

Heisprosjekt TTK4235

Kristian Nilsen og Ørjan Carlsen

01.03.2020



1 Overordnet arkitektur

Vi har valgt en arkitektur der vi benytter en tilstandsmaskin for å styre heisen mellom ulike tilstander. Ordremodulen bruker Hardware for å sjekke om det er kommet noen nye bestillinger, og FSM bestemmer deretter heisens oppførsel, ut fra hvilke ordre som finnes. I tillegg bruker FSM Hardware for å sette utgangssignaler til heisen, og sjekke inngangssignaler fra heisen.

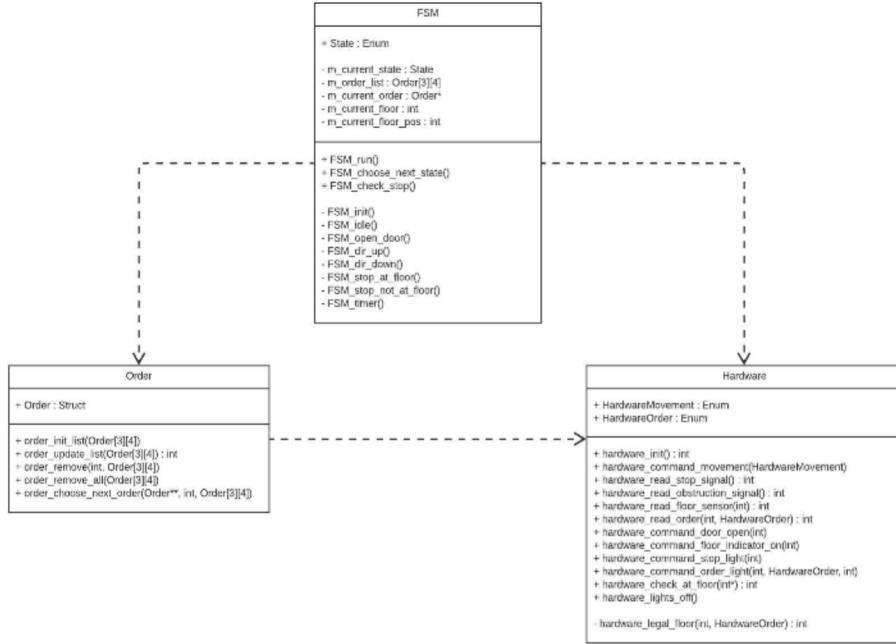


Figure 1: Klassediagram

For å vise hvordan modulene fungerer sammen, har vi satt opp et sekvensdiagram som viser utførelsen av denne sekvensen:

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.

I sekvensdiagrammet ble det valgt å fokusere på de funksjonene som er mest sentrale i kommunikasjonen mellom modulene. Det er verdt å merke seg at funksjoner som `order_update_list` og `hardware_check_at_floor` kjøres hele

tiden. Funksjonene skal henholdsvis oppdage om det er kommet noen nye ordre, og oppdatere hvilken etasje vi befinner oss i, men er utekommende utenom i de viktigste sekvensene.

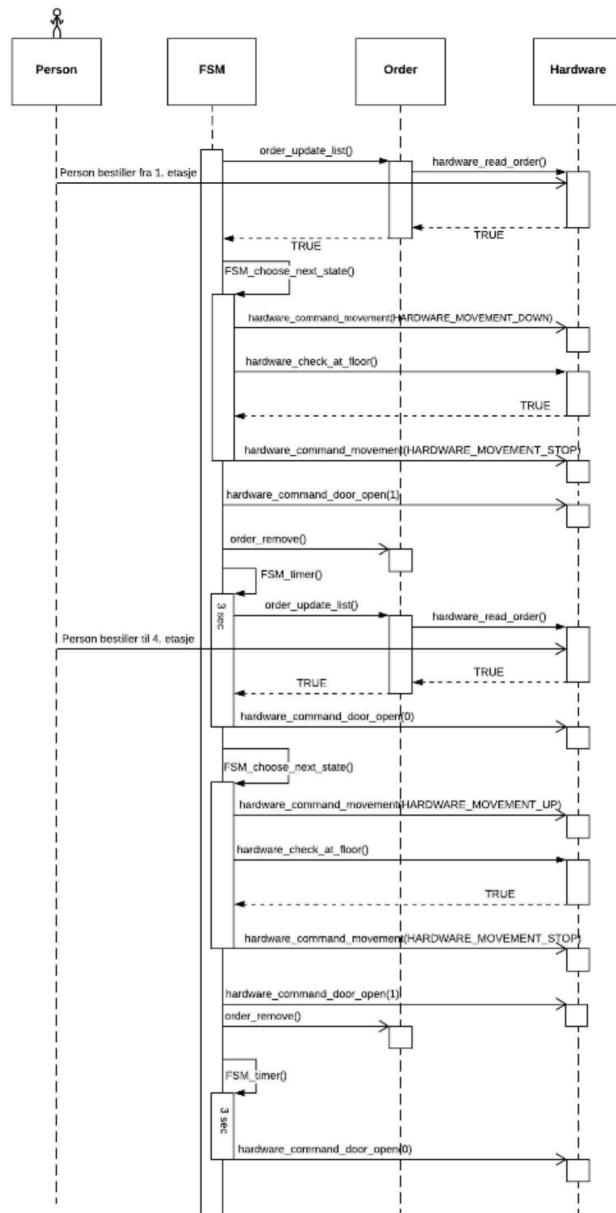


Figure 2: Sekvensdiagram

Tilstandsdiagrammet viser hvilke tilstander vi har i FSM-modulen, hva som skal til for at tilstanden endrer seg og hvilke utgangssignaler som blir satt. Supertilstanden består av de tilstandene som vil utgjøre normal drift.

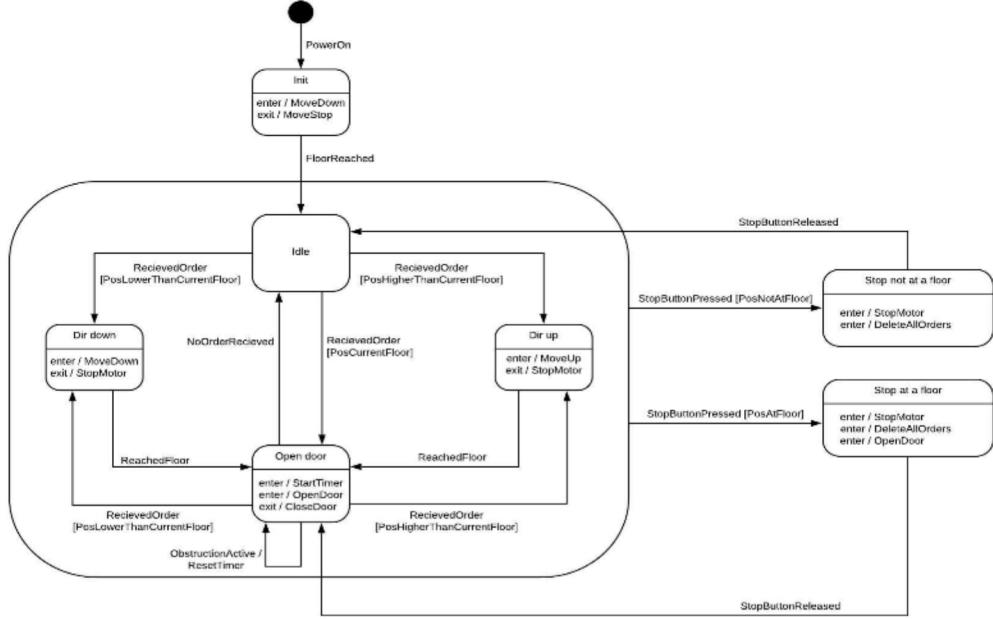


Figure 3: Tilstandsdiagram

Vi har valgt denne arkitekturen fordi den holder styringen av heisen og håndteringen av bestillinger adskilt. Dette gjør det letter å endre på en av modulene i ettertid, om man ønsker en annen heisoppførsel. Vi valgte tilstandene heisen veksler mellom på bakgrunn av hvordan heisen i grove trekk skal oppføre seg under ulike betingelser. Her har vi valgt å skille mellom bevegelse oppover og nedover, fordi det gjør det lettere å prioritere hvilke ordre som skal ekspedieres. Samtidig er det et skille mellom det å stå i ro med lukkede dører og åpne dører da vi kan stå i ro utenfor en etasje. Vi har også valgt å skille mellom stopptilstanden i en etasje og utenfor for å sikre at dørene ikke åpner seg utenfor en etasje.

2 Moduldesign

2.1 FSM

FSM-modulen vår har en hovedfunksjon *FSM_run* som kalles på i main. Den består av en evig løkke og en switch som sender programmet inn i ulike tilstander, avhengig av en variabel *m_current_state*. Hver tilstand har en egen

funksjon som styrer oppførselen i den gitte tilstanden. Vi har valgt å lage en funksjon per tilstand, selv om de i noen tilfeller gjør nesten det samme, for å ha et klart skille mellom tilsatndene og dermed legge til rette for oppdatering av oppførelsel i ulike tilstander. Å dele heisens bevegelse i to ulike tilstander, vil for eksempel gjøre det lettere å prioritere hvilke ordre som skal betjenes. I tillegg har vi to funksjoner som velger neste tilstand. *FSM_choose_next_state* bruker neste ordre for å avgjøre neste tilstand etter idle eller open door tilstandene, mens *FSM_check_stop* sjekker om stoppknappen trykkes inn, og velger riktig stopptilstand ut i fra posisjon. Ved å skille på disse, kan vi kontinuerlig kontrollere om stoppknappen trykkes inn, der vi ellers ikke ønsker å bytte tilstand. Til slutt har vi en timer-funksjon for døra. Den har vi valgt å holde adskilt fra resten av funksjonen for åpen dør tilstanden for å legge til rette for at timeren kan brukes i andre deler av programmet.

For å holde kontroll på alle aktive og inaktive ordre, har vi en todimensjonal liste *m_order_list*. Hver gang en ny ordre mottas settes den til aktiv i ordrelisten, og hver gang en ordre blir ekspedert settes den til inaktiv. *m_current_order* peker til den ordren som blir utført for øyeblikket. Vi valgte å lage *m_current_order* som en peker, slik at vi kan endre både denne og *m_order_list* samtidig, ved å kun oppdatere en av dem. I tillegg har vi modulvariablene *m_current_floor* og *m_current_floor_pos* som holder orden på henholdsvis hvilken etasje vi var i sist, og om vi er under, på eller over denne.

2.2 Order

For å beskrive en ordre, har vi en struct, *Order*, som inneholder informasjon om ordretype, etasje og om den er aktiv eller ikke. I modulen har vi også ulike funksjoner for å behandle *m_order_list* bestående av *Order*. I tillegg til oppdaterings- og settingsfunksjoner, har vi *order_choose_next_order* som velger hvilken ordre som skal betjenes. Her har vi valgt å prioritere de aktive ordrene som er lengst unna heisen i distanse, slik at for de nedre etasjene vil det prioriteres ordre fra 4. etasje, og så 3. etasje osv. For de øvre etasjene vil det prioriteres ordre fra 1. etasje, 2. etasje, osv. Denne funksjonen vil, for 4 etasjer, sørge for at vi aldri vil komme i en situasjon der en ordre ikke blir betjent. Dette vil også medføre en ganske effektiv behandling av ordrene, da heisen i stor grad vil prøve å bevege seg mellom ytterpunktene.

2.3 Hardware

Vi har valgt å gjøre noen forandringer i den utdelte hardware fila som vi anså som nyttige. Vi la til funksjonene *hardware_check_at_floor* og *hardware_lights_off*. Der *hardware_check_at_floor* sjekker om heisen er i en etasje, og setter etasjelyset til etasjen den eventuelt er i. Funksjonen tar også inn en peker den setter til den gitte etasjen. *hardware_lights_off* skrur av alle bestillingslysene.

3 Testing

Vi har valgt å dele opp testene våre i enhetstesting og integrasjonstesting. Hensikten med enhetstestingen er å sjekke at modulene fungerer hver for seg, og at de enkelte kravspesifikasjonene er oppfylt. Integrasjonstestene skal i større grad kontrollere samhandlingen mellom modulene, og kontrollere grensetilfeller som ikke ble testet godt nok under enhetstesting.

3.1 Enhetstesting

3.1.1 Oppstart

Vi startet heisen mellom 2. og 3. etasje, la inn bestillinger før den nådde 2. etasje, og sjekket at den ikke tok i mot disse. Da er **O 1** og **O 2** tilfredsstilt, gitt at vi ikke tar urealistiske startbetingelser i betraktnsing som spesifisert i **O 3**.

3.1.2 Håndtering av bestillinger

La inn bestilling fra 4. etasje, når heisen sto i 1. etasje. Før den passerte 2. etasje, la vi inn en bestilling nedover i 2. etasje og en bestilling oppover i 3. etasje. Heisen stoppet ikke i 2. etasje, men stoppet i 3. etasje, før den fortsatte til 4. etasje. Dette betyr at kravspesifikasjon **H 2** er tilfredsstilt. Til slutt tok den bestillingen i 2. etasje. Fordi den tar alle ordre, er **H 1** tilfredsstilt.

Heisen sto i 1. etasje, og vi la inn en bestilling i 4. etasje. Deretter la vi inn 2 bestillinger til 3. etasje; opp og ned. På vei opp stoppet den i 3. etasje, før den fortsatte til 4. etasje. Her sto den stille, som betyr at begge ordrene i 3. etasje regnes som ekspedert, slik at **H 3** og **H 4** er tilfredsstilt.

3.1.3 Bestillingslys og etsjelys

Heisen sto i 1. etasje, og vi la inn tre bestillinger til 3. etasje; opp, ned og fra innsiden. Disse lyste helt til heisen ankom 3. etasje, og slukket den alle 3 bestillingslysene, slik at **L 1** er tilfredsstilt. De øvrige knappene var slukket, altså er **L 2** tilfredsstilt.

Heisen sto i 1. etasje. Vi la inn en bestilling fra 4. etasje, og deretter oppbestillinger i 2. og 3. etasje. Mens heisen sto stille i 2., 3. og 4. etasje, var tilhørende etasjelys tent, i henhold til **L 3**. På vei mellom etasjene lyste alltid etasjelyset til forrige etasje, slik at **L 4** er tilfredsstilt. Dessuten lyste det alltid kun ett etasje lys av gangen, som det er spesifisert i **L 5**.

Vi trykket inn stoppknappen da heisen var på ulike steder, både mellom og i etasjer. Stopplyset lyste så lenge den var trykket inn, slik at **L 6** er tilfredsstilt. **L 3** til **L 5** fungerte fortsatt selv om vi trykket inn stoppknappen.

3.1.4 Døren

For å teste **D 1** og **D 2**, gorde vi samme test som for **H 3** og **H 4**. Døren var åpen i 3 sekunder i hver etasje, og alltid lukket da den var i bevegelse, slik at **D 1** og **S 1** er tilfredsstilt. Etter at døren lukket seg i 4. etasje, forble døren lukket i henhold til **D 2**.

For å forsikre oss om at **D 3** er tilfredsstilt gjorde vi to tester. Først lot vi den stå stille i 2. etasje og holdt inne stoppknappen. Døren åpnet seg umiddelbart, og forble åpen til vi slapp stoppknappen og så ytterligere 3 sekunder. I den andre testen lot vi heisen gå fra 1. etasje mot 4. etasje, og når den kom til 2. etasje trykket vi inn stoppknappen. Da gikk dørene opp, og forble oppe så lenge vi holdt knappen inne og så ytterligere 3 sekunder.

For å teste **D 4** lot vi heisen gå til en bestilling i 4. etasje, for så å aktivere obstruksjonsbryteren når dørene åpnet seg. Vi lot bryteren være aktiv en stund slik at det gikk mer enn 3 sekunder etter at heisen ankom før vi deaktiverte bryteren. Da observerte vi at dørene var åpne så lenge bryteren var aktiv og så ytterligere 3 sekunder.

3.1.5 Sikkerhet

Vi lot heisen gå fra 4. etasje til 1. etasje, og før vi kom til 3. etasje flippet vi obstruksjonsbryteren. Den hadde ingen effekt på systemet før vi kom til 1. etasje og dørene åpnet seg. Etter vi hadde deaktivert bryteren og dørene lukket seg, aktiverte vi den igjen og observerte at den fortsatt ikke hadde effekt på systemet. Før heisen kom til 1. etasje åpnet aldri dørene seg. Da er **S 2** og **R 1** oppfylt.

Vi sendte heisen mellom 2. og 4. etasje, og trykket på stopp når heisen var mellom 3. og 4. etasje med aktive bestillinger i 4. og 1. etasje. Da stoppet heisen momentant og bestillingene ble slettet. Heisen sto så stille, stopplyset lyste og det ble ikke tatt i mot bestillinger før vi slapp knappen. Etter at vi slapp knappen sto heisen stille til den fikk en ny bestillingen som den ekspederte. Dørene åpnet seg ikke på noe punkt før den kom til etasjen den nye bestillingen kom fra. Da anser vi **L 6** samt **S 2** til **S 7** som tilfredsstilt.

3.1.6 Robusthet

Det har ikke under noen av testene oppstått en situasjon der programmet har krevd omstart eller flere kalibreringsrunder. Vi anser da **R 2** og **R 3** som tilfredsstilt.

3.2 Integrasjonstesting

Etter enhetstestinga sjekket vi at modulene fungerer sammen slik de skal. Dette innebærer å teste en rekke grensetilfeller, og kontrollere at oppførselen er

som ønsket.

3.2.1 Stoppknappen

Første test bestod i å stoppe heisen mellom 1. og 2. etasje. Vi la inn bestillinger fra 1. og 2. etasje annenhver gang, og mellom hver bestilling stoppet vi heisen, slik at den aldri nådde en av etasjene. Heisen visste gjennom hele testen hvor den var, som vil si at den alltid beveget seg mot riktig etasje da den mottok en bestilling. Deretter la vi inn en bestilling i 3. etasje. Etter at den hadde passert 2. etasje, og før den nådde 3., stoppet vi heisen. Så la vi inn en bestilling i 2. etasje, og observerte at den korrekt gikk ned til 2. etasje. Dette indikerer at stoppknappen har ønsket oppførsel, selv ved veldig "uvanlig" bruk.

3.2.2 Bestillingshåndtering

Et annet grensetilfelle vi ønsket å teste var hvordan heisen håndterer mange bestillinger. Vi fylte opp med alle mulig bestillinger, og kontrollerte at alle ble ekspedert. Etter at bestillingene i en etasje ble ekspedert, ble de aktivert på nytt så snart heisen hadde forlatt etasjen. Dette fortsatte en liten stund, før vi sluttet å legge inn nye bestillinger og fikk sjekket at alle ble ekspedert til slutt. Deretter ble testen gjentatt, men denne gangen trykket vi på stoppknappen, og fikk sett at heisen klarte å slette alle bestillingene.

3.2.3 Timerfunksjonen

Den siste testen var for å sjekke at døren forblir åpen så lenge den skal. Da heisen ankom en etasje, og døren åpnet seg, aktiverte vi obstruksjonsknappen. Vi bestilte heisen til andre etasjer, observerte at den forble i open door tilstanden, for så å slette de nye ordrene med stoppknappen. Nå, mens døren fortsatt er åpen, deaktiverte vi obstruksjonsknappen. Før det var gått tre sekunder trykket vi stopp, for så å bytte på å legge inn nye bestillinger i samme etasje og trykke på stoppknappen, alltid med mindre enn tre sekunders mellomrom. Døren forble åpen gjennom hele testen, akkurat som den skal, og lukket seg tre sekunder etter siste gang vi la inn en bestilling i samme etasje.

4 Diskusjon

Det kunne ha vært aktuelt med en egen timermodul som var uavhengig av de øvrige modulene. Med enda mindre grensesnitt til de andre modulene, kunne det blitt lettere å detektere eventuelle feil. Likevel mener vi at vår løsning er fornuftig til dette prosjektet, da vi kun har én funksjon som tar hensyn til alt som har med tidsaking å gjøre. Dessuten vil det være enkelt å utvide funksjonaliteten til funksjonen dersom heisens oppførsel skal endres, da dette kun må gjøres i funksjonen, og ikke andre steder i koden.

Funksjonen som velger neste ordre, *order_choose_next_order*, kan få problemer om antallet etasjer utvides. Hvis vi utvider til 6 etasjer, kan det potensielt oppstå et problem der den blir stående mellom 2. og 3. etasje, uten å ekspedere ordre fra 1. etasje. Dette problemet kan for eksempel løses ved å holde styr på siste retning heisen beveget seg i, og bruke dette i prioriteringen. En styrke ved arkitekturen vår er at en slik endring stort sett kun vil kreve endringer i denne ene funksjonen.

Order-structen vår hadde ikke trengt å vite hva slags type ordre vi har, slik programmet er nå, da plasseringen i *m_order_list* holder den informasjonen. Dette kan derimot bli nødvendig i ettertid, om man skulle ønske å skille mer på hvordan en ordre skal behandles.