Heisprosjekt

Andrea Schnell, Sandra Garder Løkken

Februar 2020

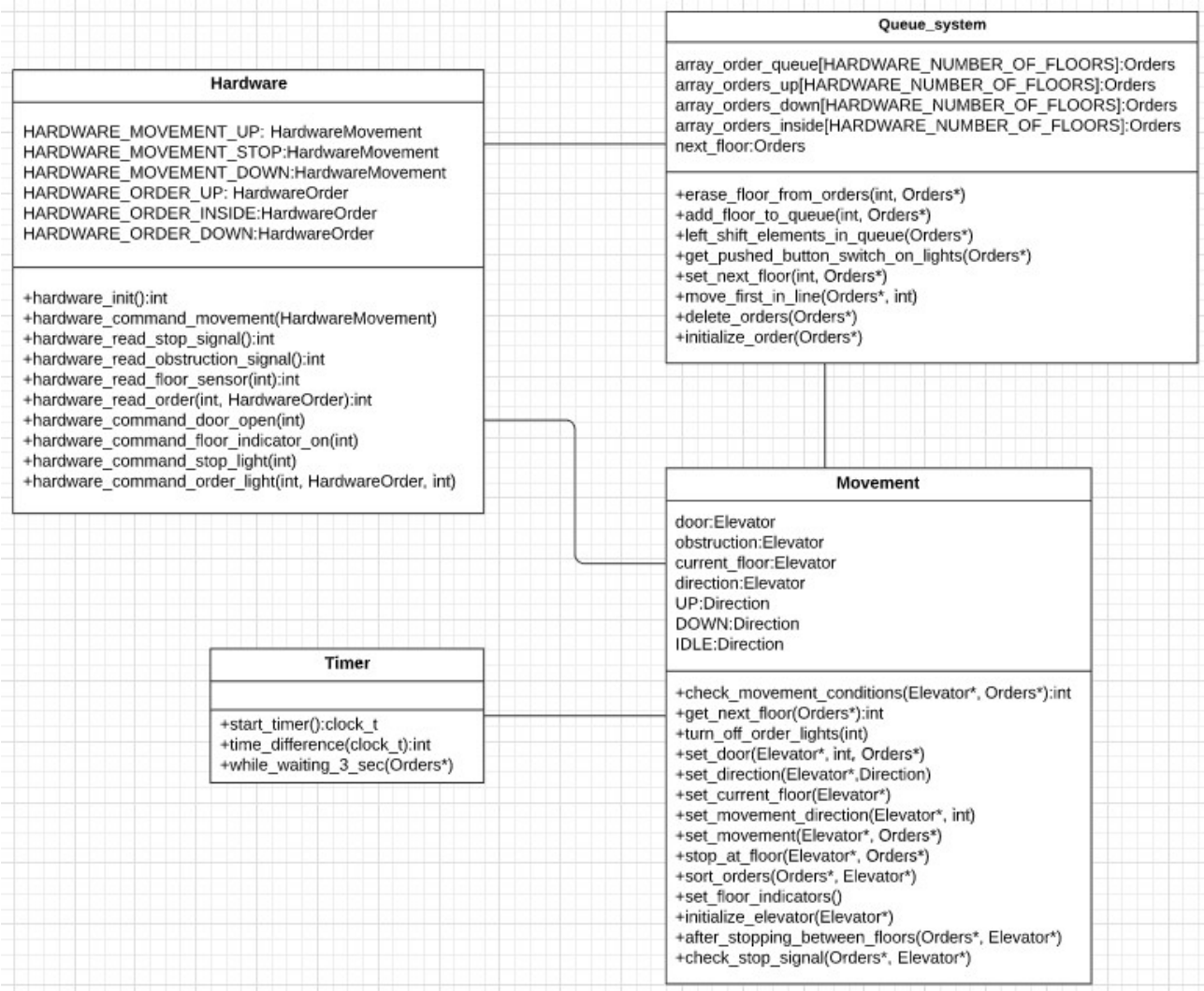
1 Arkitekturdesign

Fra kravspesifikasjonene så vi at vi trengte et køsystem for lagring av bestillinger helt til de ble ekspedert. Dette implementerte vi som en egen modul som registrerer og lagrer bestillinger som blir gjort. Denne må derfor kunne kommunisere med hardware-modulen, for å hente informasjon om hvilke knapper som er trykt. Køsystemet sender den neste prioriterte etasjen fra køen til bevegelses-modulen.

Vi har også en modul som setter heisen i bevegelse. Denne skal kunne kommunisere med hardwaremodulen for å sette outputsignaler og lese inputsignaler for å vite noe om posisjonen til heisen. I tillegg skal den ha tilgang til køsystemet for å vite hvilken etasje som har høyest prioritet. Til slutt trenger den en mulighet til å sette timeren, som er den siste modulen vår.

Kravene spesifiserer at døra skal holde seg oppe i 3 sekunder ved forskjellige anledningner. Vi implementerte en timermodul som kommuniserer med bevegelse-modulen for å gi beskjed om når døren har lukket seg og heisen er klar for neste gjøremål.

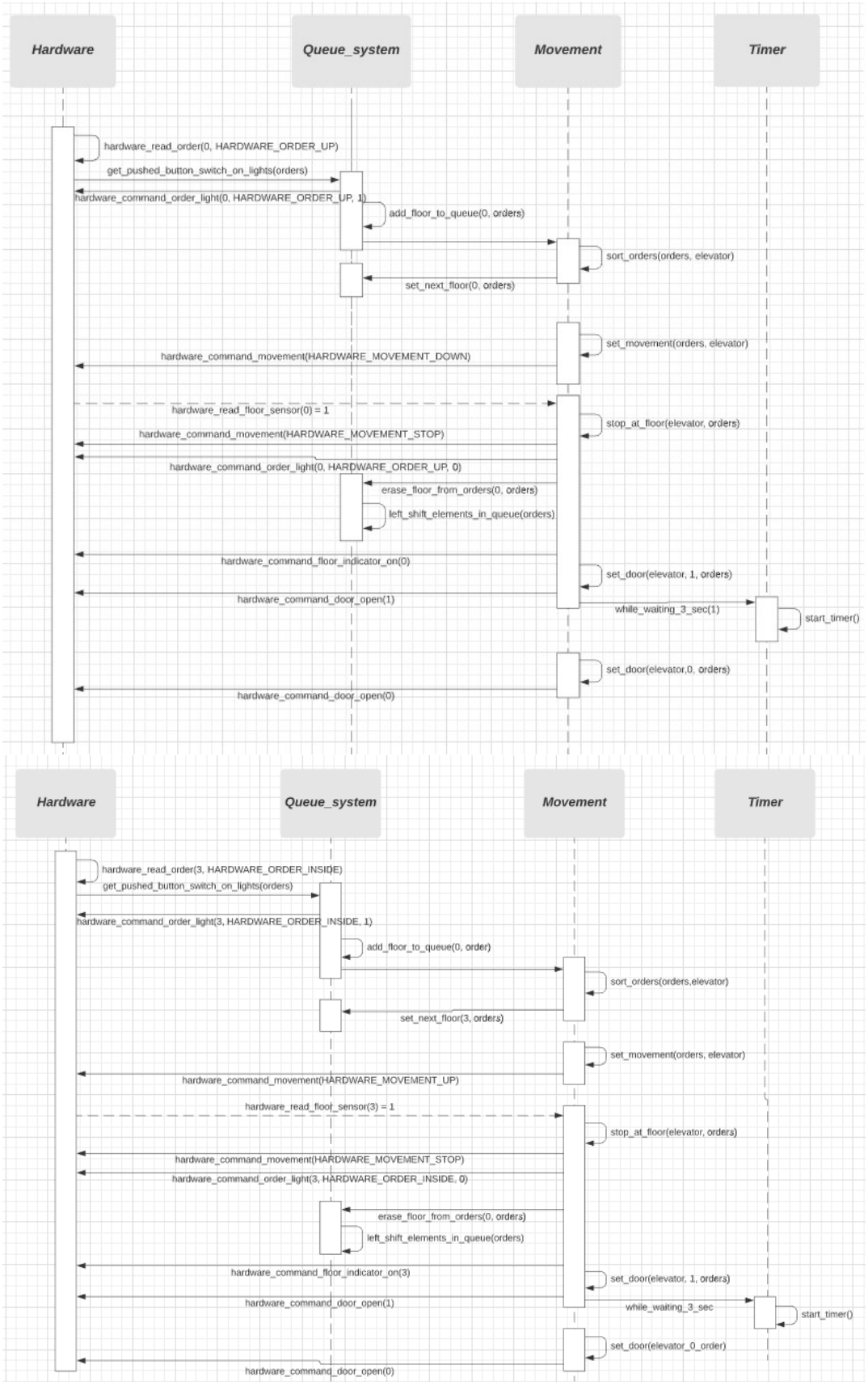
Klassediagrammet er en oversikt over de forskjellige modulene vi har i programmet, samt hvilke funksjoner hver enkelt inneholder. Diagrammet viser også hvilke moduler som kommuniserer med hverandre.



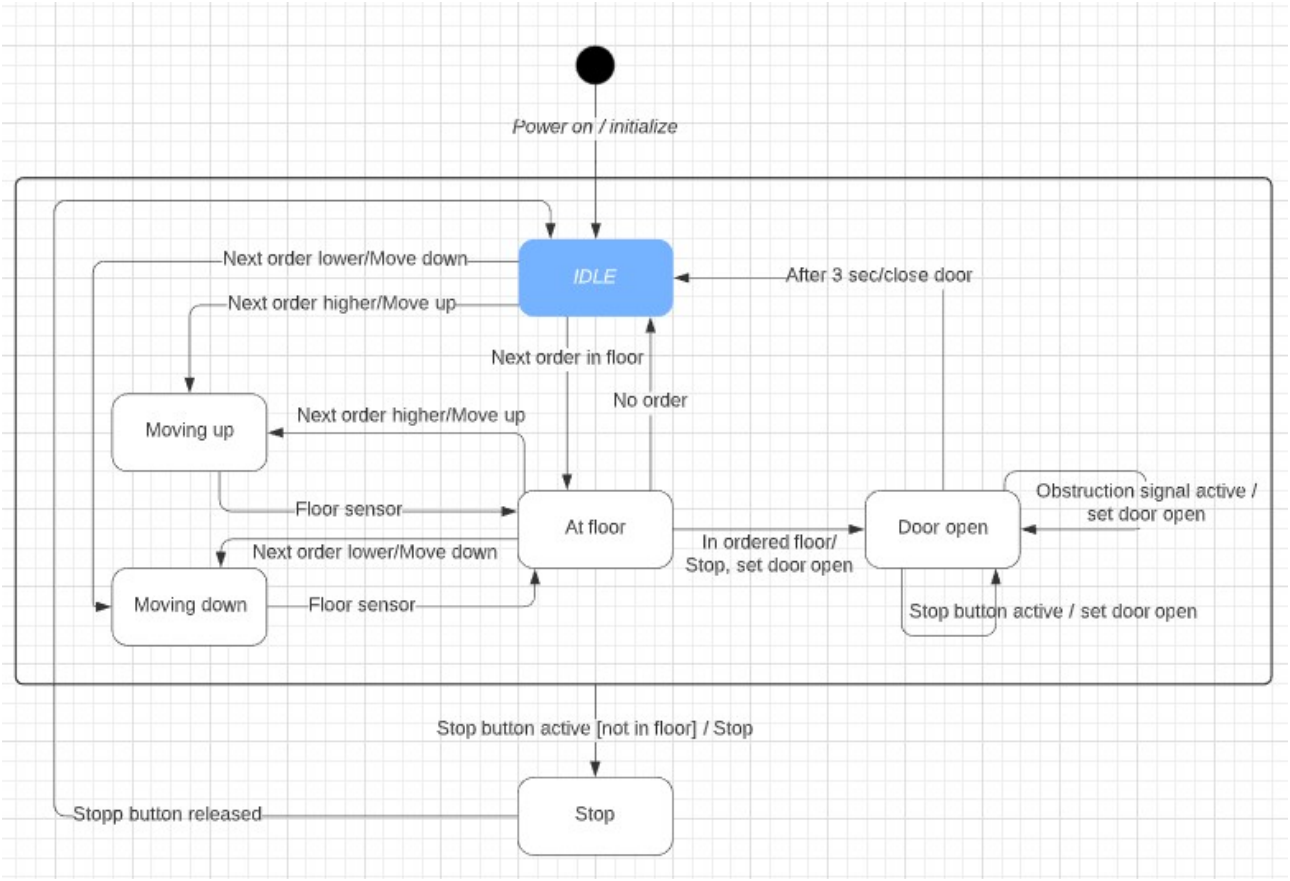
Figur 1: Klassediagram

Sekvensdiagrammet forklarer hvordan de forskjellige modulene fungerer sammen i en gitt sekvens(Måtte dele opp steg 3, da det ble for langt for programmet vi lagde diagrammet i).

Tilstandsdiagrammet i figur 3 viser tilstandene heisen kan være i. Vi har valgt å definere seks tilstander, "IDLE", "Moving up", "Moving down", "At floor", Door open"og Stop". Det er verdt å nevne at diagrammet er noe uklart ved oppstart av heisen. Initialize-handlingen vil sette heisen rett i IDLE dersom den er i en etasje, men om den ikke er det, vil heisens tilstand gå over til "Moving upfrem til neste etasje, også videre til IDLE. Dette skjer uten at heisen tar imot bestillinger, og er derfor ikke konsistent med tilstandsdiagrammet. Derfor er denne prosessen gjort til en egen handling.



Figur 2: Sekvensdiagram



Figur 3: Tilstandsdiagram

Valget av arkitekturen ble satt tidlig i prosjektet, da vi følte det var den beste måten å gjøre det på etter å ha lest kravspesifikasjonene under planleggingen. Ved å separere køsystemet og bevegelsen har vi to forskjellige moduler som kun fokuserer på en ting hver. I køsystemet tar man kun bestillinger og setter disse i riktig rekkefølge, mens bevegelsesmodulen setter på en bevegelse kun ut ifra hva det første i køen er. Bevegelsen bryr segdermed ikke om sortering av rekkefølger og det andre som foregår i køsystemet. Vi synes dette er en god arkitektur, siden det minimerer grensesnittet mellom de to modulene da ikke alt avhenger av hverandre. I tillegg valgte vi å ha en timer-modul som kun kommuniserer med bevegelses-modulen. Dette er fordi timeren aldri skal bli kalt fra køsystem-modulen.

2 Moduldesign

Køsystemet ble satt ved å se på hva det trengte å ta hensyn til. Heisen tar imot tre forskjellige bestillingstyper, bestilling opp utenfra, bestilling ned utenfra og bestilling fra inne i heisen. Disse kan ikke behandles på samme måte, og trengs å skilles. Derfor lagde vi en typedef struct kalt Orders, som inneholder fire int-arrays med fire elementer hver, og en neste-etasje-variabel. Tre av array-ene er for å skille mellom bestillingstyper, og det siste for rekkefølgen heisen skal ta

bestillinger i. Denne rekkefølgen oppdateres kontinuerlig for å sjekke om det er kommet inn bestillinger som har høyere prioritet.

En noe ulogisk implementasjon er at alle fire array-ene har fire elementer, til tross for at det er kun tre etasjer man kan bestille heisen nedover eller oppover i (ettersom det ikke vil være bestillinger ned i 1. eller opp i 4.). Dette var for å holde indekseringen lik i alle tilfeller, noe som er mer lesbart, og vil være lettere å implementere. Det vil si at et element i disse to array-ene aldri vil endre seg.

For å konstant ha en oversikt over posisjon og tilstanden til heisen valgte vi å lage en typedef struct i movement, kalt Elevator. Denne inneholder informasjon om døra, hvilken etasje heisen er i (eller var i sist), og hvilken retning den kjører i, eventuelt om den står stille. Dette gjør informasjonen om hvordan bevegelse heisen må settes i for å nå neste prioriterte etasje enkel å få tak i. I tillegg blir det lett å sjekke om alle betingelser for å kunne bevege heisen er tilfredsstilt. Retningen heisen kjører i er implementert som et enum for å forbedre lesbarheten.

Vi valgte å implementere sorteringsfunksjonen i movement-modulen. Dette er det to grunner til. Movement.h avhenger kun av køsystemet gjennom neste etasje-variabelen. Vi ønsket å beholde denne minimale avhengigheten for å kunne enklere endre på denne modulen uavhengig av movement, dersom det var ønskelig. Samtidig vil en slik sortering avhenge av hvor heisen befinner seg. Da trenger funksjonen tilgang til den nåværende tilstanden til heisen, som ligger i Elevator-structen.

Timermodulen tar utgangspunkt i tidspunktet da den blir kalt på, ettersom det er kun trengs en timer av gangen. Timeren vil kun bli starta når heisen er i en etasje og skal åpne døren. Derfor er det viktig å kunne starte timeren på nytt så lenge obstruksjonsbryteren er aktiv, eller stoppknappen er trykket inn. For å klare dette har vi implementert timeren som en funksjon som sjekker tidsdifferansen mellom starttidspunktet og nå, og en funksjon som resetter starttiden så lenge stoppknappen eller obstruksjonen er aktiv. I tillegg må heisen kunne ta bestillinger samtidig som døra er oppe. For å opprettholde dette har vi laget et funksjon som kalles imens den venter på at det har gått tre sekunder som fortsetter å ta bestillinger.

3 Testing

Undervies i implementasjonen av køsystemet, testet vi om de forskjellige funksjonen som skulle legge til, sortere og slette bestillinger, fungerte som ønsket. Dette gjorde vi ved bruk av debugging for å sette verdier og sjekke hva som skjedde. Ved eventuelle feil, så vi i GDB hvor dette var, og fikk rettet det opp. Dette var derimot vanskeligere på movementmodulen, ettersom at den fysiske heisen ble satt på pådrag, og var avhengig av at funksjoner ble kalt og utført ved nøyaktige tidspunkt. Dette kunne vært løst ved å koble ut pådraget, men dette visste vi ikke var mulig før mot slutten av prosjektet. Det vi derimot fikk debugget av movement-funksjonene med GDB

var å sette på/skru av forskjellige lys og når diverse funkjsoner ble kalt under kjøring. Dette hjalp oss med å se hva som eventuelt var feil med koden, for så å rette opp i dette.

For å teste movement sjekket vi hvordan heisen oppførte seg og observerte responsen fra funksjonene vi implementerte. Dersom oppførselen var feil, endret vi på noe, og så prøvde igjen. Da vi så at heisen bevegde seg riktig, og lysene oppførte seg som vi ønsket, var vi fornøyde med modulen.

Timermodulen måtte vi teste etter movementmodulen, da den blir kalt derfra. Vi sjekket om døren oppførte seg som ønsket når den ankom en etasje, noe den gjorde. Vi satte opp egne tester som til sammen dekte alle punktene i spesifikasjonen, og utførte de.

Test	Beskrivelse
T 1	Starter heisen mellom to etasjer og gjør bestillinger
T 2	Starte heisen utenfor definert funksjonsområde
T 3	Bestille heisen til etasje 4, bestille 1 innenfra og opp, bestille ned i 3, bestille opp i 2. Sjekke om heisen står stille etter å ha vært i alle fire etasjene en gang.
T 4	Trykke på alle tre bestillingsknappene for samme etasje (enten 2. eller 3.) og sjekke om alle lys tennes og slukkes når heisen ankommer etasjen.
T 5	Stå i 1. etasje og bestille heisen til 4. Følge med på etasjeindikatorlys.
T 6	Trykke inn stoppknapp, se at den lyser og heisen stopper og se at lyset slukkes når knappen slippes.
T 7	Gjør en bestilling, når heisen ankommer sjekk om døra åpnes i tre sekunder og holder seg lukket etterpå.
T 8	Trykk stoppknappen mens heisen er i en etasje, sjekk om døren åpner seg og holdes åpen så lenge stoppknappen er aktivert. Deretter slipp knappen og sjekk om døra holdes åpne i ytterligere 3 sekunder etter at stoppknappen er sluppet for å så lukke seg.
T 9	Gjør en bestilling, når heisen ankommer etasjen og åpner døra, aktiver obstruksjonsbryteren. Sjekk at døra holdes åpen og gjør obstruksjonssignalet lav og sjekk at døren lukkes etter 3 sekunder.
T 10	Bestill fra 1. etasje til 4. og tilbake til 1. etterpå. Sjekk at heisen ikke beveger seg utenfor dette området og at døra kun åpnes når heisen er i ro i 1. og 4.
T 11	Gjør flere bestillinger og på vei til en etasje, trykk og hold stopknappen samtidig som du gjør flere bestillinger. Sjekk at døra ikke åpnes. Slipp deretter knappen og sjekk om den holder seg i ro. Trykk så på en etasje, og sjekk om dette er eneste etasje den går til.
T 12	Gjør flere bestillinger, i første etasje heisen stopper i, hold stopknappen samtidig som du gjør flere bestillinger. Sjekk at døra åpnes. Slipp deretter knappen og sjekk om den holder seg i ro . Trykk så på en etasje, og sjekk om dette er eneste etasje den går til.
T 13	La heisen stå i en etasje med døra lukket. Gjør obstruksjonsbryteren aktiv, og deretter gjør en bestilling. Sjekk at dette ikke forhindres og at døra er igjen.
T 14	La heisen være utenfor en etasje og gjør obstruksjonsbryteren aktivt. Sjekk at oppførselen ikke påvirkes før døra åpnes i en etasje.
T 15	Generell observasjon av heisens oppførsel

Punkt	Beskrivelse	Test
	Oppstart	
O1	Ved oppstart skal heisen alltid komme til en definert tilstand. En definert tilstand betyr at styresystemet vet hvilken etasje heisen står i.	T 1
O2	Om heisen starter i en udefinert tilstand, skal heissystemet ignorere alle forsøk på å gjøre bestillinger, før systemet er kommet i en definert tilstand.	T 1
O3	Heissystemet skal ikke ta i betraktning urealistiske startbetingelser, som at heisen er over fjerde etasje, eller under første etasje idet systemet skrus på	T 2
	Håndtering av bestillinger	
H1	Det skal ikke være mulig å komme i en situasjon hvor en bestilling ikke blir tatt. Alle bestillinger skal betjenes, selv om nye bestillinger opprettes.	T 3
H2	Heisen skal ikke betjene bestillinger fra utenfor heisrommet om heisen er i bevegelse i motsatt retning av bestillingen.	T 3
H3	Når heisen først stopper i en etasje, skal det antas at alle som venter i etasjen går på, og at alle som skal av i etasjen går av. Dermed skal alle ordre i etasjen være regnet som ekspedert.	T 3
H4	Om heissystemet ikke har noen ubetjente bestillinger, skal heisen stå stille.	T 3
	Bestillingslys og etasjelys	
L1	Når en bestilling gjøres, skal lyset i bestillingsknappen lyse helt til bestillingen er utført. Dette gjelder både bestillinger inne i heisen, og bestillinger utenfor.	T 4
L2	Om en bestillingsknapp ikke har en tilhørende bestilling, skal lyset i knappen være slukket.	T 4
L3	Når heisen er i en etasje skal korrekt etasjelys være tent.	T 5
L4	Når heisen er i bevegelse mellom to etasjer, skal etasjelyset til etasjen heisen sist var i være tent.	T 5
L5	Kun ett etasjelys skal være tent av gangen.	T 5
L6	Stoppknappen skal lyse så lenge denne er trykket inne. Den skal slukkes straks knappen slippes.	T 6

Døren		
D1	Når heisen ankommer en etasje det er gjort bestilling til, skal døren åpnes i 3 sekunder, for deretter å lukkes.	T 7
D2	Om heisen ikke har ubetjente bestillinger, skal heisdøren være lukket.	T 7
D3	Hvis stoppknappen trykkes mens heisen er i en etasje, skal døren åpne seg. Døren skal forholde seg åpen så lenge stoppknappen er aktivert, og ytterligere 3 sekunder etter at stoppknappen er sluppet. Deretter skal døren lukke seg.	T 8
D4	Om obstruksjonsbryteren er aktivert mens døren først er åpen, skal den forbli åpen så lenge bryteren er aktiv. Når obstruksjonssignalet går lavt, skal døren lukke seg etter 3 sekunder.	T 9
Sikkerhet		
S1	Heisen skal alltid stå stille når døren er åpen.	T 10, T 15
S2	Heisdøren skal aldri åpne seg utenfor en etasje.	T 11, T 15
S3	Heisen skal aldri kjøre utenfor området definert av første- og fjerde etasje.	T 10
S4	Om stoppknappen trykkes, skal heisen stoppe momentant.	T 6
S5	Om stoppknappen trykkes, skal alle heisens ubetjente bestillinger slettes.	T 11, T 12
S6	Så lenge stoppknappen holdes inne, skal heisen ignorere alle forsøk på å gjøre bestillinger.	T 11, T 12
S7	Etter at stoppknappen er blitt sluppet, skal heisen stå i ro til den får nye bestillinger.	T 11, T12
Robusthet		
R1	Obstruksjonsbryteren skal ikke påvirke systemet når døren ikke er åpen.	T 13, T 14
R2	Det skal ikke være nødvendig å starte programmet på nytt som følger av eksempelvis udefinert oppførsel, at programmet krasjer, eller minnelekkasje.	T 15
R3	Etter at heisen først er kommet i en definert tilstand ved oppstart, skal ikke heisen trenge flere kalibreringsrunder for å vite hvor den er.	T 15
Ymse		
Y1	Oppførsel som ikke er «vanlig heisoppførsel» kan gi trekk på FATen. Ikke tenk for komplisert her. Dere har alle tatt en heis før, og vet hvordan de bør funke.	T 15

For å teste hvordan de forskjellige modulene fungerte sammen med de andre modulene, testet vi hele programmet sammen. Vi sjekket om alle de forskjellige kravspesifikasjonene var dekt.

Ved å utføre alle testene, og så at alle krevene ble dekt, vet vi at heisen fungerer som gitt i kravspesifikasjonene.

4 Diskusjon

Vi endte med et system med ganske få moduler. Dette kan være en svakhet i programmet vårt, siden både movement og køsystemet inneholder mange forskjellige funksjoner som utfører forskjellige små-oppgaver. Disse er i stor grad avhengig av hverandre inni hver modul, som gjør programmet vanskelig å vedlikeholde i senere tid. Vi kunne gjort dette anderledes ved å ha flere, og mindre, moduler kun har en oppgave/handling den skal utføre, og kun gjør den. Da ville det for det første blitt mer oversiktlig, men også lettere å endre på eller vedlikeholde senere. Likevel fungerer programmet med disse modulene, slik som vi tenkte i arkitetkurdesign-fasen.

Systemet vårt tar i høy grad utgangspunkt i de to structene Elevator og Orders. Funksjonene i de forskjellige modulene tar inn disse som parametere, til tross for at de sjeldent trenger hele structen som en parameter. Et forbedringspotensial er dermed å gjøre funksjonene mer generelle, og endre parameterne. Dette vil føre til enklere vedlikehold og utvidelse eller endring i programmet, og mindre avhengighet mellom modulene. Vi brukte dette fordi vi syntes det var oversiktlig å se hvilken struct, og eventuelt medlemsvariabel, som ble lest eller endret på, men ser i ettertid at dette kan endres.

Vi valgte å ikke ha en egen modul for stoppknappen, da vi tenkte det ville være greiere å kun kalle på den der den trengtes. Dette viste seg bli mer krevende enn vi trodde, da stoppknappen måtte ta hensyn til flere forskjellige tilstander til heisen, og også gjøre flere handlinger selv. Vi endte dermed opp med mye koding for å få stoppknappen til å oppføre seg ideelt, noe som kunne blitt gjort i en egen modul som ville gitt mindre kode i movement-modulen (der vi nå har implementert stopp-funksjoner). Dette ville også gjort det lettere å endre på funksjonaliteten til stoppknappen senere dersom det skulle være av interesse. Selv om det kunne ført til bedre kode ved å endre denne implementasjonen, valgte vi å beholde det vi hadde, siden det viste tydelig hva som skjedde når stoppknappen ble aktivert i de forskjellige situasjonene.