Labrapport: Heisprosjekt

av Anna Aurora Haukefær og Nina Tran

19. mars 2021

TTK4235 TILPASSEDE DATASYSTEMER
Norges Teknisk-Naturvitenskapelige Universitet

Innhold

Innhold **i**

1 Sammendrag **1**

2 Overordnet arkitektur **1**

2.1 Klassediagram 1

2.2 Sekvensdiagram 2

2.3 Tilstandsdiagram 4

3 Moduldesign **5**

3.1 Elevator 5

3.2 Queue 5

3.3 Timer 5

3.4 Controller 6

3.5 Hardware 6

4 Testing **6**

4.1 Enhetstesting 6

4.2 Systemtesting 7

5 Diskusjon **8**

1 Sammendrag

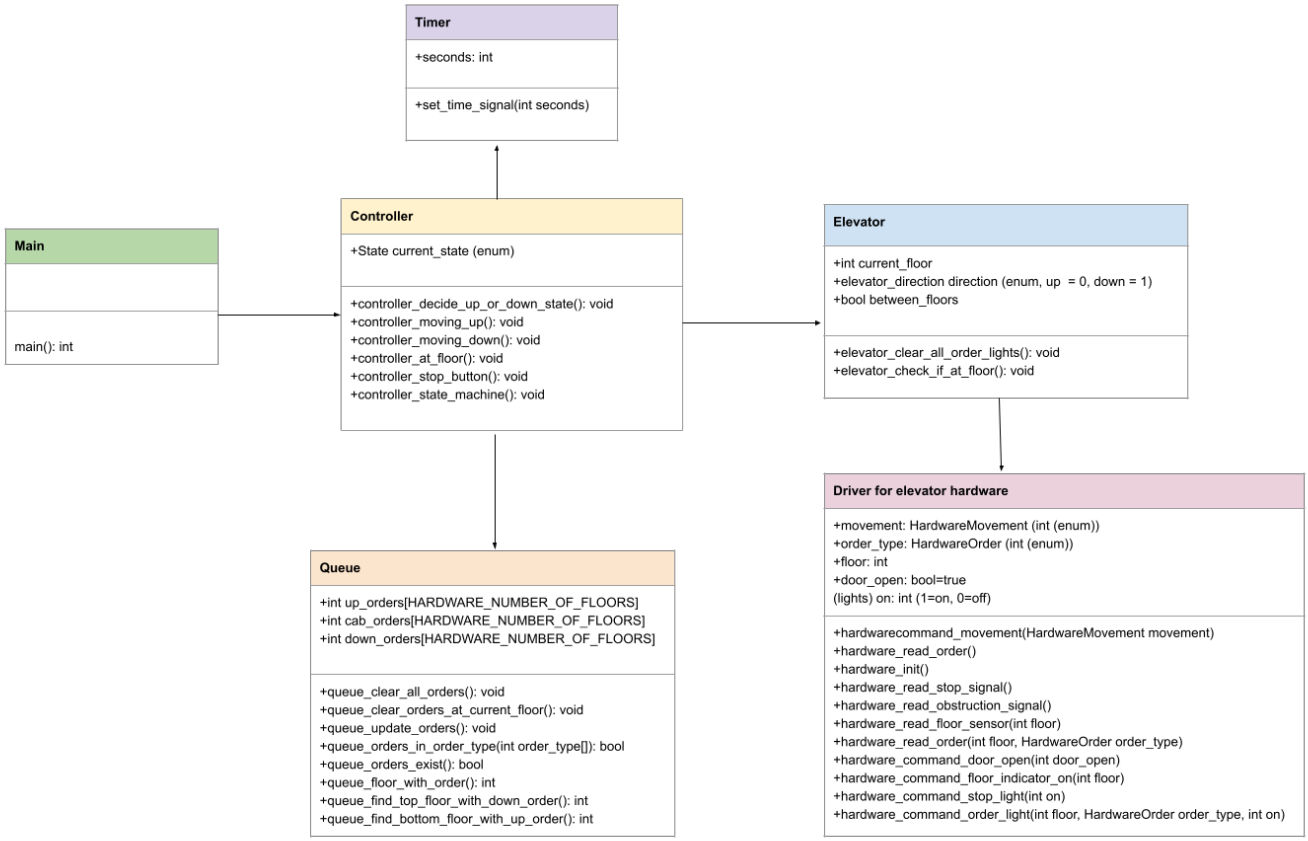
I dette heisprosjektet brukes V-modellen til å implementere et styringssystem for en heis, med gitte kravspesifikasjoner. UML brukes til design og dokumentasjon av systemet, og progrommeringsspråket C brukes til selve implementasjonen. Denne labrapporten viser den overodnede arkitekturen, en beskrivelse av modulene, testprosedyren og til slutt en refleksjon over systemet med mulige forbedringer.

2 Overordnet arkitektur

Vi har valgt å dele inn systemet i fem moduler: *elevator*, *queue*, *timer* og *controller*, i tillegg til den utdelte *hardware-modulen*. Elevator-modulen henter informasjon om etasjen heisen står i og hvilken retning den har. Queue-modulen håndterer bestillingene og henter informasjon om dem. Timer-modulen brukes til timing av døren. Alle disse modulene kobles sammen i Controller-modulen, der selve logikken til heisen er. Hver modul representerer hver sin fil i koden, slik blir det lettere å holde oversikt. Vi har lagt vekt på å ha færre moduler med flere funksjoner i, enn flere moduler med færre funksjoner. Dermed vil det være lettere å ha kontroll på den overordnede arkitekturen av systemet, i tillegg til at færre moduler snakker sammen.

2.1 Klassediagram

Figur 1 viser klassediagrammet til systemet, den viser hver av modulene som inngår i arkitekturen og hvordan de henger sammen. Inndelingen baseres på at funksjoner av samme funksjonalitet skal være samlet på et sted slik at koden blir ryddigst mulig og mer forståelig for andre utviklere. Utifra klassediagrammet kan man se at både Queue- og Elevator-modulen tar inn funksjoner fra den utdelte Hardware koden. Controller-modulen tar inn funksjoner fra Timer-, Elevator-, og Queue-modulen til å utforme et fullverdig styringssystem for heisen.

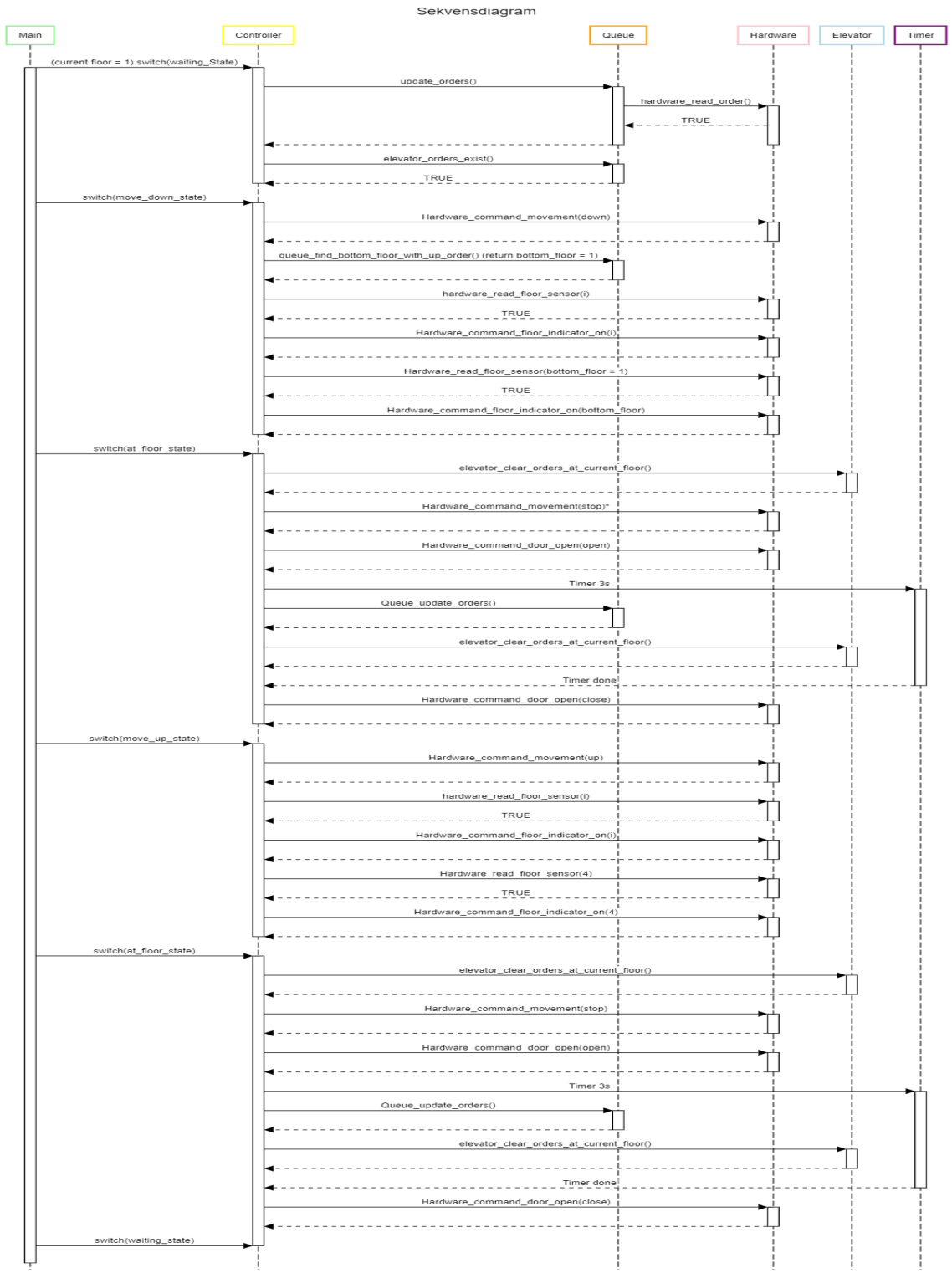


Figur 1: Klassediagram for systemet

2.2 Sekvensdiagram

Figur 2 viser sekvensdiagrammet for systemet. Det viser hvordan de ulike modulene snakker sammen i den utgitte sekvensen. Diagrammet viser følgende sekvens:

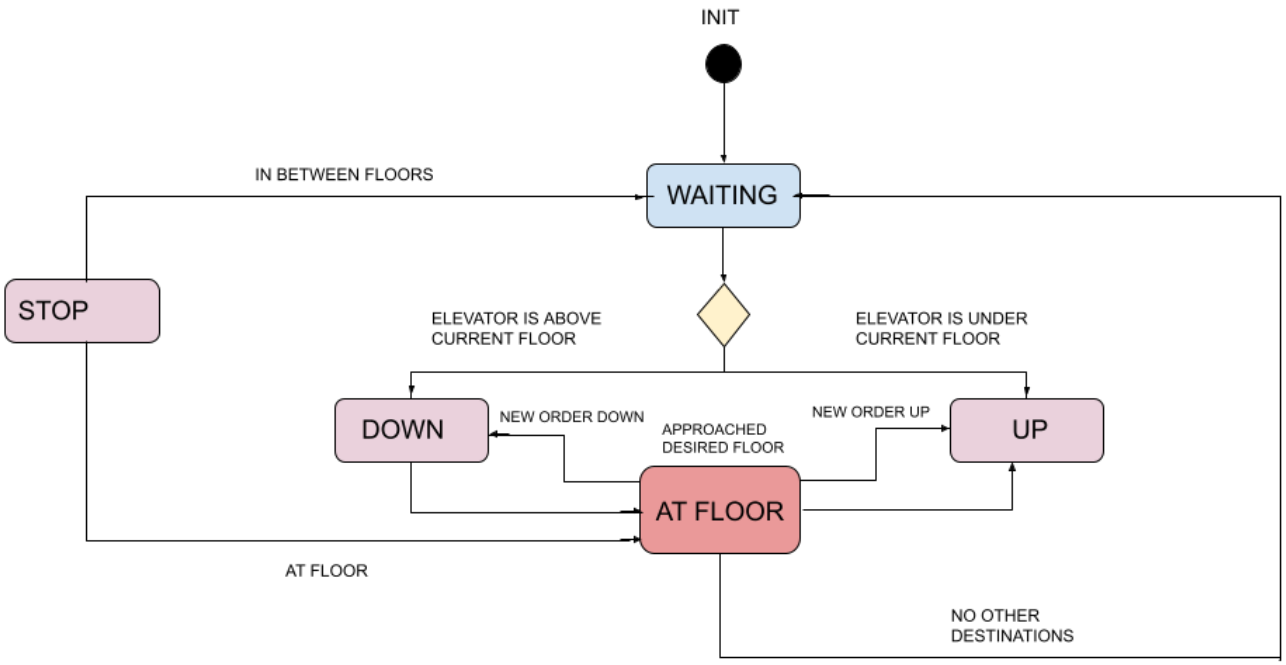
1. Heisen står stille i 2. etasje med døren lukket.
2. En person bestiller heisen fra 1. etasje
3. Når heisen ankommer går personen inn i heisen og bestiller 4. etasje
4. Heisen ankommer 4. etasje og personen går av
5. Etter 3 sekunder lukker døren til heisen seg



Figur 2: Sekvensdiagram for systemet

2.3 Tilstandsdiagram

Figur 3 viser tilstandsdiagrammet til systemet. Den viser hvordan heisen oppfører seg basert på hva som sendes inn i systemet og hva heisen selv setter ut. Altså hva som bestemmer hvilken state heisen går inn i. Vi tok utgangspunkt i de utgitte krav -spesifikasjonene fra oppgaveteksten da vi satte opp diagrammet, for å speile heisens oppførsel på best mulig måte. Måten vi har satt det opp er basert på det vi synes ser mest oversiktlig og hensiktsmessig ut, utifra systemets logikk.



Figur 3: Tilstandsdiagram for systemet

3 Moduldesign

Som nevnt tidligere har vi valgt å dele systemet inn i fem moduler. Vi har valgt å kalle alle funksjoner i de ulike modulene med samme modulnavn først. Vi har lagt vekt på å lage lengre funksjonsnavn, men mer beskrivende, enn korte. Slik vil det være lettere for andre å forstå funksjonenes funksjonalitet.

Systemets tilstandsvariabler er organisert i Elevator-modulen, sammen med et par tilhørende funksjoner. Heisens ordre er organisert i arrays i queue-modulen, sammen med funksjoner som oppdaterer og henter diverse informasjon ut av ordrene. I controller-modulen er den nåværende staten en global variabel, og den oppdateres i de tilhørende funksjonene. Det er en funksjon for hver aktuelle state, og de er alle organisert i state-machine-funksjonen.

3.1 Elevator

Elevator-modulen har tre globale variable: current floor, direction og between floors. Current floor og directon oppdateres kontinuerlig slik at heisen alltid skal vite hvilken etasje den er i og hvilke retning den har/sist hadde. Between floors er en global variabel som er sann dersom heisen blir stoppet med stopp-knappen mellom to etasjer. Modulen har også to tilhørende funksjoner, elevator_clear_all_order _lights(), som skrur av bestillingsslys, og elevator_check_if_at_floor(), som sjekker om heisen er i en etasje.

3.2 Queue

Queue-modulen tar hånd om bestillingene. Vi har laget tre globale arrays, en for hver av ordretypene, som inneholder heisens kalte ordre. I tillegg bruker de tilhørende funksjonene alle ordre-arraysene, enten for å oppdatere dem, eller hente ut informasjon av dem.

3.3 Timer

Timer-bibliotekets funksjoner har vi ikke skrevet selv, men vi har brukt de i at_floor _state og stop_state for å holde døren åpen i tre sekunder. Vi brukte clock() funksjonen for å ta inn tiden, og definerte start- og sluttid som vi brukte i en do-while-løkke. Samtidig som den oppdaterer ordre, sjekker for stop-knapp og obstruction.

3.4 Controller

Controller-modulen styrer heisens logikk og inneholder systemets state machine. Vi har laget en typedef enum State, som inneholder heisens mulige tilstander og en global variabel current state, av typen State. Vi har skrevet en funksjon for hver av tilstandene som både bestemmer hva heisen skal gjøre i gitt state, men også bestemmer neste state utifra diverste forutsetninger. Eksempelvis hvilken retning og nåværende etasje heisen er i. Til slutt er det en funskjon, controller_state_machine() som bruker disse funksjonene i en switch-case-funksjon. Den styrer heisens tilstand utifra current state, som oppdateres i hver av tilstandene.

3.5 Hardware

Hardware-biblioteket fungerer som en API-forbindelse til heis-driveren. Siden vi ikke har skrevet koden til dette biblioteket kommer vi ikke til å kommentere det ytterligere.

4 Testing

For å forsikre oss om at heisen faktisk fungerer har vi gjort en rekke tester, både tester av hver enkelt state og tester av hvordan disse fungerer sammen ved å teste hele systemet. Vi har laget et system der hver state er relativt uanvhengig av hverandre, og det var derfor ikke så vanskelig å teste hver state for seg uten at det påvirket resten av systemet.

4.1 Enhetstesting

For å teste hver enkelt state underveis i implementeringen, la vi inn print-setninger for å se hva systemet returnerte hvor. Dermed kunne vi finne ut om funksjonen gjorde det den faktisk skulle. Vi brukte også GDB debugger. Da satt vi breakpoints i den aktuelle staten ved de aktuelle områdene i koden, for så å bruke print-funksjonen i GDB til å finne hvordan heisens tilstander, altså direction, current floor osv. hadde endret seg. Dette ga oss en god pekepinn til hva som eventuelt var feil. Videre kunne vi fortsette programmet med nye inputs, stoppe ved breakpointe og eventuelt printe tilstandene på nytt. Det var slik vi testet og debugget blant annet funksjonene i waiting state og move-statesene.

4.2 Systemtesting

For å teste hvordan alle modulene fungerte sammen, altså hele styringssystemet, brukte vi printf som viste oss når heisen gikk inn i ulike states. Så testet vi heisen i ulike scenarioer og kombinasjoner av bestillinger. Dette er den generelle fremgangsmåten vår for å sjekke om heisen dekket alle de gitte kravspesifikasjonene:

- **Oppstart**: Ved oppstart mellom to etasjer vil heisen gå til en definert tilstand, altså kjøre til den mest gunstige etasjen i forhold til hvor heisen står. I tillegg til at den ignorer alle bestillinger i udefinert tilstand. Dermed er punkt O1-O2 dekket.
- **Håndtering av bestillinger**: For å teste systemets håndtering av bestillinger trykker vi på både cab orders i heisen, men også bestillinger utenfor heisen. Vi observerer at heisen tar i mot alle bestillingene, i riktig prioritert rekkefølge. Når den har betjent alle bestillingene står den stille, til den får nye bestillinger. Dermed er punkt H1-H4 dekket.
- **Bestillings- og etasjelys**: For å teste bestillings- og etasjelysene utførte vi samme test som punktet over. Vi observerer at når en bestilling gjøres, lyser bestillingsknappen helt til den er utført. Ser at dette stemmer både for cab orders og for lysene utenfor heisen, samtidig som kun ett etasjelys er tent av gangen. Dersom heisen er mellom to etasjer, observerer vi at etasjelyset til etasjen heisen sist var i lyser. Når heisen ankommer riktig etasje vil tilhørende etasjelys være tent. Dersom vi trykker inne stopp-knappen vil denne lyse, helt til vi slipper. Dermed er punkt L1-L6 være oppfylt.
- **Heisdør**: Under testingen av punktene over følger vi også med på døren og observerer at den åpner seg i tre sekunder når den ankommer riktig destinasjon, før den lukker seg. Ved test av stoppknappen, når heisen er i en etasje, ser vi at døren åpner seg så lenge vi holder knappen inne. Det går tre sekunder fra vi slipper knappen til døren lukker seg igjen. Vi tester også obstruksjonsbryteren samtidig som døren er åpen. Døren er åpen så lenge bryteren er aktiv, når den skrus av tar det tre sekunder før døren lukkes igjen. Dermed er punkt D1-D4 tilfredsstilt.
- **Sikkerhet**: For å teste sikkerheten i systemet, trykker vi på stoppknappen i ulike situasjoner. Både når den er i en etasje, og når den står mellom to etasjer. Observerer da at heisen stopper momentant, de ubetjente bestillingene slettes og det er ikke mulig å gjøre nye bestillinger så lenge knappen er inne. Når knappen slippes står heisen stille til den får en ny bestilling. Dermed er punkt S4-S7 tilfredsstilt. Punkt S1-S3 ble observert under testing av punktene over og er derfor oppfylt.
- **Robusthet**: For å teste systemets robusthet aktiverer vi obtruksjonsbryteren samtidig som heisen kjører, og observerer at dette ikke har noen påvirkning. Observerer også ved oppstart at systemet vet hvor heisen er når den har kommet til en definert tilstand. Dermed er punkt R1-R3 tilfredsstilt.

Disse testene utførte vi flere ganger for å forsikre om at systemet oppfylte kravene hver gang.

5 Diskusjon

Siden vi kontinuerlig testet koden underveis var det lettere å debugge og oppdage feil ved systemet. Ved å aktivt bruke V-modellen, altså teste etter hver implementering av modulene, sparte vi en del tid når vi skulle teste hele systemet. Likevel ble det oppdaget noen svakheter ved styringssystemet når vi testet for noen spesielle situasjoner.

Noe av det første vi oppdaget var at når vi trykket på stopp-knappen mellom to etasjer, for så å sende bestilling til det som sist var current floor, så tolket heisen det som at den fikk en bestilling i den etasjen den allerede var i. Videre gikk den fra waiting state til floor state, og åpnet dørene mellom to etasjer. Dette prøvde vi å fikse ved å legge til en ny variabel, *between floors*, en bolsk global variabel som ble oppdatert i stop state. Hvis denne var sann, skulle heisen gå til move down state dersom siste retning var opp, og move up state dersom siste retning var ned. Denne løsningen viste seg å ha en del svakheter.

Dersom man trykket på stopp-knappen og en bestilling flere ganger etter hverandre, før heisen fikk nådd frem til etasjen, skiftet heisen retning for hver bestilling. Videre gikk den helt til endestoppbryteren, og det krevdes en heis-restart dersom den gikk i feil retning. Dette løste vi igjen ved å sjekke i move statene om det var en bestilling over/under nåværende etasje, for så å breake og gå til waiting state dersom dette ikke var tilfelle.

En annen logisk brist som vi oppdaget var at dersom heisen hadde flere down-orders over seg, eller flere up-orders under seg, så kjørte den til nærmeste etasje med den typen ordre. For så å fortsette i samme retning, til alle bestillinger var tatt. Vi synes det er mer logisk at heisen skal kjøre til øverste down-bestilling, eller nederste up-bestilling, da det kan være at noen som går på skal av i de neste etasjene. Dette løste vi ved å lage to funksjoner - en som returnerte den øverste down order og en som returnerte den nederste up order. Disse verdiene ble brukt i move up state og move down state.

En annen ting vi burde ha gjort annerledes i starten av kode-implementasjonen hadde vært å sorterere heisens tilstander i en struct, istedenfor å ha egne globale variabler for hver av dem. Det hadde gjort det lettere når vi skulle debugge, for da kunne vi bare printet hele structen med GDB-print-funksjonen, i tillegg til at det hadde gjort koden mer oversiktlig og leselig.

Alt i alt har vi laget et fungerende heissystem. Koden kunne nok vært mer generalisert og mer leselig, men alle de gitte kravspesifikasjonene er oppfylt. Dette har vært et lærerikt prosjekt.