# Heislab

Embla Flatlandsmo og Ragnhild Kvist Simonsen

Mars 2020

# 1 Overordnet arkitektur

Til designet av heisen vår har vi valgt å implementere tre moduler: hovedmodulen *main* inneholder tilstandsmaskinen, i tillegg til at den tar seg av startfasen hvor heisen er i en udefinert tilstand. Kømodulen *ordermanager* tar seg av køsystemet, og forteller *main* om heisen skal opp eller ned, og når heisen skal stoppe. Til slutt har vi *hardware* modulen som fulgte med de utdelte filene, og er den som kommuniserer med den fysiske heisen. *Main* og *ordermanager* kommuniserer med hverandre på den måten at kømodulen tar inn "current floor", "driving direction", "between floors" og "order above" fra *main*, og bruker disse til å returnere en oppdatert "order above" til tilstandsmaskinen. Begge modulene ser på *hardware* for å få tilgang til funksjonene dens. Klassediagrammet med funksjoner og variabler som brukes i de ulike modulene vises i figur 1.

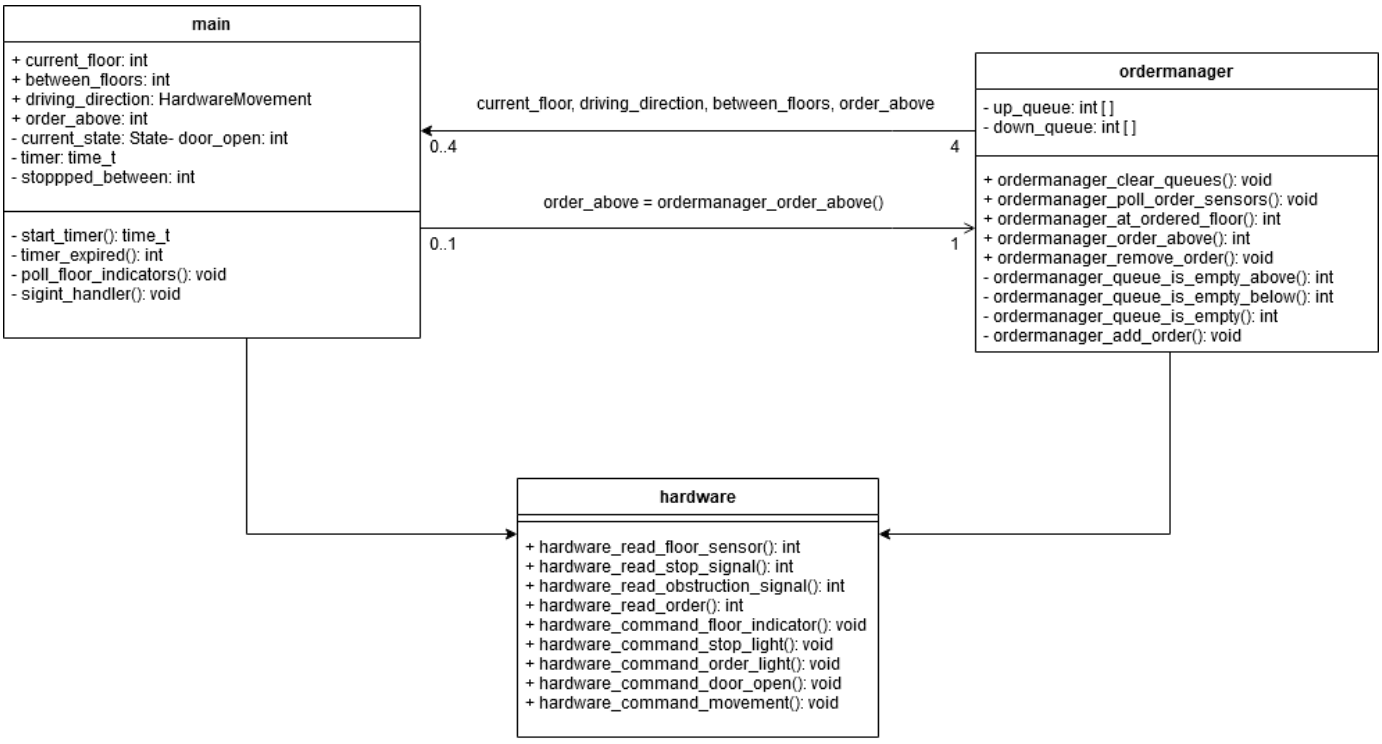


Figure 1: Klassediagram

I figur 2 viser vi et sekvensdiagram som beskriver følgende sekvens, og hvordan de ulike modulene snakker sammen:

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 dørene til heisen seg.

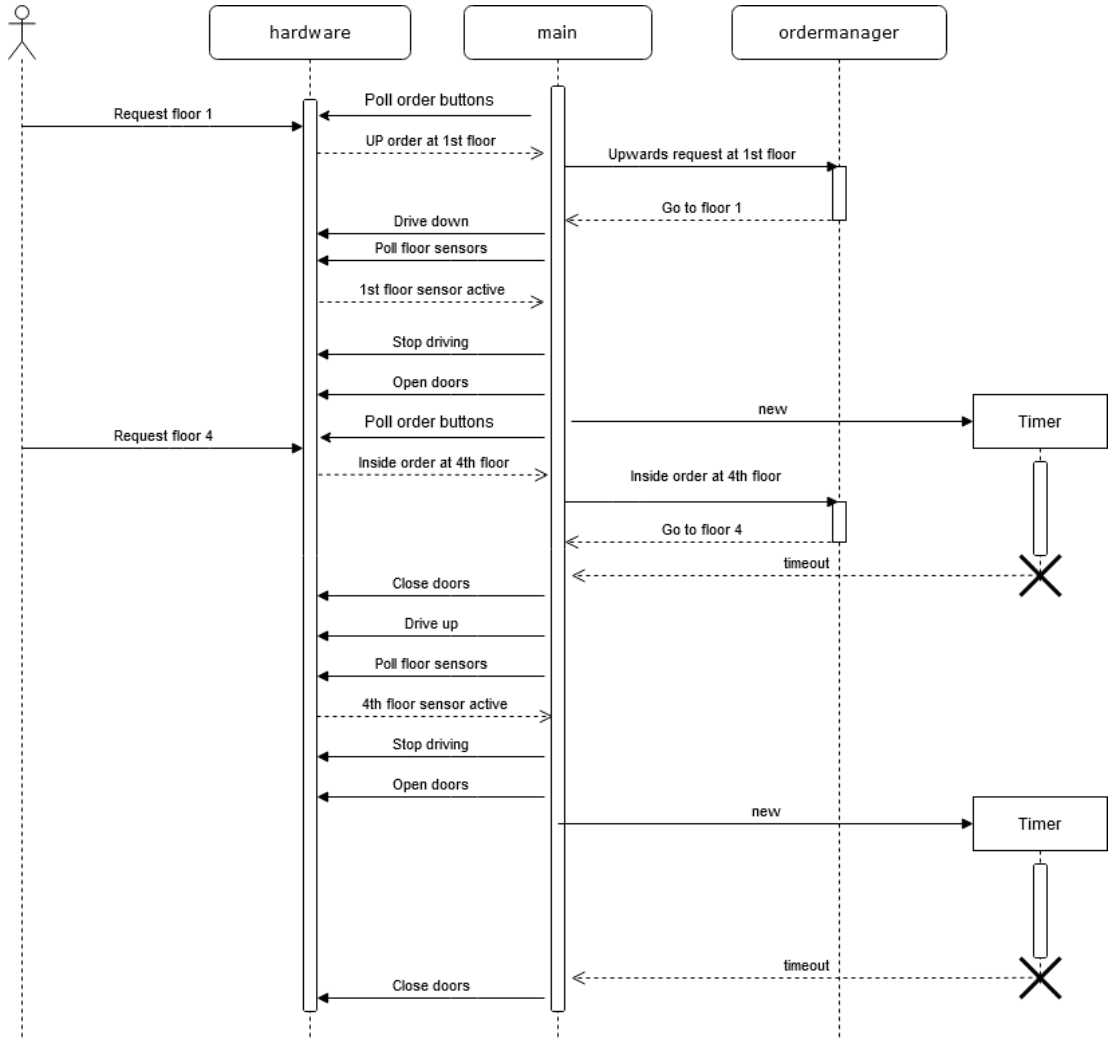


Figure 2: Sekvensdiagram

Her er *timer* en variabel som opprettes inni main, ikke en egen modul. Til slutt kan vi beskrive logikken til tilstandsmaskinen i et tilstandsdiagram, se figur 3. De ulike blokkene er da de ulike tilstandene vi har implementert, og pilene viser hvordan vi skifter mellom tilstander.

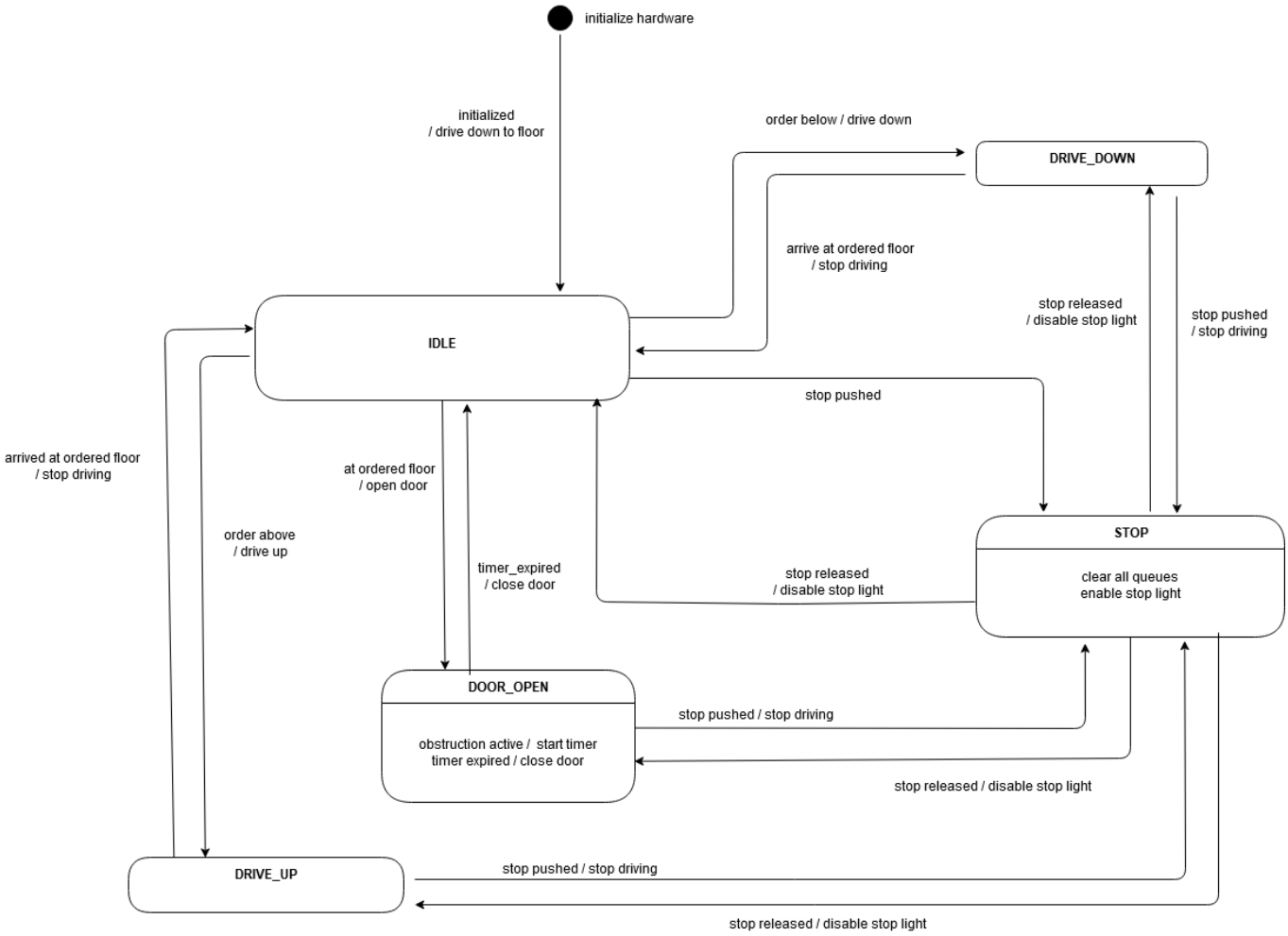


Figure 3: Tilstandsdiagram

Vi har endt opp med denne arkitekturen fordi den inneholder få moduler uten for mye avhengighet, og tilstandsmaskinen har de tilstandene som vi ser på som nødvendige for oversiktelig implementasjon. Selv om *main* og *ordermanager* ser på hverandre, har vi erfart at til tross for de store endringene som kømodulen kontinuerlig har vært gjennom, har det ikke gitt store utslag i "main". Vi mener dette viser at graden av avhengighet mellom modulene er ganske OK.

Opprinnelig hadde vi flere moduler, blant annet en egen timer modul, og main og tilstandsmaskinen var splittet i to egne moduler. Etter hvert fant vi ut at det ikke var en særlig god måte å gjøre det på for oss. Det var etter dette at vi endte opp med kun to egne moduler - en for å bestemme neste etasje,

og en for å bruke denne informasjonen til å gjøre som en heis gjør - og fant ut at det var en mye bedre løsning. Det ble mye lettere å feilsøke med GDB når tilstandsmaskinen lå i *main*. Tilstandsdiagrammet ser stort sett ut som det gjorde fra starten, og tilstandene kommer rett og slett fra at det er disse som er en vanlig heis sine viktigste funksjoner: stå stille, åpne/lukke døren, kjøre opp, kjøre ned og reagere på stopp-knappen.

# 2 Moduldesign

Nå skal vi ta for oss litt nærmere hvordan vi har implementert de ulike modulene og hvordan de virker.

I *ordermanager* har vi køsystemet, og modulen er ansvarlig for å bestemme om heisen skal kjøre opp, ned eller stå stille til enhver tid, i tillegg til at den skal si ifra til *main* når heisen skal stoppe. I starten prøvde vi å implementere dette med lenkede lister, men denne idèen var i vårt tilfelle å ta seg mye vann over hodet. Etterhvert skjønte vi dette, og da vi byttet til integer-arrays (et for "up_queue" og et for "down_queue") ble absolutt alt bedre. Akkurat nå har *ordermanager* ansvaret for å si ifra til *main* om neste ordre er over eller under heisens nåværende posisjon gjennom variabelen "order_above". *ordermanager* bestemmer også om heisen skal stoppe i den nåværende etasjen. Kort sagt vil *ordermanager* prioritere å stoppe for ordre i up_queue dersom vi kjører oppover, og stoppe for ordre i down_queue dersom vi kjører nedover. Hvis vi kjører oppover, men up_queue er tom, vil *ordermanager* så gå inn i down_queue og stoppe på den øverste ordren i down_queue. Det motsatte gjelder også for det liknende tilfellet hvor vi kjører nedover men down_queue er tom. Da vil heisen først stoppe på den nederste ordren i up_queue.

*main* er modulen hvor selve heisoppførselen blir implementert. Første del initialiserer hardware og fører heisen inn i en definert tilstand, hvor så tilstandsmaskinen tar over. Denne er implementert som en switch-case, som switcher på enum-typen "State". Den poller både ordre, etasje-indikatorene og stoppknappen kontinuerlig, og i stille-tilstanden "IDLE" tar den inn den oppdaterte "order_above" fra kømodulen. Denne variabelen forteller tilstandsmaskinen om den skal kjøre opp, ned eller forbli i nåværende posisjon. Det er alltid fra "IDLE" at tilstandsmaskinen bestemmer hva som er neste tilstand, med unntak av "STOP"-tilstanden som kan kalles når som helst. Dette kan en også se ut ifra tilstandsmaskindiagrammet i figur 3. I tilstandene "DRIVE_UP" og "DRIVE_DOWN" ser vi på "ordermanager" for å finne ut når heisen skal stoppe igjen. Dette gjør vi også i "IDLE", men da for å se om vi skal åpne døren.

# 3 Testing

Heisen har blitt grundig testet med den idiotsikre metoden vi har valgt å kalle "shit-kidding", som innebærer å trykke vilt på alle knappene helt til heisen jammer seg etter en kombinasjon du aldri hadde kommet på å teste bevisst. Vi har selvfølgelig også foretatt vanlige siviliserte enhets- og integrasjons-tester som vi skal fortelle litt om her:

I den lange prosessen som var å prøve å få heisen til å fungere som vi ville, har GDB vært et nyttig debuggings-verktøy. Måten vi stort sett har gått fram på er å kjøre heisen til vi ser oppførsel som ikke stemmer overens med slik det skulle ha vært, og så kjøre programmet med GDB for å peile oss inn på hvor feilen kan ligge. Typiske ting vi har sjekket med programmet er verdien av sentrale variabler som "order_above", innholdet i køene, "current_floor" og "between_floors", der den sistnevnte variabelen forteller oss om vi står over, under eller i en etasje. Med denne metoden har vi kunnet peile oss inn på problemer som for eksempel at ordre blir lagt inn i køen men at "order_above" ikke har oppdatert seg riktig, og dermed funnet ut av hvilken funksjon problemet kommer fra. Om en ønsker å lese om testing av de spesifikke kravspesifikasjonene er dette beskrevet under:

Ved oppstart av heisen vil den kjøre nedover helt til den registrerer en etasje, hvor den stopper og går inn i "IDLE". Hvis den allerede leser en etasje-sensor ved oppstart, blir den stående stille, og heisen går inn i "IDLE" med en gang. Dermed er **O1** oppfylt. I denne initialiserings-fasen registrerer ikke heisen bestillinger, og **O2** er oppfylt. I henhold til **O3** tar vi ikke urealistiske startbetingelser med i betraktning.

Prioriterings-rekkefølgen som er implementert i kømodulen vil sikre at **H1** er tilfredsstilt: Når vi er på vei oppover vil vi først prioritere "up_queue" over oss, deretter delen av "down_queue" som ligger over oss, og til slutt ordre som ligger under oss som blir beskrevet i avsnitt 2. Ved bevegelse nedover blir det vice-versa, og da ser vi at det ikke er mulig at vi for eksempel setter heisen fast til å kjøre mellom to etasjer så lenge det alltid kommer ordre her, som ville ha gjort at eventuelle andre bestillinger ble satt på vent for alltid. Når vi itererer gjennom opp-køen gjør vi det i stigende rekkefølge, mens vi gjør det i synkende rekkefølge på ned-køen. Dette og prioriterings-rekkefølgen gjør at ned-bestillinger ikke blir tatt med på vei opp og at opp-bestillinger ikke blir tatt med på vei ned, med unntak av når det ligger både opp- og ned-bestillinger på samme etasje. Dermed er **H2** også oppfylt. Når heisen stopper i en etasje og går inn i "DOOR_OPEN" tilstanden, fjernes ordren fra køen. Da er alle tilhørende bestillinger betjent og **H3** er oppfylt. Har heisen ingen bestillinger, står den stille i henhold til **H4**.

Bestillingslys blir satt når en bestilling legges inn i en kø, og det slukket når elementet slettes i forbindelse med at døren åpner seg eller stopp-knappen

slukkes. Dermed er **L 1-2** oppfylt. Når en etasjesensor blir lest settes tilhørende etasjelys, og dette slukkes kun når neste sensor blir lest og lyset oppdateres, slik at **L 3-5** er tilfredsstilt. Stopplyset blir satt hver gang vi går inn i "STOP" tilstanden, slik at selv om det blir skrudd av i begynnelsen av while-løkken vil det skrus på igjen med en gang. Visuelt sett er stopp-lyset da på hele tiden mens knappen er trykket inn, og **L6** er oppfylt.

Når tilstandsmaskinen går inn i "DOOR_OPEN" tilstanden, settes en timer til 3 sekunder, og når denne går ut lukkes døren. Den settes på nytt hver gang vi leser obstruksjonssignalet, slik at den lukker seg fremdeles 3 sekunder etter at dette har gått lavt. Da er **D1** og **D4** oppfylt. Heisdøren åpnes kun når sammenligningen av etasjeindikator og bestilling er sant, og når det ikke finnes bestillinger vil døren ikke åpne seg, i henhold til **D2**. **D3** tilfredsstilles ved at vi sjekker om heisen er i en etasje mens stopp-knappen holdes inne. Hvis dette stemmer, åpner vi dørene og setter "current_state" til "DOOR_OPEN".

Heisen stopper alltid før den går inn i "DOOR_OPEN" tilstanden, og den lukker alltid døren før den går ut av "DOOR_OPEN" tilstanden. Logikk for at heisen ikke blir etasjeforvirret når stopp trykkes mellom etasjer er også implementert, og dermed vil heisen alltid stå stille når døren er åpen og døren åpner seg ikke utenfor etasjer, i henhold til **S1-2**. Da fungerende kø-logikk er implementert, vil ikke heisen kjøre utenfor det definerte området, slik at **S3** er oppfylt. Ettersom at stoppknappen hele tiden blir lest og sender oss rett inn i "STOP" tilstanden, vil heisen stoppe momentant når stopp-knappen trykkes på. I "STOP" tilstanden slettes alle ordre hver gang, og da er **S4-7** oppfylt.

Obstruksjonsbryteren leses kun i "DOOR_OPEN" tilstanden, og påvirker da ikke heisen ellers. Øvrig robusthet som at vi ikke skal få minnelekkasje, eller at heisen ikke vet hvor den er etter oppstart, er tatt høyde for slik at **R1-3** er tilfredsstilt.

Heisen fungerer som en heis det går an å leve med, og vi vil påstå at **Y1** er tilstrekkelig tilfredsstilt.

# 4 Diskusjon

Det finnes et par svakheter ved måten vi har implementert koden på. Vi har ikke aktivt tatt høyde for punkt **S3**, men tenkte er at heisen aldri vil få en ordre utenfor første til fjerde etasje og vil derfor aldri få beskjed om å bevege seg utenfor dette området. Dette kan bli problematisk hvis for eksempel en av etasjesensorene slutter å fungere, ettersom vi ikke har noe sikkerhetsnett. Det kan også være problematisk at dersom heisen setter seg fast utenfor første eller fjerde etasje, har vi ingen måte å komme oss tilbake inn i heisrommet på. I koden finnes det ingen grunn til at heisen skal komme i slike situasjoner, men hvis det

uansett skulle skje, vil det være nødvendig å fysisk løfte heisen opp/dytte den ned til den er innenfor arbeidsområdet.

Den andre (og kanskje mest innlysende) svakheten er at vi ikke har implementert det vi selv anser som den viktigste funksjonen en heis kan ha, nemlig heismusikk. Dersom vi skulle implementert heismusikk, ville det åpenbare valget vært "Mii Channel Music" så lenge Nintendo hadde gitt oss lisens til å bruke den.