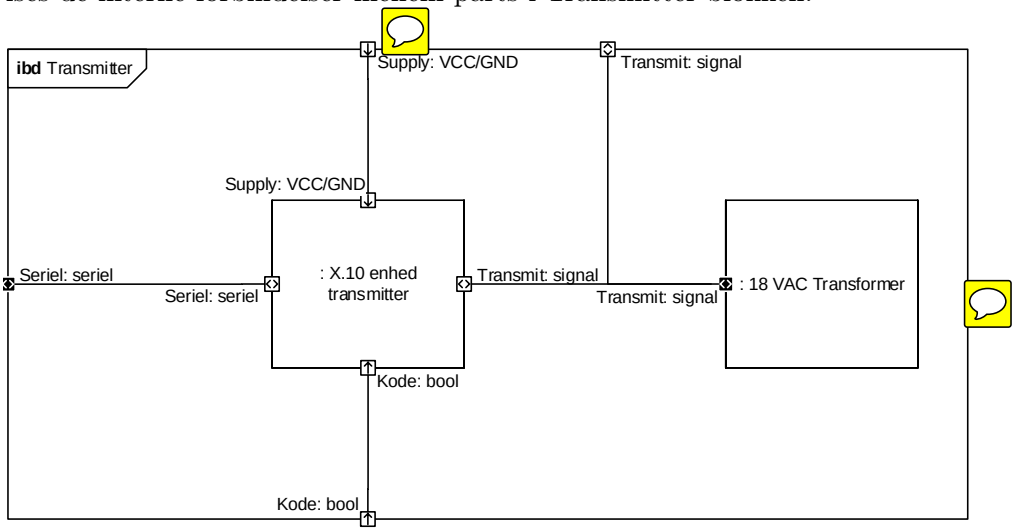
I Figur 4 vises de interne forbindelser mellem blokke i systemet.



Figur 4: IBD diagram for Home Automation systemet

1.2.5 IBD for Transmitter-blokken

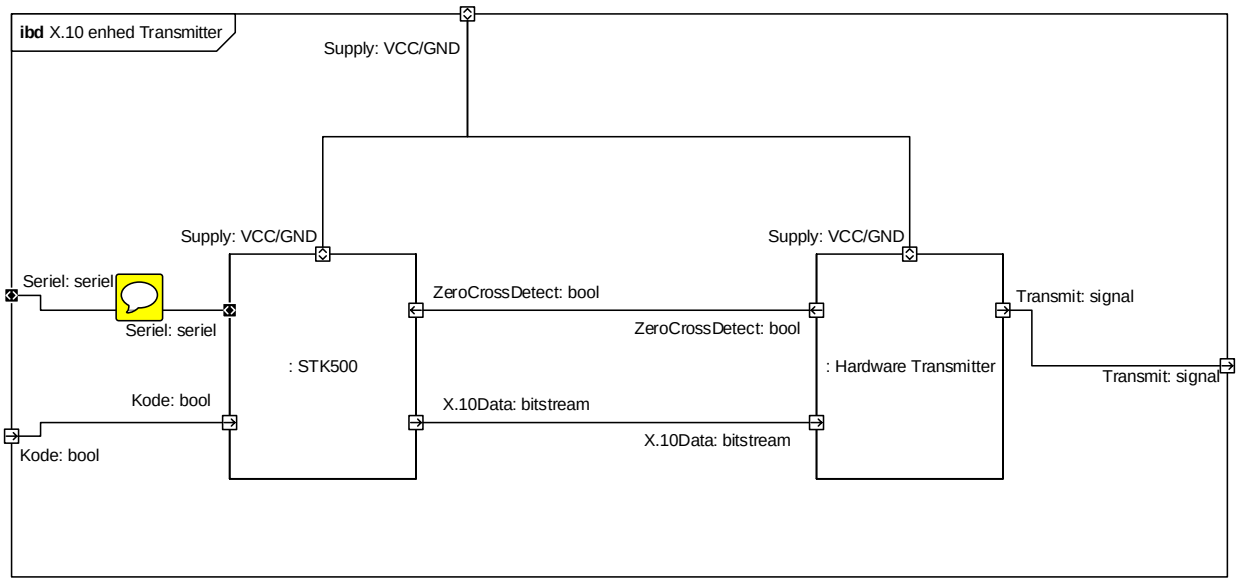
I Figur 5 vises de interne forbindelser mellem parts i Transmitter-blokken.



Figur 5: IBD diagram for Transmitter

1.2.6 IBD for X10-enhed i transmitter-blokken

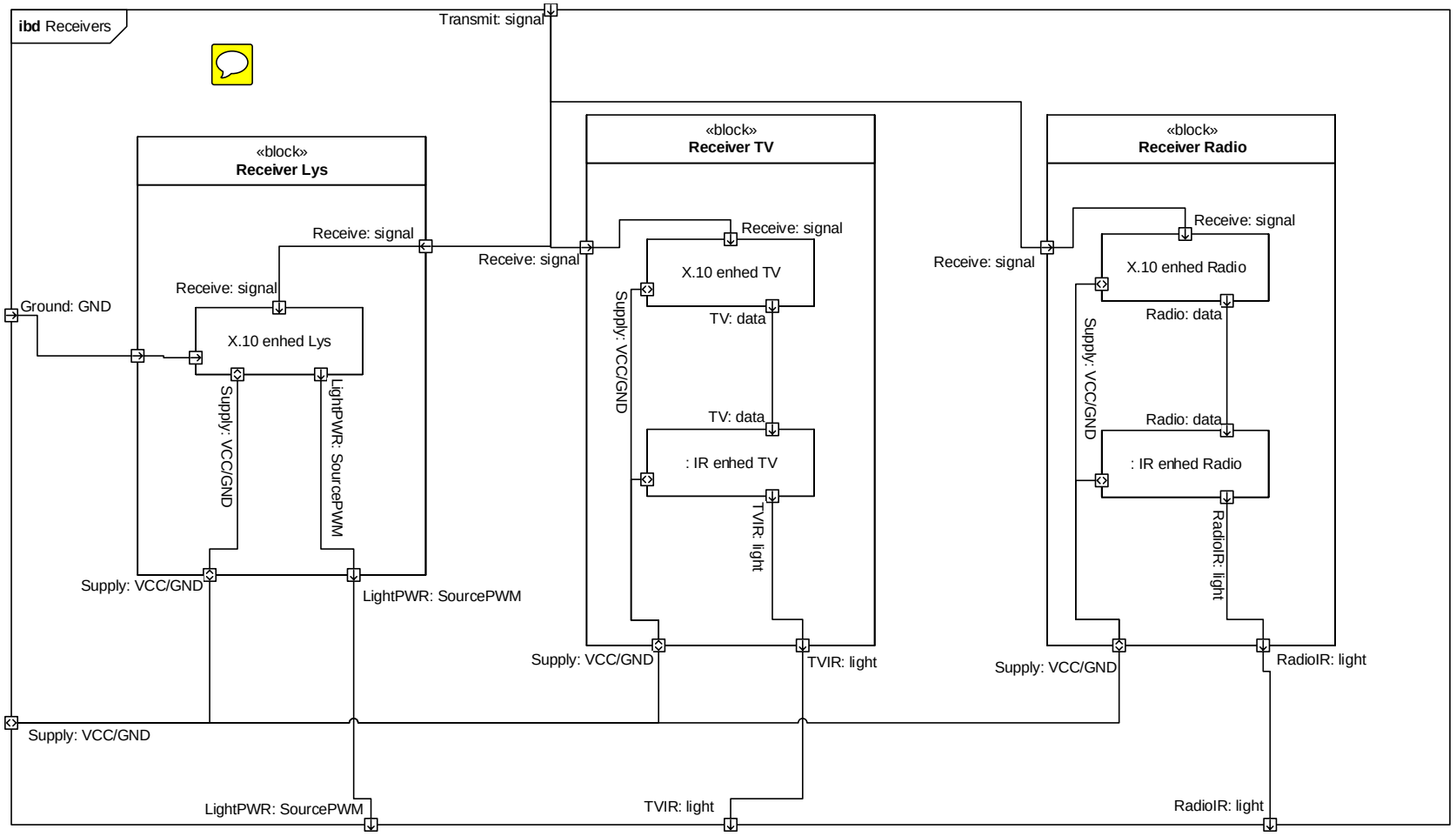
I Figur 6 vises forbindelser mellem STK500 og øvrig hardware i blokken, og dermed grænsefladen mellem software og hardware.



Figur 6: IBD diagram for X.10 enheden i transmitteren

1.2.7 IDB for receivers

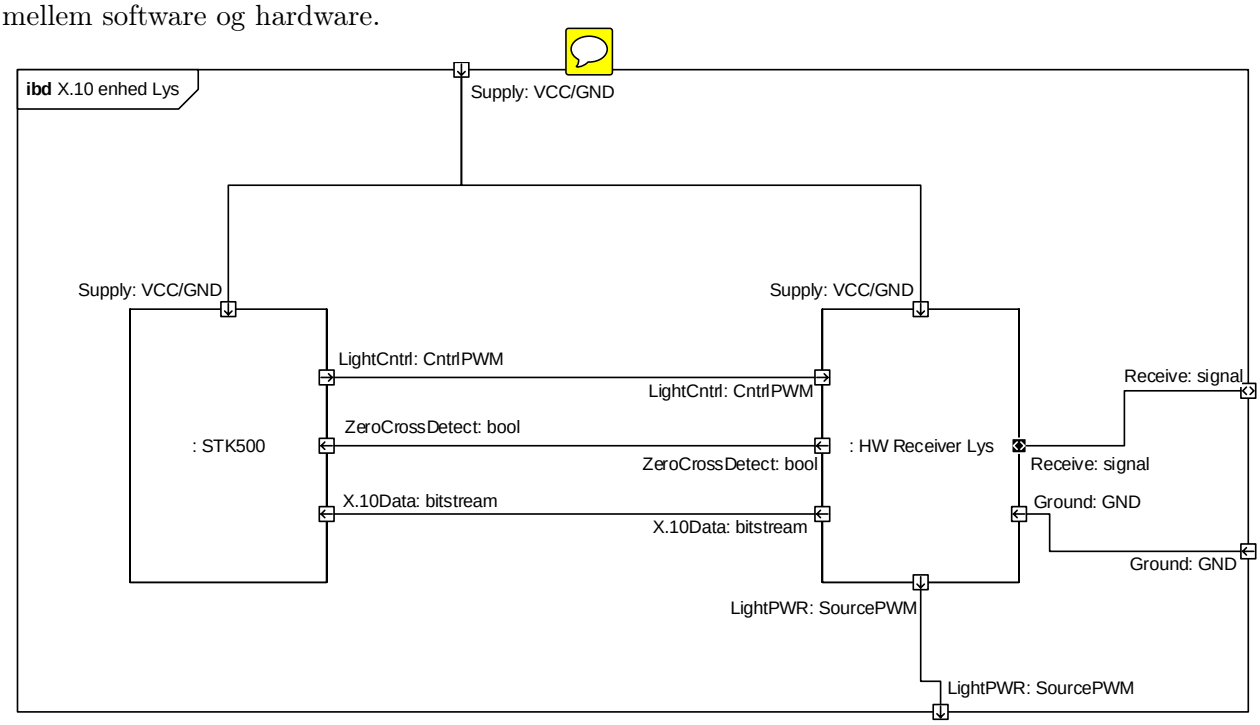
I Figur 7 vises samtlige typer af receivers i systemet. Af hensyn til overskuelighed vises der kun én instans af Receiver Lys, selvom systemet reelt indeholder to, jf multiplicitet i Figur 3 side 3.



Figur 7: IBD diagram for receivers

1.2.8 IDB for X.10 enhed i lys-blokken

I Figur 8 vises forbindelser mellem STK500 og øvrig hardware i blokken, og dermed grænsefladen mellem software og hardware.



Figur 8: IBD diagram for X.10 enheden i transmitteren

Indtil dette punkt har det komplette system været beskrevet i dokumentationen. Fremover behandles radio- og TV-delen af systemet ikke.

1.3 Signalbeskrivelser

1.3.1 Signaltyper

Signaltype	Funktion	Område	Kommentar
bitstream	Serie af 1'er og 0'er	HIGH: $4.2 - 5.0V$, LOW: $0.0 - 0.7V$	1 = Tilstedeværelse af $120kHz$ signal, som går fra LOW til HIGH. 0 = LOW
cntrlPWM	PWM signal til regulering af lysstyrke	$0.0 - 5.0V$	PWM duty cycle kan variere i området $5\% - 95\%$ i trin af $10\% \pm 1\%$
force	Brugerens input på PC		
GND	Reference til lys	$0.0V$	Stel
light	Output på PC'ens skærm		
seriel	Kommunikation mellem PC og transmitter	jf. RS232	
signal	Komposit af 18V AC og X.10	(Transformer)	
sourcePWM	PWM spændingsforsyning til lys	$0V - VCC$	
VCC/GND	Spændingsforsyning og reference	VCC: $11,8 - 12,2V$, GND: $0.0V$	To parallelle forbindelser.
bool	Kan være enten HØJ (1) eller LAV (0)	HØJ: $4.2 - 5.0V$, LAV: $0.0 - 0.7V$	1 = HIGH, 0 = LOW

1.3.2 Grænseflader

Transmitter

- *ZeroCrossDetected: bool*
  Sættes til HØJ når der registreres en nulgennemgang på *Transmit: Signal*, ellers er den LAV. Den skal være HØJ i $1ms \pm 0.1ms$.

- *Transmit: Signal*
  Består af et $18VAC$ ved $50Hz$, som for hver nulgennemgang kan indeholde $120KHz$ signaler. Hvis der forefindes et $120KHz$ signal i en nulgennemgang, betragtes det som HØJ. Hvis der ikke forefindes et $120KHz$ signal, betragtes det som LAV.

- *X.10Data: bitstream*
  Hvis man ønsker at sende HØJ ud på *Transmit: Signal* skal *X.10Data: bitstream* være 1 fra rising edge på *ZeroCrossDetected: bool* til $1ms$ derefter.

- *Kode: bool*
  Er HØJ hvis koden er indtastet korrekt og LAV hvis den ikke er indtastet korrekt.

- *Seriel: seriel*
  BAUD rate på 9600, 8 bit, 1 startbit, 2 stopbit og ingen paritet.

Receiver Lys

- *ZeroCrossDetected: bool*
  Sættes til HØJ når der registreres en nulgennemgang på *Transmit: Signal*, ellers er den LAV. Den skal være HØJ i $1ms \pm 0.1ms$.

- *Receive: Signal*
  Består af et $18VAC$ ved $50Hz$, som for hver nulgennemgang til $1ms$ efter kan indeholde $120KHz$ signaler. Hvis der forefindes et $120KHz$ signal i en nulgennemgang, betragtes det som HØJ. Hvis der ikke forefindes et $120KHz$ signal, betragtes det som LAV. *(usikkerhed vedr. amplitude på* $120KHz$*)*

- *X.10Data: bitstream*
  Når *ZeroCrossDetected: bool* går HØJ aflæses værdien på *X.10Data: bitstream*. Når der betragtes et HØJT signal på *Receive: Signal*, skal *X.10Data: bitstream* sættes HØJ og vice versa.

- *LightCntrl: CntrlPWM*
  PWM styresignal til regulering af *LightPWR: SourcePWM*.

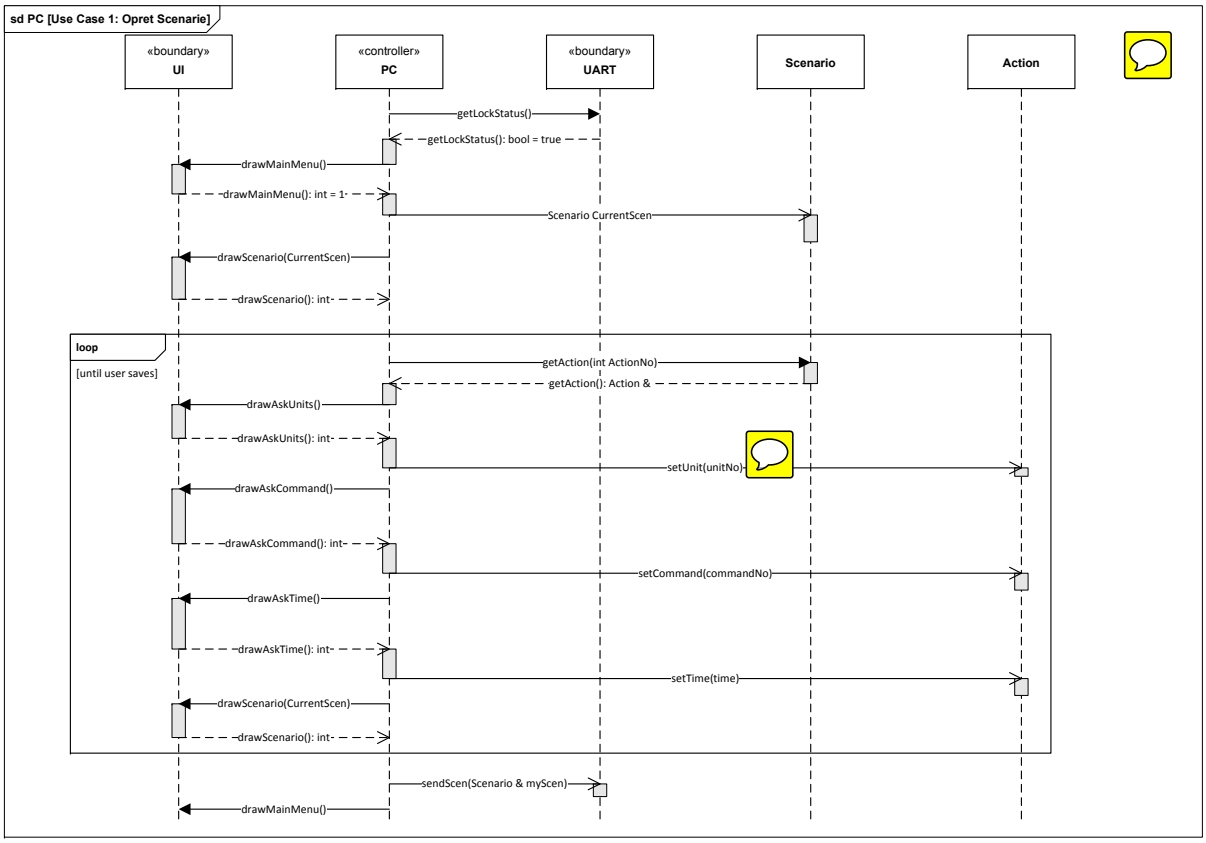
- *LightPWR: SourcePWM*
  PWM forsyningssignal til at drive lampen. Duty cycle for *LightPWR: SourcePWM* og *LightCntrl: CntrlPWM* skal være ens.

1.4 Software arkitektur

For samtlige diagrammer gælder det at prækonditioner til de enkelte UC er medtaget.

1.4.1 Sekvensdiagram for PC [Use Case 1: Opret Scenarie]

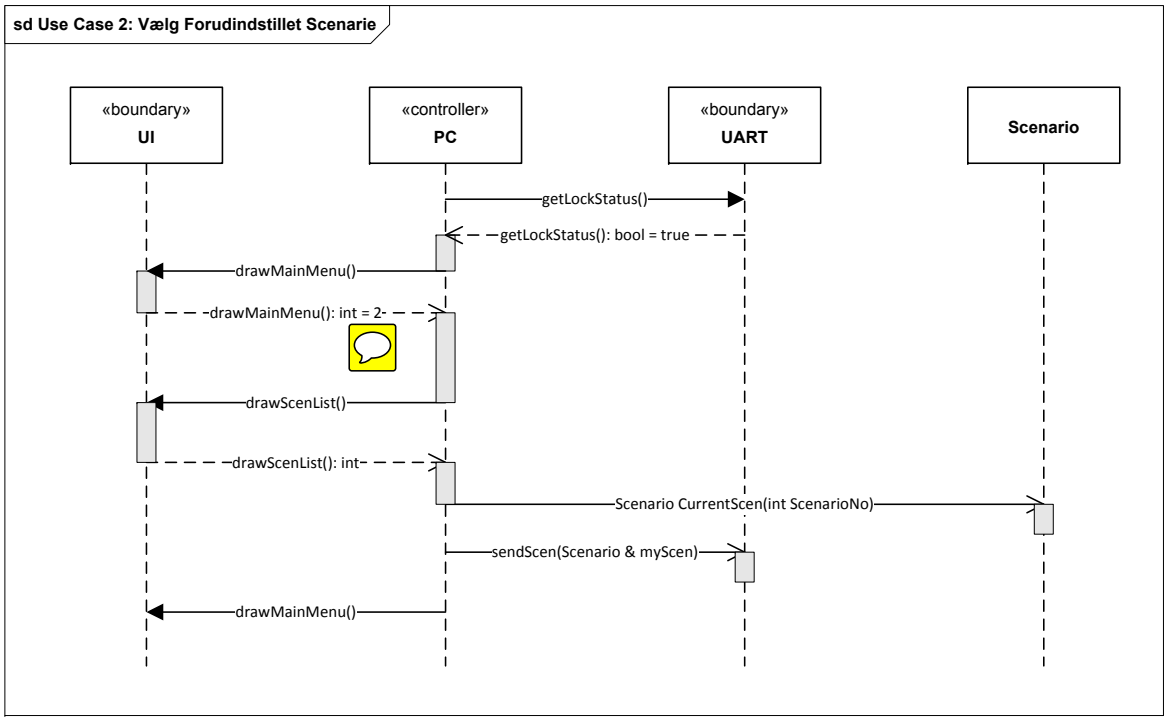
Diagrammet viser sekvenser for PC'en ved gennemgang af hovedscenariet i UC1. Når der oprettes et objekt af klassen Scenario, oprettes det automatisk med 20 tomme aktioner.



Figur 9: Sekvensdiagram for PC [Use Case 1: Opret Scenarie]

1.4.2 Sekvensdiagram for PC [Use Case 2: Vælg Scenarie]

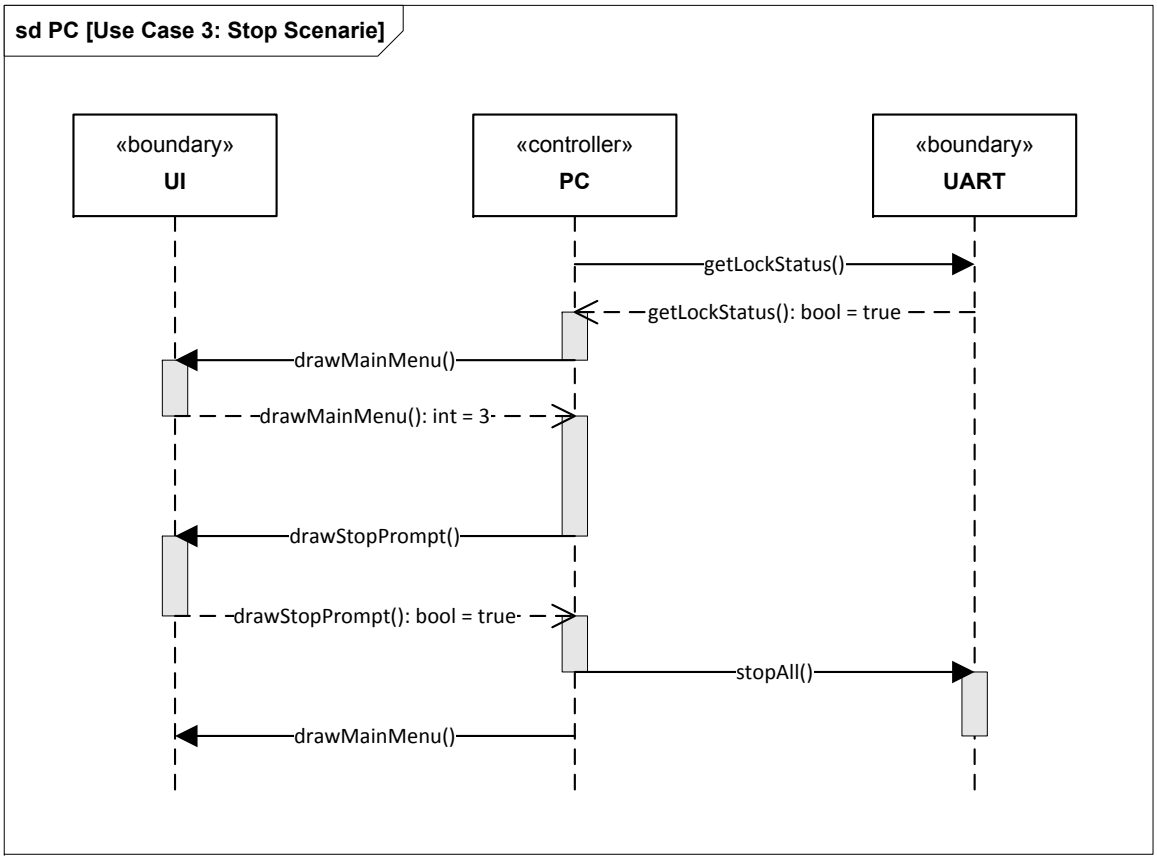
Diagrammet viser sekvenser for PC'en ved gennemgang af hovedscenariet i UC2.



Figur 10: Sekvensdiagram for PC [Use Case 2: Vælg Scenarie]

1.4.3 Sekvensdiagram for PC [Use Case 3: Stop Scenarie]

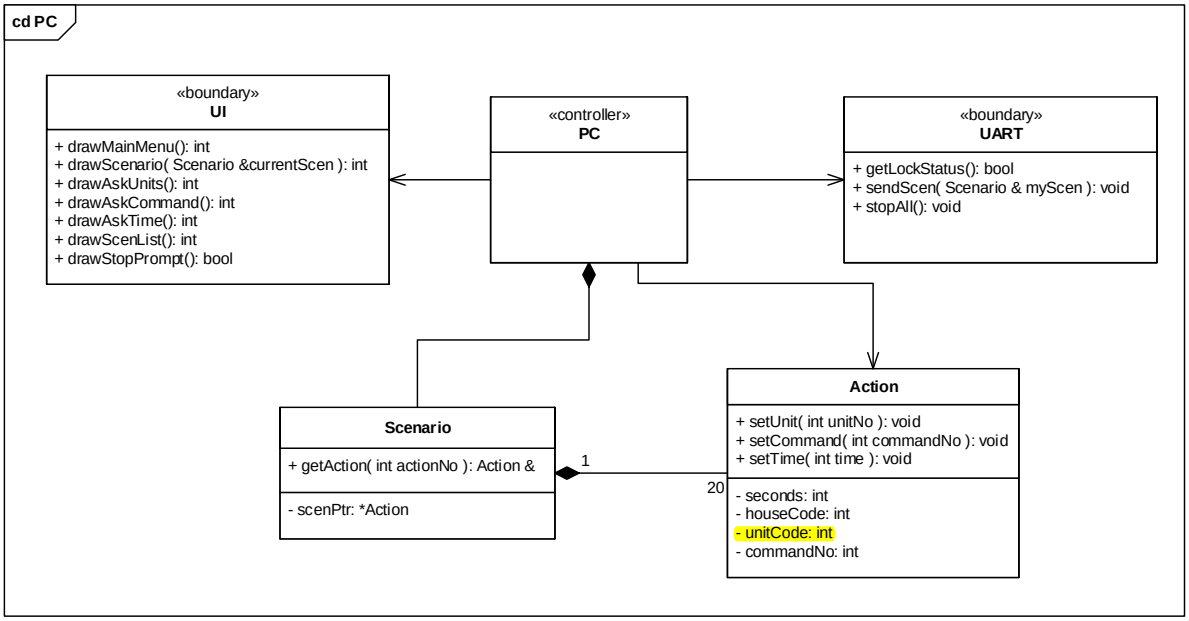
Diagrammet viser sekvenser for PC'en ved gennemgang af hovedscenariet i UC3.



Figur 11: Sekvensdiagram for PC [Use Case 3: Stop Scenarie]

1.4.4 Klassediagram for PC

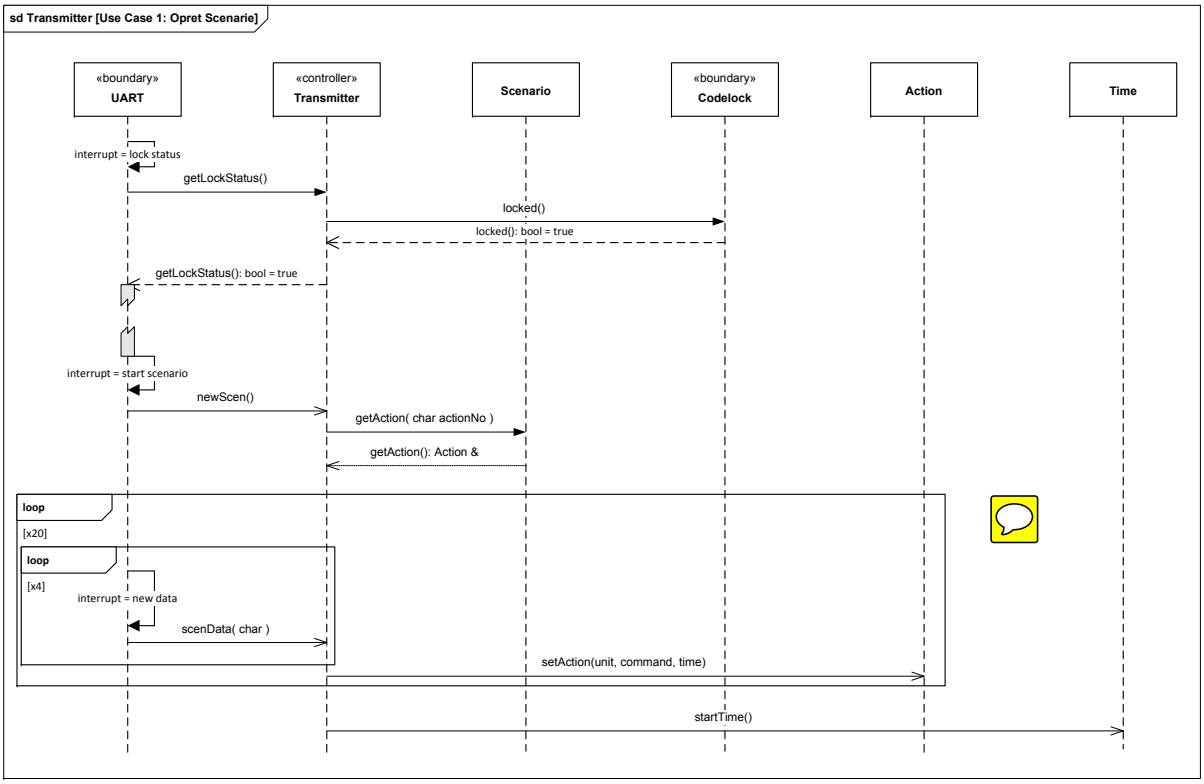
Diagrammet viser klassediagram for den software, der ligger på PC'en med domæne-, controller- og boundary klasser. Boundaryklassen UI har til formål at formidle kommunikation mellem bruger og controller klasssen PC. Boundary klassen UART har ansvar for at kommunikere mellem controller klassen PC og transmitterblokken. Domæneklassen Scenario indeholder op til 20 objekter af domæneklassen Action, der indeholder informationer om aktionen.



Figur 12: Klassediagram for PC.

1.4.5 Sekvensdiagram for transmitter [Use Case 1: Opret Scenarie]

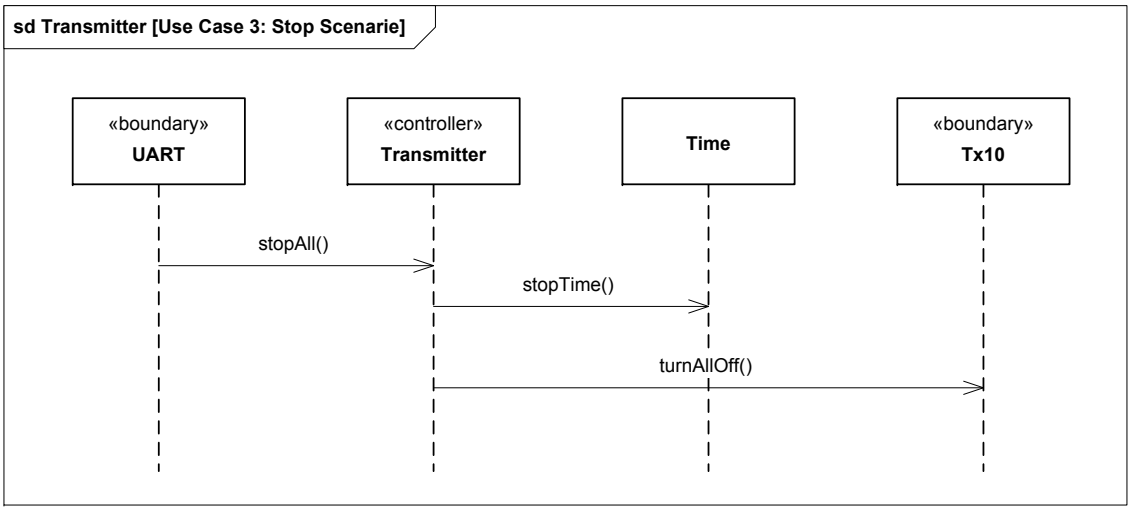
Diagrammet viser sekvenser for transmitteren ved gennemgang af hovedscenariet i UC1.



Figur 13: Sekvensdiagram for transmitter [Use Case 1: Opret Scenarie]

1.4.6 Sekvensdiagram for transmitter [Use Case 3: Stop Scenarie]

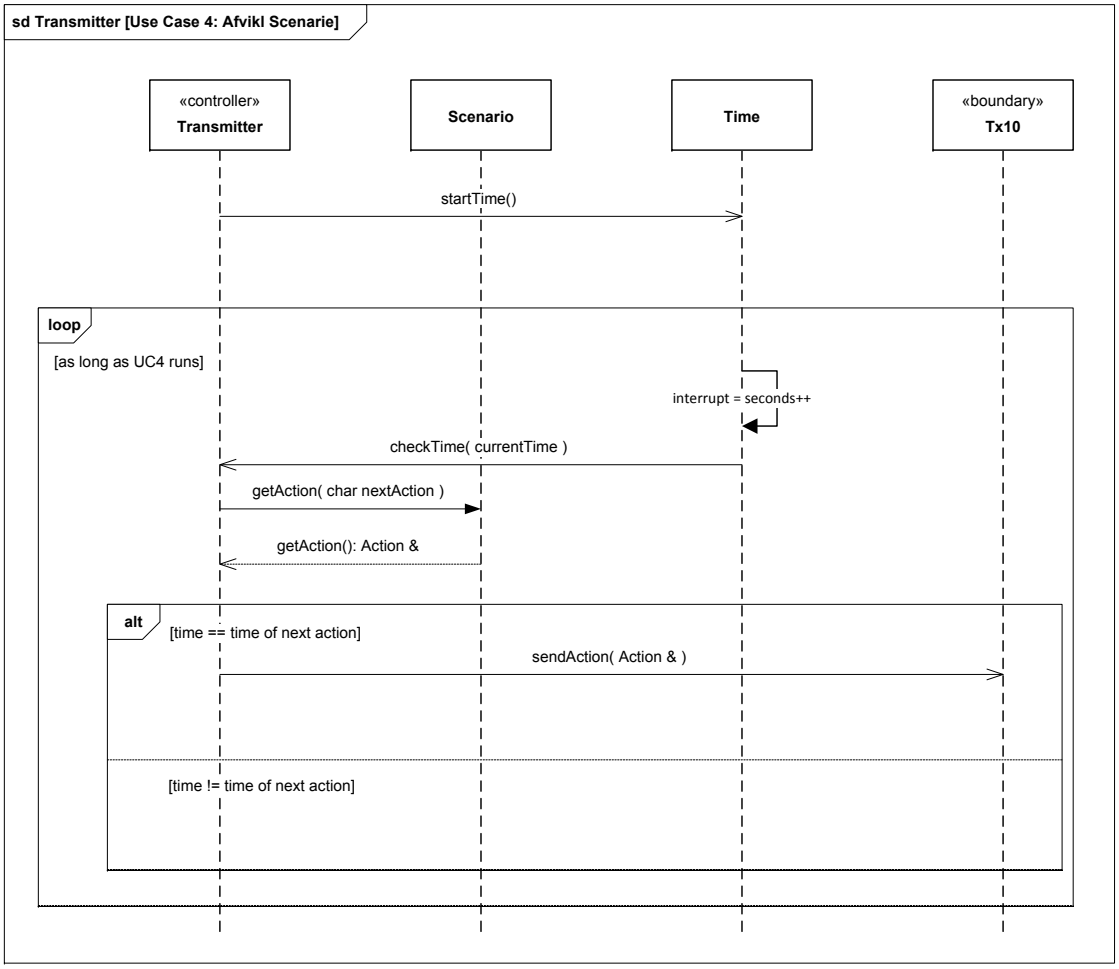
Diagrammet viser sekvenser for transmitteren ved gennemgang af hovedscenariet i UC3.



Figur 14: Sekvensdiagram for transmitter [Use Case 3: Stop Scenarie]

1.4.7 Sekvensdiagram for transmitter [Use Case 4: Afvikl Scenarie]

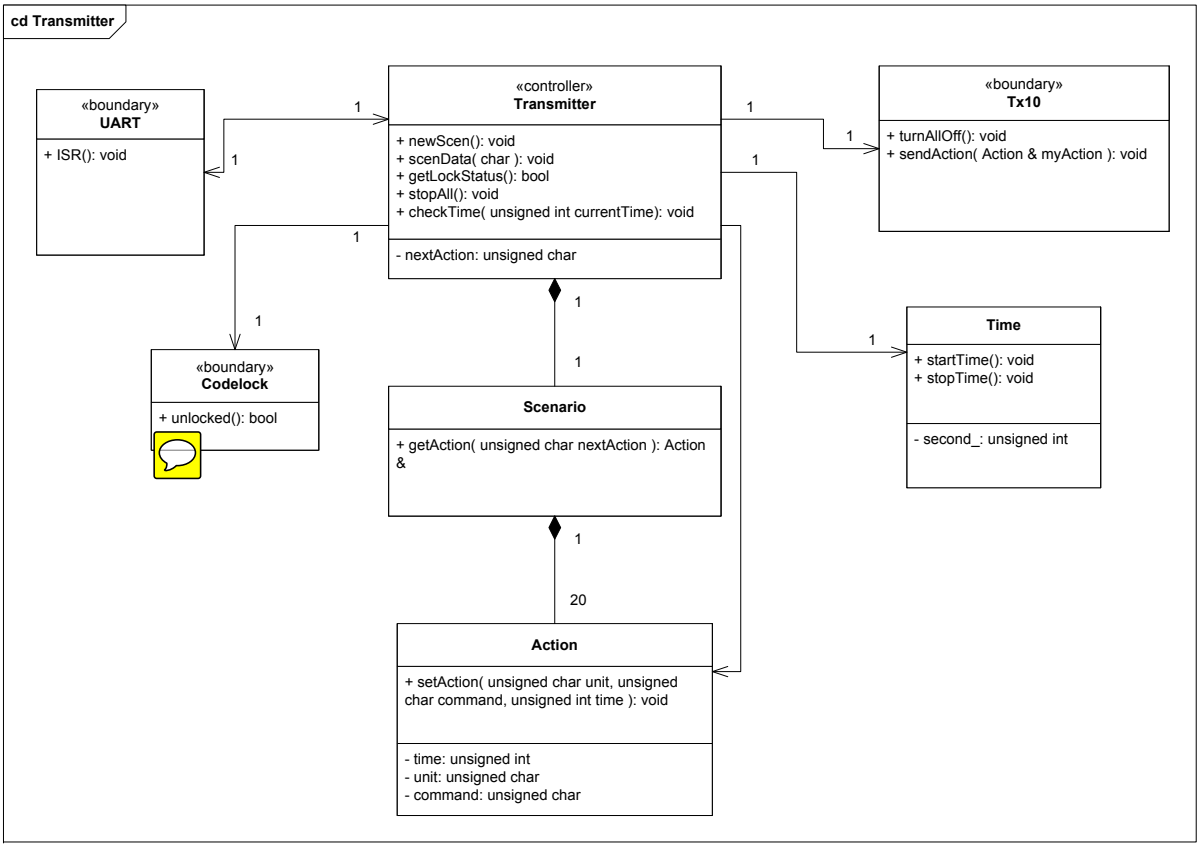
Diagrammet viser sekvenser for transmitteren ved gennemgang af hovedscenariet i UC4.



Figur 15: Sekvensdiagram for transmitter [Use Case 4: Afvikl Scenarie]

1.4.8 Klassediagram for transmitter

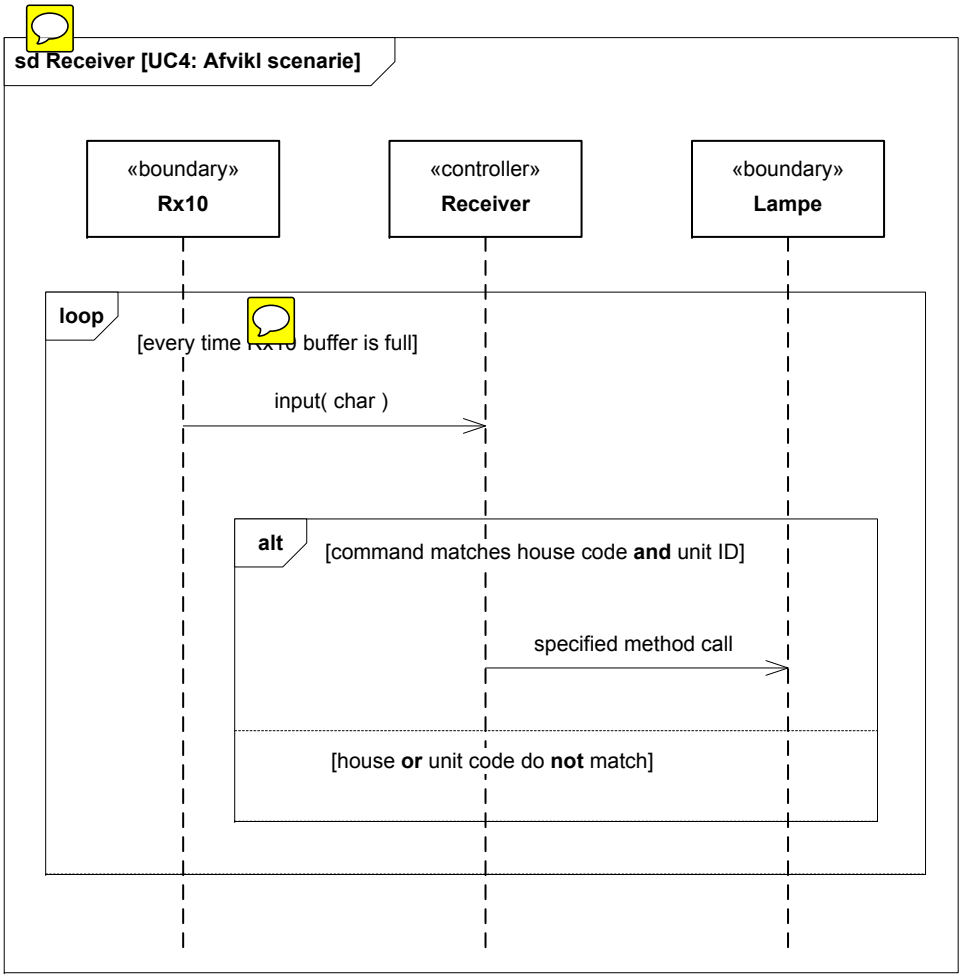
Klassediagram for transmitter.



Figur 16: Klassediagram for transmitter

1.4.9 Sekvensdiagram for receiver [Use Case 3: Stop Scenarie

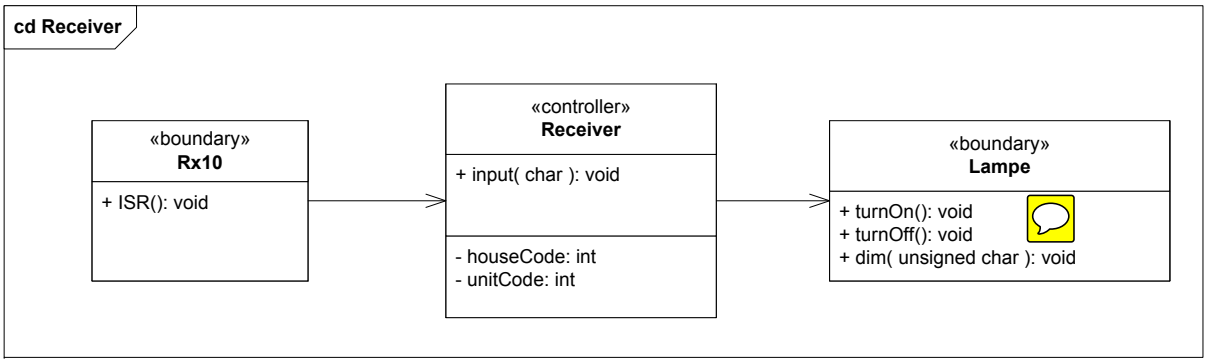
Diagrammet viser sekvenser for receiveren ved gennemgang af hovedscenariet i UC4.



Figur 17: Sekvensdiagram for transmitter [Use Case 3: Stop Scenarie]

1.4.10 Klassediagram for receiver

Diagrammet viser klassediagram for den software, der ligger på receiveren med controller- og boundary klasse(r). Boundary klassen Rx10 har til formål at fortolke information fra hardware receiver blokken og gør det tilgængeligt for controller klassen Receiver. Boundary klassen Lampe har til formål at formidle kommunikation mellem controller klassen Receiver og hardware receiver blokken.



Figur 18: Klasse diagram for receiver