Systemdokumentasjon

Håkon Leithe og Espen Lie
Gruppe: 10

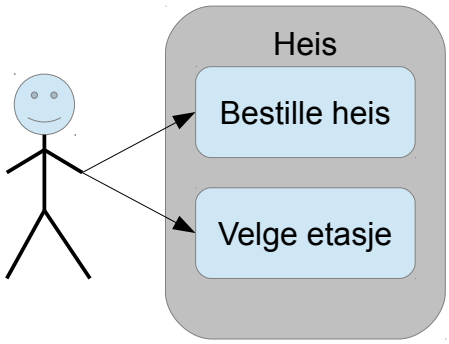
Institutt for teknisk kybernetikk
Norges teknisk-naturvitenskapelige universitet

FORORD

Hensikten med denne oppgaven var å få prøve ut V-modellen i praksis, lære om programmering og få kjennskap til dokumentering. Oppgaven ble løst med bruk av prosjekteringsspråket UML og implementering i programmeringsspråket C. Ut ifra gitte spesifikasjoner fra kunde, skal systemet leveres klar for en FAT.

Innholdsfortegnelse

1. *Use case*-analyse



Illustrasjon 1: Use case-diagram

Denne analysen identifiserer kravene til heisens oppførsel. "Illustrasjon 1: Use case-diagram" viser hvilke inputs systemet kan få.

Initialisere:

1. Heis påslått og initialisert

Trigger:

1. Bestilling til etasje X

Suksess:

1. Kjører til etasje X
2. Stopper i etasje X
3. Dørene åpne i 3 sekunder

Tilleggpunkter:

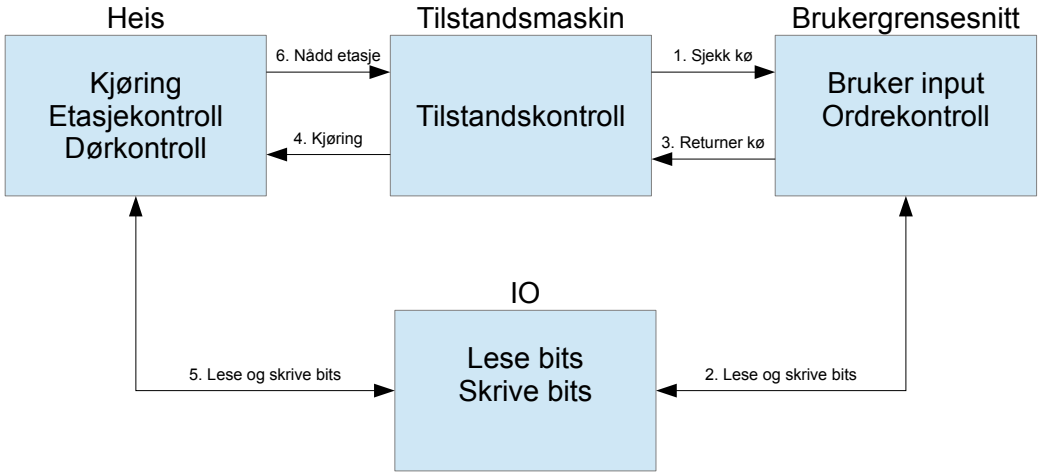
1a: Om heisen mottar en ordre i kjøreretning mellom egen posisjon og X, stopp på veien.

1b: Hvis nødstopp blir aktivert: Stopp, slett ordrer og vent på ny bestilling fra heispanel.

2a: Hvis nødstopp blir aktivert: Slett kø og vent på ordre fra heispanel.

3a: Hvis det blir obstruksjon: hold dørene åpne i 3 sek fra obstruksjon.

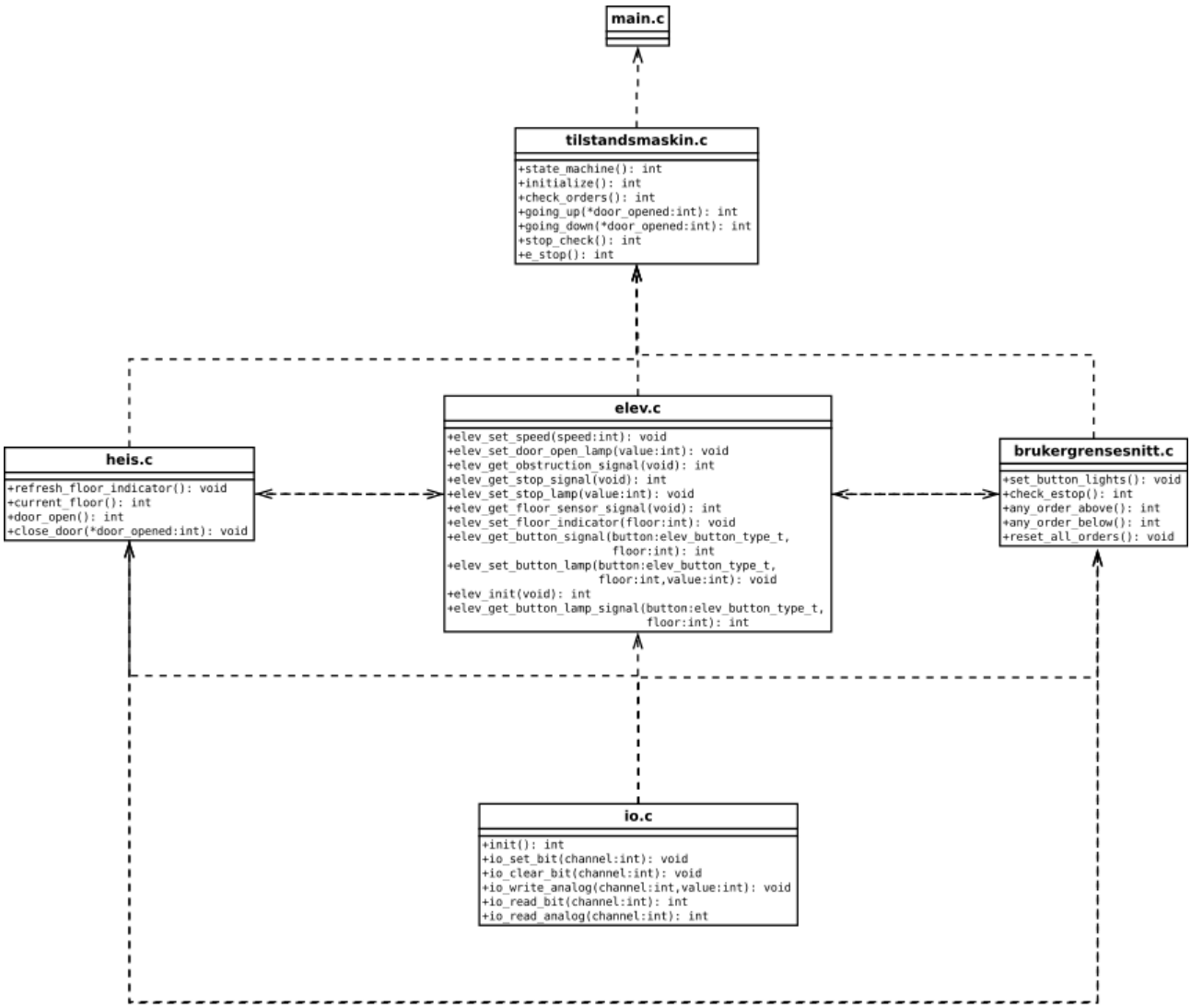
2. Kommunikasjonsdiagram



Illustrasjon 2: Kommunikasjonsdiagram

"Illustrasjon 2: Kommunikasjonsdiagram" viser den overordnede arkitekturen for hvordan de forskjellige modulene i vårt program snakker sammen.

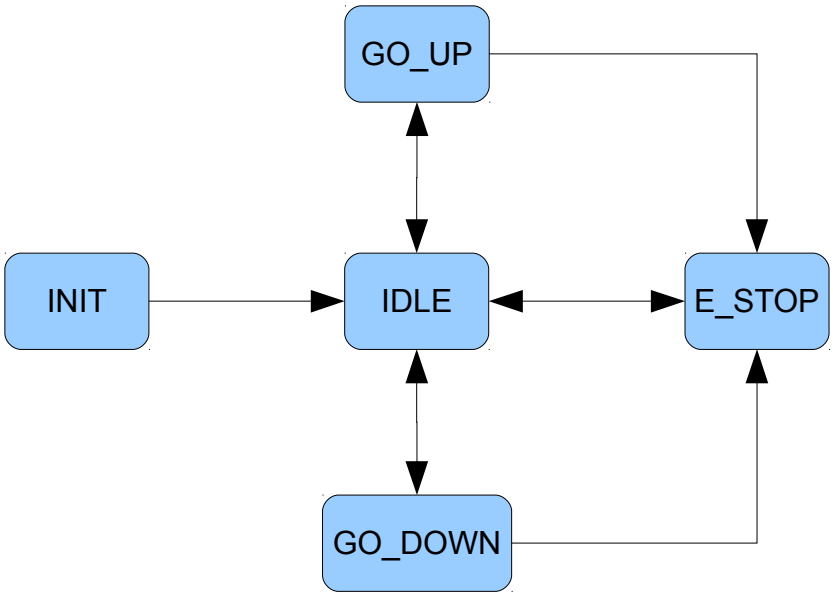
3. Klassediagram



Illustrasjon 3: Klassediagram

Klassediagrammet brukes til å gi en grundig sammenheng mellom de ulike modulene i prosjeket, og bruken av de forskjellige funksjonene. I "Illustrasjon 3: Klassediagram" er klassediagrammet til vårt styringssystem vist.

4. Tilstandsmaskin



Illustrasjon 4: Tilstandsdiagram

"Illustrasjon 4: Tilstandsdiagram" viser heisens 5 forskjellige tilstander, og hvilke tilstandsbytter systemet kan utføre.

5. Diskusjon

Vi har en viktig endring fra forhåndsinnlevert UML, til det ferdige produktet. Endringen er i tilstandsmaskinen. I første UML kunne en gå fra E_STOP og direkte i GO_UP/GO_DOWN i tillegg til IDLE, men vi har endret slik at etter E_STOP så vil heisen gå i IDLE.
Dette ble gjort for å forenkle koden vår endel. Isteden for at E_STOP skal sjekke om heisen skal fortsette opp eller ned, vil IDLE ta seg av dette.

Ellers mener vi at vi har løst oppgaven etter den strukturen som vi først tenkte. "Lesbarheten" og oppsettet av koden er enkel og forståelig. Koden skal være rettferdig ovenfor alle passasjerer, der den alltid vil inkludere endeetasjene.