Bacheloroppgave

NTNU
Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for elektroniske systemer

Trond F. Christiansen
Syver Haraldsen
Peter A. B. Knutsen
Bendik Nygård

Utvikling og realisering av 5G-basert, fjernstyrt kjøretøy for brann- og redningstjenesten

Development and realization of a 5G-Based remotely controlled vehicle for the fire and rescue services

Bacheloroppgave i Elektronikk og sensorsystemer
Veileder: Arne Midjo
Medveileder: Are Hellandsvik
Mai 2023

NTNU
Kunnskap for en bedre verden

Trond F. Christiansen
Syver Haraldsen
Peter A. B. Knutsen
Bendik Nygård

Utvikling og realisering av 5G-basert, fjernstyrt kjøretøy for brann- og redningstjenesten

Development and realization of a 5G-Based remotely controlled vehicle for the fire and rescue services

Bacheloroppgave i Elektronikk og sensorsystemer
Veileder: Arne Midjo
Medveileder: Are Hellandsvik
Mai 2023

Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for elektroniske systemer

NTNU
Kunnskap for en bedre verden

NTNU

Tittelside Bacheloroppgave BIELEKTRO

Oppgavetittel (norsk og engelsk):
NO: Utvikling og realisering av 5G-basert, fjernstyrt kjøretøy for brann- og redningstjenesten

EN: Development and realization of a 5G-Based remotely controlled vehicle for the fire and rescue services

Forfattere:
Trond F. Christiansen
Syver Haraldsen
Bendik Nygård
Peter A. B. Knutsen

Prosjektnummer:	E2320
Innleveringsdato:	22.05.2023
Gradering:	[x] åpen [ ] lukket

Studium:	Elektroingeniør - BIELEKTRO
Studieretning:	Elektronikk og sensorsystemer
Veileder internt:	Arne Midjo
Institutt:	Institutt for elektroniske systemer
Oppdragsgiver:	SINTEF Digital, Trondheim
Kontaktperson:	Are Hellandsvik, Are.Hellandsvik@sintef.no, 99712637

Sammendrag (norsk og engelsk):
Trøndelag Brann- og Redningstjeneste (TBRT) eksperimenterer med ulike fjernstyrte kjøretøy for operativ bruk. Målsetningen er å få rask situasjonsforståelse ved hendelser der det er fare for eksplosjon og/eller giftig utslipp, og generelt der det er for risikabelt å sende inn innsatspersonell før man har fått oversikt. TBRT ønsker i den sammenheng å utvikle en fjernstyrt bil over 5G.

Trøndelag Fire and Rescue Service (TBRT) is experimenting with various remote-controlled vehicles for operational use. The objective is to gain rapid situational awareness in incidents where there is a risk of explosion and/or toxic discharge, and generally in situations where it is too risky to deploy response personnel before obtaining an overview of the situation. Therefore, TBRT wants to develop a remote-controlled car over 5G.

Stikkord norsk:	**Stikkord engelsk:**
Fjernstyrt bil	Remote-controlled car
Mobilnett	Mobile network
WebRTC	WebRTC
Redningstjeneste	Search and rescue

Sammendrag

Dette prosjektet ble utført for SINTEF på vegne av Trøndelag brann- og redningstjeneste. Målet med oppgaven var å utvikle en fjernstyrt bil over 5G med mulighet for å strømme video tilbake til en operatør som kontrollerer fartøyet utenfor fareområdet. Et viktig moment ved oppgaven var at systemet skulle være satt sammen av hyllevarer slik at redningstjenesten skulle kunne kjøpe inn, og sette sammen ny bil ved tap under oppdrag.

Det er tatt utgangspunkt i en 1/10 størrelse RC-bil fra traxxas. For å kontrollere bilen er det benyttet en Raspberry Pi 4B med utvidelsesmoduler. Utvidelsesmodulene gir enkortsdatamaskinen grensesnitt til mobilnettet, samt muligheten til å drive motorer med pulsbreddemodulasjon.

Det ble utviklet programvare som setter opp WebRTC-kommunikasjon mellom bilen og operatørens datamaskin for strømming av video og data. Videostrømmen sendes fra bilen til operatøren. Datastrømmen brukes til å sende styresignaler fra operatøren til bilen og statusinformasjon fra bilen til operatøren.

Det ble utviklet et brukergrensesnitt som viser videostrømmene fra bilen og sender styresignaler til bilen fra en tilkoblet kontroller. Brukergrensesnittet er utviklet til å være intuitivt og brukervennlig og er bygd med HTML, JavaScript og CSS.

Det ferdigstilte systemet starter automatisk ved oppstart av bil, og gir fra seg pipelyd når det er klart for tilkobling. Systemets levetid er målt til 3 timer og 45 minutter gjennom praktisk test med konstant kjøring og strømming av video.

Abstract

This project was conducted for SINTEF on behalf of Trøndelag brann- og redningstjeneste. The aim of the project was to develop a remote-controlled vehicle over 5G, with the capacity to send videostreams to a designated driver outside of the restricted zone, who would control the vehicle. One key requirement of the project was that the system should be composed of off-the-shelf components, enabling the rescue service to purchase and assemble a replacement vehicle in the event of loss during an operation.

The project was based on a 1/10 scale RC car manufactured by Traxxas. Control of the vehicle was facilitated through a Raspberry Pi 4B with expansion modules, providing the single-board computer with an interface to the mobile network and the capacity to drive servo-motors with pulse width modulation.

Software was developed to establish WebRTC communication between the vehicle and the operator's computer, enabling the streaming of video and data between the two devices. The video stream was conveyed from the vehicle to the operator, while the data stream facilitated the transmission of control signals from the operator to the vehicle and status indicators from the vehicle to the operator.

A user interface was created, which visualized the video streams from the vehicle, and allowed for the sending of control signals to the vehicle via an Xbox controller on the operator's side. The user interface was designed to be user-friendly and intuitive, utilizing HTML, JavaScript, and CSS.

The final system automatically initializes upon starting the vehicle, and emits an audible signal to indicate readiness for connection. The system was tested practically and demonstrated a lifetime of 3 hours and 45 minutes.

Forord

Denne bacheloroppgaven konkluderer tre år ved utdannelsen Elektroingeniør - Elektronikk og sensorsystemer, hos Norges teknisk-naturvitenskaplige universitet i Trondheim. Oppgaven er skrevet for SINTEF, på oppdrag fra Trøndelag brann- og redningstjeneste.

Prosjektet har vært utfordrende, men også veldig lærerikt. Det har vært spennende å kunne utvikle et system for så og feltteste det i en øvelsessituasjon sammen med nødetatene og vegtrafikksentralen.

Vi vil takke vår interne veileder Arne Midjo for å ha kommet med gode og verdifulle råd underveis i prosjektet og til skriving av rapport, for ikke å glemme tilgang til svar på spørsmål til alle døgnets tider.

Vi er også svært takknemlige ovenfor vår eksterne veileder, Are Hellandsvik. Are har kommet med gode tips og råd til hvordan strukturere, planlegge og gjennomføre prosjektet og har vært en viktig ressurs underveis i arbeidet. Vi vil også takke Are for å ha tatt seg tid, vel utenfor normal arbeidstid, til å være med på demonstrasjon og brannøvelse sammen med oss, redningstjenesten og vegtrafikksentralen.

Vi vil også rette en takk til SINTEF for å ha gitt oss muligheten til å gjennomføre dette prosjektet. Vi har satt stor pris på tilgang til både kontorlokaler og makerspace hos dere.

Syver Haraldsen

Syver Haraldsen

Bendik Nygård

Bendik Nygård

Peter Knutsen

Peter A. B. Knutsen

Trond F. Christiansen

Trond F. Christiansen

19.05.2023

Dato

Innhold

Figurer

Tabeller

Kodeutklipp

1 Terminologi

.stl *Standard Triangle Language*, filformat for 3D-modeller [94]

3GPP *3rd Generation Partnership Project*, Utvikler protokoller for mobilkommunikasjon [2]

ADC *Analog to Digital Converter*, Elektrisk krets for konvertering fra analoge til digitale signaler [100]

API *Application Programming Interface* [90]

Buck Converter Komponent som regulerer en gitt inngangsspenning ned til ønsket lastspenning, samtidig som den øker strømmen. [21]

CORS *Cross-Origin Resource Sharing* ressurs deling metode inkludert i HTTP protokollen. [26]

CSI-port Camera Serial Interface [119]

CSS *Cascading Style Sheets* [27]

EEPROM *Electrically erasable programmable read-only memory* [11]

ESC *Electronic Speed Controller*, Motorkontroller [122]

FDM *fused deposition modeling*, 3D printing metode [50]

FOV *Field of View*, Synsvinkel [83]

FPS *Frames per Second*, Bilder per sekund [104]

GPIO *General Purpose Input/Output* [64]

HAT Her; *Hardware Attached on Top* [8]

HTML *HyperText Markup Language* [41]

HTTP *Hypertext Transfer Protocol*, Nettverksprotokoll for overføring av data [42]

ICE *Interactive Connectivity Establishment*, Protokoll for tilkobling av p2p kommunikasjon i WebRTC [46]

ID her; *Identity*

IKS Interkommunale Selskap, hentet fra Lovdata[62]

IP *Internett protokoll* [52]

IR *Infrarød* [54]

Incendium TBRT sin portal for deling av videostrømmer [49]

IoT *Internet of Things*, tingenes internett [129]

JSON *JavaScript Object Notation* [55]

JavaScript Programmeringsspråk [69]

MVP *Minimum Viable Product* [67]

Megapixel Måleenhet for for bildeoppløsning, oppgitt i millioner piksler [89]

NAT *Network Address Translation*, Nettverksprotokoll [71]

Open-source Åpen kildekode, fritt til bruk basert på vedlagt lisens [73]

P2P *Peer to Peer*, Kommunikasjon direkte mellom to enheter, ikke via server [57]

PGA Programmable Gain Amplifier [17]

PID *Process IDentifier*, Unikt prosessnummer for identifisering av underprosesser [29]

PLA *Polylactic acid*, Termoplast ofte brukt til 3D printing [56]

PWM *Pulse With Modulation* [15]

Python Programmeringsspråk [84]

RPi *Raspberry Pi*, Ettkortsdatamaskin [123]

RTOS Real Time Operating System [34]

RTT *Round Trip Time*, Tiden frem og tilbake [20]

SAR Search and Rescue, søk og redning [33]

STUN *Simple Traversal of UDP Through NATs*, Nettverksprotokoll for p2p kommunikasjon [101]

TURN *Traversal Using Relays around NAT*, Nettverksprotokoll for p2p kommunikasjon [113]

UDP *User Datagram Protocol* Nettverksprotokoll [115]

URL *Uniform Resource Locator*, Nettverksprotokoll for å omforme nettverksadresser til menneske-leselig tekst [116]

V4L2 Video for Linux 2 [117]

WebRTC *Web Real-Time Communication*, Nettverksprotokoll for p2p videostrømming [51]

2 Introduksjon

I November 2023 publiserte SINTEF, på vegne av Trøndelag Brann- og Redningstjeneste, en bacheloroppgave med mål om å utvikle en fjernstyrt kjøretøyplattform med videostrømmingsfunksjonalitet som kan styres via 5G nettet. Det er tiltenkt at plattformen skal brukes av brannvesenet for å speide etter faremoment og eksplosiver fra en sikker avstand.

Bakgrunn

Bakgrunnen for denne oppgaven er et ønske om et lite, billig og lett håndterbart kjøretøy tiltenkt rekognosering i områder der innsatspersonell har lite oversikt over potensielle farekilder på åstedet.

Et konkret eksempel gitt av brann- og redningstjenesten er branner på bygningsplasser hvor det befinner seg Acetylen-flasker. Disse benyttes til skjærebrenning og sveising og er svært eksplosive, med en anbefalt sikkerhetssoneradius på 300 meter, samt en halvkule på 300 meter i høyde fra flaskens posisjon. [76] Dersom det er usikkerhet rundt flaskens lokasjon vil det ved brann måtte opprettes en enda større sikkerhetssone for å ta høyde for usikkerhetsmomentet. I et slikt tilfelle vil det være hensiktsmessig å benytte fjernstyrt kjøretøy for å lokalisere gassflasken, fremfor å sende inn innsatspersonell. Opprettelse av slike sikkerhetssoner kan være med på å påvirke nasjonal infrastruktur, f.eks ved stenging av hovedferdselsårer som veier. Dette er svært kostbart for samfunnet og ønsket er at man med bedre verktøy for rekognosering kan minimere tiden det tar å identifisere slike objekter, samt unødvendig bruk av store, lange, sikkerhetssnedstengninger.

Avgrensning

Grunnet de naturlige tidsbegrensingene som omhandler gjennomføringen av en bacheloroppgave, må oppgaven avgrenses på en slik måte at den er gjennomførbar innen tidsfristen, samtidig som den oppfyller gitte minimumskrav. En liste av minumumsfunksjonaliteterer er derfor utformet på bakgrunn av innspill fra veileder og oppdragsgiver, basert på kravene spesifisert i oppgavebeskrivelsen.[16] Prosjektoppgaven har derfor som fokus å utvikle en prototype baserte på en MVP som oppfyller alle minimumskrav, samt tilrettelegger for fremtidig utbygging.

SINTEF

SINTEF er ett av europas største forskningsinstitutt og opererer som en privat og uavhengig ideell organisasjon. Det vil si at nesten alle inntekter kommer fra private eller offentlige forskningsprosjekt hvor eier ikke kan ta ut utbytte, og alt økonomisk overskudd blir investert tilbake i instituttet i form av innkjøp av utstyr og ansettelse av nytt personale. [72] Instituttet ble grunnlagt i 1950 av daværende NTH, *nå, NTNU*, med det formål å bidra til positiv samfunnsutvikling gjennom å utvikle ny teknologi og løse problemer som samfunnet og industri møter. I dag er SINTEF et

forskningskonsern og består av 6 forskningsinstitutter, med over 2000 ansatte og er involvert i samarbeid med over 75 land verden over. [120]

SINTEF Digital er ett av de seks forskningsinstituttene som til sammen utgjør SINTEF. SINTEF Digital har hovedfokus på utvikling og forskning relatert til digitalisering av samfunnet, elektronikk og digitale teknologier som kunstig intelligens, robotikk og automasjon, sensorteknologi, og IOT [98]. Dette prosjektet har blitt gjennomført i samarbeid med SINTEF Digital.

TBRT

Trøndelag Brann- og Reddningstjeneste (TBRT) er utrykningspersonell som er ansvarlig for å respondere på meldinger om brann, ulykker og andre situasjoner hvor spesialtrent personell er essensielt. Trøndelag brann- og redningstjeneste IKS er den lokale faginstansen for brannvern i Trondheim, Malvik, Indre Fosen, Oppdal og Rennebu kommuner [31]. Drone Search-and-rescue (DroneSAR) er en relativt ny avdeling innenfor TBRT og er ansvarlig for operering av dronefartøy til rekognosering og overvåkning av brann- og redningsoperasjoner. I dag opererer DroneSAR flygende- og undervannsdroner og er interessert i å undersøke andre droneløsninger til bruk i situasjoner hvor nåværende droneløsninger ikke er gunstige, f.eks i bygg, industriområder eller tuneller.

Oppgavestruktur

Teori kapittelet er tiltenk å forsyne leser med relevant kontekst og begreper, slik at de kan få en bedre forståelse for innholdet i rapporten.

Metode kapittelet gir utfyllene begrunnelser for tekniske valg og arbeidsprosess. Dette inkluderer informasjon vedrørende til organisering av gruppen og oppgavefordeling blant medlemmene, beskrivelse av planleggingsprosessen og møtestrukturer. Tekniske begrunnelser inkluderer komponentvalg, valg av programvarer, generell arkitektur og testmetodikk.

Systemdesign kapittelet skal forklare de tekniske løsningene i detalj og forklare plattformens arkitektur og virkemåte. Dette inkluderer en detaljert oversikt over systemarkitekturen, forklaring av designvalg, beskrivelser av de tekniske løsningene og implementering.

Resultater kapittelet detaljerer prosjektresultatet og det endelige produktet som ble konstruert. Den inneholder også en detaljert gjennomgang av testresultatetene fra testene som ble beskrevet i metodedelen.

Diskusjon kapittelet omhandler en mer subjektiv vurdering av prosjektet og tolkninger av resultatene fra testene. Disse blir diskutert og forfatterne presenterer sine meninger om hvorfor resultatet ble som det ble, og hvilke mulige feilkilder som kan ha påvirket testresultatene. Kapittelet inneholder også drøfting om mulige forbedringer som kan gjennomføres dersom prosjektet skal videreføres.

Konklusjon kapittelet gir en helhetlig oppsummering av prosjektarbeidet, resultatene og diskusjonen.

3 Teori

For å forstå teknologien benyttet i prosjektet er det nødvendig med teori rundt hovedkonseptene.

3.1 I2C

Inter-Integrated Circuit(I^2C) er en to-leder seriell buss utviklet av Philips Semiconductors i 1982 [45]. I^2C bussen ble utviklet for å fasilitere for kommunikasjon mellom to eller flere noder på samme kretskort. I^2C har en relativt lav overføringshastighet, og brukes ofte til å kommunisere med sensorer over korte avstander hvor hastighet ikke er kritisk.

I^2C benyttes primært i en *single-controller-multi-target* arkitektur hvor I^2C bussen implementeres med en kontrollør som styrer og kommuniserer med flere mål koblet i parallell, men enkelte enheter støtter også fler-kontrollører-arkitektur. Alle I^2C noder på en buss krever en unik addresse for at kontrolløren skal kunne kommunisere med riktig enhet.

På det fysiske nivået består I^2C bussen av to signalledere: *Serial Data*(SDA) som benyttes til dataoverføring og *Serial Clock* (SCL) som er ansvarlig for tidssynkronisering slik at signalene leses av på riktig tidspunkt. I^2C-lederene krever eksterne termineringsmotstander for å trekke busspenningen høy.

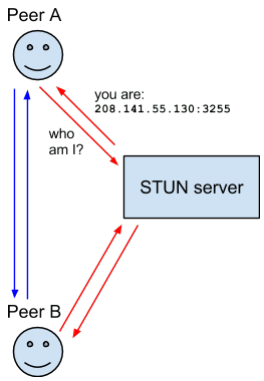
3.2 WebRTC

WebRTC (*Web Real-Time Communication*) er en protokoll som tillater overføring av video- og lyd-strømmer og rådata mellom enheter uten å måtte kommunisere gjennom et mellomledd.[51] Dette tillater strømming av data med lav forsinkelse, og uten ekstra infrastruktur.

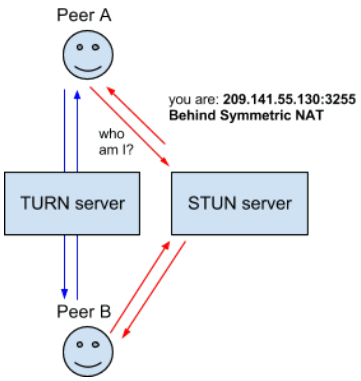
For å opprette kommunikasjon mellom enhetene bruker WebRTC ICE-protokollen for å kartlegge nettverksruten mellom enhetene, enten via STUN eller TURN.

STUN fungerer ved at enheten spør en STUN-server om sin offentlige IP adresse, hvilken type NAT den er bak og hvilken port på NAT-et som tilhører lokalporten. Denne informasjonen blir rapportert tilbake for hvert lag i nettverket. Informasjonen kan så sendes til RTC tjeneren for å informere om hvor dataen skal sendes. [44] [51]

TURN er en protokoll som brukes hvis direkte kommunikasjon med hjelp av STUN er blokkert på nettet, for eksempel ved et symmetrisk NAT, eller om klient og tjener står på forskjellige nettverk. Om dette er tilfellet vil forespørselen til STUN-serveren rapportere tilbake at den er blokkert av NAT-et, og TURN-serveren bli brukt istedenfor. Ved bruk av en TURN-server vil trafikken fra klienten gå til TURN-serveren og omdirigeres til tjeneren, og tilsvarende andre veien. [51][114]



(a) STUN [51]



(b) TURN [51]

CoTURN

CoTURN er en gratis *open source* implementasjon av STUN og TURN server. Denne programvaren kan kjøres på en Raspberry Pi for å sette opp private servere.[23]

3.3 5G

Mobilnettet er en sammensetning av flere basestasjoner som dekker hvert sitt geografiske område, og kommuniserer gjennom radioteknologi med mobile enheter. Mobilnettet er koblet sammen med det faste telefonnettet, og internettet.[130]

Betegnelsen 5G henviser til den femte generasjonen av mobilnett og ble formelt innført i 2019 under 3GPP sin revisjon 15. Denne generasjonen av mobilnett-protokoll støtter IoT og kommunikasjon mellom maskiner.[130]

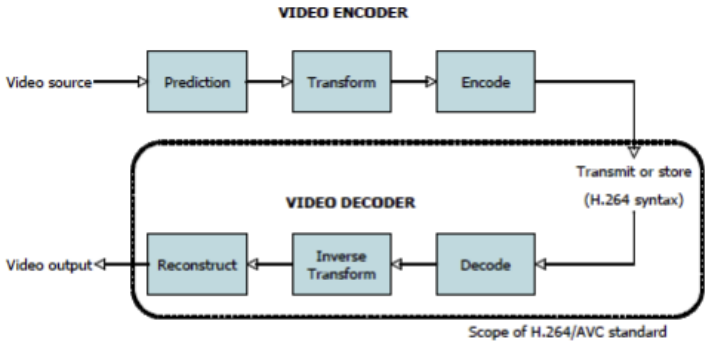
5G tar i bruk frekvensene 700MHz, 3.6-3.8GHZ, samt radiolink og testing på 26GHz.

3.4 Bildebehandling

Kompresjon

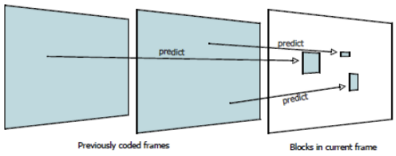
Bildekompresjon refererer til kompresjon som benyttes på bilder for å redusere datastørrelsen, og dermed gjøre den lettere å lagre eller overføre. Dette foregår ved at det som ansees som overflødig informasjon ikke lagres. Kompresjon kan gjøres enten tapsfritt (*loss-less*) eller ikke-tapsfritt (*lossy*). Ved tapsfri kompresjon kan man rekonstruere den originale dataen eksakt. Dette kan man ikke ved ikke-tapsfri kompresjon. Ved videostrømming brukes som regel ikke-tapsfri kompresjon for å minimere datamengden. Dette gjør at man ikke kan gjenskape videoen eksakt, men det gjenskapte bildet er som regel av høy nok kvalitet. [10]

h.264 Et ofte benyttet kompresjonsformat er h.264. Dette formatet ble lansert i 2003 og bygger videre på eldre formater som MPEG-2 og MPEG-4. Kompresjonsprosessen er delt inn i tre stadier; prediksjon, transformasjon og koding, som illustrert i figur 2.

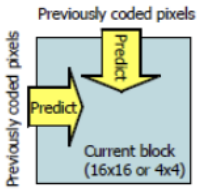


Figur 2: H.264 videokoding og -dekodingsprosess, hentet fra [118]

h.264-kompresjon deler opp bildet i blokker i størrelse fra 4x4 til 16x16 piksler. For å redusere hastigheten det tar å identifisere innholdet i disse blokkene benyttes det prediksjon. *Prediksjonsprosessen* foregår på en av to måter; *inter-* eller *intra-*prediksjon. Henholdsvis benyttes informasjon fra tidligere bilder, eller tidligere behandlet innhold i gjeldende bilde, til å estimere innholdet i den aktuelle blokken. Se figur 3a og 3b.



(a) Inter-prediksjon, hentet fra [118]



(b) Intra-prediksjon, hentet fra [118]

I transformasjonsprosessen benyttes *Diskret Cosinustransformasjon* til å beregne vektingskoeffisienter for det aktuelle bildet. Disse vektingene indikerer innholdet i bildet og benyttes i rekreasjonen. Her kan mengde kompresjon justeres etter ønsket kvalitet-størrelsesforhold. Flere null-koeffisienter gir lavere oppløsning, men resulterer i at videoen er hardere komprimert og krever mindre plass å lagre.

Den siste delen av kodeprosessen er bit-koding. Her konverteres transformasjonskoeffisientene sammen med annen nødvendig informasjon for å gjenskape bildet til en binær bitstrøm som kan lagres eller videresendes.

Denne prosessen er mer eller mindre speilet i dekodeprosessen, og det er dermed viktig at komprimeringen inneholder informasjon om på hvilken måte den er komprimert for at dekoderen skal kunne reversere prosessen. [118]

Bildestabilisering

Opplevd bildekvalitet henger ofte sammen med oppløsning og hvor skarpt eller uklart et bilde er. For å oppnå god bildekvalitet er en avhengig av faktorer som brennvidde(*focal length*), lysmengde og lukkerhastighet (*shutter speed*). Når man

ønsker så skarpt bilde som mulig for en gitt synsvinkel, er det vanlig å tilpasse disse tre faktorene til hverandre. For eksempel bør lukkertiden minimum settes til en faktor T i forhold til brennvidden.

$$T = \frac{1}{Brennvidde}$$

Når man skal fange skarpe bilder av objekter i bevegelse, eller når kamera i seg selv beveger seg, er man ofte avhengig av hurtig lukkertid, eller å benytte stabiliseringsteknikker. For video er stabilisering særdeles viktig. Det finnes flere måter å stabilisere et bilde på; *optisk stabilisering* og *digital stabilisering*[47]

Optisk bildestabilisering Optisk bildestabilisering referer til stabilisering hvor linse,*In-Lens*, eller sensor, *In-Camera*, fysisk beveger seg i forhold til enheten den er festet til. Dette foregår ved hjelp av tyngdekraft eller motorer og sensorer.

Digital stabilisering digital bildestabilisering viser til stabilisering som utføres i programvare, gjerne ved hjelp av utligningsalgoritmer. I likhet med optisk bildestabilisering finnes det flere teknikker for dette, men med videostrømmer på 30 og opp til 60 FPS kreves det ekstern prosessering i datamaskiner, og relativt høy prosessorkraft for å utføre de nødvendige beregningene uten å få høy forsinkelse.

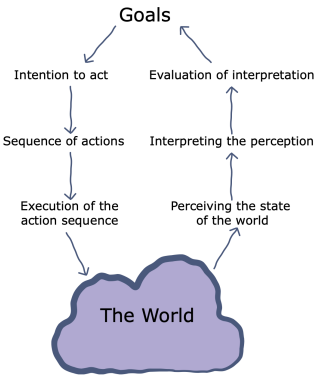
3.5 Brukeropplevelse og grensesnitt

I produkt- og webutvikling er to vanlige begreper brukergrensesnitt, *User Interface eller UI*, og brukeropplevelse, *User experience eller UX*. Hvor brukeropplevelsen viser til hvordan en bruker forholder seg til systemets helhet og følelsesaspekt, handler brukergrensesnittet om hvordan brukeren spesifikt samhandler eller interagerer med systemets skjermer, knapper og ikoner.[24]

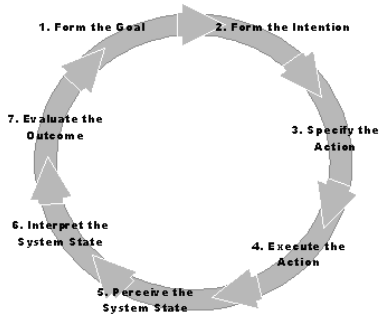
Handlingssyklus

I en artikkel for developer.com presenterer Mauro Marinilli generelle prinsipper for hvordan mennesker interagerer med systemer.[65] Marinilli viser her til Donald Normans handlingssyklus. Normans Handlingssyklus, eller *syv handlingssteg"*, er de syv stegene mennesker ubevisst går gjennom for alle interaksjoner med et system. Stegene er presentert i figur 4a og 4b.

Disse syv stegene kan deles inn i to bolker; *utførelse* og *evaluering*.[22] Disse bolkene viser til om steget er knyttet til utførelsen- eller resultatet av handlingen. I figur 4a vil stegene til venstre være knyttet til utførelsen, mens stegene til høyre er knyttet til evalueringen. Når det oppstår uoverensstemmelser mellom hva brukeren hadde som mål, gjorde for å oppnå målet, og fikk til resultat, er det å anse som dårlig design. Ved å ha et bevisst forhold til hvordan brukere tenker i møte med et system, kan designere bruke disse punktene som en sjekkliste i utformingen.



(a) Normans handlingssirkel [30]



(b) Marinillis fremstilling av handlingssirkelen [65]

Visuell kommunikasjon

Når det kommer til utforming av brukergrensesnitt og visuell kommunikasjon har Suzanne Martin for Worcester Polytechnic Institute skrevet om tre essensielle webdesignprinsipp, hentet fra grafisk design.[103]

Organiser Etterstreb å gi brukeren en tydelig konseptuell struktur ved å;

- være konsekvent internt i designet, samt utad med allerede etablerte normer innad i organisasjonen eller i verden.
- standardisere oppsettet av nettsiden og gruppere relaterte elementer.
- Koble sammen relaterte punkter til hverandre, og skill de fra ubeslektede punkter.
- skape et initielt fokuspunkt for navigasjon, styre fokus til viktige sekundære navigasjonspunkter og å hjelpe brukeren med navigasjonen i materialet.

Økonomiser Gi så mye informasjon som mulig, med så få visuelle elementer som mulig. Designet bør;

- være enkelt å forstå.
- være utvetydig i utformingen.
- gjøre det enkelt å skille viktige funksjoner fra hverandre
- gjøre det enkelt å legge merke til de viktige elementene på siden.

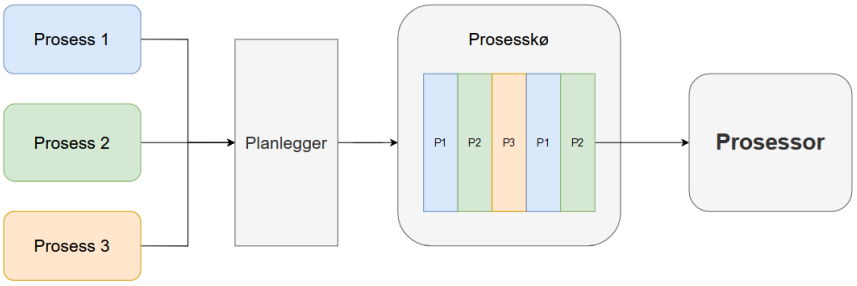
Kommuniser Tilpass presentasjonen til brukeren ved å planlegge bruken av typografi, farger, symbolisme og lesbarhet til den aktuelle brukergruppen. Lesbarhet innebærer at utformingen både er enkel å forstå, samtidig som at det må være attraktivt for brukeren å tilegne seg informasjonen på nettsiden. Dette gjøres ved å være bevisst i bruken av de ovenfornevnte elementene.

3.6 Prosessor-planlegging og trådprosessering

Trådprosessering gjør det mulig for en prosessor å utføre flere prosesser samtidig. Datamaskiner gjør dette ved å dele en prosess inn i mindre biter kalt tråder. Trådene kan deretter fordeles i tid eller blant flere prosessorkjerner.

Trådprosessering kan implementeres både i maskinvare og programvare. I maskinvare blir trådprosessering implementert ved å benytte CPUer med flere kjerner. Tråder fordeles blant kjernene og utfører instruksjoner parallelt om hverandre.

For å implementere trådprosessering med programvare blir det brukt en planlegger til å fordele trådene langs tidsaksen. Planleggeren lager en oversikt over hvilke prosesser som kjøres, finner prioritetene til prosessene og bestemmer rekkefølgen for når instruksjonene i en tråd skal utføres. Når dette er gjort blir trådene lagt til en prosess-kø som sender instruksene til prosessoren på en sekvensiell måte. [18]. Figur 5 gir en forenklet visualisering av trådprosessering av tre parallelle prosesser.



Figur 5: Trådprosessering

3.7 Multiprosessering

Multiprocessing (mp) er et python-bibliotek som gjør det mulig for et python program å opprette og styre flere underprosesser som kjører parallelt om hverandre. [70]. Mp-biblioteket har en rekke fordeler sammenlignet med andre liknende moduler. Underprosessene er uavhengige av hverandre og dersom en feil oppstår og en av prosessene stopper vil det ikke påvirke de andre prosessene. Underprosesser som blir startet er styrt av en hovedprosess, som gjør det lettere å starte, stoppe, avslutte og sammenslå underprosesser etter behov. Mp inneholder en rekke dataobjekter som gjør det lett å overføre data på tvers av underprosessene.

Process() klassen i mp gjør det mulig å opprette og styre underprosesser som et python-objekt. Klassen gjør det mulig å starte en python-funksjon, som en egen prosess med et unikt PID-nummer.

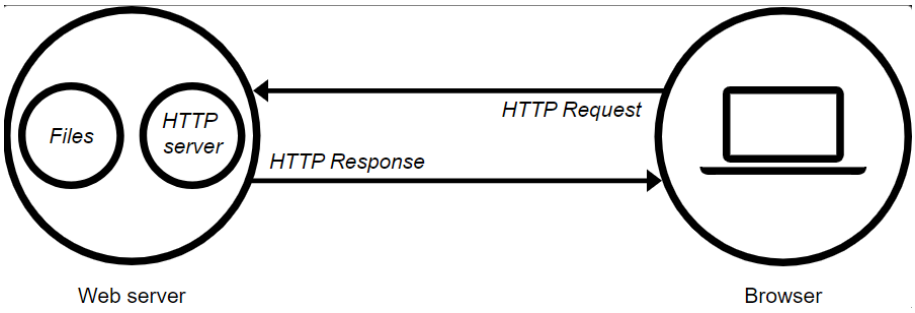
mp.Queue() er et dataobjekt som er inkludert i multiprocessing bilbioteket som oppretter et FIFO data objekt som kan overføre data på tvers av flere underprosesser. FIFO står for *First-In-First-Out* og er et dataobjekt som lagrer data i en sekvensiell rekkefølge. **Queue.put()** lagrer data på toppen av rekken og **Queue.get()** leser data fra bunnen av rekken. Data kan bare leses i samme rekkefølge som det ble skrevet. Når data leses blir det slettet fra rekken. Disse restriksjonene kommer på bekostning

av fleksibiliteten til strukturen, men fører til økt ytelse og kortere forsinkelser i enkelte bruksområder.

mp.Manager() er et mp objekt som gjør det lett for flere prosesser å kommunisere og overføre data mellom hverandre. Manager objektet kan bestå av flere forskjellige datatyper og støtter bruken av ikke-sekvensielle datastrukturer som f.eks. python-dicts. Det gjør det mulig å gjennomføre vilkårlige lese-og-skrive operasjoner, og fører til økt fleksibilitet sammenlignet med et **mp.Queue()** objekt.

3.8 HTTP-server

En HTTP-server er en programvare som er i stand til å forstå URLer og HTTP-kommandoer. En HTTP-server kan nåes fra domenenavnet til nettsiden den hoster, og sender nettsidens innhold til sluttbrukerens enhet. [124] Hver gang en nettleser trenger en fil som ligger på en nettserver sender den en HTTP-kommando til HTTP-serveren. HTTP-serveren svarer så med filen nettleseren ber om.



Figur 6: Normans handlingssirkel, hentet fra [43]

HTTP kommandoer

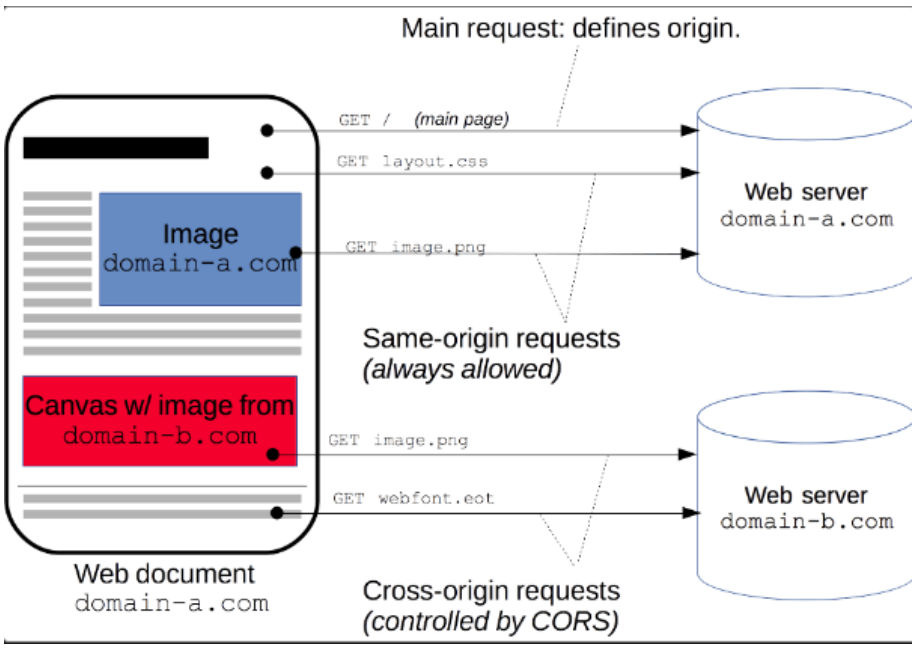
HTTP-protokollen inneholder en rekke forskjellige kommandoer som indikerer handlingen som skal utføres på ressursen som etterspørres.[42][43]

- **GET** kommandoen brukes for å etterspørre en spesifikk ressurs som skal returneres til klienten. Denne kommandoen brukes blant annet til å hente ut nettsider.
- **HEAD** fungerer på samme måte som GET, men returnerer bare headeren til ressursen, og ikke den medfølgende dataen.
- **POST** kommandoen brukes for å sende data til en server, for å oppdatere en spesifikk ressurs.
- **PUT** kommandoen brukes for å sende data til en server, for å erstatte en spesifikk ressurs.
- **DELETE** kommandoen brukes for å slette en spesifikk ressurs.

- **CONNECT** kommandoen brukes for å starte to-veis kommunikasjon med med den etterspurte ressursen.
- **OPTIONS** kommandoen brukes for å beskrive tilgjengelige kommunikasjonsmuligheter med den etterspurte ressursen.

Cors

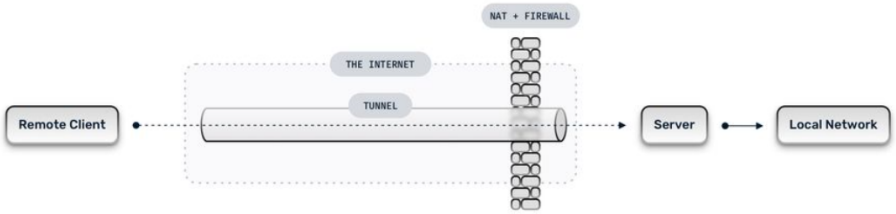
Cors er en HTTP-mekanisme som gjør det mulig for HTTP-servere å spesifisere at de tillater henting av ressurser fra andre kilder enn seg selv (andre nettsider ol.). Dette er nødvendig da nettlesere av sikkerhetsmessige hensyn normalt ikke tillater ressursdeling på tvers av opprinnelse. For at *Cors* skal fungere må begge enhetene være konfigurert til å tillate ressursdeling med andre kilder, enten spesifikt med den andre enheten, eller med alle andre enheter. Ved oppstart vil serveren så sjekke om den har tilgang på den eksterne enheten den ber etter, om den har dette vil den andre enheten kunne nåes med HTTP-kommandoer, hvis ikke vil *cors* returnere en feilmelding.[26]



Figur 7: Eksempel på bruk av cors, hentet fra [26]

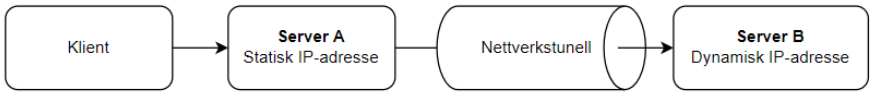
3.9 Nettverkstunneler

Nettverkstunnelering er en metode for å sende diskré data gjennom et ellers offentlig nettverk.[112][125] Datatransmisjonene foregår på et offentlig nettverk, men dataen er kun ment for bruk på et privat nettverk. Dette gjøres ved å sende dataen kryptert og pakket inn. Nettverkstunneler gir en direkte tilkobling mellom en ekstern server og det lokale nettverket. Datatransmisjonene er ikke-detekterbar på det offentlige nettet.



Figur 8: Eksempel på nettverkstunnelering, hentet fra [125]

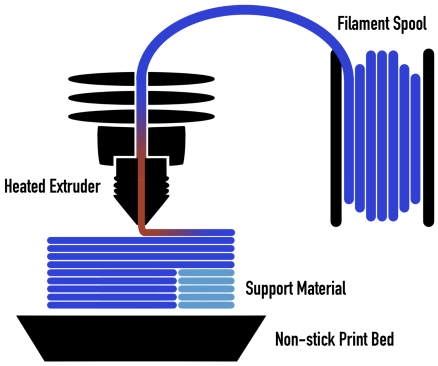
Nettverkstunneler kan også brukes til å nå en server uten en statisk IP-adresse. Dette gjøres ved å lage en tunnel mellom en server med statisk IP, og en server uten statisk IP. All data som kommer inn på serveren med statisk IP videresendes så gjennom nettverkstunnelen.



Figur 9: Nettverkstunnel til server med dynamisk IP-adresse

3.10 Fused deposition modeling

Fused deposition modeling (FDM) er en type additiv produksjonsteknologi som muliggjør konstruksjon av tredimensjonale objekter. Denne teknologien er brukt av de fleste hobby 3D-printere, men og tidvis i større industriell skala. FDM 3D-printing fungerer ved å deponere smeltet termoplast lagvis med en dyse. Disse lagene bygger oppå hverandre for å lage et tredimensjonalt objekt.[1]



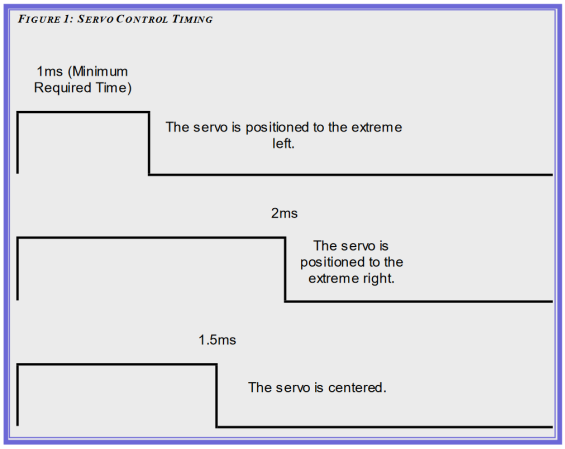
Figur 10: FDM 3D-printing, hentet fra [50]

3.11 Servomotor

En servomotor er en motor med tilbakekobling, slik at det er mulig å vite informasjon om servoens nåværende tilstand. Denne tilbakekoblingen gjør det mulig å finjustere styresignalet til servoen, og dermed øke presisjonen til motoren. [93] Billige

servomotorer for hobbyvirksomhet kommer typisk i to typer, posisjonelle eller kontinuerlige. En kontinuerlig servo kan rotere i 360 grader. Tilbakekoblingsmekanismen blir hovedsaklig brukt til å styre servoens rotasjonshastighet. Servomotoren blir derfor ofte brukt til styring av hjul og akslinger i mindre roboter og kjøretøy. I en posisjonell servo vil tilbakekoblingen hovedsakelig måle den nåværende posisjonen. Denne servotypen vil ofte være begrenset til å bare operere i rekkevidden mellom -90°og +90°.

Servomotorer kommer i mange typer og størrelser. Når man jobber med små og billige servomotorer er det vanlig at styresignalet kommer i form av et PWM signal der bredden til PWM signalet indikerer den ønskede posisjonen til servoen. For en kontinuerlig servo vil 1ms tilsvare maksimal hastighet i negativ retning, 2ms tilsvarer maksimal hastighet i positiv retning. 1.5ms tilsvarer ingen fart. I en posisjonell servomotor vil 1ms tilsvare en ønsket vinkel på -90°, 2ms tilsvarer en ønsket vinkel på +90°og 1.5ms tilsvarer en ønsket vinkel på 0°. Sammenhengen mellom ønsket vinkel og pulsbredde er indikert i figur 11. Det er vanlig at servomotorene opererer med PWM-frekvens på 50Hz.



Figur 11: Sammenhengen mellom pulsbredde og ønsket vinkel i en posisjonell servomotor, hentet fra [93]

3.12 PCA9685: 16-kanal, PWM Kontroller

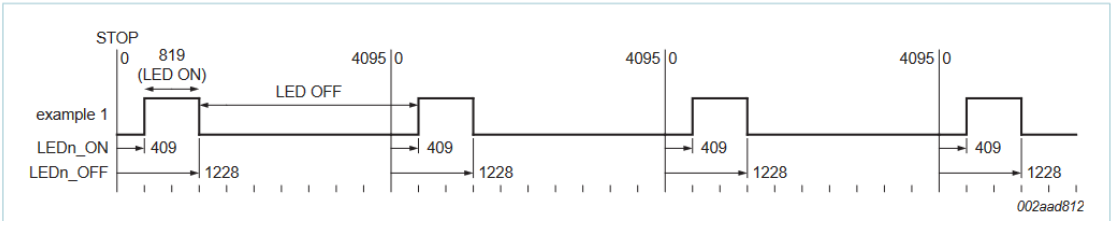
PCA9685 er en 12-bit PWM signalkontroller med 16 kanaler som styres via I^2C. Hver kanal kan generere et PWM signal med en programmerbar frekvens mellom 24Hz og 1526 Hz. Kontrolleren sin I^2C addresse blir valgt ved å konfigurere de 6 adresseinngangene. [78]

Resister 0x00 og 0x01 heter Mode1 og Mode2 og blir brukt til å konfigurere kontrolleren. Mode1 kan brukes til å konfigurere interne subadresser, autoinkrementering, bruk av eksterne klokker og tilbakestilling av kontrolleren. Mode 2 brukes til å konfigurere signalutgangen, *open-drain* strukturen, utgangssynkronisering og utgangsinvertering.

PWM frekvensen blir kontrollert ved å konfigurere pre-skalarverdien i register 0xFE. Ønsket preskalarverdi blir beregnet med funksjon 1.

$$pre\_scale = round(\frac{osc\_clock}{4096 * frequency}) - 1 \quad (1)$$

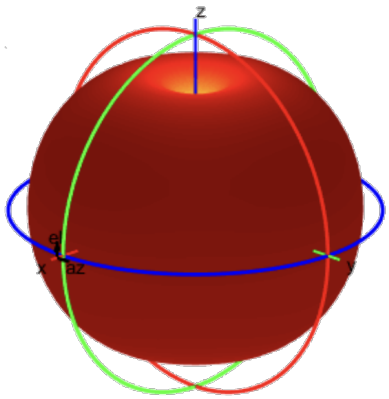
PWM signalet til en kanal blir styrt ved å konfigurere verdiene i LEDn_ON og LEDn_OFF, der n tilsvarer kanalen som blir konfigurert. LEDn_ON og LEDn_OFF er 12-bit verdier fordelt mellom to 8-bit registre. PWM signalet har en oppløsning på 12-bit der 0 tilsvarer starten på en periode og 4095 tilsvarer starten på neste periode. LEDn_ON indikerer punktet i perioden der utgangen blir HØY og LEDn_OFF indikerer punktet i perioden der utgangen blir LAV. Dette gjør det mulig å konfigurere et PWM signal med en valgfri periode og faseforskyvning.



Figur 12: PWM LEDn_ON and LEDn_OFF eksempel, hentet fra[78]

3.13 Monopol antenne

En monopol antenne består av et rett antenneelement og er omnidireksjonell. Dette vil si at antennen sender og mottar signal likt i alle retninger, men har redusert evne både opp og ned.[12] Antennen er bi-direksjonell, hvilket tilsier at den kan benyttes som både sender og mottaker av signal.



Figur 13: Strålingsmønster til en monopol antenne, hentet fra [25]

3.14 DC/DC omforming

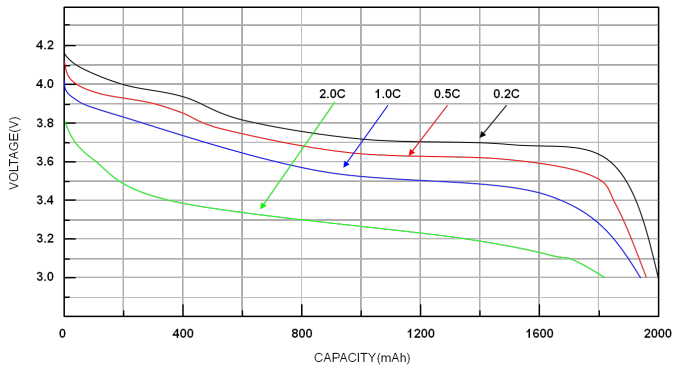
En svitsjende DC/DC omformer benyttes for å justere spenningen fra en forsyning. En type nedtrappings-omformer er Buck-omformeren. Denne fungerer ved å skru forsyningsspenningen av og på ved høy frekvens, for å så jevne ut oscillasjonene til et konstant redusert nivå. under omforming vil deler av energien tapes som varme.

Omformerens effektivitet vil gjerne variere ved forskjellig inngangsspenning, samt strømtrekk. For å regne ut omformerens effektivitet må en dele utgangseffekten på inngangseffekten.

3.15 Li-Po batteri

Litium Polymer Batterier, også kjent som Li-Po batterier, er batterier som bruker de kjemiske egenskapene til litium til å oppbevare elektriske energi. Fordelene med Li-Po batterier at de har en veldig høy energitetthet. Dette fører til at de kan oppbevare store menger energi pr. kilo. Den har også en veldig høy utladningsrate og kan forsyne store mengder strøm kontinuerlig. Kombinasjonen av disse egenskapen fører til at batteriene er hyppig brukt i radiostyrte biler, droner og andre applikasjoner hvor det er viktig med lav vekt og samtidig store strømmer.

En annen egenskap hos Li-Po batterier er at de har en veldig flat utladningskurve. Når en Li-Po celle er fulladet vil den typisk ha en spenning på rundt 4.3V. Ved kontinuerlig bruk vil spenning raskt nå sin nominelle spenning på ca. 3.7V. Batterispenningen vil deretter sakte reduseres frem til batteriet er nesten tomt. Deretter vil raten på spenningsfallet øke igjen. Denne spenningskurven er illustrert i figur 14. En lineær utladningskurve resulterer i lav spenningvariasjon sammenlignet med andre batterityper.



Figur 14: Li-Po Utladningskurve, hentet fra [82]

En nedside ved å bruke Litium-baserte batterier er at beregninger av resterende batteriladning, basert på batterispenningen, ofte er upresise. Den flate utladningskurven gjør at små spenningsforskjeller i batteriet utgjør store ladningsvariasjoner. Det er også et problem at batterispenningen blir påvirket av temperatur og utladningsrate. Dette er illustrert i figur 14.

En Li-Po celle har som regel ganske faste spenningsnivåer mellom 4.2V og 3V. For applikasjoner som foretrekker andre spenningsverdier er det derfor vanlig å konstruere batteripakker med flere celler i serie. Celler kan også legges i parallell med hverandre for å øke energikapasiteten til batteriet, uten å påvirke batterispenningen. Batteriarkitekturen til batteripakker vil ofte bli indikert i formatet nSmP, der n indikerer antall celler i serie og m indikerer antall celler i parallell. En batteripakke med 3S1P arkitektur vil derfor bestå av 3 Li-Po celler i serie med hverandre og ha en nominell spenning på omtrent 11.1V.

Den høye energitettheten i Li-Po batterier medfører også et stort faremoment. Hvis et slikt batteri blir utsatt for skade er det mulig at store energimengder blir frigjort i et kort tidsintervall. Dette gjør at Li-Po og andre litium baserte batterier kan antennes og eksplodere hvis de ikke håndteres riktig. Det er derfor anbefalt å bare lade slike batterier med spesiallagde batteriladere. [82]

3.16 Komponenter

Følgende komponenter er benyttet i utviklingen av dette kjøretøyet, med unntak av servoer, motorer og motorkontroller som var inkludert i Traxxas TRX-4.

Traxxas TRX-4

Som base for prosjektet er det benyttet en Traxxas TRX-4 Land Rover Defender Silver 1/10 RTR. Denne modellen er designet med firehjulsdrift, og en solid metallramme for å unngå deformasjon, som følge av røft terreng. [32]

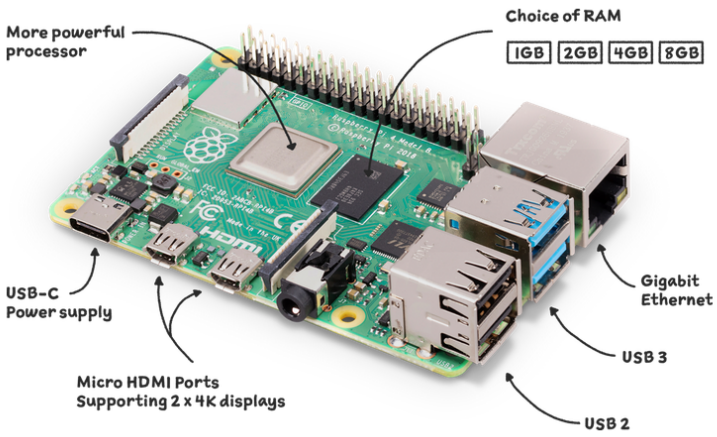
Figur 15: Illustrasjon av Traxxas TRX4 chassis, hentet fra [109]

Raspberry Pi

Raspberry Pi 4 (RPi4) er en enkortsdatamaskin utviklet av Raspberry Pi Foundation [39]. Datamaskinen er utbredt innen hobbyelektronikkprosjekter. Dette har resultert i et stort brukermiljø. Grunnet enkortsdatamaskinenes store brukermiljø eksisterer det mye programvare og kode-eksempler. Dette gjør RPi til en god enhet for testing og prototyping.

Raspberry Pi er basert på en Quad core 64-bit ARM-Cortex A72 prosessor og har en lang liste med funksjonaliteter som gjør at den er svært fleksibel og lett å bruke [87].

- H.265 (HEVC) hardware decode (up to 4Kp60)
- H.264 hardware decode (up to 1080p60)
- 2x USB3 ports
- 2x USB2 ports
- 802.11 b/g/n/ac Wireless LAN
- 1x Gigabit Ethernet port (supports PoE with add-on PoE HAT)
- 28x user GPIO supporting various interface options:
  - Up to 6x I^2C
  - Up to 2x PWM channels

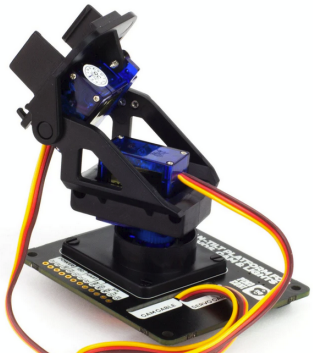


Figur 16: Raspbery Pi 4B+, hentet fra [63]

Raspberry Pi HAT Raspberry Pi HAT (*Hardware Attached on Top*), er tilleggskort designet for å monteres på RPi. For å kunne kalles en HAT kreves det at tilleggskortet møter en rekke designspesifikasjoner, blant annet knyttet til fysisk utforming og bruk av ID EEPROM.[9]

Pan-Tilt-HAT

Pan-Tilt-HAT er en tilleggsmodul bygd for 180°styring av kamera utviklet av Pimoroni. HAT´en benytter to servoer som lar brukeren justere kameravinkel langs både horisontal og vertikal akse.[77]



Figur 17: Illustrasjon av Pan-Tilt-HAT

Adafruit 16-Channel PWM / Servo HAT

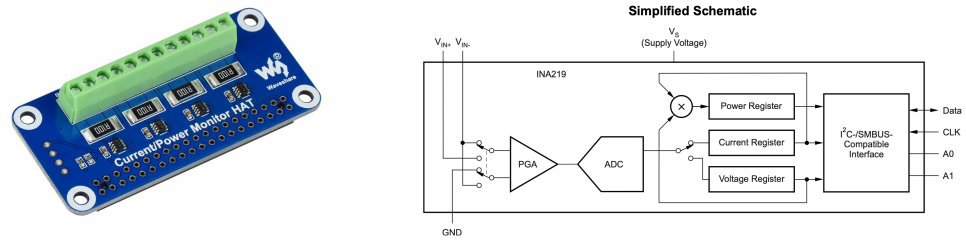
Adafruit PWM/Servo HAT muliggjør kontroll av opp til 16 servoer samtidig. Den har en PWM-oppløsning på 12-bit med frekvens opp til 1.6kHz. Servokontrollen sender ut et 3.3V logisk styresignal til tilkoblede servoer. PWM/Servo HAT kommuniserer med RPi over I^2C. [58] [59]

Waveshare Current/Power Monitor HAT

Waveshare sin *Current/Power Monitor HAT* er en 4-kanals spennings- og strømovervåker designet for å monteres på RPi.

Overvåkeren benytter en 12-bit ADC, *INA219*, og kommuniserer med RPi over I^2C-bussen. Produsent oppgir at ADC kan måle spenninger fra 0 til 26V. Kretskortet er satt opp med en 0.1 Ω samplingsresistor som tillater måling av bi-direksjonell strøm på opp til 3.2A [121]

Texas Instruments INA219 INA219 er en strøm- og spennings-overvåker bygget av Texas Instruments. Den består av en PGA, ADC, tre kalibreringsregistere og et I^2C grensesnitt. [106]



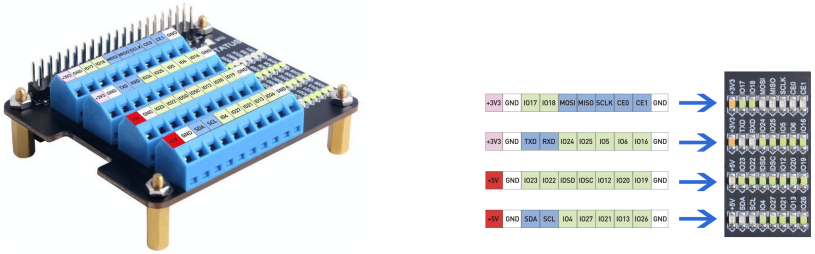
(a) Current/Power HAT, hentet fra [121]

(b) Forenklet skjematikk for INA219, hentet fra [106]

Figur 19: INA219

52PI GPIO Screw Terminal HAT

GPIO Screw Terminal HAT er en tilleggsmodul designet for RPi som enkelt lar bruker benytte skruterminaler for tilgang til GPIO pinnene, istedenfor pinnene på RPi. I tillegg har kortet lysdioder som indikerer om en pinne er satt høy.



(a) Illustrasjon av GPIO HAT, hentet fra [7]

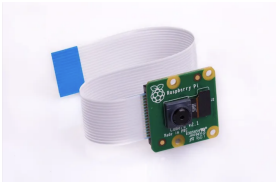
(b) GPIO HAT LED Pinout, hentet fra [7]

Figur 20: GPIO HAT

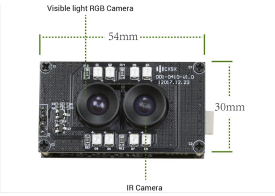
Kamera

Raspberry Pi Camera Module 2 Pi Camera Module 2, eller *PiCamera*, er en kameramodul designet av Raspberry Pi Foundation til bruk sammen med deres RPi. Modulen Benytter en 8-megapixel Sony IMX219 sensor og kobles til RPi via CSI-porten. Kameraet støtter 30fps@1080p, 60fps@720p og 90fps@VGA. Kameraet kan aksesseres ved hjelp av f.eks V4L2 API. [79]

Arducam B0198 Arducam B0198 er en kameramodul med to bildesensorer designet av Arducam. Modulen benytter to 2-megapixel OV2710 bildesensorer, som har en maksimal oppløsning på 1920x1080p og en FOV på 95°. Begge linsene har fast fokuspunkt, men skiller seg med at en av sensorene har et IR-filter. Hele modulen drives over USB og justerer automatisk følgende parametere: metning, kontrast, akutans, hvitbalanse og eksponering. Følgende bildefrekvenser er støttet: MJPG 30fps@1080p, 30fps@720p; YUY2 30fps@640x 480.[14][13]



(a) Picamera, hentet fra [79]

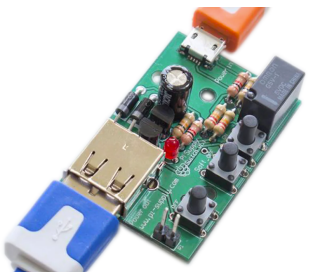


(b) Arducam, hentet fra [13]

Figur 21: Kameramoduler

Raspberry Pi on/off module

Raspberry Pi on/off moduele er en gjør-det-selv strømstyreenhet for RPi, produsert av pi-supply. Denne modulen tillater brukeren å skru av RPi via programvare med en fysisk knapp, hvilket er tryggere for enheten enn å kutte strømmen direkte. Modulen er også i stand til å kutte strømmen til RPien etter den er skrudd av, samt skru på enheten igjen.



Figur 22: Illustrasjon av Pi Switch, hentet fra pi-supply [102]

Gens Ace 3S 5000mAh 50C - batteri

Gens Ace 3S 5000mAh 50C er et LiPo-batteri designet for bruk i RC biler. Batteriet består av 3 LiPo celler i serie, hvilket gir en nominell spenning på 11.1v, men en maksimal spenning på 12.6v og en minimal på 9.0v. Kapasiteten til batteriet på 5000mAh gjør det godt egnet til enheter med høyt strømtrekk. Batteriet benytter XT60 plugg for tilkobling.

DFRobot DFR0929 - DC/DC Buck Converter

DFR0929 er en isolert 50W DC-DC buck omformer designet av DFRobot. Omformeren håndterer en inngangsspenning mellom 9 og 18V og kan levere 5V/10A til en last. Omformeren kommer i en solid innkapsling og har en oppgitt effektivitet på 83%. [68]

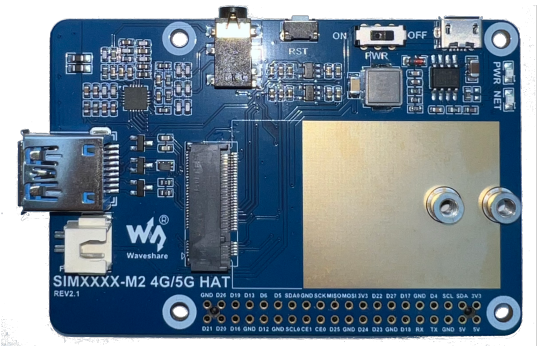
Figur 24: Illustrasjon av Godsend DC/DC omformer, hentet fra [68]

SIMXXXX-M2

For å koble Raspberry Pi til mobilnettet trengs en SIMXXXX-M2 HAT fra selskapet Waveshare, i tillegg til en M.2 Mobildatamodul fra selskapet SIMCom. [96]

SIM8200EA-M2 og SIM8262-E er 5G moduler med M.2 Grensesnitt. Raspberry Pi 4B er ikke direkte kompatibel med M.2. Derfor benyttes SIMXXXX-M2 HAT fra Waveshare. Denne kobles på GPIO-Pinnene til Raspberry Pi, og kontrolleres serielt over USB.

SIMXXXX-M2 HAT benytter seg av sin egen eksterne strømforsyning på 5V, tilført via mikro-usb port.



Figur 25: SIMXXXX-M2 HAT

Seriell Kommunikasjon

Kommunikasjonen mellom Raspberry Pi 4B og SIMXXXX-M2 HAT foregår over serielt grensesnitt. Kommunikasjon foregår med AT kommandoer. Med slike serielle kommandoer kan man ha to-veis kommunikasjon med SIM8200EA-M2, eller SIM8262E-M2.[95]

Et eksempel på en slik seriell dialog kan være at bruker sender "AT", og får tilbake "OK"hvis enheten er tilgjengelig og lytter på den serielle porten.

SIM8200EA-M2

SIM8200EA-M2 støtter 3GPP R15. 3GPP er den mobile bredbånd-standarden, og R15 er det første komplette settet av standarder for 5G. [4]

SIM8200EA-M2 har en teoretisk maksimal overføringskapasitet på 2.4Gb/s nedlastning, og 500Mb/s opplastning.

Figur 26: SIM8200EA-M2 hentet fra [40].

SIM8262E-M2

SIM8262E-M2 er et alternativ til SIM8200EA-M2. SIM8262E-M2 er basert på 3GPP R16. Revisjon 16 omtales som "5G Phase 2"[5]. Revisjon 16 ble utviklet for industri 4.0[3]. Revisjonen er utviklet for industriell IoT, og lav ventetid på signalene.

Figur 27: SIM8262E-M2 hentet fra [97].

Teleoperatør

For å koble 5G-modulene til det mobile nettet kreves det abonnement hos en teleoperatør. Dette er fordi modulen bruker oppringt nettverk. For at oppkobling til det mobile nettet skal lykkes, må abonnementet da ha mulighet for tele og SMS. Dette utelukker data-sim som er vanlig for elektronikk.

3.17 Strømnestunnelen

Strømnestunnelen ligger i Steinkjer kommune og åpnet i april 2020. Den er på grunn av sin lengde definert som et særskilt brannobjekt, noe som medfører krav om fullskala brannøvelse hvert fjerde år.[128]. Tunnelen har innlagt gjennomgående drypp-antenne for mobildekning. [111]

4 Metode

Metoden beskriver hvordan vi har gått frem for å planlegge, utvikle og teste systemets maskin- og programvare.

4.1 Samarbeid

Samarbeidsavtale

Ved starten av prosjektet skrev samtlige gruppemedlemmer under en standard samarbeidsavtale som anbefalt av NTNU. Dette for å ha et skriftlig dokument til hjelp dersom det skulle oppstå splid innad i gruppa og for å regulere arbeidsmengde og ansvar.

Digitale samarbeidsplattformer

Det er i arbeidet benyttet flere ulike plattformer for digitalt samarbeid for å samkjøre filer, kode og fremdrift. Felles for disse er at de gjør det enkelt å sørge for at alle medlemmer i gruppa har tilgang på oppdatert og aktuell informasjon, samt riktig versjoner av ny kode.

Teams Teams har blitt benyttet til kommunikasjon internt i arbeidsgruppa, ut til interne- og eksterne veiledere, samt med kunde og prosjekteier. Internt har også Teams fungert som fildatabase for bl.a. møtereferater, arbeidslogg og toukersrapporter. [61]

Github Github er benyttet til versjonskontroll og kodedeling mellom gruppemedlemmene under arbeidet, samt opplasting av kode til Raspberry Pi. [37]

Fusion360 Fusion360 har blitt benyttet til design og modellering av festebraketter for å kunne montere eksterne komponenter til TRX4-rammen, presentert i avsnitt 3.16. Det er opprettet et felles arbeidsområde i Fusion360. Et slikt felles arbeidsområde gjør det mulig å teste modellene våre opp mot hverandres modeller, samt produsenters modeller av elektroniske komponenter. Ved å arbeide på denne måten kan vi bekrefte at modulene våre er kompatible før vi 3D-printer delene. [35]

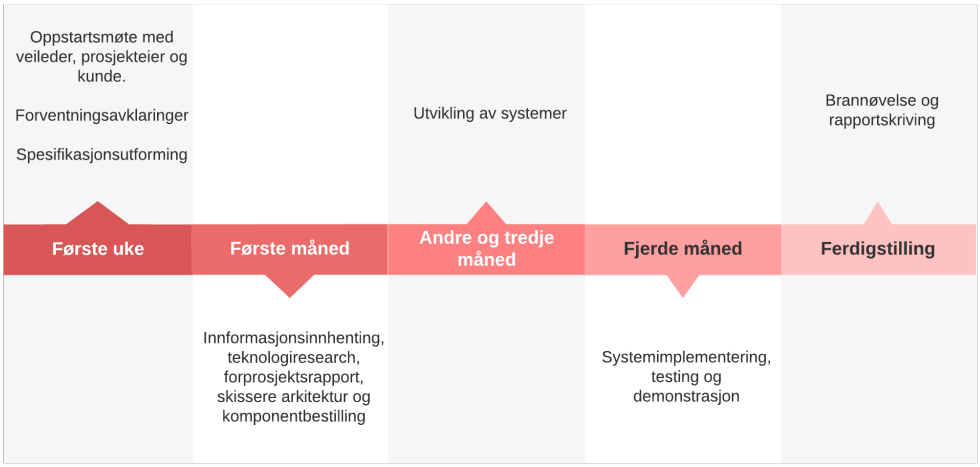
GanttProject GanttProject er et program for å tegne Gantt-skjema og ble benyttet tidlig i arbeidet for å lage en fremdriftsoversikt med milepæler og estimert tid for utvikling av de ulike modulene i prosjektet. Skjema kan leses i vedlegg-seksjonen. [92]

4.2 Planlegging

Planleggingen av prosjektet ble gjennomført som en del av forprosjektet. Her ble både arbeidsprosessen, samt makroarkitekturen definert.[19]

4.2.1 Arbeidsprosess

En fremdriftsplan ble konstruert for å dele prosjektoppgaven inn i mindre arbeidsoppgaver. Dette skaper en mer helhetlig oversikt over arbeidsoppgavene og gjør det enklere å fordele oppgaver blant gruppemedlemmene. Fremdriftsplanen ble visualisert i et gantt-diagram og er lagt til som et vedlegg. [92] Fremdriftplanen fordelte arbeidet inn i fem deler. Den første uken ble brukt til oppstart av prosjektet. Dette inkluderer skriving av forventningsavklaring, møter med veiledere og planlegging av arbeidet. Den første måneden ble reservert til planlegging, informasjonssamling og skriving av forprosjektsrapport, samt bestilling av deler. I andre og tredje måned startet utvikling av prosjektet. Den fjerde måneden ble brukt til integrering av modulene, feilsøking og 3D-Printing av festebraketter. I slutten av april gjennomførte vi en funksjonstest av plattformen sammen med TBRT. Deretter ble hovedfokus på ferdigstilling av prosjektrapporten.



Figur 28: Tidslinje for arbeidsprosess

4.2.2 Designkrav

En rekke designkrav har blir utarbeidet i samarbeid med veileder og oppdragsgiver for å sette konkrete mål for prosjektet. Disse kravene beskriver minimumsfunksjonaliteter som må være til stede for at plattformen skal kunne gjennomføre et grunnleggende rekognoseringsoppdrag. Kort oppsummert må plattformen kunne styres via mobilnettet, kjøre og navigere på en kontrollert måte og sende video og data tilbake til operatøren. Det er også krav relatert til valg av komponentene. Mer om dette i spesifiseringene under.

Kontroll av kjøretøy

I dialog med Trøndelag brann- og redningstjeneste ble det presentert en plan om at fartøyet skulle kunne kontrolleres fjernstyrt av en designert operatør, fra brannvesenets sentral. Det ble i tillegg fremmet et ønske om en enkel og tilgjengelig styremetode, gjerne Xbox kontroller.

Optikk

Fra Brannvesenet ble det fremmet et ønske om å kunne ha flere kamera til forskjellige formål. Det ble presentert ønske om å implementere IR kamera, samt standard dagslyskamera. Det var i tillegg ønske å kunne rotere på kameraene, slik at operatøren av fartøyet kan se omgivelsene sine.

Brukergrensesnitt

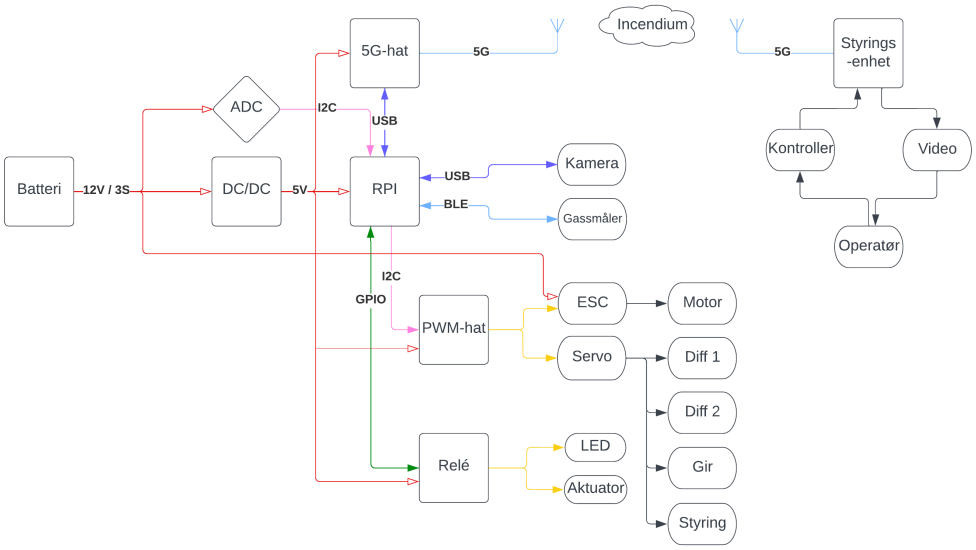
I innledende dialog angående prosjektet ble det ikke spesifisert hvordan det grafiske brukergrensesnittet skulle utformes. Det ble internt i prosjektgruppa vedtatt at grensesnittet burde være oversiktlig og enkelt utformet slik at det ikke krever høy teknisk innsikt for å forstå, samt enkelt kan brukes selv under stressende situasjoner. Dette er også i tråd med den overordnede designfilosofien for prosjektet.

Produksjon og Komponentvalg

Et av kravene som ligger i prosjektbestillingen er at produktet så langt det lar seg gjøre skal benytte eksisterende og kommersiell teknologi, som gjerne kan kjøpes fra norske forhandlere. *"For TBRT er det interessant å utforske kommersielt tilgjengelig teknologi, både for å holde kostnadene nede, da man må regne med at kjøretøy kan gå tapt under oppdrag, og for å raskt kunne ta i bruk nyheter på markedet og tilpasse dette til operative krav."*[16] Vi tolker dette dit at det ikke er ønskelig med egendesignede løsninger for elektroniske kretser, men at vi skal kjøpe hyllevarer. Dette vil i utgangspunktet gjelde for plattformen vi utvikler, mens festeanordninger som vil være spesifikke til vår RC-bil modell vil måtte spesialdesignes og 3D printes for enkel montering. Kunde har også sagt at de har mulighet for 3D-print og at det er et akseptabelt nivå av spesialtilpassede løsninger.

4.2.3 Makroarkitektur

Ved oppstart ble det skissert en plattformsarkitektur tenkt som utgangspunkt for arbeidet. Å ha en skisse for hvordan et system skal fungere og henge sammen er essensielt for å vite hvor man skal starte arbeidet, hva som er kritisk for å oppnå et fungerende produkt og å vite hvilke komponenter som må handles inn slik at man unngår unødvendig venting underveis. Dette lar i tillegg alle medlemmer i arbeidsgruppa være enig om hvordan de ulike modulene skal integreres og gjør senere implementeringsarbeid enklere.



Figur 29: Førsteutkast plattformsarkitektur

4.3 Komponentvalg

Deler av planleggingsfasen ble brukt til kartleggingen av kompatible komponenter til systemet. Det ble lagt vekt på at systemet skal være enkelt å reprodusere, samt at komponentene skal være lett tilgjengelige.

Utstyr fra TBRT

Ved oppstart av prosjektet hadde TBRT allerede kjøpt inn komponenter de mente var hensiktsmessig til prosjektet, noe som påvirket senere komponentvalg. Disse komponentene inkluderte Raspberry Pi 4B+[63], SIMxxxx-M2 HAT[96] og Traxxas TRX-4 [109].

RC-plattform

Ved oppstart av prosjeketet hadde TBRT gått til innkjøp av en RC-bil de mente var velegnet til den forespeilede bruken. I likhet med mye av utstyret Drone-teamet benytter er kjøretøyet kjøpt inn fra Elefun.no. Dette i tråd med kravspesifikasjonene 4.2.2 som sier at komponenter skal være enkle å få tak i, samt gjerne fra norske forhandlere.

5G moduler

Ved valg av mobilnett-modul er det flere aspekter som må tas i betraktning. Det mest substansielle aspektet er at modulen må støtte revisjon R15 av 3GPP som innførte 5G mobildata. Modulen må i tillegg ha god støtte for antenner, slik at den

har god dekning. Det er også viktig at modulen har høy nok overføringsevne til å kunne sende fra seg videostrøm i god oppløsning.

Ved oppstart av prosjektet ble det tatt i bruk SIM8200EA-M2 moduler fra SIMCOM. Modulene støttet 3GPP R15, samt tidligere revisjoner. Dette gjør at modulen kan bytte protokoll til 4G om den mister 5G-dekningen. Ved videre utvikling av prosjektet ble dog SIM8200EA-M2 byttet ut med SIM8262E-M2 fra SIMCOM. Dette ble gjort som en oppgradering, da SIM8262E-M2 støttet neste revisjon av 3GPP som innførte systemer for å redusere latens, samt forbedre påliteligheten til signalene. De to modulene benytter seg av samme styresignaler, hvilket tillater enkelt bytte av modulene uten å endre andre deler av systemet.

Kamera

I dialog med TBRT kom vi frem til at det var flere ting vi ønsket at kamera på plattformen skulle være i stand til å gjøre. Det skulle være vidvinklet og stabilisert for å gjøre kjøreopplevelsen best mulig. Kamera skulle kunne snu på seg for å speide etter objekter. Det var også ønske om et IR- eller lavlys-kamera for å kunne se i mørke omgivelser. Basert på disse ønskene bestemte vi oss for to forskjellige kameramoduler. En kameramodul for å kjøre med, og en med et IR-kamera for speiding.

Raspberry Pi camera module Som kjørekamera var vi ute etter et vidvinklet kamera med optisk bildestabilisering som var enkelt å koble til RPien. På grunn av dette valgte vi å gå for *Raspberry Pi camera module v3 wide* fra Raspberry Pi [86] da denne modulen oppfyller alle kravene og har mye dokumentasjon på hvordan den brukes.

Under utviklingen av programvaren til plattformen fant vi ut at *camera module v3* sitt grensesnitt mot RPien ikke var det samme som de tidligere versjonene av modulen, og derfor ikke var kompatibel med kodebibliotekene vi benyttet. Av denne grunnen ble modulen byttet ut med en *camera module v2*. Denne modulen har ikke den optiske bildestabiliseringen v3 modulen tilbyr. På dette tidspunktet var systemet kommet langt nok til å kunne teste videostrømmingen, og enkle tester viste at systemet var funksjonelt også uten stabilisert kjørekamera.

Arducam Stereo USB Camera Under utvikling av bilens kamerasystem var det ønske om å integrere et IR-kamera i tillegg til et dagslys-kamera. Det ble derfor kjøpt inn et *Arducam Stereo USB Camera, Synchronized Visible Light and Infrared Camera* fra Arducam. [13] Denne modulen består av to kameraer. Et IR- og et dagslys-kamera. Dette gjør det mulig å velge hvilket av kameraene operatøren ønsker å benytte seg av under speiding.

Spenningsmåler

For valg av spenningsmåler ble det satt to krav til elektronikkmodulen; modulen måtte passe på RPi-stacken for enkel montering og tilkobling, og den måtte tåle en

batterispenning på over 12.6V. At den skal være i stand til å måle batterispenninger uten noen modifikasjon gjør at det ikke kan benyttes en enkel ADC-HAT da disse typisk ikke kan måle spenninger høyere enn tilførselsspenningen, hvilket her er 5v. Av disse grunnene valgte vi å bruke en 4-channel Power Monitor HAT fra Waveshare. [6] Denne benytter INA219 IC-er for å måle strøm og spenning, og er i stand til å måle spenninger opp til 26v.

PWM HAT

Motorene på bilen er kontrollert med PWM signaler. I planleggingsfasen av prosjektet ble det estimert at vi ville bruke mellom 4 og 7 servoer avhengig av hvilke funksjonaliteter vi ville implementere. Den valgte PWM-løsningen måtte derfor ha mulighet til å støtte opp til 7 servomotorer, og kunne tilrettelegge for fremtidige utvidelser.

Raspberry Pi 4 innehar to integrerte PWM moduler som kan brukes til servostyring. Flere PWM kanaler kan implementeres i programvaren, men en slik løsning vil kreve at PWM utgangene styres direkte av prosessoren, som kan være ressurskrevende. I tillegg er Raspbian OS ikke et RTOS og kan ikke garantere at prosesser blir utført med harde tidsfrister. Raspbian kan derfor ikke garantere at timingen til PWM prosessen blir utført med høy presisjon. [91]

Av disse grunner ble det valgt å bruke en ekstern PWM modul til generering av PWM signaler og servostyring. Vi valgte en *16-Channel PWM HAT* fra Adafruit fordi de mange kanalene tilrettelegger for fremtidige utvidelser og er kompatibel med RPi plattformen.[58] Denne HATen bruker en PCA9685 IC med en intern oscillator og kan generere opp til 16 forskjellige PWM signaler samtidig.

GPIO HAT

For å gjøre tilkobling av komponenter som ikke går i HAT stabelen lettere ble det lagt til en skruterminal GPIO-HAT. [107] Denne gjør det enkelt å feste ledninger til de ledige GPIOene på Raspberry pien, og fjerner behovet for å lodde.

Av/på knapp

Det er viktig at RPier blir avskrudd på en sikker måte. Vilkårlig frakobling av strømtilførselen kan føre til datakorrupsjon i programvaren og kan føre til at operativsystemet må installeres på nytt. Dette er en tidkrevende prosess og kan derfor ikke gjennomføres ute i felt. For å kunne skru av/på Raspberry Pien på en trygg måte uten å gå inn i terminal ble det kjøpt inn et *pi-supply*[102] strømtilførsel-kort. Denne modulen gjør det mulig å gjennomføre en kontrollert avstengning eller omstart ved å sende et signal til RPien.

Batteri

Da de mest strømkrevende komponentene var bestemt ble det gjennomført utregninger på effektforbruket i forskjellige situasjoner for å se hvor lang teoretisk levetid forskjellige batterikonfigurasjoner ville gi. Resultatene av utregningene ble deretter presentert for TBRT som kom med tilbakemeldinger på hvilket av alternativene de ønsket seg, basert på tiltenkt bruk. Resultatet av dette ble at plattformen skulle designes med to 5000mAh batterier koblet i parallell for en total kapasitet på 10000mAh. Med en batterikapasitet på 10000mAh beregnet vi at plattformen ville ha batteri til å kunne kjøre full fart i 27 minutter, eller stå stille med videostrøm i omtrent 3.5 timer.

Det er verdt å merke seg at utregninger ble basert på maksimalt strømtrekk på komponentene, hvilket fører til at de teoretiske batteritidene som ble beregnet er en teoretisk minstetid, og vil dog være lavere enn den faktiske batterilevetiden.

Effektbudsjett			
Komponent	**Batterispenning (11.1V)**	**5V**	
Motor	18000mA*	0	[109]
Raspberry pi 4B+	0	2100mA	[85]
SIMxxxx-M2 HAT	0	3000mA	[96]
PWM HAT	0	2000mA**	[59]
DC/DC-konverter	0	1454mA***	[68]
			Sum
Effektforbruk kjørende	199.8 W	42.8 W	**242.6 W**
Effektforbruk stillestående	0	30.7 W	**30.7 W**

Tabell 1: Utregning av effektforbruk

* Strømtrekket til motoren er hentet fra databladet til motorkontrolleren
** maks strøm med 4 servoer tilkoblet.
*** omformingen til 5v har en effektivitet på 83% hvilket fører til et tap på 1454mA ved et strømtrekk på 7100mA (Kjørende) og 1044mA ved et strømtrekk på 5100mA (Stillestående).

Levetid beregning					
Tilstand	**Effektforbruk**	**Levetid**			
		5000mAh	6400mAh	10000mAh	12800mAh
Kjørende	242.6 W	13.7 min	17.6 min	27.5 min	35.1 min
Stillestående	30.7 W	108.4 min	138.7 min	216.8 min	277.5 min

Tabell 2: Utregning av batterilevetid

DC/DC Omformer

For valg av DC/DC-omformeren regnet vi ut effekten omformeren måtte være i stand til å levere basert på maksimalt strømtrekk hos de mest strømkrevende komponentene på 5v. På bakgrunn av dette fant vi at omformeren måtte håndtere mer en 36W. Med

dette effektkravet, samt et ønske om ekstramargin for eventuelle senere utvidelser bestemte vi oss for en spenningsomformer på 50W. Videre måtte også omformeren inneha hensiktsmessige fysiske dimensjoner og skruterminaler for enkel terminering av ledninger. Basert på disse kravene fant vi DC/DC omformeren *DFR0929* fra DFRobot. [68]

Xbox-Kontroller

Et av hovedmålene med denne oppgaven er å kunne styre roveren med stor grad av nøyaktighet. Roveren må kunne navigere gjennom trange korridorer og dørkarmer, samt styres rundt rot, avfall og andre hindringer som kan være tilstede under slike omstendigheter. Dette er spesielt viktig når roveren skal operere som en speider i områder der redningspersonell ikke har et visuelt overblikk over situasjonen. Dette krever at farten og kjøreretningen til roveren må kunne styres med et høyt nivå av presisjon.

Det er også viktig at brukergrensesnittet er veldig brukervennlig og lett kan benyttes av personell med minimal teknisk erfaring og opplæring. Roveren blir utviklet i samarbeid med DroneSAR avdelingen innen TBRT og utvikles med tanke på deres behov og tilbakemeldinger. De har etterspurt at roveren skal være lett å styre og operere. TBRT hadde ingen spesifikke preferanser når det gjaldt hvordan plattformen skulle kontrolleres, så lenge den var lett å bruke.

Basert på disse kriteriene ble det valgt å kontrollere roveren med en standard Xbox kontroller. Kontrolleren bruker analoge styrespaker for å oppnå en høy presisjon og er et kjent grensesnitt for mange. Xbox kontrollene fyller også kravene om brukervennlighet. Xbox kontrollerene som selges i dag har gjennomgått flere år med iterasjoner og testing for å gi kontrolleren en ergonomi som er komfortabel å holde og lett å bruke.

Andre grunner til at Xbox kontrolleren er godt egnet til formålet er at Windows operativsystemet kommer innebygd med drivere for kontrolleren. Dette fører til at den er lett å koble til en PC. Kontrolleren kan også brukes med enten kabel eller trådløs bluetooth-tilkobling avhengig av behov. [126]

Det er også noen nedsider ved bruk av Xbox kontrolleren, hovedsakelig at kontrolleren bruker AA batterier. Dette utsetter brukeren for en risiko av at kontrolleren går tom for strøm under en redningsoperasjon. Denne risikoen kan ungåes ved å bruke kontrolleren med kabel.

4.4 3D-modellering

Det er 3D modellert og printet spesialtilpassede deler for å kunne montere elektronikkmodulene til Traxxas TRX4 rammen.

Alle festeanordninger for elektronikk benyttet i dette arbeidet har blitt spesialdesignet og 3D-printet. På grunn av ikke-eksisterende 3D modell for TRX4 er all modellering utført på bakgrunn av fysiske målinger gjort med skyvelære. Dette medførte en

lengre, og i større grad iterativ designprosess, hvor deler måtte testprintes opp til flere ganger før de passet på den faktiske modellen av kjøretøyet.

Alle printede deler er printet ved Sintef Digital sin Makerspace, med printere av typen Prusa i3 Mk3. Benyttet materiale er PLA.

4.5 Testmetodikk

For testing av plattformen er det ønskelig å samle inn empiriske- eller kvantitative resultater for hvordan de forskjellige delsystemene fungerte, samt helhetlige resultater på bruk av plattformen. For å oppnå dette ble det laget en rekke forskjellige tester presentert under.

Dataoverføring

Ett av de viktigere delsystemene å få konkrete resultater fra er dataoverføringen mellom plattformen og brukeren, ettersom dette er avgjørende for hvor god kontroll brukeren har.

For å få testet dette ble det laget tre forskjellige tester som ble gjentatt flere ganger under ulike scenario.

Nettverksforbruk

Nettverksforbruket ved videostrømming ble testet ved å bruke programmet *bashtop*.[60] Dette programmet tillater overvåkning av ressursforbruk i RPien, herunder også nettverksforbruk. Ved å se på informasjonen i *bashtop* under videostrømming og kjøring var vi derfor i stand til å se nettverksforbruket.

Signalforsinkelse

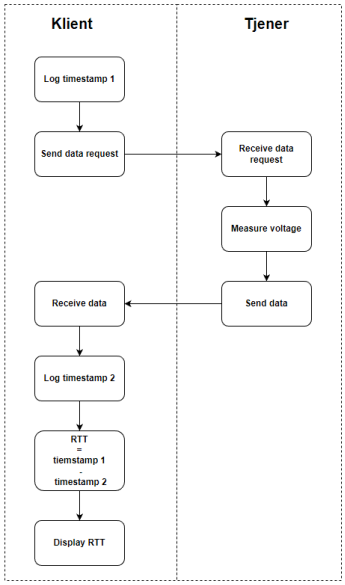
Forsinkelsen i dataoverføringen, både styresignalene og videostrømmingen, er svært viktige for kontrollen av plattformen ettersom dette påvirker hvor raskt brukeren er i stand til å se og reagere på hindringer o.l. under kjøring.

For testing av signalforsinkelsen ble det laget to ulike tester. En test for styresignal og en for video. Testene ble utført på de to forskjellige mobildatamodulene. Hver av testene ble gjennomført på både 4G og 5G for hver modul.

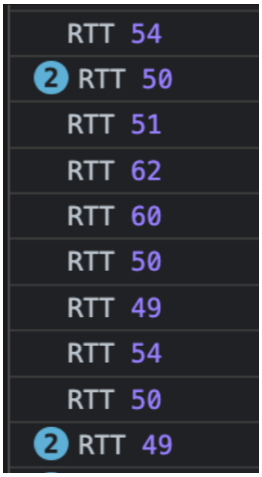
Forsinkelsen på styresignalet ble testet ved at vi la inn tidstempling når vi sendte og mottok data i klienten. Med dette kunne vi sende en data-etterspørsel fra klienten og måle hvor lang tid det tok før svaret kom tilbake.

Ved å dele den målte tiden på to vil man få et estimat på signalforsinkelsen en vei. Dette estimatet vil ikke være helt korrekt, da det ikke tar i betraktning den tiden det tar å utføre målingene. Men ettersom prosesseringen av styresignalene også vil

ta lit tid, vil dette være et godt nok estimat på hvor fort plattformen reagerer på endringer i styresignalene.

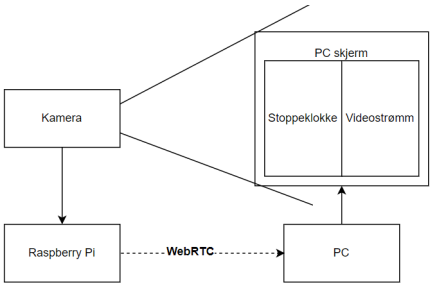


(a) Metode for å teste forsinkelse på styresignal

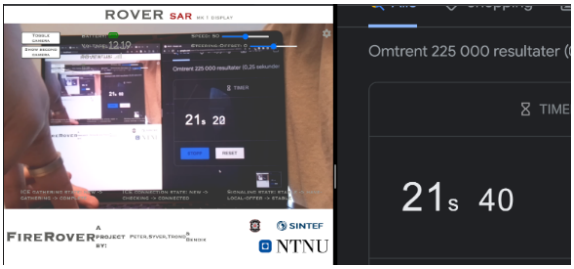


(b) eksempel på RTT målinger

For å teste forsinkelsen i videostrømmen ble bilen satt opp til å filme en PC skjerm. PC skjermen viser en stoppeklokke, i tillegg til videostrømmen fra bilen. Ved å ta skjermbilder av PC skjermen og sammenligne tiden på videostrømmen mot tiden på stoppeklokken, vil man kunne finne tidsforsinkelsen i videostrømmen.



(a) Metode for å teste forsinkelse på videostrøm



(b) Oppsett for testing av videoforsinkelse

Strømtrekk

For å måle strømtrekket i drivkraftmotoren til bilen er det benyttet et multimeter med sikring på 10A. Ved test av motorstrømmen er det viktig å være observant på at motorens teoretiske maksimale strømtrekk er på 100A. Vi gjennomførte derfor testen med jevn akselerasjon, og kontinuerlig avlesning av multimeteret, slik at testen kunne avbrytes om vi nådde sikringens maksimalverdi.

Det blir også gjennomført strømmålinger på RPi og 5G HAT. Dette blir gjort ved å bruke en USB-Strømmåler.

Batterilevetid

Med bakgrunn i de teoretiske beregningene av batterilevetid ble det gjennomført en praktisk test for å kartlegge reell levetid under naturlige kjøreforhold.

For å teste levetiden til systemet ble to 3S lithium polymer batterier på 5000mAh ladet opp til maksimal spenning på 12.6V. Disse batteriene koblet i parallell utgjør strømforsyningen til hele systemet. Bilen ble konfigurert til å kjøre på LTE+NR5G. Dette vil si at bilen benytter seg av 5G dersom tilgjengelig, og faller tilbake på LTE om den mister 5G-dekning. Det ble kjørt med to videostrømmer simultant. Disse videostrømmene var PiCamera på 720p 30FPS ref kapittel 3.16, samt Arducam IR-kamera med 720p 9FPS ref kapittel 3.16.

For å kartlegge levetiden til batteriene loggfører vi batterispenningen, tidsstempel, og forskjellige variabler som påvirker strømforbruket. Disse variablene er styrevinkel, motorpådrag samt bilens valgte gir.

Demonstrasjonstester

Demonstrasjoner har blitt utført for å fremvise funksjonalitet av systemet for eksterne aktører som kan ha interesse av å se hva en slik plattform kan tilby. I tillegg til *kontordemonstrasjoner* hos brannvesenet ble det planlagt og gjennomført en demonstrasjon under brannøvelse.

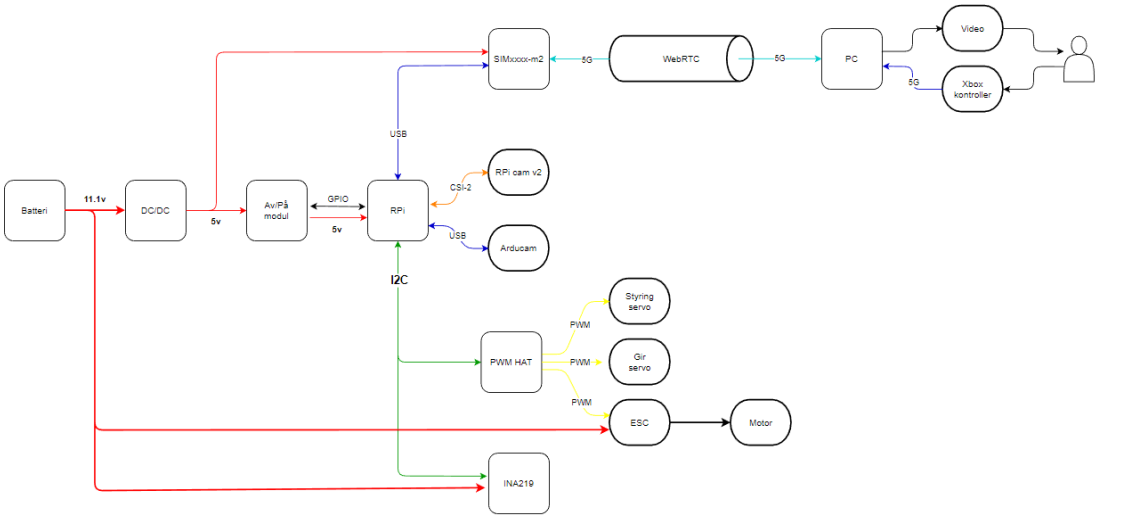
Brannøvelse - Strømnestunnelen 26.04 Den 26.04 deltok prosjektgruppen under brannøvelse i Strømnestunnelen for å gjennomføre en semirealistisk test og demonstrasjon av kjøretøyet. Mot slutten av den ordinære brannøvelsen fikk vi mulighet til å kjøre demonstrasjon og test av kjøretøyetets funksjonalitet i røyklagt tunnel. Vi installerte programvaren for å kontrollere kjøretøyet inne i TBRT sin Drone-bil. Dette ga oss tilgang til å monitorere roveren fra en større TV-skjerm montert på baksiden av bilen, slik at tilskuere enkelt kunne følge med på testen. Det ble ikke loggført noen form for data under denne demonstrasjonen.

5 Systemdesign

Systemdesign omhandler maskinvare- og programvarearkitekturen til systemet, samt detaljert beskrivelse av programvaren.

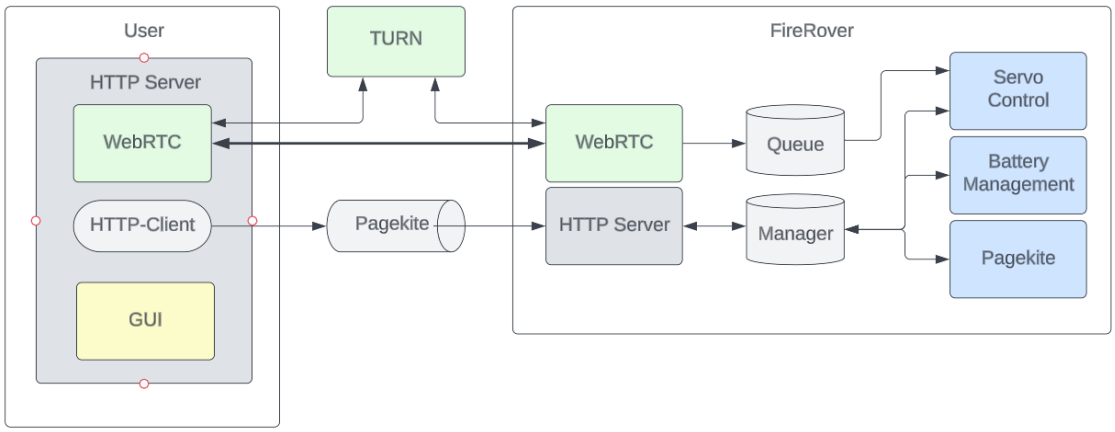
5.1 Arkitektur

I figur 32 er det fremstilt et diagram av plattformens endelige maskinvarearkitektur. Her fremstilles komponentene som er brukt, samt signalflyten gjennom systemet.



Figur 32: Maskinvarearkitekturen til plattformen

Videre i figur 33 er systemets programvarearkitektur illustrert. Her vises de forskjellige prosessene, og hvordan prosessene samhandler.



Figur 33: Programvarearkitekturen til plattformen

5.2 Utviklingsprosess

Tidlig i arbeidsprosessen ble utvikling av programvare delt opp i fire hoveddeler. Disse delene var 5G, videostrømming, brukergrensesnitt, og plattform-kontroll. Ansvaret for hver hoveddel ble fordelt på gruppemedlemmene. Programvaren til mobildatamodulen kunne isoleres, og var derfor uavhengig fra de andre hoveddelene. De resterende hoveddelene hang sammen og måtte samarbeides på for å få til å snakke sammen.

Utviklingen av programvare for mobildata baserte seg på SIMXXXX-M2 HAT, og mobildatamodulen SIM8200EA-M2, hvilket vi fikk utdelt av TBRT ved oppstart. Det ble utviklet programvare for å kommunisere med SIMCOM sine 8200-serie mobildatamoduler. [96]

Under samtaler med TBRT ble det etterspurt mulighet til å implementere video- og datastrømming via Incendium, for bedre integrering i TBRT sitt eksisterende datasystem. [48] Det ble tatt kontakt med Incendium og muligheten for integrering med deres tjenester ble etterspurt. Incendium ga tilbakemelding om at integrering var mulig, men at WebRTC protokollen måtte benyttes for å koble videostrømmer til deres servere. Gruppen gikk senere bort fra planen om å integrere plattformen med Incendium, hovedsaklig fordi integreringen ble for ressurskrevende for oppgavens begrensede tidsramme. Gruppen valgte å fortsette med bruk av WebRTC protokollen til video- og datastrømming ettersom teknologien er sikker, velprøvd, har korte ventetider og kommer med flere eksempelprosjekt som man kan lære av og ta inspirasjon fra. Bruk av WebRTC legger også tilrette for fremtidig integrering med Incendium.

Etter det var bestemt å bruke WebRTC ble utviklingen startet på tre fronter; WebRTC tilkobling, brukergrensesnitt og kontroll av plattformen. Utviklingen av programvaren for kontroll av plattformen startet med å lage kode som kunne kjøres lokalt, som styring av servo motorer, avlesning av batterispenning, og kode for av/på knapp. Koden ble deretter testet sammen med styresignaler via websocket [108] mens WebRTC-programvaren ble utviklet, og til slutt integrert med styresignaler via WebRTC-datakanaler da dette var ferdigutviklet.

Utviklingen av WebRTC startet ved å teste ulike eksempelkoder. Mange av disse eksempelkodene var skrevet for andre plattformer enn Raspberry pi, eller var for avanserte til at gruppen var i stand til å ta de i bruk med daværende forståelse for systemet. Til slutt ble Jeremy Lainés Aiortc bibliotek for python tatt i bruk. [74] Dette er et *open-source* bibliotek for WebRTC i python, skrevet for å være enkelt å forstå og bruke. Biblioteket har mange tilgjengelige eksempler. Ett av disse eksemplene ble brukt som utgangspunkt for videreutvikling av systemet, til koden hadde den funksjonaliteten vi var ute etter.

Til å begynne med baserte utviklingen av brukergrensesnittet seg på å lage et python-kontrollert vindu med en innebygget videostrøm fra WebRTC. Dette ble gått bort fra da det var tilgjengelig eksempelkode fra aiortc-biblioteket vi kunne basere et nettleserbasert brukergrensesnitt på, som ikke var like komplekst. Denne eksempelkoden ble først forandret til å ha nødvendig funksjonalitet for styresignaler og valg av kamera. Brukergrensesnittet ble deretter utviklet grafisk, for å oppnå et tilfredsstillende visuelt design.

5.3 Programvare

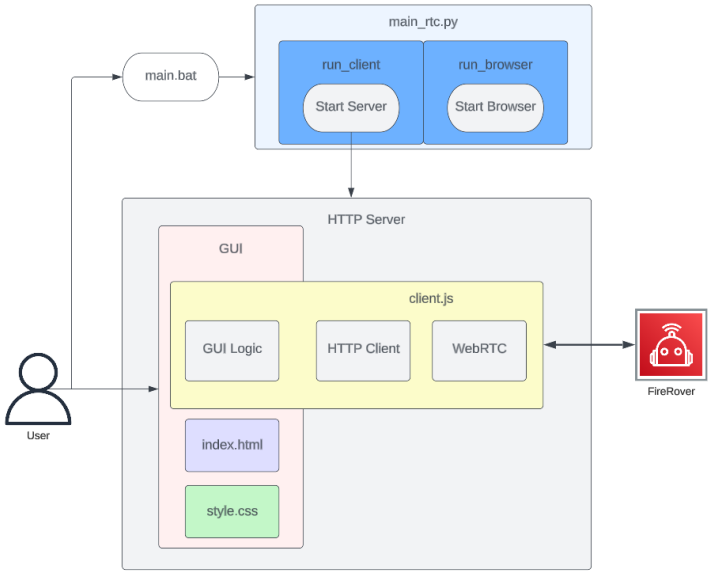
Prosjektets programvare kan deles inn i to deler, brukerdelen og plattformdelen. Begge delene består av en hovedfil som bruker Python sitt multiprosesseringsbibliotek for å kjøre flere programmer i parallell uten at de påvirker hverandre. Denne arbeidsmetoden gjør det lettere å integrere kode uten konflikter.

5.4 Programvare brukerside

Brukerdelen består av et brukergrensesnitt som blir brukt av operatøren til å kontrollere plattformen og en databaseside hvor prosessering skjer. En oversikt over brukersiden kan bli sett i figur 34.

main.bat er en BATCH fil som skal gjøre det lett å starte programmet uten å måtte bruke windows-konsollen. **main_rtc.py** bruker python multiprocessing til å kjøre to prosesser. Serverprosessen starter en lokal HTTP-tjener som er ansvarlig for å styre brukergrensesnittet og opprette kommunikasjon med plattformen. HTTP-tjeneren består hovedsaklig av tre filer. **index.html** er en html-fil som inneholder alt av innhold til nettsiden. **style.css** er en css fil og er ansvarlig for formatering og utseende. **client.js** er en JavaScript fil som er ansvarlig for implementering av logikken til nettsiden. Dette inkluderer bl.a. funksjonene som blir utført når en trykknapp aktiveres, samt opprettelse av kommunikasjon med roveren. HTTP-tjenereren blir som standard koblet til *localhost* port 8000.

Den andre prosessen er nettleserprosessen og vil automatisk åpne en nettleser til samme lokalhost port som HTTP-tjeneren bruker.



Figur 34: Programvarearkitektur Brukerside

Brukergrensesnitt

Brukergrensesnittet er bygd med HTML, CSS og JavaScript. Målet var å designe noe som var enkelt i bruk, informativt og oversiktlig, i henhold til bestilling fra kunde, samt følge designprinsipper presentert i kapittel 3.5.

Som presentert i kapittel 5.2 ble rammeverket Aiortc benyttet for å opprette webRTC forbindelse. I dette rammeverket fantes også eksempelkode med enkle brukergrensesnitt som dannet grunnlaget for videre utviklingsarbeid av brukergrensesnittet. Vi har tatt utgangspunkt i *webcam* og *server* eksemplene.[75]

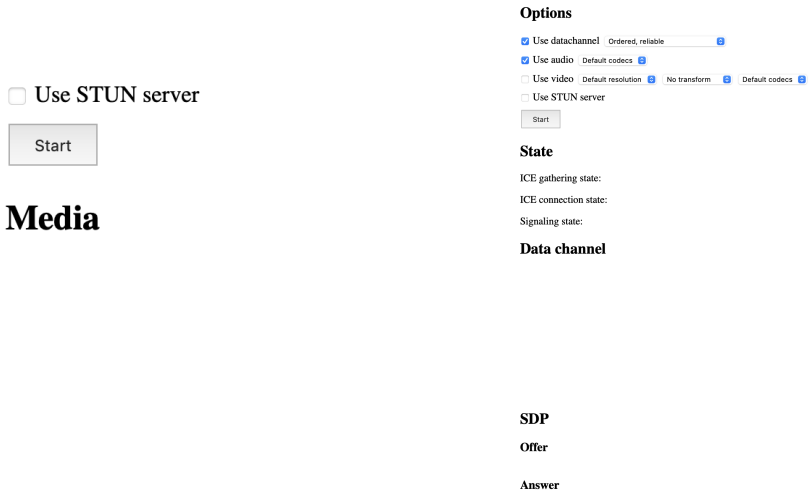
```
<body>
[...]
<div id="media">
    <h2>Media</h2>

    <audio id="audio" autoplay="true"></audio>
    <video id="video" autoplay="true" playsinline="true"></video>
</div>

<script src="client.js"></script>
</body>
</html>
```

Kodeutklipp 1: Aiortc Webcam-eksempel[75]

I kodeutklipp 1 og figurene 35a og 35b kan omfanget til eksempelkoden sees. Kodeeksempelet viser oppbyggingen av start og stopp -knapp, samt etablering av mediavindu. Eksempelkoden er ment å gi enkel tilgang til funksjonen i resten av kodebiblioteket, og er ikke særlig egnet i et sluttprodukt. Ved å slå sammen de ovenfornevnte eksemplene fikk vi et grunnleggende brukergrensesnitt som lot brukeren konfigurere oppkoblingen, se tilkoblingstilstand, samt overføre data og video.



(a) Webkamera-eksempel hentet fra eksempelkode[74]

(b) Server-eksempel hentet fra eksempelkode[74]

Figur 35: Eksempel-html

Til innledende testing av forbindelse fungerte dette, men behovet for flere funksjoner og et ønske om å designe et helhetlig produkt førte til at vi valgte å bruke tid og ressurser på å videreutvikle et nytt brukergrensesnitt med bedre brukervennlighet og mer funksjonalitet.

I utviklingen av brukergrensesnittet var det viktig å møte de overordnede designkravene skissert i oppgavens problemstilling, med brukervennlighet som et av de viktigste punktene. I tillegg til å møte designkravene var det viktig for prosjektgruppa at grensesnittet kunne fungere effektivt under testarbeidet.

Struktur Grensesnittets utforming tok utgangspunkt i at en videostrøm har størrelsesforholdet 16:9(standard for 1920x1080 eller 1280x720 oppløsning). For at informasjon ikke skal forstyrre video er den plassert i topp- og bunntekst. Disse er utformet som overlegg og er plassert over videostrømmen. Disse utformes ved å sette bakgrunnsfargen i et element til å være transparent. Her er overleggene designet med en gjennomsiktighetsgrad på 35%.

Informasjonstilgang For å lage et hensiktsmessig brukergrensesnitt må en viss mengde informasjon være umiddelbart tilgjengelig for brukeren.

Tilkoblingsstatus, forsinkelse, batteristatus, hastighet- og styringsforskyvning kan alle leses av og/eller kontrolleres fra grensesnittets overlegg. Disse elementene ble ansett som essensielle i bruken av kjøretøyet og derfor prioritert implementert. Avlesning og kontroll av verdiene skjer i samarbeid mellom index.html og client.js filene. Ved å gi hvert element i index-filen en særegen ID kan client-filen, om aksessert som et skript i index-filen, tilegne- og avlese verdier i det aktuelle elementet med korresponderende ID. I kodeeksempel 2 kan vi se elementene *battery* og *voltage* inneha hhv. *battery* og *voltage* ID, mens kodeeksempel 3 viser logikken som endrer farge og verdi i de to elementene. Funksjonen oppdaterer verdiene i HTML-dokumentet, og basert på om aktuell spenningsverdi er over eller under gitte terskelverdier, settes batteri-ikonet til å være grønt, gult eller rødt. Dersom batteriet blir rødt får brukeren beskjed om lav batterispenning via en alarm-boks satt av alert() funksjonen. Dette for å sørge for at brukeren umiddelbart får varsel om viktig informasjon.

```
    <ul id="system">
        <li>
            Battery: <span id="battery"></span>
            <span id="charging" class="fa fa-lg ">&#xf244;</span>
        </li>
        <li>
            Voltage:
            <span id="voltage" >__</span>
            <span>V</span>
        </li>
    </ul>
```

Kodeutklipp 2: Avlesing av batteri

```
function chargebattery(voltage) {
    var a;
    a = document.getElementById("charging");
    v = document.getElementById("voltage");
```

```
    v.innerHTML = voltage;
    if (voltage > 12) {
        a.innerHTML = "&#xf240;";
        a.style.color = "green";
    [...]
    } else {
        a.innerHTML = "&#xf243;";
        a.style.color = "red";
        if(!battery_low){
            alert("Battery is low, please charge the battery");
            battery_low = true;
    [...]
```

Kodeutklipp 3: Eksempel på endring av statusfarge hos batteri

For å gi brukeren tilbakemeldinger under interaksjonen med grensesnittet er et ofte brukt designvalg å gi visuelle indikatorer på om et element er aksesserbart eller ikke. Dette for å gi tilbakemeldinger til brukeren på om det brukeren tenker skal skje faktisk vil skje. I vårt brukergrensesnitt er dette gjort ved hjelp av verktøytips og fargeindikatorer når musepekeren rører et aksesserbart element.

Verktøytips bygges ved å lage to klasser, hhv. en for elementet som skal vise verktøytipset(*tooltip*), og et for elementet som skal vises(*tooltiptext*). I kodeeksemplene 4 og 5 under ser vi nødvendig kode. Kommandoen *visibility* gjør teksten i elementer med klassen .tooltipstext synlige når et .tooltip element blir berørt av en musepeker.

```
    <div class="tooltip">
        <span id="setupSymbol" class="fa-solid fa-gear fa-xl setup"
    onclick="popupOpener()"></span>
        <span class="tooltiptext">Setup</span>
    </div>
```

Kodeutklipp 4: Eksempel på kode for tooltip, HTML del

```
.tooltip .tooltiptext {
    visibility: hidden;
    [...]
}
.tooltip:hover .tooltiptext {
  visibility: visible;
}
```

Kodeutklipp 5: Eksempel på kode for tooltip, CSS del

For å redusere mengde informasjon i grensesnittet er all informasjon som ikke er relevant under aktiv bruk flyttet til et eget innstillinger-vindu. I dette vinduet er det mulig å justere tilkoblings- og videoinnstillinger, samt knapper for tilkobling og omstart av kjøretøy. Vinduet er bygd som en popup og dukker opp ved initiell innlasting av nettsiden, men kan også åpnes fra egen knapp. For å bygge et popup-vindu kreves det både HTML, CSS og Java. Identifikasjonstaggen *system-popup* tilegnes objektet som er tiltenkt å fungere som popup.

```
<div id="system-popup"class= "popup">
    <button id="closePopup">&times;</button>
    [...]
    <a href="#" onclick="start(), popupCloser()">Connect</a>
```

```
</div>
```

Kodeutklipp 6: Eksempel på kode for popup, HTML-del

Videre kreves det logikk i form av funksjoner for å endre tilstanden på elementet fra usynlig til synlig. I kodeeksempel 7 ser vi to funksjoner. Den første funksjonen open() er satt til å åpne elementer med klassen popup 2000ms etter nettsiden lastes inn. Funksjon to kan kalles på fra elementer og lukker det aktuelle vinduet. Dette gjøres ved å sette *style.display =* til *block* elle *none*.

```
<script type="text/javascript">
    window.addEventListener("load", function(){
        setTimeout(
            function open(event){
                document.querySelector(".popup").style.display = "
   block";
            },
            2000
        )
    });
    [...]
    function popupCloser() {
        var popup = document.getElementById("system-popup");
        document.querySelector(".popup").style.display = "none";
    }
</script>
```

Kodeutklipp 7: Funksjoner for popup-funksjonalitet

Stilisering Det er i arbeidet forsøkt å skape et moderne brukergrensesnitt, en nettside som er robust og som takler ulike oppløsninger og nettlesere. Dette er gjort ved å ha et bevisst forhold til bruk av farger, fonter og symboler, samt skalerbarhet i designet.

Nettsiden er satt opp med tre fonter i prioritert rekkefølge, som vist i 8 under. *Copperplate* ble valgt som font for prosjektet, mens *Arial* og *Sans-serif* er reservefonter som sikrer kompatibilitet med flere nettlesere dersom en nettleser eller pc ikke har tilgang på *Copperplate*. Disse verdiene er satt for hele dokumentet. *Copperplate* innehar en behagelig, men moderne visuell stil som passet godt inn med resten av grensesnittet.

```
* {
    [...]
  font-family: Copperplate, Arial, sans-serif;
}
```

Kodeutklipp 8: Font-prioritering for grensesnitt

Nettsiden er forsøkt bygd skalerbar med bruk av *flexbox* elementer. Dette var hensiktsmessig å implementere i blant annet overlegg-elementene i grensesnittet, da de på bakgrunn av mengden informasjon må kunne skaleres avhengig av størrelsen på nettleservinduet. Ved å benytte *justify-content og flex-grow* kan man sikre at elementene holder riktig avstand samtidig som at de har mulighet til å redusere egen størrelse. Implementeringen av dette for overlegget kan sees i kodeutklipp 9.

```
.system-overlay  {
    [...]
    display: flex;
    justify-content: space-around;
    [...]
}
.system-overlay ul{
  flex-grow: 1;
  [...]
}
```

Kodeutklipp 9: Flexbox i overlay-elementer

Xbox-kontroller

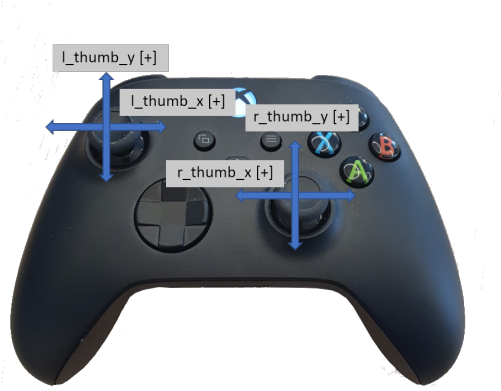
Xbox-Kontrolleren består av to analoge styrespaker, to analoge avtrekkere og 12 boolske knapper. De forskjellige funksjonene som kontrolleren skal styre har blitt fordelt blant kontrollergrensesnittet.

- **l_thumb_x: Styreretning** Rover styreretning
- **r_thumb_y: Kamera Tilt** Styrer kamerapeking langs y-aksen
- **r_thumb_x: Kamera Pan** Styrer kamerapekingen langs x-aksen
- **RB: Kamera Lås** Trykk en gang for å låse pekbart kamera til nåværende pekeretning. Trykk igjen for å låse opp kamerapeking.
- **right_trigger: Fremover** Styrer fremoverfarten til roveren
- **left_trigger: Revers/Brems** Styrer bremsing og reverseringhastighet
- **D_PAD_UP: Høyt Gir** Setter rover til høygir
- **D_PAD_DOWN: Lavt Gir** Setter Rover i lavgir
- **D_PAD_LEFT: Styringsoffset +** Øker Styringsoffset med 1°
- **D_PAD_RIGHT: Styringsoffset -** Senker Styringsoffset med 1°
- **A: Fartsfaktor Lav** Setter fartsfaktoren til 20% av maksverdi
- **X. Fartsfaktor Middels** Setter fartsfaktoren til 60% av maksverdi
- **Y: Fartsfaktor Høy** Setter fartsfaktoren til 100% av maksverdi

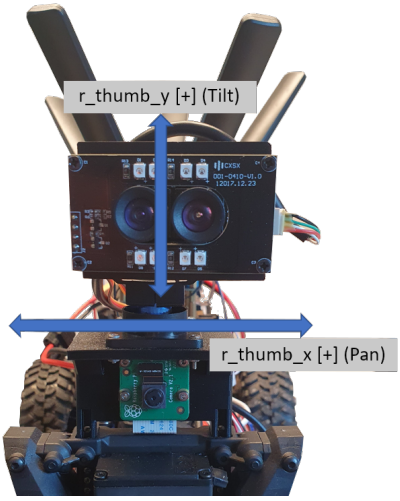
Styreretning blir styrt av styrespaken l_thumb_x. Pekeretning på de pekbare kameraene blir styrt av r_thumb_y, der y-aksen er ansvarlig for kamera *tilt* og x-aksen er ansvarlig for kamera *pan*. Kamera retningene er indikert i figur 36d.

(a) Xbox Kontroller: Knapper

(b) Xbox Kontroller: Analoge Avtrekkere og



(c) Xbox Kontroller Analoge Styrespaker. [+] indikerer positiv retning



(d) Pan-Tilt Kamera. [+] indikerer positiv retning

Figur 36: Styring fra Xbox-kontroller

For å få så lav ventetid på kontrolleren som mulig, opereres den gjennom client.js programmet, som kjører i HTTP serveren på PC-siden av prosessen. Den bruker JavaScript sin innebygde Gamepad API [99] til å initialisere kontrollerrelaterte avbrudd, og lese data fra kontrolleren. Denne implementeringen har også fordelen av å være kompatible med alle operativsystem og har blitt testet på Windows og Mac.

Når client.js startes blir det initialisert to hendelsesdetektorer. En for detektering av når kontroller blir tilkoblet (10) og en for når kontrolleren er frakoblet (11). Når disse hendelsene blir oppfattet, setter de variabelen **controller_connected** til *Sann* eller *Usann*. Startfunksjonen vil bare prøve å opprette en peer-to-peer sesjon når kontrolleren er tilkoblet. Hvis **controller_connected** er Usann vil funksjonen logge at kontrolleren ikke er tilkoblet og avbryte operasjonen (12).

```
window.addEventListener("gamepadconnected", (e) => {
    const gp = navigator.getGamepads()[e.gamepad.index];
    [...]
    controller_connected = true;
    [...]
    readController();
```

```
  });
```

Kodeutklipp 10: Hendelsedetektor: Kontroller Tilkoblet

```
window.addEventListener("gamepaddisconnected", (e) => {
    controller_connected = false;
    [...]
  });
```

Kodeutklipp 11: Hendelsedetektor: Kontroller Frakoblet

```
async function start() {
    if(controller_connected == false){
        alert("Controller not connected");
        return;
    }
    console.log("start");
    [...]
```

Kodeutklipp 12: Verifisering av kontroller-tilkobling

Funksjonen **readController()** er ansvarlig for å lese data fra kontrolleren og lagrer det i et JSON objekt. Objektet inneholder tilleggsinformasjon som brukes til å styre roveren. **topic** feltet brukes til å identifisere hva slags data JSON objektet inneholder. HTTP-serveren i roveren vil lese dette feltet og behandle objektet anderledes avhengig av hva den skal brukes til. **data** klassifiseringen indikerer at objektet inneholder data fra kontrollen og blir dermed videresendt til servostyringsprosessen. **timestamp** et tidsstempel som indikerer når objektet ble generert og kan brukes til å beregne tidsforsinkelsen til datapakkene. **buttons** inneholder de leste verdiene fra alle de boolske trykknappene. **speed_multiplier** er en brukerbestemt fartsfaktor som brukes til å begrense makshastigheten til roveren slik at den blir lettere å styre. **steering_offset** er en brukerbestemt verdi for å kompensere for evt. lening/skeivhet i styringen slik et det er lettere for føreren å kjøre roveren i en rett linje.

```
async function readController() {
    [...]
    gamepad_data = {
        topic: "data",
        timestamp: getCurrentTimestamp(),
        values: {
        buttons: {
            A: gp.buttons[0].pressed,
            B: gp.buttons[1].pressed,
            X: gp.buttons[2].pressed,
            Y: gp.buttons[3].pressed,
            D_PAD_UP: gp.buttons[12].pressed,
            D_PAD_DOWN: gp.buttons[13].pressed,
            D_PAD_LEFT: gp.buttons[14].pressed,
            D_PAD_RIGHT: gp.buttons[15].pressed,
            LB: gp.buttons[4].pressed,
            RB: gp.buttons[5].pressed,
            left_stick_button: gp.buttons[10].pressed,
            right_stick_button: gp.buttons[11].pressed,},
        left_trigger: gp.buttons[6].value,
        right_trigger: gp.buttons[7].value,
        l_thumb_y: gp.axes[1],
        l_thumb_x: gp.axes[0],
        r_thumb_y: gp.axes[3],
```

```
        r_thumb_x: gp.axes[2],
        speed_multiplier: speed_multiplier,
        steering_offset: steering_offset,}}
    [...]
```

Kodeutklipp 13: Lesing av kontrollerstatus og lagring som JSON objekt

ReadController() funksjonen inneholder også logikken for **speed_multiplier** og **steering_offset** variablene. D_PAD knappene på kontrollen kan brukes til å justere styringsoffseten til en valgfri vinkel mellom -15°og +15°. Trykknappene A, X og Y brukes til å sette fartsfaktoren til en av tre forhåndsbestemte verdier for lav, medium og høy fart. Både styreoffset og fartsfaktor kan også velges direkte på nettsiden hvis det er foretrukket.

```
    if(gamepad_data.values.buttons.D_PAD_RIGHT) {
        if(steering_offset < 15){
            steering_offset += 1;
            [...]
        }
    }
    if(gamepad_data.values.buttons.D_PAD_LEFT) {
        if(steering_offset > -15){
            steering_offset -= 1;
            [...]
        }
    }

    if(gamepad_data.values.buttons.A) {
        speed_multiplier = 20;
        [...]
    }
    if(gamepad_data.values.buttons.X) {
        speed_multiplier = 60;
        [...]
    }
    if(gamepad_data.values.buttons.Y) {
        speed_multiplier = 100;
        [...]
    }
```

Kodeutklipp 14: Styringsoffset og fartsfaktor logikk

For å sende data til roveren blir det opprettet en **setInterval** objekt som gjennomfører en rekke instrukser i et fast tidsintervall på 25ms. Først kjører den **readController()** funksjonen for å hente data fra kontrollere. Deretter sjekker den om kontroller-verdien er lik eller forskjellig fra den forrige leste verdien. Hvis den er forskjellig blir JSON objektet sendt til roveren via WebRTC datakanelen.

```
    dc.onopen = function() {
        dcInterval = setInterval(function() {
            try
            {
                readController()
                .then(res => {
                    if(res != old_controller_data){
                        dc.send(JSON.stringify(res));
                        old_controller_data = res;
                    }
```

```
                })
            }
        [...]
        }, 25);
```

Kodeutklipp 15: Sending av kontrollerdata til rover

5.4.1 Start lokal HTTP-server

Ved start av programvaren på brukersiden vil det startes en lokal HTTP-server som hoster brukergrensesnittet. Denne HTTP-serveren er satt opp til å tillate cors (3.8) slik at man kan hente inn ressurser fra plattformen.

```
class CORSRequestHandler (SimpleHTTPRequestHandler):
    def end_headers (self):
        self.send_header('Access-Control-Allow-Origin', '*')
        SimpleHTTPRequestHandler.end_headers(self)
```

Kodeutklipp 16: Tillat cors fra alle kilder

```
from http.server import HTTPServer, SimpleHTTPRequestHandler, test
[...]
def start_client():
    [...]
    print("server starting as http://localhost:8000/")
    test(CORSRequestHandler, HTTPServer, port=8000)
```

Kodeutklipp 17: Start HTTP-server

5.4.2 Hent ICE-servere

Ved oppstart av HTTP-serveren på klientsiden vil koden kjøre en funksjon for å hente inn ICE-Servere med brukernavn og passord fra Twilio. Dette er nødvendig ettersom brukernavnet og passordet til ICE-serverene forandrer seg daglig. Denne funksjonen fungerer ved at det sendes en HTTP etterspørsel til Twilio sin API med en Twilio brukers ID og passord. Gitt at IDen og passordet stemmer vil APIen returnere en liste med tilgjengelige ICE-servere med brukernavn og passord. Listen med ICE-servere blir deretter lagret til en fil lokalt på enheten så de kan brukes senere.18

```
def get_ice():
    account_sid = "<ssid>"
    auth_token = "<token>"
    client = Client(account_sid, auth_token)
    token = client.tokens.create()
    ice = token.ice_servers
    prased_ice = json.dumps(ice)
    return prased_ice
```

Kodeutklipp 18: Kode for å hente ICE-servere

5.4.3 Start videostrømming

Når *connect*-knappen i brukergrensesnittet trykkes vil funksjonen **Start** i client.js kjøres. Denne funksjonen er for å starte WebRTC kommunikasjonen mellom enhetene, og derfor starte både videostrømming og kjøring. Denne koden vil først kjøre **createPeerConnection()**-funksjonen som initialiserer *WebRTC peer-connection* lokalt. Deretter vil den sette opp WebRTC datakanalen og funksjonene som sender og motar data over denne. Til slutt kjører den **negotiate()**-funksjonen som oppretter WebRTC-kommunikasjonen mellom enhetene.

```
async function start() {
    [...]
    console.log("Creating peer connection");
    pc = await createPeerConnection();
```

Kodeutklipp 19: Initialiser WebRTC peer-connection

```
var parameters = JSON.parse('{"ordered": false}');
    console.log("Creating data channel");
    console.log("parameters", parameters);
    dc = pc.createDataChannel('chat', parameters);
    console.log("data channel", dc);
    dc.onclose = function() {
        [...]
    };
    dc.onopen = function() {
        [...]
    }
    dc.onmessage = function(evt) {
        [...]
    }
```

Kodeutklipp 20: Setter opp datakanal

createPeerConnection()

Som nevnt i 5.4.3 initierer funksjonen **createPeerConnection()** WebRTC peer-connection lokalt. Dette gjør den ved å først konfigurere ICE-serveren som skal benyttes, deretter starte peer-connection lokalt, for så og sette opp *EventListeners* for de forskjellige funksjonene i *peer-connection*.

For å konfigurere ICE-serveren leser den av innstillingene for ICE i brukergrensesnittet. Disse innstillingene bestemmer om den skal bruke ICE-serveren satt opp med CoTurn (3.2) eller ICE-serveren som er hentet fra Twilio. Om bruk av CoTurn er valgt vil funksjonen bruke nettverksadresse, brukernavn og passord til CoTurn-serveren som er lagt inn i koden. Er CoTurn ikke valgt, vil programmet lese filen med ICE-servere fra Twilio som ble lagret ved oppstart av den lokale HTTP-serveren (5.4.2) og bruke disse.

```
if(COTURN==true) {
        console.log("COTURN");
        config.iceServers = [
            {
                urls: 'stun:x.x.x.x:3478' // IP-addres to CoTurn
   server
```

```
            },
            {
                username: 'turnuser',
                credential: 'turnTBRT',
                urls: 'turn:x.x.x.x:3478' // IP-addres to CoTurn
    server
            }];
        console.log(config.iceServers);
    }
    else {
        config.iceServers = ice_options;
        var ice_options = readTextFile("./ice_servers.txt");
        console.log("ice_options", ice_options);
    }
```

Kodeutklipp 21: Valg av ICE-server

Koden initierer så et JavaScript **RTCPeerConnection()** objekt med ICE-serveren. Deretter legger den til *EventListeners* med funksjoner for ICE-tilkobling, tilstanden og video/audio strømmer.

```
pc.addEventListener('track', function(evt) {
        console.log(evt);
        if (evt.track.kind == 'video'){
            if(video_array_index == 0){
                console.log("got video track1");
                console.log(evt.streams[0]);
                //document.getElementById('media1').style.display =
    'block';
                video1 = new MediaStream([event.track]);
                document.getElementById('video1').srcObject =
    video1;
                video_array_index++;
            }
            else if ( video_array_index == 1){
                [...]
            }
            else{
                console.log("no more video tracks");
            }
            console.log("started video");
        }
        else
            document.getElementById('audio').srcObject = evt.
    streams[0];
    });
```

Kodeutklipp 22: EventListeners på WebRTC peer-connection

negotiate()

Negotiate-funksjonen starter WebRTC-kommunikasjonen med plattformen. Dette gjør den ved å først legge til sender- og mottakerelementer for datastrømmene som skal gå over WebRTC. Disse elementene inneholder informasjon om typen og retningen på datastrømmen.

```
function negotiate() {
```

```
[...]
pc.addTransceiver('video', {direction: 'recvonly'});
pc.addTransceiver('video', {direction: 'recvonly'});
pc.addTransceiver('audio', {direction: 'recvonly'});
[...]
```

Kodeutklipp 23: Legge til WebRTC sender- og mottaker-elementer

Videre genererer koden WebRTC-tilbudet som skal sendes til plattformen. Dette tilbudet inneholder informasjon om datastrømmene koden er ute etter, samt hvilke STUN/TURN-servere klienten benytter. Tilbudet inneholder også unike koder som brukes til å identifisere WebRTC-kanalen fra brukeren.

```
[...]
return pc.createOffer().then(function(offer) {
    return pc.setLocalDescription(offer);
    [...]
    console.log("creatng offer");
    console.log( pc.localDescription);
    var offer = pc.localDescription;
    var codec = "H264/90000";
    offer.sdp = sdpFilterCodec('video', codec, offer.sdp);
    [...]
```

Kodeutklipp 24: Generere WebRTC-tilbud til plattformen

Det neste koden gjør er å sende en forespørsel til STUN/TURN-serverene for å finne ut hvor i nettet den står, og hvordan plattformen kan snakke med brukeren. Informasjonen som returneres fra STUN/TURN-serverene legges så til WebRTC-tilbudet til plattformen.

```
[...]
.then(function() {
    // wait for ICE gathering to complete
    console.log("Waiting for ICE gathering to complete");
    return new Promise(function(resolve) {
        if (pc.iceGatheringState === 'complete') {
            resolve();
        }
        else {
            function checkState() {
                console.log("checkState");
                if (pc.iceGatheringState === 'complete') {
                    pc.removeEventListener('
icegatheringstatechange', checkState);
                    resolve();
                }
            }
            pc.addEventListener('icegatheringstatechange',
checkState);
        }
    });
    [...]
```

Kodeutklipp 25: Forespørsel til STUN/TURN-serverene

Etter at WebRTC-tilbudet er generert og STUN/TURN-informasjonen er lagt til blir det sendt til plattformen. Dette gjøres med en HTTP POST kommando (3.8).

Med denne HTTP-kommandoen blir det også sendt innstillinger for kameraene fra menyen i brukergrensesnittet. Dette gjør den ved å sende HTTP-kommandoen til nettverkstunnel-tjenesten *pagekite*. Koden venter deretter på responsen fra POST-kommandoen. Denne inneholder svaret fra plattformen med en mottatt melding, informasjon om hvilke datastrømmer plattformen sender, formatet på overført data, og unike koder for å identifisere WebRTC-kanalen fra plattformen.

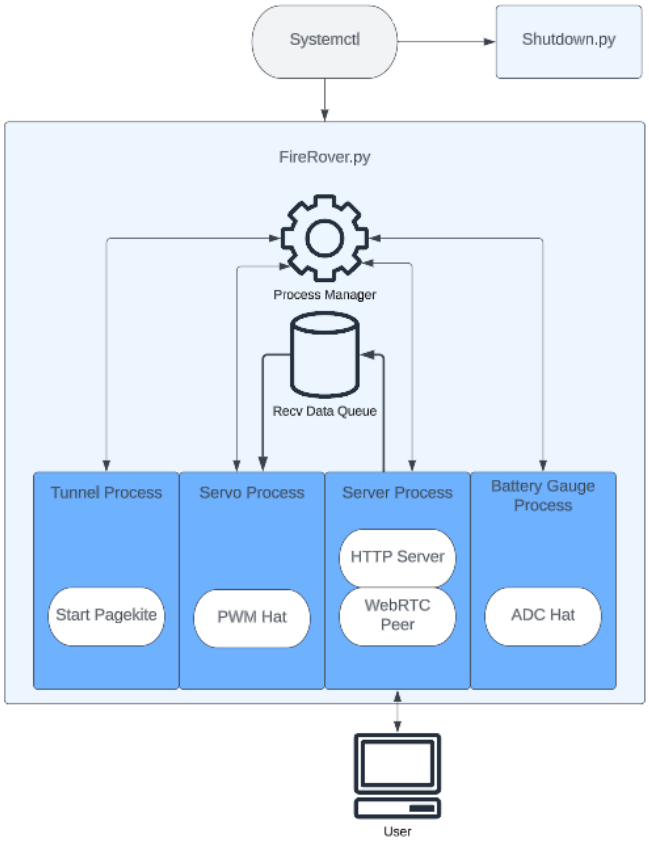
```
    [...]
console.log("Sending offer");
        return fetch('https://firerover.pagekite.me/offer', {
            body: JSON.stringify({
                sdp: offer.sdp,
                type: offer.type,
                video_fps: document.getElementById('video-fps').
   value,
                video_source: document.getElementById('video-source
   ').value,
                video_resolution: document.getElementById('video-
   resolution').value
            }),
            headers: {
                'Content-Type': 'application/json'
            },
            method: 'POST',
            cache: "no-cache",
            redirect: "follow",
            credentials: "omit"
        });
        [...]
```

Kodeutklipp 26: Send WebRTC-tilbud med HTTP POST

```
    [...]
console.log(response);
        if (!response.ok) {
            console.log("Network response was not OK");
          }
        try {
            return response.json();
          } catch (error) {
            console.log("error parsing input");
          }
    }).then(function(answer) {
        console.log("answer");
        return pc.setRemoteDescription(answer);
        [...]
```

Kodeutklipp 27: Respons fra HTTP POST-kommando

5.5 Programvare plattformside



Figur 37: Programvarearkitektur Plattformside

En oversikt over programvarearkitekturen til plattformen kan bli sett i figur 37. Ved RPi oppstart vil systemctl automatisk starte **Shutdown.py** og **FireRover.py**. **Shutdown.py** er et program som er ansvarlig for styring av RPi on/off modulen. Den vil kontinuerlig lese verdien fra inngangen koblet til soft-shutdown knappen, og skru av eller restarte RPien når knappen trykkes inn. Den er også ansvarlig for styring av indikatorlyset på On/Off modulen for indikering om RPIen er av eller på. **FireRover.py** bruker python multiprosessing til å opprette fire prosesser, en mp_queue og en mp_manager.

Recv_Data_Queue er et mp.Queue() objekt som er ansvarlig for å overføre mottatt kontrollerdata fra WebRTC datakanalen i serverprosessen, til Servostyringsprocessen. En datakø-struktur er godt egnet til formålet ettersom det er ønskelig med lav forsinkelse mellom bruker og rover.

Process Manager er et mp.Manager() objekt som brukes til å overføre status-data og styresignaler på tvers av underprosessene. 28 viser en oversikt over hvilke parameter er inkludert i objektet. **battery_gauge** og **battery_voltage** lagres av batterimålingsprosessen, overføres til serverprosessen, og sendes til brukersiden via datakanalen slik at brukeren kan vite batterispenningen til roveren. **motor_speeed**, **steering_angle** og **rover_gear_angle** blir lagret i servostyringsprosessen og lagret til en loggefil. **SYS_RESET** blir skrevet til av serverprosessen når roveren mottar

tilbakestillingskommandoen fra brukeren.

```
firerover_data_template = {
    "battery_gauge":0,
    "battery_voltage":0,
    "motor_speed":0,
    "steering_angle":0,
    "rover_gear_angle":0,
    "SYS_RESET":False,
    "Server_RESET":False
    }
```

Kodeutklipp 28: Process-Manager parametere

Tunnelprosessen er ansvarlig for å opprette kommunikasjon med pagekite-tjeneren. Dette gjør det mulig for http-klienten på brukersiden å sende forespørseler til http-tjeneren på plattformsiden.

Servoprosessen leser kontroller-data lagret i *Recv Data Queue*, og bruker verdiene til å styre servoene som er koblet til PWM-hatten.

Battery Gauge prosessen bruker ADC-hatten til å lese spenningen på batteriet og lagerer måleverdiene i **manager** objektet.

Serverprosessen er ansvarlig for styring av WebRTC og en HTTP-tjener.

5.5.1 WebRTC-plattform

Koden for WebRTC på plattformsiden ligger i **Server.py** Denne koden tar seg av oppsett av WebRTC, svar på tilbud fra klienten, oppsett av kameraene og mottak fra WebRTC datakanal.

Hovedfunksjonaliteten til koden blir styrt av funksjonen **start_server()**. Denne funksjonen setter opp en lokal HTTP-server (som tillater cors) med funksjoner koblet til underdomenene til serveren.

```
def start_server():
    [...]
    app = web.Application()
    app.on_shutdown.append(on_shutdown)
    app.router.add_post("/offer", offer)
    app.router.add_post("/CMD", cmd)
    cors = aiohttp_cors.setup(app, defaults={
    "*": aiohttp_cors.ResourceOptions(
        allow_credentials=True,
        expose_headers="*",
        allow_headers="*"
        )
    })
    for route in list(app.router.routes()):
        cors.add(route)
    web.run_app(
        app, access_log=None, host=args.host, port=args.port,
   ssl_context=ssl_context
    )
```

Kodeutklipp 29: Start HTTP-server lokalt på plattformen

Offer

offer() funksjonen er knyttet til underdomenet **/offer** og blir kalt når HTTP-serveren mottar en HTTP kommando til *ServerDomene/offer*. Når dette skjer vil funksjonen lese inn dataen i HTTP kommandoen og oversette den til JSON format. Denne kommandoen skal være en POST kommando fra brukersiden som inneholder WebRTC tilbudet. Etter dataen er oversatt til JSON vil funksjonen lese ut WebRTC tilbudstypen, tilbudsdataen og kamerainnstillingene fra datapakken. Deretter vil den initialisere WebRTC lokalt.

```
async def offer(request):
    print("offer started")
    body = await request.text()
    body_json = json.loads(body)
    params = await request.json()

    offer = RTCSessionDescription(sdp=params["sdp"], type=params["
   type"])

    pc = RTCPeerConnection()
    pc_id = "PeerConnection(%s)" % uuid.uuid4()
    pcs.add(pc)
```

Kodeutklipp 30: Lese ut datapakken som blir sendt til /offer

Etter WebRTC er initialisert vil koden initialisere kameraene med innstillingene som ble sendt sammen med WebRTC-tilbudet.

```
options = {"framerate": str(params["video_fps"]), "video_size": str
   (params["video_resolution"])}
    video_source = str(params["video_source"])
    if video_source == "IR Camera":
        video_source = find_cam(IR_CAM)
    elif video_source == "RGB Camera":
        video_source = find_cam(RGB_CAM)
    elif video_source == "Pi Camera":
        video_source = find_cam(PI_CAM)

    player1 = MediaPlayer(find_cam(PI_CAM), format="v4l2", options=
   options)
    player2 = MediaPlayer(find_cam(video_source), format="v4l2",
   options=options)
    recorder = MediaBlackhole()
```

Kodeutklipp 31: Initialisering av kameraer

Etter dette setter koden opp WebRTC-datakanalen. Koden setter datakanalen opp til å videresende mottatt data til andre prosesser basert på emnet til dataen.

```
@pc.on("datachannel")
    def on_datachannel(channel):
        @channel.on("message")
        def on_message(message):

            try:
                message_json = json.loads(message)
                if message_json["topic"] == "data_request":
                    logging.debug("Data Request Received...")
```

```
                    firerover_data_json_str = json.dumps(
    firerover_data.copy()).replace("<", "\\u003c").replace("\u2028",
     "\\u2028").replace("\u2029", "\\u2029")
                    channel.send(f"{firerover_data_json_str}")
                    logging.debug(f"Sending Data: {
    firerover_data_json_str}")

                elif message_json["topic"] == "data":
                    recv_data_queue.put(message)
                    logging.debug(f"message: {message}")
            except:
                logging.error("Message Recv Error")
```

Kodeutklipp 32: Initialisering av datakanal

Koden legger så til videostrømmene fra kameraene til WebRTC-tilkoblingen, og svarer på HTTP-kommandoen.

```
    pc.addTrack(player1.video)
    pc.addTrack(player2.video)
    print("video fps is:",str(params["video_fps"]))
    print("video sorce is:",str(params["video_source"]))
```

Kodeutklipp 33: Legge til kameraer i WebRTC

```
await pc.setRemoteDescription(offer)
    await recorder.start()

    # send answer
    log_info("sendAnswer")
    answer = await pc.createAnswer()
    await pc.setLocalDescription(answer)
    return web.Response(
        content_type="application/json",
        text=json.dumps(
            {"sdp": pc.localDescription.sdp, "type": pc.
    localDescription.type}
        ),
    )
```

Kodeutklipp 34: Legge til kameraer i WebRTC

Finn kamera

Filstien til kameraene på Raspberry Pien lages på nytt hver gang Raspberry Pien skrus av og på. Dette gjør at kameraene flytter på seg avhengig av hvilken kameramodul som starter først. For å løse dette ble det laget en funksjon som søker gjennom alle tilgjengelige filstier for kameraer, og knytter disse til riktig kamera basert på kameraets navn.

```
IR_CAM = "USB_IR"
RGB_CAM = "VIS"
PI_CAM = "service"

def find_cam(cam):
    cmd = ["/usr/bin/v4l2-ctl", "--list-devices"]
    out, err = Popen(cmd, stdout=PIPE, stderr=PIPE).communicate()
```

```
    out, err = out.strip().decode("utf-8"), err.strip().decode("utf
   -8")
    for l in [i.split("\n\t") for i in out.split("\n\n")]:
        if cam in l[0]:
            return l[1]
    return False
```

Kodeutklipp 35: Kode for å finne riktig filsti til kameraer

CMD

I tillegg til */offer* er HTTP-serveren også satt opp til å ta imot HTTP-kommandoer som kommer til /CMD. Funksjonen som kalles når det kommer en HTTP kommando hit er laget for å kunne kommunisere med plattformen når WebRTC ikke fungerer. Hvis datapakken som blir sendt til funksjonen inneholder en *'REBOOT'* kommando vil plattformen restartes.

```
async def cmd(request):
    body = await request.text()
    body_json = json.loads(body)
    if body_json["CMD"] == "REBOOT":
        reboot_cmd = ["sudo", "reboot"]
        #call(reboot_cmd, shell=False) # Initiate OS Reboot
        firerover_data["SYS_RESET"] = True
```

Kodeutklipp 36: Kode for å restarte plattformen via HTTP

5.5.2 Batterimåling

For å måle batterispenningen ble det utviklet en driver til å styre ADC-Hatten på RPien. Dette ble gjort ved å først lage en klasse for INA219 sensorene og deretter lage en ADC_Hat klasse som er bygd opp av flere INA219 sensor objekter og styringslogikk. Deretter blir et ADCHat objekt initialisert i batterimåleprosessen for måling av batterispenningen.

INA219

Når INA219 objektet blir initialisert brukes SMBus-biblioteket til å initialisere en av I^2C bussene på RPi-en, hvis den ikke allerede har gjort det. **BUS**-parameteret bestemmer hvilken av bussene som blir initialisert og **address**-parameteret bestemmer I^2C adressen til objektet.

Etter bussen er konfigurert opprettes det et sett med lokale variabler som lagrer den nåværende konfigurasjonen til INA sensoren. De kan bli endret med **set_config()**-funksjonen. Denne gjør det mulig å velge nye konfigurasjonsverdier, og lagre dem både lokalt og i sensoren. Den andre kolonnen i kodeutklippet består av konstanter, og er standardverdiene som sensoren har ved oppstart.

```
class INA219():
    def __init__(self, BUS:int=1,address:int=0x40):
```

```
        self.address = address
        self.bus = SMBus(BUS)
        sleep(1)
        [...]
        self.bus_voltage_range   = BusVoltageRange.RANGE_32V
        self.gain                = Gain.DIV_8_320MV
        self.badc_resolution     = ADCResolution.ADCRES_12BIT_1S
        self.sadc_resolution     = ADCResolution.ADCRES_12BIT_1S
        self.mode                = Mode.SANDBVOLT_CONTINUOUS
```

Kodeutklipp 37: INA219 Initialiseringsfunksjon

INA219 sensoren benytter seg av 16-bit registre for lagring av data. Siden I^2C bussen opererer med 8-bit lange datasegmenter er det nødvendig med instrukser som kan lese registeret i to omganger for å hente en fullstendig måleverdi. **read_word_data()** funksjonen blir brukt til å lese dataordet fra sensoren, men funksjonen antar at dataordet er fordelt blant to 8-bit registre. Dette fører til at de to 8-bit segmentene blir avlest i feil rekkefølge og må flyttes for at dataformatet skal bli riktig.

```
    def read_word(self, reg) -> int:
        [...]
        raw = self.bus.read_word_data(self.address, reg)
        data_string = "{:016b}".format(raw)
        MSByte = data_string[8:16]
        LSByte = data_string[0:8]

        data = MSByte + LSByte
        return(int(data, 2))
```

Kodeutklipp 38: INA219 Lesing av dataord

For å lese spenningen på inngangen brukes **r_voltage()** funksjonen. Denne funksjonen henter verdien lagret i **BUS_VOLTAGE** registeret i sensoren. I dette registeret er bitposisjonene [0-2] reservert til statusindikering for spenningsmåleren. Bitposisjonene [3-15] brukes til lagring av måledata. Måleverdiene blir derfor bitflyttet tre plasser for å isolere måleverdiene. Deretter blir måleverdiene multiplisert med ADC oppløsningen på 4mV.

```
    def r_voltage(self):
        [...]
        raw_bus_voltage = self.read_word(Reg.BUS_VOLTAGE)
        voltage = (raw_bus_voltage >> 3) * 0.004
        return round(voltage, 3)
```

Kodeutklipp 39: INA219 Spenningsavlesningsfunksjon

ADCHat

Når ADC_Hat objektet blir initialisert, opprettes det fire INA219-instanser som representerer de fire sensorene på kretskortet og blir tildelt samsvarende I^2C adresser. Det blir også definert en rekke standardverdier som brukes i sammenheng for sensorkalibrering for strømmåling. Verdiene kan konfigureres med **calibrate_all()** funksjonen.

```
class ADC_Hat():
```

```
    [...]
    def __init__(self):
        self.shunt_r = 0.1
        self.i_max = 0
        self.current_LSB = 0
        self.cal = 0

        self.Sensor0 = INA219(1, 0x40)
        self.Sensor1 = INA219(1, 0x41)
        self.Sensor2 = INA219(1, 0x42)
        self.Sensor3 = INA219(1, 0x43)
```

Kodeutklipp 40: Initialisering av ADC-HAT

Funksjonen **read_voltage()** brukes til å lese målespenningen fra en valgfri kanal.

```
    def read_voltage(self, channel):
        [...]
        if channel == 0:
            return round(self.Sensor0.r_voltage(), 2)

        elif channel == 1:
            return round(self.Sensor1.r_voltage(), 2)

        [...]

        else:
            return 0
```

Kodeutklipp 41: Spenningsavlesningsfunksjon

Batteriladning blir estimert ved å måle batterispenningen med **read_voltage()** funksjonen. Deretter blir ladningen estimert ut fra den målte spenningen.

```
    def battery_gauge(self, channel):
        [...]
        battery_voltage = self.read_voltage(channel)
        charge = 123 - 123 / (( 1 + ((battery_voltage/3)/3.7)**80)
    **0.165)
        if charge > 100:
            charge = 100

        elif charge < 0:
            charge = 0

        return round(charge, 1)
```

Kodeutklipp 42: Batterimåling

Batterimålingsprosessen er ansvarlig for å måle spenningen på batteriet i faste intervaller og lagre måledata til **manager_dict**, hvor det deretter blir videresendt til brukeren i WebRTC prosessen. Prosessen starter med at det blir initialisert en instans av ADC_Hat klassen som skal lese måleverdier fra sensorene. Deretter starter en evig løkke for datainnhenting og datalagring.

Prosessen inneholder også en try-except struktur som skal hjelpe med feilsøking av prosessen.

```
def batt_gauge(manager_dict, sample_frequency=1 ,logging_level=
    logging.ERROR):
     [...]
     try:
         ADC = ADC_Hat()
         ADC.calibrate_all()
         while True:
             gauge = ADC.battery_gauge(0)
             voltage = ADC.read_voltage(0)
             manager_dict["battery_gauge"] = gauge
             manager_dict["battery_voltage"] = voltage
             [...]
             time.sleep(1/sample_frequency)
     except IOError:
         logging.error("ADC Hat not connected to $I^2C$")
```

Kodeutklipp 43: Batterimålingsprosessen

5.5.3 Servo-kontroll

Servostyringsprosessen opererer ved å hente verdien fra **Recv Data Queue** objektet, parser JSON objektet og overfører riktig styreverdi til den spesifiserte servoen. Prosessen inneholder også logikk for manipulering av styreverdiene for å gjøre bilen mer brukervennlig. All serevostyringslogikk bli implementert i **servo_control()** funksjonen.

Data lagret i datakøen består av råe JSON-strenger. For at dataobjektet skal bli lettere å håndtere blir python sitt json bibliotek brukt til å omdanne json-strengene til python-dict objekt.

Funksjonen bruker to **try-except** strukturer for å gjennomføre sikker feilhåndtering av prosessen. Den første strukturen er ansvarlig for detektering om rover-objektet blir initialisert uten problemer. Den vanligste feilen som oppstår er en IOError og den oppstår når RPien ikke klarer å opprette I^2C-kommunikasjon med PWM-hatten. Hvis feilen oppstår blir en feilmelding logget til konsollen med forklarende tekst. Den indre try-except strukturen blir brukt til å øke påliteligheten til programmet ved å forhindre at ikke-kritiske feilmeldinger permanent stopper servostyringsprosessen. Dersom en feil oppstår under tolkningen av JSON datapakken eller skriving av feil data til en servo, vil Python returnere en feilmelding. Try-strukturen vil detektere feilmeldingen og automatisk hoppe til except-strukturen som automatisk setter roverfarten til 0 for å sette den i en sikker tilstand der den ikke kan skade seg selv eller andre. Etter roveren er i en sikker tilstand vil den deretter forsøke å gjennomføre en ny instruks.

Servostyringsprosessen har også innebygd overvåkningslogikk som skal automatisk sette roveren i en trygg tilstand dersom tilkoblingen blir brutt og rover slutter å motta nye datapakker. Variabelen **queue_last_write_time** inneholder et tidsstempel som indikerer når siste datapakke fra datakø objektet ble sist mottatt. Hvis roveren skal lese data fra **Recv Data Queue**, men objektet er tomt vil overvåknings logikken telle hvor lenge datakøen er tom. Hvis køen er tom i over 1 sekund vil roveren bli satt i en sikker tilstand med en fart på 0.

```
def servo_control(firerover_data, recv_data_queue, logging_level=
    logging.ERROR):
     [...]
     try:
         [...]
         rover = FireRover()
         [...]
         queue_last_write_time = time.time()
         camera_hold = False
         camera_lock = False
         while True:
             if recv_data_queue.qsize() != 0:

                 try:
                     commands_str = recv_data_queue.get() # Get data
     from queue
                     commands = json.loads(commands_str)["values"]
                     queue_last_write_time = time.time() #Reset
    Watchdog Timer

                     if type(commands) is dict:
                         [...]
                         [...]
                         [...]
                 except Exception as error:
                     logging.error(f"ServoLib Error: {error}")
                     rover.speed(0)
             else:
                 #Speed watchdog: If recv_data_queue is empy for >=
    1 second then set motor speed to 0
                 if (time.time() - queue_last_write_time) > 1:
                     rover.speed(0)
                     queue_last_write_time = time.time()

             time.sleep(0.001)

     except IOError:
         logging.error("PWM HAT not connected to $I^2C$")
```

Kodeutklipp 44: Servostyringsprosess feilbehandling

Motor Control delen er ansvarlig for styring av hastigheten til motoren. Verdien lagret i "right_trigger" og "left_trigger" nøkkelen tilsvarer fremoverhastighet og bakoverhastighet. **speed_factor** er en variabel som er ansvarlig for å begrense hastigheten til motoren for at den skal skal bli lettere å styre. Beregning av endelig hastighet blir gjort ved å summere fremoverfart og bakoverfart og multiplisere resultatet med fartsfaktoren. En kartleggingsfunksjon bli brukt til å omdanne styringsverdien til et -100% til +100% format.

Steering_factor og **Steering_control** er ansvarlig for styring av svingevinkelen til roveren. Kartleggingsfunksjonen omdanner kontrollerverdien fra en verdi på -1 til 1 til en servovinkel på 10°til 170°. Styrevinkelen til roveren blir begrenset med 10°på hver side fordi roveren ikke klarer å gjennomføre de bratte svingningene ved lave hastigheter. **steering_factor** er en variabel som er ansvarlig for å redusere maksimalt svingeradius ved høye hastigheter. Denne funksjonaliteten ble implementert for å redusere sannsynligheten for at roveren velter i høy fart.

Camera_Lock og **Camera_Control** styrer kamerapekingen på roveren. Låsefunksjonaliteten blir brukt for å låse pekekameraet til en fast posisjon hvis roveren skal titte i en bestemt retning over lengre tid. Den er implementert med rudimentær toggle-logikk. Dersom **RB"** blir trykket inn en gang vil kameraet låses og hvis den blir trykket inn igjen blir kameraet frigjort. Kartleggingslogikk blir brukt til å omdanne styrekontroller-signalet til en servoverdi mellom 0°og 180°. Kartleggingen for horisontal vinkel er invertert slik at pekevinkel til kameraet stemmer overens med retningen på analogspaken.

Gear_Control skifter roveren i høyt- eller lavgir når brukeren trykker på **D_PAD_UP** eller **D_PAD_DOWN**. Dette blir gjort ved å skifte gir-servoen til en fast vinkel avhengig av ønsket girverdi, der lavgir tilsvarer en servovinkel på 110°og høygir er en servovinkel på 0°. Lavgiret er satt til en redusert vinkel fordi servomotoren strever med å nå en vinkel på 180°. Vinkelen ble derfor redusert for å øke levetiden til servomotoren. De spesifikke vinkelverdiene ble funnet ved å prøve og feile.

```
    #Motor Control
    speed_int = int(map(commands["right_trigger"], 0, 1, 0, 100))
    rev_int = int(map(commands["left_trigger"], 0, 1, 0, -100))
    speed_factor = float(commands["speed_multiplier"])/100
    motor_speed = (speed_int+rev_int) * speed_factor
    rover.speed(motor_speed)
    firerover_data["motor_speed"] = motor_speed
    logging.debug(f"Motor Speed: {motor_speed}")

    #Steering factor
    steering_factor = 2 - math.exp(0.4*speed_factor)

    #Steering Control
    steering_int = int(map(commands["l_thumb_x"]*steering_factor,
   -1, 1, 10, 170)) #Steering angle limited to prevent stress on
   servo amd motor
    steering_offset = int(commands["steering_offset"])
    steering_angle = steering_int + steering_offset
    rover.steering(steering_angle)
    firerover_data["steering_angle"] = steering_angle
    logging.debug(f"Steering Angle: {steering_angle}")

    #Camera Lock
    if commands["buttons"]["RB"]:

        if not(camera_hold):
            logging.debug("Camera Lock Toggle")
            camera_lock = not(camera_lock)
            camera_hold = True
    else:
        camera_hold = False

    #Camera Control
    if not(camera_lock):
        tilt_angle = int(map(commands["r_thumb_y"], -1, 1, 0, 180))
        rover.camera_tilt(tilt_angle)

        pan_angle = int(map(commands["r_thumb_x"], -1, 1, 180, 0))
        rover.camera_pan(pan_angle)
```

```
    #Gear Control
    if commands['buttons']["D_PAD_UP"]:
        rover.shift_servo.set_angle(0) #Shift to HIGH gear
        firerover_data["rover_gear_angle"] = "HIGH"


    elif commands['buttons']["D_PAD_DOWN"]:
        rover.shift_servo.set_angle(110) # Shift to LOW gear
        firerover_data["rover_gear_angle"] = "LOW"
```

Kodeutklipp 45: Servostyring

5.5.4 Av/På knapp

Av/På knappen til plattformen brukes for å kunne skru av og restarte systemet gjennom programvare. For å gjøre dette er det skrevet kode som leser av knappene som er koblet til GPIO pinner. Er knappen trykket i to sekunder restarter den systemet. Er den trykket kortere skrur den av plattformen. For å kjøre koden er det laget en *.service* fil som kjører koden etter oppstart av roveren.

```
    while (bounce_filter_timer <= bounce_filter_ms): # While button
    not pressed
        if (GPIO.input(Shutdown_Input_pin) == True):
            bounce_filter_timer += 10
        else:
            bounce_filter_timer = 0

        sleep(0.01)

    sleep(2); # Sleep 2s to distinguish a long press from a short
   press

    if (GPIO.input(Shutdown_Input_pin) == False):
        GPIO.output(LED_Indicator_Pin,0) # Bring down PinEight so
   that the capacitor can discharge and remove power to the Pi
        logging.info("Shutdown started...")
        shutdown_cmd = ["sudo", "shutdown", "now"]
        if test_mode == False:
            call(shutdown_cmd, shell=False) # Initiate OS Poweroff
    else:
        logging.info("Reboot started...")
        reboot_cmd = ["sudo", "reboot"]
        if test_mode == False:
            call(reboot_cmd, shell=False) # Initiate OS Reboot
```

Kodeutklipp 46: Av/på knapper

```
[Unit]
Description=Startup of powerbuttons
After=network.target

[Service]
ExecStart=/usr/bin/python3 /home/TBRT/Bacheloroppgave/Code/shutdown
   .py
Restart=always
User=TBRT
```

```
[Install]
WantedBy=multi-user.target
```

Kodeutklipp 47: .service fil

5.5.5 Mobildatamoduler

Det er utviklet programvare som henter ut PIN- og PUK-kode fra en fil, og låser opp simkortet slik at SIM8262E-M2 får tilgang til mobilnettet. Under utvikling av programvaren ble det oppdaget at en manuell oppstart kun er nødvendig første gangen modulen kobler seg på mobilnettet. Etter en initiell oppkobling vil mobildatamodulen forsøke å skape tilkobling til mobilnettet automatisk ved oppstart. På bakgrunn av dette har vi utviklet systemet slik at bruker må fjerne PIN-koden på SIM-kortet før det tas i bruk, slik at modulen har mulighet til automatisk oppkobling.

For å initialisere mobildatamodulen kan en bruke Minicom, eller Python. Minicom er et serielt kommunikasjonsprogram som tillater kommunikasjon mellom enheter over et serielt grensesnitt.[66] Oppstart med Minicom krever at bruker skriver de serielle AT-kommandoene manuelt. Oppstart med Python kan automatiseres. Ved bruk av python må en benytte serial-biblioteket for kommunikasjon. Programvaren er basert på Waveshare sin programvare for SIM820X RNDIS Dial-up.[88]

Funksjonen for å sende serielle kommandoer tar inn kommandoen som skal utføres, prosesserer denne kommandoen til et format som er kompatibelt med mobildatamodulen, og sender kommandoen serielt over USB.

```
def ExecuteCommand(Command):
    ser.write((Command+'\r\n').encode())
```

Kodeutklipp 48: Utførelse av kommando

Funksjonen for å lese av svar fra mobildatamodulen vil ta ditt foretrukne svar som inngangsvariabel. Funksjonen leser av mottatt seriell data, og sammenligner ditt foretrukne svar med mottatt data. Funksjonen vil så returnere 1 om mottatt data er lik, eller returnere mottatt data som en tekststreng hvis de er ulike.

```
def ReadResponse(PerferredResponse):
    RecievedString = ''
    while True:
        if ser.inWaiting() > 0:
            BitsWaiting = ser.inWaiting()
            time.sleep(0.1)
            NewBitsWaiting = ser.inWaiting - BitsWaiting
            if not NewBitsWaiting:
                RecievedString = ser.read(ser.inWaiting()).decode()
                if PerferredResponse == RecievedString:
                    return 1
                else:
                    return RecievedString
```

Kodeutklipp 49: Avlesning av kommando

Funksjonen for å låse opp SIM-kort tar inn både PIN- og PUK-kode som inngangsvariabler. Funksjonen sjekker om SIM-kortet er låst opp. Hvis SIM-kortet ikke er

låst opp, vil mobildatamodulen sende svar om at det kreves enten PIN eller PUK for å låse opp tilgangen. Funksjonen vil deretter sende en seriell kommando for å låse opp mobildatamodulen med enten PIN, eller PUK-kode.

```
def UnlockSIM(PIN, PUK):
    ExecuteCommand('cpin?')
    UnlockStatus = ReadResponse("OK")
    if UnlockStatus == "+CPIN: PIN":
        ExecuteCommand("AT+CPIN="+PIN)
    if UnlockStatus == "+CPIN: PUK":
        ExecuteCommand("AT+CPIN="+PUK)
```

Kodeutklipp 50: Opplåsning av SIM-kort

For å aktivere mobildatamodulen må det sendes en seriell kommando som tilknytter mobildatamodulen til mobilnettet.

```
ExecuteCommand("AT+CUSBCFG=USBID,1E0E,9011")
```

Kodeutklipp 51: Oppstart av mobildatamodul

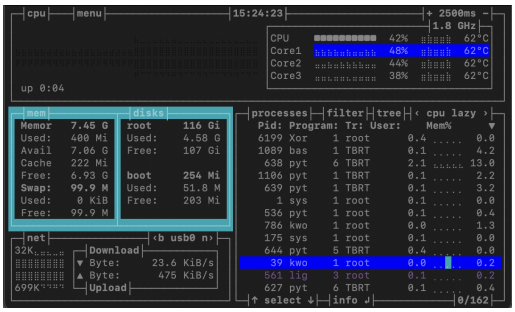
6 Resultater

6.1 Testresultater

6.1.1 Dataoverføring

Nettverksforbruk

Ved å se på informasjonen i bashtop under video strømming og kjøring kunne vi lese at RPien hadde et nettverksforbruk på 300 - 500 KiB/s



Figur 38: Bashtop mens bilen kjører og strømmer video

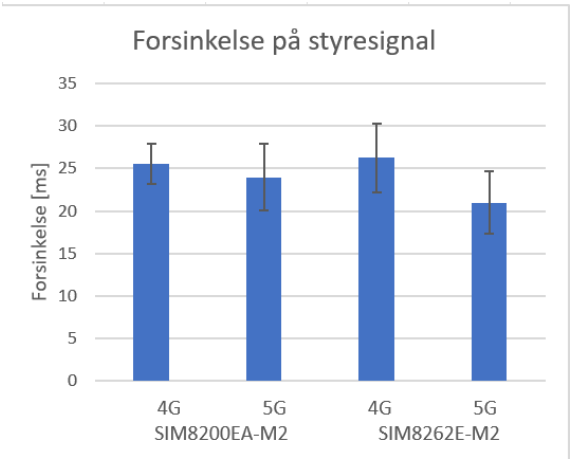
Styresignalforsinkelse

Fra testingen av styresignalforsinkelsen lagret vi 122 datapunkter per test og brukte dette til å regne ut gjennomsnitt og standardavvik. Resultatene i tabell 3 og figur 39 viser at forsinkelsen på styresignalene tilstrekkelig lave til å gi god kontroll på plattformen.

Resultatene viser også at forsinkelsen er stabil, og noe raskere på 5G enn den er på 4G.

	Forsinkelse			
Modul	SIM8200EA-M2		SIM8262E-M2	
Protokoll	4G	5G	4G	5G
Gjennomsnitt	25.6 ms	23.9 ms	26.2 ms	21.0 ms
Standardavvik	2.4 ms	3.9 ms	4.1 ms	3.7 ms

Tabell 3: Resultater fra testing av forsinkelse på styresignaler

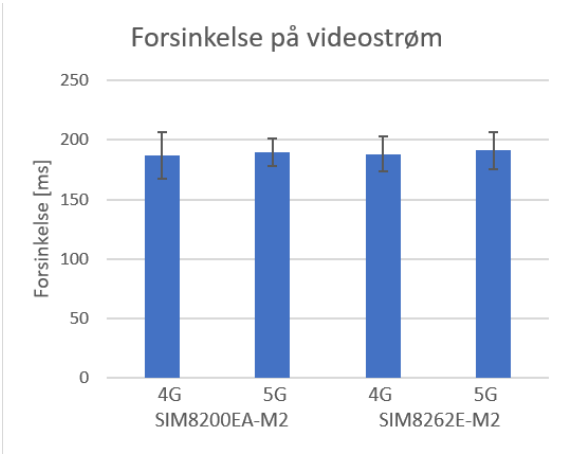


Figur 39: Resultater fra testing av forsinkelse på styresignalet

Videoforsinkelse

	Forsinkelse			
Modul	SIM8200EA-M2		SIM8262E-M2	
Protokoll	4G	5G	4G	5G
Gjennomsnitt	186.6 ms	190 ms	188 ms	191 ms
Standardavvik	19.6 ms	11.5 ms	14.8 ms	16.0 ms

Tabell 4: Resultater fra testing av forsinkelse på videostrøm



Figur 40: Resultater fra testing av forsinkelse på videostrøm

6.1.2 Strømtrekk

Målingene av strømtrekk på drivkraftmotoren viste et resultat på 3.5A under maksimal fart på første gir, samt 4.5A under 60 prosent av maksimal fart på andre gir. Dette tilsvarer henholdsvis 19.4% og 25% prosent av drivkraftmotorens reklamerte kontinuerlige strømtrekk på 18. Testen ble gjennomført med lav akselerasjon for å unngå plutselige endringer i strømforbruket. Til tross for den forsiktige akselerasjonen, ble det observert variasjoner på strømtrekket på over 1A. Resultatet viser at det

realistiske strømtrekket er betydelig lavere enn det teoretisk maksimale kontinuerlige strømtrekket på 18A. Målingene ble gjort på batteriets nominelle spenning 11.1V.

Målingene av strømforbruket på RPi viste et gjennomsnittlig forbruk på 1.0A ved streaming av video, mens strømforbruket på 5G HAT lå på 350mA. Begge målingene ble gjort på 5V DC.

Effektbudsjett etter måling			
Komponent	**Batterispenning (11.1v)**	**5v**	
Motor 1. gir	3500mA	0	
Raspberry pi 4B+	0	1000mA	
SIMxxxx-M2 HAT	0	350mA	
PWM HAT	0	0*	
DC/DC-konverter	0	276.5mA**	
Effektforbruk	38.85 W	8.13 W	**Sum 46.98 W**

Tabell 5: Effektforbruk etter test

* Ingen av servoene er under last.
** omformingen til 5v har en effektivitet på 83% hvilket fører til et tap på 276.5mA ved et strømtrekk på 1350mA.

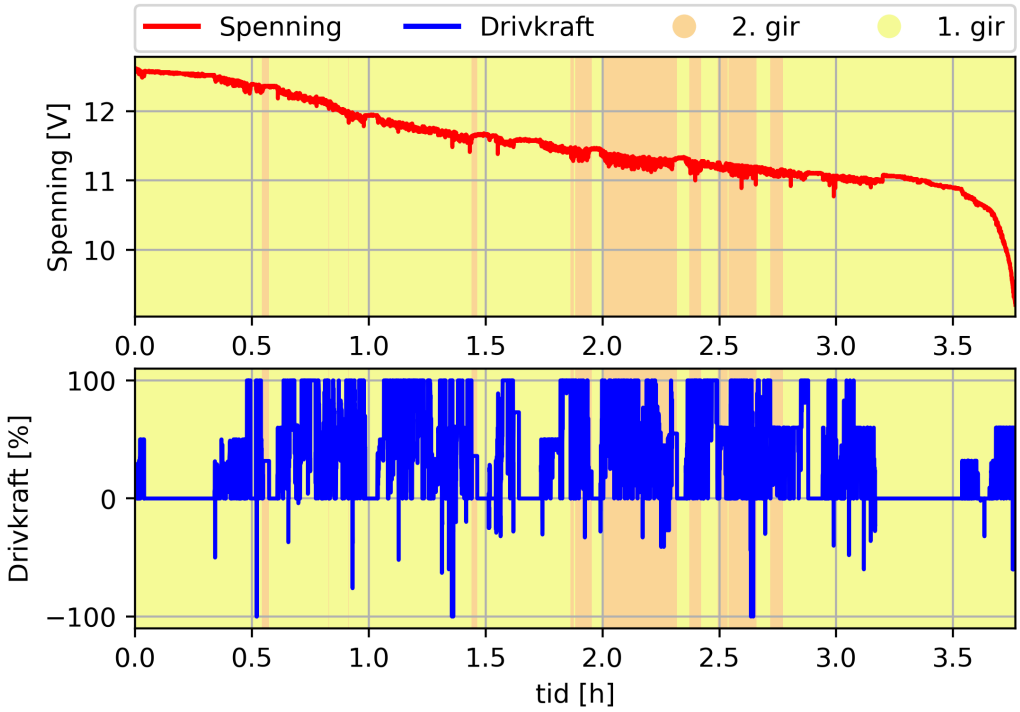
Levetid beregning				
Effektforbruk	**Levetid**			
	5000mAh	6400mAh	10000mAh	12800mAh
46.98 W	70.9 min	90.7 min	141.8 min	181.5 min

Tabell 6: Levetid med forskjellige batterier etter test

Beregning av levetid i henhold til det målte strømforbruket, samt bilens strømforsyning på 10AH, viser en kjøretid på 115.7 minutter, tilsvarende like under 2 timer.

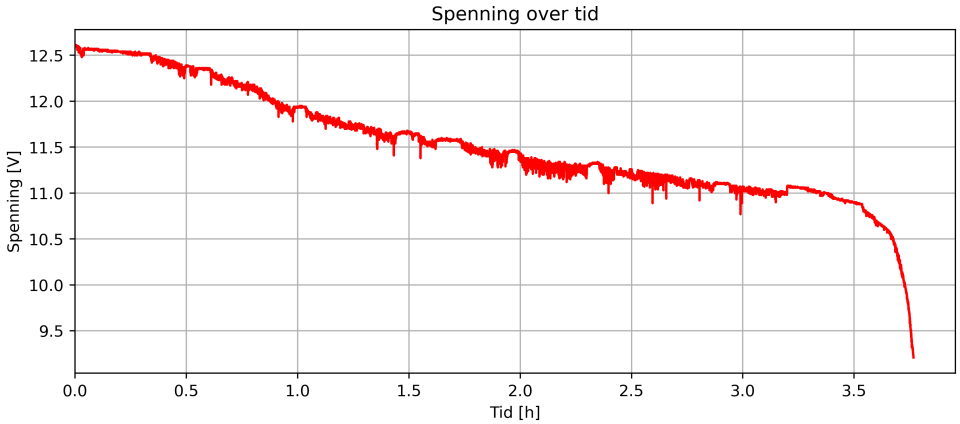
6.1.3 Batterilevetid

Resultatene fra testing av batterilevetid viser at bilen kjørte i 226 minutter med gjennomsnittsfart på 36%, der 20% av tiden var i andre gir.



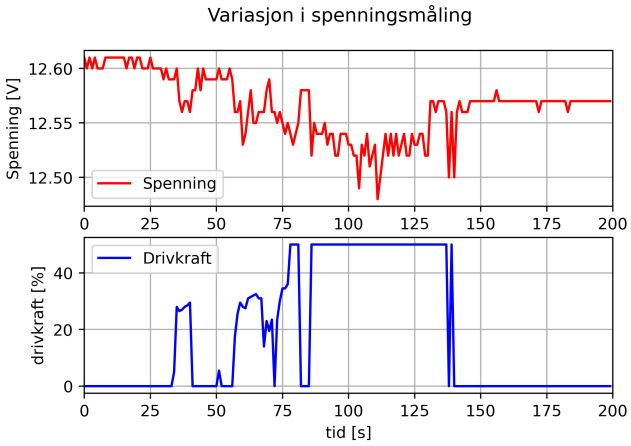
Figur 41: Graf av testdata.

De loggførte datapunktene av batterispenningen viser en karakteristisk utladningskurve for LiPo-batterier, forklart i kapittel 3.15.



Figur 42: Graf av batterispenning.

Loggføringen av batterispenning viser at batteriet synker i spenning når vi øker drivkraften på motoren, og øker når vi reduserer drivkraften. Dette er vist i figur 43.



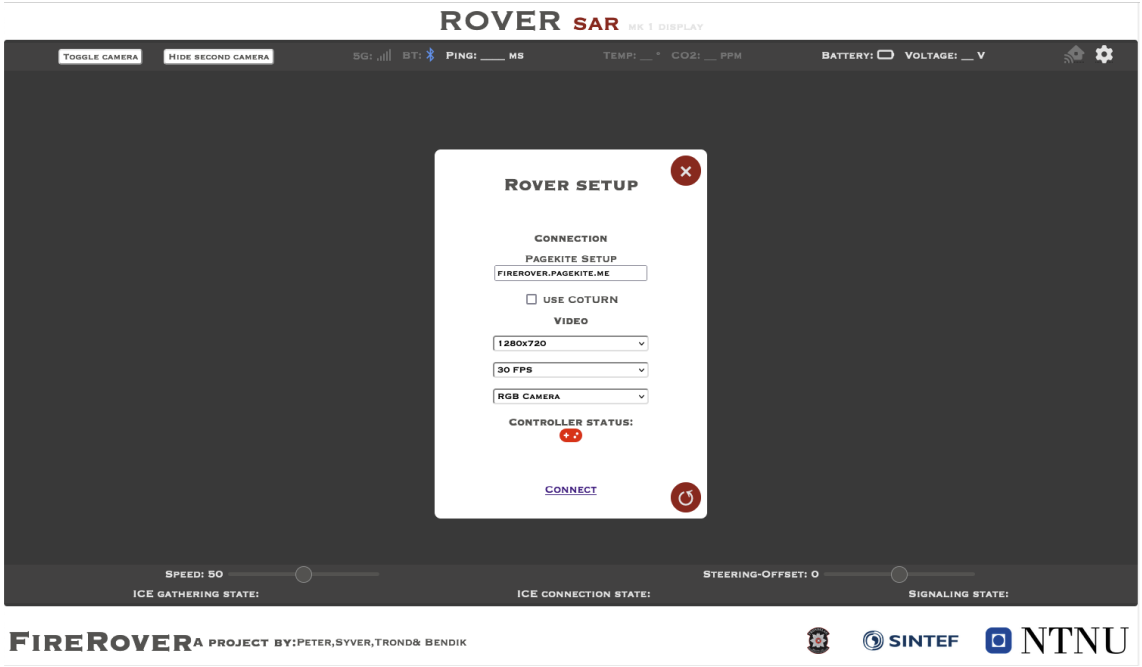
Figur 43: Variasjoner i batterispenning ved forskjellig drivkraft.

Testen ble hovedsaklig utført på asfalt, men det ble også kjørt i ujevnt terreng som grov grus, samt gressplen. Bilen viste gode resultater på alle underlag.

Testens varighet ga godt grunnlag for å observere plattformens mekaniske stabilitet. Det ble observert svakheter i festeskruene på Pan-Tilt-HAt, da disse ikke tålte vibrasjonene under kjøring, og skrudde seg selv ut. Det ble videre observert stabilitetsproblemer under høyere hastigheter. Disse stabilitetsproblemene førte til velt da bilen svinger for krapt under høye hastigheter. Under velt av bilen ble det i tillegg observert forbedringspotensiale for beskyttelse av Pan-Tilt-HAT og Arducam, dersom velt skulle oppstå igjen.

6.1.4 Brukergrensesnitt

Det ferdige brukergrensesnittet er et moderne, minimalistisk og effektivt grensesnitt som innehar essensiell funksjonalitet, og noe pynt for å skape god brukeropplevelse.



Figur 44: Landingsside

I skjermbilde 44 over ser vi det ferdige brukergrensesnittet uten videoppkobling. Nettsiden har tre deler, hhv. topptekst, brukergrensesnitt og bunntekst. Toppteksten består kun av nettsidens tittel. *ROVER SAR mk1 display*. Nettsidens midterste del, brukergrensesnittet, består også av tre deler. Det mørkegrå området i midten er området reservert til videostrømmen, mens de lysegrå områdene er overlegg som ligger over videostrømmen og innehar informasjon om bil, samt mulighet for justeringer. Bunnteksten inneholder prosjekttittel, akkreditering, samt logoer til assosierte organisasjoner. Alle navn og logoer inneholder hyperkoblinger til linkedin-profiler og organisasjonenes nettside.

I brukergrensesnittet består øverste overlegg av fem seksjoner, sortert i bokser basert på funksjonalitet. I første del har brukeren tilgang til å endre hvilke av to kamera som skal være hoved- og sekundærkamera, samt gjemme sekundærkamera om ønskelig. Seksjon to innehar oppkoblingsinformasjon. Per nåværende iterasjon av grensesnittet er det kun forsinkelsen,(*ping*), som oppdateres i sanntid. Dette er vist ved at 5G-status og bluetooth-status har en mørkere gråfarge. Denne funksjonaliteten kan implementeres senere om ønskelig, men ble ikke prioritert i dette arbeidet. Det ble derimot ansett som hensiktsmessig for helhetens skyld å legge inn i grensesnittet, for å illustrere hvordan grensesnittet kan skaleres med mer funksjonalitet. Dette gjelder også for den tredje seksjonen som har mulighet til å vise målt temperatur og CO2-partikler dersom slike sensorer er koblet til bilen.

Videre har grensesnittet visuell tilgang på batterimålinger fra bilen. For enkelhet for brukeren er det bygd inn både fargekodet nivåmåler i form av et synkende batteri, samt målt batterispenning. Dette lar brukeren enkelt og instinktivt vite hvor lang operasjonstid som gjenstår før rover bør returneres og batteri byttes.

Det øverste overleggets siste seksjon er innstillinger for oppkoblingen. Ved å trykke på tannhjulet i høyre hjørne kan brukeren åpne innstillingsvinduet. Dette vinduet er

presentert i figur 46, og diskuteres videre i neste avsnitt. For å hjelpe nye brukere å finne frem er det programert inn verktøytips som indikerer at dette ikonet tar brukeren til innstillingene ved å bevege musepekeren over ikonet. Dette er demonstrert i figur 45a og 45b. Det siste ikonet, huset med signal, er tiltenkt oppsett av Incendium og er reservert fremtidig funksjonalitet.

(a) Uten verktøytips. (b) Med verktøytips.

Figur 45: Innstillingstilgang med- og uten verktøytips ved musepekeroversvevning, her uten visibel musepeker, men fargeendring på tannhjul.

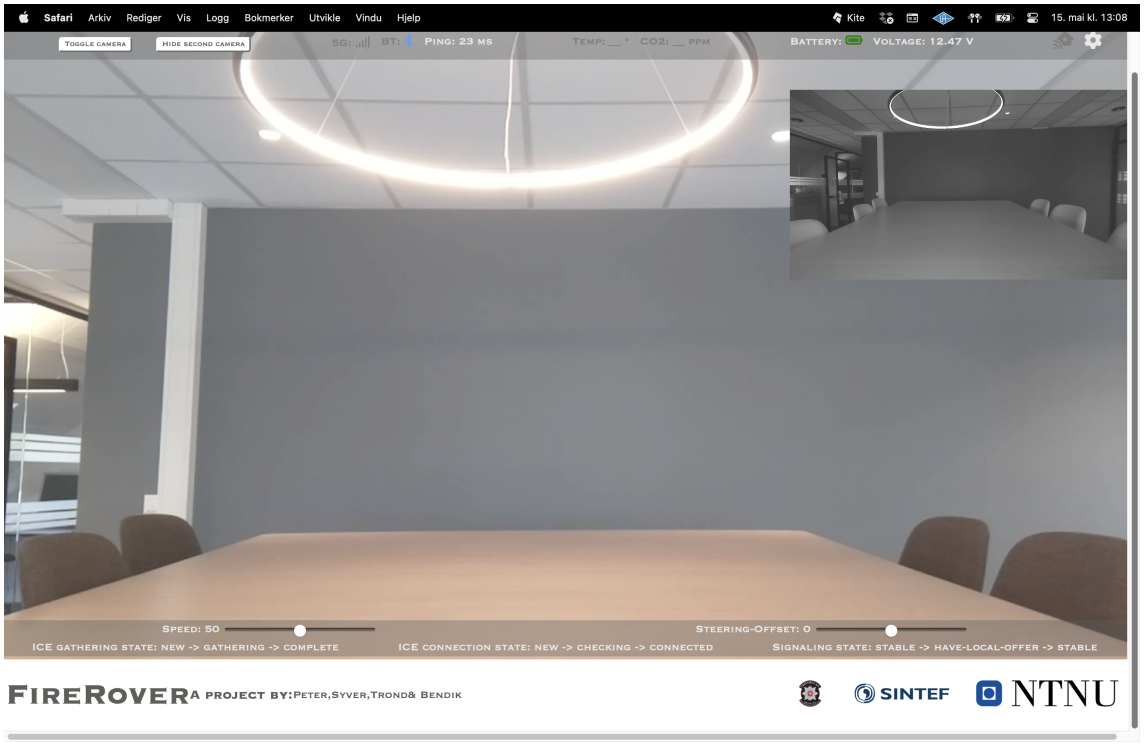
I tillegg til å kunne aksessere innstillingsvinduet fra tannhjulet i overlegget, blir det også automatisk åpnet ved innlast av nettsiden. Vinduet lar brukeren gjøre oppsett av oppkoblingen til bilen, og ett av de viktigste parameterne her er hvilken pagekite-adresse som skal benyttes, eller om CoTURN skal benyttes for å koble seg til bilen. Videre får brukeren visuell bekreftelse på om godkjent kontroller er tilkoblet, noe som kreves for å kunne starte oppkoblingen til kjøretøyet. Denne bytter fra rød til grønn dersom godkjent kontroller er koblet til. Dersom brukeren har behov for å restarte RPi-enhenten på den tilkoblede bilen kan dette gjøres ved hjelp av restart knappen nede i høyre hjørne.

Figur 46: Innstillinger for tilkobling

Det nederste overlegget innehar to glidebrytere som lar brukeren både se og justere bilens maksimale hastighet og styringsforskyvning. Begge disse kontrollene er aksesserbare fra kontroller og skjerm, og oppdateres basert på den aktuelle verdien til systemet. Under denne rekken med kontroller finner brukeren tilkoblingsstatus til bilen. Disse tre indikatorene lar brukeren se i hvilken tilstand nåværende oppkobling er.

I figur 47 kan grensesnittet sees slik det er under bruk. Verdt å nevne her er at brukeren har tilgang til å lese av; to videostrømmer, reell forsinkelse av kontrolldata, *ping*, oppgitt i ms, batteritilstand og -spenning, samt tilkoblingsstatus. All informasjon ligger i bannere med semi-transparent bakgrunnsfarge for å hjelpe brukeren å lese informasjonen under bruk.

Det er også viktig å påpeke at brukergrensesnittet er utviklet for å være kompatibelt med berøringsskjerm, da brannvesenets drone-bil har dette installert for å la observatører følge med på videostrømmen. Dette gjør det også mulig for operatøren av kjøretøyet å stå sammen med eksempelvis operasjonsleder ved bruk av en trådløs kontroller. Eksempel på dette kan sees i figur 49 senere i dette kapitlet.

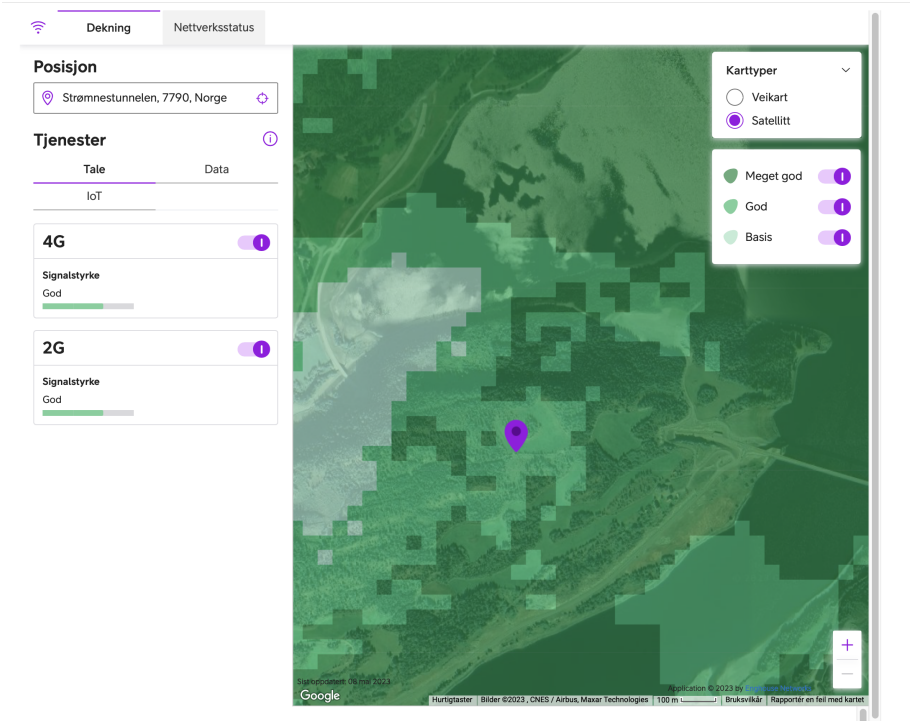


Figur 47: Brukergrensesnitt under bruk av bil.

6.1.5 Brannøvelse og demonstrasjon

Strømnestunnelen 26.04.2023 Et av hovedmålene i dette prosjektet var demonstrasjon av plattformen under en faktisk brannøvelse. Vi hadde på forhånd sjekket Telias dekningskart, og visste at det kun kom til å være tilgang på 4G-nett for oppringt-nettverk.[105] Ved oppstart av kjøretøy var det merkbart at forsinkelse på kontroll og video var høyere enn det tidligere tester i Trondheim hadde gitt.

For gjennomføringen av demonstrasjonen ble ett gruppemedlem sendt inn i tunnelen sammen med kjøretøyet, mens resten stod utenfor for å starte opp programvaren og kjøre bilen. I bilde 49 ser vi gruppemedlemmene sammen med tilskuere fra presse, brannvesenet og Nord Universitets kjøreskoleutdanning. Her ser vi også TBRT sin drone-bil i aksjon. Programvaren for roveren kjørte inne på deres datamaskin, slik at tilskuere enkelt kunne observere testen.



Figur 48: Dekningskart for Strømnestunnelen, hentet fra Telias dekningskart [105]

Figur 49: Demonstrasjon av Rover med tilskuere fra brannvesenet. Foto: ©Are Hellandsvik

Inne i røyklagt tunnel fikk vi testet kamerasikten. Bilde av skjerm kan sees i bilde 50. Merk at her er førsteutkast til brukergrensesnittet benyttet. Dette på bakgrunn av noen mindre feil i den nyere versjonen. På bakgrunn av at bilen ligger så langt nede på bakken greier den å se under mesteparten av røyken, som her er noe tettere enn den vil være under en normal brannøvelse. Her er heller ikke tunnelens viftesystemer i drift. Takket være drypp-antennene i tunnelen opplevde vi ikke tilkoblingsbaserte feil under demonstrasjonen.

Figur 50: Kamerasikt i tunnel. Foto: © Bendik Nygård

6.2 Regnskap

Basert på benyttede komponenter og materialer har vi estimert følgende enhetskostnad.

Komponent	**Pris [NOK]**
Traxxas TRX-4	7295
Raspberry Pi 4b+ 8GB	1090
SIM8262E-m2	2873
16-Channel PWN/Servo HAT	279
Current/power monitor HAT	213
PI GPIO screw terminal HAT	173
DFRobot DC/DC buck converter	447
RPi on/off module	166
RPi Camera module v2	400
Pan-tilt HAT	212
Arducam stereo USB camera	900
5000mAh LiPo batteri x2	1400
XT60 parallell tilkobling x3	300
XT60 skjøtekabel x4	300
XT60 til Traxxas overgang	90
PLA 300g	75
Sum	**16213 kr**

7 Diskusjon

7.1 Resultatdiskusjon

Strømtrekk og batterilevetid Ved test av strømtrekk på plattformen viste resultatene betydelig lavere verdier en hva som var brukt i utregninger. Dette er fordi verdiene brukt i utregningene er de maksimale verdiene for strømtrekk fra de forskjellige databladene. Strømmålingene som ble utført på Raspberry Pien samsvarer med uavhengige kilders resultater ved tilsvarende tester. [80][127][81] Tilsvarende målinger for SIMxxxx-m2 HATen er ikke tilgjengelige, men det er rimelig å anta at disse målingene også stemmer ved normalt bruk.

Ved måling av strømtrekk på motoren er det flere faktorer som gjør at dette ikke samsvarer med tallene fra databladet. Det teoretiske strømtrekket til motoren var basert på databladets høyeste kontinuerlige verdi på motorkontrolleren. Målingene av motorstrøm ble i tillegg gjort ved lav akselerasjon. Dette ble gjort fordi måleapparatet var begrenset ved 10A, mens databladet påstod et kontinuerlig strømtrekk fra motorkontrolleren på 18A. På bakgrunn av dette var det bekymring for at motorens strømtrekk kunne ødelegge måleapparatet.

Basert på utførte strømmålinger ble det regnet ut en ny teoretisk batterilevetid på omkring to timer ved kontinuerlig kjøring i maksimal hastighet. Dette er en betydelig lengre levetid en hva TBRT ønsket seg, hvilket gjør at en ved fremtidig videreutvikling kan vurdere å erstatte batteriene med mindre varianter for å spare på vekt og plass.

Levetiden på plattformen ble også validert med en praktisk test. Fra denne ble det observert en levetid på omtrentlig 3 timer og 45 minutter. Disse tallene er realistiske i forhold til utregningene som ble gjort ettersom testingen ikke ble utført med konstant maksimal hastighet.

I følge ohms lov vil spenningen over en motstand være lik motstanden multiplisert med strømmen gjennom den. Alle batterier har en indre motstand.[38] Ohms lov tilsier da at spenningen over denne indre motstanden vil være lik resistansen multiplisert med strømmen gjennom motstanden. Kjøring på høyere drivkraft tilsier høyere strømforbruk i motoren. Dette tilsier høyere strømgjennomgang i den indre resistansen på batteriet, som videre tilsier spenningstap over denne motstanden, vist i figur 43.

Ved utregning av teoretisk minstetid batteriet skal vare er det tatt utgangspunkt i at motorene kjører på maksimal fart. Dette tilsvarer full fart i andre gir. Som vi kan se fra resultatene på testen er andre gir minimalt brukt. Bilens maksimalfart på 20km/t gjør bilen unødvendig rask, samt ustabil å kjøre.[110] Ved å kjøre i lavere fart vil bilen ha substansielt lavere strømforbruk. Dette kan observeres ved å sammenligne bilens teoretiske kjøretid, med dens reelle kjøretid ved test.

Nettverksytelse og -hastighet Fra testresultatene til nettverksforbruket kan man se at videostrømmingen forbruker relativt lite data til tross for to parallelle videostrømmer. Dette gjør at datapakken til mobilabonnementet kan være mindre, hvilket gjør plattformen billigere i drift.

Resultatene fra testene av nettverksforsinkelse viser at forsinkelsen på både styresignalene og videostrømmene er lave, hvilket er svært viktig for kontrollen av plattformen. Forsinkelsene er her lave nok til at brukeren er i stand til å observere og styre unna hindringer uten problemer. Testresultatene viser også at forsinkelsen i signalene er relativ konsekvente uten store variasjoner. Dette bidrar til en stabil brukeropplevelse.

Brukergrensesnitt Det ferdigstilte brukergrensesnittet anses å være både funksjonelt og hensiktsmessig til den skisserte bruken. Det er bygd på en måte som tillater skalering og implementering av fremtidig funksjonalitet.

System I programvareversjonen til bilen som vi benyttet under tunneltesten visste vi at det eksisterte feil i koden som byttet mellom de ulike kameraene. Det var en sjanse for at bilen mistet tilkobling ved bytte. På bakgrunn av at vi hadde et gruppemedlem i tunnelen sammen med bilen valgte vi uansett å teste dette, da vi ønsket å se hvordan de ulike kameraene oppførte seg i den tette røyken. Desverre funket ikke dette på daværende tidspunkt, og vi ble nødt til å kjøre restart av systemet. Det vi merket oss i testen var at synsvinkelen hos primærkamera(*RPi cam 2*), var for lav. Hadde vi kunnet benytte den opprinnelig planlagte versjon 3, som har en bredere synsvinkel, hadde det trolig vært mer behagelig å kjøre bilen.I pilotprosjektet vårt har det blitt benyttet relativt billige kamera. Dette kan byttes ut med dyrere kameraer av høyere kvalitet om det skal produseres rovere som brukes i nødoppdrag.

Totalkostnad pr. enhet endte på 16200,- kroner, hvilket er innenfor TBRT sitt ønske om en kostnad på under 20000,- kroner.

7.1.1 Feilkilder

Ved test av ventetid på styresignaler, samt videooverføring vil resultatene variere basert på mobildata-modulens dekning til nærmeste basestasjon. Skydekke, samt nedbør kan dempe modulens signalstyrke. [28]

Under testing av batterilevetid ble det loggført hvilket gir roveren står i. Loggføringen viste første gir ved oppstart, og endret ikke verdi med mindre roveren mottok signal for å bytte gir. Dette kan ha ført til feil i loggingen, om systemet ble startet i andre gir.

Testen for maksimal batterilevetid ble utført over to dager, hvilket kan påvirke batteriets levetid. Dette var et resultat av at levetiden var vesentlig mye lengre enn først estimert.

7.2 Problemer underveis

Valg av mobiloperatør

Ved utforsking av oppkobling til mobilnettet med RPi ble Telenor benyttet som mobiloperatør. Ved bruk av serielt interface med 5G modul ble det observert at modulen hentet ut riktige detaljer fra SIM-kortet. Modulen forsøkte å koble seg til telenor.smart APN, men koblet seg ikke til nettet. Ved feilsøking ble det forsøkt å bruke private sim-kort, hvilket viste umiddelbar oppkobling til Telia-nettet. På bakgrunn av dette ble Telia benyttet som mobilnettsoperatør videre i prosjektet. Problemet med oppkobling til mobilnettet med Telenor som operatør reduserte effektiv tid satt av til å utvikle 5G systemet.

Statisk og dynamisk IP-addresse

Før sending av data mellom to enheter på internettet må man vite hvor informasjonen skal sendes. Derfor er det essensielt å vite hverandres IP-addresse. Da det nye systemet for oppkobling til mobilnettet benytter seg av oppringt nettverk er det krav om mobilabonnement med tele- og SMS-kompatibilitet. Dette gjør at tradisjonelle data-abonnementer ikke er et alternativ for vårt system. Ved utvikling av WebRTC ble det oppdaget at roveren bytter IP-Addresse jevnlig. Det ble tatt kontakt med Telia som informerte om at statisk IP kun var tilgjengelig for bedriftkunder med data-sim. Vi måtte derfor benytte oss av Pagekite og Turn-servere for å beholde kontakt med roveren.

Raspberry Pi Camera Module 3

Ved implementering av kamera ble det oppdaget at Raspberry Pi kameramodul 3 ikke var kompatibel med metoden vi henter ut videostrøm fra RPI 4, som viderestrømmes gjennom WebRTC. Kameramodul 3 måtte bruke video for linux 2 som brukergrensesnitt. Programvaren vår var satt opp til å benytte seg av utdaterte instillinger for Raspberry Pi Legacy Camera. Vi skiftet derfor ut kameramodul 3 med Raspberry Pi sin tidligere versjon, Kameramodul 2.1 for å kunne benytte den koden vi hadde utviklet. Dette gjorde at vi fikk et kjørekamera uten optisk bildestabilisering og med lavere synsvinkel.

Kalibrering av ESC

Da systemet sto ferdigstilt og skulle testes viste roveren tendenser til å akselerere kraftig. Roveren ble satt på boks, med hjulene hevet opp fra bakken. På denne måten kunne hjulene spinne fritt, mens bilen er stasjonær. Det ble oppdaget at roveren kjørte med konstant hastighet, uavhengig av brukers input. Det ble gjort omfattende feilsøking på avlesning av Xbox-kontroll, samt PWM-signaler til ESC. Alle signalene oppførte seg i henhold til programvaren vår. Det ble da oppdaget at ESC til drivkraftmotoren kan kalibreres. Dette gjøres ved å sette den i kalibreringsmodus, for

å så gi maksimal drivkraft fremover, og deretter maksimal drivkraft i bakoverretning. Etter en slik kalibrering oppførte roveren seg som ønsket.

Brukergrensesnitt

Ved oppstart av utvikling av brukergrensesnitt var tanken å bygge grensesnittet i Python. Etter flere dager med utvikling viste det seg at Python er uegnet til denne type applikasjon, og vi valgte å heller gå for en HTML/Java-basert løsning. Dette kostet oss tid, men var en lærerik prosess og indikerte at vi burde brukt enda mer tid på å undersøke dette punktet før vi begynte.

Gamepad API

I følge gamepad API dokumentasjonen skal APIen være kompatibel med alle nettlesere som støtter JavaScript, men kombinasjonen av Firefox og Mac førte til konflikt med kontroller-mappingen mellom kontrollerknappene og knappnummer i APIen. Brukersiden av prosjektet er derfor ikke kompatibelt med Firefox på Mac. Dette kan trolig fikses ved å lage en funksjon som sjekker hvilken nettleser som brukes for så å forandre kontroller-mappingen deretter.

7.3 Videreutvikling

Under utvikling av plattformen ble det oppdaget forbedringspotensiale som kan videre implementeres i systemet. Disse videreutviklingene kan resultere i et bedre, mer stabilt, og mer brukervennlig system.

Incendium

I begynnelsen av arbeidet var det ønskelig fra brannvesenets side å sende den innhentede videostrømmen til videostrøm-plattformen brannvesenet benytter, Incendium. I vårt arbeid fikk vi ikke implementert denne funksjonaliteten, men på bakgrunn av valgt teknologi for videooverføring skal det la seg gjøre å sende videostrømmen videre med noe mer arbeid. Koden for brukergrensesnittet er lagt opp for egen meny hvor en slik lenke kan implementeres.

Plattformbytte

Prosjektet er gjennomført på en Traxxas TRX-4 Land Rover Defender Silver 1/10 RTR. Under testing av kjøretøyet ble det observert ustabiliteter under svinging, spesielt ved høyere hastigheter. Dette reduserer muligheten for å kunne benytte seg av fartøyet i situasjoner med hastverk, eksempelvis brann i midten av en tunnel. På bakgrunn av dette ville vi videreutviklet systemet til å være kompatibelt med et 1/6 størrelse chassis. Dette vil da gjøre det mulig å beholde den nåværende bilen om det skal gjøres oppdrag i trangere omgivelser, eksempelvis lokasjon av gasslekkasje i et

bygg, men også bytte bilen ut med en mer stabil plattform ved oppdrag der det er nødvendig å bevege seg raskt.

Kamera-rotasjon

I fronten av bilen er det en kameramodul som kan roteres vertikalt, samt horisontalt. Bevegelsen på denne rotasjonen er dog begrenset til å peke fremover. I en situasjon der en ønsker å kartlegge omgivelsene sine kan det være attraktivt å ha muligheten til å observere omgivelsene bak bilen. Pan-Tilt-HAT som er tatt i bruk, er et oppsett av to servomotorer festet i hverandre, med 90 grader vinkling mellom servoarmene. Da den vertikale servomotoren er montert fast i den horisontale servomotoren, vil dette legge mye vekt på den horisontale servomotorens festemekanisme til bilen. For å forbedre synsvinkelen, samt lette det mekaniske stresset på servoene, ville vi i stedet utviklet en plattform som kan roteres 360 grader horisontalt og vertikalt ved bruk av steppermotorer.

Stabilisert kjørekamera

Under utvikling av systemet måtte vi bytte ut Raspberry Pi kameramodul 3 med kameramodul 2.1. Dette ble gjort fordi grensesnittet til kameramodul 3 ikke støttet vår programvare. Som en videreutvikling ville vi utviklet et system for å integrere kameramodul 3, da denne har større synsvinkel, bedre stabilisering, og bedre video-oppløsning enn kameramodul 2. Spesielt synsvinkelen er viktig da vi i våre tester opplevde at vinkelen på dette kameraet var for smalt til å ha en behagelig oversikt av omgivelsene.

Gassensor

I oppstartsfasen av prosjektet ble det utlyst ønske om å kunne montere en håndholdt gassensor til roveren. Den håndholdte gassensoren det var ønske om har et bluetooth BLE brukergrensesnitt, hvilket gjør det mulig å trådløst koble den til raspberry Pi. Signalene fra gassensoren kan da videresendes til GUI.

Lys

Til videre utvikling av bilen kan det være positivt å gi mulighet for å lyse opp omgivelsene. Dette kan implementeres med PWM, kontrollert fra GUI. På denne måten kan operatør av roveren stille på lysstyrken, dersom sikten er redusert.

Boltpistol

Ved oppdrag kan det være nødvendig eller ønskelig å punktere gasstanker. Dette gjøres tradisjonelt sett med skarpskytter. Til en fremtidig revisjon ville vi implementert en boltpistol, som kunne skutt hull på gasstank ved behov.

Termisk kamera

Et termisk kamera vil la operatør av rover få oversikt over varmepunkter i omgivelsene sine. Da systemet allerede støtter strømming av flere videokilder på en gang, ville det vært positivt å legge til en videokilde med termisk kamera. Dette er en fordyrende komponent, men med tanke på bruksområdene, og hvor redusert sikten til tradisjonelle kameraer er i tunneler, ville det vært et svært hensiktsmessig tillegg.

Antenner

På den ferdigstilte roveren er det benyttet fire monopole antenner til 5G-modulen. Disse monopole antennene er rette antenneelementer, som peker opp. Antennene er derfor et punkt som kan sette seg fast i hindringer under oppdrag. Da roveren opererer utelukkende på mobildata vil det også være viktig å undersøke hvilke alternativer vi har, i den hensikt å forbedre dekningen til kjøretøyet. Vi ønsker derfor å undersøke muligheten for å videreutvikle et system med andre antenner som kan festes på innsiden av et chassis, uten at dette reduserer dekningen.

Forbedret Av/På styring

Av/på modulen brukt i prosjektet 3.16 oppnår minimumfunksjonskravene til prosjektet, men har forbedringsmuligheter. Forriglingsmekanismen som holder RPien i på-posisjonen etter aktivering er implementert med et mekanisk relé, som er utsatt for slitasje og mekaniske vibrasjoner. Det mekaniske releet burde derfor erstattes med et *solid state* relé. Den nåværende modulen har i tillegg trykknapper og indikatorlys montert direkte på kretskortet, som vil være vanskelige å bruke når plattformen blir utstyrt med et beskyttende deksel. En mulig løsning er å montere knapper og indikatorlys direkte i dekselet, og koble de til av/på modulen med ledninger og konnektorer.

7.3.1 Innkapsling

Plattformen er ikke innkapslet i et beskyttende deksel, hvilket fører til at den interne elektronikken er utsatt for støv og vann. Før plattformen skal kunne brukes i feltet er det essensielt at den kan operere i forhold som kan forventes under brann- og redningsoperasjoner. Roveren burde tåle å bli utsatt for vannsprut fra alle retninger. Dette krever at det endelige produktet blir utstyrt med et beskyttende deksel som oppnår en IP-beskyttelsesgrad av IPX4 eller høyere.[53]

7.4 Konsekvenser av arbeidet

En raskt utviklende teknologisektor åpner for nyskapning, samt videreutvikling av eksisterende systemer for samfunnets gode. Til tross for kort utviklingstid viser systemet vårt gode resultater, og potensiale for videreutvikling. Ved videreutvikling

av et slikt system kan nødetatene få visuell avklaring av situasjoner der det er for risikabelt å sende inn personell. Ved videre forbedring kan systemet utvikles fra å være et observasjonsverktøy, til å være et viktig operasjonsverktøy. Dette kan oppnåes ved å benytte bilen til nøytralisering av farlige gassbeholdere under brann, der det er farlig å sende inn personell, og tradisjonelle skarpskyttere ikke har klar sikt på målet.

8 Konklusjon

Denne oppgaven tok for seg utviklingen av en fjernstyrt bil over 5G med overføring av videostrøm til operatør. Bilen skulle være et produkt av hyllevarer, slik at TBRT ville få muligheten til å gå til innkjøp, og produksjon av sine egne fartøy, om de skulle gå tapt.

Roveren ble utviklet til å kjøres over 5G, med videostrøm av både IR spekteret, og dagslysspekteret overført tilbake til operatør. Programvaren ble utviklet slik at operatør kan kontrollere fartøyet fra både Windows- og MAC-baserte enheter fra hvor som helst i verden.

Resultatet av oppgaven er et kjøretøy med både dagslys- og IR-kamera. Begge videostrømmene vises simultant i brukergrensesnittet. Det infrarøde kameraet er montert slik at det kan beveges både vertikalt og horisontalt for oversikt i omgivelsene. Testresultatene viser en gjennomsnittlig forsinkelse på 20-25ms for styresignalene, og 190ms for videooverføringen. Ved praktisk test viste systemet en levetid på 3 timer og 45 minutter.

I systemets nåværende tilstand vil vi ikke anbefale å benytte systemet i nødsituasjoner. Systemet er et godt konsept, men trenger videre utvikling og tester for å forbedre, samt verifisere stabilitet både mekanisk og programvaremessig. For videre utvikling vil vi anbefale fokus på integrasjon av sensorer og aktuatorer, slik at roveren kan gjennomføre oppdrag der det ikke er trygt å benytte seg av personell, samt kodeoptimalisering og -stabilisering.

Kilder

[1] *(fdm) Hva er fused deposition modellering? - definisjon fra techopedia - maskinvare 2023*. Icy Science. 2023. URL: https://no.theastrologypage.com/fused-deposition-modeling (sjekket 8. mai 2023).

[2] 3GGP. *Introducing 3GPP*. 3GPP. URL: https://www.3gpp.org/about-us/introducing-3gpp (sjekket 12. mai 2023).

[3] 3GPP. *5G for Industry 4.0*. URL: https://www.3gpp.org/technologies/tsn-v-lan (sjekket 27. apr. 2023).

[4] 3GPP. *Release 15*. URL: https://www.3gpp.org/specifications-technologies/releases/release-15 (sjekket 21. apr. 2023).

[5] 3GPP. *Release 16*. URL: https://www.3gpp.org/specifications-technologies/releases/release-16 (sjekket 27. apr. 2023).

[6] *4-ch Current/Voltage/Power Monitor HAT for Raspberry Pi, I2C/SMBus Interface*. URL: https://www.waveshare.com/current-power-monitor-hat.htm (sjekket 4. mai 2023).

[7] 52PI. *EP-0129 - 52Pi Wiki*. URL: https://wiki.52pi.com/index.php?title=EP-0129 (sjekket 9. mai 2023).

[8] James Adams. *Introducing Raspberry Pi HATs*. Raspberry Pi. 31. jul. 2014. URL: https://www.raspberrypi.com/news/introducing-raspberry-pi-hats/ (sjekket 12. mai 2023).

[9] *ADD-ON BOARDS AND HATs*. original-date: 2014-06-11T12:31:56Z. 1. mai 2023. URL: https://github.com/raspberrypi/hats (sjekket 8. mai 2023).

[10] Fritz Albregtsen og Gerhard Skagestein. «KOMPRESJON OG KODING». I: ().

[11] Paul Bjørn Andersen. *EEPROM*. I: *Store norske leksikon*. 21. jan. 2023. URL: https://snl.no/EEPROM (sjekket 12. mai 2023).

[12] *Antennetyper - Teknologiforståelse (IM-IKM vg1) - NDLA*. ndla.no. URL: https://ndla.no/nb/subject:1:81b3892a-78e7-4e43-bc31-fd5f8a5090e7/topic:1:35b4877b-4c94-479f-af07-98951659d5d2/topic:1:eefa49f5-025d-49d4-b436-7a8e00e25e14/resource:dbec19ad-1a98-4200-9ab6-0d917751a7d9 (sjekket 10. mai 2023).

[13] Arducam. *Arducam B0198*. Arducam. 2023. URL: https://www.arducam.com/product/b0198arducam-stereo-usb-camera-synchronized-visible-light-and-infrared-camera-2mp-1080p-day-and-night-mini-uvc-usb2-0-webcam-board-for-face-recognition-and-biological-detection/ (sjekket 8. mai 2023).

[14] Arducam. *Arducam OV2710 Dual-lens Camera Module*. 7. nov. 2019. URL: https://www.uctronics.com/download/Amazon/B0198_Datasheet.pdf (sjekket 9. mai 2023).

[15] Arduino. *Basics of PWM (Pulse Width Modulation) — Arduino Documentation*. 2023. URL: https://docs.arduino.cc/learn/microcontrollers/analog-output (sjekket 12. mai 2023).

[16] Are Hellandsvik, SINTEF og NTNU. *Oppgaveforslag bacheloroppgave elektroingeniør (BIELEKTRO) i Trondheim, vårsemester 2023*. (Sjekket 2. mai 2023).

[17] Howard Austerlitz. *Gain Amplifier - an overview — ScienceDirect Topics*. 2003. URL: https://www.sciencedirect.com/topics/engineering/gain-amplifier (sjekket 12. mai 2023).

[18] BillWagner. *Threads and threading*. 11. mar. 2022. URL: https://learn.microsoft.com/en-us/dotnet/standard/threading/threads-and-threading (sjekket 8. mai 2023).

[19] Trond F Christiansen mfl. «Forprosjekt Bacheloroppgave V23». I: ().

[20] cloudfare. *What is round-trip time? — RTT definition*. Cloudflare. URL: https://www.cloudflare.com/learning/cdn/glossary/round-trip-time-rtt/ (sjekket 12. mai 2023).

[21] Eric Coates. *Buck Converters*. 2020. URL: https://learnabout-electronics.org/PSU/psu31.php (sjekket 12. mai 2023).

[22] Iulian Corunga. *Don Norman's seven stages of action*. Medium. 27. apr. 2019. URL: https://medium.com/@iulian.corunga/seven-stages-of-action-to-improve-the-user-experience-a63fdcad2d82 (sjekket 5. mai 2023).

[23] *Coturn TURN server*. original-date: 2015-07-17T08:15:16Z. 10. mai 2023. URL: https://github.com/coturn/coturn (sjekket 10. mai 2023).

[24] Coursera. *UI vs. UX Design: What's the Difference?* Coursera. 15. mar. 2023. URL: https://www.coursera.org/articles/ui-vs-ux-design (sjekket 16. mai 2023).

[25] *Create monopole antenna over rectangular ground plane - MATLAB*. URL: https://www.mathworks.com/help/antenna/ref/monopole.html (sjekket 10. mai 2023).

[26] *Cross-Origin Resource Sharing (CORS) - HTTP — MDN*. 26. apr. 2023. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS (sjekket 8. mai 2023).

[27] *CSS - MDN Web Docs Glossary: Definitions of Web-related terms — MDN*. 21. feb. 2023. URL: https://developer.mozilla.org/en-US/docs/Glossary/CSS (sjekket 12. mai 2023).

[28] *Does weather impact internet connection speeds?* HighSpeedInternet.com. Section: FAQ. 28. nov. 2022. URL: https://www.highspeedinternet.com/resources/does-weather-impact-internet-connection-speeds (sjekket 10. mai 2023).

[29] DOMARS. *Finding the Process ID - Windows drivers*. en-us. Mar. 2022. URL: https://learn.microsoft.com/en-us/windows-hardware/drivers/debugger/finding-the-process-id (sjekket 19. mai 2023).

[30] Educative, Inc. *What are Norman's seven stages of action?* Educative: Interactive Courses for Software Developers. 2023. URL: https://www.educative.io/answers/what-are-normans-seven-stages-of-action (sjekket 5. mai 2023).

[31] *Eierne - TBRT.no*. URL: https://tbrt.no/om-tbrt/eierne (sjekket 15. mai 2023).

[32] Elefun. *Traxxas TRX-4 Land Rover Defender Silver 1/10 RTR*. URL: https://www.elefun.no/p/prod.aspx?v=35488 (sjekket 12. apr. 2023).

[33] *English*. Hovedredningssentralen. URL: https://www.hovedredningssentralen.no/english/ (sjekket 19. mai 2023).

[34] FreeRTOS. *Why RTOS and What is RTOS?* FreeRTOS. URL: https://www.freertos.org/about-RTOS.html (sjekket 19. mai 2023).

[35] *Fusion 360 — 3D CAD, CAM, CAE, & PCB Cloud-Based Software — Autodesk*. en-US. URL: https://www.autodesk.com/products/fusion-360/overview (sjekket 16. mai 2023).

[36] Gens ace. *Gens ace 5000mAh 11.1V 50C 3S1P Short-Size Lipo With XT60 Plug*. www.gensace.de. URL: https://www.gensace.de/gens-ace-5000mah-11-1v-50c-3s1p-short-size-lipo-with-xt60-plug-p.html (sjekket 9. mai 2023).

[37] *GitHub: Let's build from here*. en. URL: https://github.com/ (sjekket 16. mai 2023).

[38] Ivar Gunvaldsen, Steinar Mathiesen og Knut A. Rosvold. *batteri*. I: *Store norske leksikon*. 28. mar. 2023. URL: https://snl.no/batteri (sjekket 9. mai 2023).

[39] *Help Videos*. Raspberry Pi Foundation. URL: https://www.raspberrypi.org/help/videos/ (sjekket 8. mai 2023).

[40] *How to Use Simcom SIM8200EA-M2 Module with DSBOX-NX2?* URL: https://www.forecr.io/blogs/bsp-development/how-to-use-simcom-sim8200ea-m2-module-with-dsbox-nx2 (sjekket 9. mai 2023).

[41] *HTML - MDN Web Docs Glossary: Definitions of Web-related terms — MDN*. 21. feb. 2023. URL: https://developer.mozilla.org/en-US/docs/Glossary/HTML (sjekket 12. mai 2023).

[42] *HTTP Methods GET vs POST*. URL: https://www.w3schools.com/tags/ref_httpmethods.asp (sjekket 8. mai 2023).

[43] *HTTP request methods - HTTP — MDN*. 10. apr. 2023. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods (sjekket 8. mai 2023).

[44] *Hva er en STUN-server?* 3CX. URL: https://www.3cx.com/global/no/voip-sip-webrtc/stun-server/ (sjekket 2. mai 2023).

[45] *I2C Bus*. URL: https://www.i2c-bus.org/ (sjekket 5. mai 2023).

[46] *ICE - MDN Web Docs Glossary: Definitions of Web-related terms — MDN*. 21. feb. 2023. URL: https://developer.mozilla.org/en-US/docs/Glossary/ICE (sjekket 2. mai 2023).

[47] Image Engineering. *Image Stabilization - Image Engineering*. Image Quality Factors. URL: https://www.image-engineering.de/library/image-quality/factors/1076-image-stabilization (sjekket 3. mai 2023).

[48] *Incendium*. Incendium. URL: https://www.incendium.dk/?lang=en (sjekket 12. mai 2023).

[49] Incendium. *Home*. Incendium. URL: https://www.incendium.dk/?lang=en (sjekket 19. mai 2023).

[50] *Introduction to Fused Deposition Modeling (FDM)*. University of Maryland. 17. aug. 2022. URL: https://dozuki.umd.edu/Wiki/Introduction_to_Fused_Deposition_Modeling_%28FDM%29 (sjekket 10. mai 2023).

[51] *Introduction to WebRTC protocols - Web APIs — MDN*. 19. feb. 2023. URL: https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API/Protocols (sjekket 2. mai 2023).

[52] *IP Address - MDN Web Docs Glossary: Definitions of Web-related terms — MDN*. 21. feb. 2023. URL: https://developer.mozilla.org/en-US/docs/Glossary/IP_Address (sjekket 12. mai 2023).

[53] *IP Ratings Explained — IP Rating Chart*. en-GB. URL: https://rainfordsolutions.com/products/ingress-protection-ip-rated-enclosures/ip-enclosure-ratings-standards-explained/ (sjekket 19. mai 2023).

[54] *IR*. I: *Store norske leksikon*. 28. des. 2017. URL: https://snl.no/IR (sjekket 12. mai 2023).

[55] json.org. *JSON*. URL: https://www.json.org/json-en.html (sjekket 12. mai 2023).

[56] Ingun Grimstad Klepp. *PLA*. I: *Store norske leksikon*. 26. jan. 2023. URL: https://snl.no/PLA (sjekket 19. mai 2023).

[57] l3xa. *Enkle spørsmål: Hva er P2P (peer-to-peer) og hvorfor er det nyttig?* URL: https://no.l3xa.com/simple-questions-what-is-p2p (sjekket 12. mai 2023).

[58] lady ada. *Adafruit 16-Channel PWM/Servo HAT & Bonnet product page*. Adafruit Learning System. 1. feb. 2015. URL: https://learn.adafruit.com/adafruit-16-channel-pwm-servo-hat-for-raspberry-pi/overview (sjekket 8. mai 2023).

[59] lady ada. *adafruit-16-channel-pwm-servo-hat-for-raspberry-pi.pdf*. 12. jan. 2022. URL: https://cdn-learn.adafruit.com/downloads/pdf/adafruit-16-channel-pwm-servo-hat-for-raspberry-pi.pdf (sjekket 9. mai 2023).

[60] Jakob P. Liljenberg. *aristocratos/bashtop*. original-date: 2020-03-28T09:58:14Z. 2. mai 2023. URL: https://github.com/aristocratos/bashtop (sjekket 2. mai 2023).

[61] *Log In — Microsoft Teams*. en-US. URL: https://www.microsoft.com/en-us/microsoft-teams/log-in (sjekket 16. mai 2023).

[62] Lovdata. *Lov om interkommunale selskaper (IKS-loven) - Lovdata*. LOVDATA. 29. jan. 1999. URL: https://lovdata.no/dokument/NL/lov/1999-01-29-6 (sjekket 15. mai 2023).

[63] Raspberry Pi Ltd. *Buy a Raspberry Pi 4 Model B*. en-GB. URL: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/ (sjekket 5. mai 2023).

[64] Raspberry Pi Ltd. *GPIO pins — Physical Computing with Python — Python — Coding projects for kids and teens*. URL: https://projects.raspberrypi.org/en/projects/physical-computing/1 (sjekket 12. mai 2023).

[65] Mauro Marinilli. *The Theory Behind User Interface Design, Part One*. Developer.com. 21. nov. 2002. URL: https://www.developer.com/design/the-theory-behind-user-interface-design-part-one/ (sjekket 5. mai 2023).

[66] *Minicom - Getting Started With Minicom*. URL: https://wiki.emacinc.com/wiki/Getting_Started_With_Minicom (sjekket 12. mai 2023).

[67] *Minimum Viable Product (MVP)*. URL: https://www.productplan.com/glossary/minimum-viable-product/ (sjekket 12. mai 2023).

[68] Mouser Electronics. *DFRobot_Productoverview_DFR0929.pdf*. 23. jan. 2023. URL: https://www.mouser.com/catalog/additional/DFRobot_Productoverview_DFR0929.pdf?_gl=1*17l7jrv*_ga*MTc3MjExNTU1Ni4xNjgzNTM2ODYy*_ga_15W4STQT4T*MTY4MzUzNjg2MS4xLjAuMTY4MzUzNjg2Mi4wLjAuMA..*_ga_1KQLCYKRX3*MTY4MzUzNjg2MS4xLjAuMTY4MzUzNjg2Mi41OS4wLjA. (sjekket 8. mai 2023).

[69] Mozilla. *JavaScript — MDN*. 1. mai 2023. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript (sjekket 12. mai 2023).

[70] *multiprocessing — Process-based parallelism*. URL: https://docs.python.org/3/library/multiprocessing.html (sjekket 12. mai 2023).

[71] *NAT - MDN Web Docs Glossary: Definitions of Web-related terms — MDN*. 21. feb. 2023. URL: https://developer.mozilla.org/en-US/docs/Glossary/NAT (sjekket 2. mai 2023).

[72] *Om SINTEF - Anvendt forskning, teknologi og innovasjon*. no. URL: https://www.sintef.no/om-sintef/ (sjekket 15. mai 2023).

[73] *Open Source Initiative*. Open Source Initiative. 1. mai 2023. URL: https://opensource.org/ (sjekket 19. mai 2023).

[74] ortegatron. *aiortc*. original-date: 2018-02-23T22:05:16Z. 6. mai 2023. URL: https://github.com/aiortc/aiortc (sjekket 8. mai 2023).

[75] ortegatron. *aiortc/Index.html*. original-date: 2018-02-23T22:05:16Z. 6. mai 2023. URL: https://github.com/aiortc/aiortc/blob/9f14474c0953b90139c8697a216e4c2cd8ee5504/examples/server/index.html (sjekket 8. mai 2023).

[76] Oslo Brannkorpsforening. *ACETYLEN - flasker ved brann*. Brann & Redning. Section: Fagstoff. 25. okt. 2005. URL: https://brannredning.com/fagstoff/acetylen-flasker-brann/ (sjekket 2. mai 2023).

[77] *Pan-Tilt HAT*. The Pi Hut. URL: https://thepihut.com/products/pan-tilt-hat (sjekket 8. mai 2023).

[78] *PCA9685*. URL: https://www.nxp.com/products/power-management/lighting-driver-and-controller-ics/led-controllers/16-channel-12-bit-pwm-fm-plus-ic-bus-led-controller:PCA9685 (sjekket 3. mai 2023).

[79] Raspberry Pi. *Camera Module 2*. Raspberry Pi. URL: https://www.raspberrypi.com/products/camera-module-v2/ (sjekket 9. mai 2023).

[80] picockpit. *How much does power usage cost for the Pi 4?* PiCockpit — Monitor and Control your Raspberry Pi: free for up to 5 Pis! URL: https://picockpit.com/raspberry-pi/how-much-does-power-usage-cost-for-the-pi-4/ (sjekket 15. mai 2023).

[81] pidramble. *Power Consumption Benchmarks — Raspberry Pi Dramble*. URL: https://www.pidramble.com/wiki/benchmarks/power-consumption (sjekket 15. mai 2023).

[82] DNK POWER. *Complete Guide for Lithium Polymer(Lipo) Battery: History, Charging,Safety, Storage and Care*. en-US. Jan. 2019. URL: https://www.dnkpower.com/lithium-polymer-battery-guide/ (sjekket 9. mai 2023).

[83] princetoninstruments. *What is FOV and how does it influence your data?* Teledyne Princeton Instruments. URL: https://www.princetoninstruments.com/learn/camera-fundamentals/field-of-view-and-angular-field-of-view (sjekket 12. mai 2023).

[84] Python.org. *Welcome to Python.org*. Python.org. 11. mai 2023. URL: https://www.python.org/ (sjekket 12. mai 2023).

[85] Raspberry Pi Lt. *Raspberry Pi 4 Datasheet*.

[86] Raspberry Pi Ltd. *Camera - Raspberry Pi Documentation*. URL: https://www.raspberrypi.com/documentation/accessories/camera.html (sjekket 8. mai 2023).

[87] *raspberry-pi-4-datasheet.pdf*. URL: https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-datasheet.pdf (sjekket 5. mai 2023).

[88] *RNDIS Dial-up - Waveshare Wiki*. URL: https://www.waveshare.com/wiki/SIM820X_RNDIS_Dail-up (sjekket 12. mai 2023).

[89] Eirik Rossen, Kjell Bratbergsengen og Roger Pihl. *piksel*. I: *Store norske leksikon*. 12. jan. 2023. URL: https://snl.no/piksel (sjekket 12. mai 2023).

[90] Eirik Rossen og Tom Heine Nätt. *API*. I: *Store norske leksikon*. 28. des. 2022. URL: https://snl.no/API (sjekket 2. mai 2023).

[91] *RTOS Basics: Getting Started with Microcontrollers*. en-US. Apr. 2021. URL: https://www.seeedstudio.com/blog/2021/04/26/rtos-basics-getting-started-with-microcontrollers/ (sjekket 16. mai 2023).

[92] BarD Software s.r.o. *GanttProject: free project management tool for Windows, macOS and Linux*. en. URL: https://www.ganttproject.biz (sjekket 16. mai 2023).

[93] Darren Sawicz. «Hobby Servo Fundamentals». en. I: (). URL: https://www.princeton.edu/~mae412/TEXT/NTRAK2002/292-302.pdf.

[94] sculpteo. *What is an STL file? — Everything You Need to Know About This File Format*. Sculpteo. URL: https://www.sculpteo.com/en/3d-learning-hub/create-3d-file/what-is-an-stl-file/ (sjekket 12. mai 2023).

[95] *SIM8200 Series AT Command Manual*. URL: https://download.kamami.pl/p584354-SIM8200_Series_AT_Command_Manual_V1.00.01_0515.pdf (sjekket 9. mai 2023).

[96] *SIM8200EA-M2 5G HAT - Waveshare Wiki*. URL: https://www.waveshare.com/wiki/SIM8200EA-M2_5G_HAT (sjekket 27. jan. 2023).

[97] *SIM8262E-M2 SIMCom original 5G module, M.2 form factor, Qualcomm Snapdragon X62*. URL: https://www.waveshare.com/sim8262e-m2.htm (sjekket 9. mai 2023).

[98] *SINTEF Digital*. no. Mai 2023. URL: https://www.sintef.no/sintef-digital/ (sjekket 15. mai 2023).

[99] *Stor øvelse i Strømnestunnelen*. URL: https://developer.mozilla.org/en-US/docs/Web/API/Gamepad_API (sjekket 3. mai 2023).

[100] Wayne Storr. *Analogue to Digital Converter (ADC) Basics*. Basic Electronics Tutorials. 21. sep. 2020. URL: https://www.electronics-tutorials.ws/combination/analogue-to-digital-converter.html (sjekket 12. mai 2023).

[101] *STUN - MDN Web Docs Glossary: Definitions of Web-related terms — MDN*. 21. feb. 2023. URL: https://developer.mozilla.org/en-US/docs/Glossary/STUN (sjekket 2. mai 2023).

[102] pi-supply. *Pi Supply Switch - On/Off Power Switch for Raspberry Pi*. Pi Supply. URL: https://uk.pi-supply.com/products/pi-supply-raspberry-pi-power-switch (sjekket 9. mai 2023).

[103] Suzanne Martin. *Designing Effective User Interfaces*. Effective Visual Communication for Graphical User Interfaces. URL: https://web.cs.wpi.edu/~matt/courses/cs563/talks/smartin/int_design.html (sjekket 4. mai 2023).

[104] N. F. I. Team. *Frame Rate - Everything You Need to Know*. NFI. 21. sep. 2021. URL: https://www.nfi.edu/frame-rate/ (sjekket 12. mai 2023).

[105] Telia Norge AS. *Dekningskart*. URL: https://www.telia.no/nett/dekning/ (sjekket 15. mai 2023).

[106] Texas Instruments. *INA219 Zero-Drift, Bidirectional Current/Power Monitor With I2C Interface*. Des. 2015. (Sjekket 9. mai 2023).

[107] The Pi Hut. *GPIO Screw Terminal HAT*. The Pi Hut. URL: https://thepihut.com/products/gpio-screw-terminal-hat (sjekket 5. mai 2023).

[108] *The WebSocket API (WebSockets) - Web APIs — MDN*. 7. mar. 2023. URL: https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API (sjekket 12. mai 2023).

[109] Traxxas. *Traxxas TRX-4 — Scale and Trail Crawler*. URL: https://traxxas.com/products/landing/trx-4/ (sjekket 3. mai 2023).

[110] *Traxxas TRX-4 Scale Crawler Land Rover Defender D110 RTR*. EuroRC.com. URL: https://www.eurorc.com/product/13884/traxxas-trx-4-scale-crawler-land-rover-defender-d110-rtr (sjekket 9. mai 2023).

[111] Trøndelag Fylkeskommune. *2023 Orientering Strømnestunnelen.pptx*. 26. apr. 2023.

[112] *Tunneling in networking*. Cloudflare. URL: https://www.cloudflare.com/learning/network-layer/what-is-tunneling/ (sjekket 10. mai 2023).

[113] *TURN - MDN Web Docs Glossary: Definitions of Web-related terms — MDN*. 21. feb. 2023. URL: https://developer.mozilla.org/en-US/docs/Glossary/TURN (sjekket 2. mai 2023).

[114] *TURN server*. WebRTC. URL: https://webrtc.org/getting-started/turn-server (sjekket 2. mai 2023).

[115] *UDP (User Datagram Protocol) - MDN Web Docs Glossary: Definitions of Web-related terms — MDN*. 21. feb. 2023. URL: https://developer.mozilla.org/en-US/docs/Glossary/UDP (sjekket 12. mai 2023).

[116] *URL - MDN Web Docs Glossary: Definitions of Web-related terms — MDN*. 21. feb. 2023. URL: https://developer.mozilla.org/en-US/docs/Glossary/URL (sjekket 5. mai 2023).

[117] v4l2. *Part I - Video for Linux API — The Linux Kernel documentation*. URL: https://www.kernel.org/doc/html/v4.9/media/uapi/v4l/v4l2.html (sjekket 12. mai 2023).

[118] Vcodex Ltd. *An Overview of H.264 Advanced Video Coding*. Vcodex. URL: http://www.vcodex.com/an-overview-of-h264-advanced-video-coding/ (sjekket 4. mai 2023).

[119] Peter J. Vis. *Raspberry Pi CSI Camera Interface*. URL: https://www.petervis.com/Raspberry_PI/Raspberry_Pi_CSI/Raspberry_Pi_CSI_Camera_Interface.html (sjekket 12. mai 2023).

[120] *Vår historie*. no. URL: https://www.sintef.no/om-sintef/var-historie/ (sjekket 15. mai 2023).

[121] Waveshare. *Current/Power Monitor HAT - Waveshare Wiki*. Current/Power Monitor HAT. URL: https://www.waveshare.com/wiki/Current/Power_Monitor_HAT (sjekket 8. mai 2023).

[122] Tim Wells. *What Is an ESC For RC Cars?* Clutch RC. URL: https://clutchrc.com/electronic-speed-control/ (sjekket 12. mai 2023).

[123] *What is a Raspberry Pi?* en-GB. URL: https://www.raspberrypi.org/help/what-is-a-raspberry-pi/ (sjekket 5. mai 2023).

[124] *What is a web server? - Learn web development — MDN*. 23. feb. 2023. URL: https://developer.mozilla.org/en-US/docs/Learn/Common_questions/Web_mechanics/What_is_a_web_server (sjekket 5. mai 2023).

[125] *What Is Network Tunneling & How Is It Used? — Traefik Labs*. Traefik Labs: Say Goodbye to Connectivity Chaos. 4. nov. 2022. URL: https://traefik.io/glossary/network-tunneling/ (sjekket 10. mai 2023).

[126] *Xbox Wireless Controller: Xbox*. URL: https://www.xbox.com/en-US/accessories/controllers/xbox-wireless-controller.

[127] Hammad Zahid. *How Much Power Does Raspberry Pi Consume While Operating*. 2022. URL: https://linuxhint.com/power-consumption-raspberry-pi/ (sjekket 15. mai 2023).

[128] Håvard Zeiner. *Stor øvelse i Strømnestunnelen*. Trøndelag Fylkeskommune. 21. apr. 2023. URL: https://www.trondelagfylke.no/vare-tjenester/veg/nyheter-fylkesveg/stor-ovelse-i-stromnestunnelen/ (sjekket 3. mai 2023).

[129] Harald Øverby. *tingenes internett*. I: *Store norske leksikon*. 19. okt. 2021. URL: https://snl.no/tingenes_internett (sjekket 12. mai 2023).

[130] Harald Øverby mfl. *mobiltelefoni*. I: *Store norske leksikon*. 23. jan. 2023. URL: https://snl.no/mobiltelefoni (sjekket 3. mai 2023).

Appendix

Bacheloroppgave

Bendik Nygård, Trond F. Christiansen, Peter A. B. Knutsen, Syver Haraldsen

Institutt for elektroniske systemer

NTNU
Kunnskap for en bedre verden

SINTEF

Introduksjon

Dette prosjektet ble utført for SINTEF på vegne av Trøndelag brann- og redningstjeneste. Målet med oppgaven var å utvikle en fjernstyrt bil over 5G med mulighet for å strømme video tilbake til en operatør som kontrollerer fartøyet utenfor fareområdet. Et viktig moment ved oppgaven var at systemet skulle være satt sammen av hyllevarer slik at redningstjenesten skulle kunne kjøpe inn, og sette sammen ny bil ved tap under oppdrag.

Spesifikasjoner

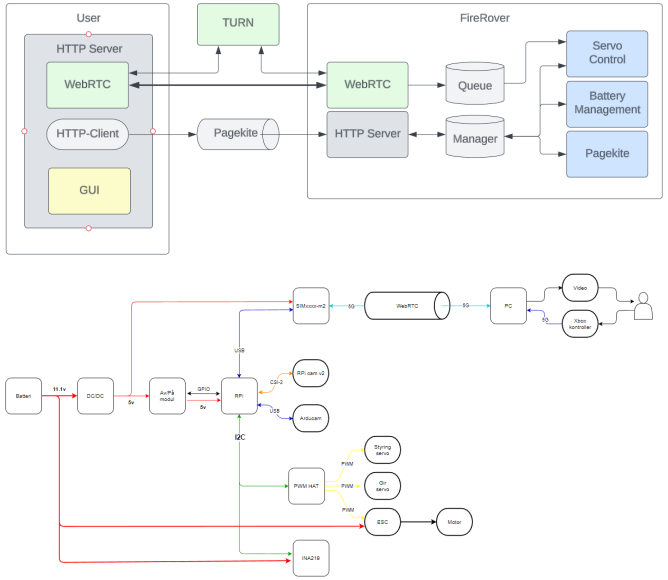
- **Batterikapasitet** 1000mAh
- **Kjøretid** 3:45
- **Maksimal hastighet** 20km/h
- **Størrelse** 1:10 skala
- **Vekt** 3.5kg

- **Mobilnett** LTE/NR5G
- **Overføringskapasitet** 2.4Gbps 500Mbps

- **Kjørekamera** 1280x720 30fps fargekamera
- **Speidekamera** 1280x720 30fps fargekamera IR-kamera
- **Responstid** 25ms

- **Est. pris** 16.000,- Nok

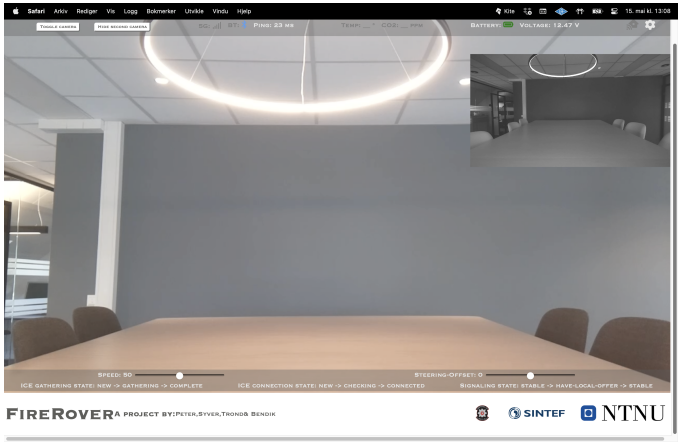
Systemarketektur



Systemt har to 3S LiPo-batterier som strømkilde. Batteriene forsyner motorkontrolleren med batterispenningen. Batterispenningen blir videre omformet gjennom en DC/DC omformer for å regulere spenningen ned til 5V. Dette spenningsnivået forsyner både Raspberry Pi og mobildatamodulen. Systemet er satt sammen av tilgjengelige hyllevarer.

Programvaren til plattformen er bygget opp rundt WebRTC kommunikasjonsprotokollen, hvilket tillater direkte kommunikasjon mellom operatør og plattform. Via WebRTC sendes to videostrømmer, sensordata og styredata. Ved å sende dataen på denne måten kan dataen strømmes med lav forsinkelse, noe som gir operatøren bedre kontroll over plattformen.

Brukergrensesnitt



I skjermdumpen over kan grensesnittet sees slik det er under bruk. Brukeren har her tilgang til å lese av; to videostrømmer, reell forsinkelse av kontrolldata, *ping*(oppgitt i ms), batteri- og spenningstilstand, samt tilkoblingsstatus. All informasjon ligger i bannere med semi-transparent bakgrunnsfarge for å la brukeren enkelt se informasjonen uten å blokkere sikten. Grensesnittet lar og brukeren velge hvilken av to kamerastrømmer som skal være i hovedfokus, og om kun en av to skal vises. Dette er praktisk om man ønsker å utnytte en av kamerastrømmene maksimalt. Estetisk er designet bygd for å fremstå moderne, samtidig som det innehar nødvendig informasjon for praktisk bruk, testing og utvikling.

Github

README.md

FireRover

GitHub repo for FireRover platformen. https://github.com/trondfc/FireRover

Utviklet som en bacheloroppgave hos NTNU i sammarbeid med SINTEF og TBRT

Plattformen skal være en fjernstyrt bil, styrt over mobilnett og utstyrt med kameraer og sensorer. Denne skal kunne brukes av brannvesenet på oppdrag for å sende inn for å skaffe seg oversikt i situasjoner hvor det ikke er trygt å sende inn mennesker, som ved brann i et annlegsområde med gassflasker.

Plattformen er konstruert rundt en Raspberry pi med tilleggsmoduler for mobilnettilkobling, PWM-styring og mer.

Koden til plattformen er laget for WebRTC-tilkobling mellom enhetene, og baserer seg på aiortc biblioteket.

For detaljert infromasjon om systemet, se bacheloroppgaven *Utvikling og realisering av 5G-basert, fjernstyrt kjøretøy for brann- og redningstjenesten* på ntnu open.

Setup

Raspberry

Enable legacy kamera og I2C på Raspberry pien med sudo raspi-config

Last ned github repoet til Raspberry pien og gå inn i mappen.

```
cd FireRover
```

Installer nødvendige python-biblioteker med

```
pip install -r Code/rover_requirements.txt
```

kopier `FireRover.service` og `PowerButtons.service` til `/lib/systemd/system/` og enable servicene med `sudo systemctl enable FireRover.service` og tilsvarende for PowerButtons

For å sette opp pien til å prioritere nettverk fra 5G HATen:

```
sudo nano /etc/dhcpcd.conf
```

her må det legges til instillinger for wlan0 og usb0

```
interface wlan0
metric 300

interface usb0
metric 200
```

Aiortc biblioteket bennytter seg av en utdatert h264 encoder på Raspberry pi. Derfor må det gjøres forandringer til biblioteket. Dette gjøres enklest ved å kjøre kommandoen:

```
sudo cp Code/h264.py /home/pi/.local/lib/python3.9/site-packages/aiortc/codecs/h264.p
```

Denne erstatter filen for h264-encoding med en oppdatert fil.

Restart raspberry pien med `sudo reboot`. Om alt fungerte vil bilen komme med tre raske pip ved oppstart.

PC

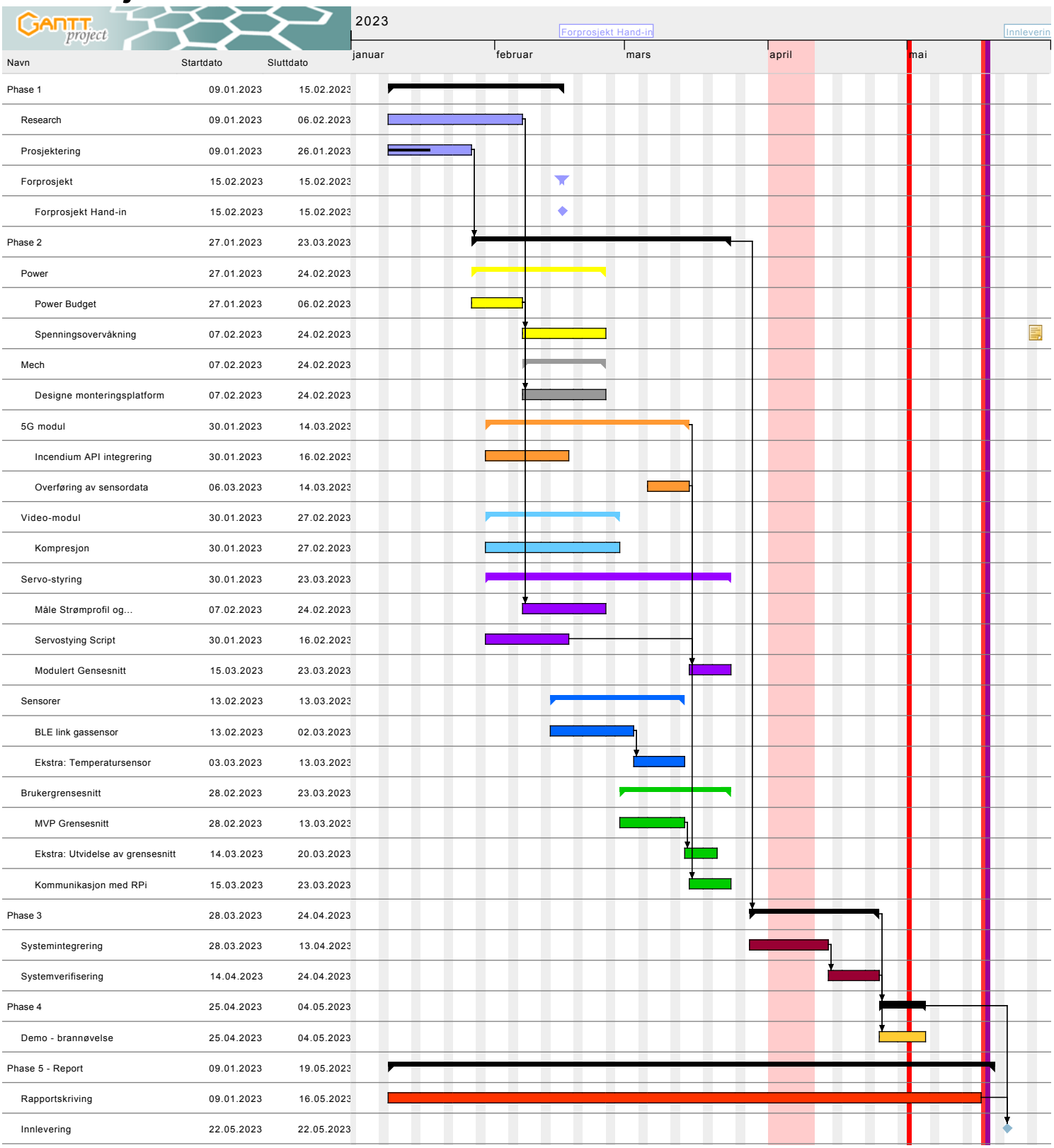
Last ned github repoet

Installer Python3 på PCen om den ikke har det

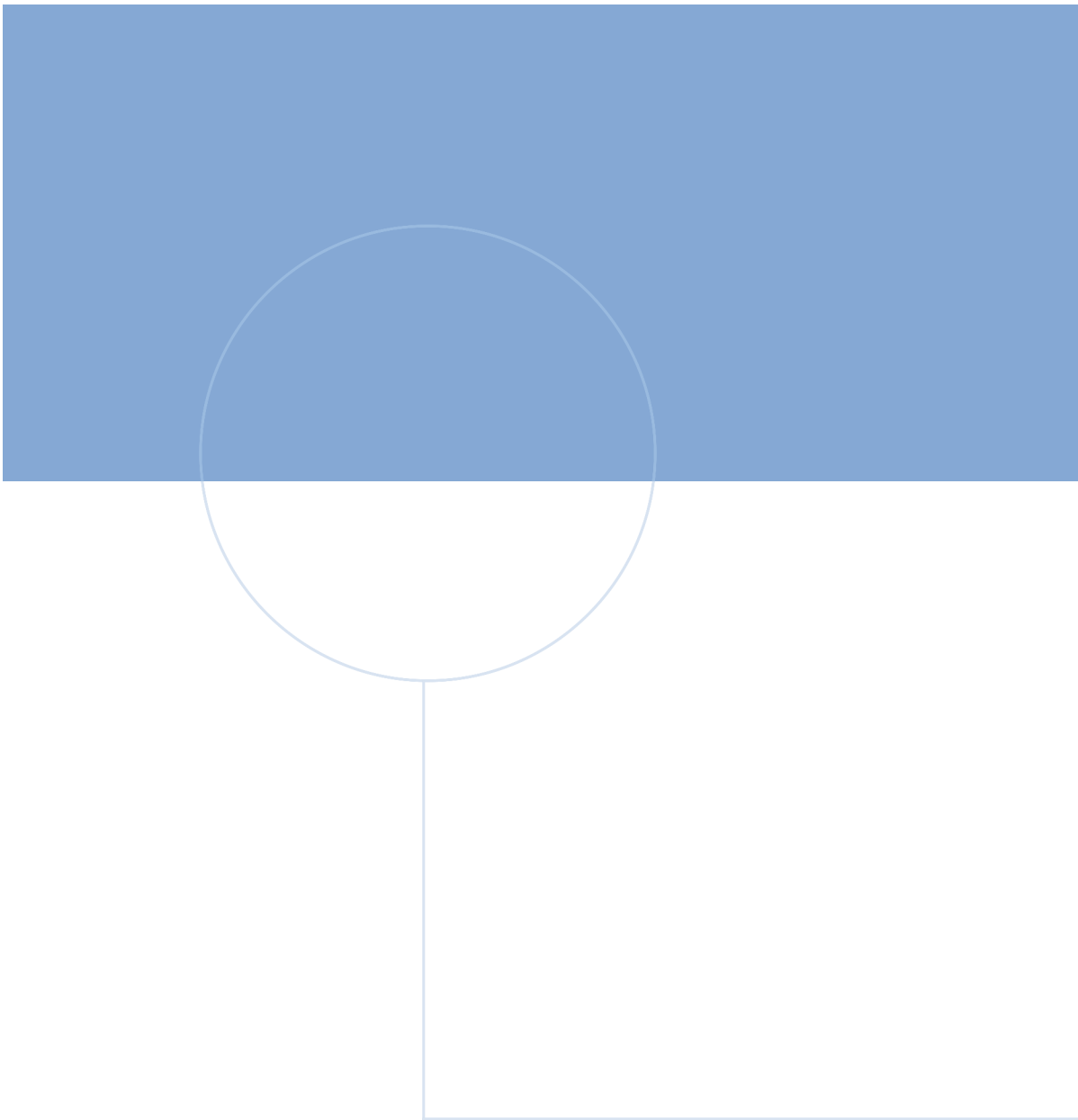
Kjør `download_packages.bat` for å installere nødvendige python-biblioteker

Kjør `main.bat` for å starte programmet

Ganttskjema



Navn	Startdato	Sluttdato
Phase 1	09.01.2023	15.02.2023
Research	09.01.2023	06.02.2023
Prosjektering	09.01.2023	26.01.2023
Forprosjekt	15.02.2023	15.02.2023
Forprosjekt Hand-in	15.02.2023	15.02.2023
Phase 2	27.01.2023	23.03.2023
Power	27.01.2023	24.02.2023
Power Budget	27.01.2023	06.02.2023
Spenningsovervåkning	07.02.2023	24.02.2023
Mech	07.02.2023	24.02.2023
Designe monteringsplatform	07.02.2023	24.02.2023
5G modul	30.01.2023	14.03.2023
Incendium API integrering	30.01.2023	16.02.2023
Overføring av sensordata	06.03.2023	14.03.2023
Video-modul	30.01.2023	27.02.2023
Kompresjon	30.01.2023	27.02.2023
Servo-styring	30.01.2023	23.03.2023
Måle Strømprofil og...	07.02.2023	24.02.2023
Servostying Script	30.01.2023	16.02.2023
Modulert Gensesnitt	15.03.2023	23.03.2023
Sensorer	13.02.2023	13.03.2023
BLE link gassensor	13.02.2023	02.03.2023
Ekstra: Temperatursensor	03.03.2023	13.03.2023
Brukergrensesnitt	28.02.2023	23.03.2023
MVP Grensesnitt	28.02.2023	13.03.2023
Ekstra: Utvidelse av grensesnitt	14.03.2023	20.03.2023
Kommunikasjon med RPi	15.03.2023	23.03.2023
Phase 3	28.03.2023	24.04.2023
Systemintegrering	28.03.2023	13.04.2023
Systemverifisering	14.04.2023	24.04.2023
Phase 4	25.04.2023	04.05.2023
Demo - brannøvelse	25.04.2023	04.05.2023
Phase 5 - Report	09.01.2023	19.05.2023
Rapportskriving	09.01.2023	16.05.2023
Innlevering	22.05.2023	22.05.2023



NTNU

Kunnskap for en bedre verden