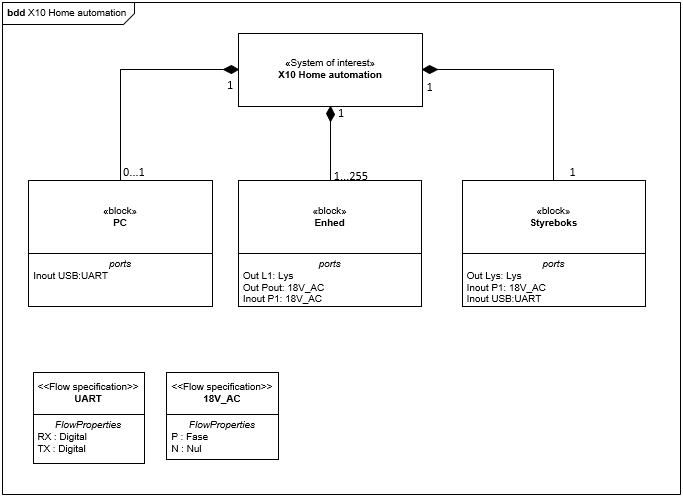
# Arkitektur

I de følgende afsnit beskrives arkitekturen for hardware- og software.

## Hardwarearkitektur (DP)

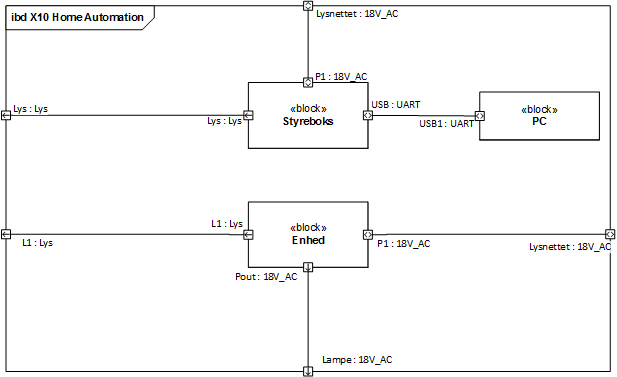
Arkitekturen for hardwaren er designet af hardwaregruppen. Arkitekturen består af BDD’er, IBD’er, blokbeskrivelser og signalbeskrivelser for det system, der skal designes.



Figur - BDD for systemet

På Figur 1 ses det overordnede BDD for systemet. Systemet består af 3 blokke. Hvor af blokkene ”Enhed” og ”Styreboks” er den hardware der skal designes kredsløb for. For mere detaljerede BDD’er for ”Enhed” og ”Styreboks” henvises til projektdokumentationen[[1]](#footnote-1).

Udover BDD’er laves også IBD’er, da disse viser hvordan de individuelle forbindelser i mellem blokkene er koblet sammen.



Figur - IBD for systemet

Figur 2 viser IBD for det overordnede system. Her ses de forskellige in- og outputs og forbindelserne i mellem disse. For mere detaljerede IBD’er for ”Enhed” og ”Styreboks” henvises til projektdokumentationen[[2]](#footnote-2), her findes også signalbeskrivelser.

|  |  |  |
| --- | --- | --- |
| Signaltype | Definition | Beskrivelse |
| 18V\_AC | 18V AC 50 Hz signal kombineret med et 120kHz X10.1 signal | 18V AC 50 Hz signal fra forsyningsnettet der også indeholder kommunikationen via X10.1 protokollen der udvikles specifikt til dette produkt, se protokol afsnit for yderligere information. |
| Lys | Lys i det synlige spektrum | Lys i 3 farver afhængig af hvilket LED indikator der lyser. |
| UART | Kommunikation følger UART-standarden | Protokollen udvikles specifikt til dette produkt. Se protokol afsnit for yderligere information. |

Tabel 1 - Signalbeskrivelser for "IBD for systemet"

Tabel 1 viser signalbeskrivelserne for forbindelserne i ”IBD for systemet” (Figur 2).

## Softwarearkitektur (DP)

Softwaren er udviklet af to softwaregrupper. En gruppe har designet softwaren for PC’en og en anden gruppe har designet softwaren for ”Styreboks” og ”Enheder”. Softwaren tager udgangspunkt i systemets tre CPU’er. På Figur 3 ses den domænemodel, der er udarbejdet for systemet.



Figur - Domænemodel for systemet

Systemets tre CPU’er: PC, Styreboks og Enhed, indeholder data og programkode der skal bruges til eksekvering af deres givne opgaver. Disse bliver derfor identificeret som domain klasser for systemet. Da der arbejdes med et distribueret system, udarbejdes individuelle applikationsmodeller for systemets CPU’er.

Til disse applikationsmodeller bruges domænemodellen til at identificere modellernes klasser: Boundary-, Controller- og Domain klaser. Ved brug af layering identificeres de forskellige lag som de valgt klasser operere på. Lagene indeles i grænseflade-, lagring-, og logisk lag.

Grænseflade laget beskriver systemets interaktion mellem de forskellige blokke, samt forbindelse til udvendige aktøre. Et eksempel på en klasse der operere på dette lag ville være interface klasse mellem Styreboks og PC.

På lagrings laget ligger de klasser der har til formål at lagre data der læses og skrives fra. SD-Kort Modul er et eksempel på en sådan klasse.

Det logiske lag er klasser der eksekvere de ønskede handlinger, som beskrives i de enkelte Use Cases. Disse klasser vil opstå som Control klasser. I applikationsmodellerne vil der blive oprettet en Control klasse per Use Case, for de Use Cases som den givne CPU bliver anvendt i.

### Overordnede Sekvensdiagrammer

For at danne et overblik over hvilke control klasser der er relevante for de individuelle CPU’er lavede vi en matrice over Use Cases og deres tilhørende CPU. Matricen vises i Tabel 2, hvor et X repræsentere at en CPU deltager i den viste Use Case.

|  |  |  |  |
| --- | --- | --- | --- |
| UseCase/cpu | PC Software | Styreboks | Enhed |
| UC1 | X | X |  |
| UC2 | X | X | X |
| UC3 | X | X |  |
| UC4 | X | X |  |
| UC5 | X | X |  |
| UC6 | X | X |  |
| UC7 |  | X | X |
| UC8 | X | X |  |
| UC9 | X | X |  |

Tabel Use Case / CPU Matrix

Før der udfærdiges applikationsmodeller for de enkelte CPU’er, benyttes den ovennævnte matrice til opsætning af overordnede sekvensdiagrammer for hver enkelt Use Case. Disse bruges til at få et overblik over hvordan og hvornår der vil blive kommunikeret mellem hver CPU via deres grænseflader. Et eksempel på et sådant sekvensdiagram vises på Figur 4.

Sekvensdiagrammet er påført første udkast til nødvendige metodekald. Disse metodekald bliver videre specificeret eller ændret videre i arkitekturfasen, under udfærdigelse af de enkelte applikationsmodeller.



Figur Overordnet sekvensdiagram for Use Case 1

Efter udfærdigelse af den overordnede arkitektur er der blevet opstillet et overblik over hvilke behov der er for kommunikation mellem CPU’er, og der kan derfor arbejdes på software arkitektur individuelt for hver CPU. Den følgende arkitektur beskrivelse er derfor delt op mellem de to software grupper.

### Softwarearkitektur for PC

Fra klasseidentifikationen og grænsefladerne vist i de overordnede sekvensdiagrammer dannes en applikationsmodel for softwaren der ligger på PC’en.

Som første led opsættes et tomt klassediagram. Dette klassediagram indeholder de identificerede klasser, og sammenhængen mellem dem. Disse inddeles videre i pakker for de klasser hvis funktion ligner hinanden. Det tomme klassediagram vises på Figur 5



Figur Tomt klassediagram for software på PC

På baggrund af Use Cases skrives sekvensdiagrammer der repræsentere handlingerne der ønskes fra de enkelte klasser. Disse sekvensdiagrammer bruges til at skabe en oversigt over det ønskede handlingsforløb rent programmatisk, ved at identificere metoder baseret på de beskeder der sendes mellem klasserne.

Et eksempel på sådan et sekvensdiagram ses på Figur 6. Herfra identificeres metoder som indskrives på klassediagrammet. Disse sekvensdiagrammer blev igennem design og implementations fasen opdateret løbende, for bedre at reflektere programmets forløb. For de færdige sekvensdiagrammer refereres der til projektdokumentationen.



Figur Sekvensdiagram for Use Case 2 : PC

1. Projektdokumentation side XX afsnit X.X.X [↑](#footnote-ref-1)
2. Projektdokumentation side XX afsnit X.X.X [↑](#footnote-ref-2)