Nr. _____ av _____

Sjøkrigsskolen

Bacheloroppgave

Testplattform for autonome systemer

av

Sondre Flaatten og Martin Tande

Levert som en del av kravet til graden:

BACHELOR I MILITÆRE STUDIER MED FORDYPNING I ELEKTRO OG AUTOMASJON

Innlevert: MAI 2018

Godkjent for offentlig publisering

I. Publiseringsavtale

En avtale om elektronisk publisering av bachelor/prosjektoppgave

Kadetten(ene) har opphavsrett til oppgaven, inkludert rettighetene til å publisere den.

Alle oppgaver som oppfyller kravene til publisering vil bli registrert og publisert i Bibsys Brage når kadetten(ene) har godkjent publisering.

Oppgaver som er graderte eller begrenset av en inngått avtale vil ikke bli publisert.

Jeg(Vi) gir herved Sjøkrigsskolen rett til å gjøre denne oppgaven tilgjengelig elektronisk, gratis og uten kostnader	[X] Ja	[ ] Nei
Finnes det en avtale om forsinket eller kun intern publisering? (Utfyllende opplysninger må fylles ut)	[ ] Ja	[X] Nei
Hvis ja: kan oppgaven publiseres elektronisk når embargoperioden utløper?	[ ] Ja	[ ] Nei

Plagiaterklæring

Jeg (Vi) erklærer herved at oppgaven er mitt eget arbeid og med bruk av riktig kildehenvisning. Jeg (Vi) har ikke nyttet annen hjelp enn det som er beskrevet i oppgaven.

Jeg (Vi) er klar over at brudd på dette vil føre til avvisning av oppgaven.

Dato: 20 – 05 – 2018

Sondre Flaatten _____
Kadett navn

_____
Kadett, signatur

Martin Tande _____
Kadett navn

_____
Kadett, signatur

II. Forord

Bacheloroppgaven er et krav for bachelor i militære studier med fordypning i elektro og automasjon ved Sjøkrigsskolen. Oppgaven har gitt oss muligheten til å ta i bruk et stort spekter av fagfeltet vårt. Oppgaven har gitt oss bedre kompetanse på elektrisk fremdrift, automasjon, elektronikk og datakommunikasjon og mekanisk konstruksjon. Arbeidet startet i januar 2018 og sluttet i mai 2018.

Takk til WAGO, for brukerstøtte og deler.

Takk til Alexander Sauter for engasjement og veiledning.

Bergen, Sjøkrigsskolen, 25.05.2018

_____

Sondre Flaatten

_____

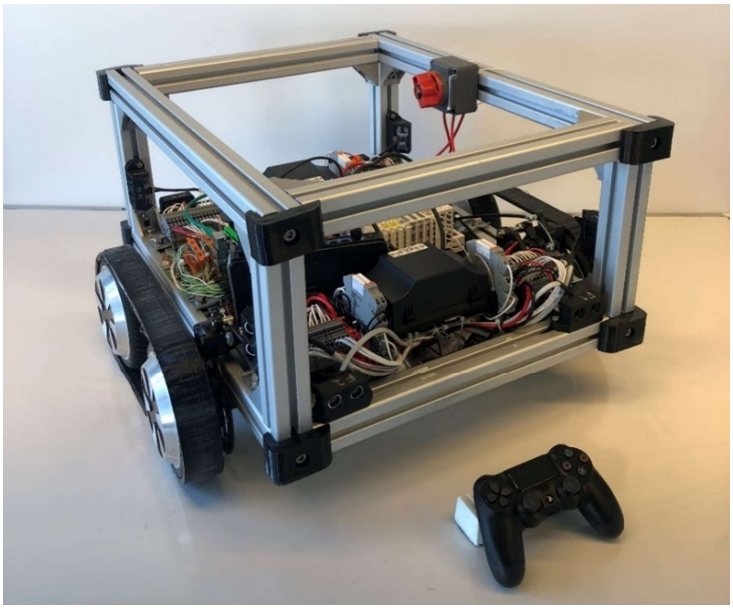
Martin Tande

III. Oppgaveformulering

I følge langtidsplanen for forsvaret skal det fokuseres mer på autonome og ubemannede systemer. Det oppgaven skal derfor ta for seg utviklingen av en testplattform for autonome systemer som skal kunne bli benyttet i undervisning og som grunnlag for kommende bacheloroppgaver. Denne plattformen skal gi et godt grunnlag for å teste ut forskjellige systemer som potensielt vil gjøre seg relevant innenfor autonomi. Derfor vil fokus i denne oppgaven ligge på modularitet, og dermed gjøre det enkelt å legge til komponenter eller endre på eksisterende.

IV. Sammendrag

Gjennom denne oppgaven har vi prosjektert, konstruert og testet en mobil plattform som skal danne et grunnlag for videre testing av autonome systemer. Den er konstruert i den hensikt å kunne fungere som en del av undervisningen på Sjøkrigsskolen, samt danne et grunnlag for fremtidige bacheloroppgaver som skal teste sensorer, styringssystemer eller beslutningssystemer. Fokuset har vært å skape en robust plattform basert på et modulært prinsipp, for å muliggjøre en enkel integrering av nytt utstyr. Robustheten ligger i styrken av rammen, valget av deler, signalformer og egenprodusert programvare som knytter alt sammen. Hele konstruksjonen er laget for å tåle stor vekt og for å lett kunne flytte på eller legge til nye deler. Delene vi valgt er mye brukt og dermed enkle å anskaffe, de er montert for å kunne lett skiftes ut ved behov og kommuniserer via standard signaltyper. PLS og Raspberry Pi er brukt som styrings- og kommunikasjonsenheter. Programvaren er bygget opp modulært slik at enkeltdeler kan endres på eller skiftes ut, uten at det er i konflikt med andre deler. Programvaren er programmert i forskjellige språk til de forskjellige enhetene. Grunnet modulariteten er en ikke avhengig av å kunne alle språkene for å kunne legge til nye funksjoner, det kan gjøres i et valgfritt språk som er støttet av den enkelte enheten. Alle variabler er mulig å gjøre tilgjengelig mellom enhetene og opp mot eventuelle nye enheter, dermed er plattformen klargjort for utvidelser, både mekanisk og datateknisk. Den batteridrevne plattformen står i dag klar til fjernstyring via medfølgende fjernkontroll, og er klar for utvidelser.



Figur IV.1: Plattformen i sin helhet, her oppsatt med beltedrift for bedre grep. Inkludert medfølgende PlayStation 4 kontroller for fjernstyring.

V. Innholdsfortegnelse

VI. Figurer

VII. Tabeller/Diagrammer

VIII. Nomenklatur / Forkortelser / Symboler

AckerMann styring	Typisk bilstyring der man fysisk svinger forhjulene
ESC	Electronic Speed Controller
FB	Funksjonsblokk
GPIO	General Purpose Input/Output
HMI	Human Machine Interface
HTML	HyperText Markup Language, Markeringspråk for formatering av nettsider med hypertekst.
LIDAR	Light Detection And Radar
MOSFET	Metal Oxide Silicon Field Effect Transistor
Msg.Payload	Melding med data i NodeRed, kan inneholde både tall og tekst.
NodeRed	Programmeringsverktøy for å koble sammen ulike hardware komponenter.
PLS	Programmerbare Logiske Styringer
RPM	Rotasjoner per minutt

1 Innledning

1.1 Bakgrunn

Autonomi er selvstyring av systemer, det vil si systemer som evner å handle basert på selvstendige avgjørelser. For Forsvaret kan dette innebære å kunne operere under risikofylte oppdrag uten fare for personell. Det kan også medføre billigere og mindre systemer da hotellvirksomhet for personell ikke er nødvendig. I følge Forskningsleder for FFI, Morten Nakjem, er autonome systemer «billigere, tryggere, mer fleksible og mer effektive» (Nakjem, 2016). Derfor er det viktig for Forsvaret å satse på og utvikle slike systemer. For at dette skal skje må det ligge til grunn en forståelse for hva autonomi er og hvordan det kan benyttes. Sjøkrigsskolen utvikler det personellet som i fremtiden vil skape og benytte seg av autonome systemer i Sjøforsvaret. Derfor er det viktig å kunne implementere autonomi i undervisningen. Dette krever undervisningsmateriell utviklet for det formålet.

1.2 Mål

Det er ønskelig å utvikle en testplattform for autonomi som kan brukes for demonstrasjon og laboratorieundervisning og til videreutviklingsplattform for kommende bacheloroppgaver. Autonomi er et konsept som ikke er fult utviklet, og det skjer stadig fremskritt innenfor teknologien. Derfor er hovedmålet med denne oppgaven å utvikle en plattform som danner grunnlaget for videre testing. For å muliggjøre dette settes stort fokus på modularitet i oppgaven. Med modularitet mener vi at denne konstruksjonen er bygget med tanke på at det lett kan integreres nye systemer ifm andre Bachelor-oppgaver eller oppgradering av de komponentene som allerede er benyttet. Modularitet går i ett med en systematisk hierarkisk oppbygning, både innover – deler er utbyttbare og funksjoner er utvidbare, og utover – plattformen er klar til bruk som modul i en større sammenheng. Denne plattformen skal oppfylle visse krav, gitt under avgrensninger, for å kunne gi det grunnlaget som kreves for å kunne teste nye systemer i fremtiden.

1.3 Avgrensninger

Denne oppgaven er blitt avgrenset innenfor et tidsrom på 5 måneder og har hatt en øvre budsjettgrense på 20 000kr. Fokuset i oppgaven har vært å skape en grunnplattform, som innebærer en konstruksjon og fremdriftslinje som kan styres for å tilrettelegge for utvikling av autonomi, fremfor å skape et autonomt system, grunnet det gitte tidskravet.

For å kunne oppnå målet har det blitt jobbet ut fra en rekke krav. Det er for å sette en minimumsstandard som sørger for at plattformen er klargjort for videreutvikling.

1. Konstruksjonsmessig ble det satt et krav om å kunne bære totalvekt på minst 100kg. Dette sikrer rom for utvidelser uten at konstruksjonens integritet blir en begrensning.
2. Krav om å kunne kjøre gjennom døråpninger på 80cm bredde. For å kunne sikre bruk av plattformen i undervisningssammenheng.
3. Sikkerhetsmessig må plattformen være beskyttet mot overbelastning og kortslutning samt ha mulighet for å kunne kutte all spenning.
4. Fremdriftslinjen må kunne operere regulerbart i minimum gangfart. Det ansees tilstrekkelig for å kunne gi rom for videre testing av sensorpakker.
5. Krav til at plattformen kan benyttes i minimum fire undervisningstimer.
6. Krav til Human Machine Interface, for å lettere kunne bruke plattformen uten å være avhengig av å justere kode.
7. Krav til Trådløs fjernstyring for å lettere kunne benytte seg av plattformen (som nødvendig del av Human Machine Interface).
8. Krav til overvåkning av temperatur og strøm i de mest sårbare komponentene for å kunne forhindre overbelastning ved bruk.
9. Krav til høy grad av modularitet i alle ledd.

1.4 Metode

Oppgaven er løst ved å undersøke hvilke komponenter som ville være aktuelle å benytte innenfor den gitte tidsrammen og budsjettet. Deretter er disse anskaffet og satt sammen til en test- og undervisningsplattform. Det er blitt brukt flere styringsenheter og programmeringsspråk for å sammensette plattformen. Dette er for å kunne utfordre den modulære målsetningen, ved å benytte ulike enheter med ulike språk og oppnå samspill mellom disse. Testing er blitt gjort for å verifisere funksjonaliteten til de valgte komponentene, og kapasiteten til plattformen.

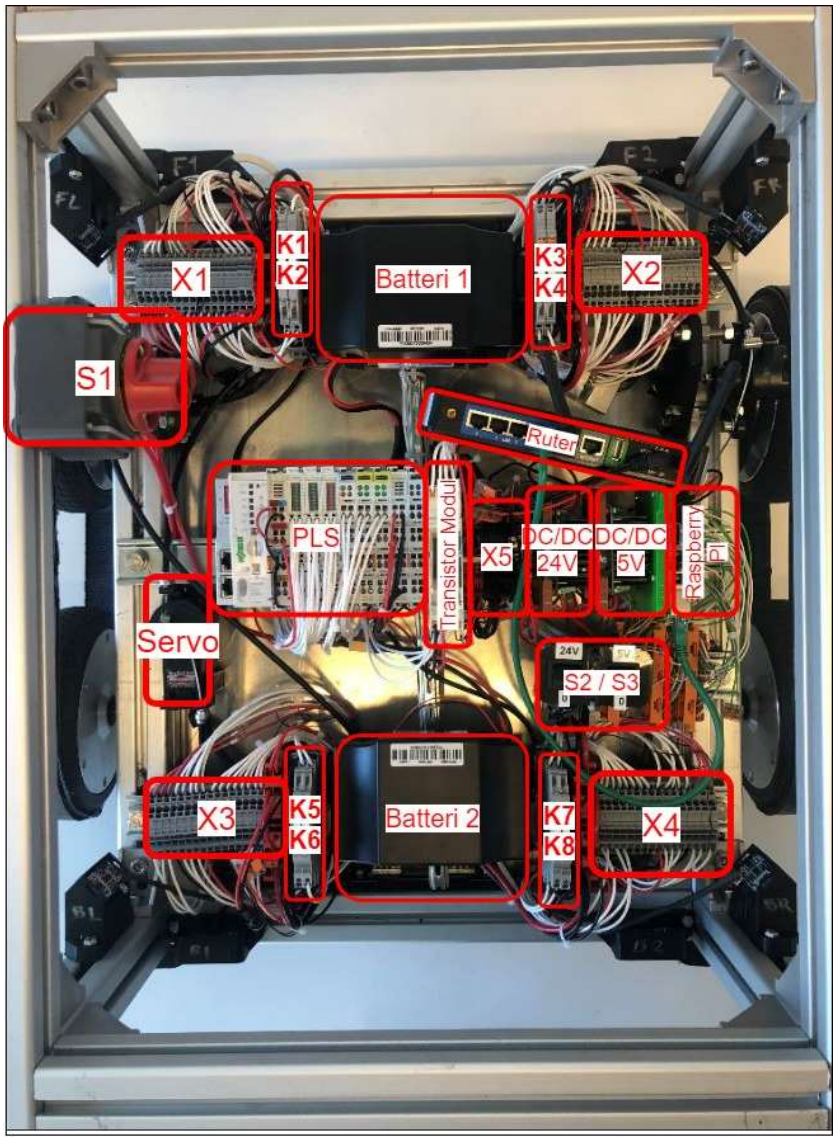
1.5 Struktur

Oppgaven tar først for seg oppbyggingen av plattformen (Kapittel 2 Hardware). Hensikten her å gi en forståelse av hvilke komponenter som er brukt, hvordan de fungerer og satt sammen. Deretter beskrives programmeringen gjort i prosjektet (Kapittel 3 Programmering). Her går oppgaven nærmere inn på hvordan programvaren er bygd opp. Til slutt går oppgaven gjennom testene som er blitt gjort (kapittel 7 Testet). Dokumentasjon og kildekode er lagt ved som vedlegg (Kapittel 8 og 9).

2 Hardware

Med begrepet Hardware mener vi alle fysiske komponenter som bygger opp plattformen. Hensikten med dette kapittelet er å gi leseren en forståelse av hvilke komponenter og løsninger som er blitt brukt, hvordan de virker og samspillet mellom dem.

Denne delen tar først for seg hvordan konstruksjonen av plattformen og fremdriftslinjen er bygd. Videre går oppgaven inn på plattformens styringsenheter, hva de består av og hvordan de virker. Deretter kommer et kapittel på design av deler ved hjelp av 3D-printing. Deretter blir kobling av komponentene nærmere forklart. Til slutt blir valg av sensorer til overvåking og hvordan de fungerer gjennomgått.

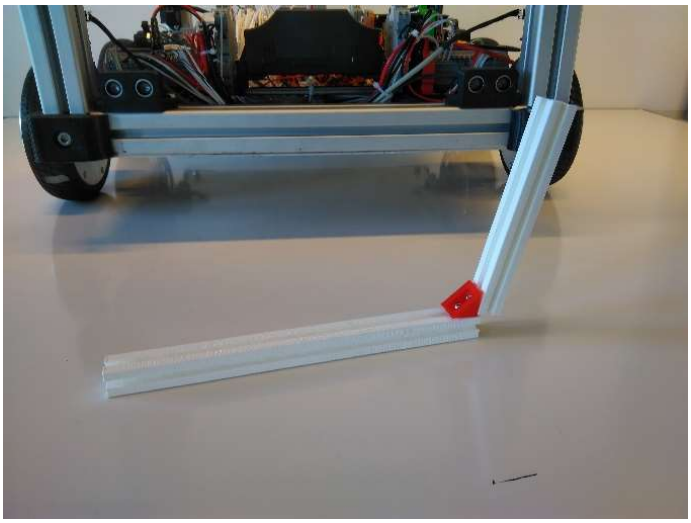


Figur 2.1: Oversiktsbilde over innsiden av plattformen, for videre forklaring se vedlegg B.4.

Figur 2.1 gir en grov oversikt over innsiden av plattformen. Det er satt opp 3 DIN-skinner. Øverste og nederste skinne består av like komponenter, batteri og motorkonkrollere, og er koblet opp identisk. På øverste og nederste skinne er det plassert batteri i midten. På hver side av batteriet er det plassert 2 releer og rekkeklemmer for inputs og outputs fra motor. Motorkontroller er plassert under rekkeklemmene. På den midterste skinnen er styringsenhetene, PLS og Raspberry Pi, samt 2 DC-DC omsettere (36-24V og 36-5V). På venstre side i figur 2.1 ser man servomotoren som styrer bremsewiren (svart boks i skyggen). Ultralydsensorene som er montert på rammen (8 stk, to på hvert hjørne), kobles sammen med Raspberry Pi gjennom Wago-klemmer i de oransje holderne.

2.1 Ramme

Rammen til plattformen skal sørge for grunnstabiliteten og er det som skal holde alle komponenter sammen. I følge krav 1, skal konstruksjonen kunne tåle en totalvekt på 100kg. Det er derfor viktig at valg av konstruksjonsmateriale og sammensetningen er robust og godt gjennomtenkt. Viktige kriterier for rammen var modularitet, materialets styrke, bruksvennlighet og pris. Ut i fra disse kriteriene ble 4x4cm Aluminiumprofiler benyttet som konstruksjonsmaterialet til ramme. Det ble printet ut en testmodell av profilen for å vurdere profilens egenskaper som ramme til plattformen (figur 2.2). Dette ble gjort for å unngå unødvendige innkjøp av materiale.



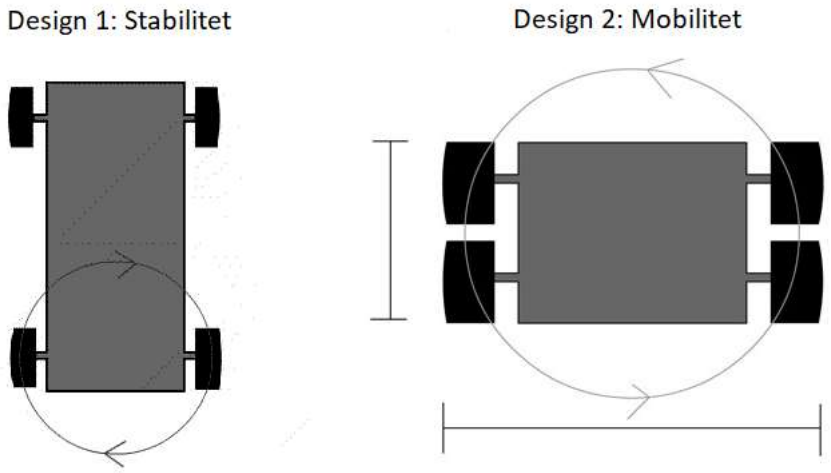
Figur 2.2: Testmodell av profil brukt til ramme

Ifølge krav 1 må plattformen kunne ha relativt stor lastekapasitet. Aluminium er da et godt valg sammenlignet med plast. Med en tykkelse på 4x4cm gir det rammen gode utvidelsesmuligheter før materialet svikter. På denne måten sørger vi for modularitet i valget av ramme. Sammenlignet med andre konstruksjonsmaterialer som stål og rustfritt stål, er aluminium lettere i vekt og enklere å kutte og skjære i. For å kutte profilene til riktige lengder ble det brukt båndsag til grovkutting og fres til finkutting ved verkstedene på Haakonsvern.

I profilene er det spor på hver langside som er laget for å entre spesial-muttere. Med 90 graders braketter og bolter blir sammensettingen av denne konstruksjonen svært enkel og kan fort endres på.

Valg av styringsmetode stod mellom AckerMann styring og elektrisk differensialstyring. AckerMann styring, innebærer en fysisk endring av vinkelen på hjulene for å svinge, slik som på en vanlig bil. Dette krever kompleks mekanisk design med akslinger, overføringer og servomotor. Det andre alternativet var elektrisk differensialstyring, som vil si at man skaper en forskjell i rotasjonshastighet i hjulene på venstre og høyre side for å svinge. Valget falt på denne metoden da den innebærer mindre mekanisk arbeid og krever ingen ekstra deler.

Posisjoneringen av hjulene med tanke på avstand mellom dem har en del å si for både mobilitet og stabilitet. Som demonstrert i figur 2.3 vil kortere avstand mellom bakhjulene og forhjulene, øke mobiliteten plattformen. Med mobilitet menes evne til å rotere rundt sin egen akse, uten å gli med hjulene (forutsett at man ikke bruker AckerMann styringsmetoden). Men på grunn av lengden på plattformen, vil for liten avstand mellom hjulene skape et overheng i front og bak på plattformen. Dette overhenget vil skape ustabilitet i den forstand at den lettere kan tippe over og miste grep med alle 4 hjulene. Vi valgte en kombinasjon av dette, og endte med en avstand på 26 cm fra hverandre. Dette ga oss minimalt med slep på hjulene ved svinging. Plattformen mister litt stabilitet ved denne løsningen, men for denne oppgaven er mobilitet viktigere. Hvis det kreves mer stabilitet anbefales det å montere ett vognhjul på hver ende.



Figur 2.3: Forskjellige hjulplasseringer med tilhørende vendesirkler (Ron Robotics 2014)

Et krav til dimensjonering av rammen var at den skulle kunne kjøre gjennom døråpninger. Likevel var det ønskelig at den er stor slik at det var plass til ekstra komponenter og andre utvidelser. Samtidig betyr større plass at komponenter er lettere tilgjengelig for måling, utskifting og vedlikehold. Lengden måtte heller ikke bli for lang, da det kan innvirke på stabiliteten til rammen. Dimensjonene til rammen ble 60cm lang, 48cm bred, 28cm høy.

2.2 Fremdriftslinje

For plattformen ble det valgt helelektrisk fremdrift som består av batterier, motorkontrollere og børsteløse likestrømsmotorer. Oppgaven skal nå greie ut om valg av disse delene samt virkemåten til komponenentene.

2.2.1 Batteri

Plattformen benytter to Lithium-ion batterier, med nominellspenning på 36V og 4 Ah. Hvert batteri er bygget av 10 celler, som gir maksspenning på 42V. Totalt effektuttak er på 144Wattimer. Batteriene er parallellkoblet som gir samme spenning med doblet effekt, altså 288 Wattimer. Batteriene er vedlikeholdsfrie, har kort ladetid og lav selvutlading. De har også høy energitetthet, som gjør at de tar lite plass i forhold til et Bly-batteri med samme energiinnhold. Dette var et viktig kriterium da ekstra plass er ønskelig å ha til eventuelle utvidelser og nye komponenter, og er ikke ønskelig å bruke plassen på store

Bly-syre batterier. I henhold til krav 5 kreves det at Plattformen skal kunne ha en utholdenhet på 4 skoletimer i forbindelse med klasseromsundervisning. I følge test 7.6, kan plattformen kjøres kontinuerlig i 5km/t i 1,5 time. En klasseromsundervisning vil hovedsakelig bestå av programmering til plattformen og enkle tester, der lader kan være tilkoblet så lenge plattformen står i ro. Ut ifra dette kan vi med sikkerhet si at den vil holde i 4 undervisningstimer.

Batteriet har et Battery Management System, BMS. BMS har spenningsovervåking på batteriet og sørger å sikre mot overbelastning, og forhindrer total utladning da dette kan være skadelig for batteriet. BMS sørger også for sikker lading av batteriet i den forstand at den overvåker hver enkelt celle i batteriet og brenner av eventuell overskuddseffekt til cellene under lading. Ved ønske om utvidelser kan batteriene byttes ut med tilsvarende batterier så lenge de leverer samme spenning og har en egen BMS.

For plattformen har vi benyttet oss av laderne som fulgte med batteriene. Laderen gir opptil 42V og 1,5 A. På grunn av at plattformen har 2 batterier som ligger i parallell har vi laget tilkobling for to ladere, dermed kan begge batteriene lades opp like hurtig som i et tilfelle med ett batteri og en lader.

2.2.2 Motor

Viktige momenter ved valg av motor var pris, størrelse, moment og riktig turtallsområde. Da autonomi og styring står i fokus er plattformen tiltenkt å hovedsakelig operere i gangfart, og romme mye utstyr. Det er derfor viktig at motorene må kjøre med høy virkningsgrad i lave turtall og gi mye moment. Plassbesparelse og pris er også viktige kriterier da det er ønskelig å ha plass og budsjett til nye og flere komponenter ved eventuelle utvidelser. Basert på disse kriteriene stod valg av motor stod mellom DC-motor, asynkronmotor og børsteløs DC-motor.

DC-motor er billig og turtall er enkelt å styre med variabel inngangsspenning. Bruk av DC-motorer innebærer vedlikehold av børster, moderat effektivitet grunnet tap i børstene, relativt liten effekt per kubikkmeter og kort levetid sammenlignet med andre elektromotorer. Asynkronmotoren er velkjent og mye brukt i industri. Den har relativt høy virkningsgrad (rundt 95%) og krever lite vedlikehold. For å regulere turtall og gi ut maksimalt moment på asynkronmotoren kreves frekvensomformer for hver motor. Dette ville blitt unødvendig dyrt og ville tatt mye plass i plattformen. Børsteløse DC-motorer har høy

effektivitet, har mye effekt per kubikkmeter og krever lite vedlikehold. De er også effektive i et bredt turtallområde og gir ut stort moment. De krever en styringsenhet for å kjøre, men disse enhetene er betraktelig billigere enn frekvensomformere. DC-motoren ble utelukket på grunn av børstene, og asynkronmotoren ble utelukket grunnet pris og plass. Valget ble dermed børsteløs DC-motor.

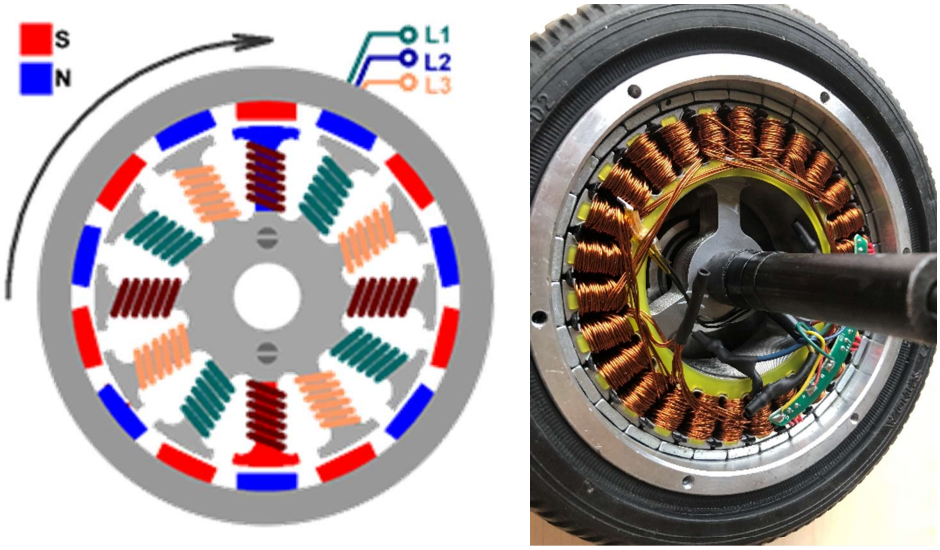
Børsteløse DC-motorer blir for mange formål integrert inn med selve hjulet også, se figur 2.4. Dette kalles Hub-motor. Bruk av Hub-motor er plassbesparende, det er lettere å installere og innebærer færre komponenter. Ulempen med en slik variant er at man ikke kan bytte motor og hjul uavhengig. Fordelene overveide ulempene og vi valgte derfor en Hub-motor. Plattformen benytter seg av 4 Hub-motorer på 36V, dimensjonert for opptil 350W hver, som blir 1400W totalt til fremdriftslinjen. Dette gir plattformen mulighet til å utvides med tanke på tilgjengelig effekt til å flytte ekstra vekt.

Det viste seg å være billigere å anskaffe motor og batteri ved å kjøpe 3 Hoverboard istedenfor å kjøpe delene separat. Ved å gjøre det slik visste vi også at komponentene passet sammen. Batteri og motorer ble brukt fra 2 Hoverboard, mens det tredje brukes som reservedeler hvis en av motorene skulle bli ødelagt. .

I det neste avsnittet beskrives virkemåte og oppbygging til motorene.

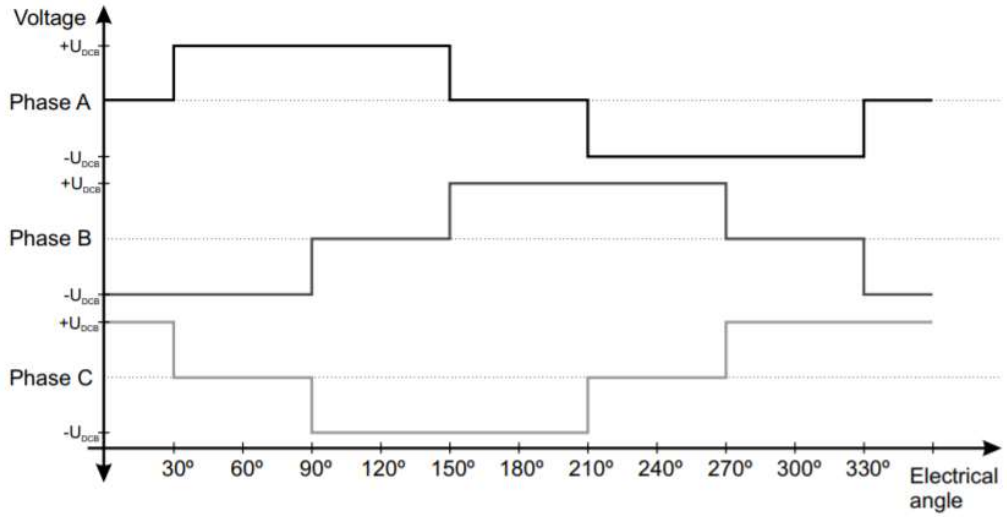
Rotoren er bygget opp av 45 permanentmagneter og ligger på utsiden av statoren. Dette kalles en Outrunner. Magnetene ligger med annenhver nord- og sydpol inn mot statoren.

I statoren som er på innsiden og fastmontert på akslingen, er det 3 faser viklet rundt jern-kjerner.



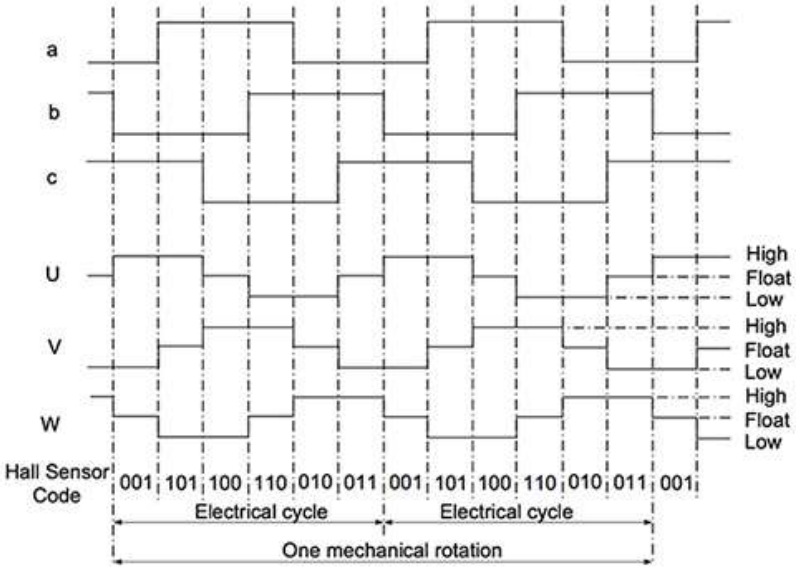
Figur 2.4: Innside av motor. Stator ligger på innsiden med spoler viklet rundt jern-kjerner, rotor består av permanentmagneter og ligger på utsiden (Heli-planet, 2018).

For å få rotoren til å rotere settes det en DC spenning på den ene fasen som da vil generere et magnetfelt som trekker permanentmagneten til seg. For å forsterke denne effekten settes negativ spenning på spolen «bak» permanentmagneten, som vil si at strømmen gjennom spolen går andre veien. Da vil spolen lage et magnetfelt med motsatt rettet kraft og dytter permanentmagneten lenger fra seg. Når permanentmagneten i rotoren har flyttet seg bort til neste elektromagnet repeteres denne prosessen med å sette positiv spenning på fasen foran, og negativ spenning på fasen bak, og ingen spenning på fasen i midten, se figur 2.5. Den kombinerte kraften gir konstant moment opp til merketurtall.



Figur 2.5: Spenning inn på de 3 fasene (Elevich 2005)

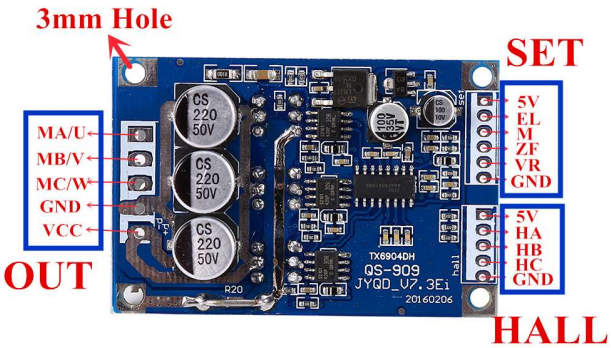
For å få rotoren til å rotere på denne måten må rotoren være i fase med polaritetsendringene til elektromagnetene på statoren. For å gjøre det brukes Hall effekt sensorer, som er mest vanlig i BLDC. Hall effekt sensorer er givere som endrer utgangsspenning i respons til magnetfeltet den blir utsatt for. I denne motoren er det plassert 3 hall effekt sensorer som er forskjøvet elektrisk 120º fra hverandre. Sensorene vil gi et høyt signal og lavt signal for respektivt nord- og sydpol. Kombinerer vi signalene til alle 3 sensorer har vi 8 mulige utfall, fra 000 til 111. Men på grunn av måten denne BLDC-motoren er bygget på er ikke utfall 000 og 111 mulig, som gir oss da 6 forskjellige kombinasjoner, se vedlegg (Test av Hall effektsensorer). Disse 6 mulige utfallene deler kommuteringssyklusen på elektrisk 360º opp i 60º, slik at hver 60'ende elektriske grad endres én hall effekt sensor-utgang som igjen endrer hvilke faser som blir trigget.



Figur 2.6: Diagram for hall effekt sensor status. a, b og c er Hall effekt sensor-utganger. U, V og W er spenning tilført til motorviklingene (Digi-Key Electronics 2016).

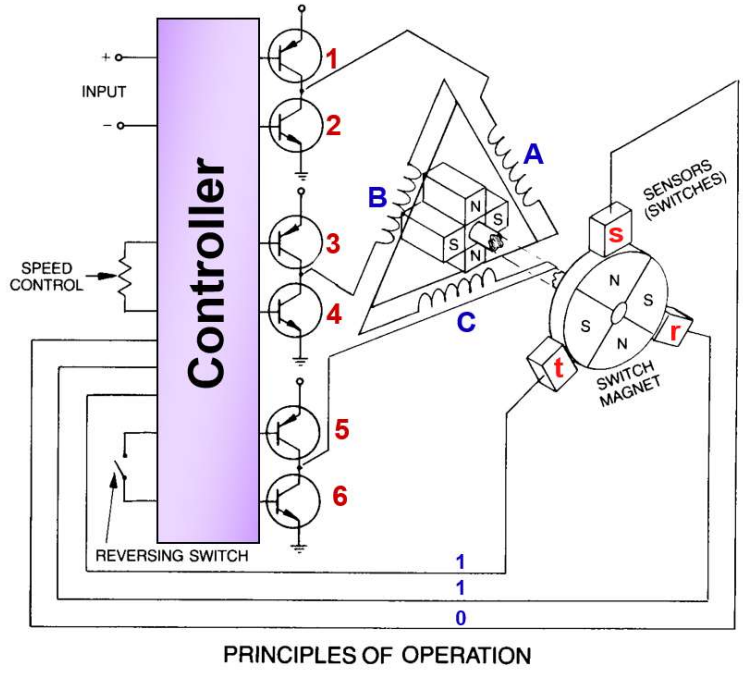
2.2.3 Motorkontroller

For å styre motorene trengs en motorkontroller. I Hoverboard-pakken som ble anskaffet fulgte det med motorkontroller innebygget i Hoverboard for å styre motorene. For å slippe å anskaffe nye motorkontrollere, ble det prøvd å ta i bruk de medfulgte ved å omprogrammere disse. Det viste seg å være vanskelig da vi ikke fant noe dokumentasjon på styringen til Hoverboard og de ikke var tilrettelagt for ekstern kontroll. Vi endte opp med å kjøpe ny BLDC Motorkontroller, JYQD_V7.3E2, se figur 2.7.



Figur 2.7: Motorkontroller. Power og motorutganger på venstre side, Hall-effekt-sensor-utganger og styring på høyre side (Ebay 2018).

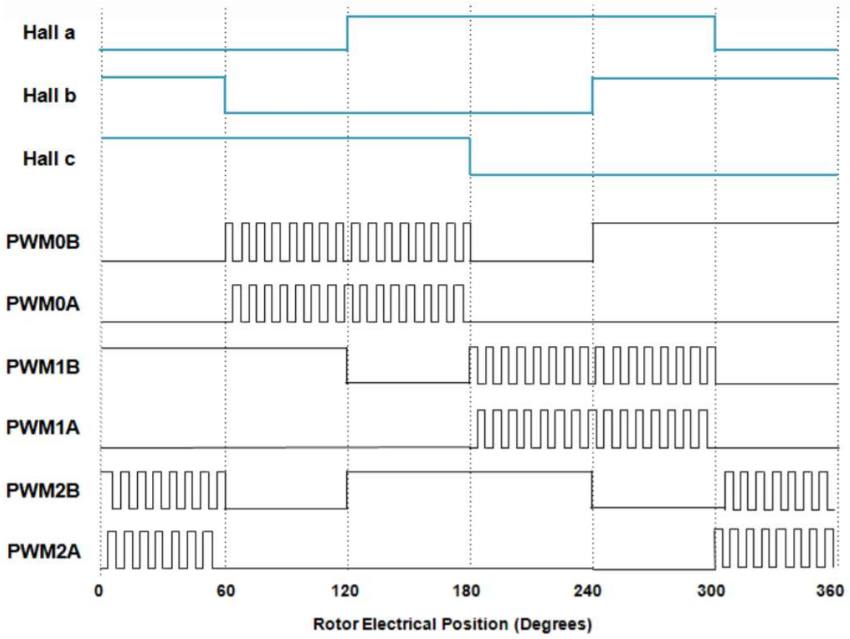
Disse er billige og godt dokumentert sammenlignet med Hoverboard motorkontrolleren. Motorkontrolleren trenger en input-spenning på 36VDC som den får fra batteriet og tåler 500W kontinuerlig. Den har justerbart pådrag som styres med et analogt 0-5V spennings-signal, og en reversmulighet som styres med et binært 0/5V-signal.



Figur 2.8: Prinsippskisse over BLDC med kontroller (Wilson 2011, 140)

Figur 2.8 er en prinsippskisse over motorkontrollerens funksjoner og virkemåte. Hall effekt-sensorene gir konstant tilbakemelding til motorkontrolleren om posisjon til rotoren. Denne informasjonen brukes til styring av motoren. I vår kontroller er det også mulig å måle dette signalet fra en utgang for å regne ut turtall.

For at motoren skal rotere må det i hvert øyeblikk sendes spenning inn på 2 av 3 faser, 1 fase med positiv spenning, og 1 med negativ. Hvilke faser som blir trigget bestemmes av Hall effekt sensor kombinasjonen. I motorkontrolleren er det 6 MOSFET som gjør denne jobben.

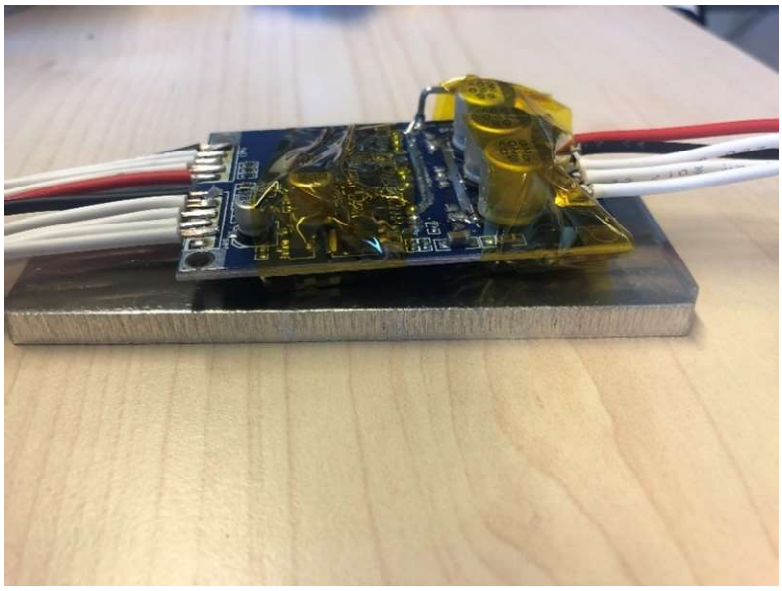


Figur 2.9: PBM-signaler fra MOSFET (Gao 2013).

Motorkontrolleren bruker PBM, Puls Bredde Modulasjon, for å endre rotasjonshastigheten til motoren. Det betyr at MOSFET i motorkontrolleren generer et pulstog ved å bytte på å være høy og lav. Forholdet mellom tiden den er aktiv på og perioden kalles duty cycle. Gjennomsnittet av dette pulstoget er det motoren tolker som sitt innsignal. Er duty cyclen 100 % går motoren på maks fart, mens er den 0 % gir den ingen spenning til motoren. For å justere pådraget krever motorkontrolleren et analogt signal mellom 0-5V. Pådragssignalet blir deretter brukt til å bestemme duty-syklusen til MOSFET.

Motoren har konstant moment, så endring av pådraget blir bare å endre turtall på motoren.

MOSFET på motorkontrollerene genererer en del varme som bør håndteres. Det er laget en kjøleplate av aluminium som hjelper å spre varmen generert av transistorene.
MOSFET ble teipet over med isolerende teip for å ikke kortslutte dem over kjøleribben. Her kan også en isolerende og varmeledende lim brukes.



Figur 2.10: 500W BLDC Motorkontroller med halleffekt-sensor tilbakemelding. Metallplaten under sprer varmen. Gul teip sikrer mot kortslutning ved kontakt med metallplaten

2.2.4 Bremsing

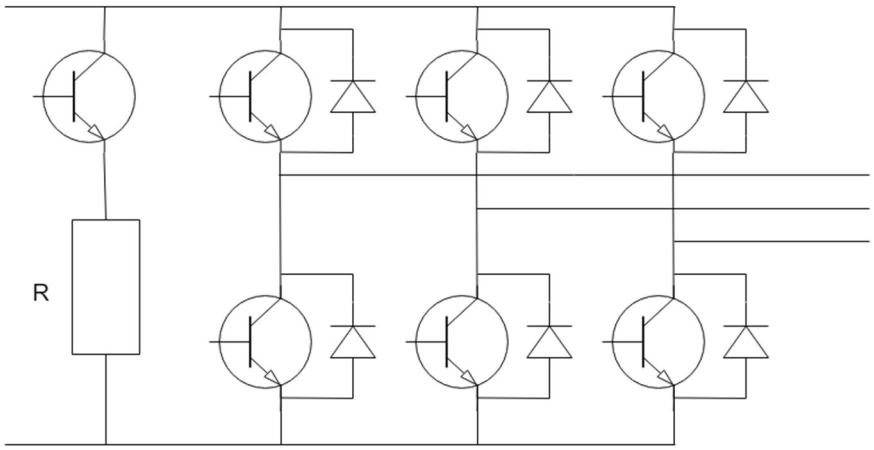
For enhver konstruksjon er det viktig med sikkerhet. Den mest universale sikkerhetsfunksjonen for en farkost er brems. Gode bremser gir også gode manøvreringsegenskaper som gjør det igjen lettere å styre. For plattformen er det undersøkt en rekke bremsemuligheter. Vi skal nå gå kort gjennom hvilke bremsemuligheter som ble undersøkt, hvordan de virker og hvilken metode som ble implementert.

Bremsing av elektriske motorer kan deles inn i 2 hovedkategorier. Elektrisk bremsing og mekanisk bremsing. Elektriske motorers roterende last har kinetisk energi proporsjonal med lastens treghetsmoment og kvadratisk med lastens roterende fart (Wilson 2011). Det er derfor mye energi som må håndteres under bremsing. Med dette som utgangspunkt så vi på en rekke bremsemetoder.

Elektrisk bremsing

En type elektrisk bremsing er det som kalles **Rheostatisk bremsing**. Her kutter man spenningstilførsel til motorviklingene og lar motoren gå som generator. Magnetfeltet i rotor går da fortere enn magnetfeltet i stator, og induserer en spenning i statorviklingene

og sender effekt tilbake til DC-bussen. For å bli kvitt denne effekten kan det legges inn en bremsemotstand rett før transistorene i serie med en transistor, som vist i figur 2.11.



Figur 2.11: Bremsemotstand R, ligger i serie med egen transistor

Transistoren bestemmer hvor lenge bremsemotstanden skal ligge inne og vil brenne av effekten. Denne metoden krever både utregninger i forbindelse med størrelsen på bremsemotstanden og en styring til transistoren for å kunne bestemme hvor stor grad av bremsing man ønsker. Det innebærer også at det genereres varme i motstanden, som da kan trenge kjøling.

Regenerativ bremsing baserer seg på mye av det samme som Rheostatisk bremsing. Forskjellen er at energien ikke blir svidd av som varme i en motstand, men heller sendes tilbake til batteriet. Denne metoden krever at motoren må generere høyere spenning enn det som er på batteriet, som vil si over 36 V. Testing viser at dette krever et turtall på 588 rpm. Da plattformen kommer til å operere i gangfart mye av tiden vil ikke dette være en brukbar løsning. Ved høye turtall vil det være vanskelig å ha kontroll på ladingen i den forstand at den ikke vil vite når den skal slutte å lade batteriet selv om det et fullt. Både Rheostatisk og regenerativ bremsing krever at hjulene roterer for å kunne bremse, som vil si at man ikke kan stå i ro i for eksempel oppoverbakker. Denne formen for bremsing kaller dynamisk bremsing.

DC injeksjon baserer seg ikke på dynamisk bremsing. Istedenfor sendes en kraftig DC spenning inn på 2 av fasene for å retardere magnetfeltet i rotoren og på den måten stoppe motoren. På denne måten vil plattformen kunne stå i ro i oppoverbakker. Med denne

bremsemetoden forlater ingenting av energien ut fra motoren, men blir heller avgitt som varme grunnet motstanden i motorviklingene. Dette gjør at motorviklingene vil bli varm.

Plugging er en annen måte der man vrir om på 2 av fasene for å stanse motoren. Denne metoden er aggressiv og kan være krevende for motoren i og med at det blir stor forskjell i rotorens og statorens magnetfelt. Dette gjør at det vil bli store strømmer i rotoren som igjen vil generere varme i motorviklingene. Dette kan være skadelig for motorisolasjonen.

Den siste elektriske bremsemåten vi så på var **kortslutning av fasene**. Ved denne metoden blir spenningstilførsel til motorene blir kuttet som gjør at motoren går som generator. Når en generator går med fasene kortsluttet genereres en stor strøm i statorviklingene pga ankerreaksjonen. Dette lager igjen et kraftig magnetfelt som motvirker rotorfeltet og bremser motoren. Bremsing vil bare skje når motoren roterer, og det vil generes mye varme i motorens viklinger.

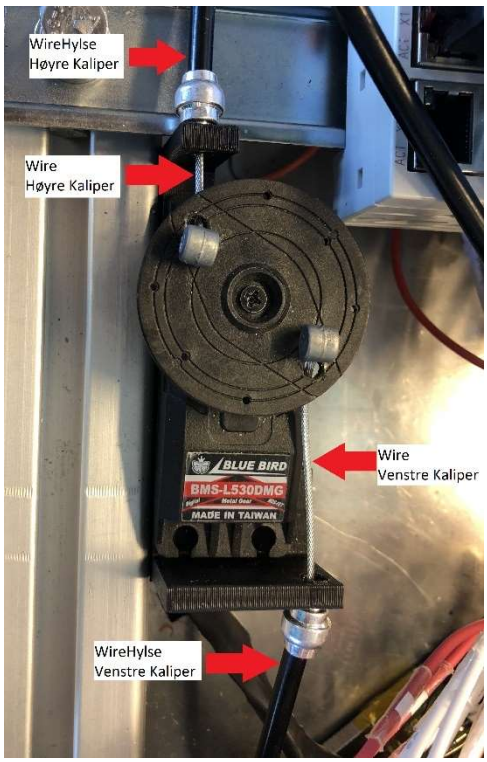
Ved mange av disse bremsemetodene må hjulene være i bevegelse for å ha en effekt. Dette er ikke ønskelig da det vil være gunstig å ha mulighet til også å stå i ro på flater med helning. Det vil også genereres store strømmer og mye varme i motorene ved noen av metodene. Uten god kjøling til motorene kan dette være skadelig for motorisolasjonen

Motorkontrolleren som brukes på plattformen støtter få av disse metodene. For å ta i bruk mange av disse metodene ville det vært nødvendig å enten anskaffe dyrere motorkontrollere som støtter bremsing, eller lage egne bremsekretser der man forbikobler motorkontrolleren. Å lage egne bremsekretser krever mange komponenter for å styre kretser og begrense strømmer. Det ville også vært nødvendig å forrigle kretsen slik at motorkontrollerene ikke kunne kortsluttes. Da dette krever mye utregning og kobling, samtidig som at det kunne blitt dyrere, valgte vi derfor en mekanisk bremsemetode som er robust og billig.

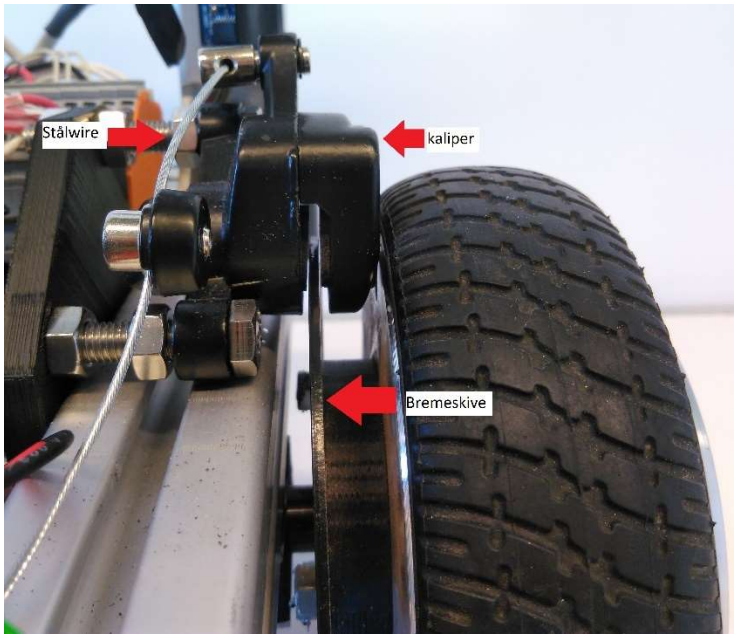
Mekanisk bremsing

Mekanisk bremsing er en velkjent, velutprøvd og trygg å måte å bremse en farkost på. Vi valgte en typisk metode som blir brukt på mange sykler i dag, skivebremse. Skivebremsene er beregnet for store laster som for eksempel en sykkel med en person. Dermed anser vi det som tilstrekkelig med brems på 2 hjul. Under utprøvning av plattformen har bremselengde vært uproblematisk. På denne måten får vi brems som er effektiv

ved lave turtall også. Bremselinjen består av en servomotor som betjener to stålwire som går til hver sin kaliper.



Figur 2.12: Servomotor for bremsing



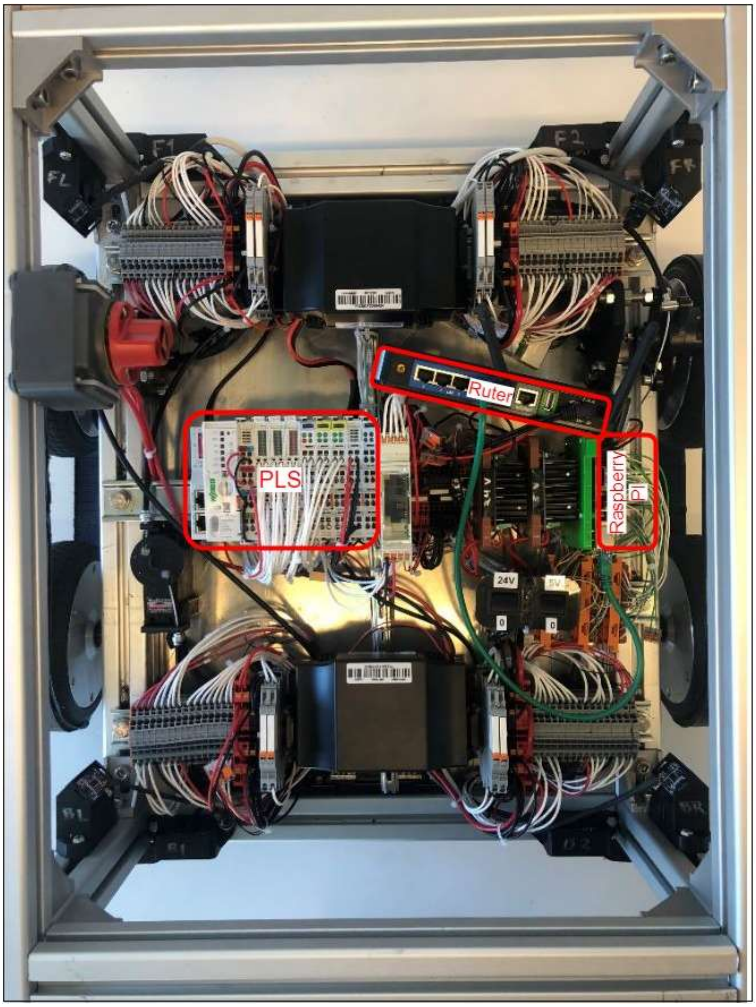
Figur 2.13: Bremsekaliper og skive høyre fremhjul

Kaliperene er fastmontert med braketter over bremseskivene som er skrudd fast på hvert forhjul. På denne måten kan man enkelt regulere grad av bremsing, og det kreves ikke

rotasjon på hjulet for å kunne bremse. Ulempen med denne bremsevarianten er at klossene slites ned over tid, og må byttes ut.
Servomotoren vår er en Blue Bird BMS-l530DMG og forsynes med kraft fra en dedikert spenningsomsetter på 5 volt. Styringen krever et pulsbreddemodulert signal som en Raspberry Pi leverer via en valgfri GPIO pinne. Bredden på pulsen avgjør posisjonen til hjulet på bremseservoen mellom 0 og 180 grader.

2.3 Styringsenheter

Som hovedplattform for styring og overvåking av plattformen er det blitt brukt PLS. Det er også brukt Raspberry Pi til å kommunisere med bremseservo, ultralydsensorer og kommunisere med PlayStation 4 kontrolleren og så er Raspberry Pi tenkt som kommunikasjonsgrensesnitt til fremtidig utvidelser. Ved å bruke PLS holder vi oss til standard-signaler som 0-10V for analogt signal, og 0/24V og 0/5V for digitale signaler. Dette gir plattformen gjennomgående modularitet, i og med at det det da blir lettere å bytte ut komponenter og legge til nye komponenter på alle nivå. Wago PLS er selv en modulbasert PLS, som gir oss mulighet til å bytte ut enkeltmoduler eller utvide med ekstra moduler. For å knytte sammen PLS og Raspberry Pi er det blitt benyttet en trådløs ruter, som er koblet til Ethernet portene til enhetene. Denne setter da opp et trådløst nettverk som andre enheter kan koble seg til for å få tilgang på Human Machine Interface (HMI). Via dette nettverket kan PLS programmeres og Raspberry Pi konfigureres.



Figur 2.14: Styringsenheter til plattformen

2.3.1 Programmerbar Logisk Styring

Programmerbar Logisk Styring (PLS) er en programmerbar prosessorenhet, lignede en datamaskin, som brukes for å styre blant annet industrielle prosesser. PLS er veldig robust, sikker, modulær og godt dokumentert, og er brukt i industrien for styring, regulering og måling. Plattformen er konstruert for både testing og undervisning, og på grunn av at PLS er en del av undervisningsfagplanen, ble PLS valgt som hovedstyringsenhet. I fagene «Automatiserte systemer» og «Systemtenkning» ble vi introdusert for Wago PLS. Wago PLS er modulbasert og godt dokumentert, noe som gjorde det enkelt å implementere i plattformen. Modulene kan ansees som hvert sitt grensesnitt mellom interne variabler og

eksterne elektriske signaler. Wago PLS programmeres med et program som heter eCockpit! og har programmeringsspråk som baserer seg på den industrielle standarden IEC 61131-3 (basert på codesys).

I plattformen er det benyttet følgende Wago PLS-moduler.

Tabell 2.1: PLS moduler

Hovedmodul 750-8101	Hovedmodul – Ethernet Programmerbar Feltbus kontroller med 2 x Ethernet tilkoblingspunkt. Har tilkobling for 24VDC som videreføres ut på koblingsskinner til neste modul.
RTD 750-450	4 kanalers Resistance Temperature Detector (RTD)-modul. Brukes for temperaturmåling i motorer og ESC'er. Det brukes 2 slike moduler. Dette gir totalt 8 innganger.
DO 750-1504	16 kanalers Digital Output-modul for 0/24VDC. Brukes for å styre inngangene Enable og Forward/Reverse til ESC gjennom releer.
AO 750-559	4 kanalers Analog Output-modul, som leverer 0-10VDC. Brukes for å regulere pådraget mellom 0 og 5 volt til ESC.
Up/Down Counter 750-404/000-005	2 kanalers Opp/Ned-teller-modul for 0/24VDC med 5 kHz oppdateringsfrekvens. Brukes for å måle turtall på motorene. Det benyttes 2 slike moduler. Dette gir 4 kanaler totalt.
Power Measurement Module 750-494/000-005	Effektmålermodul, med 2 spennings- og strømmålingsinnganger for DC. Strømmåling må gå via Shunt motstand (0,2 Ohm). Kan måle opp til 277VDC og 20 kA DC via ekstern shuntmotstand. Brukes for å overvåke strøm ut fra hvert batteri og felles spenning på batteriene.
Endemodul 750-600	Endemodul som fullfører den interne data kretsen (K-bus terminering) og sikrer korrekt dataflyt.

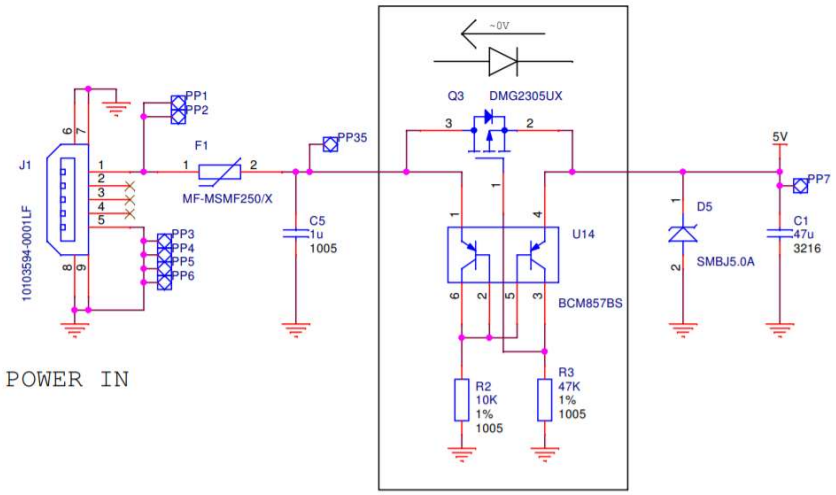
2.3.2 Raspberry Pi

For å knytte sammen de ulike delene i denne oppgaven har vi benyttet oss av en Raspberry Pi 3, denne fungerer da som server for NodeRed og kommuniserer med fjernkontrollen. Raspberry Pi er en lavenergi-datamaskin basert på operativsystemet Linux. Den er designet for å være et billig programmeringsverktøy og har en stor brukermasse som bidrar til gode Open-Source kilder med dokumentasjon, som er hovedårsakene til at vi ønsket å benytte oss av den i denne oppgaven. Da den blir benyttet i utallige prosjekter, er mye gjort og testet før, og hjelp er da tilgjengelig via forum og blogger.

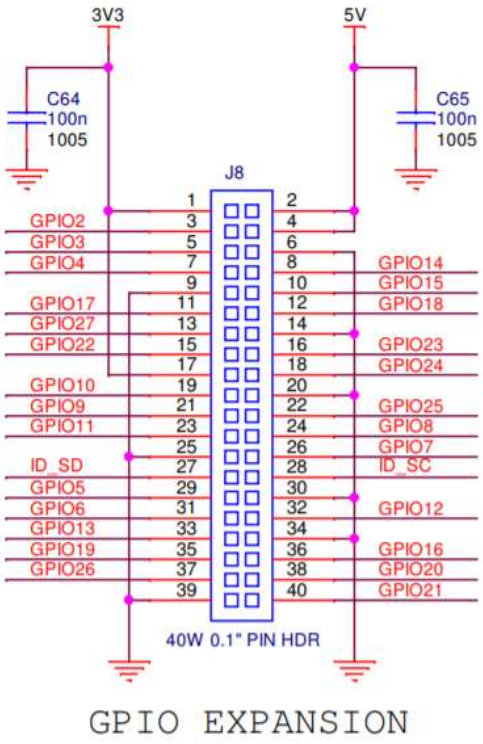
Raspberry Pi har en rekke muligheter for tilkoblinger, i denne oppgaven benytter vi oss av disse tilkoblingene:

Tabell 2.2: Benyttede tilkoblinger til Raspberry Pi

WiFi	Benyttes til å koble til et eksisterende nettverk. Dette gir enheter på nettverket tilgang til å styre Raspberry Pi og NodeRed
Bluetooth	PlayStation 4 kontrolleren kobles til Raspberry Pi via Bluetooth. Dette blir primært benyttet som fjernstyring av plattformen via NodeRed.
Ethernet	Raspberry Pi kobles til PLS via Ethernet og kommuniserer ved å bruke OPC Unified Architecture. Denne kommunikasjonen styres av NodeRed.
GPIO	General Purpose Input Output, Raspberry Pi har 26 kombinerte inn-/ut-ganger. Disse har en 3,3 volt logikk som vi benytter til å styre bremse-servoen og alle ultralydsensorer (totalt 17 GPIO). Disse styres også via NodeRed.



Figur 2.15: Oversikt over USB Spenningsforsyning til Raspberry Pi. Spenningsforsyning er koblet inn etter sikring F1 og filter (Raspberrypi.org 2018, 34).



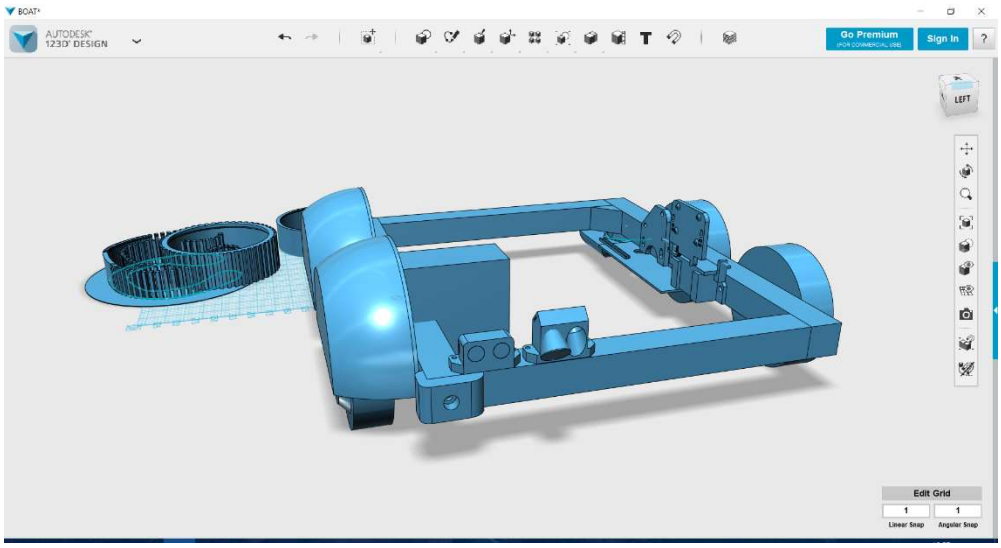
Figur 2.16: GPIO header strømfordeling (Raspberrypi.org 2018). Oversikt over hvilke innganger som er brukt finnes i vedlegg 8.9 (Raspberry Pi GPIO tilkoblinger)

Figur 2.15 og 2.16 er to utsnitt av linjeskjemaene til Raspberry Pi. Det første skjemaet viser hvordan Micro-USB porten er koblet opp mot 5V-linjen. Figur 2.16 viser hvordan 5V-linjen er koblet til GPIO-Header. Vi har tilkoblet spenningskilden til pinne 2 for å

forsyne Raspberry Pi med energi. Som vi ser ut fra figur 2.15 er vi da tilkoblet på baksiden av sikring F1 og filteret. Det vil si at Raspberry Pi er ikke kortslutnings- og overbelastnings-beskyttet eller beskyttet mot feil polaritet. Til normalbruk har dette ingen betydning, da denne beskyttelsen har til hensikt å beskytte Raspberry Pi ved tilkobling av utstyr på GPIO header. Ved utvidelser må det derfor tas hensyn til dette, og ekstra utstyr bør kobles til/fra når Raspberry Pi ikke er tilkoblet spenning. Det anbefales å koble til spenningskilden via USB-tilkoblingen for å få sikkerhetsfunksjonen tilbake ved tilkobling av nytt utstyr.

2.4 Design av deler via 3D-printing

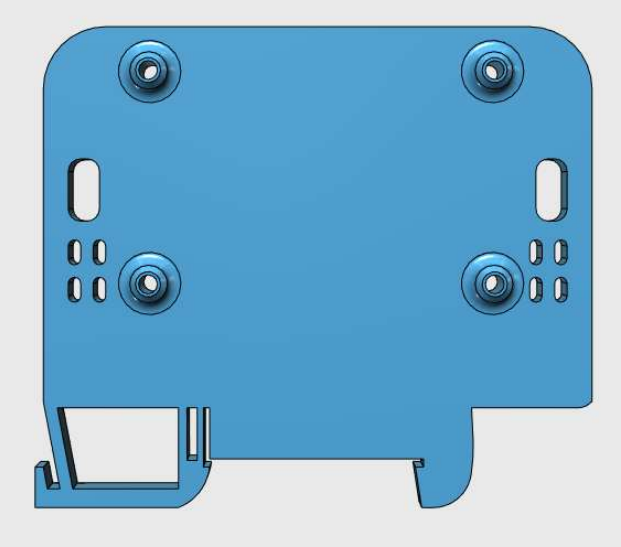
Denne oppgaven baserer seg på å lage noe som ikke allerede eksisterer, da blir behovet for å produsere egne deler stort. Vi har i stor grad kunne utnytte hyllevare fra en rekke leverandører, men integreringen av disse komponentene krever tilpasning. Da vi fra før hadde erfaring med bruk av 3D-printing, og har tilgang på 3D-Printer både privat og via Sjøkrigsskolen, ble det naturlig å utnytte dette i oppgaven.



Figur 2.17: 123D-Design modell av plattformen

For det meste er 3D-Printing benyttet til å lage braketter til diverse komponenter som har vært nødvendig å installere på plattformen. En Raspberry Pi brakett for montering på DIN-skinne ble designet som Open-Source av bruker Zwieberteje på Thingiverse.com (Zwieberteje 2017). Resten er designet selv i programmet Autodesk 123D-Design.

For å feste alle elektroniske komponenter ble det laget egne braketter for ultralydsensorer, DC-DC omformere, servomotoren og batterier. Alle disse er lagt inn i en modell som vist i figur 2.17, på den måten har vi kunnet tilpasse dem både til rammen og hverandre.



Figur 2.18: 3D-modell av DIN skinne brakett for DC/DC-omsetter.

Figur 2.19: Ultralydsensor plassert inne i 3D-pirntet brakett, montert på ramme.

I tillegg har vi laget braketter til mekaniske formål – braketter for å feste bremsekalipere til rammen, adapter til bremseskivene og mutteradaptere for å kunne bruke standard metriske muttere i V-profilene. Alt dette er blitt skrevet ut i PolyLactic Acid plast. Det er en type hardplast som er laget av biomasse. For andre formål har vi også benyttet oss av en type myk plastikk. Dette har vi brukt til å lage beskyttelse for kabler som er ført gjennom bunnplaten, støtfangere og belter til hjulene.

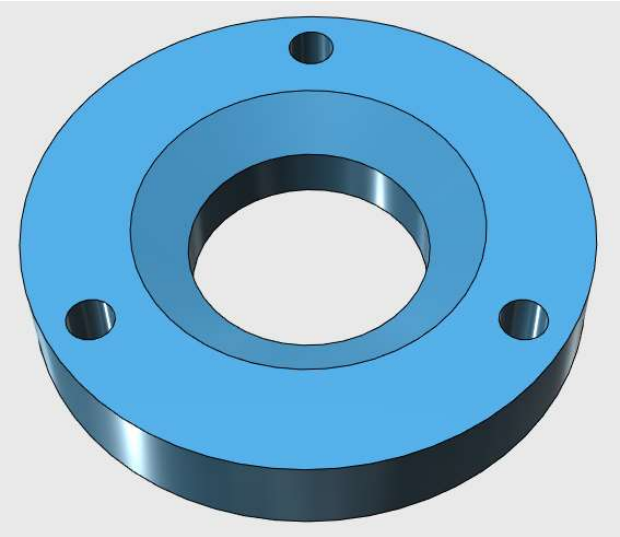
Den største fordelen med å benytte 3D-printing i design prosessen er muligheten til å teste plassering av de fysiske komponentene. Vi benytter oss av en 3D-modell for å kunne få en oversikt over plassering og størrelser på de enkelte komponentene, men det kreves av

og til at vi har tilgang på de fysiske gjenstandene for å kunne bedømme om designet er solid nok og faktisk passer der det er tiltenkt å stå. Servo-brakettene er et godt eksempel på dette, der vi designet og printet flere utgaver. Den første utgaven ble laget utfra en idé om hvor vi ønsket å plassere servoen og hvor wirene burde ligge. Da brakettene ble printet ut og plassert, ble det oppdaget at plasseringen av wirene ikke var helt optimale og at det ikke var tatt hensyn til servo-ledningen. Dette ble korrigert i modellen og det ble printet ut en oppdatert versjon.

En av utfordringene vi hadde med implementering av mekaniske bremser var å få festet bremseskiven til motoren. Motorene/hjulene vi har valgt er ikke designet for å kunne feste på ekstra utstyr. Vi måtte derfor lage en brakett som kunne holde bremseskiven sentrert på hjulet.

Figur 2.20: Deksel til motor med feste til bremseskive

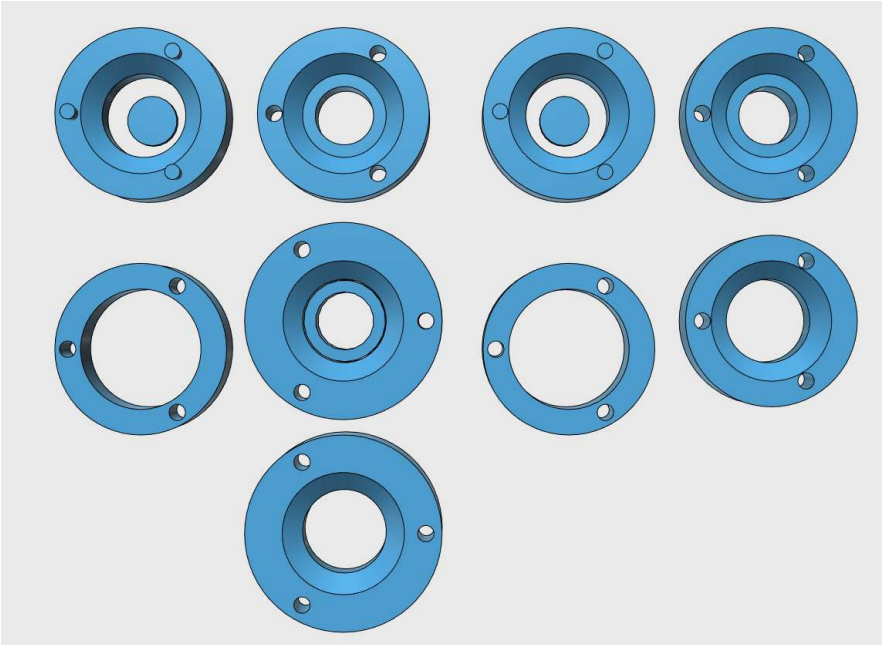
Bremseskiven festes med tre 6mm bolter, disse måtte vi feste til motordekselet med 120° forskyvning på samme sirkel for å unngå kast. For å kunne bore ut hullene nøyaktig, lagde vi et 3D-design av en brakett som også fungerte som mal. Ved å bruke skyvelære målte vi topp og bunn av kjeglen i senter av dekselet og kopierte formen inn i 3d-modellen.



Figur 2.21: 3D-Modell av bremseskive brakett

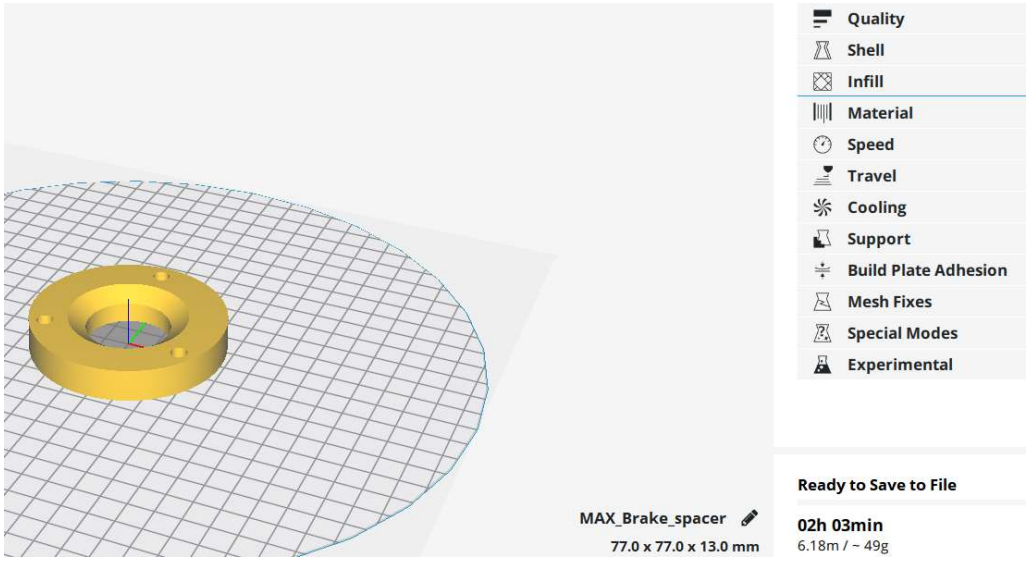
Resultatet ble en sirkel som kan legges over lokket til motoren og sentrerer seg selv, på den måten kan skruehullene bores ut nøyaktig selv med en håndholdt drill.

Figur 2.22: Bremseskive brakett i sort plast, sentrerer bremseskiven på motorlokket.



Figur 2.23: Versjoner av mal for bremseskive-brakett

Det er flere variabler som inntreffer i dette designet, det mest prekære er posisjonen til boltene som må ha tilstrekkelig avstand til statorviklingene. Derfor ble det laget flere modeller for å lettere kunne se forskjellene og vurdere hva som ville fungere best, men bare to versjoner ble printet ut.

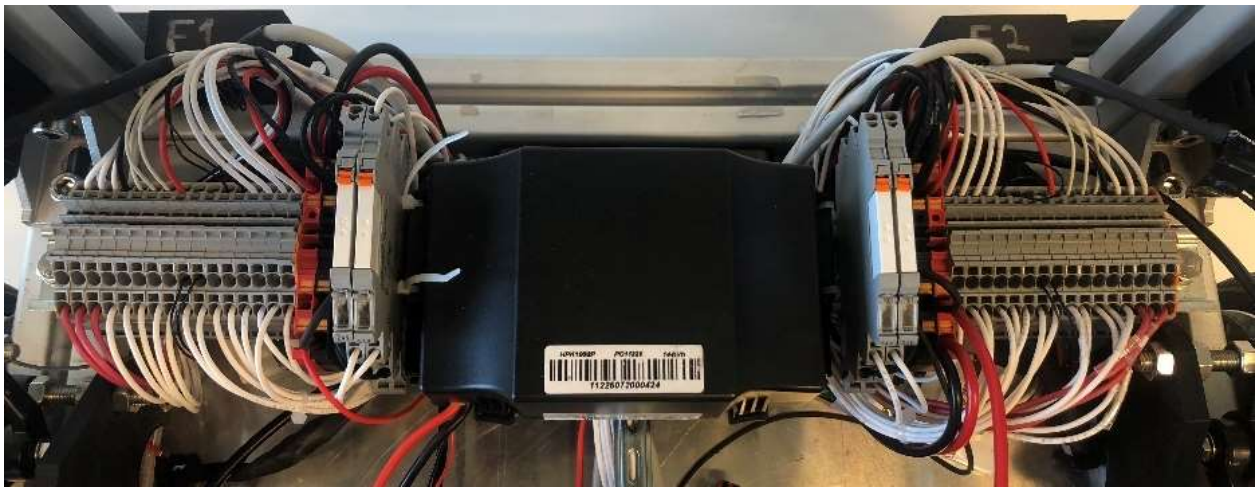


Figur 2.24: Bremseskive mal klar til printing

3D-printing kan være en tidkrevende affære, noe så enkelt som en sirkel kan trenge mer enn to timer på å skrive ut. Vi har derfor erfart verdien av å bruke mer tid på selve designet, for å kunne redusere antall deler vi printer. Dette er også kostnadsbesparende, da vi ikke kaster bort mer plastikk enn nødvendig.

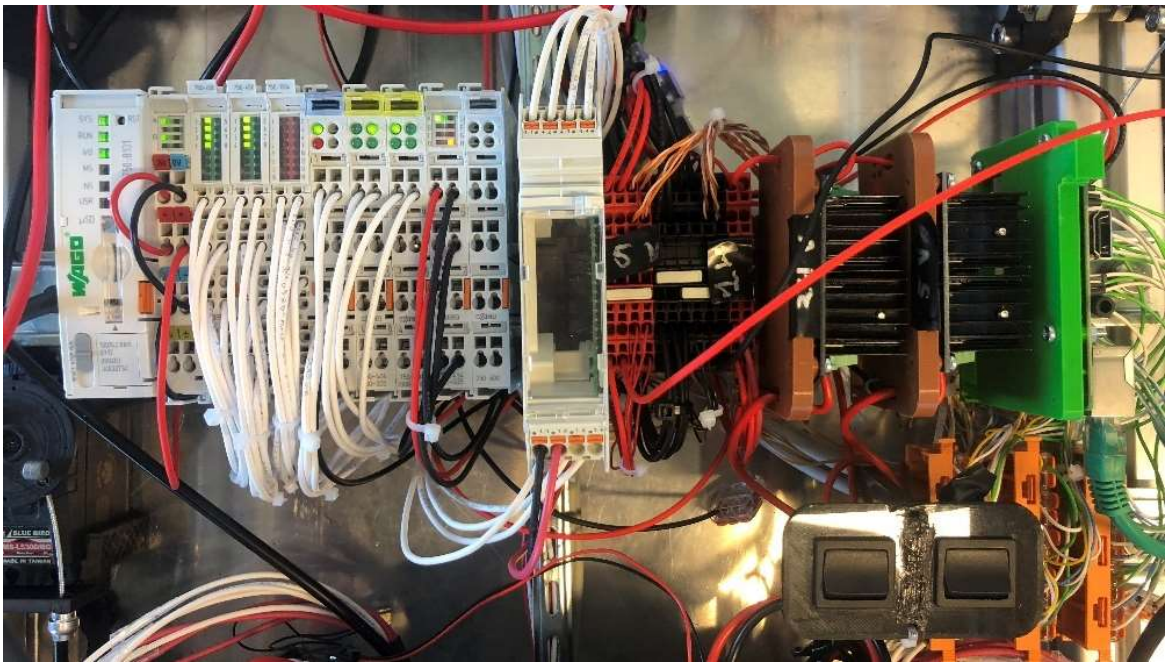
2.5 Kobling

I plattformen er det lagt opp 3 DIN-skinner som komponentene er montert på. På øverste DIN-skinne er batteri 1 montert på 3D-printete braketter. På hver side av batteriet er det påmontert to 24V releer og et sett med rekkeklemmer. Releene sørger for å oversette 24V-signalene fra PLS til 5V-signaler som motorkontroller leser. Rekkeklemmer er satt opp for å koble sammen motor, motorkontroller og PLS. Rekkeklemmer sørger for god modularitet ved å gjøre eventuell utbytting av motorkontroller og andre komponenter lettere. De sørger også for bedre tilgang for måling i alle koblingspunktene. For 36V hovedfordelingen har vi anskaffet mer robuste rekkeklemmer, som kan håndtere større strømmer. Motorkontrollerene er plassert under rekkeklemmene.



Figur 2.25: Øverste DIN-skinne

På midterste DIN-Skinne er Wago PLS med moduler montert. Ved siden av PLS står den egenproduserte transistormodulen som sørger for omsetting av 5V-signaler fra motorkontrollerene til 24V-signaler til PLS tellemodul (750-404/000-005). To DC-DC omsettere er montert med DIN-skinne-brakett i brun farge. Fra 36V omsettes det til 5V og 24V. Omsetterne forsyner de røde rekkeklemmene på samme DIN-skinne. De svarte rekkeklemmene er felles minus for alle spenningsnivåene.



Figur 2.26: Midterste DIN-skinne

Det er også koblet inn brytere for å slå av og på disse spenningsnivåene. Raspberry Pi er montert ytterst på høyre side med grønt DIN-skinne-feste. På undersiden av Raspberry Pi er Wagoklemme-holdere plassert for lettere tilgang til koblingene mellom Raspberry Pi og ultralydsensorene. Tilkoblingen er åpen og lett å koble til og fra, noe som gjør den modulær.

Hovedstrømsbryter er lagt inn for å bryte hovedstrømmen mellom batteriene og hovedfordelingen. Bryteren kan skille mellom batteriene og koble hvert enkelt batteri til hovedfordelingen, slik at bare batteri 1 eller batteri 2 forsyner hovedfordelingen. På denne måten kan man teste batterier individuelt ved behov.

Figur 2.27: Hovedstrømsbryter, med stillinger: OFF, 1, 1+2 og 2. Den velger om batteri 1, batteri 2 eller begge skal være tilkoblet hovedfordelingen.

Det er tilkoblet 2 ladeplugger til hovedfordelingen på 36V. Disse er tilkoblet etter hovedstrømsbryter som betyr at de kan forsyne komponentene selv om bryteren er AV og ingen batterier er tilkoblet. Fordelen er da at plattformen kan jobbes med uten å belaste batteriene, for å lade batteriene må hovedstrømsbryteren være innkoblet.

2.5.1 Kabel og vern

De fleste koblinger mellom komponenter er gjort med 0,5 mm^2 og 0,75mm^2, da strømtrekket mellom dem er i underkant av 1 A, se Test 7.6 (Test av effektforbruk). Det som potensielt kan trekke mye strøm er motorfasene som forsynes fra motorkontrollerutgangene. Motorkontrollerene forsynes direkte fra hovedfordelingen på 36V. Det er derfor viktig at kablene på hovedfordelingen er godt dimensjonert. Ved maks pådrag på alle motorene trekkes det teoretisk nærmere 40 A totalt. Fordelt på 2 batterier blir det da 20 A.

1400W / 36 V = 38,9 A.

På grunn av lengden på plattformen blir det relativt korte kabler (<1m), nærmeste forleggningstype er åpen, og driftstemperatur på plattformen er 30 ºC. Ut i fra dette så vi det som sikkert å legge opp 2,5 mm^2 på hovedfordelingen. Dette er også det samme tverrsnittet som batteriene blir levert med.

Spenningen på anlegget er maksimalt 42,1V. Dette anses som ufarlig ved berøring, og det er derfor ikke tatt hensyn til ved å sette inn egne vern.

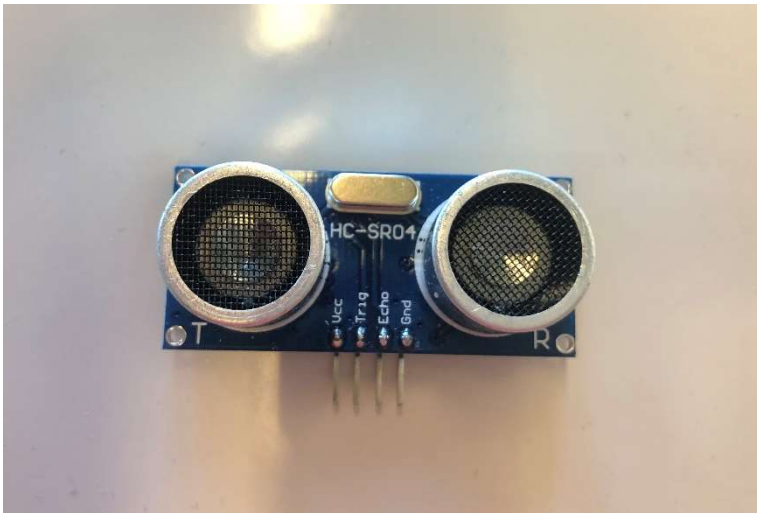
2.6 Overvåkingssensorer

Den autonome testplattformen benytter en rekke overvåkingsmetoder for ulike verdimålinger. Avstandsmåling er gjort med arduino-baserte ultralydsensorer. Overvåking av temperatur i motorene og motorkontrollerene er gjort med PT100-elementer. Turtallsmåling i motorene, måles med en Wago teller-modul (750-404/000-005). Strøm- og spenningsmåling til batteriene blir målt med en Wago effektmålermodul (750-494/000-005). I delkapitlene under drøftes hvilke målinger som blir gjort og hvilke sensorer som brukes til dette.

2.6.1 Avstandsmåling

Avstandsmåling er et viktig moment ved enhver fjernstyrt/autonom plattform. Foreløpig blir avstandsmålingen hovedsakelig brukt til sikkerhetsfunksjoner som automatisk bremsing ved for nær kontakt. Fremover er det viktig å ha avstandsmålere på denne plattformen for at disse skal kunne brukes til å programmere diverse enkle autonome funksjoner. Dette vil også lettere muliggjøre videre utbyggelser gjennom nye bacheloroppgaver.

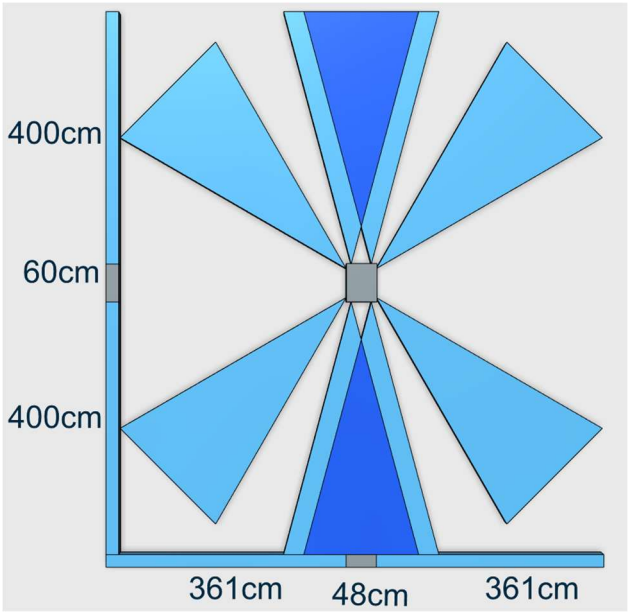
Når det kom til avstandssensorer baserte vi valget på hvor lett det var å koble opp, installere og implementere i programmeringen. Vi så først på Biltemas ryggesensorer tiltenkt biler. De var enkle å anskaffe. Vi gikk vekk fra disse da de var fast innstilt med en gitt avstand der de ga signal og ikke signal på en binær måte. Dette gjør det umulig å programmere med justerbare avstander som det trengs på denne modellen, særlig i utviklingsfasen.



Figur 2.28: Ultralydsensor HC-SR04, brukes til avstandsmåling

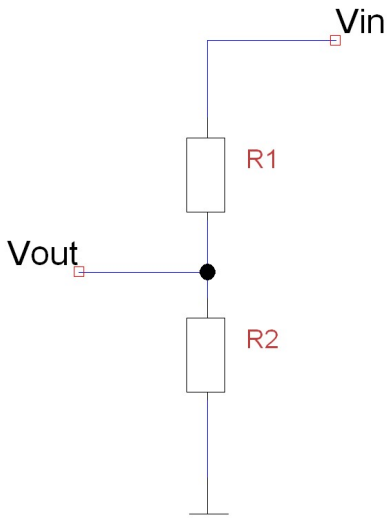
Videre ble arduino-baserte Ultralydsensorer (HC-SR04) testet. Disse er billige, lette å jobbe med grunnet god markering, og store pins sørger for god lodding. Hver av disse sensorene har 4 pins: 5V + (VCC), GND, Trigger Pulse Input (TRIG) og Echo Pulse Output (ECHO), se figur 2.28. Vi tilfører spenning på VCC og GND, og sender et innsignal til TRIG (5VDC), som gjør at sensoren sender en ultralyd puls. Hvis denne pulsen treffer et objekt innenfor rekkevidden sendes pulsen tilbake til sensoren, og sensoren bruker s=v*t for å finne ut hvor langt vekke objektet er. Den sender da et digitalt 5V signal tilbake ut fra ECHO. Ved å koble ultralydsensoren rett på GPIO på Raspberry Pi kan man bruke noden rpi-srf i NodeRed for å tolke signalet og gi ut en avstand. Dette gjør det lett å implementere disse sensorene i programmeringen.

Det ble 3D-printet monteringsbraketter for å montere disse sensorene på rammen, (se figur 2.19). Vi monterte 8 ultralydsensorer, 2 foran, 2 bak, 2x2 på hver side med 45º vinkel fram og bak. På denne måten har vi god kontroll på foran og bak av plattformen, med redundans. Sensorene har 4 meters rekkevidde, med en nøyaktighet på 2 cm ved lengre avstander, se Test A.2. Den oppgitte spredningsvinkelen var på 15º fra senter (dekker 30º synsfelt), men målt spredningsvinkel viste seg å være 25º på hver side av senter, altså 50º totalt (Test A.2). Montering av sensorene ble gjort med den teoretiske spredningsvinkelen i tankene. Vi valgte å montere sensorene med overlapp på grunn av at dette ville gi plattformen sikkerhet, i den forstand at hvis én av sensorene ikke trigget av noe rett foran ville den andre gjøre det. Dessuten er programmet bygget opp på den måten at det ikke var interessant å vite hva som var på hver side av plattformen, men heller ha god oversikt foran og bak. Figur 2.29 viser de forskjellige sonene som dekkes av avstandssensorene, med den teoretiske spredningsvinkelen. På grunn av universale monteringsbraketter kan sensorplasseringene endres uten for mye arbeid, og bør optimaliseres ut i fra hvordan man vil at programmeringen skal fungere.



Figur 2.29: Teoretiske avstandssensor-soner. Grafene gir lengder i full skala

Et viktig punkt i integreringen av disse sensorene er kommunikasjonen til Raspberry Pi. Sensoren sender ut et 5V signal ut fra ECHO pinnen, mens GPIO på Raspberry Pi går på 3,3V logikk og vil bli ødelagt om de får 5V over lengre tid. For å løse dette problemet har vi tatt i bruk enkel elektroteknikk og laget en spenningsdelingskrets.



Figur 2.30: Spenningsdeling Ultralydsensor (MODMYPI 2017), Vin=5V, Vout=3,3V, R1=1,13kΩ, R2=2,2kΩ

V_{in} = 5V kommer ut fra ultralydsensoren, og V_{out} = 3,3V kommer inn til Raspberry Pi. For å beregne resistansene som må brukes må vi starte med enkel spenningsdeling:

$$Vin = I * (R1 + R2)$$

$$Vout = I * R2$$
$$I = \frac{Vout}{R2}$$

$$Vin = Vout * \frac{(R1 + R2)}{R2}$$
$$\mathrm{R1} = R2 * \frac{(Vin - Vout)}{Vout}$$

Da det er forholdstallet mellom R1 og R2 som er viktigst her, kunne vi velge størrelse på R2. Vi valgte R2 på 2,2 kΩ for å få lav tapsstrøm, 1,5 mA. Inngangsresistansen til GPIO må være større enn spenningsdelingen, og dette kan være en begrensning for motstandsvalget, men verdiene fungerte til dette fungerer for R2-verdien settes inn i formelen for å regne ut verdi for R1. Setter vi inn R2=2,2kΩ, V_{out}=3,3V og V_{in}=5V.

$$R1 = \frac{2{,}2 * (5 - 3{,}3)}{3{,}3}$$
$$R1 = 1{,}13k\Omega$$

Utregningene våre sier at vi da skal bruke 1,1 kΩ som R1 og 2,2 kΩ som R2.

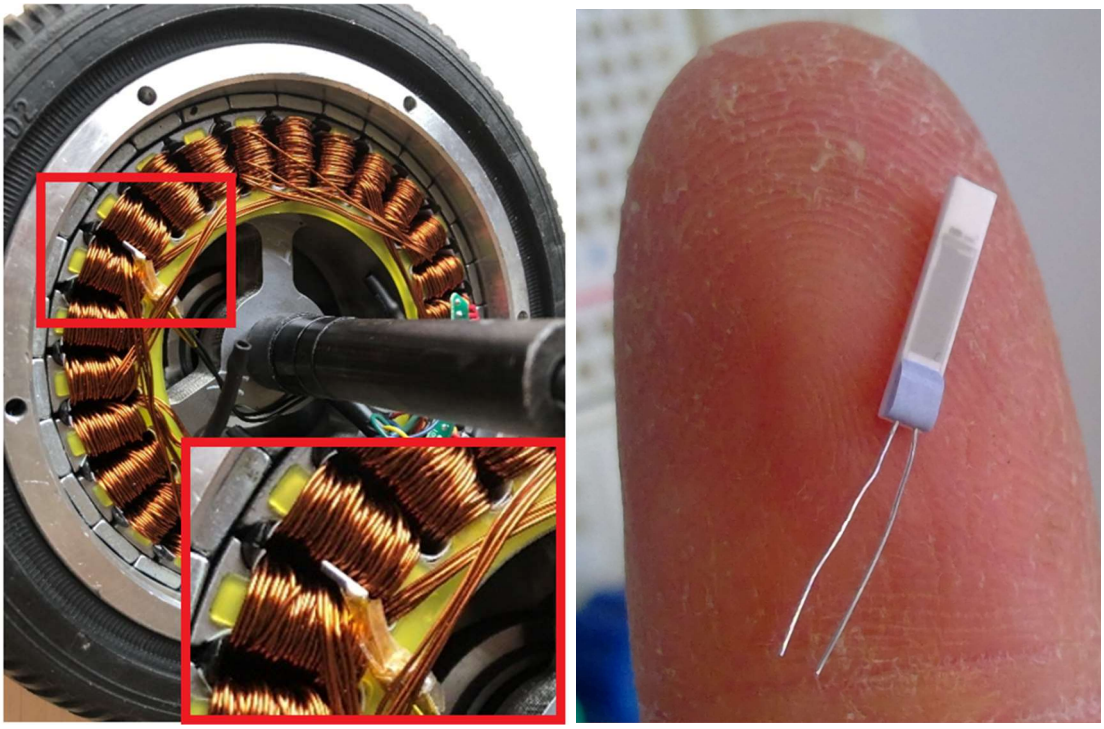
Den store ulempen med å bruke disse avstandssensorene er at alle sensorene sender lydbølger med samme frekvens. Dette gjør at lydsignalet som blir sendt av en sensor kan også bli registrert av en annen sensor. Dette gjør at det blir mye støy hos hver enkelt sensor. For å unngå dette problemet kan vi omprogrammere sensorene til å sende ut singaler med forskjellige frekvenser. En annen måte er å kun la en sensor sende lydsignal av gangen, mens alle andre lytter etter retursignalet. På denne måten kan man registrere det samme signalet med flere sensorer og dermed krysspeile ut hvor objektet ligger i forholdet til plattformen.

2.6.2 Temperaturmåling

I en konstruksjon med elektriske komponenter, kan det genereres en del varme, spesielt i komponenter som motorer og motorkontrollere. Varme kan potensielt skade elektriske komponenter og skade kabelisolasjon. Permanentmagnetene i rotor kan også bli demagnetisert hvis rotor blir for varm. Det er derfor viktig å overvåke temperatur. Valg av

temperatursensorer falt på PT100-elementer. PT100-elementer er platina-baserte motstander som endrer resistans med temperaturen. Ved 0º Celcius er resistansen 100 Ω, og korrelasjonen er tilnærmet lineær med et stigningstall på 0,5 Ω/ºC. Sensoren dekker et temperaturområde på -70 ºC til 600 ºC. Vi valgte tynn-film PT100 fordi de er små og dermed tar lite plass og kan plasseres nesten hvor som helst. PT100 elementer kan lett kobles opp mot Wago PLS ved bruk av passende modul (her 750-450).

Vi valgte å overvåke temperatur der det blir størst varmgang. I vår konstruksjon blir det i motorene og i motorkontrollerene.



Figur 2.31: PT100-element. Synlig som liten hvit komponent og ligger i statorviklingene. Bildet ved siden av gir en forståelse av størrelse

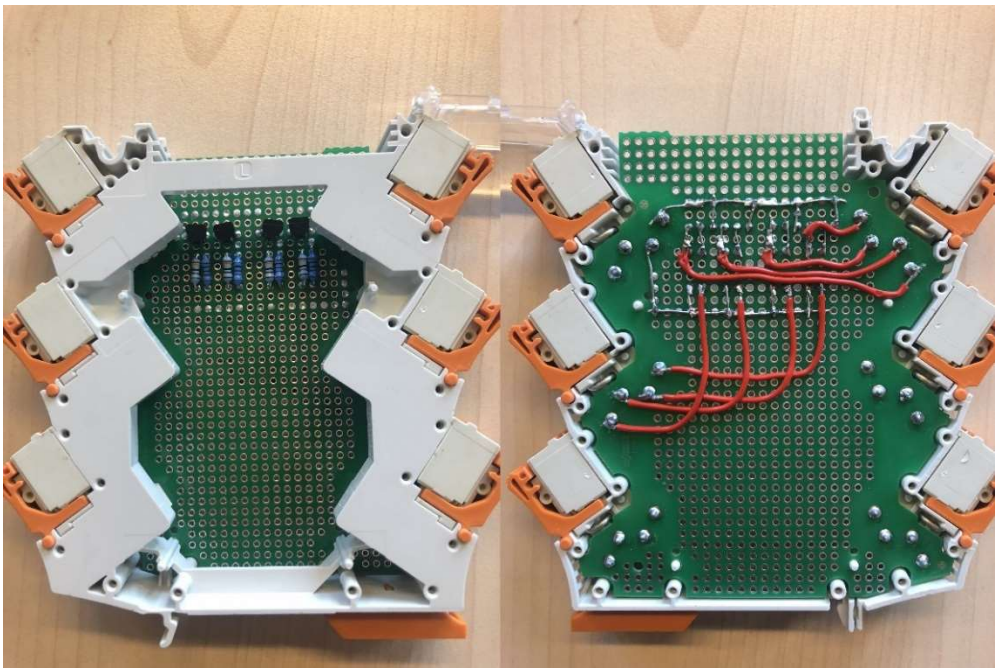
Som figur 2.31 viser er disse type sensorene veldig små og derfor meget anvendelige. De kan plasseres stort sett hvor som helst.

2.6.3 Turtallsmåling

Turtallsmåling er en viktig måling i konstruksjoner med motordrifter for å hele tiden ha oversikt over hvor hardt man kjører motorene og for å måle fart. I motorene vi har måles turtall ved hjelp av Hall effekt sensorene som er innebygget i motoren.

Det er 30 magneter i motoren, som ligger annenhver med pluss og minus inn mot statoren. I motoren ligger det 3 Hall effekt sensorer som ligger med elektrisk 120º forskyvning. Sensorene reagerer på magnetfeltene til permanentmagnetene i rotor, og vil da gi et høyt signal eller et lavt signal ettersom det permanentmagneten nærmest er negativ eller positiv. «1» for sør-magnet, og «0» for nord-magnet. Én sensor vil gi 15 høye 5V-signaler for en omdreining. Tre sensorer gir da totalt 45 5V-signaler i løpet av en omdreining til motorkontrolleren vår. Med 45 signaler pr omdreining kan antall signaler i minuttet bli mange. Ved 15 km/h roterer hjulene 482 ganger i minuttet, som igjen gir 21690 5V-pulser i minuttet. For å klare å ta opp alle disse signalene må vi da ha et måleapparat med høy oppdateringsfrekvens. Wago har Opp/ned teller-moduler til slike formål med svært høye oppdateringsfrekvenser. Vi har brukt slike moduler i tidligere fag, så da falt valget naturligvis på disse.

Disse modulene har ikke WAGO tilgjengelig til 5 volt, som våre ESC'er benytter, så vi. Vi måtte omsette det digitale 5 volt signalet til 24 volt for å kunne bruke standard modulene til WAGO.



Figur 2.32: Selvlagd transistor modul til omsetting av spenning fra 5V til 24V for turtallsmåling.

Vi laget derfor en egen modul med fire transistorer, som omsetter 5 voltsignalet til et 24 volt signal, kretstegning vises i Vedlegg B.7 Transistor Modul koblingsskjema. Måten vi

valgte å koble opp kretsen på inverterer signalet fra ESC, men til dette formålet har det ingen praktisk betydning.

2.6.4 Strøm- og spenningsmåling

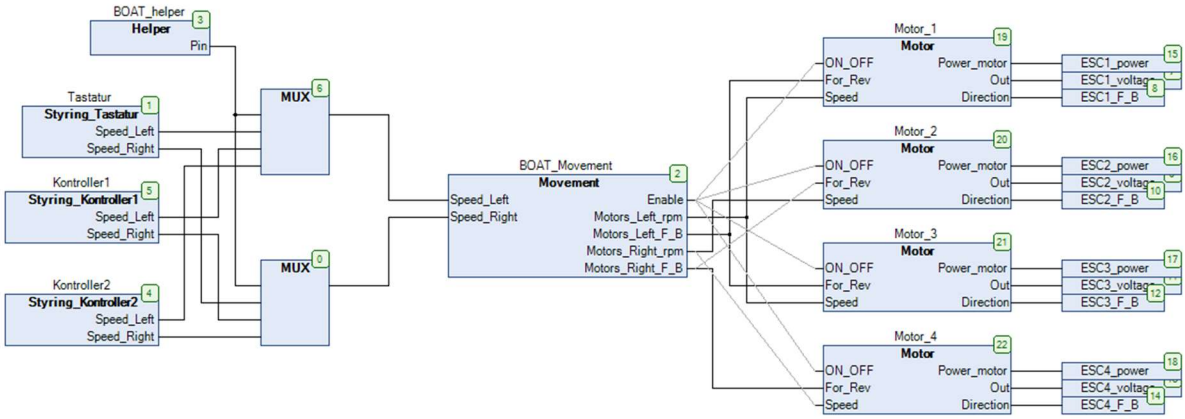
Med elektrisk fremdrift og batterier er det naturlig å ha strøm- og spenningsmåling i oppgaven. Både for å ha oversikt over strømtrekk til komponentene, men også for å ha status på batteriene. For å måle strøm og spenning har vi en PLS-modul som er kompatibel med DC strøm og spenning (750-494/000-005). Måling av spenning skjer ved å koble positiv og negativ fra 36VDC rekkeklemme, og rett inn i modulen. Siden batteriene ligger i parallell, trengs det bare én måling for spenning. Strøm måler vi med bruk av en shunt motstand. Det er en lav-ohmig resistans som tåler høye strømmer (0,2 Ohm). Den tåler 50A, og det vil gå et spenningsfall på 75mV over den ved maks belastning. Den kobles i serie med resten av lasten og er det siste som går inn til batteriet på minus-siden. På denne måten forsikrer vi oss om at shunten ikke blir bypasset av noe. Vi har en shunt for hvert batteri. Spenningsfallet over shuntmotstanden blir målt av PLS-modulen som igjen beregner hvor mye strøm som går gjennom motstanden. Strøm og spenning blir da brukt til å beregne momentan effektuttak og til å overvåke estimert gjenværende energi i batteriene.

3 Programmering

I dette kapittelet tar vi for oss oppbygging og virkemåte til programvarene for prosjektet. Vi går gjennom e!Cockpit som er programmet som brukes til å programmere Wago PLS. Dette består av både styring og overvåking. Videre går vi gjennom de forskjellige brukergrensesnittene i prosjektet som består av visualiseringen laget i e!Cockpit som kjøres på PLS, samt fjernstyring med PlayStation 4 Kontroller som er koblet til via Raspberry Pi. Til slutt blir kommunikasjonsgrensesnittet gjennomgått som bygger på NodeRed som kjøres på Raspberry Pi. Alle variabler og kommentarer er skrevet på engelsk. Dette for at flest mulig skal kunne lese og endre på koden (open-source tankegangen). Via OpcUa-protokollen er alle variabler er mulig å gjøre tilgjengelige (eller delvis allerede gjort tilgjengelige) mellom enhetene på plattformen og opp mot eventuelle nye enheter. På denne måten er plattformen klargjort for utvidelser og man kan bruke programmeringsspråket man er mest komfortabel med.

3.1 e!Cockpit

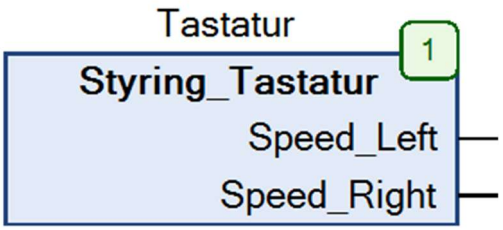
I prosjektet er programmet e!Cockpit blitt brukt for å styre og overvåke plattformen. e!Cockpit baserer seg på standarden IEC 61131-3, som omfatter 5 forskjellige programmeringsspråk. Det gjør at man kan velge det programmeringsspråket man føler seg mest komfortabel med. Programmeringen i e!Cockpit består av to programmer. Ett for styring og ett for overvåking. Ved å separere i to programmer kan man endre på oppdateringsfrekvens til hvert program individuelt, da overvåking trenger én oppdateringsfrekvens og styring én annen. Man kan da også jobbe på det ene programmet uten å forstyrre det andre programmet. Hovedprogrammet for styring og overvåking er laget i Continuous Flow Chart (CFC), mens alle funksjoner og funksjonsblokker er programmert i Strukturert Tekst. I hovedprogrammet settes alle funksjonsblokkene sammen til et fungerende system, som gjør at det blir ryddig, oversiktlig og modulært. Funksjonsblokker er laget for å dele opp de forskjellige oppgavene som blir gjort.



Figur 3.1: Oversiktsbilde over hovedprogrammet for styring i e!Cockpit

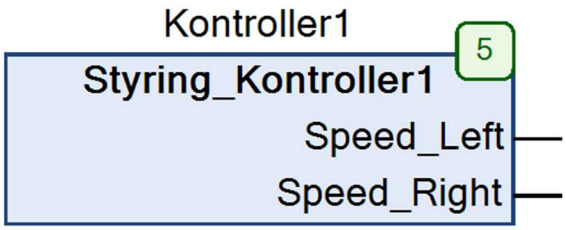
I figur 3.1 over ser man hvordan de forskjellige funksjonsblokkene er satt sammen. I korte trekk ser man at det er 3 forskjellige styringsmetoder som sender ut variabler til funksjonsblokken kalt Movement. Denne bearbeider inngangsverdiene fra styring-funksjonsblokkene samtidig som at den tar hensyn til brems. Deretter sendes verdiene videre til hver enkelt motor som omformer det til passende verdier som sendes til de respektive motorkontrollerene. Vi skal nå gå nærmere inn på funksjonene til de forskjellige funksjonsblokkene.

3.1.1 Styring



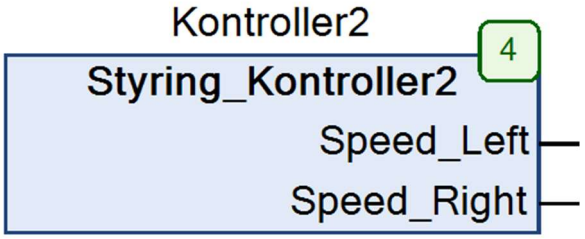
Figur 3.2: Tastatur-funksjonsblokk, mottar verdier fra visualiseringen og sender til Movementblokken

Vi har 3 forskjellige måter å styre plattformen på. Den første måten er ved hjelp av tastatur. Denne funksjonsblokken henter verdier på variabler som blir styrt fra e!Cockpits Visualiseringsvindu. Fart styres med et podmeter, og retning med 4 forskjellige knapper. Etter prosessering sender den ut en verdi for hjulene på venstre side og en for høyre side, kalt Speed_Left og Speed_Right. Dette er likt for alle styrings-FB'er.



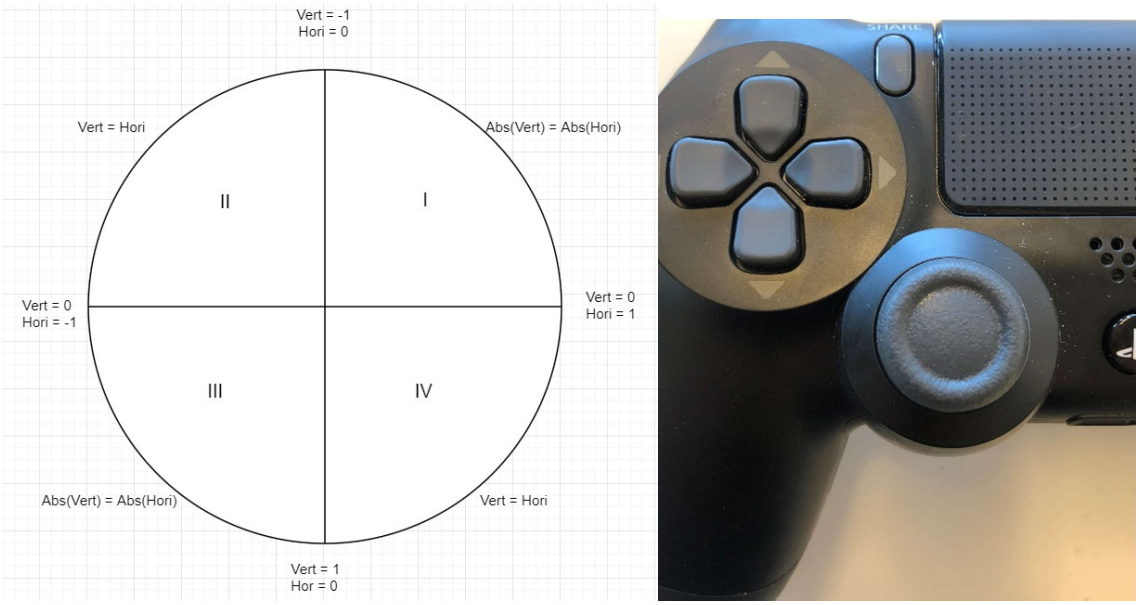
Figur 3.3: Kontroller1-funksjonsblokk. Mottar verdier fra NodeRed og sender til Movementblokken.

Med PlayStation 4 Kontroller kan den styres på 2 måter. Den ene måten er ved hjelp av 2 Joysticker der man styrer fart til venstre og høyre side individuelt med hver sin Joystick. Her justerer man fart med hvor langt joysticken presses oppover eller nedover. I denne blokken filtreres de forskjellige verdiene til joystickene for at den skal stå helt i ro når man Joysticken står i senter. Speed_Left og Speed_Right sendes ut fra blokken med en verdi mellom -1 og 1.



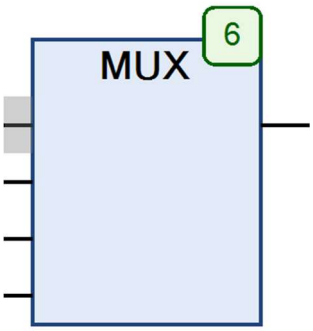
Figur 3.4: Kontroller2-funksjonsblokk. Mottar verdier fra NodeRed og sender til Movementblokken

Denne funksjonsblokken sørger for styringsmetode nummer 2 for PlayStation 4 kontrolleren. Med denne i bruk styres plattformen med én joystick. FB'en deler styringen først opp i 2 deler, styring av motorer på venstre og høyre side. Deretter deles joysticksområde opp i 4 kvadranter, som vist i figur 3.5. Horisontalt -1 på venstre side til 1 på høyre side. Vertikalt 1 på bunn til -1 på topp.



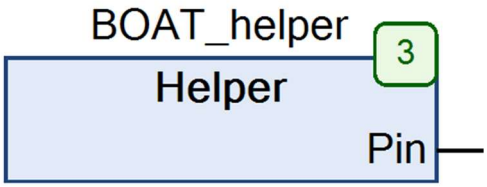
Figur 3.5: Joystickens kvadrantinndeling. Verdiene for Vert og Hori sendes fra NodeRed til PLS

Motorene på venstre og høyre side av plattformen blir tilegnet individuelle verdier mellom 0 og 1 ut i fra hvilken posisjon joysticken er i. Desto lengre ut mot sirkelperiferien man drar joytsticken, desto høyere pådrag gis til motorene. Ved joystick i første kvadrant går plattformen fremover, og svinger til høyre. Dette skjer ved at hjulene på venstre side går fortere enn hjulene på høyre side. Ved verdiene vert=0 og Hori=1, går hjulene på venstre side fremover, og høyre side bakover. Dermed spinner den til høyre rundt sin egen akse. Så lenge Hori=0 og Vert<0 går plattformen rett fram. Ved joystick i andre kvadrant går bort fremover, og svinger til venstre. Da går hjulene på høyre side fortere enn på venstre side. Ved verdiene vert=0 og Hori=-1, går hjulene på venstre side bakover, og høyre side framover. Dermed spinner den rundt sin egen akse mot klokken. I tredje kvadrant styres plattformen bakover og svinger til venstre. Ved Hori=0 og Vert>0 styres den rett bakover. I fjerde kvadrant styres den bakover og svinger til høyre.



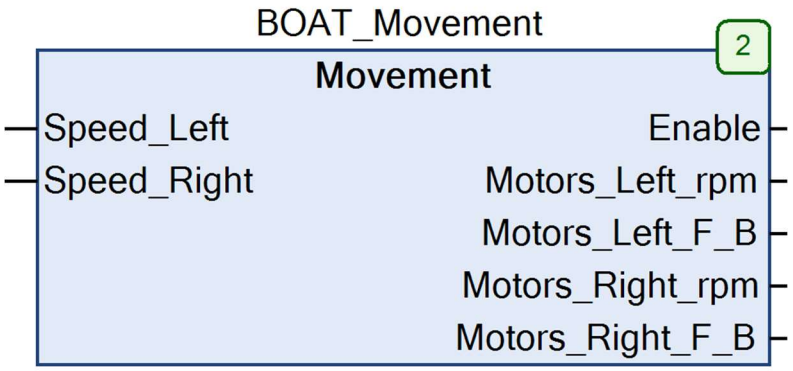
Figur 3.6: MUX-funksjonsblokk. Tilegner én av inngangene til utgangen.

Funksjonsblokken MUX får i programmet vårt inn 4 verdier. Den brukes for å velge en av mange innganger. Den første verdien er en styringsverdi som styrer hvilken verdi som skal bli valgt, ved å sende en heltallsverdi fra 0 og oppover. Sender den 0, velger MUX den første inngangen. Sender den 1, velger MUX den andre inngangen. På denne måten er det mulighet for å legge til flere styringsmetoder i ettertid ved utvidelser.



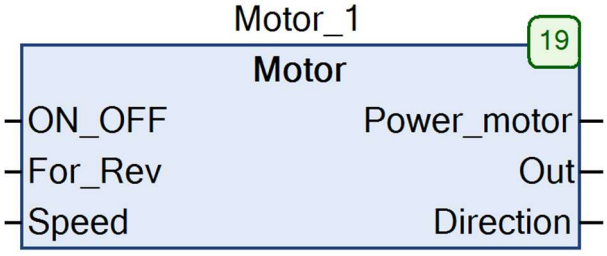
Figur 3.7: Helper-funksjonsblokk. Sender Pin-nummer ut i fra hvilken styringsmetode som er bestemt.

Denne sender styringsverdien til MUX-FB'en. Den mottar informasjon fra PlayStation 4 knappene og de interne knappene på visualiseringen, for å velge hvilken styringsmetode som skal bli brukt. Den sender ut en verdi på Pin-utgangen, som er 0, 1 eller 2. Gir den ut verdi 0 styres plattformen fra visualiseringen. Ved verdi 1 styres plattformen med to joysticker, og ved verdi 2 styres den med én joystick.



Figur 3.8: Movement-funksjonsblokk. Mottar verdier fra styringsblokkene gjennom MUX. Sender videre til motorblokkene.

Movement-blokken styrer hovedsakelig 3 ting. Den styrer om systemet er på eller av. Den styrer motorfart og rotasjonsretning. Den styrer også brems. Blokken mottar to hastighetsverdier fra MUX-blokken, Speed_Left og Speed_Right, som ligger mellom -1 og 1. Disse verdiene deles opp til én hastighetsverdi mellom 0 og 1 og én boolsk verdi som bestemmer rotasjonsretning for motoren. Verdiene sendes til de 4 funksjonsblokkene for motorene. Blokken mottar også informasjon fra PlayStation 4 knapper og interne visualiseringsknapper for styring av Power-variabelen som slår av og på systemet. Movement-FB'en styrer bremsing ved å kalle opp 2 funksjonsblokker for bremsing. Én for manuell bremsing og én for automatisk bremsing. Blokken er programmert slik at når bremsing er aktivert stopper den å gi pådrag til motorene, slik at motorene ikke blir ødelagt. Den automatiske brems-FB'en fungerer som en sikkerhetsfunksjon, som gjør at plattformen stopper før den kjører inn i noe. Funksjonsblokken mottar verdier fra avstandssensorene, og er programmert slik at den stopper å gi pådrag til motorene ved én avstand, og bremser ved én en avstand. Disse avstandene er justerbare. Den automatiske brems-funksjonsblokken kan slås av og på med i visualiseringen og med PlayStation 4 kontrolleren.

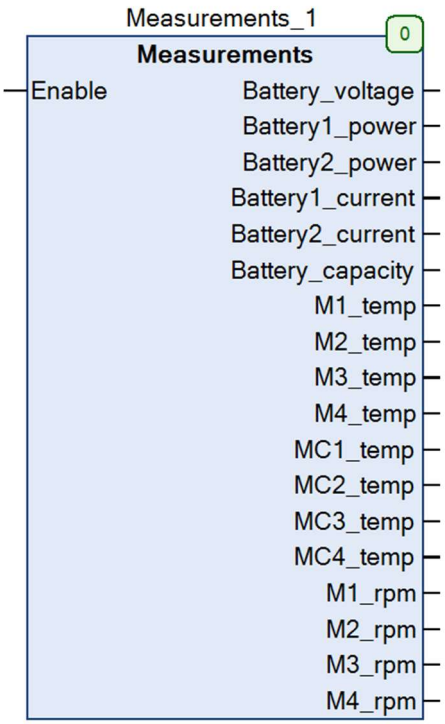


Figur 3.9: Motor-Funksjonsblokk. Mottar verdier fra Movementblokken, og sender til utgangene på PLS.

Denne blokken mottar 2 boolske verdier for av/på og rotasjonsretning fra Movement-blokken som sendes videre til utgangene Power_motor og Direction. Speed-verdien kommer inn med en verdi mellom 0 og 1. Denne verdien blir transformert opp til en verdi mellom 0 og 3276 som er maksverdi ved oppstart. Denne maksverdien kan endres til ønsket verdi enten i visualiseringen eller med PlayStation 4 kontroller. Verdiene sender til utgangene på PLS som derfra sendes til motorkontrollerens innganger.

Motorkontrolleren er tåler ikke mer enn 5V på pådrags-inngangen, mens Analog Ouput-modulen (750-559) gir ut 10V ved maks. Derfor er det lagt inn en Limit-funksjon som begrenser tillat makspådrag til motorkontrolleren til 5V.

3.1.2 Overvåking



Figur 3.10: Measurement-funksjonsblokk. Gir ut overvåkingsverdier viktige parametere.

For overvåking er det laget én funksjonsblokk som holder oversikt over alle variablene. Denne kalles opp i et program separert fra hovedprogrammet, for å kunne ha egen oppdateringsfrekvens for funksjonsblokken. På denne måten kan målingene bli oppdatert tregere som vil gi en mer nøyaktig måling.

For spenning, strøm og effekt, kaller den opp en ferdiglagd funksjonsblokk som kommuniserer med effektmålermodulen (750-494/000-005), og gir ut verdier for strøm, spenning og effekt. For utregning av kapasitet på batteriene er det blitt brukt en tabell for Li-ion batterier, som gir et godt estimat over kapasitet ut i fra spenningen på batteriene. Som vist i figur 3.11 er det 4,2 V per celle ved 100 %. Da batteriene våre har 10 celler, gir det oss 42V ved fulloppladet batteri.

Charge level (V/cell)	Discharge cycles	Available stored energy
[4.30]	[150–250]	[110–115%]
4.25	200–350	105–110%
4.20	**300–500**	**100%**
4.15	400–700	90–95%
4.10	600–1,000	85–90%
4.05	850–1,500	80–85%
4.00	1,200–2,000	70–75%
3.90	2,400–4,000	60–65%
3.80	See note	35–40%
3.70	See note	30% and less

Figur 3.11: Batterikapasitet som funksjon av spenning (Battery University 2018)

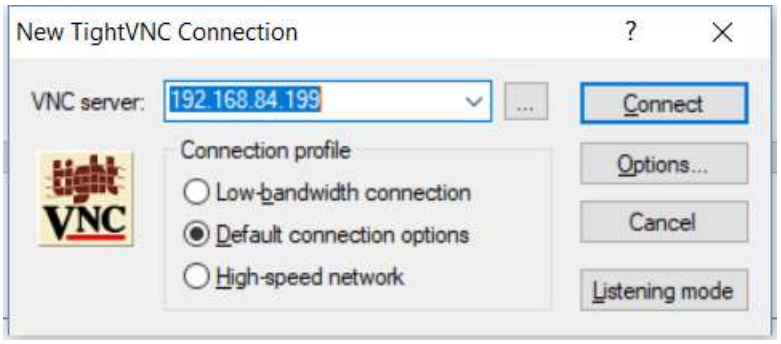
Måling av temperatur gjøres med PT-100 elementer. RTD-modulen (750-450) som sensorene kobles til, omgjør resistans-verdien i sensoren til en temperatur-verdi som hentes ut i programmet.

Måling av rotasjoner per minutt (RPM) gjøres i en funksjonsblokk som blir kalt opp av Measurement-blokken. Tellermodulene (750-404/000-005) teller pulser som kommer på hver av de 4 inngangene. Disse pulsene sammen med tiden til én oppdateringssyklus brukes til å finne ut hvor mange pulser som ble registrert den syklusen. Denne verdien er en frekvens som regnes om til en RPM. RPM brukes videre til å regne ut en fart i km/t også.

3.2 Kommunikasjonsgrensesnitt

For å knytte sammen de ulike delene i denne oppgaven har vi benytte oss av en Raspberry Pi 3. Denne fungerer da som server for NodeRed og kommuniserer med fjernkontrollen. Raspberry Pi er satt opp med standard Raspbian operativsystem hvor NodeRed er inkludert. Den har Bluetooth drivere installert, men det krever installasjon av ds4drv driver for å benytte PlayStation 4 kontrolleren med Raspberry Pi (Rosell 2017). Både Chromium, den medfølgende nettleseren til Raspbian, og Firefox er blitt testet opp mot webgrensesnittet som er laget for denne oppgaven. FireFox verson 52.7.3 gir tilgang til alle knapper og joysticker på PlayStation 4 kontrolleren, i motsettning til Chromium version 60.0.3112.89 som blander noen knapper med joysticker. Derfor er Firefox installert og benyttet når det er ønskelig å bruke kontrolleren direkte tilknyttet Raspberry pi.

For å kunne enklere konfigurere og styre Raspberry Pi benyttes eksternt skrivebord, på den måten kan Firefox åpnes for å klargjøre bruk av PlayStation 4 kontrolleren, fra en pc tilkoblet samme nettverk. Utfordringen med eksternt skrivebord og Linux er at i utgangspunktet så vil enhver tilkobling åpne en selvstendig økt. Det vil si at hvis det er tilkoblet en skjerm, mus og tastatur og dette benyttes til å åpne et program som Firefox, så vil ikke en tilkobling gjennom eksternt skrivebord kunne se og videre konfigurerer den samme nettleseren. Utfordringen er da at potensielt åpnes flere nettlesere, som er resurskrevende, hvis det forsøkes å endre noe fra forskjellige maskiner på nettverket. Det skaper også potensielle konflikter hvis flere økter prøver å koble seg opp mot NodeRed og sender data om kontrolleren. Derfor er det ønskelig at uansett klient som kobler seg til, vil den koble seg til samme økt og dermed konfigurere samme program. Dette er blitt gjort ved å installere en Virtual Network Computing server som kun setter opp en økt. X11VNC er en type VNC server som gjør dette, den har blitt installert og benyttes i denne oppgaven (Karlrunge 2002). For å kunne koble seg til med eksternt skrivebord krever det en klient som støtter denne serveren.



Figur 3.12: TightVNC Viewer. IP adressen til Raspberry pi skrives inn under VNC server.

Vi benytter TightVNC Viewer til Windows for å koble til Raspberry Pi, maskinen som brukes må være tilknyttet samme nettverk (TightVNC 2018). IP adressen til Raspberry Pi skrives inn, deretter trykkes det på Connect og et passord må skrives inn, «BOAT» i dette tilfellet.

3.2.1 NodeRed

NodeRed er en form for visuell programmering som er blitt benyttet i denne oppgaven til å knytte sammen de ulike komponentene vi har, til ett felles system. NodeRed er laget for å kunne enkelt «koble» sammen komponenter av ulikt fabrikat som kommuniserer med ulike protokoller. Det gjøres ved å benytte noder som er scriptet i JAVA, disse finnes tilgjengelig gjennom biblioteker/paletter til NodeRed og er Open-Source. Dette er noe vi fra før hadde kjennskap til gjennom Automatiseringsfaget, hvor vi benyttet det til å knytte et web-basert grensesnitt opp mot en WAGO-PLS.

Det ble da et naturlig valg å benytte dette i denne bacheloren når vi hadde valgt en WAGO-PLS som utgangspunkt og vi i tillegg hadde ønske om å benytte oss av sensorer fra ulike leverandører som var støttet av Raspberry pi.

Tabell 3.1: Oversikt over benyttede Noder i NodeRed

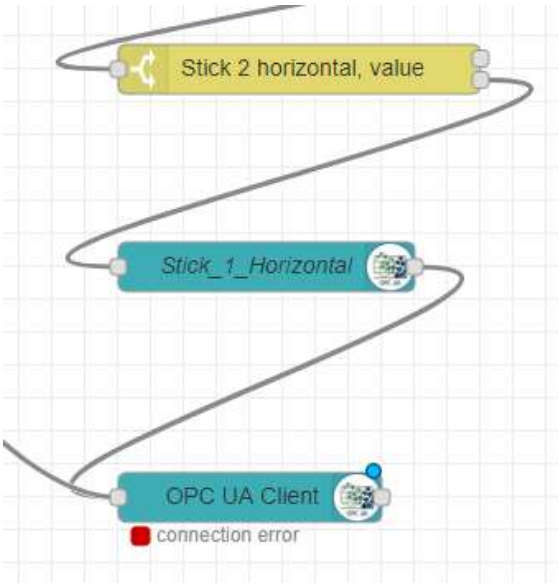
Type	Bilde	Beskrivelse	Palette	Versjon

Switch		separerer data fra websocket	node-red	0.18.2
Spilt		Skiller ut tall-verdi fra web-socket data	node-red	0.18.2
Trigger		Omgjør signal til en puls	node-red	0.18.2
OpcUa-Item		Inneholder re-feranse til vari-abler i PLS programmet	node-red-con-trib-opcua	0.2.22
OpcUa-Client		Kobler sam-men NodeRed og PLS	node-red-con-trib-opcua	0.2.22
Websocket in		Setter opp en websocket for mottak av data	node-red	0.18.2
http in		Setter opp stien for websiden	node-red	0.18.2
http respons		Sender websi-den på fore-spørsel fra kli-ent	node-red	0.18.2
template		Setter opp websiden med HTML kode.	node-red	0.18.2
rpi-srf		Trigger og le-ser av ultralyd-sensorer	node-red-node-pisrf	0.1.0

Rpi-gpio out		Styrer GPIO pinne på Raspberry Pi	node-red	0.18.2
Inject		Trigger andre noder ved gitt interval	node-red	0.18.2

I denne oppgaven er NodeRed i hovedsak benyttet som bindeledd mellom sensorer, fjernstyring og PLS. Fysisk er alle disse komponentene koblet til Raspberry Pi, vi kjører derfor også NodeRed serveren på Raspberry Pi. Man kunne også ha valgt å separere sensorene, fjernstyringen og NodeRed i hver sin Raspberry Pi. Det kan vurderes om ved eventuelle fremtidige utvidelser kreves mer enn én Raspberry Pi kan levere. Så langt er det mest kompakt å ha alt i én enhet.

Kommunikasjonen mellom PLS og NodeRed går via OPCUA protokollen, denne har egne noder for å kunne knytte sammen variabler i NodeRed med variabler i PLS programmet. En node konfigureres med adressen til PLS og definerer om det skal skrives til eller leses fra programmet, denne kobler seg til når programmet starter og gir en feilmelding hvis den ikke finner PLS. Den andre noden konfigureres med hvilken variabel i PLS programmet som skal brukes, og hvilken type variabel det er.



Figur 3.13: Bruk av OPC-UA til kommunikasjon med PLS. Den øverste noden inneholder verdien, den midterste er variabelen det skal skrives til, og den nederste er koblingen til PLS.

Ultralydsensorene styres av et trigger signal og sender tilbake et ekko, dette må styres av Raspberry Pi i rett sekvens for å unngå at det sendes et signal samtidig som det kommer et ekko tilbake. Her fant vi en node i biblioteket som er laget for de ultralydsensorene vi benytter, som kobler seg til valgfrie GPIO pinner. Denne noden sender trigger signalet og venter på ekko signalet, dermed sendes det ikke en ny trigger før ekko signalet er tilbake. Det gjør konfigurasjonen enkel, det vi må stille inn er ønsket oppdateringsfrekvens og hvilke pinner vi har koblet sensoren til. Det vi får ut fra noden er da en Real variabel med en tallverdi som representerer avstand i cm. Denne sender vi da til PLS'en via OPCUA Nodene.

Figur 3.14: Ultralydsensor i NodeRed. Venstre node styrer en ultralydsensor og sender avstand i cm til OPCUA nodene som sender verdien til PLS programmet.

Bremseservoen styres av et pulsbreddemodulert signal fra en GPIO pinne på Raspberry Pi, denne pinnen styrers via en node i NodeRed. Her fant vi en node som er laget for å generere pulsbredde-signaler på en valgfri GPIO pinne. Der legger vi inn hvilken pinne servoen styres fra og ønsket frekvens på signalet. Noden forventer en Real variabel med verdi mellom 10 og 20 som korresponderer med posisjonen til servoen mellom 0 og 180

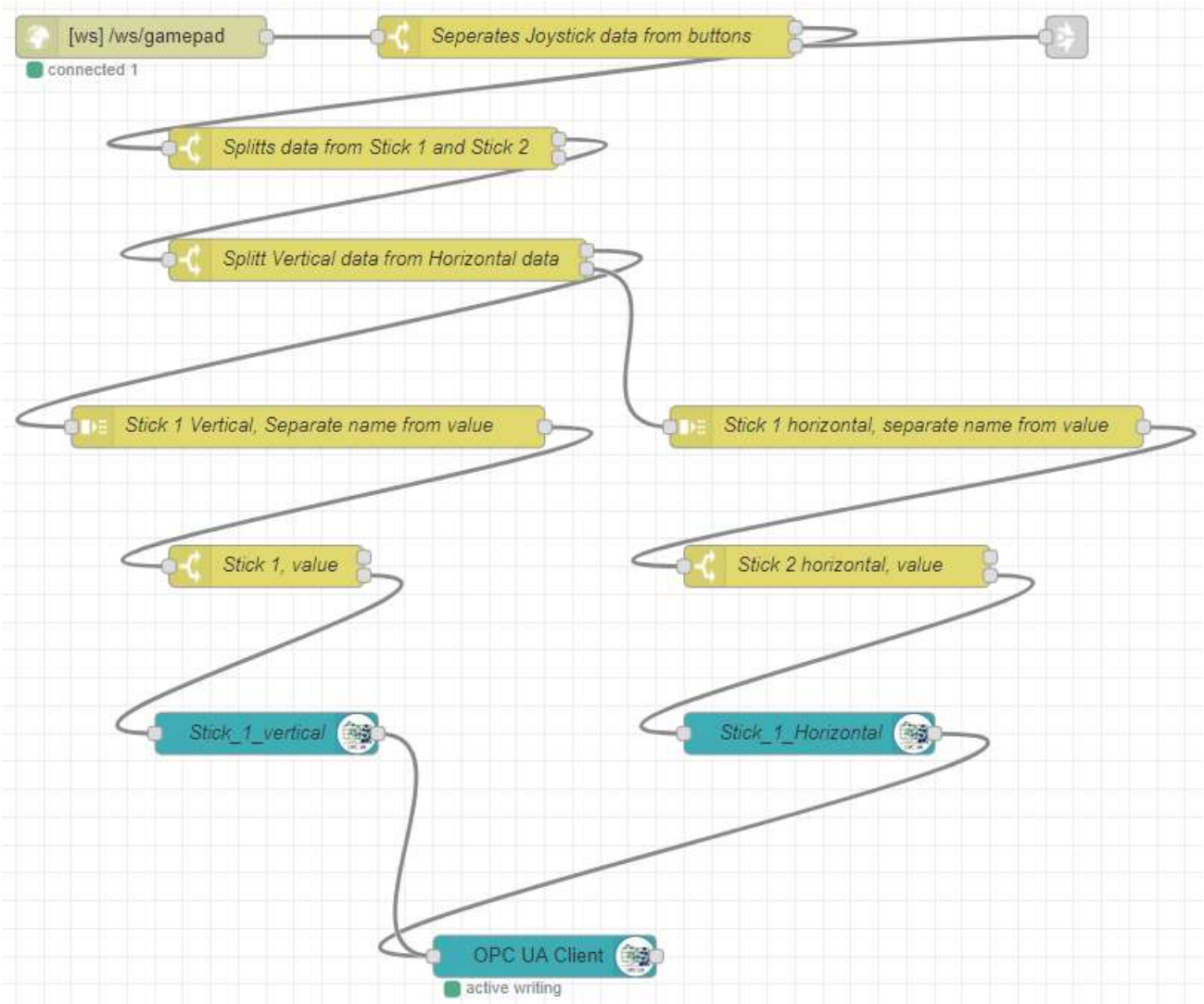
grader. Denne verdien blir gitt fra PLS programmet via OPCUA nodene som henter verdien fra riktig variabel.

NodeRed blir benyttet som webserver for scriptet som håndterer kommunikasjonen med fjernkontrollen. Dette gjøres ved å bruke en «http in» node som setter opp en filbane under NodeRed serveren, «127.0.0.1:1880/gamepad» er den lokale adressen. Tilknyttet denne noden er det en «template» node som HTML skriptet legges inn i. Dette kobles videre til en «http respons» node, denne sørger for at en forespørsel på den gitte adressen blir knyttet opp mot HTML skriptet i «template» noden.

Figur 3.15: Oppsett av webserver i NodeRed. Fra venstre er http in noden koblet til template noden som igjen er koblet til http respons noden.

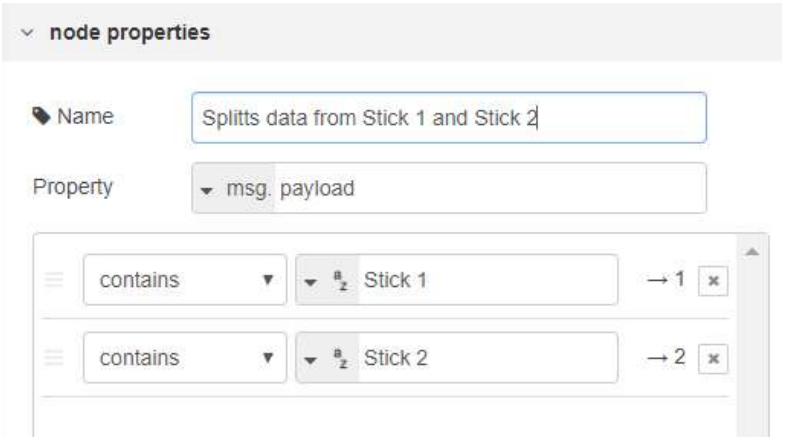
Websocket blir benyttet for å kunne hente data tilbake fra HTML siden som kommuniserer med fjernkontrollen. I node red er det satt opp en Websocket node, denne setter opp en egen adresse under NodeRed serveren, som det sendes data til. Samtidig videresender denne noden all data til den noden som kobles til.

Figur 3.16: Websocket noden setter opp en adresse og viderefører data sendt til denne adressen.



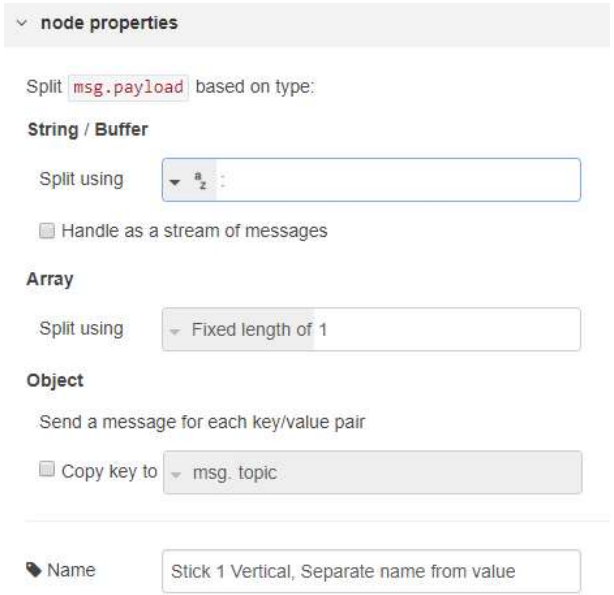
Figur 3.17: Datastrøm fra Websocket node oppe til venstre separeres gjennom switch og split noder. Ferdig separert verdi sendes til PC UA node.

Data som sendes til Websocket blir behandlet som meldinger med data referert til som msg.payload, det er behov for å skille ut den relevante informasjonen fra meldingene. Dette gjøres via en rekke switch og splitt noder. Switch noden kan konfigureres til å separere innholdet i msg.payload basert på valgfrie kriterier.



Figur 3.18: Konfigurasjon av en Switch Node. Msg.Payload, som inneholder verdien "Stick 1", sendes videre til utgang 1 av noden og tilsvarende for 2.

I tilfellet vist i figur 3.18 er det ønskelig med informasjon fra joystick en og to. msg.Payload inneholder verdier og navn fra joysticker og knapper, det sendes da først gjennom en switch node som separerer ut alle meldinger som inneholder ordet Stick. Mer informasjon om hvordan datastrømmen er bygget opp er beskrevet under HTML5. Som vist i figur 3.19 separeres disse meldingene basert på nummer og sendes videre på gitte utganger. Meldinger fra joystick 1 sendes videre til en swich node som skiller meldinger som inneholder horisontale verdier fra meldinger med vertikale verdier. Meldingen som gjenstår inneholder da navnet på den aktuelle joysticken med tilhørende verdi som en streng, dette sendes til en splitt node, som deler strengen i navn og verdi.



Figur 3.19: Konfigurasjon av en splitt node. Msg.payload inneholder navn og verdi skilt med kolon fra en joystick, den splittes etter kolon.

Meldingen kan da sendes til en siste switch node som skiller ut meldinger med tekst, da er det kun meldinger med tallverdi som sendes videre til PLS programmet via OPC UA.

3.2.2 Webgrensesnitt, bruk av Websocket og Gamepad

Vi bestemte oss tidlig for å benytte en fjernkontroll som er lett gjenkjennelig og intuitiv for de fleste, derfor ønsket vi en kontroll fra en spillkonsoll som PlayStation eller Xbox. I starten hadde vi en PlayStation 3 kontroller tilgjengelig, denne benytter seg av Bluetooth for trådløs kommunikasjon som også støttes av Raspberry Pi. Vi fikk også tak i noen forskjellige Xbox kontrollere for å kunne sammenligne hva som ville egne seg best. Utfordring ble å kunne lese av dataene fra kontrollen og implementere dette i PLS programmet, spesielt med tanke på at det er små forskjeller mellom kontrollene, som antall knapper og joysticker. Vi visste også at om vi bestemte oss for en type kontroller så ville det bli den nyeste, PlayStation 4 kontroller i dette tilfellet, og vi var derfor ikke sikre på om en eventuell kode vi laget for PlayStation 3 kontrollen ville fungere med den nyeste. Det ble brukt mye tid på å undersøke forskjellige løsninger for å kunne hente ut nødvendig data, de få eksemplene vi fant hvor PLS var benyttet med en PlayStation kontroller var det brukt kablet forbindelse. Etter hvert fant vi noen eksempler på hvordan forskjellige kontrollere kan benyttes i enkle nettbaserte spill, dette kan gjøres via HTML5 i nettlesere som støtter dette.

Figur 3.20: Webgrensesnitt. Informerer om at Websocket er tilkoblet og at kontrolleren er koblet til, samt viser verdiene for de ulike knappene/stickene.

HTML5 støtter JavaScript, som har muliggjort utviklingen av Gamepad (Mozilla 2018). Dette er et script som kan identifisere de fleste typer spillkontrollere og dermed hente ut informasjon fra dem. Den største fordelen med dette skriptet for oss er at vi kan utvikle fjernstyringen vår basert på dette, uten at vi begrenser oss til en spesifikk kontroller. Dermed står de som eventuelt ønsker å videreutvikle plattformen fritt til å velge sin egen favorittkontroller.

Vi fant eksempelkode på hvordan å identifisere en kontroller og hente ut status fra knapper og joysticker, denne har vi bygget videre på(Camden 2014). . Scriptet setter opp en enkel nettside som, når åpnes av en klient, sjekker om det er en kontroller tilkoblet klienten. Hvis det er det, i dette tilfellet en PlayStation 4 kontroller, skrives verdiene fra alle knapper og joysticker ut i vinduet. Den samme informasjonen sendes også til NodeRed via kommunikasjonsprotokollen Websocket, men i et annet format.

```
for(var i=0;i<gp.buttons.length;i++) {
    html+= "Button "+(i+1)+": ";
    if(gp.buttons[i].pressed) html+= " pressed";
    ws.send(i+" "+gp.buttons[i].value);
    html+= "<br/>";
}
```

Figur 3.21: For-sløyfe som skriver ut antall knapper som er tilgjengelig og sender verdien til knappene over websocket.

For å kunne håndtere informasjonen i NodeRed har vi laget et eget format på data som sendes til websocket noden. Som vist i figuren over sendes verdien til hver knapp sammen med nummer på knappen til websocket noden. Formatet blir da «nummer verdi».

```
for(var i=0;i<gp.axes.length; i+=2) {
    html+= "Stick "+(Math.ceil(i/2)+1)+": "+gp.axes[i]+","+gp.axes[i+1]+"<br/>";
    ws.send("Stick "+(Math.ceil(i/2)+1)+" "+"Horizontal "+":"+gp.axes[i]);
    ws.send("Stick "+(Math.ceil(i/2)+1)+" "+"Vertical "+":"+gp.axes[i+1]);
}
```

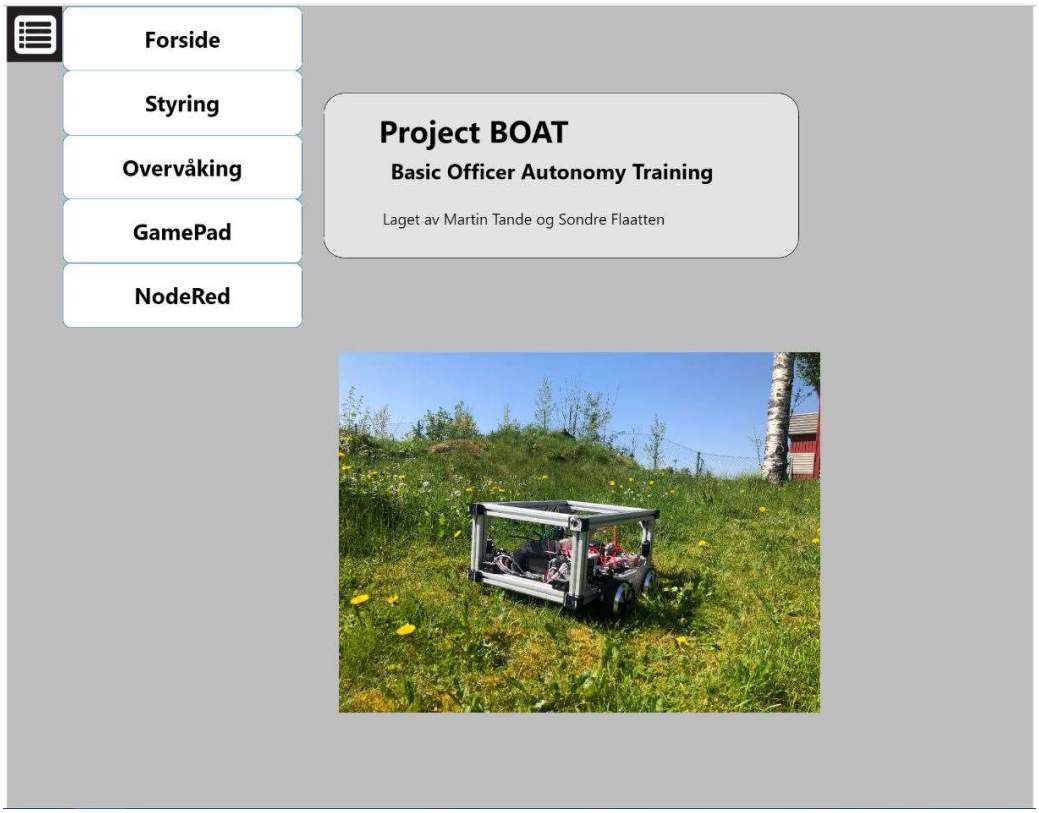
Figur 3.22: For-sløyfe som skriver ut antall joysticker og verdiene for horisontale og vertikale akser.

For-sløyfen i figuren over sjekker antall akser som er tilgjengelig, en joystick består av to akser, og skriver ut nummeret på joysticken sammen med verdien for horisontal og vertikal aksen. Data som sendes til websocket noden er delt opp i horisontale og vertikale verdier. Meldingen består da av formatet «Stick nummer Horizontal :verdi» og tilsvarende for vertikal. Dermed kan meldingene skilles på ordene de inneholder, som beskrevet tidligere.

3.3 Brukergrensesnitt

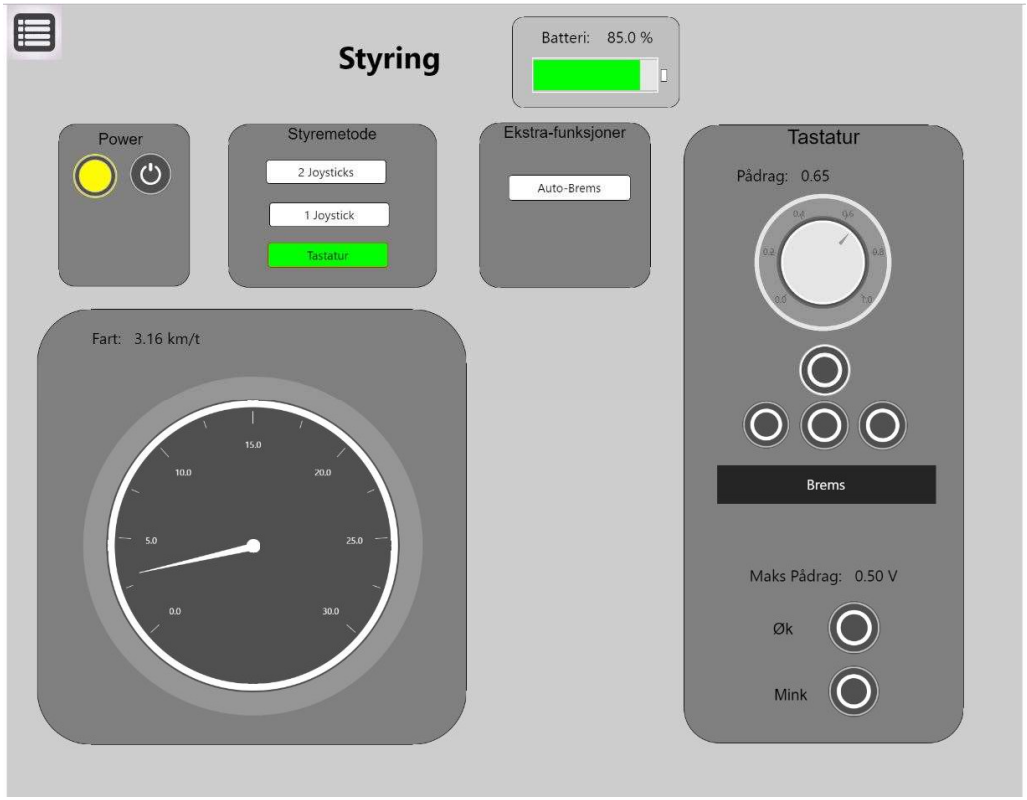
Brukergrensesnittet for plattformen består av en visualisering fra e!Cockpit og en PlayStation 4 kontroller. Det er laget en Human Machine Interface ved hjelp av visualiseringsfunksjonen i e!Cockpit for å ha mulighet til å styre og overvåke viktige parametere for plattformen. Denne er delt opp i flere sider for å gi oversikt.

Visualiseringen starter med en forside som vist i figur 3.23.



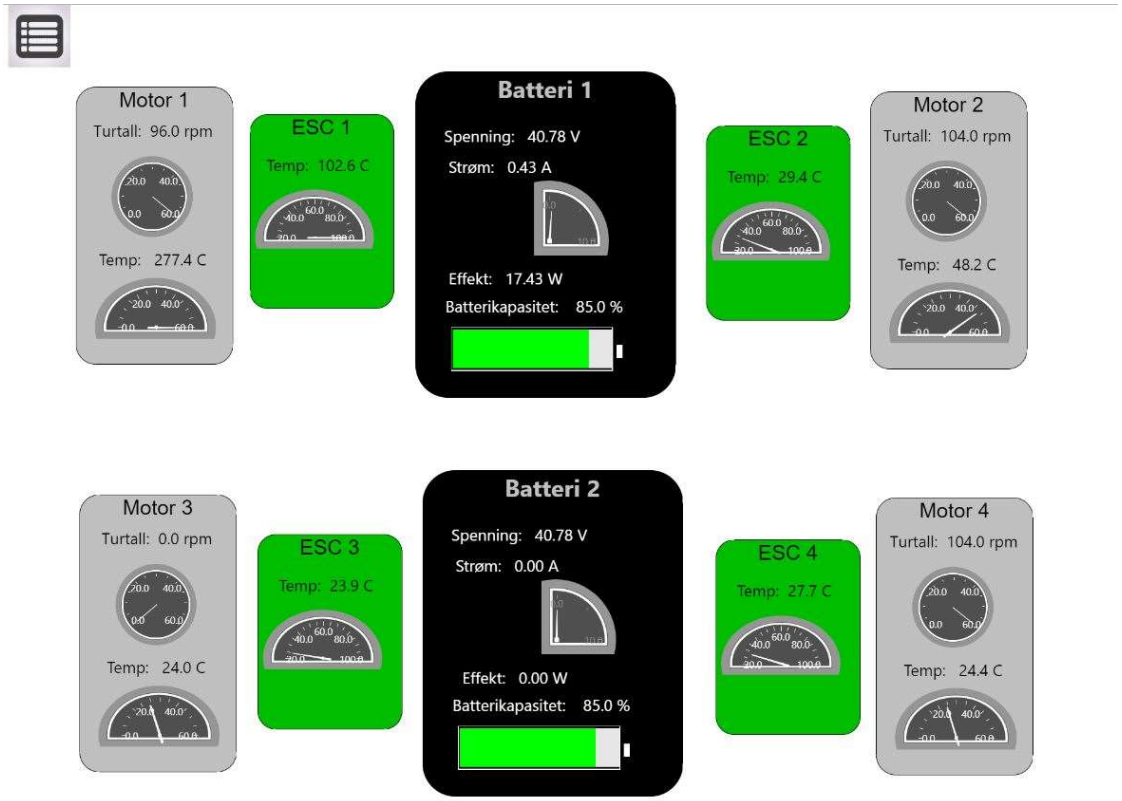
Figur 3.23: Forside av visualiseringssiden laget i e!Cockpit. Meny oppe i venstre hjørne.

Oppe i venstre hjørne er det en menyliste. Fra menylisten kan man videreføres til visualisering av styring via HMI, overvåking av viktige komponenter på plattformen. Den viderefører også til nettsidene for Test av GamePad og NodeRed programvaren ved behov.



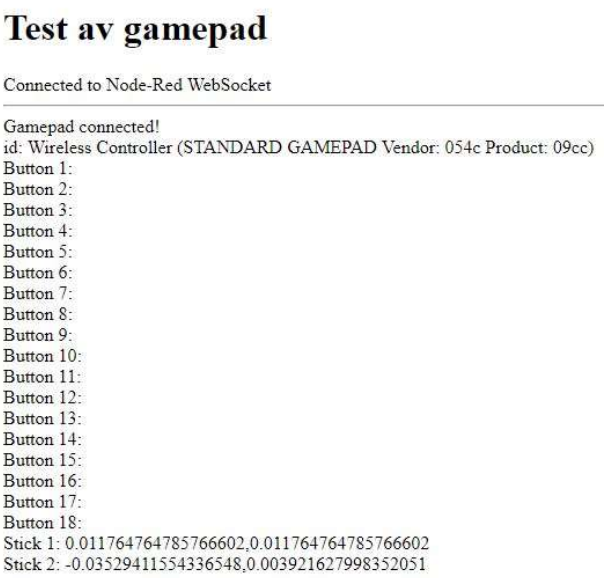
Figur 3.24: Visualisering for «Tastaur»-styring via HMI

Figur 3.24 viser hvordan styring fra HMI, også kalt «Tastatur»-styring, er satt opp. I visualisering for styring ligger den interne knappen for Power, som må slåes på for å kunne styre og overvåke plattformen. I visualiseringen kan det også velges mellom de tre styringsmetodene som laget for plattformen, 2 Joysticks, 1 Joystick og Tastatur. Ekstrafunksjonen Auto-brems kan skrues av og på, og er en sikkerhetsfunksjon som sørger for at den bremser når den holder på å kjøre i objekter, som en vegg. Ved valg av styringsmetode «Tastatur» brukes den borterste boksen. Potensiometeret brukes for å endre prosentandel pådrag mellom 0 og bestemt maks pådrag. Knappene for fremover og bakover vil kjøre den rett fram eller rett bak. Knappene for sideveis, gjør at den roterer rundt sin egen akse, enten med klokken eller mot klokken. Knappene for bevegelsesstyring samt brems er tappere som gjør det lettere å styre fra en enhet med Touch Screen. Nederst kan man endre maks pådrag som vil enten øke eller minke skalaen for Pådrag. Alle funksjoner og knapper kan betjenes fra den andre delen av brukergrensesnittet, som er PlayStation 4 kontrolleren. For oversikt over hvilke knapper som gjør hva, se Vedlegg B.2. Ved å ha flere styringsmetoder gir det brukeren mulighet til å endre til metoden som er mest komfortabel for brukeren.

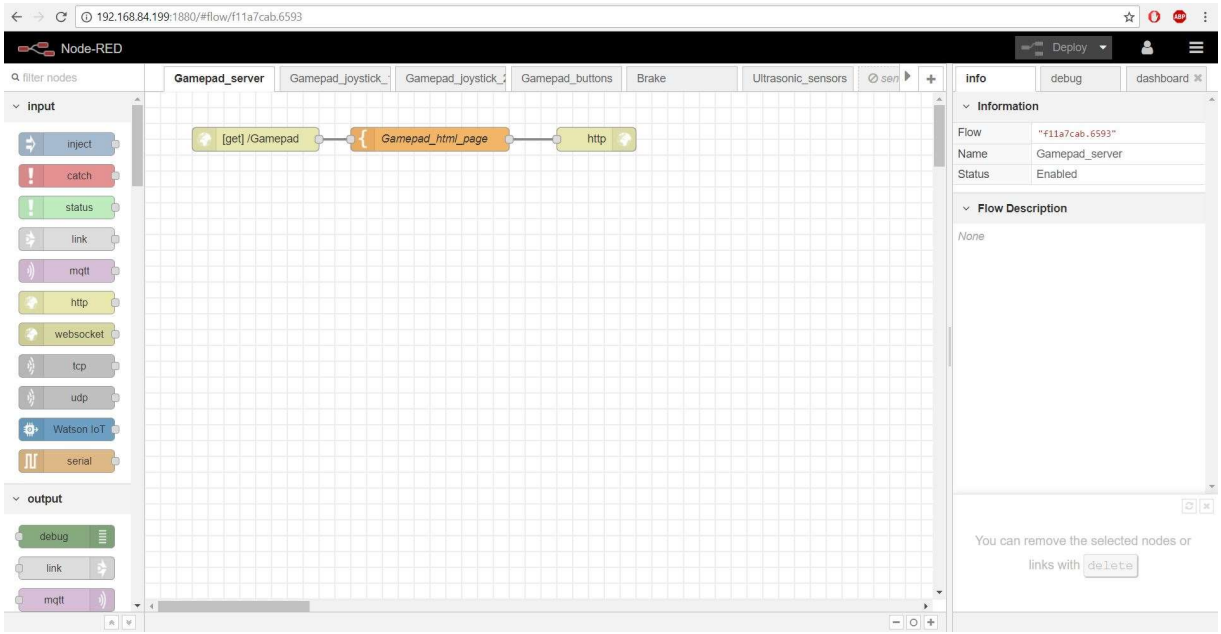


Figur 3.25: Visualiseringsvindu for overvåking av plattformen. Overvåker motorene, motorkontrollerene og batteriene.

I visualiseringsvinduet for overvåking vises status på temperatur og turtall for alle 4 motorene, og temperatur for motorkontrollerene. På batteriene overvåkes spenning, strømtrekk, effektuttak og kapasitet.



Figur 3.26: GamePad-nettside, som gir tilkoblingsstatus til PlayStation 4 kontroller.



Figur 3.27: NodeRed-nettside. Den er delt opp i flere faner for å gi oversikt. Her er det mulig å feilsøke og legge til komponenter.

Ved å ha lett tilgang til GamePad-nettsiden kan man enkelt verifisere at verdiene fra PlayStation 4 blir registrert av Raspberry Pi. Tilgang til NodeRed-nettsiden gir brukeren oversikt over programvaren brukt i NodeRed, som kan være nyttig ved feilsøking og integrering av komponenter.

Nyttige utvidelser i forbindelse med brukergrensesnitt kan være å integrere Raspberry Pi Touch Skjerm og kamera for videooverføring til plattformen. Dette vil forbedre brukervennligheten og kan enkelt kobles opp til en Raspberry Pi og integreres som fast brukergrensesnitt til plattformen.

4 Drøfting

Vi har konstruert en plattform som både skal fungere i undervisningsøyemed og for testing av sensorer og beslutningssystemer. Den består av en konstruksjon laget av aluminiumsprofiler, med en fremdriftslinje som består av børsteløse DC-motorer, motorkontrollere og Lithium-ion batterier. PLS og Raspberry Pi er brukt som styrings- og kommunikasjonsenheter for plattformen.

Modularitet er svært viktig for denne oppgaven, da det er dette som vil gi nødvendig rom for utvidelser og oppgraderinger. Det har vi oppnådd ved de løsningene som er valgt innenfor både konstruksjonen og programmeringen. Rammen er av en god størrelse og styrke som gir rom for utvidelser. Aluminiumprofilene med spor sammen med bruk av DIN-skinner gjør det lett å skru fast nye og flytte på eksisterende komponenter. Dermed ivaretar den mekaniske konstruksjonen målene vårt om modularitet og belastningsstyrke.

Plattformen benytter seg av batteridrevet fremdriftslinje, med børsteløse likestrømsmotorer. Per dags dato er motor og hjul fastmontert. Dette var en enkel løsning mekanisk sett, men gjør det ikke mulig å svinge med hjulene. Ved behov for bedre manøvreringsevne (sideveis kjøring) anbefales det å gi mulighet til at hjulene kan svinge individuelt.

Opprinnelig ville vi prøve å ta i bruk og omprogrammere Motherboardet som var brukt i Hoverboard til å styre motorene, altså å gjenbruke de originale motorkontrollerne. Da dokumentasjon på Motherboardet ikke medfulgte og det ikke var tilrettelagt for ekstern kontroll viste det seg å være tidkrevende og vanskelig å modifisere dette til vårt formål. Vi valgte derfor å anskaffe nye motorkontrollere, og vi endte med 500W BLDC motorkontroller med Hall effekt sensor-tilbakemelding. Disse er mer brukt, vi fant eksempler på bruk med samme motor som plattformen benytter, og dermed bedre dokumentert. Dokumentasjonen skulle også være nyttig for å videre kunne bygge på disse. Fordelen med disse motorkontrollerene er at de er billige og lett å implementere. De største ulempene er at de ikke støtter elektriske bremsing og endring av akselerasjonskurven. For mulighet til å endre akselerasjon og elektrisk bremsing har det blitt sett på andre motorkontrollere. Hvorav Kelly controller KBS36051X virker å være et godt alternativ, men grunnet tidskravet hadde vi ikke mulighet til å bestille og teste denne (Kelly 2018).

Det er blitt greid ut om og vurdert en rekke elektriske bremsemetoder for plattformen. Dette hadde vært lurt hvis det hadde vært mulighet til å ta vare på energien som oppstår og kunne spart oss for ekstra komponenter. Da motorkontrollerene våre ikke støttet dette

måtte det blitt laget egne bremsekretser som ville krevd ekstra kobling og komponenter. Istedenfor valgte vi en mekanisk bremseløsning, som består av skivebrems, kaliper og wire. En servomotor som er koblet opp mot Raspberry Pi styrer wirene. Med denne metoden får plattformen en kraftig brems som også muliggjør at den kan stå i ro på skråplan.

Som hovedplattform for styring og overvåking er det blitt brukt Wago PLS. Da dette er en del av fagplanen på Sjøkrigsskolen vil det gjøre det enklere for lærere å integrere i undervisningen og for kadetter i ettertid å videreutvikle plattformen. Wago PLS er et modulbasert system som gjør at man kan erstatte og legge til nye moduler etter behov. Bruk av PLS gir plattformen en sikkerhet, i og med at hver modul er sikret mot overbelastning og kortslutning. Hvis det skulle oppstå feil i en modul så er det lett å bytte ut. Raspberry Pi er blitt brukt som kommunikasjonsledd mellom de forskjellige delene i plattformen. Det er viktig for Forsvaret å kunne implementere ting fort. Raspberry Pi muliggjør dette da det er et billig programmeringsverktøy, og på grunn av den store brukermassen finnes det mye Open-Source dokumentasjon til mye forskjellig.

Nettverkskommunikasjonen er ivaretatt av en ruter, denne gjør i dag at både Raspberry Pi og PLS er mulig å nå via trådløs tilkobling. Alternativet var å koble Raspberry Pi og PLS direkte sammen via Ethernet, dette er blitt forsøkt og fungerer, med ulempen med dette er at andre enheter da ikke vil ha tilgang. Derfor ble det vurdert at det enkleste er å legge inn en dedikert ruter. Dette sørger også for ekstra tilgjengelige Ethernet porter om det skulle bli nødvendig med mer nettverksbasert utstyr. Ruteren er nå satt opp med sitt eget trådløse nettverk. Dette har vært en fordel når vi har testet plattformen forskjellige steder samtidig som vi ønsker tilgang på HMI via en PC. Ruteren kan også konfigureres til å koble seg opp mot et annet eksisterende trådløst nettverk, eksempelvis Sjøkrigsskolen sitt nettverk. Dette vil muliggjøre kommunikasjon med plattformen uavhengig av din og dens posisjon i bygget, fjernstyring er også mulig via nettverket om ønskelig.

Et annet punkt som er verdt å snakke om er måten fjernkontrollene ble tilkoblet på. Det ble vurdert og forsøkt å koble fjernkontrollen direkte til PLS, da det ville vært en mer robust løsning som også tilbyr lengre rekkevidde med ekstern antenne. Dette kunne også minsket konfigurering av fjernkontrollen ved å kutte Raspberry Pi som mellomledd. Det ble bestilt en Bluetooth modul til WAGO PLS og denne ble forsøkt koblet til PlayStation 4 Kontrolleren. Vi fant da ut at selv om både modulen og kontrolleren benytter samme Bluetooth standard, så har kontrolleren en spesifikk parrings sekvens som ikke støttes av modulen. Det var derfor ikke mulig å benytte Bluetooth for å koble kontrolleren direkte

til PLS uten å eventuelt endre koden til selve modulen. Dette er ikke Open Source så vi konkluderte med at det ikke var mulig innenfor denne oppgaven. Eventuelt kan man vurdere på et senere tidspunkt å ta det opp igjen om det blir utgitt nye versjoner av Bluetooth moduler for Wago PLS. Fordelen ville vært at Raspberry Pi ble avlastet, og man ville fått en mer robust enhet som bare hadde som oppgave å holde kommunikasjonen med fjernkontrollen. Alternativt kunne denne kommunikasjonen vært gjort på en separat Raspberry Pi eller en annen enhet hvis det anses som nødvendig å avlaste eksisterende Raspberry Pi, men ved å bruke den som allerede eksisterte ble det en mer kompakt og enkel løsning.

For å ivareta modularitet innenfor elektriske tilkoblinger har vi valgt å legge inn rekkeklemmer mellom alle komponenter. Dette gjør det enklere å måle signaler og effekter inn til enkeltkomponenter, som bidrar til enklere feilsøking om nødvendig. Det gjør det også enklere å koble fra og bytte ut enkeltkomponenter eller legge til nye komponenter imellom. Samtidig med dette er det blitt brukt standardiserte signaltyper, både på inn- og utganger av de forskjellige komponentene. Dette gjør det enklere å legge til eller bytte ut komponenter.

Diverse sensorer er brukt til overvåking av plattformen. Det er også blitt implementert ultralydsensorer for avstandsmåling som er blitt brukt til programmering av enkle sikkerhetsfunksjoner. Disse er også tiltenkt å brukes for videreutvikling av autonomi og beslutningstaking, da posisjonene på sensorene kan endres. Sensorene sender pulser med samme frekvens. Grunnet at vi har montert disse med overlapp gjør at de registrerer pulser som hverandre, ikke var sendt ut av dem. Dette har gjort at vi ikke har kunnet benyttet alle sensorene til det fulle. Ved å endre på posisjoneringen eller omprogrammere sensorene vil sensorene kunne tas i bruk for fult. For større prosjekter som 3D-mapping og beslutningssystemer anbefales det å implementere LIDAR.

All koding i PLS og NodeRed er programmert i blokker. Blokkbasert koding gjør det oversiktlig og lett å forbedre deler av koden eller implementere nye blokker som skal ha nye funksjoner. Alle variabler er mulig å gjøre tilgjengelige for alle enheter, som gjør at man kan legge til en ny enhet og programmere all kode på den med et programmeringsspråk man er mer kjent med. Dette muliggjør lettere integrering av kode da det ikke er nødvendig å lære seg et nytt programmeringsspråk.

5 Konklusjon med anbefaling

Denne oppgaven har tatt for seg oppbygningen av en testplattform for autonome systemer, hvor fokus har vært på modularitet for at den skal kunne enkelt utvides ved test av nye sensorer og beslutningssystemer.

Plattformen kan benyttes til videre undervisning og fremtidige bacheloroppgaver. Med tanke på den modulære oppbygningen, styrken til rammen og kraften i motorene er plattformen klargjort for utvidelser. Dette kommer til å være essensielt for å kunne teste ut nye sensorer, styrings- og beslutningssystemer i fremtiden, da det kan påmonteres uten å kreve redesign av eksisterende plattform. En plattform som har muligheten til å bevege seg basert på innputt fra sensorer og styringssystemer som testes vil være avgjørende for å kunne vurdere verdien av sensoren. Plattformen er tilpasset å kunne operere inne på Sjøkrigsskolen, og i området rundt. Derfor vil terskelen for å kunne teste noe nytt være lav. Da det ikke kreves flytting til en ekstern lokasjon kan plattformen bli tatt i bruk i undervisning og tester uten ekstra tilrettelegning.

Modulariteten i forbindelse med kobling har vi testet kontinuerlig gjennom oppgaven, da det har vært behov for å fjerne eller flytte på deler. Eksempelvis har vi testet en annen type motorkontroller, dette krevde da kun til/fra kobling ved rekkeklemmer. Noe som går hurtig og risikerer ikke å skade integriteten til eksisterende kabler.

Det vi ser som den største begrensningen i skrivende stund er de motorkontrollene som er i bruk. De har god evne til å styre de motorene som er i bruk, men akselerasjonskurvene er ikke justerbare. Dermed må et eventuelt nytt styringssystem ta hensyn til en tregere respons fra motorer enn nødvendig. Derfor anbefaler vi å anskaffe motorkontroller av større kvalitet med mulighet for å justere akselerasjonskurver selv. Dette åpner også for muligheten til å anskaffe en kontroller med regenerativ bremsing. Alternativet vi har vurdert er Kelly controller KBS36051X (Kelly, 2018). Grunnet tidsperspektivet hadde vi ikke muligheten til å bestille og teste denne.

Alt i alt har vi lykkes med å lage en robust, modulær og fleksibel plattform for undervisning og videreutvikling av autonomi-konseptet i alle lag.

6 Referanser

Battery University

2018. BU-808: How To Prolong Lithium-Based Batteries. Hentet 20.04.18 fra:

http://batteryuniversity.com/learn/article/how_to_prolong_lithium_based_batteries

Camden, Raymond

2014. Using the HTML5 Gamepad API to Add Controller Support to Browser Games. Hentet 06.02.18 fra:

https://gamedevelopment.tutsplus.com/tutorials/using-the-html5-gamepad-api-to-add-controller-support-to-browser-games--cms-21345

Digi-Key Electronics

2016. How To Power And Control Brushless DC Motors. Hentet 25.04.18 fra:

https://www.digikey.jp/en/articles/techzone/2016/dec/how-to-power-and-control-brushless-dc-motors

Ebay

2018. DC 12v-36v 15a 500w Brushless Motor Controller Hall BLDC Driver Board LJ. Hentet 20.05.18 fra:

https://www.ebay.com/p/DC-12v-36v-15a-500w-Brushless-Motor-Controller-Hall-BLDC-Driver-Board-LJ/596268454

Elevich, Leonard N.

2005. 3-Phase BLDC Motor Control with Hall Sensors Using 56800/E Digital Signal Controllers. Hentet 20.03.18 fra:

https://www.nxp.com/docs/en/application-note/AN1916.pdf

Gao, Xianhu

2013. BLDC Motor Control with Hall Sensors Based on FRDM-KE02Z. Hentet 16.04.18 fra:

https://www.nxp.com/docs/en/application-note/AN4776.pdf

Heli-Planet

2018. E-motoren und Flugregler. Hentet 24.05.18 fra:

https://www.heli-planet.com/motor_und_regler.html

KarlRunge

2002. x11vnc: a VNC server for real X displays. Hentet 07.05.18 fra:

http://www.karlrunge.com/x11vnc/

Kelly

2018. Mini Brushless DC Controller. Hentet 24.05.18 fra:

http://kellycontroller.com/kbs36051x25a24-36v-mini-brushless-dc-controller-p-501.html

MODMYPI

2017. HC-SR04 Ultrasonic Range Sensor on the Raspberry Pi. Hentet 24.05.18 fra:

https://www.modmypi.com/blog/hc-sr04-ultrasonic-range-sensor-on-the-raspberry-pi

Mozilla

2018. Using the Gamepad API.

https://developer.mozilla.org/en-US/docs/Web/API/Gamepad_API/Using_the_Gamepad_API

Nakjem, Morten

2016. UT AV FARLIGE SITUASJONER. Hentet 20.05.18 fra:

http://2016.ffi.no/mine

Parker Electromechanical Automation

2018. Hall Sensor: 60 vs 120 degrees. Hentet 15.05.18 fra:

https://www.parkermotion.com/dmxreadyv2/faqsmanager/faqsmanager.asp?question=175

Raspberry.org

2018. Raspberry Pi 3 Model B schematics. Hentet 25.04.18 fra:

https://www.raspberrypi.org/documentation/hardware/raspberrypi/schematics/rpi_SCH_3b_1p2_reduced.pdf

Ron Robotics

2014. 4 Wheeled Robot Design Basics and Challenges. Hentet 24.05.18 fra:

https://www.rakeshmondal.info/4-Wheel-Drive-Robot-Design

Rosell, Christopher

2017. A Sony DualShock 4 userspace driver for Linux. Hentet 20.03.18 fra:

https://github.com/chrippa/ds4drv

TightVNC

2018. Fast and Reliable Remote Control / Remote Desktop Software. Hentet 07.05.18 fra:

https://www.tightvnc.com/download.php

Wilson, Dave

2011. Motor Control Compendium. Texas Instruments

Zwiebertje

2017. Raspberry pi DIN rail holder. Hentet 01.02.18 fra:

https://www.thingiverse.com/thing:2403648

A. Tester

A.1 Test av Hall effekt-sensorer

Formål

Teste om Hall effekt-sensorene ligger elektriske 60º forskjøvet eller 120º forskjøvet fra hverandre.

Metode

Vi satt en målepinne fra et multimeter i hver av de 3 utgangene til hall effekt-sensorene. Ved høyt signal skal sensoren gi 5V, ved lavt signal skal den gi 0V. Vi skrev da ned kombinasjonen av høye og lave signal som ble vist. Deretter roterte vi hjulet helt til en av verdiene endret seg, og skrev ned kombinasjonen. Dette gjentok vi helt til vi hadde vært gjennom elektrisk 360º, og signalkombinasjonen viste kombinasjonen som vi startet med.

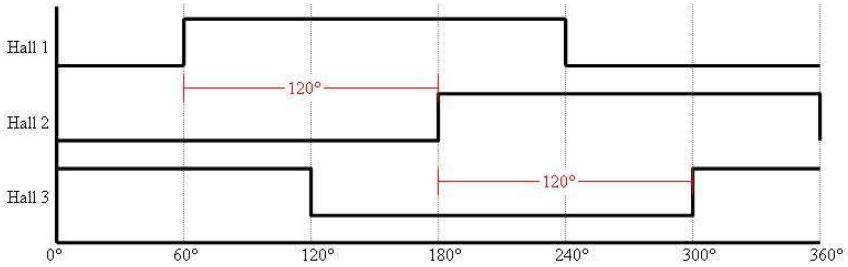
Resultat

Tabell A.1: Hall effekt-sensor kombinasjoner

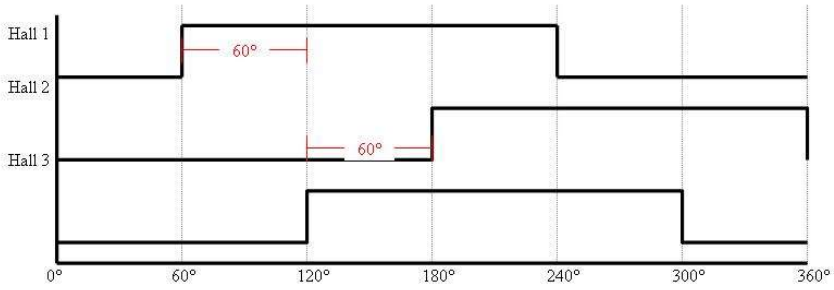
Elektrisk grad	Hall A	Hall B	Hall C
0º	0	1	0
60º	1	1	0
120º	1	0	0
180º	1	0	1
240º	0	0	1
300º	0	1	1

Drøfting

Vi endte med 6 forskjellige kombinasjoner. Ingen av kombinasjonene var 000 eller 111. Som vist i figur A.1 og A.2 under er det ikke mulig at Hall effekt-sensorene ligger 60º forskjøvet. De må derfor ligger med en forskyvning på 120º fra hverandre. Dette er viktig for kommuteringssyklusen.



Figur A.1: Hall effekt-sensorer med 120º forskyvning (Parker Electromechanical Automation 2018)



Figur A.2: Hall effekt-sensorer med 120º forskyvning (Parker Electromechanical Automation 2018)

A.2 Test av ultralydsensorer

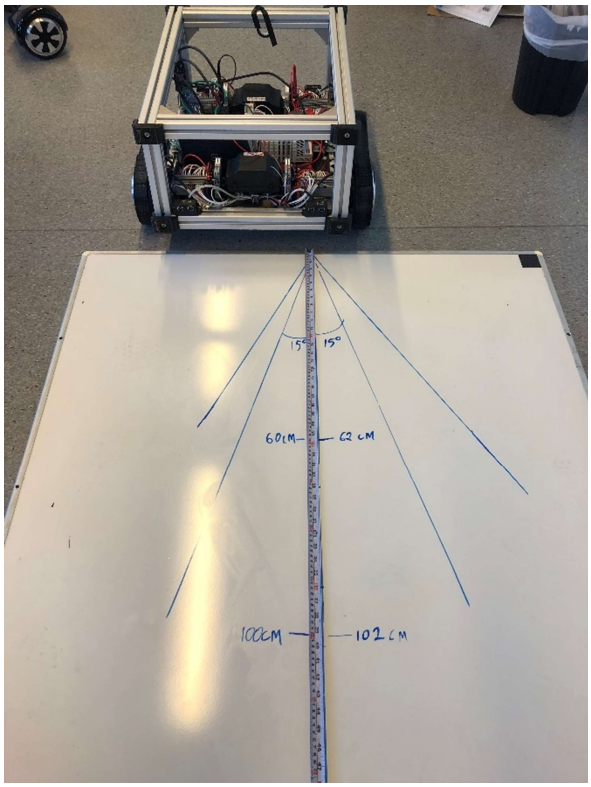
Formål

Teste nøyaktighet i måling av avstand til ultralydsensorene, og teste om de fysiske vinklene stemmer med de oppgitte vinklene (15º) til ultralydsensorene.

Metode

Vi satte opp en ultralydsensor ved enden av en whiteboard-tavle, og markerte opp senterlinjen til ultralydsensoren. Den teoretiske vinkelen ble tegnet opp på hver side av senterlinjen. Deretter brukte vi en stor kasse som utslagsgiver for sensoren. Vi startet ved senterlinjen og la den gradvis lenger fra senterlinjen, helt til vi ikke fikk utslag på den. Punktet ble markert. Dette ble gjort på begge sider.

For testing av nøyaktighet ble boksen satt opp ved flere markerte punkter langs senterlinjen. Den målte verdien ble skrevet opp sammen med den verdien ultralydsensoren ga.



Figur A.3: Teste vinkler på ultralydsensorene

Resultat

Den oppgitte vinkelen var 15º fra senterlinjen. Vinkelen som ble målt under testen ble 25º fra senter, altså 50º totalt. Under testing av nøyaktighet målte vi opp 2 punkter, ett på 60 cm, og et på 100 cm. Ultralydsensoren ga ut avstand på henholdsvis 62cm og 102cm.

Drøfting

Den oppgitte vinkelen viste seg å være litt for konservativ, da våre tester ga oss totalt 20º mer. Det gir oss større område å overvåke per sensor. Sensorene ga ikke utslag for bakken, som da vil si at vinkelen ikke er konet. Dette må tas i betraktning ved bruk av sensorene. Nøyktighetstesten ga oss en differanse på ca 2 cm, over relativ lang avstand (>50cm). Dette er ikke like nøyaktig som selger reklamerer for, (3mm nøyaktighet… kilde).

Feilkilder under testingene kan være unøyaktig merking og måling.

A.3 Test av maks turtall og fart på motor

Formål

Teste maks turtall og fart på motor med motorkontrollerene som blir brukt.

Metode

Plattformen ble satt på opp på den ene siden, slik at hjulene kunne rotere fritt i luften.

Motorene ble kjørt med maks pådrag inn på motorkontrolleren (5V). Turtall i rpm og fart i km/t ble målt med både ekstern måler (ELMA DT-2236) og PLS tellemodul (750-404/000-005). Testing ble gjort både med og uten belte, ved spenning på 42V.

Figur A.4: Turtallsmåler. ELMA DT-2236

Resultat

	Med belte	Uten belte
Turtall [rpm]	820	840
Fart [km/t]	26,3	26,9

Det viste seg at målingene fra den eksterne måleren og PLS programmet var tilnærmet like.

Drøfting

Testene ble gjort under ideelle omstendigheter, uten friksjon fra bakken og maksimal spenning. Som resultatet viser gir belte litt mer friksjon som gjør at turtallet blir likt lavere. Ved en maksfart på 26,9 km/t uten belte kan vi sikkert si at den klarer å operere i gangfart på 5-8 km/t. Lignende tester ble gjort ved lavere spenning, som førte til litt lavere turtall. Lignende tester er også gjort med en sterkere motorkontroller, som kunne gi 1680W. Denne ga motoren et turtall på nærmere 2500 rpm. Ved ønskede utvidelser på fremdriftslinjen er det altså mulig å ta i bruk større motorkontroller, så lenge motorkontrolleren er termisk dimensjonert for motorisolasjonen. Ved større effekter enn nominell-effekt (350W) anbefales det å bare kjøre motorene i korte perioder samtidig som man har god overvåking av temperatur i motor, for å ikke skade motorisolasjonen.

A.4 Test av regenerativ bremsing

Formål

Teste om regenerativ bremsing vil kunne tas i bruk for plattformen.

Metode

Plattformen ble satt på opp på den ene siden, slik at hjulene kunne rotere fritt i luften.

Belte ble satt på motor 2 og 4. Spenningstilførsel til motorkontroller 4 ble kuttet, mens motorkontroller 2 fortsatt var koblet til batterispenning. Ved kjøring av motor 2, ble da motor 4 dratt rundt som generator på grunn av beltedriften. Spenningen generert av motor 4 ble målt sammen med turtallet på motoren.

Resultat

Et turtall på 588 rpm genererte 36,6V i motor 4. 588 rpm tilsvarer en fart på 18,8 km/t.

Drøfting

For at regenerativ bremsing skal fungere må generert spenning fra motor som går som generator være høyere enn spenning på batteriet. Resultatet sier oss at vi må kjøre plattformen med en fart på ca 19 km/t for å kunne lade batteriet så lenge batterispenningen ligger 36,6 V eller under. Da batteriet er 42V fulladet, og plattformen hovedsakelig opereres i gangfart, blir det tilnærmet ingen regenerativ bremsing. Kun ved høye turtall når batterispenning er lav vil det bli bremsing.

A.5 Test av lastekapasitet

Formål

Teste lastekapasitet på plattformen.

Metode

Vi kjørte plattformen på veier med forskjellig grad av helning og med forskjellig last, samtidig som vi tok notater om last og helning. Testingen ble gjort uten belter.

Resultat

Plattformen kjørte opp en bakke med 5⁰ helning, med en last på 82 kg på det meste. Den hadde noen problemer med grep, da rammen er så stiv at hjul kan ende over bakken ved ulent terreng.

Figur A.5: Helning på bakke målt til 5⁰

Drøfting

Da motorene er tatt fra Hoverboard, som er laget for å forflytte personer, er det ikke unaturlig at plattformen klarer å kjøre i oppoverbakke med tyngre last. Med belter vil plattformen få bedre grep på bakken, slik at spinning av motor vil forekomme sjeldnere.

A.6 Test av effektforbruk

Formål

Teste effektforbruk for komponenter og batterikapasitet.

Metode

Vi brukte et multimeter Amprobe 38XR-A med Strøm Sonde Tektronix A622 for måling av strømtrekk av komponentene. Målingene ble gjort i 5 km/t for å få et realistisk bilde over effektforbruk ved operasjonsfart.

Figur A.6: Multimeter Amprobe 38XR-A med strøm Sonde Tektronix A622

Resultat

Time-kolonnen er et tidsestimat for hvor lenge komponenten trekker strøm i forhold til en 1 timers periode.

Last	Antall	Effekt [W]	Spenning [V]	Strøm [mA]	Tid i bruk [h]	Effekt*tid*antall [Wh]
Motor	4	36	36	1000	1	144
Servo	1	3,75	5	750	0,05	0,1875
PLS	1	5,52	24	230	1	5,52
Raspberry Pi	1	2,3	5	460	1	2,3
Ultralydsens-orer	8	0,01	5	2	1	0,08
Ruter	1	3,5	5	700	1	3,5
Totalt						155,6

Drøfting

Hvis plattformen kjøres kontinuerlig i én time i 5 km/t, hadde det effekttrekket vært 155,6 W.

Med en batteripakke på 288 Wh, ville plattformen kunne kjørt i 1 time og 51 minutter, forutsatt at man kunne bruke all kapasitet på batteriene. Da batteriene kutter spennings-tilførsel før de er totalt utladet vil ikke dette være tilfelle. Ved normalt driftsmønster vil ikke motorene trekke strøm kontinuerlig, som vil endre effektforbruket. Vi kan da med sikkerhet si at plattformen kan kjøres kontinuerlig i hvert fall 1 time og 30 minutter, i 5km/t på flate med 0º helning.

B. Dokumentasjon

B.1 Regnskap

Kostnadsoversikt for Bachelor					
Artikkel	Antall	Stykkpris (kr)		Total kostnad (kr)	
		Beregnet	Faktisk	Beregnet	Faktisk
Konstruksjon					
Aluminiumsprofiler 4x4mm, 2m	4	275	275	1100	1100
Al. Braketter 4x4mm	24	8.3333	18.5	200	444
Al. Muttere 8 mm	50	3	2.6	150	130
Bolter 8 mm	30	6.7	2	201	60
Plater	1	200		200	0
Plastikk til 3D-printing	1	350	1750	350	1750
Andre Festemekanismer	1		382	0	382
Kabler	1	500	50	500	50
Bolter 8 mm	30		2.4	0	72
Shunt resistans	3		50	0	150
Strips	1		160	0	160
Diverse				0	0
Fremdrift					
Hjul og motor, par	3	2000	1600	6000	4800
Batteri*	3	0	0	0	0
Motorkontroller*	6	100	150	600	900
Servomotor	1	200	300	200	300
Bremsewire	1	60	60	60	60
Bremsekaliper og bremseskive	1	200	158	200	158
Lader*	1	600	0	600	0
Styring					
PLS	1	2000		2000	0
Raspberry Pi 3	1	500	350	500	350
HC-SR04 Ultralydsensor (ebay)	10	30	18	300	180
HC-SR04 Ultralydsensor (Kjell)	1	100	70	100	70
PS kontroller	1	500	600	500	600
DC-DC omformer 36->5V	1	100	100	100	100
DC-DC omformer regulerbar	2	36	36	72	72
Hovedstrømsbryter	1	200	229	200	229
Minnekort Pi	1	150	150	150	150
Totalsum				14283	12267
Uventede kostnader					
Legg til 30 %				4285	0
Endelig totalsum				18568	12267

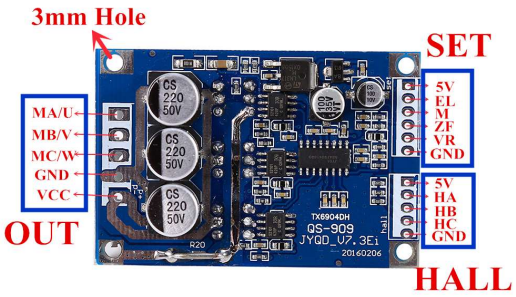
B.2 PlayStation 4 Kontroller Oversikt

Figur B.1: PlayStation 4 Kontroller

Pil Opp	
Pil Ned	
Pil Høyre	
Pil Venstre	
L1	Mink maks fart
L2	Brems
L3	Styring med 1 joystick, og 2
PlayStation knapp	Power knapp

X	Autobrems
Runding	Styring fra HMI
Firkant	2 Joystick styring
Trekant	1 joystick styring
R1	Øk maks fart
R2	
R3	Styring med 2 joy-sticker
Start	

B.3 Motorkontroller Oversikt



Figur B.2: Motorkontroller, tabell under forklarer innganger og utganger.

Power terminal Out:

MA/U	Motor fase U
MA/V	Motor fase V
MA/W	Motor fase W
GND	0V
VCC	Positiv spenningstilførsel

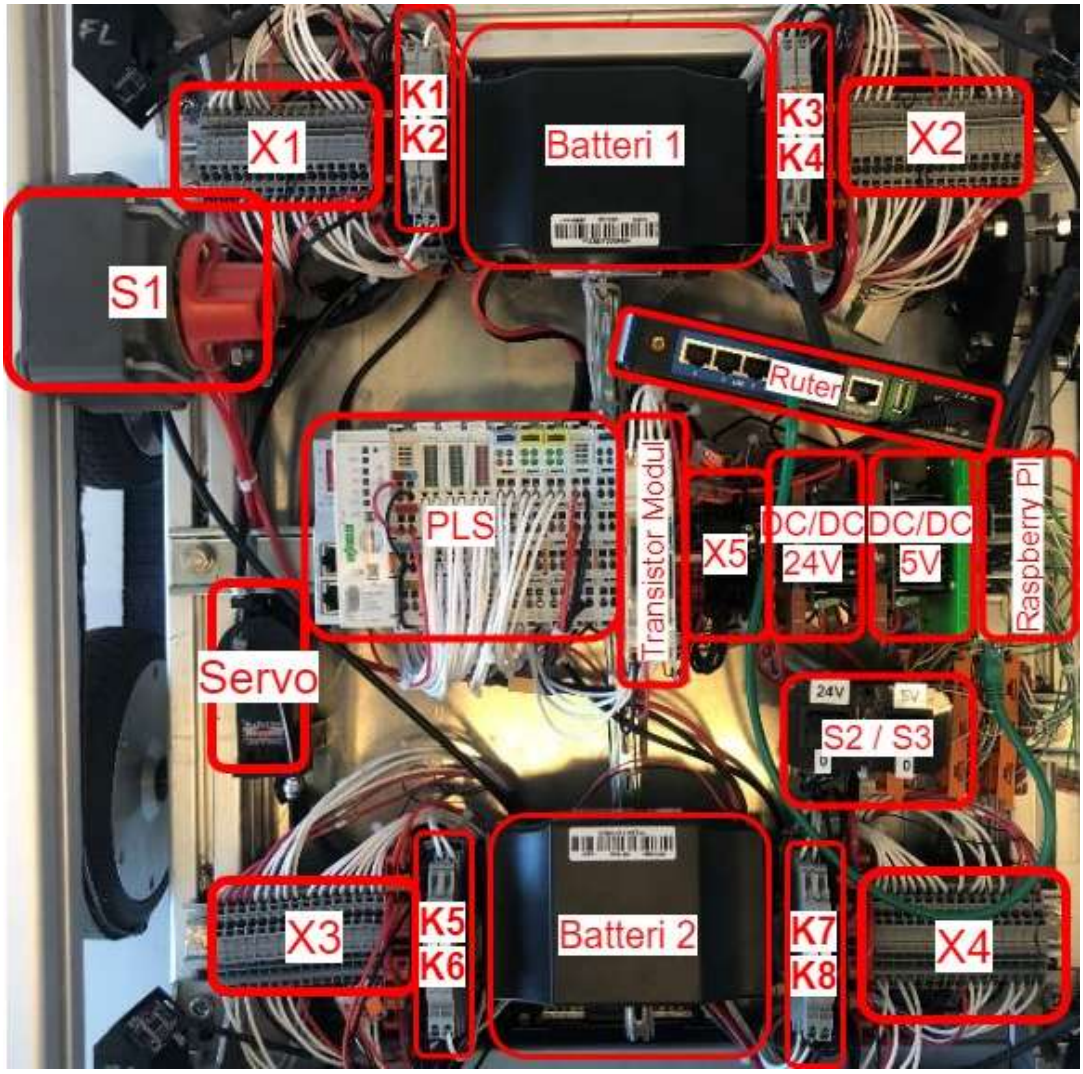
Set for the control side:

5V	5V utgang
EL	På/av kontroll. 5V inn: Motor kan rotere, GND inn: Motor star i ro
M	Tachometer Puls Utgang. Gir 5V-pulser for hver Hall-effekt sensor puls
ZF	Revers. 5V inn: motor spinner ene veien, GND inn: motor spinner andre veien.
VR	Pådragsinngang. 0-5V ingang.
GND	0V

Hall effekt-sensors:

5V	5V inngang til Hall effect-sensorene
Ha	Signal fra Hall effect-sensor a
Hb	Signal fra Hall effect-sensor b
Hc	Signal fra Hall effect-sensor c
GND	0V

B.4 Arrangementstegning



S1: Hovedstrømsbryter

S2: Bryter 24V

S3: Bryter 5V

X1-X4: Rekkeklemmer

K1-K6: 24V Releer

B.5 Rekkeklemmetabeller

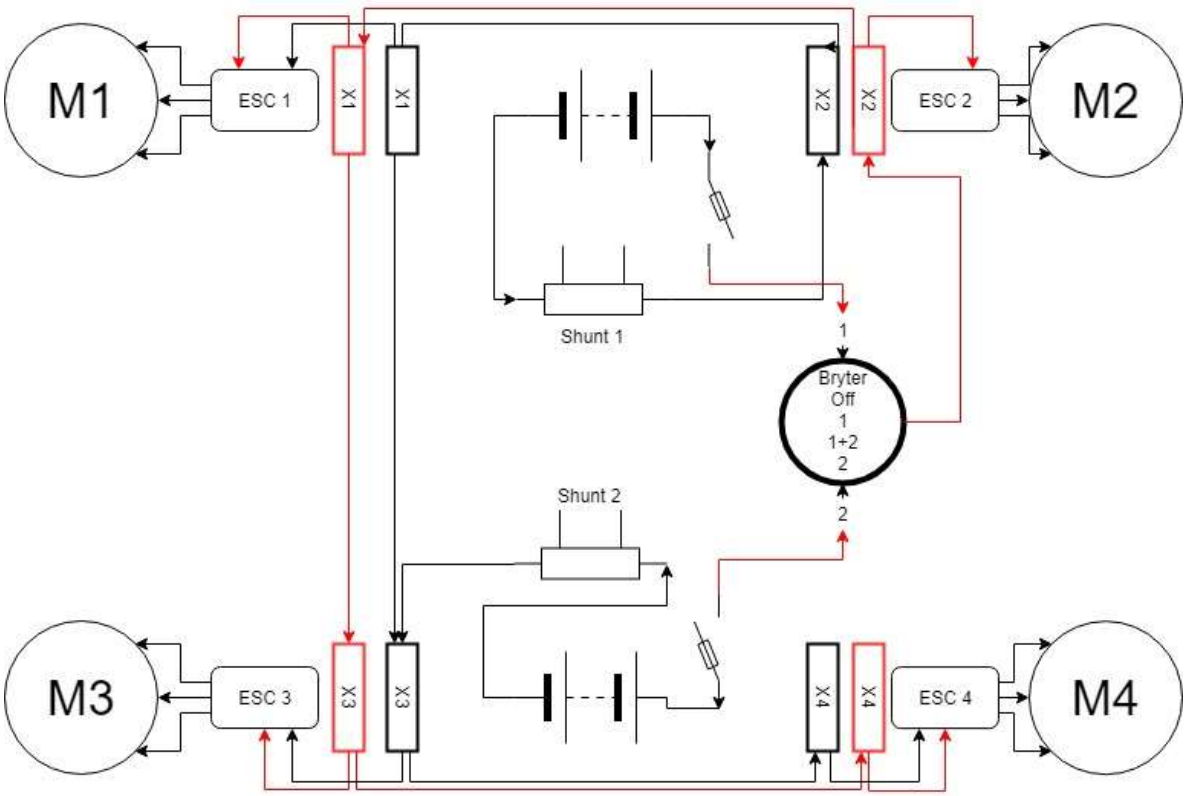
X1		
M1: C	1	ESC 1: C
M1: B	2	ESC 1: B
M1: A	3	ESC 1: A
M1: Hall +	4	ESC 1: Hall +
M1: Hall C	5	ESC 1: Hall C
M1: Hall B	6	ESC 1: Hall B
M1: Hall A	7	ESC 1: Hall A
M1: Hall -	8	ESC 1: Hall -
M1: PT	9	PLS: RTD 1.1
M1: PT	10	PLS: RTD 1.9
ESC 1: PT	11	PLS: RTD 2.1
ESC 1: PT	12	PLS: RTD 2.9
ESC 1: VR	13	PLS: AO 1
ESC 1: ZF	14	K2: 14
ESC 1: M	15	TM: 2.1
ESC 1: EL	16	K1: 12
36 V	17	
0 V	18	
0 V	19	
0 V	20	

X2		
M2: C	1	ESC 1: C
M2: B	2	ESC 1: B
M2: A	3	ESC 1: A
M2: Hall +	4	ESC 1: Hall +
M2: Hall C	5	ESC 1: Hall C
M2: Hall B	6	ESC 1: Hall B
M2: Hall A	7	ESC 1: Hall A
M2: Hall -	8	ESC 1: Hall -
M2: PT	9	PLS: RTD 1.3
M2: PT	10	PLS: RTD 1.11
ESC 2: PT	11	PLS: RTD 2.3
ESC 2: PT	12	PLS: RTD 2.11
ESC 2: VR	13	PLS: AO 3
ESC 2: ZF	14	K3: 12
ESC 2: M	15	TM: 2.2
ESC 2: EL	16	K4: 12
36 V	17	
0 V	18	
0 V	19	
0 V	20	

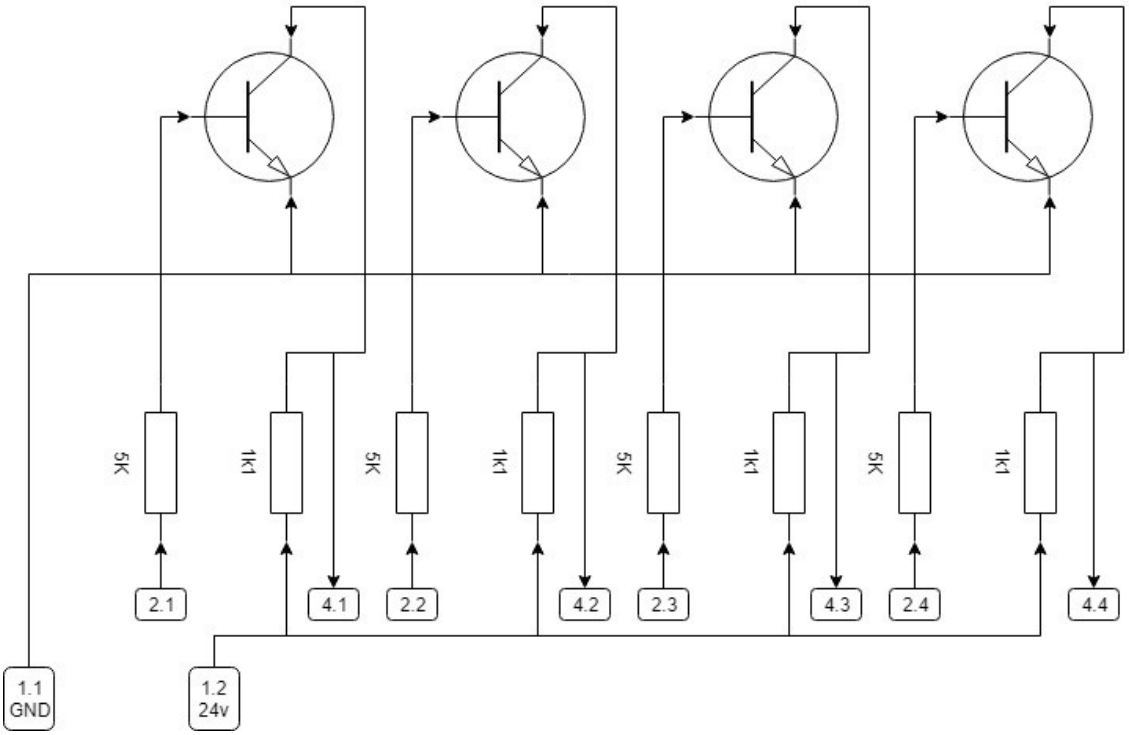
X3		
M3: C	1	ESC 1: C
M3: B	2	ESC 1: B
M3: A	3	ESC 1: A
M3: Hall +	4	ESC 1: Hall +
M3: Hall C	5	ESC 1: Hall C
M3: Hall B	6	ESC 1: Hall B
M3: Hall A	7	ESC 1: Hall A
M3: Hall -	8	ESC 1: Hall -
M3: PT	9	PLS: RTD 1.5
M3: PT	10	PLS: RTD 1.13
ESC 3: PT	11	PLS: RTD 2.5
ESC 3: PT	12	PLS: RTD 2.13
ESC 3: VR	13	PLS: AO 5
ESC 3: ZF	14	K6: 14
ESC 3: M	15	TM: 2.1
ESC 3: EL	16	K5: 12
36 V	17	
36 V	18	
0 V	19	
0 V	20	
0 V	21	

X4		
M4: C	1	ESC 1: C
M4: B	2	ESC 1: B
M4: A	3	ESC 1: A
M4: Hall +	4	ESC 1: Hall +
M4: Hall C	5	ESC 1: Hall C
M4: Hall B	6	ESC 1: Hall B
M4: Hall A	7	ESC 1: Hall A
M4: Hall -	8	ESC 1: Hall -
M4: PT	9	PLS: RTD 1.7
M4: PT	10	PLS: RTD 1.15
ESC 4: PT	11	PLS: RTD 2.7
ESC 4: PT	12	PLS: RTD 2.15
ESC 4: VR	13	PLS: AO 7
ESC 4: ZF	14	K7: 12
ESC 4: M	15	TM: 2.1
ESC 4: EL	16	K8: 12
36 V	17	
36 V	18	
0 V	19	
0 V	20	
0 V	21	

B.6 Hovedstrømskjema



B.7 Transistor Modul koblingsskjema



B.8 Oppstartsprosedyre

For å kunne kjøre plattformen må hovedstrøms bryter stå i 1+2 posisjon.

Bryter for 24V og 5V DC-DC omformere må skrus på.

For å benytte Playstation 4 kontroller direkte koblet til Raspberry Pi:

Trykk først på Playstation knappen i midten av kontrollen, kontroller blinker da blått og vil lyse blått kontinuerlig når den er tilkoblet Raspberry Pi.

Fra en pc på samme nettverk kan TightVNC benyttes for å koble til Raspberry Pi, eventuelt kan ekstern skjerm, mus og tastatur benyttes.

På Raspberry Pi må Firefox åpnes og adressen 127.0.0.1:1880/gamepad skrives inn. Når kontrollen betjenes kan verdier på knapper og joysticker ses i nettleservinduet.

Plattformen er nå klart til å kjøres.

Hvis Playstation 4 kontrolleren skal kun benyttes direkte koblet til Raspberry Pi kan Raspbian operativsystemet konfigureres til å starte Firefox og åpne gamepad nettsiden ved oppstart. Dermed er det ikke nødvendig å koble seg til Rapsberry Pi hver gang ved oppstart.

B.9 Raspberry Pi GPIO tilkoblinger

Variabelnavn	Fysisk pin
F1 – trigger	8
F1 – echo	3
F2 – trig	5
F2 – echo	7
B1 – trigger	23
B1 – echo	29
B2 – trigger	31
B2 – echo	33
FR – trigger	11
FR - echo	10
FL – trigger	15
FL – echo	13
BR – trigger	35
BR – echo	37
BL – trigger	19
BL – echo	21
Brake	16

C. Kildekode:

Kildekode for PLS program er lastet opp i PDF format, programfilen (bachelor.ecp) er også lastet opp med oppgaven.

Kildekode for NodeRed er vedlagt som tekstfil. Innholdet i denne kan kopierers og limes inn i NodeRed. I tillegg er bibloteksfilene til NodeRed vedlagt som en .rar fil, disse kan importeres i NodeRed.

HTML koden for Gamepad nettsiden er inkludert i NodeRed koden, men også lastet opp som en separat html fil.

All kildekode, 3D-print design og ekstra bilder fra byggeperioden ligger også tilgjengelig via Get Sky-linken:

https://www.getsky.no/p/get-60197696/_9f0f0499e2384a58a790b878990e1c2e