Bacheloroppgave

NTNU
Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for datateknologi og informatikk

Haziri, Donjetë
Kallevig, Isac Mikal
Muggerud, Håkon Trøan
Tøn, Aslak

Ambulansesimulator

Bacheloroppgave i Dataingeniør og i Programmering
Veileder: Haug, Frode

Mai 2021

NTNU
Kunnskap for en bedre verden

Haziri, Donjetë
Kallevig, Isac Mikal
Muggerud, Håkon Trøan
Tøn, Aslak

Ambulansesimulator

Bacheloroppgave i Dataingeniør og i Programmering
Veileder: Haug, Frode
Mai 2021

Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for datateknologi og informatikk

Bacheloroppgave: Ambulansesimulator

Haziri, Donjetë
Kallevig, Isac Mikal
Muggerud, Håkon Trøan
Tøn, Aslak

19. mai 2021

Sammendrag

Tittel: Ambulansesimulator		Dato: 19. mai 2021
Deltakere:	Donjetë Haziri, Isac Mikal Kallevig, Håkon Trøan Muggerud, Aslak Tøn	
Veileder:	Frode Haug	
Oppdragsgiver:	Hans Martin Lilleby	
Stikkord:	Simulering, UX design, Maskinvare, wxWidgets, C++	
Antall sider: 69	Antall vedlegg: 9	Publiseringsavtale inngått: 2021-05-19

Kort beskrivelse av bacheloroppgaven:

I forbindelse med nytt studieprogram i paramedisin på NTNU i Gjøvik var det ønsker om å få oppgradert det nåværende systemet for ambulansesimulering. Nåværende system består av en reell ambulanse plassert på et hydraulisk bevegelig understell. Simulatoren kan spille av sekvenser som inneholder gitte bevegelser ambulansen skal følge, og man kan styre ambulansen med et spillratt fra førersetet. Bachelorgruppen har levert en splitter ny programvare med et intuitivt brukergrensesnitt og god dokumentasjon. Dette systemet inneholder også ny funksjonalitet, blant annet fysikksimulasjon for styring av ambulanse og generering av sekvenser under simulering. Gruppen hadde ambisjoner om flere utvidelser slik som VR og AR, men fikk ikke nok tid. Systemet er uansett lagt opp slik at utvidelser lettere kan legges til i fremtiden. Oppdragsgiver var veldig fornøyd med sluttproduktet.

Abstract

Title:	Ambulance Simulator	Date: May 19, 2021
Participants:	Donjetë Haziri, Isac Mikal Kallevig, Håkon Trøan Muggerud, Aslak Tøn	
Supervisor:	Frode Haug	
Employer:	Hans Martin Lilleby	
Keywords:	Simulation, UX design, Hardware, wxWidgets, C++	
Number of pages: 69	Number of appendix: 9	Availability: 2021-05-19

Short description of bachelor thesis:

NTNU Gjøvik will start a new programme of study in paramedics. Therefore, it was desirable to upgrade the current system for ambulance simulation. The current system consists of a real ambulance placed on a hydraulically movable chassis. The simulator can play sequences that contain given movements the ambulance must follow, as well as controlling the ambulance with a steering wheel from the driver's seat. The bachelor group has delivered a brand new software with an intuitive user interface and a comprehensive documentation. This system also includes new functionality, including physics simulation for ambulance control and generation of sequences during simulation. The group wanted to implement several expansions such as VR and AR, but did not get enough time. Nevertheless, the system arranged so that extensions easily can be added in the future, and the client was very pleased with the end product.

Forord

Denne bacheloroppgaven er skrevet av Donjetë Haziri, Isac Mikal Kallevig, Håkon Trøan Muggerud og Aslak Tøn. Alle er studenter ved NTNU i Gjøvik under institutt datateknologi og informatikk.

Vi ønsker å takke Frode Haug for for strukturert veiledning gjennom hele semesteret. Uten deg ville rapportens struktur være markant dårligere, og gruppens stressnivå markant høyere.

Vi ønsker også å takke Hans Martin Lilleby for fantastisk entusiasme og sammarbeidsvilje gjennom prosjektet. Det har vært en fryd å jobbe sammen med deg.

Innhold

1 Introduksjon **1**
1.1 Omfang . . . 1
1.2 Valg av oppgave . . . 2
1.3 Målgruppe . . . 3
1.4 Gruppens kunnskaper . . . 4
1.5 Rammer . . . 4
1.6 Rollefordeling . . . 5
1.7 Rapportens struktur . . . 6

2 Kravspesifikasjon **8**
2.1 Funksjonelle krav . . . 8
2.2 Ikke-funksjonelle krav . . . 9
2.3 Use Case . . . 9

3 Utviklingsprosess **13**
3.1 Valg av utviklingsmodell . . . 13
3.2 Trello . . . 13
3.3 Designdokumenter . . . 13
3.4 Møter . . . 14
3.5 Milepæler . . . 14
3.6 Møter med veileder . . . 14
3.7 Versjonskontroll . . . 14
3.8 Parprogramering . . . 15

4 Grafisk Design **16**
4.1 Gruppens UX prosess . . . 16
4.2 Brukergrensesnittet . . . 19
4.3 Hvorfor UX-testing . . . 23

5 Teknisk Design **25**
5.1 Systemarkitektur . . . 25
5.2 Argumentasjon for valgt systemarkitektur . . . 25
5.3 Alternative valg av systemarkitektur . . . 26
5.4 Eksterne teknologier . . . 26
5.5 Programmeringsspråk: C++ . . . 30
5.6 Styring av hydraulikk . . . 31

6 Implementasjon **33**
6.1 Input Controller . . . 33
6.2 Brukergrensesnittet . . . 34
6.3 Styring fra Mus . . . 38
6.4 Virtuelle krefter: Konvertere krefter til volt . . . 41
6.5 Styring fra Spill og Euro Truck Simulator 2 . . . 44
6.6 Fri styring og virituell bil . . . 46

6.7 AmbuGL: OpenGL visualisering 47
6.8 CoreToolz 48
6.9 Les Sekvens: .sequence filer 49
6.10 Les Sekvens: Physics Toolbox fil / .csv fil 51

7 Kvalitetssikring **54**
7.1 Kodestandard 54
7.2 Code Review 54
7.3 Testing 54
7.4 Dokumentasjon 55
7.5 Risikoanalyse 55

8 Installasjon **56**
8.1 NI-DAQmx 56
8.2 wxWidgets 56
8.3 Logitech G920 56
8.4 GLEW 57
8.5 Euro Truck Simulator 2 57
8.6 Repository 58
8.7 Vedlikehold og utvikling 58

9 Diskusjon **59**
9.1 Kvalitetssikring 59
9.2 Risikoanalyse 60
9.3 Tidføring 60
9.4 Tidsestimering 61
9.5 Gruppeledelse 62
9.6 Koden 63
9.7 Planlegging 64

10 Konklusjon **65**
10.1 Implementert funksjonalitet 65
10.2 Systemets kvalitet 65
10.3 Planlegging 66
10.4 Gjennomføring 66
10.5 Gruppesamarbeid 66

11 Videre arbeid **68**
11.1 Ferdigstilling 68
11.2 Utvidelser 68

A Definisjoner **71**

B Framdriftsplan **72**

C Designdokumenter **74**

D Orginal Prosjektplan **77**

E Arbeidskontrakt **95**

F Arbeidsdokumentasjon **98**

G Referater **101**
G.1 Møte 19.01.21 . . . 101
G.2 Møte Frode 2021-01-26 . . . 102
G.3 Møte Frode 2021-02-02 . . . 102
G.4 Møte 2021-02-09 . . . 104
G.5 Møte Frode 2021-02-16 . . . 104
G.6 Møte Frode 2021-02-23 . . . 104
G.7 Møte Frode 2021-04-20 . . . 105
G.8 Møte Frode 2021 . . . 106
G.9 Møte Frode 2021-05-11 . . . 106
G.10 Møte 2021-01-26 . . . 106
G.11 Møte 19.01.21 . . . 106
G.12 Møte Gruppen 2021-01-20 . . . 107
G.13 Møte 2021-01-25 . . . 108
G.14 Møte gruppen 2021-01-28 . . . 108
G.15 Møte Gruppen 2021-02-02 . . . 108
G.16 Møte gruppen 2021-02-16 . . . 108
G.17 Møte Gruppen 2021-02-23 . . . 109
G.18 Møte Gruppen 2021-03-02 . . . 109
G.19 Møte gruppen 2021-03-09 . . . 110
G.20 Møte Gruppen 2021-03-16 . . . 110
G.21 Møte Gruppen 2021-03-19 . . . 110
G.22 Møte gruppen 2021-04-06 . . . 111
G.23 Møte Gruppen 2021-04-13 . . . 111
G.24 Møte Gruppen 2021-04-19 . . . 112
G.25 Møte Gruppen 2021-04-27 . . . 113
G.26 Møte gruppen 2021-05-04 . . . 114
G.27 Møte Gruppen 2021-05-11 . . . 114
G.28 Møte Gruppen 2021-05-12 . . . 115
G.29 Møte Gruppen 2021-04-27 . . . 115
G.30 Mote Hans Martin 21-01-19 . . . 116
G.31 Møte 2021-01-26 . . . 119
G.32 Møte Hans Martin 2021-02-02 . . . 120
G.33 Møte Pål Erik 20.01.21 . . . 120
G.34 Referat fra research i labben . . . 121

H Akseptanse tester **122**

I UX Test Skjemaer **126**

Figurer

1.1 Ambulansen står på en styringsplate som er ankret foran. Den beveger seg vha. to sylindere lenger bak. (kreditering: Isac Kallevig) . 2
2.1 UseCase diagram for prosjektet 10
4.1 De største elementene fra første iterasjon av wireframe 18
4.2 De samme elementene som i figur 4.1, i andere første iterasjon . . . 18
4.3 De samme elementene som i figur 4.1 og figur 4.2, i ferdig versjon . 19
4.4 Elementer fra det ferdige programmet 20
4.5 Vestre: Windows GUI. Høyre: Selvlagd knapp 21
4.6 Vestre: Wireframe. Høyre: Brukergrensesnittet 21
4.7 Lag sekvensknapper og Fri styringknappene separert 22
4.8 Lag sekvensknapper og Fri styringknappene kombinert 23
5.1 Endelig systemarkitektur 25
5.2 Eksempel på en gjennomført enhetstest 27
5.3 Systemsekvensdiagram for generell styring av hydraulikk 32
6.1 Hovedløkken for styring av hydraulikk 33
6.2 Figur viser museposisjon og korresponderende possisjon til platen ambulansen står på. 40
6.3 Figuren viser en tilnærmet modell av planet ambulansen står på. . . 41
6.4 Figuren viser hvilken retning gravitasjonen, sett bakfra, peker “lokalt” inne i bilen. 42
6.5 Objektorientert design av AmbuGL 48
7.1 Klasse referanse til en klasse (“Collaboration diagram” slettet for bedre overblikk) 55
8.1 Tre struktur etter ferdig instalasjon 57
B.1 Gantt-diagram 73
C.1 UseCase diagram for prosjektet 74
C.2 Originale systemarkitektur 75
C.3 Klassediagram for det nye systemet 76
F.1 Tid fordelt per uke 99
F.2 Total tid brukt, fordelt på kategorier 100

1 Introduksjon

1.1 Omfang

1.1.1 Fagområde

Bruk av datamaskin for å trene på ulike kunnskaper og ferdigheter kalles "maskinbasert læring". Til dette brukes pedagogisk programvare, som kan deles inn i drill/trening, dialog, simulering, spill, oppdagelse, problemløsning og modellbygging. Drill vil f.eks. være mer programstyrt, mens modellbygging vil være brukerstyrt.

Fordeler med en slik læringsmetode inkluderer større tilgjengelighet, kosteffektivitet ved opplæring av mange studenter, og det kan lett tilpasses ulike fagområder. En tilpasset læringsplattform kan ta lang tid og kreve mye ressurser å sette i stand, men på lang sikt er det en investering som kan spare kostnader på instruktører, klasserom, reising osv.

1.1.2 Avgrensning

I denne oppgaven skal vi se nærmere på ferdighetstrening i simuleringer. Dette er en effektiv måte man virtuelt kan trene på virkelige situasjoner. Dette gjelder spesielt scenarioer der trening i den reelle verden vil være for kostbar, farlig, tidskrevende eller kompleks. Simulering med ferdighetstrening har et spesifikt læringsmål, f.eks. i forbindelse med styring av kjøretøy eller faste arbeidsrutiner.

1.1.3 Eksisterende løsning

Senter for simulering og pasientsikkerhet ved Institutt for helsevitenskap i Gjøvik har flere fasiliteter for simulering innen helsefagutdanninger, inkludert ambulansesimulatoren. Simulatoren består av en reell ambulanse som er plassert på et hydraulisk bevegelig understell styrt av programvare på PC. Denne programvaren ble opprinnelig utviklet av studenter i perioden 2008 til 2009, og har utviklet seg en del siden den tid. Det er mangel på dokumentasjon, som fører til at systemet er vanskelig å drifte og videreutvikle.

1.1.4 Oppgavebeskrivelse

Til høsten starter NTNU i Gjøvik opp med et studieprogram i paramedisin[1]. I denne forbindelsen ønskes det å få oppgradert det nåværende systemet for ambulansesimulering.

Oppgavebeskrivelsen går ut på å utvikle et nytt system fra bunnen av, som vil ha et intuitivt grensesnitt og god dokumentasjon. Det er også et ønske om oppgradering av PC i kontrollrom samt ratt og pedaler i ambulansen.

Simulatoren har i hovedsak to bruksområder. Den første er avspilling av en eksisterende sekvens som inneholder gitte bevegelser ambulansen skal utføre. Slike

[1] `https://www.ntnu.no/ihg/paramedisin`

Figur 1.1: Ambulansen står på en styringsplate som er ankret foran. Den beveger seg vha. to sylindere lenger bak. (kreditering: Isac Kallevig)

sekvenser varer omtrent 1-5 minutter. Ambulansepersonell utfører ferdighetstrening bak i bilen når disse bevegelsene simuleres. Denne delen av simuleringen er dermed programstyrt.

Det andre bruksområdet innebærer i tillegg en sjåfør som styrer ambulansen fra førersetet. Sjåføren vil spille et spill/simulasjon som vises på prosjektor. Hydraulikken vil reagere når sjåføren gasser, bremser og svinger. Denne delen av simuleringen er dermed mer brukerstyrt.

Begge disse scenarioene vil alltid ha en person som kan starte/stoppe simuleringen fra et kontrollrom. Simuleringen kan også stoppes ved hjelp av en sikkerhetssløyfe som består av lysgardiner, sensorer i dørene og en nødstoppknapp. Personen i kontrollrommet vil også få en forhåndsvisning av ambulansens sanntidsposisjon.

Uavhengig av simulasjonen skal programvaren tilby verktøy for å lage en skreddersydd sekvens. Denne sekvensen lagres som en fil, og kan senere avspilles med ambulansesimulatoren. En sekvens kan også bli skapt ved å gjøre et opptak av en brukerstyrt simulering.

1.2 Valg av oppgave

Prosjektgruppen valgte denne oppgaven da det var enighet innad i gruppen om at dette ville være en oppgave som tilbød alle gruppens medlemmer forskjellige problemer de kunne engasjere seg innenfor. Problemstillingene som primært kom fram var maskinnær arbeid, muligheter for “gamification” og 3d modellering.

Oppgavebeskrivelsen var også tydelig og gruppen så flere utvidelsesmuligheter gjennom samtaler med arbeidsgiver.

1.2.1 Maskinnær arbeid

Prosjektgruppen så i utgangspunktet to relevante oppgaver som lå mot maskinnær programmering. Den ene var styring av hydraulikken som bevegde ambulansen. Den andre var et digitalt ratt og pedaler som brukes i ambulansen til simulering av kjøring.

1.2.2 "Gamification"

Et av medlemmene på prosjektgruppen har tatt sin utdanning innenfor spillutvikling[2]. Tre av de fire studentene i gruppen har også tatt valgfaget spillprogramering[3]. Det er derfor relativt høy interesse på prosjektgruppen innenfor temaet spillprogrammering. Ambulansesimulatoren ga indirekte en del oppgaver relatert til spillprogrammering. Dette inkluderer områder som simulering av kjøring ved hjelp av prosjektor og 3d modellering av en ambulanse for visning i programmet.

1.2.3 Klar oppgavebeskrivelse

Et annet moment som hjalp prosjektgruppen å bestemme seg for denne oppgaven var at oppgavebeskrivelsen hadde klare forventninger og mål. Det var en del andre oppgaver prosjektgruppen så på som interessante, men de hadde ikke en tydelig utformet kravspesifikasjon. Andre oppgaver hadde en oppdragsgiver som ikke var inført i hvor lang tid utvikling av forskjellige funksjonaliteter krevde. Gruppen så for seg at en slik oppdragsgiver ville presentere arbeidsoppgaver som enklere enn de var. Prosjektgruppen valgte derfor heller å fokusere på en oppgave hvor usikkerheten var minimal.

1.2.4 Utvidelsesmuligheter

Gruppen så også flere måter å kunne utvide prosjektet på, dersom prosjektet viste å seg være lett løslig. Dette inkluderte utvidelser innenfor VR (Virtual Reality) og AR (Augmented Reality), hvor man kunne bruke VR briller mens man brukte ambulansen. Det inkluderte også elementer av CV (Computer Vision) som enkelte medlemmer hadde kunnskaper innenfor. CV skulle brukes til å utvikle moduler som kunne ta en video fra bilkjøring, og simulere nærmest mulig bevegelsene som opplevdes i bilen under filmingen.

1.3 Målgruppe

Det er hovedsakelig to grupper denne programvaren er utviklet for. Den ene er undervisere innen paramedisin. Det antas at underviserne ikke har store kunn-

[2] `https://www.ntnu.no/studier/bprog`

[3] `https://www.ntnu.edu/studies/courses/IMT3601`

skaper innenfor data. Den andre gruppen er elevene innenfor paramedisin. Brukergrensesnittet er i hovedsak målrettet mot underviseren, men hvilken funksjonalitet ambulansesimulatoren tilbyr er målrettet mot læringsmålene i studieprogrammet Paramedisin, med andre ord studentene.

1.4 Gruppens kunnskaper

Gruppen har alle gjennomført store deler av den samme utdanningen og har her delte kunnskaper. Dette er kunnskaper innenfor: Algoritmer og datastrukturer, databaser, systemutvikling og operativsystemer. Enkelte eller flere medlemmer på gruppen har også erfaring innenfor: brukerdesign, matematikk for data og kalkulus, Datasyn, og 3d modellering.

Et gruppemedlem har erfaring og kurs innenfor kommunikasjon. En annen har mer bakgrunn innen matematikk. En tredje har et godt øye for design og brukervennlighet. Det siste gruppemedlemmet har mer erfaring med innføring i nye systemer, og kommer til å ta hovedansvaret for forsking på forskjellige teknologier gruppen trenger.

1.5 Rammer

1.5.1 Gruppemedlemmer

Prosjetkgruppen består av fire studenter, tre tar studieretningen BIDAT[4] og en tar studieretningen BPROG[5] med underspesialisering i spillprogramering.

1.5.2 Tidsrammer

Tidsrammene til prosjektet tar utgangspunkt i tidsrammene til et semester ved NTNU i Gjøvik. Semesterstart var 11. januar 2021, og innlevering av bachelor rapport var satt til 20. mai 2021. Dette gir prosjektgruppen omtrent 5 måneder på planlegging, utvikling og dokumentasjon av systemet samt skriving av denne rapporten.

Tidsrammen kommer på totalt 130 dager inkludert 20. mai. Dette gir utvikler gruppen 130 dager totalt, og 89 dager utenom helger og feriedager. Studieprogrammet er verdt 20 ECTS[6], hvert studiepoeng forventer at en student skal arbeide 25 til 30 timer i snitt. Dette fører til at total arbeidsmengde per gruppemedlem er mellom 500 og 600 arbeidstimer og totalt mellom 2000 og 2400 arbeidstimer for hele gruppen.

1.5.3 Økonomiske rammer

Oppdragsgiver nevnte aldri noe formelt budsjett før eller etter arbeidskontrakten var underskrevet. Likevel ble det uttrykt et ønske om å holde de økonomiske

[4] https://www.ntnu.no/studier/bidat
[5] https://www.ntnu.no/studier/bprog
[6] https://ec.europa.eu/assets/eac/education/ects/users-guide/index_en.htm

kostnadene så lave som mulig. Oppdragsgiver nevnte tidlig at de ønsket å oppgradere nåværende maskin som eksisterte i simulasjonsrommet, og hadde mulighet til å bidra økonomisk til dette formålet. Det var også avsatt penger til å kjøpe et passende kjørespill som kan brukes mens en ambulansesimulatoren benyttes.

1.5.4 Tilgang til simulasjonsrommet

Simulasjonsrommet skulle sjeldent bli brukt til andre formål gjennom vårsemesteret. Prosjektgruppen hadde derfor god tilgang til rommet. Det ble opprettet en uformell avtale mellom oppdragsgiver og gruppen om tilgang til simulasjonsrommet hver torsdag fra klokken 0900. rommet kunne benyttes hele dagen, med enkelte unntak hvis rommet skulle bli brukt for undervisning.

1.6 Rollefordeling

1.6.1 Gruppens roller

Prosjektleder: Aslak Tøn. Tar ansvaret for kommunikasjon innad i gruppen og eksterne grupper i samarbeid med prosjektgruppen. Prosjektleder tar avgjørende valg hvis gruppen ikke kan komme til enighet i en diskusjon. Prosjektleder tar også ansvaret for at utviklet programvare og prosjektrapport har en helhetlig og gjennomført stil. Prosjektleder har tidligere erfaring med prosjektledelse, både fra studie og utenom i fritidsaktiviteter og arbeid.

Hovedansvarlig "UI" og "UX" design: Donjetë Haziri. Dataingeniørene på gruppen har svært lite erfaring med god grafisk design. Denne arbeidsoppgaven falt derfor på personen gruppen mente hadde mest ekspertise innenfor både grafisk- og opplevelsesdesign. Hovedansvaret var å utvikle et intuitivt brukergrensesnitt for brukere av systemet. Etter en skisse var laget for brukergrensesnittet ble designdokumentet delt med hele prosjektgruppen. Dette ble så brukt for å utvikle et brukergrensesnitt i samhold med designdokument (mer info under avsnitt 4).

Utvikler: Isac Mikal Kallevig, Håkon Muggerud. Utviklerne tok større ansvar for utforsking av forskjellige tekniske løsninger, og hvordan å implementere disse i systemet. Disse arbeidsoppgavene inkluderte for eksempel: utvikling av 3d renderer, kommunikasjon mellom tråder, kommunikasjon med diverse I/O elementer. Arbeidet til utviklerne var mindre fokusert enn både prosjektleder og grafisk brukergrensesnitt ansvarlig. Dette førte til at arbeidsoppgavene var mer varierte og vanskelige å liste opp.

1.6.2 Øvrige roller

Arbeid med bacheloroppgaven inkluderte også en del roller utover kjernegruppen som blir vurdert for arbeidet.

Veileder: Frode Haug. Hver bachelorgruppe får tildelt en veileder gjennom semesteret. Formålet med en veileder er å gi gruppen generelle tilbakemeldinger

på foreløpig arbeid, og å gi struktur og arbeidsmetodikk til prosjektgruppen. Veileder har en time per uke tilgjengelig for gruppen. Dette fører til at veileder har noe mindre evner til å hjelpe direkte med tekniske problemer i utviklingen av programvare. Prosjektgruppen vil selv gjerne ha større kunnskaper om problemstillingen innenfor utvikling, og vil ofte komme til bedre løsninger der. Det er likevel mulig å forhøre seg med veileder om spesifikke problemer innenfor prosjektplanlegging og utvikling.

Oppdragsgiver: Hans Martin Lilleby. Oppdragsgiver var ansvarlig for utforming av oppgavebeskrivelsen og anskaffelse av nødvendige midler til prosjektgruppen. Prosjektet krevde hovedsakelig tilgang til simulasjon rommet, samt en datamaskin det nye systemet kunne installeres på.

1.7 Rapportens struktur

Rapporten inneholder de følgende kapitlene: Introduksjon, kravspesifikasjon, grafisk design, utviklingsprosess, teknisk design, implementasjon, kvalitetssikring, installasjon, diskusjon og konklusjon.

1.7.1 Introduksjon

Tar for seg rammer og bakgrunn for prosjektet, samt roller og strukturen til rapporten.

1.7.2 Kravspesifikasjon

Kravspesifikasjonen tar for seg de forskjellige kravene oppdragsgiver hadde for systemet. Dette kapittelet inkluderer også designdokumenter gruppen lagde for å oppklare hva som skulle være med i kravspesifikasjonen.

1.7.3 Grafisk design

Grafisk design inneholder metodikk og beskrivelser for hvordan gruppen kom fram til et grafisk brukergrensesnitt oppdragsgiver var fornøyd med. Dette kapittelet beskriver hvordan prosjektgruppen gikk fram for å utforme et intuitivt brukergrensesnitt.

1.7.4 Utviklingsprosess

Utviklingsprosess tar for seg hvilke metoder gruppen benyttet for å gjennomføre prosjektet.

1.7.5 Teknisk design

Teknisk design tar for seg valg av teknologier og andre løsninger gruppen benyttet seg av. Kapitelet tar også for seg flere begrunnelser for andre valg som blir beskrevet i andre kapitler.

1.7.6 Implementation

Implementasjon tar for seg hvordan gruppen implementerte spesifikke løsninger nevnt tidligere i rapporten. Det er også mange kodeeksempler i dette kapittelet. Kodeeksemplene viser til kode som eksisterer i systemet. Kapittelet inneholder også forklaringer til hvorfor denne koden ble brukt.

1.7.7 Kvalitetssikring

Dette kapittelet tar for seg hvilke metoder gruppen brukte for å sikre at systemet både ved ferdigstillelse, og etter senere oppdatering, skal fortsette å levere forventet nytteverdi.

1.7.8 Installasjon

Systemet har en rekke med eksterne kodebaser som er nødvendig for et fungerende program. Dette kapittelet beskriver hvordan og hvorfor man installerer forskjellige pakker for å få systemet til å bygge uten feil.

1.7.9 Diskusjon

Diskusjonskapittelet vil ta for seg all drøfting gruppen har gjort gjennom prosjektet, som ikke er blitt inkludert i tidligere kappitler.

1.7.10 Konklusjon

Inneholder konkluderende tanker fra gruppens side, både om hva som ble gjennomført bra, og hva som kunne forbedres. Seksjonen inneholder også hva gruppen anser som den beste veien videre med nye prosjekter innenfor emnet.

1.7.11 Vedlegg

Rapporten inneholder også en rekke vedlegg. Disse vedlegge er relevante for prosjektoppgaven, men hadde ingen passene seksjoner å bli satt under. Store filer ble også lagt til i vedlegg.

1.7.12 Utforming

Enkelte ord i rapprten benytter seg av hermetegn: “”. Dette benyttes hvis et engelsk låneord brukes og det ikke finnes en god norsk oversettelse. Andre ord i teksten benytter seg av `blok tekst`. Dette inkikerer at teksten er hentet ut fra prosjektgruppens kodebase.

2 Kravspesifikasjon

Oppdragsgiver ga igjennom flere møter forskjellige ønsker og krav til hva systemet burde kunne gjennomføre. Kravspesifikasjon ble tidlig oppklart i et møte 19. januar 2021. Enkelte ekstra funksjonaliteter ble også senere diskutert som mulige addisjoner til systemet.

2.1 Funksjonelle krav

Følge Sekvenser

Systemet må kunne følge en predefinert fil med data for sylinderne. Disse sekvens filene eksisterer i det gamle systemet brukt nå, og det er et behov for at denne funksjonaliteten også eksisterer i et nytt system.

Lage Sekvenser

Systemet må ha funksjonalitet for å kunne lage sekvenser, som kan spilles tilbake ved et senere tidspunkt. Nøyaktig hvordan disse sekvensene blir lagd var ikke spesifisert av oppdragsgiver, men det var gitt ønsker om å generere disse ved hjelp av installert ratt i ambulansen. Oppdragsgiver mente det også ville være nyttig å kunne lage disse sekvensfilene andre steder enn i kontrollrommet.

Fri styring av hydraulikk

Ambulansesimulatoren inneholder et digitalt ratt. Dette rattet er tidligere brukt for styring av hydraulikken. Det var et krav fra oppdragsgiver at denne funksjonalitet også eksisterer i det nye systemet, da dette er et sentralt element i opplæringen som simulatoren skal brukes til.

Nødstoppindikator

Under første installasjon ble et nødstoppsløyfesystem også installert. Dette systemet skal sikre at ingen blir skadet ved å oppføre seg usikkert rundt ambulansen under aktivt bruk. Nødstoppsløyfen er selvstyrt og trenger ikke å styres av det nye systemet. Det nye systemet trenger derimot å ha kontroll over status på nødstoppsystemet. Det kreves at en indikator er tilgjengelig i brukergrensesnittet som viser om nødstoppsløyfen er blitt aktivert.

Avbrytning av simulasjon

Administrator må ha mulighet til å avbryte simulasjonen når som helst. Det er behov for denne funksjonaliteten slik at administrator eller instruktør kan gi veiledning til brukeren av ambulansesimulatoren.

Intuitivt brukergrensesnitt

Det var gitt ønsker fra oppdragsgiver å ha et intuitivt brukergrensesnitt for kontrollering av ambulansesimulatoren fra kontrollrommet. Systemet vil bli brukt av flere personer enn tidligere. Det er dermed viktig at man ikke bruker unødvendig mye tid på opplæring for å bruke programvaren.

2.2 Ikke-funksjonelle krav

Responsivitet

Forrige system slet med responsivitet. Dette var spesielt tydelig når man forsøkte å styre ambulansen ved hjelp av mus. Det tok sekunder fra en bruker flyttet på musen, til systemet reagerte ved å bevege hydraulikken.

Miniatyrmodell

Oppdragsgiver har tidligere hatt problemer på mørke dager å se ambulansen fra kontrollrommet. Det var derfor gitt et ønske om å lage en 3d modell i programmet som kunne vises til administrator i kontrollrommet, slik at posisjonen til ambulansen alltid var kjent.

Visualisering

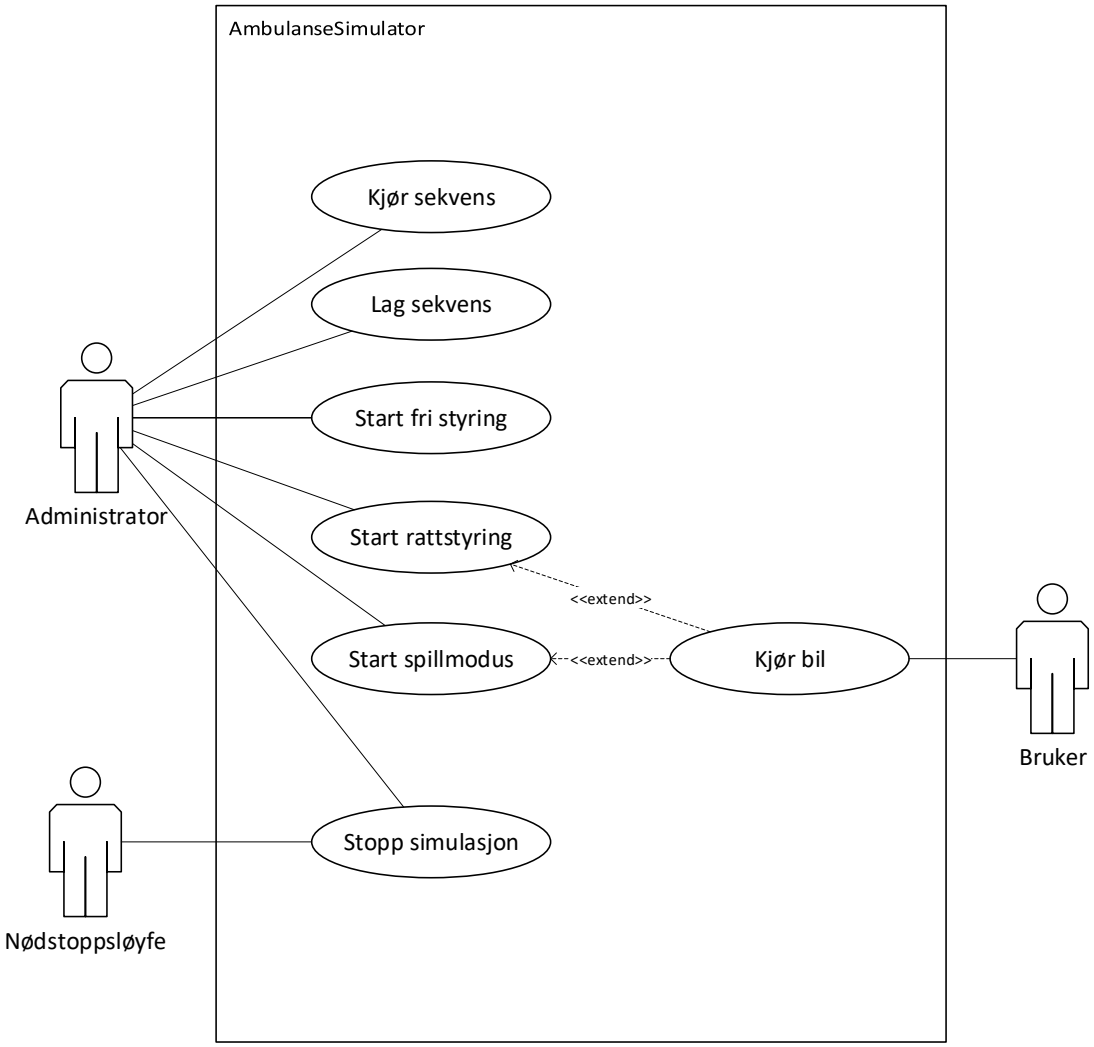
Det forrige systemet hadde en graf som viste volt-nivået til begge sylindrene i sanntid. Dette er nyttig for å få en mer nøyaktig visualisering enn hva man ser på miniatyrmodellen.

VR funksjonalitet

Det ble diskutert VR funksjonalitet som en mulig utvidelse til programmet. En sjåfør vil styre ambulansen og spille et spill med et VR-headset på seg. Dette ble ikke diskutert i detalj, men det var også mulighet for en kombinasjon med både AR og VR for å få en mer livsnær simulering.

2.3 Use Case

Gjennom diskusjoner i gruppen tidlig i utviklingsprosessen kom det fram et oversiktlig bilde av hva som var nødvendig funksjonalitet i systemet. Et overblikk av denne funksjonaliteten kan sees i figur C.1. I starten av utviklingsprosessen var tanken at programvaren skulle håndtere lesing av data fra ratt og pedaler samtidig som spillmodus var aktiv. Grunnet limitasjoner med Logitech sin maskinvare viste det seg at dette var vanskelig å implementere. Dermed er det ingen kobling mellom rattstyring og spillmodus. Dette ser man i endelig UseCase diagram i figur figur 2.1.



Figur 2.1: UseCase diagram for prosjektet

Navn:	Kjør Sekvens
Aktør:	Administrator
Mål:	Styre hydraulikk etter en definert sekvens
Beskrivelse:	Administrator gir programmet en predefinert sekvens som systemet skal følge for å styre hydraulikken.

Navn:	Start fri styring
Aktør:	Administrator
Mål:	Administrator styrer simulasjonen med mus
Beskrivelse:	Administrator kan starte styring av ambulansen fra kontrollrommet. Administratoren kan styre hydraulikken ved hjelp av musepekeren på skjermen.

Navn:	Start rattstyring
Aktør:	Administrator
Mål:	Start styring med ratt
Beskrivelse:	Administrator ber systemet lytte på det digitale rattet og pedalene montert i ambulansen. Når dette rattet roteres eller pedaler presses vil hydraulikken simulere, etter beste evne, hvordan disse kreftene vil føles for en person sittende i en bil.

Navn:	Start spillmodus
Aktør:	Administrator
Mål:	Styre hydraulikk vha. et spill
Beskrivelse:	Administrator kan velge å starte spillmodus. Dersom Euro Truck Simulator 2 er åpent i bakgrunnen vil hydraulikken simulere de reelle kreftene påført i spillet.

Navn:	Lag Sekvens
Aktør:	Administrator
Mål:	Lage en sekvens for senere avspilling
Beskrivelse:	Administrator kan velge å lagre en sekvens etter at simulering er ferdig. Dette kan være sekvenser generert gjennom fri styring, rattstyring og spillmodus

Navn:	Stopp Simulasjon
Aktør:	Administrator, Nødstoppsløyfe
Mål:	Stopp hydraulikken ansvarlig for å styre ambulansen
Beskrivelse:	Sikkerhetsmessig er det svært viktig at systemet stopper hvis en bruker gjør noe usikkert. Det er også nødvendig for administrator å kunne stoppe hydraulikken når de ønsker å prate med en bruker. Nødstoppsløyfen kutter strømmen til hydraulikken dersom den blir brutt.

Navn:	Kjør bil
Aktør:	Bruker
Mål:	Kjøre bil
Beskrivelse:	En bruker sitter i ambulansen. Ved hjelp av ratt og pedaler vil brukeren kjøre bilen på normalt vis. Dersom ”Start rattstyring“ er aktiv vil ambulansen bevege seg i samhold med hvordan brukeren styrer ambulansen.

3 Utviklingsprosess

3.1 Valg av utviklingsmodell

Gitt tidlige samtaler med oppdragsgiver ble det tydelig at det var klare krav om hva systemet trengte. Det ble derfor naturlig for gruppen å benytte seg av en plandrevet modell for utvikling av systemet. Det har senere kommet et par ekstra ønsker fra oppdragsgiver, slik som avspilling av lyd gjennom systemet, men dette har vært lette problemer å fikse.

Prosjektgruppen valgte å gå for en inkrementell sekvensiell utviklingsmodell [1]. Dette ga gruppen mulighet til å tidlig få en oversikt over hvordan systemet overordnet skal være strukturert, men gir fortsatt noe mulighet til fleksibilitet i utviklingen. En plandrevet utviklingsmodell gir også bedre grunnlag for et gjennomført og enhetlig system som burde være lettere å vedlikeholde.

Prosjektgruppen kunne valgt å bruke en fossefallsmodell for utviklingen av systemet, men ønsket å ha noe fleksibilitet i hvordan systemet ble utviklet. Dette ville gi mulighet til gruppen å forandre kurs, hvis det ble klart at ikke alle systemets krav var beskrevet i det originale designdokumentet.

Gjenbruksmodellen ble også vurdert som mulig, da det allerede eksisterte et system for styring av ambulansen. Det var derimot spesifisert av oppdragsgiver at de ønsket et nytt system. Prosjektgruppen bestemte seg derfor ikke å gå for en gjenbruksmodell. Prosjektgruppen vil trekke ut konsepter fra eksisterende system, og utvikle samme funksjonalitet på nytt i det nye systemet.

3.2 Trello

Gjennom utvikling av både systemet og rapporten var det behov for å ha kontroll over hvem som jobbet på hva. Trello var verktøyet benyttet av gruppen for slike arbeidsoppgaver. Trello ble brukt i form av at nye kort ble lagt til under møter, og personer la seg på kort etterhvert som det passet seg. Gruppemedlemmer la også til kort utenom møter, hvis dette medlemmet så behov for en spesifikk funksjonalitet. Dette ble gjort slik at flere gruppemedlemmer ikke skulle starte på samme arbeidsoppgave.

3.3 Designdokumenter

Tidlig i prosessen ble det utviklet forskjellige designdokumenter som skulle hjelpe gruppen med å utvikle systemet. Dette var et systemarkitektur diagram, et Use Case diagram og et designdokument for brukergrensesnittet.

3.3.1 Systemarkitektur

Hensikten med systemarkitektur-diagrammet var å vise den overordnede strukturen for systemet tidlig. Gruppen var klar over at det manglet enkelte klasser og metoder i dette diagrammet. Gruppen var enig i at dette kunne legges til systemarkitektur-diagrammet senere. Diagrammet kan bli funnet i figur C.2.

3.3.2 Use Case Diagram

Use Case diagrammet (sett i figur C.1) skulle gi prosjektgruppen et overblikk over hvilke funksjonaliteter systemet skulle gjennomføre. Diagrammet ble også brukt for å bekrefte bruksområdene til systemet med oppdragsgiver.

3.3.3 Designdokument brukergrensesnitt

Dette designdokumentet ble brukt som en veileder gjennom utviklingsprosessen for oppsett av brukergrensesnittet. Prosjektgruppen lette igjennom funksjonaliteten til wxWidget for å finne hvilke klasser som best kunne reprodusere utseende til det orginale brukergrensesnittet gitt av designdokumentet. Dette dokumentet er stort, og ligger derfor vedlagt med koden.

3.4 Møter

Gruppen holdt jevnlige møter for å sikre at det var enighet i gruppen. Gruppen holdt minst et møte i uken på tirsdager klokken 12.30. Ved behov kunne også ytterligere møter planlegges. Gruppen opplevde ikke at det var behov for flere møter, da mye av systemet ble designet i samarbeid i januar. Møtene kunne derfor fokusere på arbeidsoppgaver for uken og foreløpig status. Hyppigheten av møter kan sees fra referatene i vedlegg G.

3.5 Milepæler

Gruppen satte opp noen milepæler i sammenheng med utviklingen. Disse kan bli sett i Gantt-Diagrammet (se vedlegg D). Milepælene ble brukt som målepunkter, for å sikre at gruppen hadde kontroll på progresjonen i prosjektet.

3.6 Møter med veileder

Gjennom bachelor perioden ble gruppen tilbudt ukentlige møter med en veileder utpekt av NTNU. Veileder skulle ha en spesiell kunnskap innen feltet gruppen jobbet med. Gruppen hadde størst behov for møtene tidlig i utviklingsfasen, da usikkerheten var større. Arbeidsoppgavene fikk mer klarhet utover i mars og april. Hyppigheten til møtene ble dermed redusert. Dette kommer fram av møtereferater tatt av møtene med veileder (se vedlegg G).

3.7 Versjonskontroll

Utviklingen av systemet ble gjennomført ved at flere funksjonaliteter ble inkludert samtidig. Hovedsakelig i form av et kontroller-system ansvarlig for logikken i systemet ble utviklet parallelt med et grafisk brukergrensesnitt. Det ble også skrevet en tilhørende rapport til systemet. For å holde kontroll på systemet ble "git"[7] brukt

[7] `https://git-scm.com/`

som et versjonskontrollsystem. “Git” ble foretrukket av gruppen, da alle i gruppen allerede hadde noe kjennskap til hvordan å bruke systemet.

“Git” repoet ble også lastet opp til Bitbucket[8]. Bitbucket ble brukt da det er en avtale mellom NTNU og Atlassian[9]. Gitlab ble også vurdert da NTNU hoster en gitlab server, men ble valgt bort da det var usikkerhet rundt hvordan å sette opp et nytt repository ved `https://git.gvk.idi.ntnu.no/`. Dette var grunnet at NTNU sin gitlab hadde et begrenset antall repositories en kunne ha samtidig. Gruppen la ikke stor vekt på hvilken online git host som ble brukt, da alle er relativt like. Funksjonalitet ikke tilbudt av en “git” løsning kan relativt greit løses med eksterne verktøy.

3.8 Parprogramering

Parprogrammering var ikke en metodikk direkte koblet til utviklingsmodellen gruppen valgte. Parprogrammering kom fram som et behov etter en del tekniske problemer som dukket opp i utviklingsprosessen. Dette var spesielt problemer rundt Logitech’s digitale ratt (se avsnitt 5.4 for mer info), samt utvikling og sammenkobling av “frontend” grafikkløsning til “backend” logikken.

[8] `https://bitbucket.org/`

[9] `https://www.ntnu.no/wiki/display/copcse/bitbucket+-+git+repository`

4 Grafisk Design

Hva er UX testing?

UX[10](User eXperience eller brukeropplevelse) testing er en metode for produkttesting der man tester hvordan et produkt blir oppfattet og brukt av produktets målgruppe, for å da kunne forbedre produktet basert målgruppen sine tilbakemeldinger. Målet med denne metoden er å få et sluttprodukt som på best mulig og enkel måte kommuniserer hvordan den skal brukes, i hvilken situasjon den skal brukes, og at å bruke produktet er både lett og praktisk å bruke for målgruppen.

Hvordan fungerer UX testing

Før man kan starte UX prosessen trenger man en prototype. Denne prototypen er laget basert på research om målgruppen, hva målgruppen ønsker ut av produktet, hvilken problemstilling produktet skal løse, og hvordan andre produkter har løst de samme problemstillingene. For å starte UX testingen må man få tak i en testgruppe som enten er en del av målgruppen eller kan tilsvare målgruppen. Testingen går ut på å få testgruppen til å bruke produktet hvordan de ønsker og/eller føler er naturlig. Som testansvarlig skal man være nøytral og observere, og bare spørre spørsmål for å oppklare hva brukeren tenker og opplever ved å bruke produktet. I denne fasen finnes det ingen bruker feil, for det er her man skal finne problemene i designet.

Etter testingen er fullført vil man gå tilbake og forbedre prototypen. Denne prosessen gjentas til det ikke er noe mer som kan forbedres eller til tidsfristen er gått ut. Basert på produktet kan man velge mellom standard UX testing eller smidig UX testing. Standard UX testing betyr at prototypen forblir uavhengig av produktet gjennom hele løpet. Smidig UX testing itererer på prototypen, dvs. prototypen blir mer og mer avansert til det til slutt blir det ferdige produktet.

4.1 Gruppens UX prosess

Det ble ikke gjort mye research for programmet. Ettersom oppgaven var å gjenskape et eksisterende program, var det da å basere det nye programmet på det gamle. Det ble da å gå gjennom programmet for å se hvordan nåværende funksjoner var løst, snakke med oppdragsgiver om hvilke svakheter han mente programmet hadde, og liste opp alle de ønskede funksjonene programmet skulle ha. Når alt det var på plass og gruppen ble enige om designet gjennom kladder ble det laget en wireframe.

For dette prosjektet passet det best å bruke standard UX testing, ettersom testingen av brukergrensesnittet kunne gjøres separat fra programmet ved en interaktiv wireframe. Gruppen valgte denne metodikken da rammeverket brukergrensesnittet skulle utvikles i var relativt ukjent. Det ville dermed ta lang tid å iterere seg framover med en smidig metodikk for UX testing. Man slipper og deretter å skrive om en del kode for å forandre utseende basert på tilbakemeldingene

[10] https://www.hotjar.com/usability-testing/

fra testgruppen. Separasjonen av UX testing og utvikling gjorde det lettere å lære wxWidgets og fokusere på de riktige elementene ved å ha en klar mal å følge.

Gjennom hele UX prosessen ble det tatt utgangspunkt i at sluttprogrammet ville ha blitt utviklet ferdig med absolutt alle ønskede funksjonaliteter, uansett hvor realistisk det så ut til å være. Planen var at designet skulle dekke all planlagt funksjonalitet, så det var noe å følge uansett hvor prosjektet stoppet. Wireframen ble laget i pencil[11] for Ubuntu.

4.1.1 Testgruppen

Som nevnt i avsnitt 1.3 er målgruppen undervisere og studenter i paramedisin. Ettersom brukergrensesnittet var rettet mot underviserne i paramedisin, da en underkategori av helse og medisin, var det viktig å få tak i brukere med bakgrunn i enten helse og medisin eller undervisning, ideelt begge.

Ifølge Jacob Nielsen er 5 personer stor nok testgruppe pr runde for å teste et program [2]. Han mener at fler enn 5 er bortkastet tid og ressurser. Siden programmet var så lite og oppgaven hadde hovedfokus på utvikling, ga det mening å følge Nielsen sin størrelse. Det ble også bestemt å ha tre testrunder over tre uker, ettersom gruppen ikke ville bruke for mye tid på designet i et utviklingsfokusert oppgave. Hver runde bestod av å lage/endre på wireframen, finne deltakere, planlegge møter og reflektere resultatene. Dette ble antatt å ta omtrent en uke. Brukergrensesnittet skulle også være relativt lite, og derfor tenkte gruppen at tre runder burde holde for dette projektet.

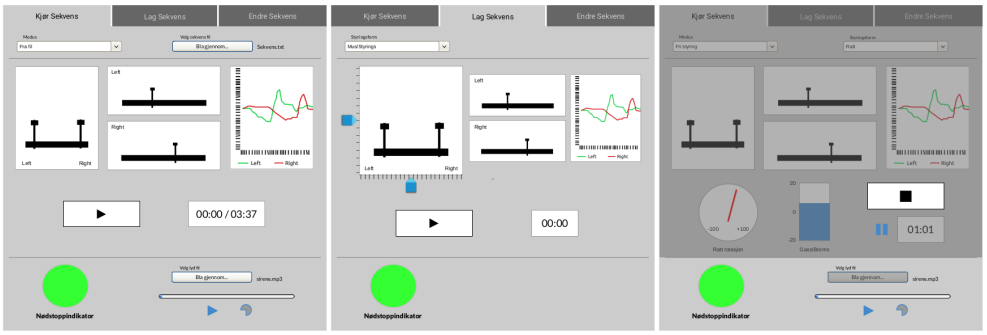
Ettersom oppdragsgiver er den som kom til ha hovedansvaret for programmet og den som kom til å bruke det mest var det viktig at han var med i hver test runde. Dette vil si at testgruppene i hver runde skulle ha oppdragsgiver og 4 tilfeldige deltakere som passet målgruppen. Størrelsen på testgruppen ble opprettholdt de første to rundene, selv om 3 av 10 ikke passet målgruppen. Dette var pga. frafall av oppsatt deltakere, der disse tre kom inn for å fylle opp kvoten på kort varsel.

GDPR ble fulgt ved å spørre hver enkel deltaker om det var greit at informasjon om dem ble lagret før testingen begynte. Informasjon som ble lagret var utdannings- og arbeidsbakgrunn, om de hadde syns- og/eller andre tilstander som kunne gjøre det vanskeligere å se eller lese på en skjerm og deltakerens tilbakemeldinger.

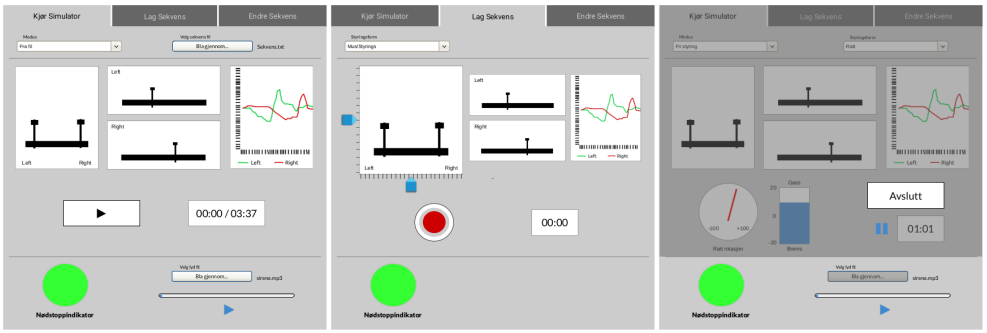
4.1.2 Testingen

Tilbakemeldingene etter første runde var for det meste positive om hvor lett det var å navigere og forstå. Det var noen tilbakemeldinger om små justeringer. Dette inkluderte f.eks. større skrift, og hva i gass/brems som var gass og hva som var brems. De viktigste tilbakemeldingene kom fra oppdragsgiver som ville ha mye bedre klarhet mellom start/spill knappene relatert til å “kjøre simulator fra fil”, “kjøre simulator uten fil” og “lage ny sekvens”.

[11] `https://pencil.evolus.vn/`



Figur 4.1: De største elementene fra første iterasjon av wireframe

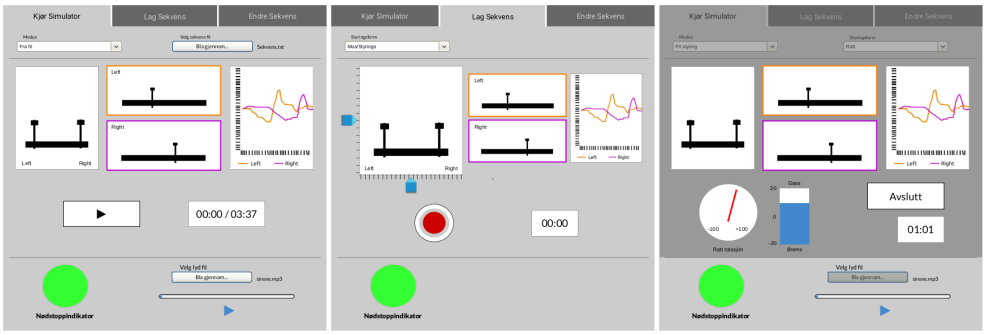


Figur 4.2: De samme elementene som i figur 4.1, i andere første iterasjon

Det var få forslag til endringer i andre runde. Noen av disse tilbakemeldingene var for små til å ha en dedikert test runde, f.eks "manuell styring"istedenfor fri styring". Andre tilbakemeldinger var ofte spesifike for den brukeren og ved å forandre design for denne brukeren, kunne det blitt mer forvirrende for flertallet. Disse forslagene på forandring ble tatt med videre til senere brukere, der de som regel enten ble bekreftet eller avkreftet.

Grunnet at det var såpass små forandringer fra runde to til runde tre, ble det heller kjørt en runde 2,5. Denne runden brukte omtrent samme wireframe som runde to, men med enn mer spesialisert testgruppe. Det ble her tatt inn to deltakere med spesielle kunnskaper innenfor emnet. Den ene var en utvikler som utviklet et lignende system, bare redningshelikopter i stedet for ambulanse. Den andre var en ingeniør som brukte og vedlikeholdt maskiner og programvare på et sykehus. Det var her noen større mangler på klarhet kom frem, f.eks. en farget ramme rundt høyre og venstre side av 3D modellen av ambulansen som matchet fargene på grafen. Det kom også fram at det burde unngås å benytte rødt og grønt hvis det ikke er relatert til stopp/gå eller galt/riktig. Dette design dokumentet hadde også en “Start”-knapp og en “timer” som var helt identiske. Det ble derfor oppfordret til å sikre at alle knapper hadde et unikt utseende.

Da runde 2,5 var over ble det laget en siste liten prototype med forandringer som kunne bli gjort basert på tilbakemeldingene. Sammen med oppdragsgiver ble det diskutert hva som skulle forandres og hva som skulle forbli fra design versjon



Figur 4.3: De samme elementene som i figur 4.1 og figur 4.2, i ferdig versjon

to i sluttdesignet.

Det siste som ble gjort før utviklingen av brukergrensesnitt startet var å ferdigstille design dokumentet, som er en siste interaktiv wireframe, og et fargekart som bestemte alle fargene som skulle brukes i programmet. Denne wireframen ligger vedlagt ved siden av rapporten. Fargene ble valgt ved hjelp av nettsiden davidmathlogic[12] for å unngå farger som kunne blitt til samme farge for fargeblinde. De ble og valgt basert på hvor mørke og lyse fargene var for å sikre klarhet for brukeren. Rød er f.eks. valgt mørkere enn grønn så selv om skjermen skulle blitt svart hvitt ville brukeren kunne se at det har skjedd en fargeforandring i programmet.

4.2 Brukergrensesnittet

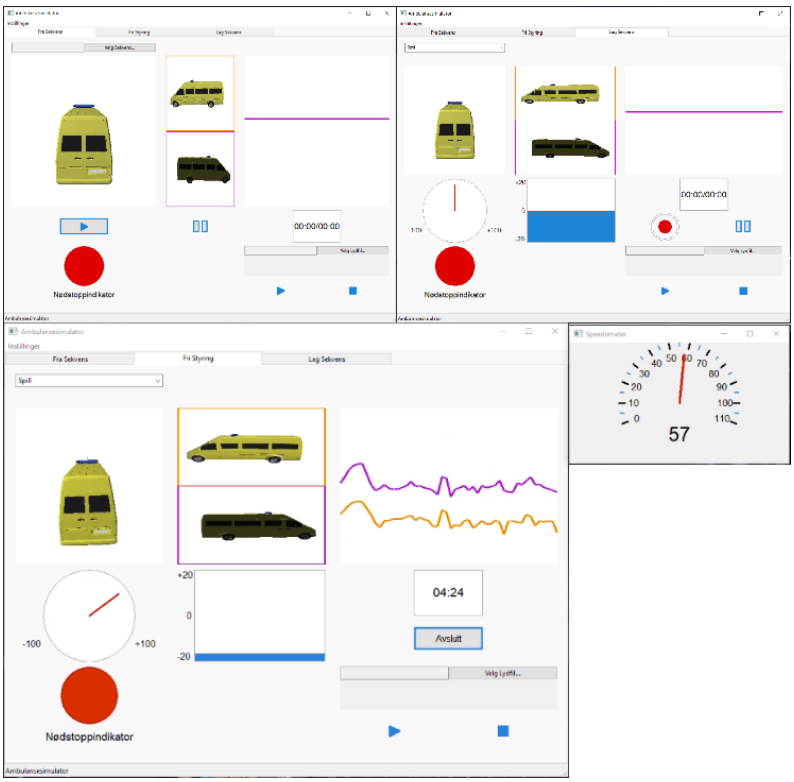
På grunn av at wxWidgets var en større utfordring enn det først ble anntatt, ble det en del forandringer fra wireframe som ikke bare var rent grafisk. Mye av det som blir nevnt her blir forklart under avsnitt 6, ettersom mye er basert på hvordan komponenter ble og måtte implementeres. Det er likevel nevnt her fordi det og har ført til klare forskjeller fra wireframen.

4.2.1 Grafiske forskjeller

Da UX prosessen startet var det fortsatt planen å bruke Qt for brukergrensesnittet. Da det ikke var mulig etter det ble funnet ut at Qt ikke ville fungere sammen med testene (mer under avsnitt 5.4) ble wxWidgets tatt i bruk. I Qt var det mye mer frihet i hvordan hvert element kunne se ut, ettersom Qt forventet at utvikleren ønsket å lage nesten alt fra bunnen av. wxWidgets derimot lager en del av elementene for utvikleren, sånn som popup-vinduer og knapper, med utseende basert på operativsystemet. Ettersom programmet hovedsakelig skal være på en PC med Windows, ble flere av de tilpassede elementene laget etter Windows sitt utseende. Blåfargene ble også forandret til å ha samme stil som Windows sitt fargepalett.

En av de største forskjellene man kan se i figur 4.4 fra figur 4.3 er at de tre fanene på toppen ikke lengre er “Kjør simulator”, “Lag sekvens” og “Endre sekvens”,

[12] `https://davidmathlogic.com/colorblind/`



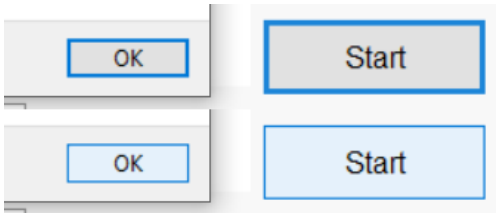
Figur 4.4: Elementer fra det ferdige programmet

men isteden “Fra sekvens”, “Fri styring” og “Lag sekvens”. “Endre sekvens” ble fjernet da nytteverdien var mye mindre enn arbeidet krevd for å løse problemet på en god måte. “Kjør simulator” ble også delt opp i to, en for når simulatoren kjøres fra fil, og en for alt annet av kjøring av simulatoren med fri kontroll. Disse to måtene å kjøre på hadde noen små, men klare forskjeller i backend. Gruppen følte dermed at det var lettere å skille dem i brukergrensesnittet.

Å kunne spille lydfil forble i alle faner. Mens den originale planen var å bare la brukeren spille lyd mens simulasjonen kjørte, var det ikke noen grunn til hvorfor den måtte bli begrenset til kun kjøring av simulator. Ideen for begrensningen var fordi funksjonen ikke skulle bli brukt utenfor “Kjør simulator”. Etter at “Kjør simulator” ble delt i to, og flere faner skulle ha tilgang til lydavspilling, var det lettere å implementere den i alle tre faner. Funksjonen påvirket heller ikke funksjonaliteten eller fokuset til “Lag sekvens”.

4.2.2 Mangel på brukervennlighet

At knapp for avspilling og stopping av lyd ble separert var fordi wxWidgets sin lydspiller ikke kunne sjekke om lyd faktisk spilte eller ikke. Det gjorde at det var stor sjanse for at feil grafikk kom i feil situasjon. Mens det heldigvis ikke stopper bruker fra å kunne stoppe lyd når de hører den, kan det bringe forvirring til bruker om lyd ikke spilles av. Bruker har ingen klar indikasjon på at programmet mener

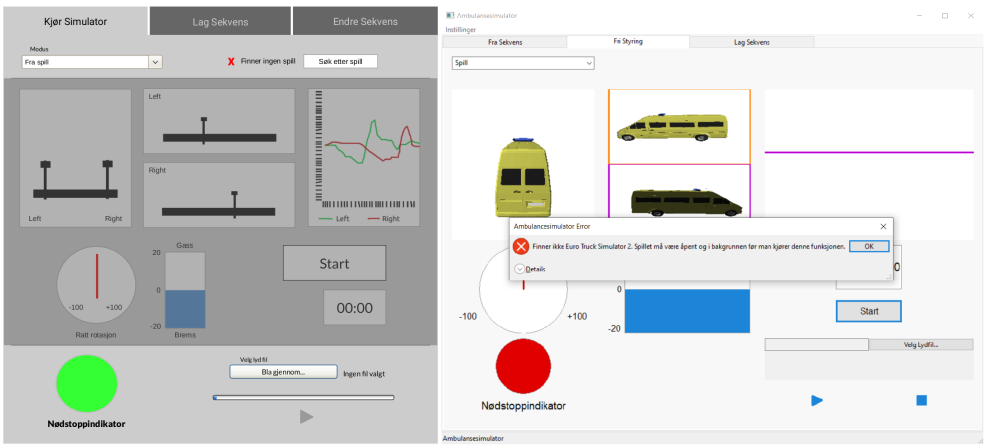


Figur 4.5: Vestre: Windows GUI. Høyre: Selvlagd knapp

at lyd spilles av. Dersom programmet ikke har registrert at bruker har trykket, om det er noe galt med programmet eller noe galt eksternt fra programmet er ikke tydelig.

En liten ting som ikke ble rukket å implementere var signal på at sekvens var lagret. Dette er ikke et stort problem ettersom bruker selv har valgt et sted de ønsket å lagre det og dermed kan gå og sjekke det opp selv, men det gjør at programmet ikke kommuniserer like klart som ønsket med bruker.

Noe som fikk en del skryt i UX test fasen var hvor klart det var når knapper ikke kunne brukes lengre pga at simulatoren var i gang. Dette var noe som dessverre ikke ble implementert i sluttproduktet. Kanskje med mer tid til å lære og forstå wxWidgets hadde det blitt laget en løsning på det. Dette lager da og et stort hull i programmet der brukeren kan skifte mellom faner selv under kjøring av ambulansen, der brukeren kan ende opp på feil sted og misforstå hva som skjer fordi de startet på et annet sted enn brukergrensesnittet forteller dem at de er. Løsningen på dette er da at om brukeren skulle trykke en startknapp et annet sted enn hvor de startet så vil det dukke opp et popup vindu som forteller at en simulering allerede er aktiv. Det samme gjelder å kjøre simulatoren fra spill. Dersom brukeren ikke har ETS2 oppe vil det komme et popup-vindu som forteller brukeren at programmet ikke kan finne spillet.



Figur 4.6: Vestre: Wireframe. Høyre: Brukergrensesnittet

Disse løsningene passer på at programmet ikke krasjer eller gjør noe den ikke skal, men er det ingenting som forteller brukeren at de er på feil sted eller at ikke

alt er på plass. Dette er et ganske stort tap i brukervennlighet.

4.2.3 Implementasjon vs brukeropplevelse

Ett element av designet som skapte litt diskusjon i gruppen var om "Lag sekvens" og "Fri Styring" skulle være samme panel eller holdes separat ettersom de fungerer likt i logikken til systemet. Eneste forskjellen i "Backend" er en bool som blir satt. Denne boolen kontrollerer om bruker skal bli spurt om de ønsker å lagre sekvensen etter de har trykket på stoppknappen. Dette ga spørsmålet om det var nødvendig å måtte produsere mer kode for et nesten identisk resultat. Dette er spesielt relevant da det allerede finnes brukergrensesnitt som har mulighet til å lagre en ny sekvens etter aktiv kjøring er ferdig. Dette kan sammenliknes med f.eks. moderne mobilkameraer. "Lag sekvens" og "Fri styring" har de samme prinsippene som blir utnyttet av mobilkameraer, der kameraet allerede er aktivt, men det er kun når brukeren trykker på "ta bilde" knappen at et bilde faktisk lagres. Gruppen kom til slutt til konklusjonen at det var mer brukervennlig å ha disse to funksjonalitetene delt opp i egne paneler i dette brukergrensesnittet.

Prinsippene som fungerer for et mobilkamera vil ikke fungere for dette programmet. For å oppnå noe lignende må simulasjonen først startes deretter må bruker trykke en knapp for å "ta bilde", dvs. "lagre". En løsning er en egen knapp som kunne blitt trykket på midt under simulasjonen og lagret sekvensen laget til da. En annen metode er at hver gang man trykket avslutt så kom det et spørsmål om lagring.

Figur 4.7: Lag sekvensknapper og Fri styringknappene separert

Første metoden ville ikke gått ettersom når sesjonen er over kan det være farlig å la simulatoren fortsette å kjøre mens bruker finner riktig mappe og skriver inn navn. Risikoen ved dette er en "Race Condition" i systemet, da lagring av sekvens skjer på hovedtråd, mens styring av hydraulikk skjer på en sidetråd.

Den andre metoden luker ut de farlige elementene ved å stoppe simulatoren med en gang, men lager og en ulempe for brukeren hver gang de ønsker å avslutte. Hovedbruken til programmet er å styre simulatoren for undervising. Det var et ønske om å kunne lage sekvenser, men det er ikke denne funksjonen som kommer til å bli brukt mest. Dette kan føre til at forespørselen om lagring av sekvens blir frustrerende for brukeren. Dette kunne blitt løst ved at den lagret automatisk hver gang med et ferdig navn som på et mobilkamera. På den andre siden ville brukeren da mistet evnen på å navngi sekvensene, og de ville hatt mange sekvenser de aldri ønsket.

En annen måte å få "Lag sekvens" og "Fri styring" til å fungere i samme panel, hadde vært å tenke på det som videoopptak. Slik som på et kamera starter man opptaket etter sekvensen er aktiv. Et kjent problem med kameraer er at brukeren tror de har startet et filmopptak, når de ikke har det. Dette skyldes at visningen av hva kameraet ser alltid er aktiv og de andre indikasjonene, som at en rød sirkel som blir til en firkant, er for små i forhold. I programmet er det flere store indikasjoner på at simulatoren kjører, sånn som 3D modellen, grafen og ambulansen. Dette tar ofte oppmerksomheten vekk fra mindre synlige indikasjoner som f.eks. formskifte fra runding til firkant. Det er to forskjellige paneler der den ene er en knapp med et hvitt og blått rektangel og den andre er en rød sirkel. Dette gjør det klart for brukeren hvilken modus de bruker, i forhold til et enkelt panel med begge knappene der den røde sirkelen blir til en rød firkant. Dette økte preferansen til gruppen for å skille "Lag Sekvens" fra "Fri Styring".

Figur 4.8: Lag sekvensknapper og Fri styringknappene kombinert

Et element fra mobilkameraer som passet bra i programmet var filmopptak knappen. Det var et element som fikk alle UX-deltakere til å forstå hva som skulle skje i "Lag sekvens" uten forklaring.

4.3 Hvorfor UX-testing

Programmet var allerede relativt lite, og det eksisterende brukergrensesnittet var forståelig. Det å sette sammen et nytt og fungerende brukergrensesnitt burde ikke ha vært ødeleggende for programmet. Ettersom oppgavebeskrivelsen ønsket et godt brukergrensesnitt, ville det vært dumt å bare kaste noe sammen og håpet på at det var forståelig. Håpet var at programmet skulle vare lenge. Det er stor sannsynlighet for endring i hvem som har ansvaret og skal bruke programmet, noe som gjør det lurt å passe på at en som ikke har rørt programmet før ikke har noen problemer med å bruke det i fremtiden.

Ved å bruke UX testing ble også visse problemstillinger lettet. wxWidgets er et bibliotek med mange gode verktøy, men er vanskelig å lære. Siden det nye systemet hadde et klart design å følge, var det lettere å lære elementer i wxWidgets og hvordan disse elementene skulle henge sammen. Dette så gruppen som en bedre løsning enn å prøve ut forskjellige implementasjoner og bruke det første som fun-

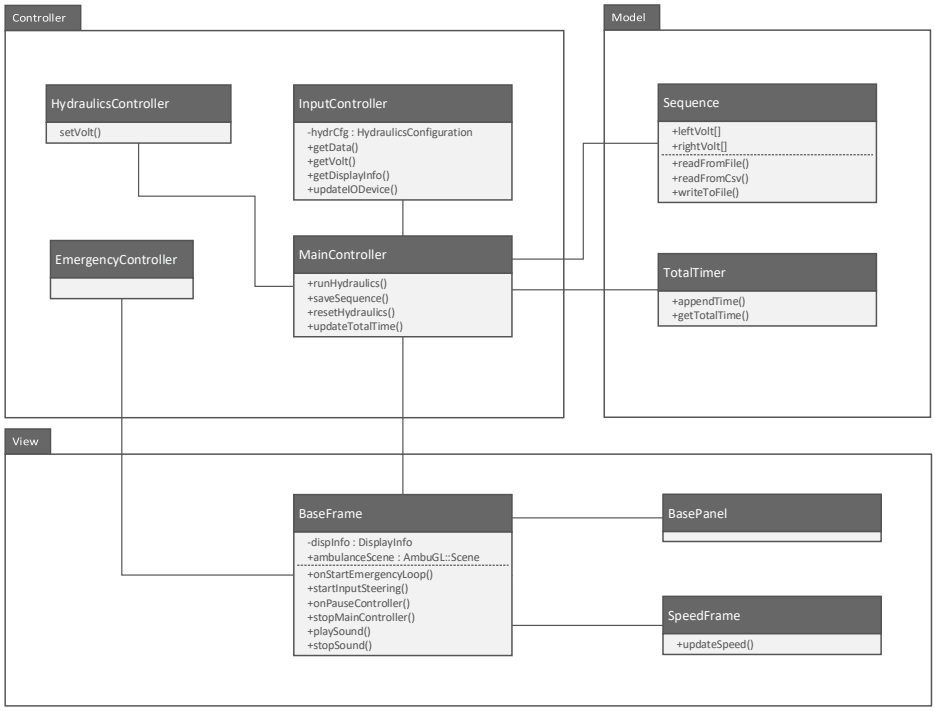
ket. Den sistnevnte metoden kunne økt sjansen for et lite brukervennlig program, ettersom målet ville gått fra å følge et planlagt design til å bare få elementer til å funke uten hensyn til om det hadde vært forståelig for brukeren.

Det ble bestemt å bare bruke UX testing for den grafiske delen av programmet. Dette ble bestemt ettersom de største endringene en bruker ville lagt merke til, lå i brukergrensesnittet. Utenom det grafiske, er det bare styringen av ambulansen som brukeren vil oppleve, der styring av ratt allerede har blitt perfeksjonert av spillratt produsenten, og styring med mus var basert på hvordan det tidligere programmet allerede fungerte. For styring med mus holdt det å implementere noen forskjellige metoder og la oppdragsgiver bestemme hva som var best. Det finnes sansyligvis en bedre måte å styre med mus, som kunne ha blitt funnet ved UX design. UX testing kunne blitt brukt, men det ville tatt altfor mye tid og ressurser på noe som allerede hadde en grei løsning. Hvis det skulle bli gjennomfør en UX testing for styring med mus, ville det blitt brukt en smidig UX metode.

5 Teknisk Design

5.1 Systemarkitektur

Gruppen bestemte seg for å følge systemarkitekturen til MVC (Model View Controller). Første utkast av arkitekturen kan finnes i figur C.2. Denne arkitekturen gjenspeiler i stor grad det som ble den endelige arkitekturen, med unntak av View, som fikk et stort antall flere metoder og objekter enn tidligere planlagt. Endelig systemarkitektur kan sees i figur 5.1.



Figur 5.1: Endelig systemarkitektur

5.2 Argumentasjon for valgt systemarkitektur

Det virket ganske naturlig å utvikle dette prosjektet i to deler: kontrollerene som styrte hydraulikken og brukergrensesnittet som presenterte det som skjedde i bakgrunnen. Disse to kunne fungere uavhengig av hverandre ved å bruke MVC, og det gjorde det derfor lettere å jobbe parallelt. Det var på en annen side usikkert hva som skulle være med i modellen. Modellen virket litt tynn, da det ikke jobbes med store mengder data, men fordi det var et tydelig "View" og en tydelig "Controller" inne i bildet ble MVC valgt.

5.3 Alternative valg av systemarkitektur

Et alternativ til systemarkitektur kunne vært en lagdelt arkitektur. Denne lagdelte arkitekturen kunne naturlig blitt delt inn i et lag for presentasjon og et lag for business logikk. På en annen side ville dette i praksis vært MVC presentert som en lagdeling arkitektur, noe som gjorde det mer naturlig å velge MVC.

Et annet alternativ ville vært en microservice arkitektur. Mange delelementer kunne mulig blitt utviklet separat fra hverandre, og blitt utviklet som egne bibliotek, noe som kunne gitt lav kobling. Eksempler på dette kunne vært å lage et bibliotek for å kommunisere med ETS2 spillet, eller et bibliotek for å lettere kunne kommunisere med rattet. Bakgrunnen for at dette ikke ble hovedarkitekturen var at det var vanskelig å definere hva som var en microservice. En microservice kunne potensielt gjøre så lite som mulig, og ha en veldig liten oppgave. En annen tolkning for en microservice kunne vært et bibliotek som håndterer flere oppgaver, men innenfor et avgrenset område, som f.eks. å kommunisere med ETS2. Grunnet en del usikkerheter og uenigheter virket det vanskelig å gå videre med en ren microservice arkitektur.

5.4 Eksterne teknologier

wxWidgets kontra Qt

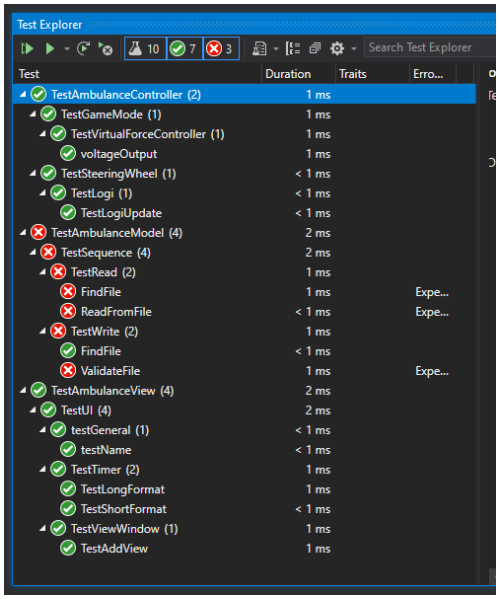
For å utvikle brukergrensesnittet ble wxWidgets tatt i bruk. wxWidgets er et C++ bibliotek for utvikling av brukergrensesnitt[13].

Grunner til at wxWidgets ble valgt, var at det var åpen kildekode, noe som gjorde det lett å finne dokumentasjon. Det hadde støtte for tabs, OpenGL og tråder, og det var til dels intuitivt å bruke. I tillegg fulgte det operativsystemets standard for brukergrensesnittdesign, samt hadde implementert mange ferdige brukbare widgets.

I utgangspunktet var planen å bruke Qt[14], som er et annet rammeverk for utvikling av brukergrensesnitt. Ulempen med Qt er et det er et stort rammeverk som krever en del konfigurering og arbeid for å få det koblet sammen med andre prosjekter. Fordelene med wxWidgets er at det er mindre, men tilfredsstiller likevel alle kravene og behovene i dette prosjektet. Etter gruppens egne erfaringer gjort de første ukene av bachelorprosjektet er det også bedre dokumentert, og lettere å finne ressurser. En annen erfaring som ble gjort var at Qt hadde mange ekstra “dependencies” som ble inkludert i bakgrunnen. Dette kunne bli automatisk satt opp, men problemet var at dette måtte settes opp manuelt dersom et annet testrammeverk enn Qt testrammeverket ble benyttet. I dette prosjektet skulle Google sitt testrammeverk benyttes, da dette var integrert opp imot Visual Studio. Det å bruke Qt sammen med et slikt testprosjekt viste seg å føre til kryptiske kompileringsfeil. Det virket derfor mer lønnsomt å bruke wxWidgets, da dette fungerte fint sammen med alle de andre rammeverkene og bibliotekene som ble benyttet.

[13] https://www.wxwidgets.org/

[14] https://www.qt.io/



Figur 5.2: Eksempel på en gjennomført enhetstest

National Instruments

Det var allerede utstyr fra National Instruments fra før for å styre hydraulikken: en CompactDAQ. Det var derfor åpenbart at det skulle fortsettes å benytte seg av denne teknologien.

For å programmere dette utstyret benyttes NI-DAQmx[15], en driver i fra National Instruments som kommer med en SDK å programmere National Instruments DAQ-er.

Det eneste dette prosjektet trengte fra en slik CompactDAQ var å kunne skrive til to analoge outputs, altså venstre og høyre sylinder, og å kunne lese av en digital input, nødsløyfen. Dette er abstraktert bort i `HydraulicsController` klassen.

Visual Studio Test Adapter for Google Test

Fordi det var ønskelig å kunne kjøre enhetstester i Visual Studio, var det var hovedsakelig to testrammeverk å velge mellom. Ironisk nok fungerte Google testrammeverket bedre for dette prosjektet enn Microsoft sitt eget testrammeverk. Microsoft sitt rammeverk gav kryptiske feilmeldinger når det ble brukt sammen med andre rammeverk, noe Google testrammeverket ikke gjorde. Dette er hovedgrunnene til at Google testrammeverket ble valgt i dette prosjektet. Det kan kjøres i Visual Studio, og det fungerer bra sammen med andre rammeverk.

Det å utvikle tester i Visual Studio gjorde det mulig å kjøre testene etter at programvaresom var praktsik i forhold til å oppdage feil tidlig. Et eksempel på hvordan dette ser ut kan sees i figur 5.2

[15] `https://www.ni.com/en-us/support/downloads/drivers/download.ni-daqmx.html`

Logitech Steeringwheel SDK

Oppdragsgiver kjøpte i løpet av utviklingsperioden et nytt digitalt ratt. Det nye rattet var et Logitech G920[16]. Tilhørende til rattet var et SDK for å hente og sende data til rattet[17]. Rattet skulle brukes til direkte styring av hydraulikken, eller indirekte gjennom ETS2.

Rattet måtte både kobles opp mot prosjektgruppens eget system, samt opp mot ETS2. ETS2 hadde allerede en egen løsning for kobling av rattet mot spillet, gruppen trengte derfor ikke å utvikle denne funksjonaliteten selv. Gruppen måtte derimot koble inn Logitech's SDK i eget system.

Dette ble løst ved å lage en `WheelController` klasse som tok ansvaret for kommunikasjon fra og til rattet. Klassen abstraherer bort mesteparten av funksjonaliteten til Logitech Steering Wheel SDK. Det er ikke mulig å kontrollere rattet ved bruk av `WheelController` klassen. `WheelController` klassen gjennomfører selv styringsopperasjoner eller "setters" på rattet. Spesifikt kjører den forskjellige "setters" for avspilling av krefter man ville følt i et ekte ratt.

Dette er en "spring force" som er en sentrerende kraft mot et midtpunkt. I dette tilfelle er midtpunktet nøytralverdien for rattet (hverken svinger til høyre eller venstre). Denne kraften blir brukt hvis bruker tråkker på gassen. Rattet har også en "damper force" som er invers proporsjonal med hvor langt nede brukeren tråkker på gassen (0%-80% av maks, avhengig av hvor mye gasspedal er tråkket ned). "Damper force" er en treghetskraft som hindrer rattet i å endre sin nåværende posisjon. Disse kreftene er konfigurert slik gir en relativt grei opplevelse av realistisk rattkrefter ved kjøring av en faktisk bil.

`WheelController` vil også hente ut informasjon om nåværende posisjon til rattet, samt akselerator og bremsepedal. Denne informasjonen vil lagres og brukes videre for å beregne hastighet og retning til en virtuell bil. Egenskapene til denne virtuelle bilen blir så brukt for å generere volt til venstre og høyre hydrauliske sylinder.

Endelig implementasjon av `WheelController` viste seg å være relativt enkel, men veien fram til endelig kode var svært tidskrevende. Gruppen brukte først lang tid på å forstå hvordan å hente ut data fra rattet, som var den viktigste funksjonaliteten som krevdes. Gruppen lagde testprosjekter i eget system for å prøve å hente ut og skrive denne dataen til skjermen. Ingen av disse prosjektene ga ønsket resultat. Rattet ga beskjed om at det var koblet til programmet riktig, men ville ikke skrive ut nåværende data. Gruppen brukte mer en 80 arbeidstimer på å løse dette problemet. Grunnen stammet fra flere problemer i metodikken til test prosjektene. Første problem var at rattet kun kobles til et program av gangen. Rattet vil så kun kommunisere med det ene programmet. Det vil derfor være lite nytte i å benytte seg av forskjellige debug verktøy innebygd i en IDE, som breakpoints og steg for stegs gjennomføring av kode, siden det ikke er IDE'en som er koblet til rattet. Det andre problemet viste seg å være at rattet måtte ha et faktisk vindu den var koblet opp mot for at henting av data skulle være mulig. Gruppen

[16] `https://www.logitechg.com/en-us/products/driving/driving-force-racing-wheel.html`

[17] `https://www.logitechg.com/en-us/innovation/developer-lab.html`

brukte et konsollvindu for testprosjektet. Selv om rattet klarte å initialisere seg med dette konsollvinduet, ville ikke rattet hente ut data hvis den var koblet til et konsollvindu.

Når rattet fikk et nytt testprosjekt med et mer tradisjonelt vindu var det mulig å hente ut data og skrive data til rattet. Dette viste seg å være det største hinderet i sammenheng med Logitechs SDK. Det er uheldig at dokumentasjonen som tilhørte til Logitechs SDK ikke spesifiserte at rattet trengte et dedikert vindu, og ikke ville fungert gjennom et konsollvindu.

Det er fortsatt noen kjente problemer rundt Logitech SDK. Tilkobling til rattet gjennom det nye systemet vil slutte å fungere hvis man prøver å koble seg til rattet uten at Logitech GHUB er åpen i bakgrunnen[18]. Programmet må startes på nytt før man kan koblet til rattet i dette tilfellet. Programmet har også hatt problemer med å koble til rattet hvis programmet er åpnet før USB'en til rattet er koblet til PCen. Gruppen er usikker på hva som forårsaker disse problemene. Problemene er relativt små, og gruppen har ikke ressurser for å finne årsaken. Disse problemene er derfor fortsatt i systemet.

Euro Truck Simulator 2

Modding av et spill opp imot ambulansesimulatoren kan være en tidskrevende prosess. Det var derfor kun blitt valgt ut ett spill i dette prosjektet for å kunne spilles opp mot simulatoren.

ETS2 var det spillet som passet best sammen med de kriteriene for hvilken type spill som var ønskelig å benytte i sammenheng med ambulansesimulatoren. Det var ønskelig å bruke et spill hvor det var mulig å kjøre i byer med en del sammensatte veger med kryss, rundkjøringer og fartsdumper, i tillegg til å ha fungerende trafikkregler. Det var og ønskelig at spillet også kunne spilles uten internettforbindelse. ETS2 har fungerende trafikkregler, og har et større nettverk av veger og landeveger. Spillet kan starte uten internett, selv om det går gjennom plattformen Steam. Vegene i ETS2 klarer også fortsatt å genere interessante sekvenser man kan gjøre medisinske øvelser i. ETS2 har også støtte for VR, noe som potensielt kunne ha blitt en del av prosjektet.

På en annen side er spillet mindre fritt og har flatt terreng. Offroad kjøring uten modding er også meget begrenset. Fordelene med offroad er at terrenget mange ganger kan genere mer interessante sekvenser, da det er mer variert terreng. Terrenget i ETS2 er i liten grad varierende, men som nevnt over, er det fortsatt variert nok til å kunne generere noen interessante sekvenser. Av andre spill som ble undersøkt, har flertallet et stort fokus på racerkjøring, og en del av disse spillene hadde ikke trafikkregler implementert, eller var oversimplifisert. GTA V var en sterk kandidat, da både kjøre simulasjonen og kjøre-fysikken i spillet var meget godt implementert. Den hadde et stort nettverk av veger i alle fasonger og størrelser, i tillegg til eksisterende kjørbar ambulanse i spillet. Spillet hadde også et stort modding community som ville gjort det lettere å modde. GTA V var på en annen side kjent for å inneholde kontroversielle elementer, men dette var noe som

[18] `https://www.logitechg.com/en-us/innovation/g-hub.html`

kunne ha blitt moddet bort. Hovedgrunnene til at GTA V ikke ble valgt var fordi spillet krevde internettforbindelse for å bli spilt. Det måtte ha blitt moddet i større grad for å få bort uønskede elementer. GTA V måtte også moddes for å inkludere andre ønskede elementer som f.eks. bedre kjøretøy kontrollere, noe ETS2 var mye bedre på.

Open Graphics Library (OpenGL)

For å visualisere en ambulanse i brukergrensesnittet, måtte et eksternt grafikkbibliotek benyttes.

Biblioteket GLEW[19] med OpenGL[20] ble tatt i bruk. Istedenfor å bruke GLM[21], et annet matematikk-bibliotek ofte brukt i sammenheng med OpenGL, ble det utviklet 3D matematikk fra bunnen av. "stb_image.h"[22] er brukt for å bruke bilder sammen med OpenGL.

OpenGL ble tatt i bruk fordi wxWidgets hadde støtte for OpenGL, og det er godt anerkjent, noe som gjør det lett å finne dokumentasjon, i tillegg til at det er støttet av de aller fleste grafikkort. Det er også raskt å utvikle grafikk og krever få kodelinjer med OpenGL, noe som var viktig med tanke på at koden skal være vedlikeholdbar, i tillegg til den begrensede tiden på dette prosjektet. Noen av ulempene med OpenGL er at feilhåndteringen ikke er av den beste sorten. Feilkoder blir returnert fra funksjoner, men hva feilkodene betyr må slås opp på internett. OpenGL er også i mindre grad konfigurerbar enn biblioteker som Vulkan[23], grunnet at dette er blitt abstraktert bort for å kunne kode grafikk med færre linjer med kode. Dette kan leses mer om i `CoreToolz` under avsnitt 6.8.

Et alternativt bibliotek som kunne ha blitt tatt i bruk var Vulkan, en allerede godt anerkjent kandidat. Vulkan er også støttet av de fleste nye grafikkort, og gir mange flere muligheter til å konfigurere det som skjer i bakgrunnen. Vulkan tar på en annen side mye lengre tid å lære, grunnet dens lavnivå natur og overveldende mengde med kode som må skrives for å få til det samme som OpenGL. Grunnet tidsbegrensning og usikkerhet på om det støttes av wxWidgets er dette valgt bort.

5.5 Programmeringsspråk: C++

Det var mange muligheter for valg av programmeringsspråk, samt mange styrende elementer for hvilke programmeringsspråk som skulle bli brukt i dette prosjektet. Disse styrende elementene var at koden skulle være vedlikeholdbar og det valgte språket skulle by på lærerike utfordringer.

Prosjektet er utviklet i ren C++. Grunnene til dette er at det skulle utvikles et "Real Time" system som knytter sammen mange elementer, hvor det er ønskelig at koden kompileres av ytelsesmessige årsaker. C++ er objektorientert, noe som gjør koden enklere å skrive, lese og forstå med færre kodelinjer. Prosjektgruppen

[19] `http://glew.sourceforge.net/`
[20] `https://www.opengl.org/`
[21] `https://www.opengl.org/sdk/libs/GLM/`
[22] `https://github.com/nothings/stb/blob/master/stb_image.h`
[23] `https://www.vulkan.org/`

er kjent med språket i fra før, noe som førte til mer tid til utvikling og mindre tid på opplæring. NI-DAQmx, ETS2 og Logitech Steering Wheel støttet også C/C++ direkte. I tillegg ønsket flere av gruppemedlemmene å lære mer om linkeren og hvordan biblioteker brukes sammen med C/C++.

En sterk kandidat til C++ var C#. C# er også støttet av ETS2 og Logitech Steering Wheel, og bruker de kjente .NET eller .NET Core rammeverkene. C# i likhet med C++ er også objektorientert og har god ytelse. I tillegg eksisterer det gode og enkle verktøy for å utvikle brukergrensesnitt i C# i Visual Studio, som er "native" for Windows plattformen, med en editor hvor man kan lage et brukergrensesnitt visuelt. Grunnen til at C++ ble valgt over C# var at det ble antatt at det ville være en mer lærerik prosess å bruke C++, grunnet at gruppemedlemmene hadde lite erfaring i fra før med å bruke biblioteker.

Til slutt er det verdt å nevne to andre kandidater. Python er et programmeringsspråk som også støtter alle biblioteker nevnt over, da det uansett hadde vært mulighet for å lage en C-wrapper. Grunnen til at Python ikke ble valgt, var fordi det er et runtime programmeringsspråk og kompileres ikke. Programmet blir heller kjørt igjennom en "interpreter", noe som er mye mindre effektivt på litt mer kompliserte algoritmer. I tillegg kjører Python i praksis kun på en tråd, da tråder som kjøres må dele en felles ressurs. En siste kandidat verdt å nevne er LabView[24], da dette ble brukt tidligere, i tillegg til at det er et produkt av National Instruments, noe som at gjør den direkte er støttet av NI-DAQmx. LabView kan også brukes til å lage brukergrensesnitt ment for maskinvare, DAQ-er og målinger, noe som kunne ha vært en fordel. Grunnen til at LabView ble valgt bort var at det samme kunne gjøres i C++, noe gruppemedlemmene var kjent med i motsetning til LabView. LabView er visuell programmering, som ikke bare var et ukjent programmeringsspråk, men et ukjent programmeringskonsept for gruppemedlemmene.

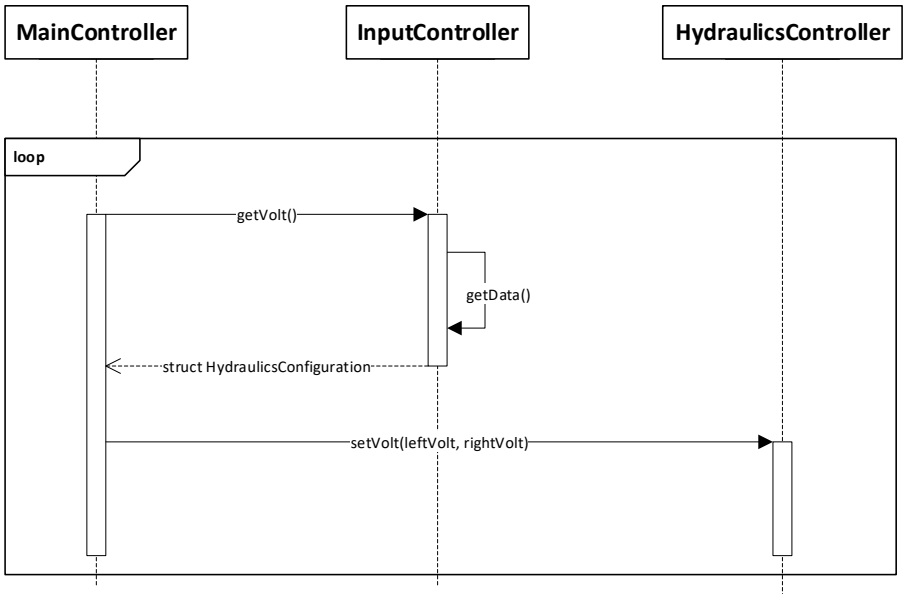
5.6 Styring av hydraulikk

Systemet benytter seg av flere I/O elementer for å gi en bruker mulighet til å styre ambulansen. Disse elementene inkluderer: musepeker, digitalt ratt, sekvensfil og et dataspill. Originalt ble alle disse elementene implementert med en egen metode for å lese av input fra bruker, og gi beskjed videre til hydraulikken om hvordan den burde bevege seg. Etter diskusjon i gruppen ble det etterhvert klart at den generelle prosessen for å styre hydraulikken kunne bli generalisert. Denne generaliseringen ble fremstilt i et systemsekvensdiagram, som kan bli sett i figur 5.3.

Ved å bruke en generalisert struktur kan man simplifisere prosessen for styring av hydraulikk. En simplifisert struktur vil så hjelpe en vedlikeholder av systemet å få oversikt over hvordan systemet fungerer. `InputController` inneholder generelle virtuelle funksjoner som kan få ny definisjon i klasser som arver fra `InputController`. I figur 6.1 ser man et eksempel på hvordan dette kan gjøres.

Ved å sende inn forskjellige typer controllers vil metoder som `inputCnt->updateIODevice()` ha forskjellige implementasjoner. Dette gjør ho-

[24] `https://www.ni.com/en-us/shop/labview.html`



Figur 5.3: Systemsekvensdiagram for generell styring av hydraulikk

vedløkken svært oversiktlig, slik at debugging relatert til hovedløkken kan gjøres i `MainController`. Mens problemer med en spesifikk input kan debuges i dens input klasse.

6 Implementasjon

I denne delen forklares og begrunnes implementasjonen av utvalgte områder av programvaren. Kapittelet viser også til flere kodeeksempler som er implementert og brukes i det nye systemet utviklet av prosjektgruppen.

Et overblikk over endelig implementasjon kan sees i klassediagrammet figur C.3.

6.1 Input Controller

Hydraulikken blir styrt ved å sende inn spesifikke voltverdier til DAQ'en. Disse voltverdiene blir generert ved hjelp av forskjellig utstyr i det nye systemet: mus, sekvensfil, ratt og pedaler eller spill. Felles for alle disse styringstypene er at de har en form for avlesning av input data. Siden disse hadde en del likheter, men likevel noen forskjeller, var det muligheter for å redusere kodemengden.

Det ble laget en klasse, `InputController`. Alle kontrollere som styrer hydraulikken arver i fra denne klassen. Klassen inneholder fire virtuelle metoder som vil bli overskrevet. Disse virtuelle metodene har ulike ansvarsområder, men implemententasjonen varierer fra de ulike kontrollerene.

Under ser man et eksempel på hvordan `InputController` blir brukt i programvaren. `updateIODevice()` er en funksjon som kun brukes av ratt og pedaler. Denne må kalles mye oftere enn de andre funksjonene slik at rattet påføres riktig mengde krefter. Dette er grunnen til at det er en ytre løkke med `updateIODevice()` og en if-test med `iterations++ % 10 == 0`. Denne if-testen sikrer at indre løkke kun blir kjørt hver tiende gang ytre løkke blir kjørt. `getData()` og `getVolt()` vil hente og konvertere data til volt som skal sendes til hydraulikken. `getDisplayInfo()` vil sørge for å returnere leselig data som vil vises i brukergrensesnittet.

Bruken av en superklasse gjorde det mulig å oppnå mer med færre linjer med kode, da alt var en `InputController` som kunne kalle sine virtuelle metoder. Et alternativ kunne vært å ikke bruke en superklasse, hvor hver kontroller var av en egen unik type. Dette kunne blitt løst ved å teste hvilke kontroller som skulle ha blitt brukt ved f.eks. en switch eller if statements. Det hadde vært praktisk om alle klassene hadde en variabel som fortalte hva slags kontroller de var. Denne

```
while (!TestDestroy())
{
    inputCnt->updateIODevice();

    if (iterations++ % 10 == 0)
    {
        hydrCfg = inputCnt->getVolt();
        displayInfo = inputCnt->getDisplayInfo();

        hController.setVolt(hydrCfg.voltageLeft, hydrCfg.voltageRight);
    }
}
```

Figur 6.1: Hovedløkken for styring av hydraulikk

variabelen kunne bli brukt i slike tester, noe som gjorde det mer naturlig at de arvet fra en superklasse uansett. Fellestrekkene var så mange at det ville ha ført til en del store likheter i implementasjonen mellom dem, som også gjorde det mer logisk å arve i fra en superklasse. I tillegg gjør det kontrollerene mer konsistente og er med på å standardisere hvordan de oppfører seg.

6.2 Brukergrensesnittet

Det å utvikle et brukergrensesnitt kan by på flere problemer. Kode skal helst ikke dupliseres, og elementene skal kommunisere med hverandre, og være mest mulig synkronisert med det som skjer i "backend".

6.2.1 Layout

Brukergrensesnittets layout er laget og modellert etter UX Design prosessen, men er blitt modifisert for å gjøre det enklere å implementere. Brukergrensesnittet består av 3 "notebooks". "Notebooks" i wxWidgets er i praksis faner. Disse har titttlene "Fra sekvens", "Fri styring" og "Lag Sekvens". Disse tre utgjør en tre-siders applikasjon, og er lagt til i hovedrammen kalt `BaseFrame`. Alle disse sidene har en del fellestrekk. Alle har tre displays av ambulansen, et grafdisplay, en nødstopp-indikator, muligheten til å spille av lyd og en timer. Derfor er det laget en en baseklasse kalt `BasePanel` som setter dette opp, i tillegg til tomme rom hvor ting kan legges til i klasser som arver i fra `BasePanel`. En annen metode som er brukt for å unngå duplisering av kode er å lage egendefinerte widgets som i seg selv har sin egen interne trestruktur. Et eksempel på dette er avspilling av lyd.

Koden under viser konstruktoren til `BasePanel`. Det er i denne konstruktoren felles layout blir satt opp.

```
BasePanel::BasePanel(wxNotebook* t_parent, BaseFrame* t_mainFrame) :
    wxPanel(t_parent)
{
    m_mainFrame = t_mainFrame;
    // Sizers for layout
    m_hbox = new wxBoxSizer(wxHORIZONTAL);
    m_fgs = new wxFlexGridSizer(4, 3, 9, 25);
    m_sideViews = new wxBoxSizer(wxVERTICAL);
    m_startPanel = new wxBoxSizer(wxVERTICAL);

    // 1st row is reserved for later use.

    // 2nd Row, displaying the ambulance.
    m_glBackView = new AmbuGL::Canvas((wxFrame*)this);
    m_glLeftView = new BorderView((wxFrame*)this, g_mainOrange);
    m_glRightView = new BorderView((wxFrame*)this, g_mainPurple);
    m_sideViews->Add(m_glLeftView, 1, wxEXPAND);
    m_sideViews->Add(m_glRightView, 1, wxEXPAND);
    m_glGraphView = new GraphView(
        (wxFrame*)this,
        -9,
        6.5f,
        GRAPH_POINT_SIZE,
        m_mainFrame->graphData.voltsLeft,
        m_mainFrame->graphData.voltsRight
```

```
    );

    // Set the cameras of the Canvases:
    m_glBackView->setScene(m_mainFrame->ambulanceScene);
    m_glBackView->setCamera(m_mainFrame->backCamera);
    m_glLeftView->setScene(m_mainFrame->ambulanceScene);
    m_glLeftView->setCamera(m_mainFrame->leftCamera);
    m_glRightView->setScene(m_mainFrame->ambulanceScene);
    m_glRightView->setCamera(m_mainFrame->rightCamera);

    // 3rd row:
    m_timer = new TimerWidget(this, wxID_ANY, t_mainFrame);
    m_startPanel->Add(m_timer, 1, wxCENTER);

    // Row 4
    m_emergencyStop = new EmergencyStop(this, wxID_ANY);
    m_emergencyStop->setData(m_mainFrame->getAddressEmergencyLoop());
    SelectAudioFile* selectAudioFile = new SelectAudioFile(
        this, wxID_ANY, m_mainFrame
    );

    // Layout to be changed later:
    m_gridPanels[0][0] = new wxBoxSizer(wxHORIZONTAL);  // Reserved for later.
    m_gridPanels[0][1] = new wxBoxSizer(wxHORIZONTAL);  // Reserved for later.
    m_gridPanels[0][2] = new wxBoxSizer(wxHORIZONTAL);  // Reserved for later.

    m_gridPanels[1][0] = m_glBackView;
    m_gridPanels[1][1] = m_sideViews;
    m_gridPanels[1][2] = m_glGraphView;

    m_gridPanels[2][0] = new wxBoxSizer(wxHORIZONTAL);  // Reserved for later.
    m_gridPanels[2][1] = new wxBoxSizer(wxHORIZONTAL);  // Reserved for later.
    m_gridPanels[2][2] = m_startPanel;

    m_gridPanels[3][0] = m_emergencyStop;
    m_gridPanels[3][1] = new wxBoxSizer(wxHORIZONTAL);  // Reserved for later.
    m_gridPanels[3][2] = selectAudioFile;

    // Adding all elements to the flexGridSizer
    m_fgs->Add((wxSizer*)m_gridPanels[0][0], 1, wxEXPAND | wxALL);
    m_fgs->Add((wxSizer*)m_gridPanels[0][1], 1, wxEXPAND);
    m_fgs->Add((wxSizer*)m_gridPanels[0][2], 0, wxEXPAND);

    m_fgs->Add(m_glBackView, 1, wxEXPAND | wxALL);
    m_fgs->Add(m_sideViews, 1, wxEXPAND | wxALL);
    m_fgs->Add(m_glGraphView, 1, wxEXPAND | wxALL);

    m_fgs->Add((wxSizer*)m_gridPanels[2][0], 1, wxEXPAND | wxALL);
    m_fgs->Add((wxSizer*)m_gridPanels[2][1], 1, wxEXPAND | wxALL);
    m_fgs->Add(m_startPanel, 1, wxEXPAND | wxALL);

    m_fgs->Add(m_emergencyStop, 1, wxEXPAND | wxALL);
    m_fgs->Add((wxSizer*)m_gridPanels[3][1], 0, wxEXPAND);
    m_fgs->Add(selectAudioFile, 1, wxEXPAND | wxALL);

    m_fgs->AddGrowableRow(1, 1);
    m_fgs->AddGrowableCol(0, 1);
    m_fgs->AddGrowableCol(1, 1);
    m_fgs->AddGrowableCol(2, 1);

    //Adding the grid sizer to the main box sizer for the specific panel
```

```
    m_hbox->Add(m_fgs, 1, wxALL | wxEXPAND | wxFIXED_MINSIZE, 15);
    this->SetSizerAndFit(m_hbox);
    Centre();
}
```

I andre rad blir grafikk av ambulansen lagt til, og blir satt opp med scenen og kameraer som er definert i `BaseFrame`. I tredje rad blir timeren satt opp, og i fjerde rad blir nødstopp-indikatoren og valg av lyd satt opp. Selve layoutet blir ikke laget før etter at rad 4 er definert. Her blir det også lagt til "wxBoxSizere", med andre ord tomrom som er reservert til bruk senere.

6.2.2 Egendefinerte Widgets

Prosjektet benytter seg av en rekke `CustomWidgets` og `CustomButtons` som er egendefinerte da de har spesielle egenskaper når de f.eks. klikkes eller skal tegne unike ikoner. I disse tilfellene er det laget egne klasser som arver i fra et eksisterende "Widget". Deretter implementeres egne tegnefunksjoner for hvordan en "Widget" skal se ut, og funksjoner som kalles ved eventer som f.eks. når en "Widget" klikkes og musepekeren beveger seg over det. En del av disse funksjonene er implementert av kosmetiske årsaker med hensyn til respons og brukervennlighet. For å fjerne flimring blir den forrige tegnede "Widgeten" ikke fjernet i det "Widgeten" tegnes på nytt. I stedet tegnes hele "Widgeten" over den forrige "Widgeten".

Eksempelet nedenfor viser metodene for den egendefinert "Widgeten" `EmergencyStop`. De andre egendefinerte "Widgetene" bruker akkurat samme metode.

```
BEGIN_EVENT_TABLE(EmergencyStop, wxPanel)

EVT_PAINT(EmergencyStop::paintEvent)
EVT_SIZE(EmergencyStop::onResize)
EVT_ERASE_BACKGROUND(EmergencyStop::eraseBG)

END_EVENT_TABLE()


EmergencyStop::EmergencyStop(wxPanel* t_parent, wxWindowID t_winID)
        : wxWindow(t_parent, t_winID, wxDefaultPosition, wxSize(100, 150))
{
        m_radius = 50;
}

EmergencyStop::~EmergencyStop()
{

}
void EmergencyStop::paintEvent(wxPaintEvent& t_paintEvent)
{
        wxBufferedPaintDC dc(this);
        render(dc);
}

void EmergencyStop::render(wxDC& dc)
{
        wxSize size = GetSize();
        wxString txt = "Nødstoppindikator";
        wxSize txtSize;
```

```
        wxFont font;

        dc.SetPen(g_colourBG);
        dc.SetBrush(g_colourBG);
        dc.DrawRectangle(0, 0, GetSize().x, GetSize().y);


        dc.SetPen(g_colourBorder);
        if (*m_active)
        {
                dc.SetBrush(g_mainGreen);
        }
        else
        {
                dc.SetBrush(g_mainRed);
        }

        dc.DrawCircle(size.x/2, m_radius, m_radius);

        font.SetPointSize(14);

        dc.SetFont(font);

        wxSize textSize = dc.GetTextExtent(txt);
        int txtStartX = textSize.GetWidth();
        int txtStartY = textSize.GetHeight();


        dc.DrawText(txt, size.x/2-txtStartX/2, m_radius*2+txtStartY/2);
}

void EmergencyStop::onResize(wxSizeEvent& sizeEvent)
{
        Refresh();
}

void EmergencyStop::eraseBG(wxEraseEvent& eraseEvent)
{
        // Empty function, but needed to initialize the erase event
}

void EmergencyStop::setData(bool* isActive)
{
        m_active = isActive;
}
```

6.2.3 Koblig til hydraulikken

Det å koble brukergrensesnittet til hydraulikken bød på flere problemstillinger. En av disse var å synkronisere "Widgetene" til å vise den samme informasjonen. For eksempel er det totalt ni instanser av "Widgets" som viser ambulansen fordelt på de tre sidene i applikasjonen, og alle disse skal vise den samme rotasjonen. En annen problemstilling var å hente denne informasjonen i fra hydraulikken på en tråd-trygg ("thread safe") måte.

For å synkronisere "Widgetene" har de flese egendefinerte "Widgets" delte ressurser i form av delte addresser. Dette betyr i praksis at de har pekere til den

addressen i minnet som blir benyttet, f.eks. addressen til en felles scene (en klasse i `AmbuGL` delprosjektet) som blir vist, eller en addresse til en felles bool. Denne dataen blir lagret i `BaseFrame`, da `BaseFrame` instansieres bare en gang. "Widgets" som trenger å aksesere denne dataen kan hente ut pekere til denne dataen i fra `BaseFrame`.

I kodeeksempelet over er variablen `m_active` en peker til en bool. Denne blir satt i `BasePanel` som kan sees i kodeeksempelet i avsnitt 6.2.1.

Et problem med denne implementasjonen er at private member data ikke lenger i praksis er private member dersom en slik peker til dataen kan hentes. Dersom addressen til en slik variabel kan hentes ut, kan den i praksis settes og leses fra hvor som helst etter at addressen er hentet ut, noe som gjør at poenget bak "getters" og "setters" funksjoner blir litt borte. Et av formålene til "setters" og "getters" er at de gjør det lettere å finne feil i koden. Hvis en variabel har en feil verdi kan man relativt lett sette et pausepunkt inni enten "setter" eller "getter" for å se når denne verdien blir satt feil. Dette gjør det lettere å lokalisere når i koden en verdi blir satt eller lest av feil.

På en annen side i dette tilfellet er disse variablene designet for å kunne bli delt. Dette er fordi det i utgangspunktet var meningen at f.eks. `EmergencyStop` "Widgeten" skal dele en bool, og denne må lagres et sted. Alternativt kunne disse variablene vært public (offentlig) data, da de er ment for å deles, eller vært globale variabler, da dette i praksis omtrent gir den samme effekten. Forskjellen hadde vært at ikke alle klassene som bruker denne dataen hadde trengt noen referanser til `BaseFrame` for å hente ut denne dataen. En bedre teknikk ville vært å ha brukt pekere til "getters" funksjoner i `BaseFrame` for å hente ut dataen. Da hindres dataen å bli overskrevet utenfra, noe som bevarer prinsippet bak å bruke "getters" og "setters" funksjoner til å begynne med.

En fordel med teknikken brukt er at man følger KISS (keep it simple, stupid), og holder det så enkelt som mulig, da alt som trengs er en peker til en addresse, noe som er intuitivt og enkelt å forstå. Når det er sagt, hadde det ikke vært et så mye større problem med å implementere funksjonspekere, men mer grundig dokumentasjon burde forkart hvorfor funksjonspekere i det hele tatt blir brukt for en vedlikeholder, som f.eks. ikke er så godt kjent med C++. I dette prosjektet virket det å bruke funksjonspekere litt unødvendig da det kunne gjøres enklere, men på bekostning av å bryte litt med private member prinsippene. Gruppen ble også ikke klar over hvordan funksjonspekere kunne brukes i C++ før sent i utviklingsfasen, det var dermed dårlig tid for implementasjon av denne løsningen. Dette vil føre til økt risiko for en dårlig implementert løsning.

6.3 Styring fra Mus

Det var ulike utfordinger med å styre ambulanen med mus. Skulle hydraulikken kunne nå alle mulige posisjoner? Det vil si, gitt en volt konfigurasjon $K = (V_{left}, V_{right})$ der $V_{left} \in [V_{min}, V_{max}]$ og $V_{right} \in [V_{min}, V_{max}]$, skulle det finnes en museposisjon $p_{mouse} = (x, y)$ der $x \in [-1, 1]$ og $y \in [0, 1]$ slik at konfigurasjonsfunksjonen $f(p)$ hadde en $f(p_{mouse}) = K$. Dette er bedre kjent som en bijektiv

funksjon. Samtidig skulle denne funksjonen være intuitiv å bruke. Tidligere var det brukt to "slidere" til å bestemme posisjonen til høyre sylinder og venstre sylinder. Problemet med dette var at det var ikke mulig å kontrollere begge sylindrene samtidig. Det var derfor ønskelig å lage en forbedret modell på dette.

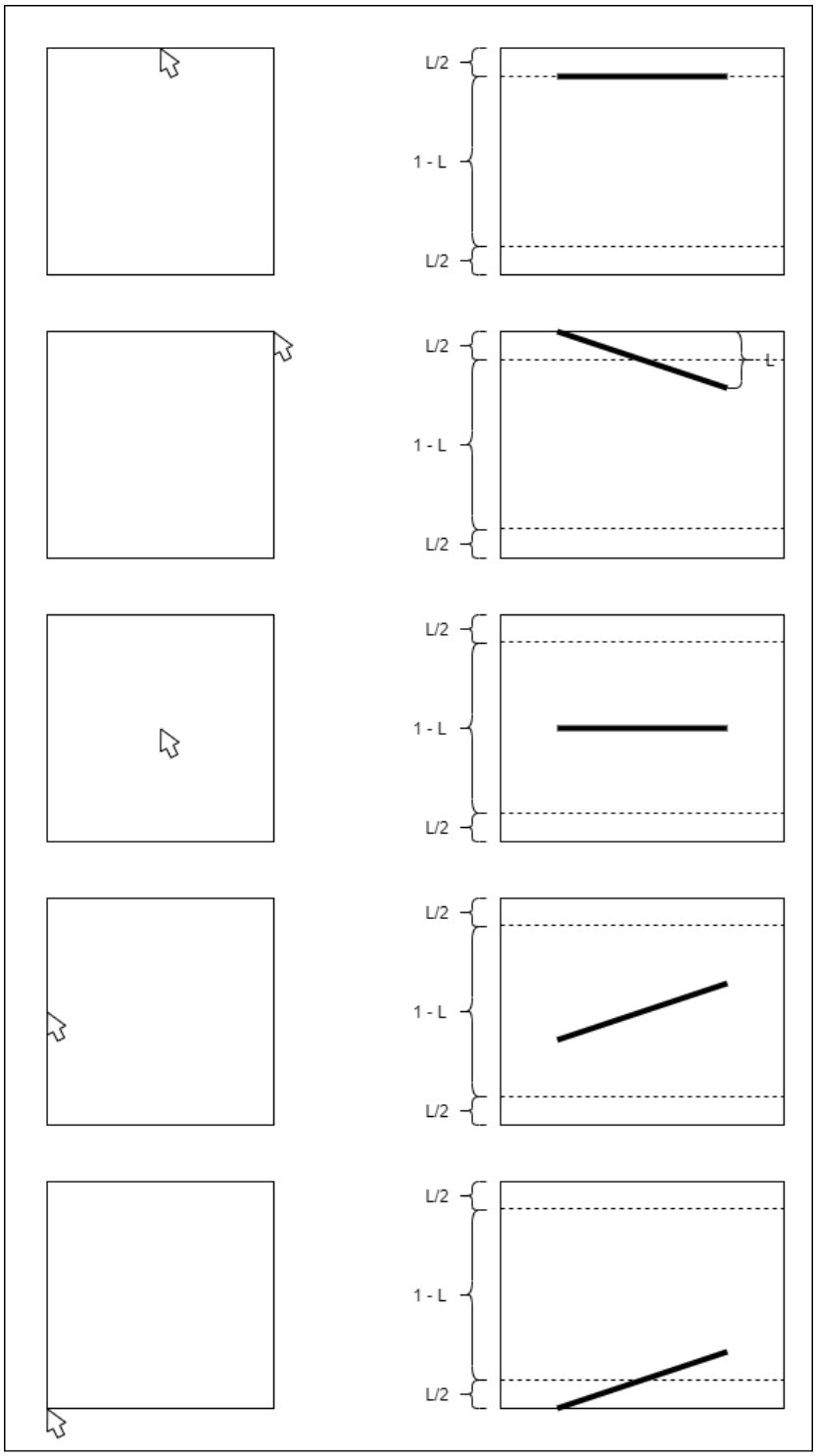
For å løse dette problemet ble det testet to prototyper av oppdragsgiver. Prototypen som kom seirende ut bruker y aksen som høyde, og x aksen for å rotere, eller lage høydeforskjell mellom sylindrene. Gitt museposisjonen $p_{mouse} = (x, y)$ der $x \in [-1, 1]$ og $y \in [0, 1]$, var venste sylinders volt $V_{left} = \frac{l}{2} + (1 - l)y + \frac{lx}{2}$ og tilsvarende $V_{right} = \frac{L}{2} + (1 - L)y - \frac{Lx}{2}$, der L er en gitt maks forskjell i avstand mellom venste og høyre sylinder, se også figur 6.2 for et visuelt eksempel.

Kodesnutten som implementerer dette, ser slik ut:

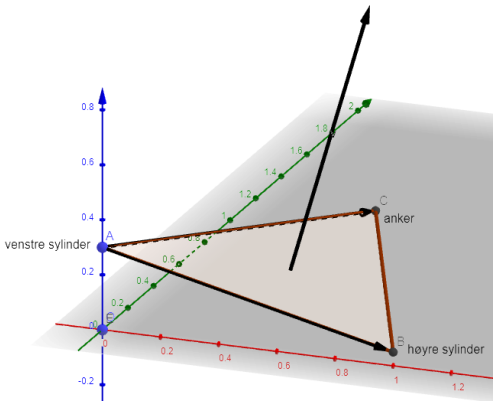
```
HydraulicConfiguration MouseController::getVolt()
{
        getData();

        /* This indicates the maximum distance from the y value(the height) on each
        cylinder. This will limit the maximum angle possible as you move the
        mouse horizontally. Higher value means the maximum angle will be steeper,
        0 will mean no horizontal movement at all and negative values will
        invert the movement */
        float d = 4.0f;

        // x will be between -1 and 1
        float x = m_normX * 2 - 1;

        // After this y will have a new range as follows:
        // ( MIN_VOLT + d  <=  y  <=  MAX_VOLT - d )
        float y = (MAX_VOLT - MIN_VOLT - 2 * d)
                * (1 - m_normY) + MIN_VOLT + d;

        // The height of the cylinder will be the y value adjusted by
        // the d value. The d value will be larger when you move to the
        // horizontal edges of the screen.
        m_hydrCfg.voltageLeft = y + d * x;
        m_hydrCfg.voltageRight = y - d * x;

        return m_hydrCfg;
}
```

`getData()` henter ut I/O data, i dette tilfellet er dette museposisjonen. Dermed brukes formelen over til å regne ut posisjonen til venstre og høyre sylinder.

Ulempen med denne metoden er at ikke alle mulige kombinasjoner kan nåes. Sylindrene kan altså ikke ha en høydeforskjell større enn det predefinerte L. På en annen side, virket det som om oppdragsgiver foretrakk at dette var tilfellet, i tillegg til metodens intuitive natur. En alternativ løsning kunne vært å ha en 3D ambulanse oveninfra i brukergrensesnittet, også rotert ambulansen etter som slik denne virtuelle ambulansen hadde blitt rotert. Dersom man hadde klikket og dratt til venstre, ville den rotert mot venstre, og motsatt mot høyre, og tilsvarende frem og tilbake. Dette designet hadde også vært intuitivt i tillegg til at det ville ha kunnet truffet alle punkter. Normalen til denne virtuelle ambulansen kunne blitt oversatt til volt med en teknikk som blir sett på i avsnitt 6.4. Utfordringen med denne løsningen er hvordan den skulle blitt begrenset, da det ikke er noen klare



Figur 6.2: Figur viser museposisjon og korresponderende possisjon til platen ambulansen står på.



Figur 6.3: Figuren viser en tilnærmet modell av planet ambulansen står på.

grenser. Simulasjonen risikerer dermed å bli hakkete.

6.4 Virtuelle krefter: Konvertere krefter til volt

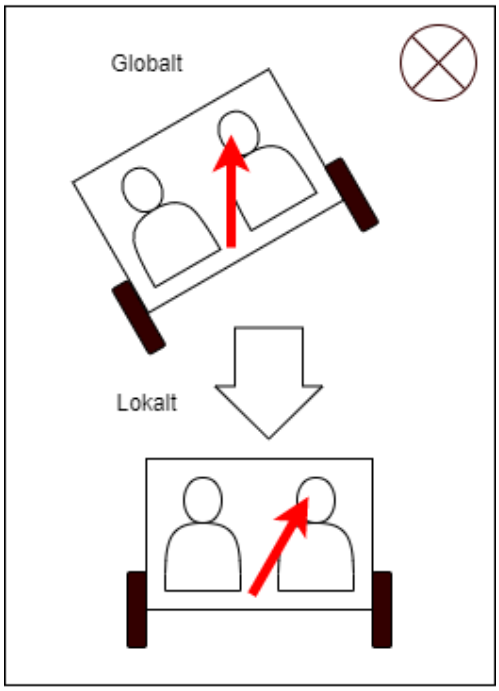
Det er flere tilfeller hvor det ville vært praktisk å kunne konvertere krefter til volt output for å styre hydraulikken. I et spill som ETS2 må man basere seg på hvilke krefter som påvirker bilen. Når man kjører fritt uten et spill, kjører man i praksis fortsatt en virtuell bil i bakgrunnen. Denne bilen blir påvirket av krefter som akselerasjon, bremsing og sentripentalkraft. I tillegg finnes det verktøy som physics toolbox. Dette er en applikasjon på mobiltelefonen som måler krefter, og skriver disse dataene til en fil. Ved å bruke denne applikasjonen i en kjørende bil, kunne man fått krefter en bil opplever under kjøring. Dette ga motivasjonen til å utvikle et verktøy som enkelt kunne oversette slike krefter til volt data.

Det ble laget en enkel modell basert på å rotere ambulansen slik at normalkraften fra setene i ambulansen ville peke i størst mulig grad i samme retning som den kraften man ville føle lokalt inne i bilen. Hvis man f.eks. akselererer forover, ville ambulansen rotere med snuten oppover slik at man kjenner et lite trykk i ryggen. Dersom man bremser, ville den tippet i motsatt retning.

Ambulansen står på et "plan". Dette planet roterer mer eller mindre for det meste rundt et anker som er foran bilen. Det er en sylinder på hver side som av ambulansen som er festet til dette planet. Disse sylindrene kan justere høyden sin. Med dette ble det laget en forenklet modell av det planet ambulansen står på, se figur 6.3. Normalen til dette planet har en retning lik kryssproduktet av differansen mellom posisjonene til de to sylinderene og differansen mellom posisjonen til ankeret og en av sylinderene.

$$N = (S_r - S_l) \times (A - S_l)$$

der S_r er posisjonen til synlinderen på høyre side, S_l er posisjonen til sylinderen på venstre side når ambulansen er sett bakfra, og A er posisjonen til ankeret foran. $S_l = (0, 0, h_l)$, og $S_r = (d_s, 0, h_r)$, der h_l og h_r er de to ukjente i ligningen som er høydene til venstre og høyre sylinder, og d_s er den målte avstanden mellom dem.



Figur 6.4: Figuren viser hvilken retning gravitasjonen, sett bakfra, peker "lokalt" inne i bilen.

Ankeret A vil ha sin posisjon lik $A = \left(\frac{d_s}{2}, d_a, h_a\right)$ der d_a er den målte gulv-avstanden fra midten av sylindrene til ankeret. Med dette kan en normal N bli tatt inn som input, og kan brukes til å løse det endelige ligningsettet:

$$N \times ((S_r - S_l) \times (A - S_l)) = 0$$

med hensyn på h_l og h_r. Kryssproduktet $(S_r - S_l) \times (A - S_l)$ skal være parallelt med normalen. Disse er parallelle dersom kryssproduktet mellom dem er lik 0. Siden punktene A, S_r og S_l er tredimensjonale punkter får vi tre ligninger, og kan løse ligning med de to ukjente. Selve utregningen var lang og har en del varaibler. Derfor ble "sympy", som er et bibliotek i Python, tatt i bruk for å løse utregningen.

Normalen N skal peke i en retning slik at normalkraften fra setet i ambulansen påført av gravitasjonen N_g peker i retningen av den kraften som føles lokalt inne i bilen. Når ambulansen er rotert, vil den lokale kraften kjent inne i bilen være rotert i motsatt retning, fordi den ikke roterer med ambulansen. Med andre ord, den vil roteres med den inverse rotasjonen (se figur 6.4).

Hvis ambulansens rotasjon er representert som en kvaternion Q er den lokale kraften F som kjennes inne i bilen $F = Q^{-1} \cdot N_g$ dersom F og N_g har enhetslengde. Normalen N til planet ambulansen står på er gitt ved $N = Q \cdot N_g$. Kvaternionen Q^{-1} blir i praksis konstruert først for å tilfredsstille $F = Q^{-1} \cdot N_g$. Den inverse av denne brukes for å regne ut normalen $N = Q \cdot N_g$. Denne normalen brukes i lignigene nevnt tidligere for å regne ut posisjonen til venstre og høyre sylinder.

Her er et kodeeksempel som viser hvordan normalen til planet ambulansen står på blir regnet ut:

```
mathz::Vector3 counterDirection = mathz::Quaternion::fromTo(
    mathz::Vector3(0, 0, 1),
    forcesDirection.normalized()
).inverse() * mathz::Vector3(0, 0, 1);
```

`counterDirection` er et annet navn på normalen.

Med den funnede normalen brukes følgende kodesnutt for å konvertere normalen til høydene til den venstre og høyre sylinderen:

```
// The right cylinder height.
float cylinderRight = (
    m_anchorHeight * counterDirection.z +
    m_distanceAnchorToCylindersLine * counterDirection.y -
    (m_distanceBetweenCylinders * counterDirection.x / 2)
) / counterDirection.z;

// The left cylinder height.
float cylinderLeft = cylinderRight +
(
    (m_distanceBetweenCylinders * counterDirection.x) /
counterDirection.z
);
```

Formelen ble laget ved å løse ligningene over i "sympy" som nevnt tidligere. Endelig resultat for h_l og h_r kan sees i følgende kodesnutt:

```
float voltLeft = (
    (cylinderLeft - m_anchorHeight) / m_voltToMeter
    ) + m_flatLevelVoltage;
float voltRight = (
    (cylinderRight - m_anchorHeight) / m_voltToMeter
    ) + m_flatLevelVoltage;
```

`m_flatLevelVoltage` er volt outputen der ambulansen står flatt, uten noe rotasjon. `m_voltToMeter` er en variabel regnet ut for å konvertere volt til meter, f.eks. en forskjell på 1 volt tilsvarer `m_voltToMeter` stor forskjell i meter.

En stor fordel ved denne metoden er at alle moduler som har krefter som output kan konvertere disse til volt. På en annen side tar ikke metoden høyde for de kreftene som kjennes i det ambulansen roterer fra en posisjon til en annen. En stor ulempe med dette er at de som sitter lenger bak i ambulansen vil kjenne sterkere krefter enn de som sitter foran. En alternativ modell kunne tatt høyde for den gjennomsnittlige kraften som kjennes i bilen, der den gjennomsnittlige kraften er tilnærmet lik den samme som kraften som kjennes når man sitter i midten av ambulansen. I tillegg fører modellen til at ambulansen kanskje ikke roterer slik en hadde ønsket. Når det svinges til høyre har ambulansen i den tidligere versjonen også rotert mot høyre, noe oppdragsgiver ønsket at den skulle fortsette å gjøre. Oppdragsgiver ønsket denne styringstypen da dette følger en dossert veibane, som gjerne vil lene seg mot venstre i en venstre sving. Problemet er at man da antar at alle svinger er dosserte, noe som ikke er realistisk. Dette er ikke tilfellet for denne modellen, da den vil rotere den andre veien for å simulere den kraften man ville ha kjent. Dette er på en annenside enkelt løst ved å konfigurere en instillning

for å reversere de kreftene man kjenner til sidene inne i programmet. En annen ulempe er det lille ambulansen faktisk kan rotere. Det ble målt at ambulansen maksimalt kan rotere 6 grader oppover, nedover og til sidene, noe som fører til at ambulansen når sitt maksutslag omtrent med en gang kreftene peker framover. Dette er løst ved å introdusere "virtuelle vinkler". Dette vil si at en gitt vinkel i den "virtuelle verden", f.eks. ETS2 vil tilsvare en fysisk vinkel. For eksempel kan en 30 graders helling oppover tilsvare maksutslaget på 6 grader.

6.5 Styring fra Spill og Euro Truck Simulator 2

Med en matematisk modell for å styre hydraulikken gitt en kraft (se avsnitt 6.4) kunne styringen av simulasjonen ut i fra ETS2 enklere blitt implementert.

ETS2 har en SDK[25] som gjør det mulig å hente ut data som akselerasjonen til bilen man kjører. SDK'en bruker delt minne for å kunne lese av data i fra spillet. "telemetry.h" er en fil som kom med ETS2 SDK-en. Denne filen inneholder en struct det delte minnet kan castes til for å hente ut dataene. Dette brukes for å hente ut akselerasjonen til bilen lokalt. Lokal akselerasjon vil si at positiv x-retning peker til høyre for sjåføren, positiv y peker opp mot taket og positiv z peker forover i henhold til koordinatsystemet brukt i ETS2. Rotasjonen av bilen blir også hentet ut for å legge til gravitasjonskraften, da denne ikke ble inkludert i akselerasjonen av bilen. Gravitasjonen måtte oversettes til akselerasjon slik at den endelige akselerasjonen kunne være $a = a_{truck} + a_{gravity}$ der a står for akselerasjon. a_{truck} hentes direkte, og $a_{gravity}$ regnes ut som $a_{gravity} = Q^{-1} \cdot (-g)$. Q er rotasjonskvaternionen til bilen, og g er gravitasjonen $g = (0, 0, -9.81)$. Det multipliseres med $-g$, fordi det er normalkraften, det at man akselererer mot gravitasjonen som kjennes. En annen måte å se det på, er som et helhetlig akselererende system, der bilen kjører oppå en stor disk som akselererer oppover, da dette i praksis kjennes helt likt ut som å bli akselerert nedover og bakken presser oppover. Det multipliseres med den inverse av rotasjonskvaternionen, fordi kraften ikke roteres med bilen.

For å koble til ETS2 brukes følgende kodesnutt:

```
bool ETS2DataQuery::connect()
{
    // Tries to open shared memory with given name:
    m_hMapFile = OpenFileMapping(
        FILE_MAP_READ,
        FALSE,
        ETS2_PLUGIN_MMF_NAME
        );
    if (!m_hMapFile)
    {
        // Connection failed if not m_hMapFile.
        connected = false;
        return false;
    }

    // If connection was successfull, get a pointer to the shared memory:
    m_sharedMemory = (LPSTR)MapViewOfFile(
```

[25] https://github.com/nlhans/ets2-sdk-plugin/releases/tag/revision_5_rel_1_4_0

```
        m_hMapFile,
        FILE_MAP_READ,
        0,
        0,
        16384
        );

    if (!m_sharedMemory)
    {
        // Connection failed if not m_sharedMemory.
        connected = false;
        return false;
    }

    // Fetches data from the shared memory.
    getData();

    // The connection was successfull:
    connected = true;
    return connected;
}
```

`m_sharedMemory` er peker til det delte minnet og `ETS2_PLUGIN_MMF_NAME` er "navnet" på det delte minnet. `getData()` bare kopierer minnet direkte over i en strukt.

Følgende kodesnutt er brukt for å oversette akselerasjon og rotasjon i ETS2 til volt data som kan brukes til hydraulikken:

```
HydraulicConfiguration ETS2DataQuery::getVolt()
{
    getData();

    mathz::Vector3 acceleration, virtualAcceleration;

    // Get the acceleration of the truck.
        /*
        * Through testing we know:
        *   the X axis is the acceleration to the right.
        *   the Y axis is the acceleration up and down.
        *   the Z axis is the acceleration forwards.
        */
    acceleration = mathz::Vector3(
        m_euroTruckData.tel_rev1.accelerationX,  // Left right.
        m_euroTruckData.tel_rev1.accelerationY,  // Up down.
        m_euroTruckData.tel_rev1.accelerationZ  // Forward -z
    );

    // Combine the forces with the diretion of the normal force from the truck.
    acceleration +=
        getRotation().inverse() *
        mathz::Vector3(0, 9.81, 0);

    // We need to convert to another coordinate system.
    virtualAcceleration = mathz::Vector3(
        acceleration.x,
        -acceleration.z,  // coordinate system was mirrored.
        acceleration.y
    );

    // Convert the final forces to voltage data:
    m_hydrCfg = m_vController->voltageOutput(virtualAcceleration);
```

```
    return m_hydrCfg;
}
```

6.6 Fri styring og virituell bil

Teknikken som er beskrevet i avsnitt 6.5 er også brukt i fri kjøring med ratt.

I dette tilfellet er det laget en "virtuell bil", som kjøres i bakgrunnen. Her brukes man mer matematisk modell heller enn en fysisk nøyaktig modell, da denne bilen er usynlig og det er vanskelig å si noe om presisjonen. Lengden på bilen l og rotasjonsposisjonen til hjulene θ brukes for å regne ut radiusen på sirkelen bilen kjører i. Om $\theta = 0$, kjører bilen i en rett linje. Formelen for denne radiusen r er $r = \frac{l}{\sin\theta}$. Sentripental-askelerasjonen a_x er $a_x = \frac{v^2}{r}$. Dette gir at $a_x = \frac{v^2 \sin\theta}{l}$. Modellen for akselerasjon bygger på en forenklet modell at bilen får tilført konstant effekt. Dette vil si at arbeidet W er:

$$W_{t+1} = \frac{1}{2} m {v_{t+1}}^2 = \frac{1}{2} m {v_t}^2 + P\Delta t$$

t angir tidssteg, v er hastigheten til bilen, m er massen til bilen, P er effekten i watt og Δt er endring i tid. Dette gir:

$$v_{t+1} = \sqrt{{v_t}^2 + \frac{2P\Delta t}{m}}$$

Akselerasjonen er gitt ved

$$a = \frac{v_{t+1} - v_t}{\Delta t}$$

Som kan skrives om til

$$a = \frac{\sqrt{{v_t}^2 + \frac{2P\Delta t}{m}} - v_t}{\Delta t}$$

Denne likningen er implementert i koden for å regne ut akselerasjonen forover til den virituelle bilen.

Følgende kode implementerer matematikken nevnt over. Funksjonen returnerer en hydralikk konfigurasjon med volt til venstre og høyre sylinder.

```
HydraulicConfiguration VirtualCar::timeStep(
    float t_steeringWheelPosition,
    float t_accelPosition,
    float t_brakePosition,
    float t_deltaTime
)
{
    // Multiplies by 746 to convert the horse powers to watt.
    // We add deltaTime to avoid dividing by 0. The lower the delta time, the
    // smaller the number we need to add.
    // The formula for acceleration given power, mass and speed is power/mass*speed.
    // The power is the power times the accelerationPosition which is a value
    // between 0 and 1.
    float acceleration = (
        sqrtf(
```

```
        2 *
        t_accelPosition * m_horsePowers * mathz::HORSE_POWERS_TO_WATT *
        t_deltaTime / m_mass
        + m_speed * m_speed) - m_speed
        ) / t_deltaTime;

    float airDrag = m_airResistanceConstant * m_speed * m_speed;
    float brake = m_brakeFriction * 9.81 * t_brakePosition;
    float nettoAcceleration = acceleration - airDrag - brake;
    float oldSpeed = m_speed;
    m_speed = max(0, m_speed + nettoAcceleration * t_deltaTime);
    float accelY = (m_speed - oldSpeed) / t_deltaTime;
    float angle = t_steeringWheelPosition * m_wheelMaxAngle;
    /// Acceleration innwards, aka. centripental acceleration:
    float accelX = m_speed * m_speed * sinf(
        angle * mathz::PI / 180
        ) / m_carLength;
    return m_forceController.voltageOutput(
        mathz::Vector3(accelX, accelY, 9.81f)
    );
}
```

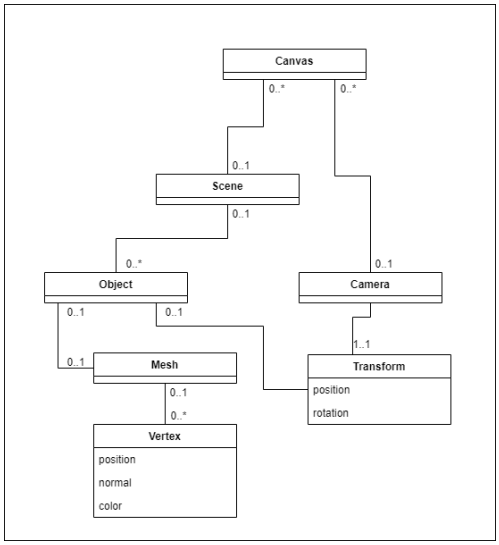
Det er i tillegg brukt en enkel modell for luftmotstand. Denne er bare en konstant ganget med hastigheten kvadrert. Modellen for bremsenkraften er normalkraften fra bakken ganget med en konstant μ som i dette tilfellet er kalt `m_brakeFriction`, ganget med brems som er et tall fra 0 til 1. `accelX` er sentripentalkraften.

En alternativ metode kunne vært å ha utviklet en mer fysisk realistisk simulasjon som tar hensyn til andre fysiske egenskaper med bilen, som kraftkurven til forskjellige gir og dreiemoment. Fordelene med en slik metode er at en kan putte inn spesifikasjonene på en slik bil som f.eks. vekt, hestekrefter, RPM og lignende, også ville modellen simulert denne typen bil. Ulempene med en slik modell er at den fort blir mye større og mer komplisert, noe som gjør koden vanskeligere å vedlikeholde. I tillegg vil det være noe unødvendig for hva som prøves å oppnås, da det er vanskelig å merke forskjell på disse to metodene grunnet at bilen er usynlig for den som kjører. En enklere modell, og sterkere kandidat hadde vært å bruke regressjon til å tilpasse f.eks. en kurve på formen $a \cdot ln(1 + bt)$ til en målt hastighetskurve over tid til en bil med automatgir. Dette kan brukes for å regne ut akselerasjonen ved å ta den deriverte av hastigheten. Med dette kan $t(v)$ regnes ut gitt $v(t)$, som kan brukes for å regne ut $a(v)$. Grunnen til at denne ikke ble valgt, var at modellen som blir brukt kan ta inn kjente parametere som input, som f.eks. hestekrefter og masse for tilpasse modellen, noe som virket lettere å modifisere.

6.7 AmbuGL: OpenGL visualisering

Det var ønskelig å både kunne generere en sekvens uten å måtte være koblet til ambulansen, samt kunne se omtrent hvordan ambulansen beveget seg på skjermen fra kontrollrommet.

For å løse dette, er OpenGL brukt sammen med wxWidgets for å visualisere bevegelser til ambulansen med en 3D modell som blir vist i brukergrensesnittet. Det er laget en minimalistisk 3D-renderer med OpenGL-bibloteket glew, hvor hele



Figur 6.5: Objektorientert design av AmbuGL

hensikten var å rendre en ambulanse. I et forsøk på å lage et forståelig og brukervennlig 3D-renderer ble OpenGL funksjonalitet abstraktert bort i klasser (se også figur 6.5 for alle klassene brukt). En `Vertex` inneholder data til et punkt som er en byggekloss i en `Mesh`, eller et 3D objekt. Når en `Mesh` er opprettet kan det lages et objekt som har et `Mesh` og en `Transform`. En `Transform` inneholder posisjonsdata og rotasjonsdata til objektet. Deretter kan en `Scene` bli opprettet, og et slikt `Object` kan bli lagt til i denne scenen. For å vise scenen i wxWidgets opprettes en `Canvas`, som er i seg selv en wxWidget. En slik `Canvas` har en peker til en `Scene` som skal bli sett. I tillegg har den en peker til et `Camera` som også må opprettes, som en `Scene` blir sett gjennom.

Koden for `AmbuGL` kan finnes i `AmbuGL` prosjektet, hovedsakelig i `AmbuGL.cpp` og `AmbuGL.h`.

6.8 CoreToolz

I prosjektet var det behov for blant annet en del matematikk og andre verktøy som klasser trengte. `CoreToolz` er en samling av slike felles verktøy, som klasser for å håndtere vanlig matematikk. Z-en står ikke for noe spesielt, men er heller ment for å gjøre et forsøk på å skille namespacet fra andre ting som kan ligne. I underprosjektet finnes blant annet `mathz`, et namespace for å håndtere matematikk med både 2D og 3D vektorer, `Matrix4` for å håndtere en transform og `Quaternion` for å håndtere rotasjon.

`CoreToolz` kan sees i `toolz.cpp` og `toolz.h` for flere metoder. Følgende kodesnutt viser `Quaternion::fromTo()` metoden fra `CoreToolz` som et eksempel i fra `CoreToolz`:

```
Quaternion Quaternion::fromTo(Vector3 t_from, Vector3 t_to)
{
    Vector3 unitFrom = t_from.normalized();
```

```
    Vector3 unitTo = t_to.normalized();
    if ((unitFrom - unitTo).squareMagnitude() < 0.00001f)
    {
        // Return Identity quaternion.
        return Quaternion(1, 0, 0, 0);
    }
    Vector3 crossProduct = unitFrom.cross(unitTo);
    crossProduct = crossProduct.normalized();

    Quaternion tmp;
    float halfTheta = acosf(unitFrom * unitTo) / 2;
    tmp.w = cos(halfTheta);
    tmp.i = sin(halfTheta) * crossProduct.x;
    tmp.j = sin(halfTheta) * crossProduct.y;
    tmp.k = sin(halfTheta) * crossProduct.z;

    return tmp;
}
```

De første linjene bare sørger for at vektorene er enhetsvektorer og sjekker at de ikke er like, eller peker i samme retning. Deretter initialiseres kvaternionen som vanlig med rotering rundt kryssproduktet av disse vektorene med vinkelen mellom dem.

6.9 Les Sekvens: .sequence filer

En .sequence fil er et egendefinert format med to kolonner. Formatet ser slik ut:

linje nummer	Høyre kolonne	venstre kolonne
01	time_t	<antall sekunder siden 1. Januar 1970 klokken 00:00>
02		
03	Date	<dato på form yyyy/mm/dd>
04	Time	<tidspunkt på form hh:mm:ss hvor h er mellom 0 og 23>
05		
06	Refresh rate	<antall herz> Hz
06		
07	Left	Right
08 - *	<left cylinder volt>	<right cylinder volt>

Et eksempel på dette er:

```
time_t          1620049594

Date            2021/05/03
Time            15:46:34

Refresh rate    10 Hz

left            right
-2.404203       -3.250477
-2.404203       -3.250477
-2.404203       -3.250477
-2.394104       -3.248715
-2.378071       -3.241021
```

```
-2.859220       -3.080168
-3.826433       -0.036959
        .           .
        .           .
        .           .
-2.349701       -3.162625
-2.349058       -3.186995
-2.363802       -3.243427
```

Siden en slik sekvensfil inneholder allerede volt dataen. Denne kan leses direkte inn i en Sequence klasse.

Slik ser kodesnutten som leser en sekvensfil ut:

```
void Sequence::readFromFile(const char* t_path)
{
        std::ifstream inFile;
        inFile.open(t_path);

        if (inFile)
        {
                const int BUFFER_MAX = 100;
                char buffer[BUFFER_MAX];
                time_t t;

                inFile >> buffer >> t; // Fetch creation time in time_t

                // Convert time_t to our m_creationTime tm struct
                localtime_s(&m_creationTime, &t);

                // Loop until refresh rate is hit
                while (strcmp(buffer, "rate"))
                {
                        inFile >> buffer;
                }
                int hz;
                inFile >> hz;
                // Convert hz to ms
                m_timeStep = 1000 / hz;

                // Loop past the header
                while (strcmp(buffer, "right"))
                {
                        inFile >> buffer;
                }

                // Loop through the rest of the file
                double leftVolt, rightVolt;
                while (!inFile.eof())
                {
                        inFile >> leftVolt >> rightVolt;
                        m_left.push_back(leftVolt);
                        m_right.push_back(rightVolt);
                }

                inFile.close();
        }
        else
        {
                throw "ERROR: Fil ikke funnet";
        }
}
```

6.10 Les Sekvens: Physics Toolbox fil / .csv fil

Med appen Physics Toolbox kan en annen type sekvensfil genereres. Denne filen har et .csv format med kolonnene tid, g-kraft i x-retning, g-kraft i y-retning, g-kraft i z-retning og størrelse på g-kraften. Et eksempel på dette er i følgende snutt:

```
time,gFx,gFy,gFz,TgF
0.004228646,0.0095,1.0140,0.1398,1.024
0.004555208,0.0119,1.0133,0.1477,1.024
0.004705937,0.0134,1.0153,0.1538,1.027
                .
                .
                .
0.004846666,0.0088,1.0104,0.1628,1.023
0.004990364,0.0093,1.0106,0.1660,1.024
```

Innlesningen av denne typen sekvens var fortsatt i prototype-stadiet etter fullført prosjekt og er eksperimentell. Dette skyldes at det var noe ekstra tid på slutten til å utvikle, men ikke nok tid til å fullføre den.

En Physics Toolbox sekvensfil har ikke et fiksert tidsintervall, slik som er implementert i dette prosjektet. Når denne filen leses, hoppes det derfor over linjer der forskjell i tid i fra forrige innleste line er mindre enn 0.1 sekunder. Disse kreftene leses inn i 3-dimensjonal vektor. Etter noen utførte eksperimenter, ble det funnet ut at y og z aksen er byttet om på, eller at koordinatsystemet brukt er speilvent fra det `VirtualForceController` bruker. Filens y-verdi leses derfor inn i vektorens z-verdi og z-verdien til filen inn i y. Det blir gjort et forsøk på å kalibrere sekvensen, ved å finne ut av i hvilken rotasjon systemet der appen kjører fra har. Det antas derfor at den første innleste vektoren vil peke i retningen av gravitasjonen. Den første vektoren brukes til å opprette en rotasjonskvaternion ved å bruke `Quaternion::fromTo()` metoden, og deretter ta den inverse av denne. Denne inverse multipliseres med alle innleste vektorer deretter, i et forsøk på å standardisere dem, slik at x-aksen peker mot høyre, y peker fremover, og z peker oppover. Deretter brukes `VirtualForceController` til å konvertere disse kreftene til volt data.

Etter at volt data er blitt lest inn, er det dukket opp en god mengde med støy. Dette skyldes sannsynligvis unøyaktigheter og støy i måleinstrumentet brukt, kombinert med at instrumentet brukt har forskjellige orienteringer som blir forsøkt rettet opp, i tillegg til en konverteringsprosess til volt. Sannsynligvis er dette grunnene til at resultatet har en del støy. Dette er løst med å å glatte ut kurven med bruk av numerisk løsning for varmeligningen $u_{t+1}^n = u_t^n + \alpha\left(u_t^{n-1} - u_t^n + u_t^{n+1}\right)$ [3].

Følgende kodesnutt leser inn en Physic Toolbox sekvens:

```
void Sequence::readFromCsv(const char* t_path)
{
    std::ifstream inFile;
    inFile.open(t_path);

    VirtualForceController vCnt;
    HydraulicConfiguration hydrCfg;
```

```
    mathz::Vector3 force;
    mathz::Quaternion orientation;
    double t0 = 0, t1 = 0, temp;

    const int BUFFER_MAX = 100;
    char buffer[BUFFER_MAX];

    if (inFile)
    {
        inFile.getline(buffer, BUFFER_MAX);

        inFile >> t0;
        inFile.ignore();

        inFile >> force.x;
        inFile.ignore();
        inFile >> force.z;
        inFile.ignore();
        inFile >> force.y;
        inFile.ignore();
        inFile >> temp;

        orientation = mathz::Quaternion::fromTo(
            mathz::Vector3(0, 0, 1), force
            ).inverse();

        while (!inFile.eof())
        {
            inFile >> t0;
            inFile.ignore();

            inFile >> force.x;
            inFile.ignore();
            inFile >> force.z;
            inFile.ignore();
            inFile >> force.y;
            inFile.ignore();
            inFile >> temp;
            force = orientation * force;

            hydrCfg = vCnt.voltageOutput(force);

            m_left.push_back(hydrCfg.voltageLeft);
            m_right.push_back(hydrCfg.voltageRight);

            while (t1 - t0 < 0.1 && !inFile.eof())
            {
                inFile >> t1;
                inFile.getline(buffer, BUFFER_MAX);
            }
        }

        int itr = 5;
        double alpha = 0.1;

        for (int i = 0; i < itr; i++)
        {
            for (int j = 1; j < m_left.size() - 1; j++)
            {
                m_left[j] += (
```

```
                    m_left[j - 1]
                    - 2 * m_left[j]
                    + m_left[j + 1]
                    ) * alpha;

                m_right[j] += (
                    m_right[j - 1]
                    - 2 * m_right[j]
                    + m_right[j + 1]
                    ) * alpha;
            }
        }
    }
}
```

En alternativ metode i stedet for å hoppe over data hadde vært å lese inn alle datapunktene i intervaller av 0.1 sekunder og gjort et gjennomsnitt av dem. Dette kunne redusert støy i måleinstrumentet. Et problem kalibreringen har er at startvektoren ikke har en entydig rotasjon. Dette vil si at det finnes uendelig mange rotasjoner, som ligger rundt aksen i samme retning som gravitasjonen. Dette kan potensielt føre til rar oppførsel. Trolig kan også noe av støyen være forårsaket av dette. Et annet problem kalibreringen har er at den er avhengig av at instrumentet brukt for å måle ligger i ro, da det ikke kontinuerlig kalibreres. På en annen side er en uansett avhengig av at instrumentet ligger i ro for å måle riktige retninger av kreftene.

7 Kvalitetssikring

Arbeidsgiver uttrykte et ønske for å ha en programvare som kunne brukes og vedlikeholdes lenge etter prosjektgruppen var ferdig med utvikling. Det var derfor viktig å inkludere en del prosedyrer for kvalitetssikring av systemet.

7.1 Kodestandard

Motivasjonen bak kodestandarder var å ha en mer lettleselig kode. Generelt er ikke hvilken kodestandard det viktigste, heller er motivasjonen at alle utviklere holder samme standard. Prosjektgruppen fant fram til "lefticus C++ best practices"[26] og valgte å bruke disse standardene.

Prosjektgruppen ønsket også at en linter ble brukt i sammenheng med denne kodestandarden, men inkludering av forskjellige lintere viste seg å være svært vanskelig for gruppen. Problematikk rundt linter kom rett etter en uke hadde blitt brukt på oppsett av utviklermiljø med fungerende tester og grafisk brukergrensesnitt i samme kodebase. Videre leting etter en linter ble derfor utsatt på ubestemt tid, til gruppen mente de hadde tid til å se nærmere på forskjellige løsninger for lintere. Siden prosjektet manglet en spesifikk linter ble kontroll av kodestandard også inkludert i koderevidering (se avsnitt 7.2).

7.2 Code Review

Koderevidering ble gjennomført med to formål: sikre at alle utviklere hadde kontroll over hele prosjektet og forbedre kvaliteten på koden som ble produsert. Koderevidering ble gjennomført ved de fleste Pull-Requests til dev eller master, men ble ikke gjennomført på samme kode flere ganger. Koderevidering kom med forslag til forbedring i: navn av variabler, kodesnutter som kunne bruke mer forklaring i form av "inline" kommentarer, rettelse av språkfeil i eksisterende kommentarer, oppdagelse av minnelekkasjer eller bugs spesifikk til en gruppemedlems datamaskin.

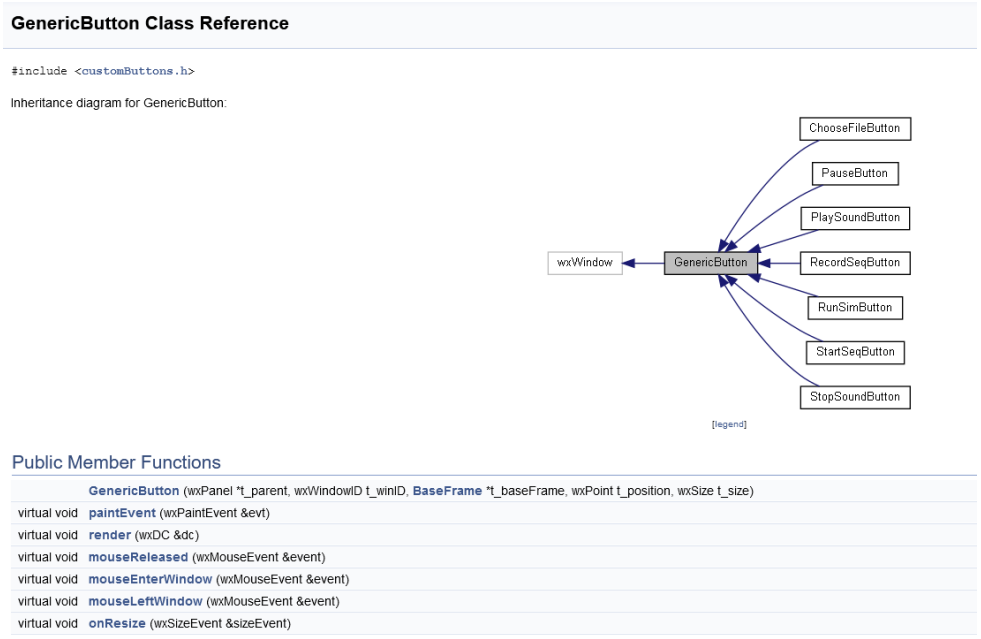
7.3 Testing

For å sikre nåværende og framtidig holdbarhet i systemet ønsket prosjektgruppen å ha en rekke automatiserte tester. Formålet med testene var å automatisk oppdage hvis en forandring i systemet introduserer problemer i resten av systemet. Testing av systemet ble gjennomført med Google Test adapter for Visual Studio[27].

Det ble også lagd et lite ark med akseptansetester, som kan sikre at essensiell funksjonalitet til systemet fungerer etter at kodebasen har blitt endret. Disse testene inkluderte et navn og en beskrivelse til hvordan å gjennomføre testen. Etter

[26] https://github.com/lefticus/cppbestpractices/blob/master/03-Style.md

[27] https://marketplace.visualstudio.com/items?itemName=ChristianSoltenborn.GoogleTestAdapter



Figur 7.1: Klasse referanse til en klasse ("Collaboration diagram" slettet for bedre overblikk)

testen var gjennomført kunne testeren markere resultatet av testen ("alt ok", "feilet"). Tester kunne også legge til egne kommentarer hvis ønskelig i siste feltet i malen. Malen til akseptansetestene kan bli funnet i vedlegg H.

7.4 Dokumentasjon

Hver metode, klasse og funksjon fikk en kort beskrivelse i sin header fil som forklarer hva som er formålet med komponenten. Disse kommentarene ble så sendt igjennom "doxygen"[28], for å automatisk generere en samlet dokumentasjon for hele prosjektet. "Graphviz"[29] ble også brukt med doxygen for å generere avhengighets og arv grafer til dokumentasjonen. Et eksempel på hvordan dokumentasjon ser ut kan bli sett i figur 7.1.

7.5 Risikoanalyse

Ved starten av prosjektet ble det gjennomført en risikoanalyse for en rekke risikoer prosjektgruppen identifiserte. Disse risikoene, og tilhørige tiltak, kan bli sett vedlegg D. Gruppen gjennomførte kun en risikoanalyse for prosjektet gjennom utviklingsperioden. Diskusjon rundt risikoanalyse kan bli sett i avsnitt 9.2.

[28] https://github.com/doxygen/doxygen
[29] https://graphviz.org/

8 Installasjon

For å bygge prosjektet fra source trenger man en del ekstra biblioteker. Dette er biblioteker for styring av hydraulikken, kommunikasjon med Logitech rattet og biblioteker for brukergrensesnitt. Restart må gjøres minst en gang i løpet av installasjonsfasen før man kan bygge prosjektet.

Prosjektgruppen valgte etter problemer med relative paths å lagre alle nødvendige libraries på statiske lokasjoner. Det er derfor viktig at eksterne biblioteker ligger under "`C:\SDK\`". Det vil også være spesifikke paths eksterne biblioteker skal være lagret under. Følg nøye med på installasjonsinstruksjoner for å sikre at alt blir lagret på riktig plass. Endelig filstruktur for eksterne biblioteker kan sees i figur 8.1.

8.1 NI-DAQmx

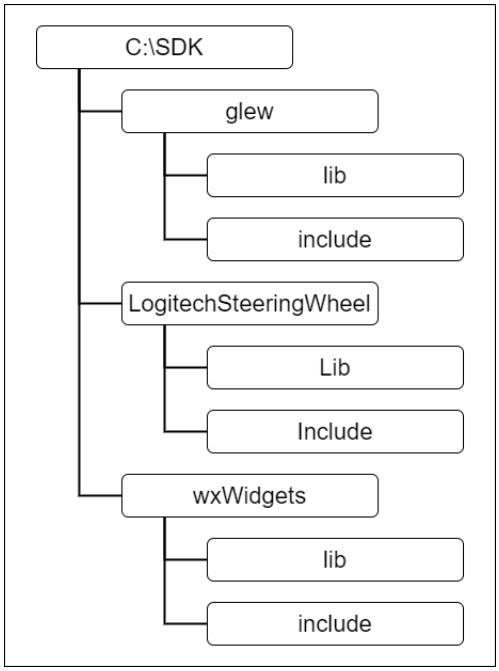
Hydraulikken og nødstoppsløyfen kommuniserer med programvaren vha. maskinvare fra National Instruments. Det er en cDAQ chassis med ulike moduler for både digital og analog input/output. NI-DAQmx er en NI-instrumentdriver som styrer alle aspekter av DAQ-systemet, fra konfigurasjon til programmering. 64-bits versjon kan lastes ned fra `https://www.ni.com/en-no/support/downloads/drivers/download.ni-daqmx.html`. Svar nei til automatiske oppdateringer etter installasjon av NI-DAQmx er ferdig.

8.2 wxWidgets

wxWidgets er et bibliotek til C++ som brukes for å lage applikasjoner. Det bruker plattformen (Windows i dette tilfellet) sitt innebygde API istedenfor eget brukergrensesnittdesign. Dette er nødvendig slik at programvaren blir brukervennlig. Last ned wxWidgets kildekoden som Windows ZIP fra `https://www.wxwidgets.org/`. Pakk ut mappen og legg den i "`C:\SDK\`". Kall denne mappen "wxWidgets" Åpne siste versjon av "`wx_vcXX.sln`" hvor XX står for versjonen av Visual Studio. Filen ligger under: "`C:\SDK\wxWidgets\build\msw`". Etter solution er åpnet i Visual Studio velger man *build* → *batch build*. I batch build velger man select all of build. Hvis det er build errors for wxWidget burde en eller to ekstra batch builds fikse problemer.

8.3 Logitech G920

I ambulansen er det installert et Logitech G920 spillratt. For å kunne hente data fra dette rattet må Logitech GHUB installeres og være åpent i bakgrunnen (kan lastes ned fra `https://www.logitechg.com/en-us/innovation/g-hub.html`). I tillegg må Logitech Steering Wheel SDK lastes ned fra `https://www.logitechg.com/en-us/innovation/developer-lab.html`. LogitechSteeringWheelSDK_XXXXXX pakkes ut til mappen
"`C:\SDK\`". Her står "_XXXXXX" for versjonsnummer. Gi denne mappen navnet "LogitechSteeringWheel".



Figur 8.1: Tre struktur etter ferdig instalasjon

8.4 GLEW

OpenGL Extension Wrangler Library er et bibliotek som hjelper med å laste inn OpenGL-utvidelser. Dette brukes for å vise en miniatyrmodell av ambulansen under simulering.

Gå til `http://glew.sourceforge.net/` for å laste ned GLEW. Klikk på windows 32 og 64 bit binaries, og last ned binaries derfra. Pakk ut denne filen (hvor som helst, f.eks. i nedlastninger). Åpne mappen og kopier mappen kalt "glew-XXXXXX" over i "`C:\SDK\`", der "XXXXXX" står for versjonsnummeret. Gi denne mappen navnet "glew".

8.5 Euro Truck Simulator 2

Programvaren kommuniserer med spillet ved hjelp av Euro Truck Simulator 2 Telemetry. Dette er en plugin som gjør det mulig å frakte data fra spillet til tredjepartsapplikasjoner.

Last ned release 1.4.0. Åpne opp fillokasjon for EuroTruck Simulator 2 under "`Steam\steamapps\common\Euro Truck Simulator 2\bin`" Alt som kommer før steam kommer an på nøyaktig hvor steam er installert. under "`bin\win_x86`" lag en ny mappe kalt plugins og kopier over "ets2-telemetry.dll" fra "`release_v1_4_0\Win32`" mappen. under "`bin\win_x64`" lag en ny mappe kalt plugins og kopier over "ets2-telemetry.dll" fra "`release_v1_4_0\Win64`" mappen og kjør spillet. Det skal dukke opp en tekst på start menyen som starter slik: "Request to use advanced SDK features detected..." Dette vil kun dukke opp hvis man

har en eksisterende profil.

8.6 Repository

Klon repository til ambulansesimulatoren fra `https://bitbucket.org/AslakTon/bachelor2021/src/master/`. Dette repositoriet kan ligge på vilkårlig sted på datamaskinen. Åpne "AbulanseSimulator.sln" og bygg.

8.7 Vedlikehold og utvikling

Systemet inneholder tre prosjekter for testing av programvaren. Kjør igjennom test prosjektet før eventuelle oppdateringer til programvaren pushes til git-repository. Legges ny funksjonalitet til, oppfordres det også til å legge inn egne tester i passende test-prosjekt.

Vær oppmerksom på at rattet ikke vil fungere hvis det ikke eksisterer et applikasjonsvindu. En kommandolinjevindu er ikke et gyldig vindu og vil ikke tillate rattet å oppdatere data tilhørende rattet (slik som ratt og pedalposisjon). Rattet vil kunne koble seg til og alle tester vil fungere mot et kommandolinjevindu, så ingen feilmeldinger vil komme.

9 Diskusjon

Hvert gruppeprosjekt som gjennomføres er en lærerik prosess. I denne prosessen læres det om hva som fungerte bra og hva som kunne fungert bedre. Ved å reflektere og diskutere dette kan man enklere ta med seg disse erfaringene videre for å gjøre forbedringer i fremtiden. I denne seksjonen blir det diskutert hvordan prosjektet har gått for seg, argumentasjon for de valgene som er blitt tatt og hvilke valg som ble tatt som kunne ha vært gjort annerledes.

9.1 Kvalitetssikring

I starten av prosjektet var det klart at kvalitetssikring skulle nøye gjennomføres for å få et mest mulig vedlikeholdbart prosjekt. For å nevne noen av punktene skulle tester lages for å automatisk oppdage feil. Koderevidering skulle gjennomføres for å avdekke feil, mulige bugs i fremtiden og for å sikre koden i å følge utvalgte standarder. Dette har fungert bra, men ikke uten feil og problemer i prosedyren.

I starten var det kun master branchen i "git" som var der sluttproduktet skulle samles. Det var meningen at all kode som ble samlet her skulle følge standarder og være godt leselig. Problemet med dette var at det ble fort oppdaget at det å være helt konsistent og å følge alle disse standarden til enhver tid var ganske vanskelig. Spesielt siden disse standardene var nye for flere i gruppen. Dette medførte at det ble opprettet en "dev" branch i "git", hvor reglene var mindre strenge. Fordelen med dette var at hastigheten på arbeidet gikk opp. Ulempen med dette var at standarden ble gradvis mindre fulgt, noe som desverre kan sees i sluttproduktet. En vanlig feil er f.eks. at privat member variabler ikke har prefixen `m_` eller at en parameter ikke har prefixen `t_`. Det er også kommentert mindre i koden enn det som i utgangspunktet var meningen. På en annen side kan dette for noen virke mer leselig, da volumet på koden går ned.

Et annet problem som dukket opp underveis var hvor mye tid som ble brukt på kvalitetssikring. For at programmet skulle være vedlikeholdbart, var det f.eks. ideelt å ha gode feilhåndteringsmekanismer og enhetstester. Disse feilhåndteringsmekanismene og enhetstestene finnes, men ikke i så stor grad som var meningen i utgangspunktet. Hovedårsaken til dette var at det var tidkrevende, og det ble estimert at det å gjennomføre nøye feilhåndtering og enhetstesting ville medføre mangel på funksjonalitet ved innleveringsdato. Dette medførte at prosjektgruppen måtte velge mellom nøye feilhåndtering og enhetstesting eller funksjonalitet i programmet. Et kompromiss ble valgt hvor feilhåndtering og enhetstesting skulle gjennomføres, men i mindre grad. Oppdragsgiver skulle også få et ferdig funksjonelt program som oppfyller systemkravene, men med noen forenklede løsninger, som f.eks. at brukergrensesnittet ikke følger UX designet helt nøyaktig.

I fremtidige prosjekter kan dette forbedres ved at kvalitetssikringsprosessen blir mer nøyaktig planlagt. Dette inkluderer å beregne mer tid for kvalitetssikringen ved tidsestimeringen.

9.2 Risikoanalyse

Flere risikoer var gjort høyde for i planleggingsfasen av prosjektet, men det var flere andre problemer som dukket opp underveis som ikke ble tatt høyde for. Blant disse var det vanligste tekniske problemer på personlig datamaskin. Medlemmet hadde commits lokalt som ble borte i det denne maskinen ble korrupt. Risikoen for tap av kode var tatt høyde for, men grunnet for sjeldne commits og push til "remote" ble denne koden tapt.

Koronarestriksjonene førte til at et gruppemedlem ble sittende fast i Oslo, og ikke hadde muligheten til å møte opp fysisk på NTNU for å jobbe med ambulansen. Det var ikke stort problem for gruppen, grunnet at dette medlemmet var ansvarlig for UX og UI Design. Dette gjorde medlemmet mindre avhengig av å møte opp, som har fungert bra. På en annen side er lengden av disse restriksjonene blitt undervurdert. Dette var ikke et stort problem da gruppen er vant med å jobbe digitalt, etter ett år med koronarestriksjoner.

Gjennom utviklingsperioden trengte gruppen jevnlig tilgang til simulasjonsrommet hvor ambulansesimulatoren stod. Dette var uproblematisk i utgangspunktet da simulatoren skulle brukes relativt lite til undervisning gjennom semesteret. Gruppen fikk i utgangspunktet fri tilgang til rommet hver torsdag, med noen unntak når rommet ble brukt av andre. Da ble det gjort en spesifikk avtale for den uken om hvilken dag som passet å arbeide i simulasjonsrommet. Gruppen tenkte derfor ikke å legge til tilgjengelighet til simulasjonsrommet i risikoanalysen. Dette viste seg å være en dårlig antakelse av gruppen. Hydraulikken som styrte ambulansesimulatoren sluttet å virke rundt april, noe som gjorde det umulig for gruppen å teste nye versjoner av programvaren i simulasjonsrommet. Dette gjorde det noe vanskeligere å samarbeide for gruppen, samt introduserte noe usikkerhet rundt kommunikasjon med DAQ'en i simulasjonsrommet. Hydraulikken sluttet å fungere etter gruppen var ferdig med det meste av testing. Konsekvensene var dermed relativt lave. Hadde hydraulikken sluttet å fungere tidligere i prosjektet ville dette definitivt vært et stort problem, og det burde blitt lagt til i risikoanalysen med spesifikke tiltak gruppen kunne gjøre.

9.3 Tidføring

De første ukene av prosjektet har hvert gruppemedlem vært gode på å notere tiden brukt på prosjektet i Clockify. Ettersom arbeidsmengden og intensiteten i prosjektet økte har flere av gruppemedlemmene blitt dårligere på å notere tid. Den endelige tids mengden brukt per oppgave bør derfor tas med en klype salt, da det er notert en del timer i etterkant. Arbeidsfordelingen per oppgave reflekterer på en annen side i stor grad det faktiske resultatet.

For å forbedre tidsføringen kunne det blitt innført strengere prosedyrer for notering av tid. I Clockify kunne gruppemedlemmene se omtrent hvor mye tid de andre gruppemedlemmene hadde notert. En del av prosedyren kunne vært å passe på at de andre gruppemedlemmene hadde husket å notere ned tid. En annen alternativ løsning kunne ha vært å automatisere dette ved bruk av verktøy i Visual Studio for å loggføre hva som blir jobbet med til hvilke tider. Dette ville fortsatt

ikke dekket noen aspekter av prosjektet, som undersøking hvor Visual Studio ikke en gang blir åpnet. En annen løsning hadde vært å bruke verktøy som Jira[30], der oppgaver får et tidsestimat, så noterer man tid brukt på disse oppgavene.

9.4 Tidsestimering

Tidsestimeringen har for enkelte ting fungert greit, men har bommet totalt på andre ting. Utviklingen av lesing og skriving til hydraulikken gikk fortere enn antatt, dette er grunnet god planlegging og undersøking på forhånd i planleggingsfasen av prosjektet. Utviklingen og abstrakteringen av rattets SDK har gått tregere enn estimatet, grunnet at kompleksiteten til SDK-en ble undervurdert, og at dokumentasjonen ikke har vært like klar og tydelig som andre ting som er blitt undersøkt. Brukergrensesnittets utvikling har også tatt mye lenger tid enn estimatet. Dette skyldes at integreringen tok lenger tid enn antatt, og en del uenigheter og diskusjoner i hvordan ting skulle løses. En annen del av prosjektet som tok mye lenger tid enn antatt var utviklingen av OpenGL 3D renderen, `AmbuGL`. Dette skyldes mye opplæring, mangel på erfaring og mye prøving og feiling. Grunnet tidsbruken på `AmbuGL`, ble det etter hvert bestemt at det ikke kunne jobbes mer på `AmbuGL`. Dette kuttet resulterte i en litt rotete kodebase i `AmbuGL` prosjektet.

Det er vanskelig å si hvordan tidsestimatet kunne blitt forbedret, da estimering et vanskelig tema. Underveis i planleggingsfasen er "Planning Poker" blitt benyttet for å komme frem til et estimat. Estimatene har mest sannsynligvis bommet grunnet mangel på erfaring og kunnskap innenfor emnene. På en side hadde det vært mulig å være mer pessimistisk i måten man estimerer tid på. På en annen side er dette noe som er blitt forsøkt, men gruppen kom frem til at dette ville medføre et uferdig prosjekt. Et sted gruppen bommet veldig på tidsestimat var brukergrensesnittet, der siste frist for utvikling var satt til 30. mars 2021, men den ble ikke ferdigstilt før 04. mai 2021. I etterkant burde gruppen vært mer pessimistisk med tidsestimatet til frontend utvikling da få i gruppen hadde kunnskaper innen utvikling av grafiske brukergrensesnitt. Dette er reflektert i designdokumenter som den orginale systemarkitekturen (figur C.2). Denne systemarkitekturen er relativt spesifikk på hva som kreves fra "Controlleren", og gruppen var svært klar over at lite funksjonalitet krevdes fra "Model". "View" var derimot gruppen klar over at trengte flere forskjellige komponenter for å fungere riktig. Arkitekturen gjenspeiler likevel en svært enkel model for brukergrensesnittet.

Et annet moment som burde varslet gruppen om at brukergrensesnitt kom til å ta mer ressurser enn forventet, var behovet for å forandre på rammeverk for utvikling av grafisk brukergrensesnitt. Det nødvendige byttet fra et kjent rammeverk i Qt til et ukjent rammeverk i wxWidgets vil naturlig nok føre til at ekstra tid kreves for å lære seg det nye rammeverket. Gruppen tok ikke nok høyde for dette, da dette byttet viste seg nødvendig etter at hoveddelen av planleggingsfasen var overstått, og gruppen var klar for å begynne å produsere kode.

[30] `https://en.wikipedia.org/wiki/Jira_(software)`

9.5 Gruppeledelse

Gjennom prosjektperioden har gruppeledelse vist seg å være en svært gjennomførbar arbeidsoppgave. Dette skyldes i stor grad alle gruppemedlemmenes evne til selvstendig arbeid. Gruppen fungerte også bra sammen, trolig grunnet at alle var kjent med hverandre fra tidligere. Samarbeidet kan også skyldes at gruppen hadde samme forventninger i faget, og lå rundt samme nivå fagmessig.

Gruppen bestod heller ikke av for mange med samme personlighetstype. Samme personlighetstype kunne vært problematisk, da to stykker på gruppen hadde sterke meninger om det meste under design og utviklingsprosessen. Samarbeidet på gruppen kunne blitt problematisk hvis alle fire på gruppen skulle diskutert ethvert design valg like mye. Dette betyr ikke at enkelte gruppemedlemmer meldte seg ut av all felles diskusjon. Alle var aktive i gruppen, og alle delte sine tanker under oppfordring. Alle delte også tankene sine hvis gruppen bevegde seg mot det noen oppfattet som en dårlig løsning.

Gjennom utviklingsprosessen ble prosjektleder mer oppmerksom på rollen en prosjektleder ofte får. Beslutningsansvaret står på prosjektleder hvis det er flere valg i et prosjekt. Dette ansvaret kan også lede til at prosjektleders ord eller beslutninger har ekstra vekt, når valgene ikke burde få det. Det er derfor viktig som prosjektleder å høre godt på andres forslag, før prosjektleder kommer med egne løsningsforslag. Prosjektleder har erfart dette selv gjennom prosjektet, og har forsøkt gjennom prosjektet å litt mer passiv i diskusjoner.

Prosjektleder vil også få større ansvar for en del små oppgaver som burde gjøres gjennom et prosjekt. I prosjektet innebar dette oppgaver som kommunikasjon med eksterne grupper, møteplanlegging og generelt arbeid for å sikre at alle gruppemedlemmer jobbet mot et felles mål. Dette er en arbeidsoppgave som lett kan glemmes og nedprioriteres. Prosjektleder opplevde selv gjennom prosjektet at mer tid burde blitt brukt på disse arbeidsoppgavene. Enkelte momenter viser uklart rolleansvar. Et eksempel som kommer opp er et element i brukergrensesnittet kalt `GasAndBreak`. Denne klassen viser brukeren hvor nedtrykt gassen og bremsen er. Tilhørende til klassen står det tekst for “-20” og “20”. Disse verdiene ble lagt inn tidlig i utviklingsfasen av klassen, og har lite intuitiv kobling til verdier rattet faktisk har. Verdiene “-1” og “1” ville vært mer nøyaktig å benytte seg av. Det er svært lett å bytte på disse verdiene, men grunnet uklart ansvarsområde var det ingen i gruppen tok denne oppgaven.

Prosjektleder selv opplever ikke at de har gjort mye aktivt som prosjektleder. Likevel har gruppemedlemmer og prosjektleder selv opplevd at dårlig ledelse raskt kan lede til dårlig gruppesamarbeid. Det kan derfor være at prosjektleders manerer og generelle holdning til utvikling, kommunikasjon, forventninger og samarbeid har gjort gruppen mer effektiv.

9.6 Koden

Deling av variabler mellom klasser

Som nevnt tidligere i diskusjonen var det enkelte deler av kodebasen som ikke ble ferdigstilt grunnet tidsestimering. Det var også enkelte andre elementer i kodebasen som burde blitt vurdert bedre. Dette gjelder spesifikt bruk av delte ressurser. Deling av ressurser kommer fram under f.eks. avsnitt 6.2.3. Her trenger flere paneler tilgang til samme variabler. Den enkleste metoden er gjerne å ha "public member data" til klasser, som alle andre klasser kan lese av når det trengs. Dette er noe problematisk, da man lett mister kontrollen over hvor disse variabelene blir modifisert. Standard er å benytte seg av "private member variables" og bruker "setters" eller "getters" istedenfor. Disse er nyttige da man kan benytte seg av breakpoints inni en slik metode og se hva som skjer hvis noe feil skjer i systemet. Problemet er at man da fortsatt må ha tilgang på klassen som eier disse metodene. Dette vil fort lede til mye duplisering av kode for svært enkel funksjonalitet.

En annen mulighet er heller å kun ha en referanse til verdiene direkte. Dette reduserer duplisering av kode, da man kun må oppdatere en verdi. Andre klasser som har en referanse til denne verdien vil dermed automatisk få sin egen verdi oppdatert.

Arv

Bruken av klasser som `BaseFrame` og `MainController` arver kun fra wxWidgets da dette er krevd for å lage et flertrådet program med et grafisk brukergrensesnitt. Programmet har også klasser som `InputController` og `GenericButton` som inneholder alle metoder som er felles for sine subklasser. Dette er gjort for å gjøre det lettere å få et rask overblikk over hvilke funksjonalitet disse klassene skal gjennomføre. Det er også grupper av klasser som burde fått en superklasse, men som prosjektgruppen ikke rakk å implementere. Spesifikt er dette alle `CustomWidget` klassene, som trolig kunne fått en egen superklasse.

Stabilitet

Koden er stabil så lenge bygging av prosjektet følger installasjonsinstruksjonene. Gruppemedlemmene har tidligere erfart at programvare utviklet av medlemmene på gruppen er svært ustabil, og vil lett slutte å fungere når noe uforutsett skjer i programvaren (uformelt kalt et krasj). Det skal sies at det eksisterer en rekke uhåndterte avvik i programvaren som kan kreve en omstart av programmet. Den mest markante av disse er trolig rattet som slutter å fungere fullstendig hvis logitech GHUB ikke er åpen når man prøver å starte styring med rattet. Dette resulterer i at man må starte programmet på nytt. Programmet vil likevel ikke slutte automatisk under noen av testene gruppen gjorde på det nye systemet. Det vil også stort sett gi en naturlig feilmelding for hva som gikk galt, og vil fortsette å fungere etter brukeren har rettet opp i feilen. Det nye systemet har ingen kjente alvorlige feil.

9.7 Planlegging

Planlegging innenfor systemutvikling er ofte motivert av minst mulig omskriving av kodebasen. Det var enkelte elementer i kodebasen som krevde en omskriving. Utviklingen av `InputController` var en av disse, da gruppen kom fram til først ved senere møter at en felles arkitektur for alle stryingskontrollerene ville være hensiktsmessig. `MainController` måtte også bli lagd på nytt en del ganger for å gi støtte for flertråds aksjoner og kommunikasjon til hovedtråden som det grafiske brukergrensesnittet kjørte på.

10 Konklusjon

10.1 Implementert funksjonalitet

Gruppen så tidlig i utviklingsfasen potensielle utvidelsesmuligheter i det nye systemet. Disse utvidelsesmulighetene var motivert av at frykt på gruppen på for lite arbeidsmengde. Det viste seg at utvidelser ikke ble relevant i løpet av prosjektperioden. Dette skyldes at utviklingen, spesielt av frontend, tok markant lengre tid en gruppen forventet. Disse problemene ble diskutert nærmere i avsnitt 9.4.

Gruppen opplever likevel at de jobbet jevnt gjennom hele utviklingsperioden og fikk implementert all funksjonalitet oppdragsgiver hadde spesifisert i den initiale kravspesifikasjonen. Denne kravspesifikasjonen kommer fram av møtereferat med Hans Martin 19. januar 2021. Det kommer også fram av initial Use Case diagram (figur C.1) og endelig Use Case diagram (figur 2.1).

Det var spesielt ønsker for å implementere VR funksjonalitet i prosjektet, som en ekstra addisjon til originale kravspesifikasjon. Gruppen ser det som uheldig at det ikke var tid til å implementere denne funksjonaliteten. Prosjektgruppen ser seg fortsatt fornøyd med valget om å fokusere på den mest essensielle funksjonaliteten og produsere disse til en god standard. Produksjonen av god kode til essensiell funksjonalitet, istedenfor å strekke seg for å implementere all funksjonalitet med dårlig kvalitet, ser gruppen som riktig valg i denne oppgaven.

10.2 Systemets kvalitet

Tidlig i utviklingsfasen ble det bestemt flere prosedyrer for å sikre kvaliteten i koden som utvikles. Disse prosedyrene kan bli sett i avsnitt 7 og diskuteres ytterligere i avsnitt 9.1. Gruppen så i stor grad at prosedyrene for kvalitetssikring var hensiktsmessige, men at tidsforbruket for denne kvalitetssikringen var en del høyere enn det originale tidsestimater ledet gruppen til å tro. På grunn av denne feilen i tidsestimatet så gruppen seg nødt til å redusere fokuset noe på kvalitetssikring. Dette resulterte i mindre dokumentasjon for funksjonalitet utviklet senere i utviklingsfasen, færre tester til funksjonalitet utviklet senere og en dårligere oppfølging av kodestandard i kodebasen. Kvaliteten på systemet er fortsatt relativt høy, med dokumentasjon og forklaring til de fleste metoder og funksjoner brukt i det nye systemet.

Det nye systemet holder seg også godt til prinsipper innenfor MVC. Dette gjør systemet lettere å vedlikeholde da ansvarsområder er klart delt, og søking etter spesifikke problemer vil være lettere. De fleste klasser er også naturlig oppdelt i enkeltkomponenter der det er hensiktsmessig, og i super- og subklasser der det er mest naturlig.

Koden er ikke feilfri, men både oppdragsgiver og gruppemedlemmene er fornøyde med sluttproduktet.

10.3 Planlegging

Flere gruppemedlemmer har tidligere erfaring med samarbeid i forskjellige prosjekter gjennom bachelor perioden. Et felles problem som har oppstått i disse prosjektene har vært mangel på planlegging og forberedelse før utvikling starter for fullt. Dette leder ofte til et behov for å slette store deler av kodebasen, og starte på nytt med erfaringene gruppen har gjort. Bachelorprosjektet hadde planlagt mye bedre. Det var færre ganger et behov for å forandre kursen drastisk grunnet feil i tankegangen tidlig i prosjektet.

Det var flere momenter i planlegging som kunne blitt gjennomført bedre. Spesielt kunne tidsestimering blitt gjennomført bedre, da utviklingen sjelden tok tiden gruppen estimerte. Det var enkelte funksjonaliteter som tok markant kortere tid en estimert. Dette var ikke problematisk for gruppen da dette kun ga mer tid til utvikling av ny funksjonalitet. Det var derimot mer problematisk med den funksjonaliteten som tok markant lengre tid, som diskutert nærmere i avsnitt 9.4.

Gruppen ser på det som realistisk at bedre tidsestimater vil komme etter hvert som gruppen får mer erfaring innenfor programmering. Selv om gruppen ikke vet nøyaktig hvor lang tid det tar å sette seg inn i nye projekter, vil gruppemedlemmene etter hvert få et erfaringsgrunnlag å bygge på.

10.4 Gjennomføring

Utviklingen av det nye systemet ble på mange måter gjennomført bra. Gruppen kom ikke over mange overraskelser i løpet av utviklingsfasen. Hovedsakelig var det kun kommunikasjon med digitalt ratt og grafisk brukergrensesnitt som viste seg å være mer utfordrende. Gruppen kom også tidlig fram til at en plandrevet utviklingsmodell ville være mest hensiktsmessig da kravspesifikasjonen var godt definert tidlig i prosjektet. Dette viste seg også å være riktig i slutten av prosjektet, med kun små ekstra addisjoner til den originale kravspesifikasjonen.

Gruppen bestemte seg også tidlig for å gjennomføre et fast gruppemøte i uken, og eventuelt ta seg tid til flere hvis dette viste seg å være nødvendig. Denne løsningen virket også bra for gruppen. Prosjektleder tok gjennom prosjektperioden opp spørsmålet om ekstra eller færre møter i løpet av uken. Gruppen så seg fornøyd med det ene faste møtet pluss eventuelle ekstra møter for spesifikk arkitektur diskusjon eller diskusjon rundt andre løsninger. Det skal også nevnes at gruppen tok oftere møter under planleggingsfasen av prosjektet, da det er hensiktsmessig å diskutere samlet når hele systemets arkitektur bestemmes.

10.5 Gruppesamarbeid

Samarbeidet på gruppen fungerte stort sett særdeles bra på tross av koronasituasjonen som gjør det vanskeligere å samles fysisk. Gruppen har alle erfaring med digital kommunikasjon mellom venner. Etter et år inn i pandemien begynner medlemmene også å bli vant til hvordan digital kommunikasjon burde gjennomføres. Det skal nevnes at enkelte problemer oppstod gjennom prosjektperioden. Det ble spesielt nevnt noe manglende kommunikasjon mellom gruppemedlemmene som

kunne møte opp fysisk på NTNU i Gjøvik. Det var enkelte hendelser hvor beskjeder gitt av oppdragsgiver ikke ble videreført til gruppemedlemmene som ikke hadde mulighet til å møte opp på simulasjonsrommet. Disse hendelsene skjedde relativt sjeldent, og tiden fra beskjeden var gitt til gruppemedlemmene fysisk på NTNU i Gjøviktil alle medlemmer hadde fått beskjeden overskride sjeldent to dager.

Det var også enkelte arbeidsoppgaver som hadde for uklart rolleansvar. Disse problemene var nevnt tidligere i avsnitt 9.5. Gruppen er noe usikker på nøyaktig hvordan disse problemene kunne blitt håndtert riktig, men det er trolig at prosjektleder burde tatt en mer bestemt rolle, eller simpelthen listet opp alle problemer prosjektleder fant i programvaren. Dette måtte i så fall oppklares tidlig i utviklingsfasen slik at resterende gruppemedlemmer ikke ble oppgitte eller irriterte.

Gruppen har ellers ikke opplevd noe særlig problematikk rundt samarbeid. Humøret og stressnivået har vært relativt lavt gjennom hele utviklingsperioden, og det har vært god kommunikasjon mellom gruppens medlemmer.

11 Videre arbeid

Videre arbeid innenfor dette emnet kan deles opp i to kategorier: ferdigstilling og utvidelser.

11.1 Ferdigstilling

Utviklingen av det nye systemet fikk implementert det meste av funksjonalitet som gruppen ønsket å implementere. Det gruppen derimot ikke fikk tid til var å sikre at all kode hadde tilhørende enhetstester, og heller ikke at systemet alltid fulgte kodestandarden bestemt i starten av prosjektet. Det er også trolig andre mindre problemer i kvaliteten til det nye systemet. Dette burde være første prioritering for videre arbeid på det nye systemet. Gruppen estimerer at denne arbeidsoppgaven ville tatt rundt 100 arbeidstimer for en av gruppens medlemmer. Det er vanskelig å estimere hvor lang tid en ekstern utvikler ville brukt for å gjøre de nødvendige forbedringene.

Det er også enkelte funksjonaliteter som ble lagt til det nye systemet, som ikke fikk nok utviklingstid. Disse funksjonalitetene er dermed trolig ustabile, og burde kontrolleres og forbedres. Dette er funksjonalitet for å lage en sekvens fra en .csv fil samt funksjonaliteten som måler total tid hydraulikken er aktiv.

11.2 Utvidelser

Gruppen kom tidlig i utviklingsfasen med en rekke mulige utvidelser, som det senere viste seg gruppen ikke hadde tid til å utvikle. Dette inkluderte momenter som: VR støtte, bedre 3D-modeller, “fra film til sekvens” og flytte ratt over på venstre side.

VR støtte har vært et interessemoment både for prosjektgruppen og oppdragsgiver. Dette vil trolig være et godt område å fokusere på først.

3D-modeller er også et moment gruppen ønsket å ha i applikasjonen. Per nå benytter gruppen seg av en grei 3D-modell hentet på nett. Denne burde erstattes av en bedre modell samt en modell gruppen er sikker på har en god brukslisens. Den nåværende 3D-modellen har en ukjent brukslisens, noe som ikke er svært problematisk da systemet kun skal brukes offline. Det er likevel bedre hvis denne opphavsretten er kjent.

Fra film til sekvens beskriver en relativt kompleks metode for å omdanne en filmsnutt til en estimering av hvordan bilen kjøres. Dette kan igjen brukes til å finne kreftene som påvirker bilen, som kan brukes for å generere en sekvens. Dette er en relativt ambisiøs oppgave, og burde bli implementert av noen med kunnskap innenfor datasyn eller kunstig intelligens. Dette kan trolig bli gjennomført med å finne nøkkelpunkter i hvert bilde, for så å finne ut hvordan disse nøkkelpunktene flytter seg mellom bilder. Dette er teknikker innenfor feltet "Multi Target Tracking (MTT)".

Flytting av ratt til venstre side. Per nå er det digitale rattet på høyre side av bilen. Dette er ikke standard i norske biler da rattet er på venstre siden i ambulan-

ser. Dette er en oppgave bedre rettet mot maskiningeniører og elektroingeniører enn datautviklere.

Referanser

[1] I. Sommerville, *Software engineering, tenth edition*. Pearson Education, 2019, s. 49–51.

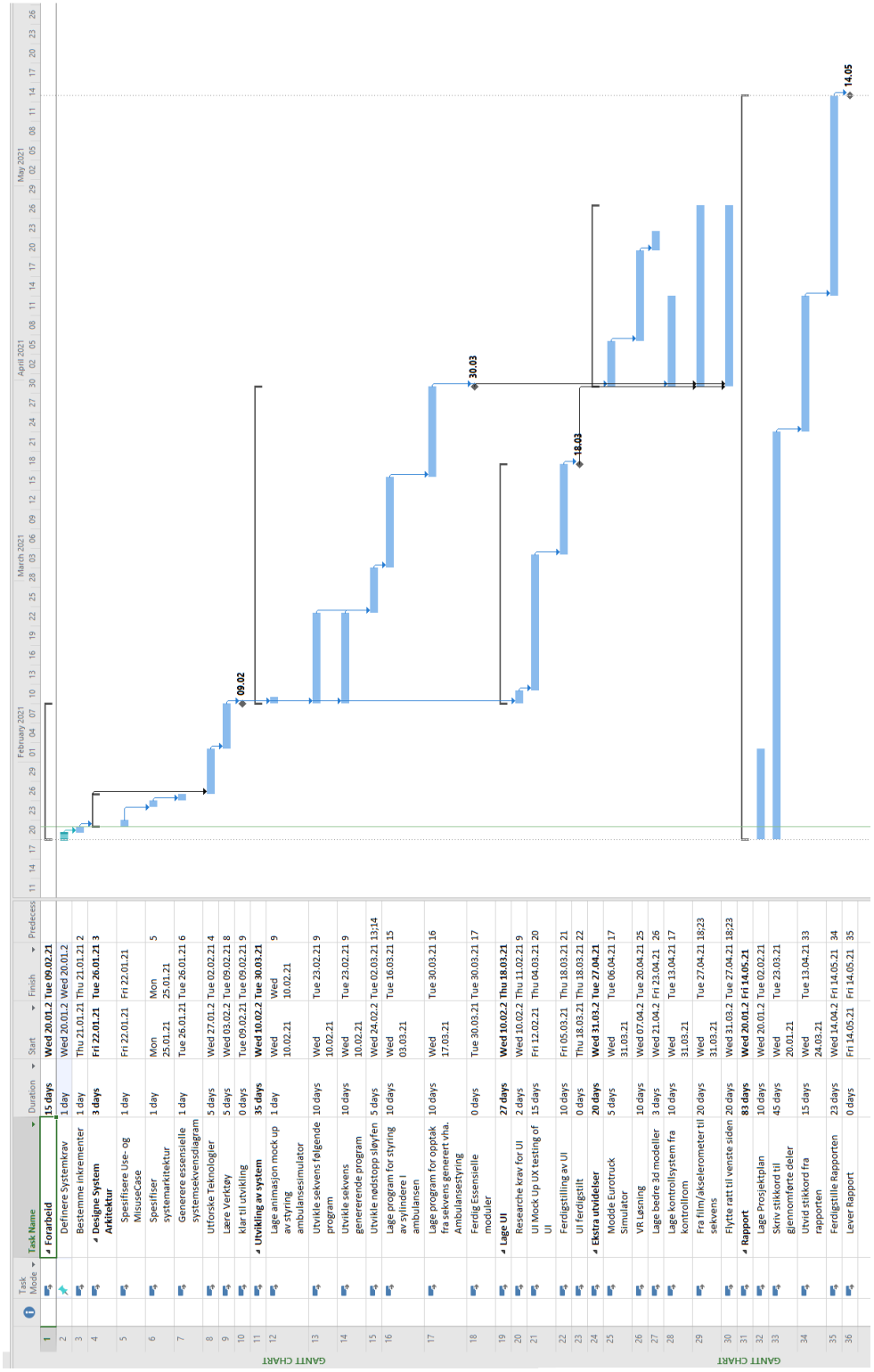
[2] J. Nielsen og T. K. Landauer, «A mathematical model of the finding of usability problems», *CHI '93: Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems*, s. 206–213, 1993.

[3] D. V. Widder, *The Heat equation*. Academic Press, 1975, s. 3–4.

A Definisjoner

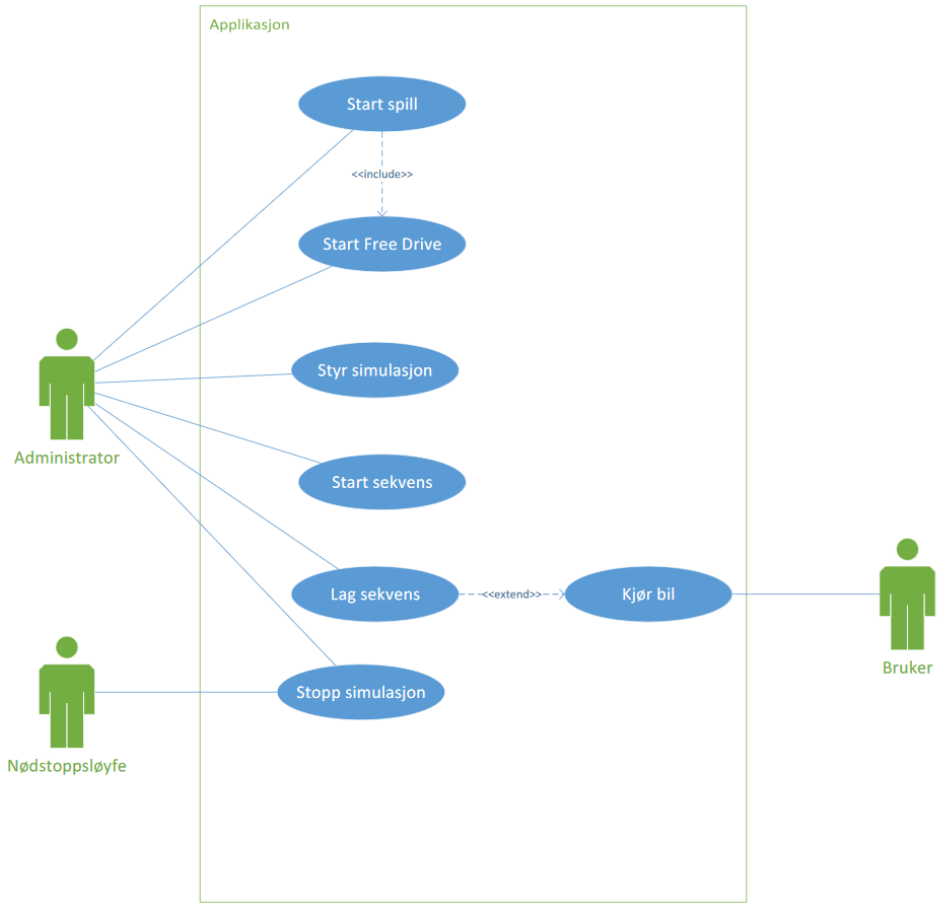
AR	Augmented Reality eller Utvidet virkelighet. Addisjon av ekstra virituelle elementer legges til den virkelige verden. Skiller seg fra VR i at bruker fortsatt kan se sine virkelige omgivelser.
DAQ	National Instruments Data Acquisition. Brukes til styring av hydraulikken.
ETS2	Forkortelse for Euro truck Simulator 2
Euro truck Simulator 2	Et spill som simulerer livet til en truckfører.
GTA V	Spill: Grand Theft Auto.
Kvaternion	Et komplekst tall i fire dimensjoner som representerer en rotasjon i tre dimensjoner.
Modde	Å modifisere noe ved å f.eks. legge til kode eller ressurser i et spill
NI-DAQmx	Driveren til DAQ'en.
Rendere	Prosessen å konvertere et 3d data objekt til et 2d bilde som kan vises på skjermen.
SDK	Software Development Kit. En kodebase utviklet for kommunikasjon og bruk av kode utviklet av andre aktører.
VR	Virtual Reality eller virituell virkelighet. Skjermer settes rett forran øynene til en bruker, disse skjermene viser et eget bilde av en virituell virkelighet for brukeren
Wireframe	Et bilde eller et sett med bilder som viser de funksjonelle elementene ved av en nettside eller program. Vanligvis brukt for å planlegge en nettside eller et programs grafiske struktur og funksjonalitet

B Framdriftsplan

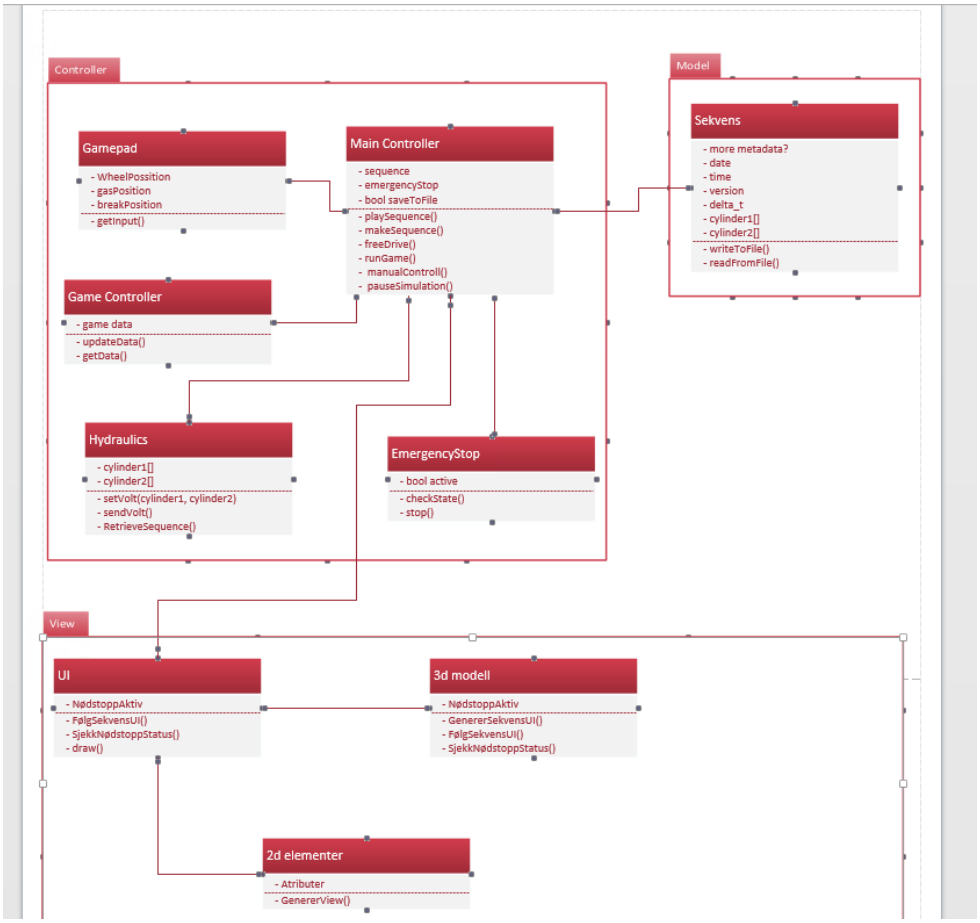


Figur B.1: Gantt-diagram

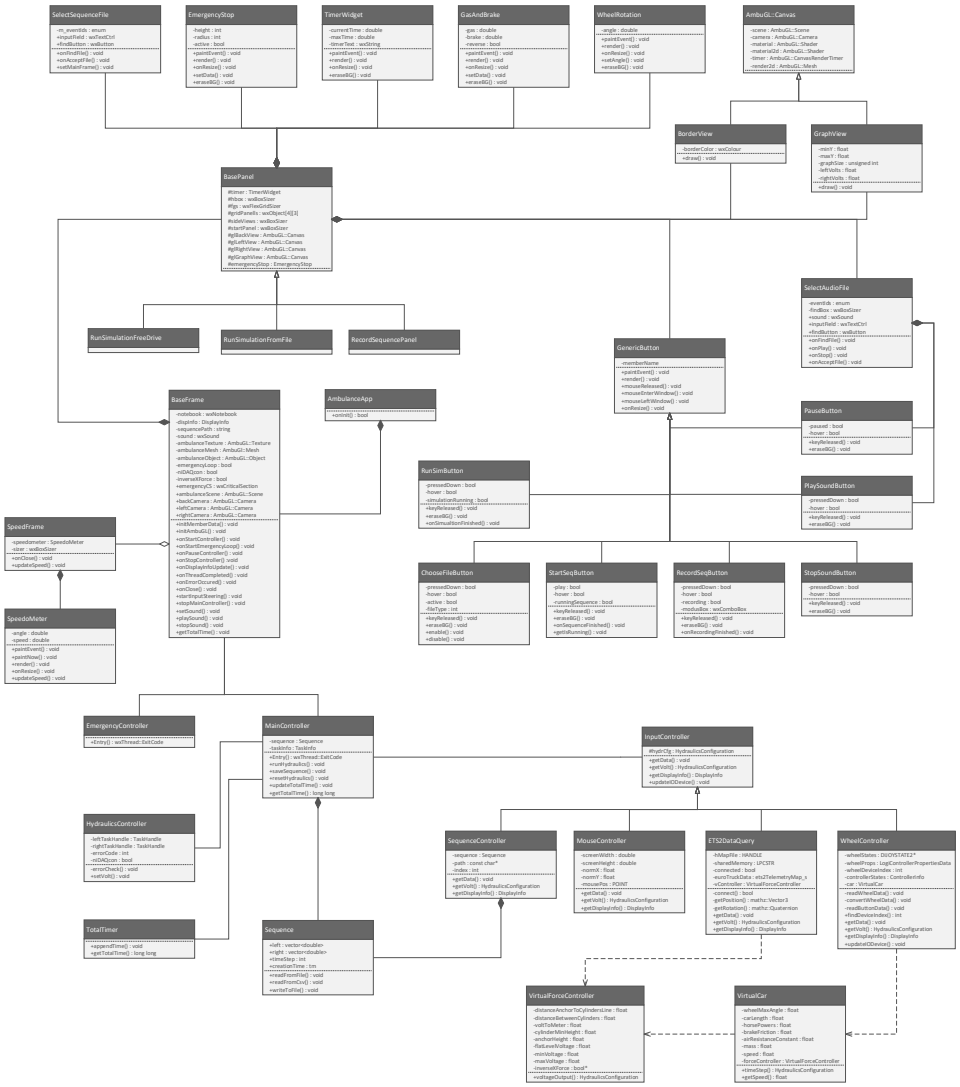
C Designdokumenter



Figur C.1: UseCase diagram for prosjektet



Figur C.2: Originale systemarkitektur



Figur C.3: Klassediagram for det nye systemet

D Orginal Prosjektplan

Prosjektplan

Aslak Tøn
Håkon Trøan Muggerud
Donjetë Haziri
Isac Mikal Kallevig

February 15, 2021

Innhold

1 Mål og rammer **1**
1.1 Bakgrunn 1
1.2 Prosjektmål 1
1.3 Rammer 2

2 Omfang **2**
2.1 Fagområde 2
2.2 Avgrensning 2
2.3 Eksisterende løsning 2
2.4 Oppgavebeskrivelse 3

3 Prosjektorganisering **4**
3.1 Ansvarsforhold og roller 4
3.2 Rutiner og regler i gruppen 4

4 Planlegging, oppfølgning og rapportering **6**
4.1 Valg av SU-modell 6
4.2 Vurdering av plandrevene utviklingsmodeller 7

5 Organisering av kvalitetssikring **8**
5.1 Formål 8
5.2 Standarder 8
5.3 Konfigurasjonsstyring 9
5.4 Code Review 9
5.5 Planlegging 9
5.6 Testing 9
5.7 Arbeidsflyt 10
5.8 Risikoanalyse 11
5.9 Risikoanalyse Løsninger 11

6 Plan for Gjennomføring **12**
6.1 Gantt-skjema 12
6.2 Milepæler, beslutningspunkter og statusrapporter 15

1 Mål og rammer

1.1 Bakgrunn

I sammenheng med bachelorgraden tatt av datastudenter ved NTNU i Gjøvik, gjennomfører alle studenter et endelig prosjekt mot en oppdragsgiver. Prosjektet gitt til denne gruppen var å utvikle en ny ambulansesimulator til bruk for opplæring av helsestudenter i faget paramedisin[1]. Det var allerede utviklet en tidligere versjon av ambulansesimulatoren som nå ønskes å oppdateres med bruk av dagens teknologi og god dokumentasjon. Vedlikehold av foreløpig system blir gjenomført av Pål Erik Enderud, som var med på utviklingen av det forrige systemet. Det vil være viktig gjennom utviklingen av dette systemet å holde dialog med vedlikeholder da det trolig vil bli dems ansvar å føre vidre vedlikehold av programvaren utviklet.

1.2 Prosjektmål

1.2.1 Effektmål

Det nye systemet basseres på funksjonalitet allerede eksisterende i et gammelt system, med et ønske om bedre dokumentasjon, reponsivitet og vedlikeholdbarhet. Relevante effektmål vil da være:

- Redusere ressurser brukt på vedlikehold av programvaren
- Spare tid ved at systemet blir mer responsivt og intuitivt
- Systemet skal være robust nok til å fungere i minst 10 år

1.2.2 Resultatmål

Systemet skal brukes som et digitalt opplæringsverktøy for studenter ved bachelor i paramedisin. Systemet skal også være bedre dokumentert og systematisert, slik at vedlikehold er mindre tidskreven. Resultatmål vil da være:

- Tilby platform for digital opplæring av nye studenter ved paramedisin
- Systemet gir en livsnær simulasjon av kjøring og helsehjelp i en ambulanse
- Gi mulighet for rask utvikling av sekvenser til bruk i opplæring
- Tilby funksjonalitet for selv å styre simulatoren, eller styring fra en definert sekvens
- Alle funksjoner og metoder skal ha tilhørende dokumentasjon

1.2.3 Læringsmål

Læringsmål kan hentes fra emnebeskrivelsen[2]. I bachelorprosjektet vil studentene kommunisere med en reell oppdragsgiver der de knytter sin tidligere kunnskap om systemutvikling til å organisere og lede et større prosjekt. De vil lære om profesjonell bruk av versjonskontroll, kodestandarder, testrammeverk og dokumentasjon.

[1] `https://www.ntnu.no/studier/bparamed`
[2] `https://www.ntnu.no/studier/emner/BIDAT39` (hentet 2021-01-22)

Spesifikt for dette prosjektet skal studentene få inføring i styring av hydraulikk og robotikk. De vil sette seg inn i C++ programmering knyttet til National Instruments sine kontrollere, og de vil bruke UX design og grafikkprogrammering til å sette sammen et grafisk brukergrensesnitt.

Dersom studentene får tid til å implementere ekstra funksjonalitet vil de også sette seg inn i programmering med VR og modding av spill.

1.3 Rammer

Ambulansesimulatoren skal utvikles av en gruppe på fire studenter fra fagene BIDAT og BPROG. Studentene veiledes av Frode Haug. Veileder disponerer en time per uke, hvorav 30 minutter er dedikert til møte med studentene, og 30 minutter er dedikert til forberedelse til møte med studentene. Oppdragsgiver plikter også til å disponere 25 timer til studentene i løpet av prosjektperioden. Prosjektet, som inkluderer programvaren og prosjektrapport, skal være ferdigstilt innen 2021-05-20.

Systemet skal tilby digital opplæring av studenter ved paramedisin gjennom simulasjon av situasjoner som kan oppstå mens man gjennomfører førstehjelp i en ambulanse.

2 Omfang

2.1 Fagområde

Bruk av datamaskin for å trene på ulike kunnskaper og ferdigheter kalles “maskinbasert læring”. Til dette brukes pedagogisk programvare, som kan deles inn i drill/trening, dialog, simulering, spill, oppdage, problemløsning og modellbygging. Drill vil f.eks. være mer programstyrt, mens modellbygging vil være brukerstyrt.

Fordeler med en slike læringsmetode inkluderer større tilgjengelighet, kosteffektivitet ved opplæring av mange studenter, og det kan lett tilpasses ulike fagområder. En tilpasset læringsplattform kan ta lang tid og kreve mye ressurser å sette i stand, men på lang sikt er det en investering som kan spare kostnader på instruktører, klasserom, reising osv.

2.2 Avgrensning

I denne oppgaven skal vi se nærmere på ferdighetstrening i simuleringer. Dette er en veldig effektiv måte man virtuelt kan trene på virkelige situasjoner. Dette gjelder spesielt scenarioer der trening i den reelle verden vil være for kostbar, farlig, tidskrevende eller kompleks. Simulering med ferdighetstrening har et spesifikt læringsmål, f.eks. i forbindelse med styring av kjøretøy eller faste arbeidsrutiner.

2.3 Eksisterende løsning

I forbindelse med et nytt studieprogram i paramedisin som starter til høsten 2021 vil NTNU oppgradere sitt nåværende system for ambulansesimulering. Senter for

simulering og pasientsikkerhet ved Institutt for helsevitenskap på Gjøvik har flere fasiliteter for simulering innen helsefagutdanninger, inkludert ambulansesimulatoren.

Simulatoren består av en reell ambulanse som er plassert på et hydraulisk bevegelig understell styrt av programvare på PC. Denne programvaren ble opprinnelig utviklet av studenter i perioden 2008 til 2009 og har utviklet seg en del siden den tid. Det er mangel på dokumentasjon, som fører til at systemet er vanskelig å drifte og videreutvikle.

2.4 Oppgavebeskrivelse

Vår oppgave er å utvikle et nytt system fra bunnen av, som vil ha et intuitivt grensesnitt og god dokumentasjon. Det er også et ønske om oppgradering av PC i kontrollrom samt ratt og pedaler i ambulansen.

Simulatoren har i hovedsak to bruksområder. Den første er avspilling av en eksisterende sekvens som inneholder gitte bevegelser ambulansen skal utføre. Slike sekvenser varer omtrent 1-5 minutter. Ambulansepersonell utfører ferdighetstrening bak i bilen når disse bevegelsene simuleres. Denne delen av simuleringen er dermed programstyrt.

Det andre bruksområdet innebærer i tillegg en sjåfør som styrer ambulansen fra førersetet. Sjåføren vil spille et spill/simulasjon som vises på prosjektor. Hydraulikken vil reagere når sjåføren gasser, bremser og svinger. Denne delen av simuleringen er dermed mer brukerstyrt.

Begge disse scenarioene vil alltid ha en person som kan starte/stoppe simuleringen fra et kontrollrom. Simuleringen kan også stoppes ved hjelp av en sikkerhetssløyfe som består av lysgardiner, samt sensorer i dørene. Personen i kontrollrommet vil også få en forhåndsvisning av ambulansens nåværende posisjon i sanntid.

Uavhengig av simulasjonen skal programvaren tilby verktøy for å lage en skreddersydd sekvens. Denne sekvensen lagres som en fil, og kan senere lastes opp for avspilling på ambulansesimulatoren. En sekvens kan også genereres ved at det gjøres opptak av en brukerstyrt simulering.

Ambulansen står på en styringsplate som er ankret foran. Den beveger seg vha. to sylindere lenger bak. (kreditering: Isac Kallevig)

3 Prosjektorganisering

3.1 Ansvarsforhold og roller

- Aslak Tøn - Prosjektleder
- Håkon Trøan Muggerud - Utvikler
- Isac Mikal Kallevig - Utvikler
- Donjetë Haziri - User Interface (UI)/User eXperience (UX) Hovedansvarlig

3.2 Rutiner og regler i gruppen

- Arbeid
  - Kjernetid kl. 09:00 - 15:00, mandag - fredag.
    Dvs. at alle skal være tilgjengelig på Discord[3] i det spesifiserte tidsrommet. Det blir tatt hensyn til at ikke alle ønsker/forespørsler/spørsmål kan bli løst snarest, men de involverte gruppemedlemene må anerkjenne ønsket/forespørselene/spørsmålene innen dagen, og gi et ca.

[3] https://discord.com/

estimat når de kan komme tilbake til gruppemedlemet med ønsket/-forespørselen/spørsmålet.

- Arbeidsmengde: 30 timer per uke, per gruppemedlem
- Aktiv deltagelse.
  Det involverer at alle i gruppen ikke bare er med på å skrive, utvikle og møte opp til møter, men og er med å diskutere, argumentere og komme med ideer.
- Fast møte for gruppen hver mandag på 5-10 min, for å sammenkjøre agendaer for kommende møter. Agendaene generers individuelt på forhånd. Møtetidspunkt avtales felles på Discord
- Faste møter for planlegging av videre utvikling og status holdes minst to ganger i uken. Et av møtene holdes med eller uten veileder tirdager kl. 08:30. Det andre møte holdes torsdager kl. 10:00 Det kan settes av tid for flere møter per uke hvis det er behov, disse avtales over discord.
- Referat fra møtene skrives av Aslak. Hvis Aslak ikke er til stede eller ikke har kapasitet vil en av de andre gruppe medlemene ta oppgaven.
- Statusoppdateringer på progresjon holdes over gruppens Discord kanal. Hvis et gruppemedlem henger etter skal de melde i fra så fort som mulig, slik at gruppen kan ta en felles avgjørelse om ny arbeidsfordeling. Om det er nødvendig med et møte for å fordele oppgaver på nytt, vil dette også diskuteres.
- Faglige uenigheter skal diskuteres fram til en majoritet i gruppa er enige om veien framover. Demokratiet skal så respekteres. Dvs. hvis flertallet har bestemt en retning framover, vil de som er unienige fortsatt jobbe for en felles interesse. Alle skal bidra aktivt i diskusjoner ved uenigheter.

- Kommunikasjon
  - Hoved kommunikasjonskanal for gruppen er på Discord. Beskjeder, møter og annen informasjon deling innad i gruppen holdes på gruppens Discord kanal.
  - Møte med veileder er satt opp hver tirsdag kl. 08:30 over Zoom. Om gruppen mener et møte ikke skulle være nødvendig, må gruppen informere veileder om det dagen før.

- Møte med oppdragsgiver planlegges på slutten av hvert møte, for å ta hensyn til oppdragsgivers flytende timeplan. Dette møtet holder sted over teams, om ikke annet er spesifisert.

- Brudd på regler
  - Lette brudd på grupperegler resulterer i at den ansvarlige må betale lunsj på NTNU for resten av gruppen. Lette brudd inkluderer; dårlig punktlighet til oppmøte for felles møter, lite bidrag i felles diskusjon, gjentatte problemer med å overholde tidsfrister.

  - Ved større brudd, vil det holdes et møte med alle på gruppen om hva som ble brutt og hva som er problematisk. Alle forventes å møtes i tide når det er enighet om møtepunkt. Tre lette brudd vil telles som et større brudd. I tilegg vil mangel på levering av avtalt arbeid til avtalt tid telles også som større brudd, dette gjelder ikke hvis det varsles på forhånd om problemer med gjennomføring av arbeid.

  - Ved to større brudd vil en skriftlig melding sendes per mail til aktuelt gruppemedlem, med kopi til resten av gruppen.

  - Ved tre større brudd vil det settes opp et møte med veileder. Hvor det vurderes om gruppen burde splittes opp. Arbeid opp til og med splittelse sees på som felles arbeid og har felles eierskap. Alt arbeid etter splittelse sees på som de enkeltes arbeid.

4 Planlegging, oppfølgning og rapportering

4.1 Valg av SU-modell

4.1.1 Styrende momenter for valg av utviklingsmodell

Karakteristika ved valg av smidige utviklingsmodeller er at de har et stort behov for fleksibilitet. Dette er ofte nødvendig i dagens utviklingsmiljø, når mange produkter har et behov for rask endring i retning, etterhvert som det blir større klarhet i hva kunden trenger.

Dette prosjektet krever ikke denne fleksibiliteten, da oppdragsgiver allerede har spesifikke ønsker for hvilke funksjonaliteter som skal være inkludert i systemet som skal utvikles. Oppdragsgiver har også klarhet i hva de ønsker fra tidligere produkt som leverte samme funksjonalitet. Oppdragsgiver har spesifisert at ønsket er å bygge opp et liknende system til dagens system, og at dette systemet bygges opp fra grunnen av med god dokumentasjon for lett vedlikehold.

Gitt denne informasjonen kom utviklingsteamet raskt fram til at en smidig utviklingsmodell ville ikke være nødvendig i dette prosjektet. Utviklingsmodeller slik som Scrum og Kanban vil kunne utnyttes i dette prosjektet, men vil gjøre

elementer som dokumentasjon og planmessig systemarkitektur vanskeligere å implementere. Behovet for planmessig systemarkitektur kommer fra et behov om å holde systemet vedlikeholdbart i lang tid etter at utviklingsteamet er ferdig med utviklingen av systemet.

4.2 Vurdering av plandrevene utviklingsmodeller

4.2.1 Fossefall

Fossefall inneholder trekk som passer godt i utviklingen av simulatoren. Fossefalls plandrevene modell krever at systemkrav og systemarkitektur er på plass før utvikling av systemet begynner. Dette vil bidra til at systemet i størst mulig grad har et helhetlig design som er lett å vedlikeholde. Problematikken med fossefall er at det krever god forståelse av problemstillingen tidlig i prosjektet. Da feil antagelser under design av systemarkitektur fort krever at store deler av systemet må genereres på nytt. Utviklingsteamet mangler kunnskaper innenfor kontrollering av hydrauliske sylindere, og velger derfor å unngå å følge en fossefallsmodell i tilfelle viktige momenter er glemt under planleggingsfasen.

4.2.2 Gjennbruksorientert systemutvikling

Gjennbruksorientert utviklingsmodell ville også vært relevant i vårt tilfelle, da systemet som skal utvikles allerede eksisterer. Gruppen vil til tider benytte seg av gjenbruk av gammel kode til ambulansesimulatoren, men oppdragsgiver har gitt utrykk for at de ønsker et nytt system med god dokumentasjon. Det vil derfor være et fokus på å generere ny kode framfor gjenbruk av gammel kode.

4.2.3 Inkrementell Sekvensiell

Inkrementell Sekvensiell utviklingsmodell tilbyr utviklingsteamet nok fleksibilitet til å forandre på kursen hvis problemer dukker opp under utviklingen. Samtidig vil det være nok struktur på forhånd for at systemet skal bli godt dokumentert og arkitektur blir gjennomført. Inkrementell Sekvensiell utviklingsmodell gir også studentene mulighet til å utvide systemet noe hvis det viser seg at det er mer tid til utvikling etter at essensiell funksjonalitet er implementert.

4.2.4 Valg av metode og tilnærming

Utviklingsteamet har valg å gå for en inkrementell sekvensiell utviklingsmodell for dette prosjektet, da metoden tilbyr den fleksibiliteten som er nødvendig for gruppens egen sikkerhet. Metoden hjelper også med en struktur som oppfordrer til et helhetlig system og god dokumentasjon. Det vil også forekomme elementer av gjenbruksorientert systemutvikling, da det allerede eksisterer et system som kan utføre store deler av oppdragsgivers ønsker. Gjenbruksorientert utvikling vil først og fremst brukes for læring og det vil være oppfordringer innad i gruppen til å utvikle ny kode framfor å bruke gammel kode direkte.

5 Organisering av kvalitetssikring

Denne kvalitetssikringplanen inneholder standarder og aktiviteter som vil bli gjennomført, i tillegg til risikoanalyse med løsninger for Ambulansesimulatoren på NTNU i Gjøvik.

5.1 Formål

Formålet med denne planen er å sette standarder og aktiviteter for gruppemedlemmer i prosjektet for å fremme god kvalitet og vedlikeholdbarhet av kildekoden, dokumentasjonen og sluttproduktet. Kvalitet og vedlikeholdbarhet er spesielt viktig, da det er en ansatt utenfor dette prosjektet ved NTNU som vil ha ansvar for fremtidig vedlikehold.

- Standarder: Kodestandarder og dokumentasjonsstandarder vil bli gitt for at kildekonden skal være konsistent.
- Planlegging: Nøye planlegging fremmer bedre kvalitet av arkitektur, slik at produktet blir mer vedlikeholdbart.
- Code-Review: Code-Review er et annet middel for å sikre kodestandarer holdes.

Disse punktene er forklart i mer detalj nedenfor.

5.2 Standarder

C++ Kodestandard

Lefticus C++ best practices[4] vil bli brukt som standard for skriving av C++ kode.

Dokumentasjon

Verktøyet Doxygen[5] vil bli brukt både for å dokumentere store deler av koden og for å automatisk generere dokumentasjon. Kommenteringen av koden vil følge guiden gitt i Doxygen.

Utvikler kommentarer

Det skal bli brukt kommentarer som er ment for utviklere i kildekoden. Denne skal ha formatet `//` for enkel linje og ha `/*` og `*/` på sine egne linjer for multilinjekommentarer. Disse kommentarene vil ikke bli en del av den genererte dokumentasjonen. Formålet er å forklare hvordan koden fungerer i detalj for utviklere.

[4] `https://github.com/lefticus/cppbestpractices/blob/master/03-Style.md`
[5] `https://developer.lsst.io/cpp/api-docs.html#cpp-doxygen-notes`

Conventional Commits

Commit messages skal være konvensjonelle og følge formen til Conventional Commits[6].

5.3 Konfigurasjonsstyring

Bitbucket[7] vil bli brukt for konfigurasjonsstyring.

5.4 Code Review

Code Review vil bli benyttet i stor grad. Commits skal aldri bli merget til master før den har gått gjennom code review. Når man setter reviewers er det krav om at minst én annen person skal godkjenne koden før den kan gå videre i prosedyren.

5.5 Planlegging

Før utvikling av en separabel modul skal følgende være på plass:

- Det skal diskuteres med oppdragsgiver og lage kravspesifikasjoner og use-case, dersom det ikke er på plass alerede.
- Design og arkitektur skal lages for modulen.
- Forslag til løsning skal diskuteres innad i gruppen.
- Gruppen må kollektivt godkjenne løsningen.

En separabel modul i dette prosjektet er definert som en modul som etter overordnet planlegging viser seg å kunne lages separat fra alle andre moduler, gitt en grunnarkitektur.

5.6 Testing

Testingrammeverket "Microsoft Unit Testing Framework for C++" [8] vil bli brukt i dette prosjektet. Siden dette er integrert i visual studio blir testingen enkel og oversiktlig.

I dette prosjektet vil følgende tester bli benyttet:

Unit Testing

Unit tester skal skrives for alle klasser og alle isolerte funksjoner i og utenfor en klasse. Isolerte funksjoner vil si funksjoner som har input og output som kun tar i bruk simple types (int, float, char, string og lignende) eller varialber som ikke refererer til andre klasser eller complex types. Alle unit-tester skal organiseres i namespacet "TestUnitTests". En klasse har organisert sine unit-tester i en egen klasse, hvor det er en unit test per funksjon i denne klassen. En testklasse har ingen

[6] https://www.conventionalcommits.org/en/v1.0.0-beta.2/

[7] `https://bitbucket.org/`

[8] `https://docs.microsoft.com/en-us/visualstudio/test/how-to-use-microsoft-test-framework-for-cpp?view=vs-2019`

egendefinerte attributter. Alle funksjoner skal starte med å lage en ny instans av klassen som blir brukt gjennom testen.

Integrasjonstesting

I dette prosjektet er Integrasjonstestingen en del av unit-testingen, men er organisert i et eget namespace kalt "TestIntegrationTests". Når moduler integreres, pleier det å være funksjoner i og utenfor klasser som binder modulene sammen ved å eksempelvis ta input fra hverandre. For alle slike moduler på tvers av klasser og funksjoner skal det skrives unit-tester. For at en modul skal være ferdig integrert, må alle disse testene bestås.

Acceptance Testing

Acceptance testingen vil fungere som en black box testing hvor resultatet av systemet skal testes ut. Dette vil hovedsaklig skje i brukergrensesnitettet. I acceptance tester skal det være to typer tester. Den ene skal være automatisk og påse at det ikke skjer feil. Disse testene skal organiseres i namespacet "TestAcceptanceTests". Den andre er at oppdragsgiver tester programmet, og gir tilbakemeldinger.

5.7 Arbeidsflyt

Arbeidsflyten oppsummert:

- Planlegging gjennomføres og løsninger diskuteres. Se Planlegging.
- Ny branch lages før det begynnes på en ny modul, eller ny funksjonalitet i en modul.
- Koden skrives etter gitt standard. Kommentering skal skje underveis.
- Det skrives unit-tester. Disse skal også kommenteres.
- Resterende dokumentasjon skrives, dersom noe mangler.
- Koden blir pushet til repositoriet.
- Det blir laget en ny merge request, hvor alle relevante personer blir satt som reviewer.
- Hver reviewer går inn og godkjenner eller avviser merge requesten. Hvis den blir avvist skal det forklares i detalj hvorfor, gjennom kommentarer i Bitbucket, og utvikler må rette opp koden og gjenta code review på nytt.
- Når koden er blitt godkjent av alle reviewers, kan den merges til relevant branch.

5.8 Risikoanalyse

5.8.1 Kategorisering av risikoer

	Ubetydelig 1	*Mindre alvorlig* 2	*Alvorlig* 3	*Katastrofalt* 4
Lite sannsynlig 1	**1**	**2**	**3**	**4**
Sannsynlig 2	**2**	**4**	**6**	**8**
Meget sannsynlig 3	**3**	**6**	**9**	**12**
Svært sannsynlig 4	**4**	**8**	**12**	**16**

5.9 Risikoanalyse Løsninger

Lite prosjekt

Som nevnt over, er prosjektet i utgangspunktet relativt lite. Dette kan gå ut over sluttkarakter. Dette kan løses ved å legge til funksjonalitet der det er muligheter for det. Alternative funksjonaliteter er allerede diskutert med oppdragsgiver, som har vist interesse.

Langvarig sykdom (Covid-19)

En stor utfording ved dette er at nøkkelpersoner blir syke og ikke kan gjøre noe produktivt i flere uker fremover. Dette løses ved bruk av code review for å hindre at bare en person har muligheten til å utvikle videre på en gitt modul.

Forsinkelser

Forsinkelser vil bli løst ved at gruppemedlemmene må jobbe på overtid.

Tap av Kodebase

Tap av kodebase ansees som lite sannsynlig, da kildekoden ligger i skyen på en anerkjent platform (Bitbucket) og på minst fire forskjellige datamaskiner. Det forutsetter at commits gjøres ofte, for ikke lokale endringer skal kunne gå tapt. Derfor er hyppige commits et tiltakt vi vil ta i bruk for å hindre tap av lokal kode.

Manglende kompetanse

Bacheloroppgaven har gitte minimumskrav for å få en godkjent vurdering av arbeidet. Om det er mangel på kompetanse skal det fortsatt være mulig å oppnå minimumskravet. Minimumskravet er ikke så stort, og inneholder kun akkurat

de kravspesifikasjonene oppdragsgiver har gitt. Sannsynligheten for å ikke få til minimumkravet anses for å være ubetydelig lite.

Sikkerssløyfen trigger ikke

Dette løses ved at det alltid skal være en person i kontrollrommet, under kjøring av Ambulansesimulatoren, slik at det i verste tilfelle skal gå an trykke nødstopp fra styringsrommet. Det eksisterende systemet for nødstopp har fungert siden det siste utviklingsprosjektet av ambulansesimulatoren. Det ansees som å være et trygt nødstoppsystem.

Problemer med leveranse av nytt utstyr

Det er en risiko at utstyret som er bestilt kommer sent i prosjektet. Dette håndteres ved å bruke eksisterende utstyr, da det allerede er et eldre system på plass.

6 Plan for Gjennomføring

6.1 Gantt-skjema

Gantt-skjema kan sees i figur 1

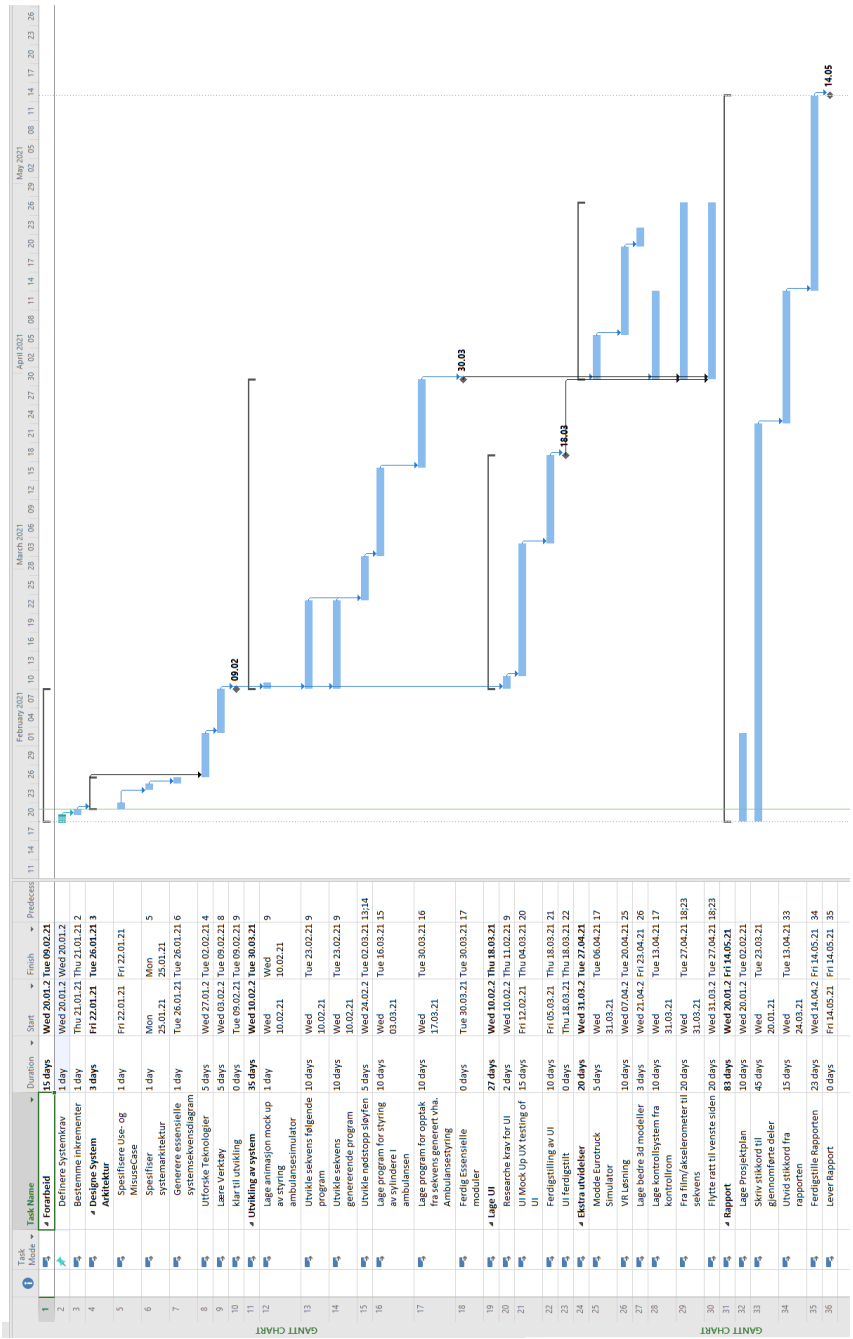
6.1.1 Utdyping av av Gantt-diagram

Gjennom prosjekt vil det forekomme en periode med parallelt arbeid på to inkrementer. Valget om å jobbe på to inkrementer samtidig bygger på at medlemmene av gruppen har forskjellig kunnskaper, og vil derfor utvikle mer effektivt hvis arbeid splittes opp. Inkrementene kan også utvikles i parallell da de er stort sett separable, med unntak av behov for kommunikasjon mellom systemene. Dette vil tas høyde for under utviklingen av modulene.

Hovedblokkene i utviklingen er styringssystemet for simulatoren og User Interface som sluttbruker skal kunne bruke for all nødvendig styring av ambulansesimulatoren. Etter utvikling av de essensielle funksjonalitetene til systemet er ferdigstilt er det en del flere funksjonaliteter oppdragsgiver har vist interesse for. Disse funksjonalitetene er lagt til som en siste blokk på slutten av utviklingsperioden. Det vil settes av et møte med arbeidsgiver hvis det viser seg at utviklingsteamet har tid til å utvikle noen av disse funksjonalitetene, slik at oppdragsgiver kan gi en prioritert rekkefølge.

Progresjon av utvikling vil måles gjennom milepæler ved forskjellige punkter i utviklingsprosessen. Det holdes også åpen kommunikasjon mellom utviklingsteamet over Discord. Gruppereglene vektlegger at forsinkelser fra gruppemedlemmer skal meddeles for å kunne legge om på planene. Det forventes dermed at uventede forsinkelser skal tas opp tidlig i utviklingen av inkrementer, som gjør det lettere å komme med en god løsning.

Det vil også holdes møter minst to ganger i uken for å bli enige om status og veien videre. Oppgavebeskrivelsen er relativt klar på hva som må utvikles, det



Figur 1: Gantt-Diagram Laget med MS Project

kreves derfor få avsatte tidspunkter for beslutningspunkter før essensielle funksjonaliteter er på plass i systemet. Det vil heller legges av tid til slike beslutningspunkter når utviklingen beveger seg mot ekstra funksjonaliteter som ønskes av arbeidsgiver. Det er også åpent for alle i gruppen å innkalle til møter hvis behov for oppklaring av oppgaver er nødvendig.

Forarbeid

Denne delen av utviklingsfasen gjennomgår forarbeid for videre utvikling. Forarbeid skal i utviklingsteamet mulighet til å søke ut alternative teknologier som burde testes videre for bruk i systemet.

Utvikling av systemet

Utvikling av systemet inneholder implementasjon av funksjonalitet som skal styrer ambulansesimulatoren. Utviklingen av disse funksjonalitetene skal kunne styres gjennom CLI slik at det senere kan kobles opp mot en UI. Utviklingen av hvert inkrement under denne delen inkluderer: utvikling av kode, generere tester tilhørende koden og utvikling av dokumentasjon for koden.

Utvikling av UI

For utviklingen av UI har vi bestemt oss for å gå frem med fokus UX-design. Kravet: *et godt og forståelig brukergrensesnitt*, er ett vagt og subjektivt krav, som tilsier at det ikke finnes noen fasit svar. Det er her vi mener UX design er en smart prosess for oss å bruke. UX design innebærer at vi kommer frem med kladder og prototyper som forskjellige brukere får leke seg frem med, og det observeres/testes hvordan brukerne opplever prototypen. Etter hver omgang med brukertester vil vi forbedre prototypen og begynne en ny runde med tester til vi har et produkt vi og oppdragsgiver er fornøyd med. Med denne prosessen sikrer vi at flertallet som bruker dette brukergrensesnittet vil ha en intuitiv og brukervennlig opplevelse. Ettersom oppdragsgiver er personen som kommer til å bruke programmet mest, er det viktig at oppdragsgiver er med hver runde med tester, og har siste ordet hvis det er usikkerheter rundt det beste designet.

Rapport skrivning

Underveis mens utvikling av moduler utføres vil det også skrives korte beskrivelser av hva som blir gjort i rapporten. Disse beskrivelsene vil senere utfylles til den komplette rapporten som skal leveres 2021-05-20.

Fleksibilitet

Fra dagens plan vil det være en god del tid fra planlagt ferdigstilling av rapport til innleveringsfrist. Gruppen ser på dette tidsrommet som nødvendig for å ha fleksibilitet til å gjennomføre uforutsett oppgaver eller hvis arbeidsmengde på oppgaver var større enn forventet.

6.2 Milepæler, beslutningspunkter og statusrapporter

Prosjektet inkluderer 4 milepæler. Den første milepælen er satt til 2021-02-09 og vil er avsatt som planlagt dato for ferdigstilling av forarbeid. Ved denne milepælen skal utviklingsgruppen være klar til å begynne utvikling av systemet. Etter denne milepælen vil utviklingen deles i to parallelle utviklingsprosesser. En av disse inkrementene er avsatt for utviklingen av systemet som styrer ambulansesimulatoren. Dette inkrementet har en milepæl 2021-03-30 hvor utvikling skal ansees som ferdig. Det andre parallelle inkrementet skal utvikle en UI for brukere av ambulansesimulatoren. Utviklingen av denne funksjonaliteten skal være ferdigstilt til 2021-03-18 og markeres med en milepæl.

Parallelt med utviklingen av systemet, står medlemmene av bachelorgruppen også ansvarlig for å skrive om det som utvikles i bachelorrapport. Det estimeres at hovedarbeid relatert til skriving av rapport er ferdig til 2021-05-14, hvor en milepæl for levering av rapport er lagt inn. Denne milepælen er noe fleksibel for å gi tid til rettinger av rapporten.

Det er lagt inn få beslutningspunkter i prosjektet, da det er tydelige systemkrav fra oppdragsgiver. Etter essensiell funksjonalitet er ferdigstilt, vil det settes opp møte med oppdragsgiver for å diskutere ytterligere funksjonaliteter nyttig for systemet. Beslutningspunktet vil komme etter milepælene for utvikling av systemet (2021-03-30) og utvikling av UI (2021-03-30). Møtet vil dermed bli satt opp etter 2021-03-30 hvis ingen forsinkelser i utviklingen oppstår. Det vil også være et behov for tidlige beslutningspunkter for å bestemme systemarkitektur og utviklingsrammeverk som brukes i faget. Systemarkitektur og utviklingsrammeverk skal være bestemt innen 2021-02-09 og samfaller med milepælen for ferdigstilling av forarbeid.

Gjennom prosjektet vil det være behov for å gi statusrapporter til veileder om foreløpig progresjon. Disse statusrapportene skal leveres circa. 15. februar. 1. April og 1. Mai. Disse rapportene vil følge en mal gitt av veileder.

E Arbeidskontrakt

NTNU

Norges teknisk-naturvitenskapelige universitet

Vår dato Vår referanse

Prosjektavtale

mellom NTNU Fakultet for informasjonsteknologi og elektroteknikk (IE) på Gjøvik (utdanningsinstitusjon), og

Senter for simulering og pasientsikkerhet
v/ Hans Martin Lilleby (oppdragsgiver), og

Aslak Tøn Håkon Muggerud
Isak Mikal Kallevig Donjeta Haziri
(student(er))

Avtalen angir avtalepartenes plikter vedrørende gjennomføring av prosjektet og rettigheter til anvendelse av de resultater som prosjektet frembringer:

1. Studenten(e) skal gjennomføre prosjektet i perioden fra 11.01.21 til 20.05.21.

   Studentene skal i denne perioden følge en oppsatt fremdriftsplan der NTNU IE på Gjøvik yter veiledning. Oppdragsgiver yter avtalt prosjektbistand til fastsatte tider. Oppdragsgiver stiller til rådighet kunnskap og materiale som er nødvendig for å få gjennomført prosjektet. Det forutsettes at de gitte problemstillinger det arbeides med er aktuelle og på et nivå tilpasset studentenes faglige kunnskaper. Oppdragsgiver plikter på forespørsel fra NTNU å gi en vurdering av prosjektet vederlagsfritt.

2. Kostnadene ved gjennomføringen av prosjektet dekkes på følgende måte:
   - Oppdragsgiver dekker selv gjennomføring av prosjektet når det gjelder f.eks. materiell, telefon, reiser og nødvendig overnatting på steder langt fra NTNU i Gjøvik. Studentene dekker utgifter for ferdigstillelse av prosjektmateriell.
   - Eiendomsretten til eventuell prototyp tilfaller den som har betalt komponenter og materiell mv. som er brukt til prototypen. Dersom det er nødvendig med større og/eller spesielle investeringer for å få gjennomført prosjektet, må det gjøres en egen avtale mellom partene om eventuell kostnadsfordeling og eiendomsrett.

3. NTNU IE på Gjøvik står ikke som garantist for at det oppdragsgiver har bestilt fungerer etter hensikten, ei heller at prosjektet blir fullført. Prosjektet må anses som en eksamensrelatert oppgave som blir bedømt av intern og ekstern sensor. Likevel er det en forpliktelse for utøverne av prosjektet å fullføre dette til avtalte spesifikasjoner, funksjonsnivå og tider.

12. Deltakende personer ved prosjektgjennomføringen:

NTNUs veileder (navn): Frode Haug

Oppdragsgivers kontaktperson (navn): Hans Martin Lillebø

Student(er) (signatur): Aslak Ton dato 18.01.21

Håkon Muggerud dato 19.01.21

Isar Mikal Kalberg dato 20.01.21

[signature] dato 21.01.21

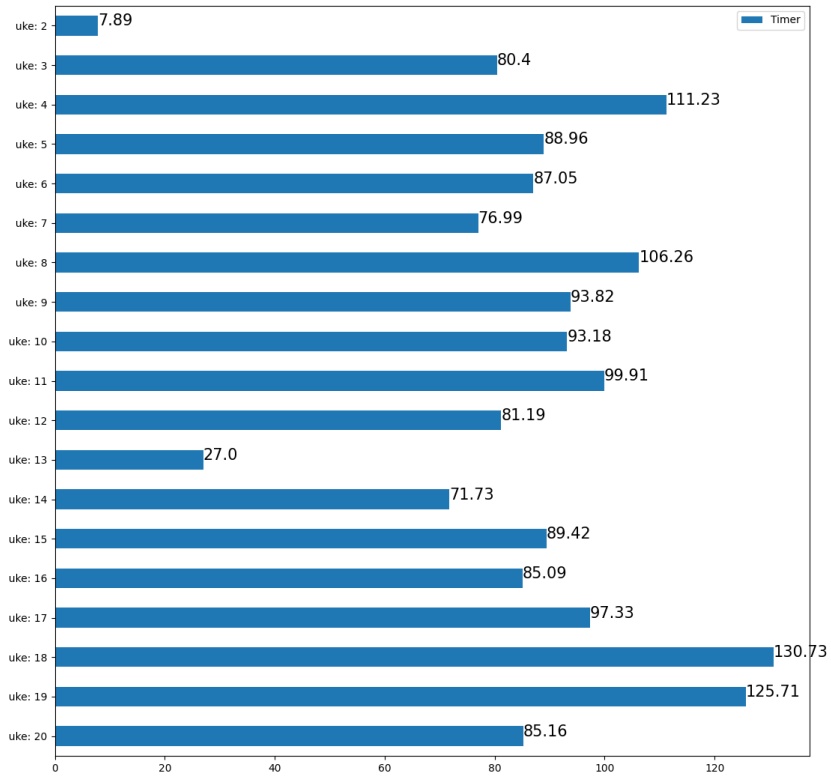
Oppdragsgiver (signatur): [signature] dato 27/1-21

Signert avtale leveres digitalt i Blackboard, rom for bacheloroppgaven.
Godkjennes digitalt av instituttleder/faggruppeleder.

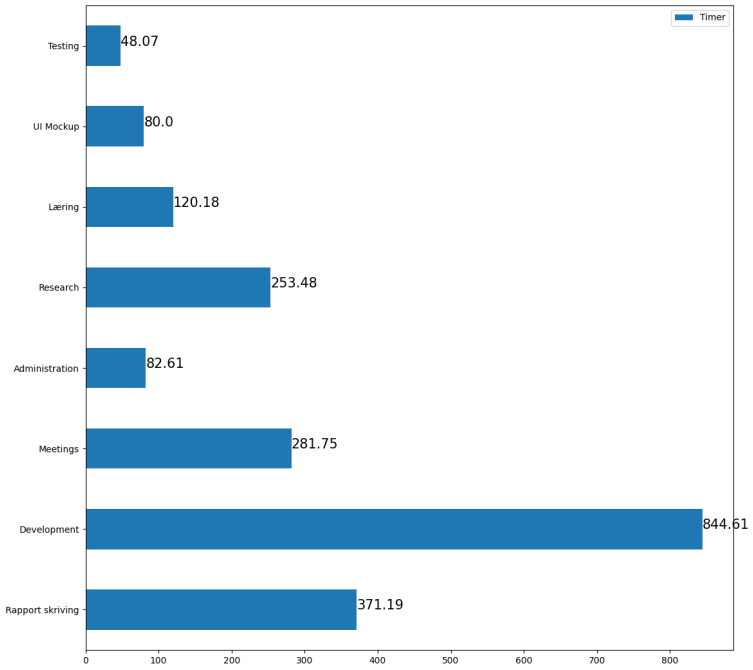
Om papirversjon med signatur er ønskelig, må papirversjon leveres til instituttet i tillegg.
Plass for evt sign:

Instituttleder/faggruppeleder (signatur): ______________________ dato ________

F Arbeidsdokumentasjon



Figur F.1: Tid fordelt per uke



Figur F.2: Total tid brukt, fordelt på kategorier

G Referater

G.1 Møte 19.01.21

- I praksis kan det gjennomføres digital signatur
- Forskningskisse != Prosjektplan
- Prosjektavtale trenger ikke å legges til i prosjektplan
- Gannt-diagram legges ved i prosjektplan
- Kan vedlikeholdbar kode ca. like godt som Frode H.
  - God dokumentasjon
  - Gode kommentarer
  - Gode tester?
- Design en god og ryddig løsning
  - Del opp i moduler
  - Designe gode beskrivelser
- Hva hva hva hva som skal lages
- Mal for kravspec.
  - Titt på lynkurs 1
  - Titt på ekstra: lynkurs 1 - bra rapporter
  - Det som skal lages, skal være målbart
- Vente med VR til vi ser målet
- Kvalitetssikring: Sjekk lynkurs 1
- Frode H. Ser ikke noe problem med å bruke bachelorrapporten i framtidig jobbsøk
- Prosjektmål
  - effektmål
  - resultatmål
  - læringsmål
- Rammer
  - praktisk
  - teknisk

  - andre føringer

- Ambulansesimulator er et verktøy for læring
- avgrenset til å simulere

G.2 Møte Frode 2021-01-26

- Norsk vs Engelsk? (hva vi eller oppdragsgiver foretreker)
- Svarer på språk litt selv. Spiller ikke noen rolle hvis oppdragsgiver ikke har noen føringer.
- Stille spørsmål til Hans Martin
- Hvis det skal publiseres, så er det kanskje mer verdt å skrive på engelsk.
- Hvis koden skal bli noe som helst public, så burde kommentarer være på engelsk
- 13.01 som første dato (møte med Fode)
- Ta med domenemodell. Glad i visualisering (viktig for kravspec)
- Hadde gruppe som en ble kastet ut. Det som er produsert til da er felles. Alt etter er egenutviklet. Alle har mulighet til å stå.
- Definere større brudd og gjentatte brudd (annet gangs større brudd, tredjegangs større brudd taes opp med veileder)

G.3 Møte Frode 2021-02-02

- når oppdragsgiver skal tjene noe, er det tid, penger, ressurser, konkuransefortrin
  - ressurser kan være alt fra matrialer til personer.
  - Tid: Sparer tid på vedlikehold
  - Penger: sparer penger fordi sparer tid på vedlikehold
  - Ressurser: færre folk som må jobbe på det
- Læringsmål
  - Lære å programere opp mot niDAC-mx
  - Prosjektledelse
  - Kontakt med oppdragsgiver
  - Organisering
  - literatursøk

- Hvordan gule setningen burde bli. sitere frodes hovedfagsoppgave
- Hva vil det si konkret å simulere.
- Prøv å si litt mer om hva simulering er.
- Ny seksjon med dagens løsning, så oppgavebeskrivelsen
- Flere møter? Kanskje spesifiser at vi har møte etter møte med veileder.
- Først kommer stikkord, så kommer avsnitt.
- avsnitt 5.1: slett stikkord. eller si "alt dette vil vi se på i de neste seksjonene": Isac
- Flytte over risiko til å følge mal som bildet brukte. Donjetë
- 180 grader rundt. Vanlige formatet for bøker.
- Hvis vi ikke tar avgjørelsen innen det tidspunktet blir prosjektet forsinket.
- Bachelorraport: lynkurs med frode om hvordan å skrive rapport
- Fokus på Gantt-Diagrammet
- Satusrapport: ca. 15 februar, ca. 1. april. ca. 1. mai. Sendes til Frode.
- Aslak tar seksjon 2. Håkon tar seksjon 1
- Læringsmål
  - Kommunikasjon med oppdragsgiver
  - Robotikk, elektronikk, hydraulikk og signalbehandling
  - DAQMX programmering, programmering av Compact DAQ
  - Modding av spill, bruken av api-er
  - Grafisk brukergrensesnitt, QT i C++
  - Praktisk bruk av kodestandarder.
  - Bedre bruk av bitbucket og conventional commits og version kontroll.
  - UX design
  - Systemutviklingsprosess i praksis
  - Testrammeverket VS C++
- Beslutningspunkter
  - Valg av GUI rammeverk. 2021-02-09
  - Valg av systemarkitektur. 2021-02-09

G.4 Møte 2021-02-09

- Ønsker en kravspec
- til noen nøkkelpunkter
- 3 ila. bacheloroppgaven
- Skriv litt til slutt om erfaringer og analyser og hva vi lykkes med og ikke.
- Vi brukte noe annet selv om vi sa noe i prosjektplan. Mer nevnes senere.
- Kunne vi bruke UX design med Donjete. Donjete sender mail om når. Helst mellom 0830 og 0900 tirsdager.
- Statusrapport kontakt med oppdragsgiver og veileder.

G.5 Møte Frode 2021-02-16

- Donjete laster opp HTML fil, hun forklarer underveis hvordan å gå framover
- stille noen spørsmål.
- Sitter på bakrommet
- Kjør sekvens
- Nytt grensesnitt, men mange elementer fra gammelt grensesnitt.
- Donjete ber Frode finne spesifikke elementer i UI
  - Styring med mus
  - Spill av lydfil
  - Lag sekvens med Ratt
- Frode kan ikke tenke på noe så langt som mangler
- Tekst er bra nok
- mangler kryss, minimer, maksimer muligens. Strider med vanlig windows UI.

G.6 Møte Frode 2021-02-23

- Spør litt om hvordan gruppen funker.
- Tror ikke karantene er nødvendig lengre tror Frode.
- Finner på Innsiden hvilke holdinger man burde ta ift. Korona.
- Ikke karantene, men anbefales å ha lav terskel for å teste seg.
- Info hentes fra innlegg på innsiden 29. Januar: Reiseråd om Østlandet oppheves, hjemmekontorordningen forlenges..."

G.7 Møte Frode 2021-04-20

- Hva var tanken bak grafisk design? kan vi få en oppklaring?
  - Kapitel 3 er om enten grafisk eller teknisk design. Om vi forklarer programet eller hviser grafisk om hvordan programmet så ut til slutt (m. begrunnesle om valg osv.)
  - Om grafisk blir kap. 3 er som regel kap. 4 teknisk forklaring.
- Hvordan/hvor skrive om hva vi har lært?
  - Skal på slutten, som regel et kapittel, men mer objectivt og sakling.
  - Refleksjons notat er mer subjectivt, som er mer spesifikke til hvert enkelt person
- Hvordan burde vi sette inn refleksjoner og argumentering av valg vi gjorde gjennom utviklingen?
  - Ikke uvanlig med noen refleksjoner/argumentasjoner litt underveis, og deretter henvise til tidligere i refleksjoner/argumentasjoene på slutten. Kan ha alt på slutten om man ønkser.
  - Pass på at ha det klart hvordan vi har løst det teknisk, og ikke bare refleksjoner/argumentasjoner
- Installasjon kapittel unødvendig (for oss)?
  - Om det bare er teknisk rett frem oppskrift holder det som vedlegg, mer avansert er det greit å ha det som tilegg
  - Vi har for lite til å ha det som eget kappitel. Mer å legge det til i andre steder.
- Gjerne sende Frode rapport 2-3 ganger når vi føler det er greit.
- Aslak sender første utkast løpet av tirsdag april 20.

G.8 Møte Frode 2021

- Holder at feltene er der i abstract
- Avgresning: Problemområde
- Rammer: Det som står under rammer(tidsmessig (jfr. fremdriftsplan), fysisk, praktisk, arbeidsmetoder, teknisk, prosjektorganisering)
- Klikkbart design: skal legges til på slutten? tja, litt langt. Kanskje legge til en kort versjon

G.9 Møte Frode 2021-05-11

- generere forside
- Ligger utsagn på BB
- Innsida skrive og levere bacheloroppgave
- Ferdigstille oppgaven
- Genererer automatisk forside
- Får spørsmål om publisering i NTNU Open under levering i inspera
- navformat: etternavn - oppgavens kortnavn .pdf
- Demonstrere software live
-

G.10 Møte 2021-01-26

- Lagde en systemarkitektur
- Unit testing gjennom Visual Studio

G.11 Møte 19.01.21

- Kjernetid fra 0900 til 1500 mandag til fredag
- Innenfor kjernetid skal alle på gruppen i det minste gi respons for sett beskjed, om de ikke har tid til å løse problemet
- Etter alle i gruppen har underskrift, sender Aslak til Hans Martin
- Isac Skriver under i løpet av i dag

G.12 Møte Gruppen 2021-01-20

- Oppgaver
  - Rapport må utvides
  - Gantt-diagram
  - LabView må testes
  - Finne andre måter å styre sylinderene på
- Kravspec
  - Lage UI
    * Vise nåtidsposisjon til ambulansen
    * Informere når nødstopp er aktiv
    * Vise hvor langt ut i simulasjonen man er
  - Lage sekvensgenerene program
  - Lage sekvensfølgende program
  - Lage program som blir styrt av ambulansens ratt og pedaler
  - Lage program som tar opp sekvens som blir generert av ambulasens ratt og pedaler
  - Legge til støtte for VR
  - Styre simulator fra kontrollrommet
  - Nødstopp sløyfen må på plass
- Ovrige oppgaver
  - Lage prosjektavtale
  - Lære LabView
  - Utforske Windows Vs. Linux
  - Utforske andre alternativer til LabView
- Skal rapporten være med i Gantt-Diagram? Frode
- Styring av simulator med mus
- Keyframes legges inn ved trykking på en tast/museklikk
- Flere språk, spør Hans Martin
- Oppgaver Prosjektplan
  - Plan for Gjennomføring - Aslak
  - Kvalitetssikring - Isac

  - Valg av utviklingsmodell - Aslak
  - Prosjektorganisering - Donjetë
  - Omfang - Håkon
  - Mål og rammer - Aslak

- Prosjektplan leveres innen søndag kl 23.59

G.13 Møte 2021-01-25

- nidaqmx driveren for styring
- Fortsatt Eurotruck simulator?
  - Litt bekymringer rundt Steam og støtte
- Ser på egenhånd om hvordan MVC skal utformes tar felles møte i morgen kl 13.00 for å diskutere
- Aslak: generer PDF dokument for å underskrive grupperegler. Grupperegler skrives under og legges ved.
- UseCase generert, MisUseCase er noe unødvendig, men har blitt laget også.

G.14 Møte gruppen 2021-01-28

- Møte gjelder finskrivning av prosjektplan
- Flytter det som var i effektmål over til 2.3 oppgavebeskrivelse (mye er der allerede)
- Donjete fikser risikoanalyse
- Alle jobber med sine deler. Når alle er ferdig gjør vi et felles review. Må være ferdig til kl 15.00

G.15 Møte Gruppen 2021-02-02

- Tweaking: Vet ikke om det er veldig relevant å kunne gjøres fra bilen.

G.16 Møte gruppen 2021-02-16

- Ingen problem med hvordan møter organiseres.
- Noe problemer med at Donjete ikke er med på utvikling på campus. Fikk ikke vite at lydfil skulle være med før senere.

G.17 Møte Gruppen 2021-02-23

- Sette opp Dev enviroment:
  - Clone Project Repository.
  - wxWidgets
    * Last ned wxWidgets zip
    * Dekomprimer til et passende sted. Gjerne nær C:
    * Legg til path til wxWidgets i system variables. Kall den WXWIN
    * åpne wx_vc16.sln og kjør batch build på alle prosjekter
  - niDAQmx
    * Last ned niDAQmx 20.1 driver
    * ha på disable windows quick startup.
    * Kjør default installasjon.

G.18 Møte Gruppen 2021-03-02

- Hva er status
  - Håkon ikke gjort så mye, vil ta opp litt veien framover
  - Don: Prøver seg litt fram med greier i wxWidgets.
- Håkon:
  - Vi trenger mainController
  - Hvordan kommunisere mellom view og main controller.
    * Litt usikkerhet enda. Trolig noen eventer fra UI som åpner og lukker tråder.
  - UI - Ikke funksjonalitet til å sette volt flere steder sammitig (starte free drive og følg sekvens samtidig)
  - Lagre sekvens. UI må prate med flere objekter.
  - Håkon lager mainController objekt
  - Lage en dev Branch
  - Isac lager en dev branch
  - Aslak Kjører code review på Isac PR
  - Don kjører code review på Isac PR

G.19 Møte gruppen 2021-03-09

- har funnet ut av det meste med wxWidgets
- Don har funnet farger som passer for fargeblindhet
- Ligger foreløpig en del commits bak dev branch
- Ting i wxWidgets trenger openGL, dette skal kanskje Isac fikse?
- Isac fokuserer på OpenGL innenfor UI
- Alle burde få sjekke om man kan kjøre branchen
- Vil ha oversikt over hele solution
- Bruker google docs for noe review som faller igjennom PR's
- Don tar code review
- Isac lager ny PR for oppdatert VFC og GameMode?
- Aslak lager flere kommentarer wheelController
- Isac fikser Error/Debug message boxes
- Håkon finner ut hvordan å kjøre en error event

G.20 Møte Gruppen 2021-03-16

- Glemt PR - Don og Isac fikser nå
- Noen problemer med å få wxWidgets til å skrive ting riktig. Don ønsker noe hjelp - Aslak eller Håkon
- events i wxWidgets - Håkon fått til noen ting, noen errors som er vanskelig å debugge. - Aslak titter på det med Håkon senere i dag
- vente til etter merge med master for å forandre tilbake til makro i main.cpp
- Aslak Skriver en statusrapport - folket ser igjennom

G.21 Møte Gruppen 2021-03-19

- Ikke input2volt klasse
- Hver controller klasse som henter/skriver data til "I/O - device"har egen metode for å lese data getData().
- Hver kontroller klasse som henter/skriver data til "I/O - device"har egen metode for getVolt().

- Hver kontroller klasse som henter/skriver data til "I/O - device"har egen metode for getDisplayInfo().
- Hver kontroller klasse som henter/skriver data til "I/O - device"har egen metode for updateIODevice().
- Disse 3 metodene arves fra en parent klasse
- toolz.h/.cpp VFC.h/.cpp og VirtualCar.h/.cpp flyttes i eget prosjekt kalt CoreToolz.vcxproj
- WheelController får et member variable av typen VirtualCar.
- WheelController får en member function kalt generateWheelForces()
- Håkon fixer

G.22 Møte gruppen 2021-04-06

- Flytte DisplayInfo struct over i toolz.h/cpp
- Flytte TaskInfo struct over i toolz.h/cpp
- toolz.h/cpp har verktøy
- "constructz.h/cpp"har structer og klasser
- begge disse overnevnte filene flyttes over i eget prosjekt "CoreToolz.vcxproj"
- Ikke mulig å inkludere libs for OpenGL, må legges inn i solution
- Håkon er straks ferdig med input controller
- Aslak Dropper ut av koding, skriver prosjektoppgave. Blir med på parprog hvis trengs.
- Errorhandeling, Håkon fikser etter input controller. Feilkoder i constructz enum.
- bruker 2 draw funksjoner fordi vi var nødt for knapper, trenger OpenGL for polygons

G.23 Møte Gruppen 2021-04-13

- Hyppigere møter?
  - Tar opp ting etterhvert som det kommer opp
  - Trenger ikke flere oppsatte møter ila. uken
  - Folk sier ifra hvis de mener de trenger mer
- Instillinger

  - Dropdown meny for instilling

- Kanskje fjerne endre sekvens delen
  - Mye arbeid, lite nytte
  - Møte med Hans Martin - Aslak skriver mail
  - Vi finner ikke når endre sekvens kom med på agendaen
  - Vi vurderer å fjerne endre sekvens

- PR til dev torsdag. Ferdig UI?

- Håkon Popper en PR senere i dag, har error handling.

G.24 Møte Gruppen 2021-04-19

- niDAQ gir feilmeldinger hvis ikke koblet til.
  - Kan ikke sjekke om niDAQ ikke er koblet til.
  - Checkbox for om man er koblet til ambulansen.
  - Trenger vi niDAQ errors?
    * Generell niDAQ error kunne vært nyttig?
    * Ikke så viktig å ha errorhandlig tenker Isac, Håkon, Aslak?
    * Løsninger:
      · Ignorere error handling. Stemmer: 1
      · Kun error handling hvis systemet er i DEBUG modus. stemmer: 2
      · kan ha en checkbox som sier om man er koblet til niDAQ. stemmer: 4
    * Vi legger til checkbox i systemet som man kan bekrefte eller avkrefte om man er koblet til niDAQ.
    * Dette kommer ann på om HM godkjenner løsningen
  - Variable refresh rate nødvendig?
    * Nei. Folket mener det ikke er nødvendig.
    * Har testet systemet for 10Hz, ser ingen grunn for å ha det så fleksibelt
    * Lett å fikse på senere, for andre utviklere hvis nødvendig.
    * Fortsette med Hz i sekvensfilen? er ikke farlig at det er der
  - Aslak Foreslo baseFrame lagt inn i constructor.
    * Dette funker ikke grunnet bugs.
  - Aslak foreslår en parent klasse med med en BaseFrame og en metode setMainFrame(BaseFrame t_mainFrame); (low priority)
  - Merge to dev? Ja i kveld.

G.25 Møte Gruppen 2021-04-27

- Gjenværende på progging:
- AmbuGL, dokumentasjon - Isac
- AmbuGL, lefticus - Isac
- AmbuGL, bytte til å bruke CoreToolz.vcxproj (istedenfor egen impl. av core toolz) - Aslak
- AmbuController, SequenceController sørge for at program ikke krasjer på slutten av sekvens - Håkon
- AmbuView, oppdatere playSeq btn etter ferdig sekvens - Håkon
- AmbuView, skifte farge på graf - Don
- AmbuView, border på kameraer (L/R) - Isac og Don
- AmbuView, øke størrelse på notebooks - Don
- AmbuView, bakgrunnsfarge - Don
- AmbuView, bestem en minSize - Don
- AmbuView, niDAQ-toggle - Aslak
- AmbuView, Speedometer - Aslak og Don
- Etter folk opplever at de har kontroll på rapport
  - AmbuGL, hindre at flat ambulanse (lav prio) - Isac
  - AmbuView, graf viser voltinfo (lav prio) - Isac og Don (Hør først med Hans Martin før dette gjøres)
  - AmbuView, fikse bug ratt ikke koblet (lav prio) - Aslak
  - AmbuCont/Model, total timer (lav prio) - Aslak
  - AmbuView, disable"andre notebooks mens en kjører (mid prio) - Don
  - AmbuController, MouseController (mer enn 1 skjerm)(lav prio) - Håkon
  - AmbuController, spille av fra csv fil (lav prio) - Isac
  - AmbuController, implementere revers (lav prio) - Isac
  - AmbuGL, fjerne glfw (lav prio) - Isac
- Rapporten
- Aslak skriver litt random overalt om hva han føler for.
- Aslak anbefaler at andre gjør dette også, hvis de før det til
- Don har hovedansvaret for Grafisk Design
- IMRAD ish struktur under implementasjon. Isac har en plan

G.26 Møte gruppen 2021-05-04

- Nå er all funksjonalitet ferdigstilt. Trenger bare PR til dev
- Rapport:
  - Folk flest har latex compiler
  - Aslak skal hjelpe de som fortsatt mangler
  - Leser det de andre skriver.
  - Hvis man oppdager et avvik i tekstkvalitet så gjør man:
    * bruker (Navn) i en kommentar for å finne igjen suspekte greier.
    * tar opp problemet når rapporten er ferdig draft
    * kan også legge inn en @ hvis man mener det er noen som kan fikse
    * hvis en @ ikke kan fikses av den som er nevnt i @. bytter den personen @ til
  - Folk tar ansvar for 1 - 2 sections
- Clockify:
  - Oppdatere arbeid i April (i hvertfall, mer hvis dere har gjort mer)
- Statusrapport:
  - Aslak Fikser og sender til Frode.
- readme.md
  - Isac oppdaterer med GLEW greier

G.27 Møte Gruppen 2021-05-11

- Alle lager hver sin nye sys.Arch til i morgen
- vi tar felles møte klokken 1000 for sys.Arch
- Videre arbeid flytte til slutten av konklusjon
- Alle ser igjennom hele rapporten og kommenterer på hva de ser. Frist mandag.
- Beholder definisjoner av roller i introduksjon.

- Lesing og retting rapport:
  - Stavefeil: rett med en gang
  - Hvis dårlig skrevet: spør på discord om retting. Rett selv og gi forslag
  - Hvis ønsker å utfylle: si ifra hvor man fylte ut.
  - Hvis man ikke forstår: Ta det opp på discord, legg inn notat i problem-avsnittet (comment)

G.28 Møte Gruppen 2021-05-12

- Lager ny Sys.Arch
- Håkon pynter på sysArch
- Håkon lager Klassediagram også
- Folket ser igjennom klassediagram

G.29 Møte Gruppen 2021-04-27

- Gjenværende på progging:
- AmbuGL, dokumentasjon - Isac
- AmbuGL, lefticus - Isac
- AmbuGL, bytte til å bruke CoreToolz.vcxproj (istedenfor egen impl. av core toolz) - Aslak
- AmbuController, SequenceController sørge for at program ikke krasjer på slutten av sekvens - Håkon
- AmbuView, oppdatere playSeq btn etter ferdig sekvens - Håkon
- AmbuView, skifte farge på graf - Don
- AmbuView, border på kameraer (L/R) - Isac og Don
- AmbuView, øke størrelse på notebooks - Don
- AmbuView, bakgrunnsfarge - Don
- AmbuView, bestem en minSize - Don
- AmbuView, niDAQ-toggle - Aslak
- AmbuView, Speedometer - Aslak og Don
- Etter folk opplever at de har kontroll på rapport
  - AmbuGL, hindre at flat ambulanse (lav prio) - Isac

  - AmbuView, graf viser voltinfo (lav prio) - Isac og Don (Hør først med Hans Martin før dette gjøres)
  - AmbuView, fikse bug ratt ikke koblet (lav prio) - Aslak
  - AmbuCont/Model, total timer (lav prio) - Aslak
  - AmbuView, disable"andre notebooks mens en kjører (mid prio) - Don
  - AmbuController, MouseController (mer enn 1 skjerm)(lav prio) - Håkon
  - AmbuController, spille av fra csv fil (lav prio) - Isac
  - AmbuController, implementere revers (lav prio) - Isac
  - AmbuGL, fjerne glfw (lav prio) - Isac

- Rapporten
- Aslak skriver litt random overalt om hva han føler for.
- Aslak anbefaler at andre gjør dette også, hvis de før det til
- Don har hovedansvaret for Grafisk Design
- IMRAD ish struktur under implementasjon. Isac har en plan

G.30 Mote Hans Martin 21-01-19

- Trenger ikke noe adgangsstyring.
- Fri internett tilgang PC
- Budsjett
  - Ny PC
  - På sikt kunne få laget ratt og pedaler til kontroll.
  - Oppgradere også det eksisterende elektroniske ratt og pedaler.
- VR/AR med briller
  - Må trekke ny fysisk kabel i så fall
- Skal ligge lokalt på 2 forkjellige PC'er.
  - Kan utvide til protabelt til mange PC'er hvis mulig
- Laste opp fil
  - trenger ikke det
  - Lager nye sekvenser i starten
  - Når man får et nøkkelknippe trenger man ikke så mange.

  - Trenger ikke legge mest fokus på generering av nye sekvenser

- Lage sekvens bruker grensesnitt
  - Hadde en bil visuelt. Kan bruke musepeker og styre den.
  - Relativt åpent. Kan ha kreativ frihet
  - Viktigere å se den bak i fra.

- Hvordan simulatoren funker 2 modus:
  - Lager en sekvens på forhånd
  - Tar imot data fra ratt og pedaler og kjører det ut.
  - ønsker begge modusene.
  - modus 3 ikke så nødvendig.
  - Sitte med en mus og kjøre fra kontroll rommet. Har ikke funket bra.

- Simuleringen er på rundt 3 min typisk. 1-5min
- Sekvensene beskriver volt. Pål Erik vet litt mer om hvordan dette funker
- Sensorer
  - Sensor som vet om noen er i en faresone,
  - Nødstoppsløyfe, gjerne ha en lampe i programmet som sier om den er åpen eller lukket.
  - Dørene på ambulansen.
  - Nødstoppsløyfen: 2 knapper, alle dører, 2 lysgardiner"?
  - Får ikke pressisert hvor det er.
  - vi kan forske selv om vi kan løse å gi beskjed nøyaktig hvor sløyfen brytes

- Spør Pål Erik om det er noe sikkerhet rund å ødelegge hydraulikken.
- Pumpen vet når den er ute av sync. Spør Pål Erik
- Bruksområder:
  - Lage sekvenser - MÅ
  - Kjøre etter sekvenser - MÅ
  - Kjøre fritt simulasjon og hydraulikk følger - MÅ
  - bonus: styre hudraulikk med mus. (ser ikke dette som noe særlig bruksområde) - Kan
  - Må kunne nødstoppe, og indikere om nødstopp er aktiv. - MÅ

  - Intuitivt Grensesnitt - MÅ
  - Kunne programmet vist en modell av hvordan ambulansen ser ut under simulasjon (fint å ha men ikke nødvendig) - Kan
  - Ofte skal man kjøre i mørke, så ser ikke alltid hvordan ambulansen beveger seg. Blir for den som sitter i kontrollrommet - Kan
  - VR simulasjon - Kan
  - Tidsakse i simulering for kontrollrommet - Kan
  - Simulasjon stopper etter sekvens er ferdig, heller kjøre samme sekvens på nytt. - MÅ
  - Avbryte sekvens når som helst - MÅ
  - Skal ikke lage en sekvens sammtidig som sekvensen genereres. I dag lager man sekvens mens hydraulikken går opp og ned. Ser ikke noe poeng i det.
  - Kjøre med ratt inn i bilen for å generere en sekvens. - Kan

- Sitter alltid en i kontrollrommet.
- Standarer:
  - Ingen formening
  - Ikke noe krav om eller ikke viktig å følge NTNU grafisk standard
- Når har vi tilgang på rommet?
  - Har lite planlagt nå. Til høsten blir det en annen situasjon.
  - Hans eller Terje blir med når vi trenger det.
  - Hans bor 5 min unna, så er veldig fleksibel
- Hvilken periode er det
  - trolig mye koding februar og mars
  - trolig mindre i april og mai
- Forbedringer:
  - Bruk av musen til å løfte opp og ned sylindere
  - Modernisere det mest
  - Det gamle systemet har vært lappet på i flere omganger
  - Dårlig dokumentasjon
  - Starte litt på Scratch
  - Skal kunne bruke vår dokumentasjon for å videreutvikle

- Effektmål:
  - Skal ikke være hakkete når man generer et program
  - Leveres til 20 mai.
  - Skal kunne generere sekvenser
  - Kjøre sekvens
  - Lage sekvens etter ratt og pedaler
  - Kjøre simuliering styrt av ratt
- Ukentlige møter:
  - Planlegge for neste møte i slutten av forrige.
- Hans Martin mener vi har god kontroll på hva som skal legges ved.

G.31 Møte 2021-01-26

- muligheter for å bruke eksisterende ratt. Høre med elektro.
- Må ikke være koblet til nett.
- Sende mail til Pål Erik om det er greit å koble ting fra.
- Hans Martin skal ta en prat med Pål Erik om risikovurdering på utstyret, om det tåler forflyttning.
- Følger veien i spillet er kult og nyttig. Krasjing simulasjon er ikke så nødvendig.
- Hydraulikken motar signaler fra rattet i dag, ikke fra spillet.
- Hans Martin vet ikke om det er mulig å generere "usikkresekvenser.
- Spretter tilbake til nullpossisjon etter den har siget.
- Det er fjerning i bilen som demper utslaget.
- Hans Martin tviler på at hydraulikken kan ødelegge seg selv.
- Korte oppklaringer:
  - Studiene som skal bruke simulatoren: Bachelor i paramedisin.
  - Kilden i første bilde, kreditere til NTNU.
- Pause en simulasjon er nyttig, men ikke veldig viktig. Spesielt på en sekvens.
- Den mest brukte sekvensen kjører ganske rolig i starten og hopper mye på slutten.
- Demonstrasjon av systemet. Før 11 eller etter 13. Kl 0900 passer bra for alle.

G.32 Møte Hans Martin 2021-02-02

- Trenger ikke bekymre oss for kostnader
- Vet ikke hvor detaljert ETS er ift. kjøring i by. Fartsdumper veldig fint, Lyskryss, Rundkjøringer.
- Kan kanskje videreutvikle et annet prosjekt som vi kjøper. Kostnadden er kanskje et spørsmål.
- Jo mer realistisk, jo bedre er det.
- Kjøring er ikke fokus, det er mer viktig at det er bevegelse i bilen.

G.33 Møte Pål Erik 20.01.21

- begynte med en studentavtale
- Hvordan man kunne ta inn og sende ut signaler.
  - LabView var en del av et emne
  - Er et national Instruments produkt.
  - Om annen software kan brukes er Pål Erik.
  - Har hatt litt trøbbel med PC'er som blir for gammel.
  - Timing problematikk
  - Lurt å begynne helt fra scratch
  - Kunne vært fornuftig å ha API mot LabView
  - Annet kan være nyttig, Pål Erik vet ikke hva som funker
- Er sensorer som sier hvor de står. (-4 til +6 volt)
- Noe vi burde holde oss unna?
  - Nei, kan sette seg inn i det meste
  - En brukermanual
  - Dokumentasjon skal være med.
- Når labview starter, så sender den et nullsignal til sylindere.
- Volt tilsvarer possisjon.
- Uavhengig av om PC'en er på eller ikke vil Simulator nullstille seg.
- Hever seg til et Nullpunkt".
  - Vi må finne ut om den finner nullpunkt selv
- Ingen standarer fulgt.

- Sekvens systemet
  - Ferdig innebygde funksjoner som gjør jobben.
  - Lese fra fil og skrive til fil.
  - Lager og spiller av en fil i LabView
  - National Instrument på funksjonsblokken.
- Pitfalls:
  - 12 - 13 studenter
  - Var veldig lite dokumentasjon
  - Prosjektet kom litt på sparket
  - Noen innleid fra Intek.
  - Husker ikke særlig til pitfalls
- Kall inn Pål Erik han kommer hvis mulig
- Møte 12.00 til 12.30 tirsdag
- IO-boksen med embedded PC,
- Software som begynner med UI også kommer annet i bakgrunnen.
- Det er LabView på software.ntnu.no også link til learning platform under siden for labview
- National instruments kontrolleren heter: husker ikke i farten

G.34 Referat fra research i labben

- samplerate ca 100/sek for nødstoppsløyfen

H Akseptanse tester

Akseptanse tester

Test	Instruksjon	Resultat	Kommentar
Test Start Program	• Start programmet fra .exe fil		
Test Bygg Program	• Bygg programmet på nytt i Visual Studio Community		
Test koble fra hydraulikk	• Koble fra USB til DAQ'en • Velg en sekven • Trykk play knappen over Nødstoppindikator • Programmet bør Gi en feil • Gå til instillinger • Trykk på "Koble fra hydraulikk..." • Trykk på play knappen igjen • Volt grafen burde nå bevege seg • La "Koble fra hydraulikk" være aktivert for de neste testene		
Test Kjør Sekvens (sekvens fil)	• Trykk "Velg Sekvens..." • Åpne en sekvens fil med .sequence extension • La sekvensen kjøre ferdig • Sjekk at ingen feilmeldingsbokser dukker opp		
Test Kjør Sekvens (drag and drop)	• Finn en .sequence fil gjennom windows filsystem • Dra filen fra filsystemet inn i tekstboksen ved siden av "Velg Sekvens..." • Spill av sekvens filen. • Verifiser at volt grafen reagerer		
Test Kjør Sekvens fra .csv	• Finn en vilkårlig .csv fil med riktig format (genereres av physics toolbox appen) • Importer filen ved hjelp av "Velg Sekvens..." knappen • Verifiser at Volt Grafen beveger seg		
Test Avspill lydfil	• Trykk "Velg lydfil..." • Velg en vilkårlig .wav lydfil • Trykk play knappen under "Velg Lydfil..." knappen • Verifiser at lyden spilles av • Verifiser at lyden repiteres • Verifiser at lyden stopper når stopp knappen er trykket		

Test Avspill lydfil (drag and drop)	• Finn en vilkårlig .wav fil fra windows filsystem • Dra filen fra filsystemet inn i tekstboksen ved siden av "Velg Lydfil..." • Spill av lydfil • Verifiser at lydfil gjentas • Verifiser at lydfil stopper avspilling når "Stop" knappen trykkes		
Test Strying Mus	• Trykk på "Fri Styring" knappen • Åpne nedtrekksmenyen • Velg "Mus" • Trykk på Start • Verifiser at ambulansen beveger seg når musepeker beveger seg på skjermen		
Test styring Spill	• Trykk på "Fri Styring" knappen • Åpne nedtrekksmenyen • Velg "Spill" • Trykk på "Start" knapen • Verifiser at feilmelding dukker opp • Åpne Euro Truck Simulator 2 • Når ETS2 er åpen trykk på Start • Verifiser at applikasjonen reagerer når man kjører bil i spillet		
Test Opptak	• Trykk på "Lag sekvens" knappen • Åpne nedtrekksmenyen • Velg "Mus" • Trykk opptaksknappen • Verifiser at opptaksknappen bytter til en stoppknapp • Beveg på musen i noen sekunder • Trykk på stopp knappen • Lagre den nye sekvensen • Trykk "Fra Sekvens" • Åpne den nye sekvens filen • Verifiser at sekvensfilen er samme som akkuratt ble tatt opp		

Test	Instruksjon	Resultat	Kommentar
Test Nødstoppindikator	• Koble til DAQ igjen • Sikre at Logitech rattet er koblet til • Start programmet på nytt • Kontroller at nødstopp lyser rødt når nødstoppsløyfen er aktiv. • Kontroller at nødstopp lyser grønt når nødstoppsløyfen er inaktiv • Gjennomfør de neste testene uten å trykke på "Koble fra hydraulikk"		
Test Kjør Sekvens	• Gjenta første Test Kjør Sekvens • Verifiser at ingen feilmeldinger kommer		
Test Ratt	• Trykk "Fri Styring" • Velg Ratt under nedtrekksmenyen • Trykk "Start" Knappen • Verifiser at ingen feilmeldingsvindu dukker opp		

I UX Test Skjemaer

Runde: 1
Test Object: 1

☑ Er det greit vi lagrer følgende informasjon?

Bakgrunn: (utdanning/arbeid) Informatikk utdanning

Nærsynt ☐ Langsynt ☑

Fargeblindhet ☑ Dyslektiker ☐

Andre tilstander som kan gjøre det vanskeligere å se og/eller lese: ☐

Kommentarer:

Trafikk lys differens er tydelig

Runde: 1
Test Object: Hans Martin

☑ Er det greit vi lagrer følgende informasjon?

Background: (utdanning/arbeid) Høyskole kandidatur i informasjons behandling
Sykepleier, master i Perfusjonist

Nærsynt ☐ Langsynt ☐

Fargeblindhet ☐ Dyslekt: ☐

Andre tilstander som kan gjøre det vanskeligere å se og/eller lese: ☐

Kommentarer:

- rød prikk for opptak
- fri styring/spill har start istedenfor play
- Likte god nedslopp overalt
- "kjør simulator" istedenfor "kjør scenario"

Runde: 1
Test Object: 3

☑ Er det greit vi lagrer følgende informasjon?

Bakgrunn: (utdanning/arbeid) Helsesekretær VGS Helse ver

Nærsynt ☐ Langsynt ☑

Fargeblindhet ☐ Dyslektsi: ☐

Andre tilstander som kan gjøre det vanskeligere å se og/eller lese: ☐

Kommentarer:

- Kunne vært litt større skrift?

Rune le: 1
Test Object: 4

☑ Er det greit vi lagrer følgende informasjon?

Bakgrunn: (utdanning/arbeid) master i Nordisk språkvitenskap
Lærer

Nærsynt ☐ Langsynt ☐

Fargeblindhet ☐ Dyslektis: ☐

Andre tilstander som kan gjøre det vanskeligere å se og/eller lese: ☐

Kommentarer:

- Repeat tegn er utfyttelig
- Stopp knapp ukleart? (Lagres)
- gass "over", bremse "under"

Runde: 1
Test Object: 5

☑ Er det greit vi lagrer følgende informasjon?

Backgrunn: (utdanning/arbeid) Bachelor i Programmering Utvikler

Nærsynt ☐ Langsynt ☑

Fargeblindhet ☐ Dyslektiker ☐

Andre tilstander som kan gjøre det vanskeligere å se og/eller lese: ☐

Skjeve Hornhinner

Kommentarer:

- endre i pauset modus?
- kjøre på repeat vannsett
  - hardkode lyd?
- Kaun endre med ratt ???

Runde: 2
Test Object: Hans Martin

☑ Er det greit vi lagrer følgende informasjon?

Bakgrunn: (utdanning/arbeid) ~~~ // ~~~ som forrige

Nærsynt ☐ Langsynt ☐

Fargeblindhet ☐ Dyslekt: ☐

Andre tilstander som kan gjøre det vanskeligere å se og/eller lese: ☐

Kommentarer:

- liker kolor inaktiv/aktiv status
- fornøyd m. endringer fra forrige gang

Runde: 2
Test Object: 2

☑ Er det greit vi lagrer følgende informasjon?

Background: Sykepleier, Leder
(utdanning/arbeid)

Nærsynt ☐ Langsynt ☑

Fargeblindhet ☐ Dyslektis: ☐

Andre tilstander som kan gjøre det vanskeligere å se og/eller lese: ☐

Kommentarer:

- manuell styring istedenfor fri styring?
- klarere at fil er valgt
- Bra med modell og ikke bare graf

Runde: 2
Test Object: 3

☑ Er det greit vi lagrer følgende informasjon?

Background: (utdanning/arbeid) Bachelore i kulturledelse avsluttet

Nærsynt ☐ Langsynt ☐

Fargeblindhet ☐ Dyslektiker ☐

Andre tilstander som kan gjøre det vanskeligere å se og/eller lese: ☐

Kommentarer:

- kan dere litt tydligere på lyd?
- rett klarere v og h? Men ikke nødvendig

Runde: 2
Test Object: 4

☑ Er det greit å lagre følgende informasjon?

Background: (utdanning/arbeid) Bachelor i Sykepleie (sykepleier, hjemmesykepleier) fagbrev helsearbeider psykiatri

Nærsynt ☑ Langsynt ☐

Fargeblindhet ☐ Dyslektisk ☐

Andre tilstander som kan gjøre det vanskeligere å se og/eller lese: ☐

Kommentarer:

- ulike lagring?
  - hvordan lagre uten å avslutte
  - da altså egen lagre
- hadde likt å kunne åpne for lyd mens simulator spilles

Runde: 2
Test Object: 5

☑ Er det greit vi lagrer følgende informasjon?

Bakgrunn: (utdanning/arbeid) Bachelor i sykepleie

Nærsynt ☐ Langsynt ☑

Fargeblindhet ☐ Dyslektis: ☐

Andre tilstander som kan gjøre det vanskeligere å se og/eller lese: ☐

Kommentarer:

- ingen ting å klage på

Runde: 2, 5
Test Object: 1

☑ Er det greit å lagre følgende informasjon?

Backgrunn: Bachelorer i nyelmedier (utdanning/arbeid) 6 år utvikler

Nærsynt ☑ Langsynt ☐

Fargeblindhet ☐ Dyslekt: ☐

Andre tilstander som kan gjøre det vanskeligere å se og/eller lese: ☐

Kommentarer:

- navigering er lett
- Litt klarere på knapper
- Tegn på modus (f.eks rygg/kjør osv.)
- farge på venstre og høyre modell
- ikke rødt å grønt på graf
- skal man huske videre?
  (f.eks valgte mus: kjør, er mus da valgt: lag?)

Runde: 2,5
Test Object: 2

☑ Er det greit vi lagrer følgende informasjon?

Background: medisinsk teknisk ingeniør
(utdanning/arbeid) jobber på sykehuset

Nærsynt ☐ Langsynt ☐

Fargeblindhet ☐ Dyslektiker ☐

Andre tilstander som kan gjøre det vanskeligere å se og/eller lese: ☐

Kommentarer:

- virker enkelt og forståelig
- det virker kult
- mener mer forurensende med mer lagring
- kunne sett "smoothere" ut

D. Haziri, I. Kallevig, H. Muggerud, A. Tøn

Ambulansesimulator

NTNU
Kunnskap for en bedre verden