



# UNIVERSITETET I AGDER

## MODULBASERT DRONE TILPASSET AUTONOM BRUK I MOTION CAPTURE SYSTEM

VÅR 2018

AV

JØRGEN MIKAL BENUM  
SIMON HUS  
JAN PETTER OTTESEN  
OLA CHRISTOFFER VÅGE

VEILEDER:  
Kristian Muri Knausgård

INSTITUTT FOR INGENIØRVITENSKAP  
FAKULTET FOR TEKNOLOGI OG REALFAG

---

## 1 Forord

Bacheloroppgaven markerer slutten på bachelorprogrammet Mekatronikk ved Universitet i Agder. Oppgaven dekker flere av aspektene som inngår i Mekatronikk. Studiet har krevd strukturert og målrettet arbeid, som både har vært krevende og svært lærerikt.

Ved Universitet i Agder får studentene lov til å benytte seg av Mekatronikk laben under arbeidet med bacheloroppgaver. Dette gir studentene en unik mulighet til å sette teori de har lært gjennom studiet ut i praksis. Den valgte oppgaven representerer bredden innenfor mekatroniske systemer på en god måte, og inneholder elementer fra en betydelig andel av fagfeltene vi har blitt undervist.

Samtidig har Universitetet i Agder en "open door policy" hvor forelesere og professorer på tvers av linjer og fakulteter tar imot og svarer utfyllende på spørsmål. Tilknyttet prosjektet ønsker vi å takke Trond-Ivar Lynghaug for hjelp og opplæring i forbindelse med sammenstilling av kretskort, Karl-Berge Rød og Roy Werner Folgerø for hjelp med 3D-print av skrog.

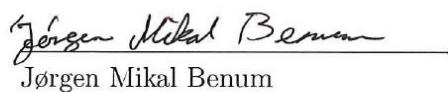
Stor takk til vår veileder Kristian Muri Knausgård god veiledning og støtte gjennom prosjektet.



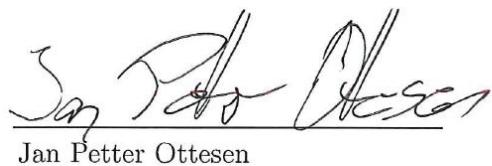
Simon Hus



Ola Christoffer Våge



Jørgen Mikal Benum



Jan Petter Ottesen

---

## 2 Sammendrag

Prosjektet tar for seg utviklingen av en modulbasert drone tilpasset bruk i MotionLab ved Universitetet i Agder. MotionLab er utstyrt med et motion capture system fra Qualisys. Dronen er designet med utgangspunkt i å bruke dette systemet til å erstatte GPS og magnetometer for å hente ut posisjon og orientering. I tillegg til motion capture tilpasning, har dokking-stasjon for lading, masseproduserbarhet og krav til vekt under 250 gram spilt betydelige roller i utforming av skrog og komponentvalg.

Testing av ulike motororører og propeller er utført for å danne et godt grunnlag til å velge den best egnede konfigurasjonen til dronens applikasjon.

ROS (Robot operating System) ble tidlig valgt som rammeverk for software implementasjon. ROS deler opp funksjonaliteten til dronen i noder, og bidrar med det til høy modularitet i software. Denne strukturen tillater separat testing og verifisering av hver enkelt node, uavhengig av hverandre og resten av systemet.

For kommunikasjon mellom hardware moduler ble det valgt å benytte CAN bus-nettverk. CAN bus gjør det mulig å modularisere hardware ved hjelp av et ryddig meldings-grensesnitt, med en oppbygning av noder som kan minne om ROS. Med hensyn til vektkrav og begrenset plass er det designet, produsert og programmert kretskort med CAN-grensesnitt. Disse er godt egnet for gjenbruk i andre typer mobile roboter.

En matematisk modell av dronen er bygd opp fra bunnen av og verifisert. Nødvendige forenklinger er gjennomført på modellen for å kunne dekoble dens dynamikk. Dette for å muliggjøre SISO tuning av kompensatorer/PID-kontrollere. Den dekoblede modellen er videre testet og sammenlignet med den kobledes i simulator, for å verifisere at den er representativ.

Dronens stabilisering er bygd opp rundt bruk av IMU og motion capture. Stabiliseringen er bygd opp ved bruk av tilbakekoblingsløkker i kaskade, på en slik måte at navigasjon kan implementeres i senere arbeid.

Som sluttresultat endte prosjektet med en sammenstilling hvor all software og hardware fungerte godt sammen. Tekniske problemer med Qualisys systemet ga problematikk med å kontinuerlig spore merkede objekters posisjon og orientering i labvolumet. Derimot fungerte vinkelposisjon og vinkelrate i lukket sløyfe med IMU under tester utført på bakken.

# Innholdsfortegnelse

<b>1</b>	<b>Forord</b>	<b>i</b>
<b>2</b>	<b>Sammendrag</b>	<b>ii</b>
<b>3</b>	<b>Introduksjon</b>	<b>1</b>
3.1	Bakgrunn og motivasjon . . . . .	1
3.2	Oppgavebeskrivelse . . . . .	1
<b>4</b>	<b>Konsept</b>	<b>2</b>
4.1	Mekanisk konsept . . . . .	2
4.2	Udstyr . . . . .	4
4.3	Motor og drivverk . . . . .	6
4.4	Systemarkitektur og styring . . . . .	7
4.5	Kommunikasjon . . . . .	8
4.6	Konseptvalg . . . . .	9
<b>5</b>	<b>Teori</b>	<b>11</b>
5.1	Konstruksjon av deler og skrog . . . . .	11
5.1.1	3D-modellering og simulering . . . . .	11
5.1.2	Material og 3D-print . . . . .	12
5.1.3	Vakuumstøping . . . . .	13
5.1.4	Duct design . . . . .	13
5.2	Strøm . . . . .	14
5.2.1	Batteri . . . . .	14
5.2.2	Ledninger . . . . .	16
5.3	Kretskort . . . . .	16
5.3.1	Sporbredde strømbaner . . . . .	16
5.3.2	Beregning av strømforbruk . . . . .	18
5.4	Konfigurasjon av attiny mikrokontrollere . . . . .	19
5.5	Robot Operating System (ROS) . . . . .	21
5.6	Qualisys - motion capture . . . . .	21
5.7	Koordinatsystemer og referanserammer . . . . .	22
5.7.1	Lokalt north-east-down (NED) koordinatsystem: . . . . .	22
5.7.2	Lokalt egedefinert koordinatsystem: . . . . .	23
5.7.3	Kroppssentrert north-east-down (NED) koordinatsystem . . . . .	23
5.7.4	Kroppsfast koordinatsystem . . . . .	24
5.7.5	Oppsummering av koordinatsystemer og referanserammer . . . . .	25
5.8	Orientering og rotasjon . . . . .	25
5.8.1	Rotasjonsmatriser . . . . .	26
5.8.2	Euler-vinkler . . . . .	26
5.8.3	Transformasjoner . . . . .	29
5.9	Dynamikk . . . . .	31
5.9.1	Dronens konfigurasjon . . . . .	31
5.9.2	Roterende referanserammer . . . . .	32

5.9.3	Treghetsmoment og akser . . . . .	34
5.9.4	Gyroskopisk momentbidrag . . . . .	36
5.10	Regulering . . . . .	37
5.11	Motortest . . . . .	38
5.11.1	Mekanikk . . . . .	38
<b>6</b>	<b>Metode</b>	<b>41</b>
6.1	Skrog . . . . .	41
6.1.1	Duct . . . . .	44
6.2	FEM-analyse . . . . .	47
6.2.1	Forskyvning i skroget som følge av kraft og moment fra motor . . . . .	49
6.2.2	Sammenstøt . . . . .	50
6.3	Strøm . . . . .	52
6.3.1	Batteri . . . . .	52
6.3.2	Ledere . . . . .	53
6.4	Ladestasjon . . . . .	53
6.5	Produksjon . . . . .	55
6.6	Motortest . . . . .	56
6.6.1	Thrust test . . . . .	56
6.6.2	Test av vridningsmoment . . . . .	59
6.7	Robot Operating System (ROS) . . . . .	60
6.8	Kretskort . . . . .	62
6.8.1	Egenskaper . . . . .	62
6.8.2	Design . . . . .	62
6.8.3	Valg av komponenter . . . . .	63
6.8.4	Eagle . . . . .	67
6.8.5	Sammenstilling og testing . . . . .	71
6.9	Programmering . . . . .	72
6.9.1	Oppsett av CAN bibliotek . . . . .	72
6.9.2	Programmering . . . . .	74
6.9.3	Oppkobling av CAN . . . . .	76
6.9.4	Funksjonstesting . . . . .	76
6.10	Qualisys . . . . .	78
6.11	Matematisk dronemodell . . . . .	79
6.11.1	Kraftligning . . . . .	79
6.11.2	Momentligning . . . . .	81
6.11.3	Gyroskopsisk moment . . . . .	82
6.11.4	Treghetsmoment-matrisen . . . . .	83
6.11.5	Andre system påvirkninger og forstyrrelser . . . . .	84
6.11.6	Grafisk dronemodell . . . . .	84
6.12	Motormodell . . . . .	85
6.12.1	Moment- og thrust-kraft-funksjon . . . . .	85
6.12.2	Motordynamikk . . . . .	87
6.12.3	Motormodell i Simulink . . . . .	87
6.12.4	Linearisering av Motormodell . . . . .	88
6.13	Forenklet dronemodell . . . . .	89

6.13.1 Dekobling . . . . .	89
6.13.2 Grafisk forenklet modell . . . . .	91
6.14 Sammenligning av modeller . . . . .	92
6.15 Regulering og stabilisering . . . . .	95
6.15.1 Signalmiksing . . . . .	95
6.15.2 Kontroller design . . . . .	96
6.15.3 Simulator . . . . .	96
6.16 IMU . . . . .	102
6.16.1 ROS node . . . . .	102
6.16.2 Komplementærfilter . . . . .	103
6.16.3 Kompensering for IMU plassering og orientering . . . . .	104
<b>7 Resultater</b>	<b>107</b>
<b>8 Diskusjon</b>	<b>110</b>
8.1 Skrog . . . . .	110
8.2 Qualisys . . . . .	111
8.3 Kretskort . . . . .	112
8.4 Modeller og regulering . . . . .	112
8.5 Testing . . . . .	113
<b>9 Konklusjon</b>	<b>114</b>
<b>10 Forkortelser, variabler og definisjoner</b>	<b>115</b>
<b>Referanser</b>	<b>116</b>
<b>Appendiks</b>	<b>120</b>
<b>A Skjematiske tegninger</b>	<b>120</b>
A.1 RPIzw CAN module . . . . .	120
A.2 ESC CAN module . . . . .	121
A.3 Supply Board . . . . .	122
<b>B Kildekode</b>	<b>123</b>
B.1 ESC CAN modul . . . . .	123
B.2 motor_controller ROS node . . . . .	126
B.3 MPU6050 ROS node . . . . .	127
B.4 PS4controller ROS NODE . . . . .	129
<b>C Tekniske tegninger</b>	<b>134</b>
C.1 Toppdel skrog . . . . .	134
C.2 Bunndel skrog . . . . .	135
C.3 Komponentfeste 1 . . . . .	136
C.4 Komponentfeste 2 . . . . .	137
C.5 Ladestasjon del 1 . . . . .	138
C.6 Ladestasjon del 2 . . . . .	139
C.7 Duct 3" (76.2mm) . . . . .	140

<b>D FEM-analyse</b>	<b>141</b>
D.1 Spenninger og forskyvninger som følge av kraft og moment fra motor. Skalering 50:1	141
D.2 Spenninger fra fall. Skalering 5:1	142
<b>E Simulator</b>	<b>143</b>
E.1 Komplett modell av drone	143
E.2 Motormodel	144
E.3 Kompensator	145
E.4 Matematisk modell	146
E.5 Dekoblet modell	147
E.6 Transformasjon forenklet modell	148
E.7 Sum av krefter	149
E.8 Sum av moment	150

## Figurliste

1 Virkninggrad til propell med variabel og fast bladvinkel som funksjon av rotasjonshastighet	3
2 Illustrasjon av docking stasjon med slepering prinsippet	6
3 CAN distribuert system	8
4 Systemskisse	10
5 Testprofiler for materialtest	12
6 Tverrsnitt av duct designet ved University of Maryland	13
7 Spenningsfallet i en LiPo celle over tid	15
8 Utdrag fra IPC-2221A - "External Conductors"	17
9 Spenningsfall over shunt resistor	18
10 Utdrag fra formelsamling MAS-218	19
11 Oversikt over tilgjengelige pwm moduser for timer 0	19
12 Illustrasjon av CTC mode	20
13 Illustrasjon av ROS nettverk	21
14 Lokalt NED koordinatsystemet i globalt perspektiv	22
15 Kroppsfast, kroppssentrert NED og lokalt NED koordinatsystem	25
16 Rotasjonssekvens z-y-x	27
17 Rotasjon om z-aksen	28
18 Motorplassering og rotasjonsretning	32
19 Illustrasjon av gyroskopisk bevegelse	36
20 PID-kontroller med filtrert derivasjonsdel	37
21 FLD - Motortest thrust	39
22 FLD, første test av vridningsmoment	40
23 Motortest vridningsmoment	40
24 V-modell	41
25 Bunndel utformet av med hensyn til batteri	42

26	Komponenter montert på batteri med komponentfester . . . . .	43
27	Tverrsnitt av ferdig montert skrog med batteri og komponenter . . . . .	43
28	Totalvekt av skrog og oppsett av markører til Motion Capture system . . . . .	44
29	Duct montert i testbenk med motor og propell . . . . .	45
30	Thrust . . . . .	46
31	Strømforbruk . . . . .	46
32	Symmetri og fix-point . . . . .	47
33	Sammenligning av mesh . . . . .	48
34	Effekten av å bygge skroget inn mot batteriet . . . . .	49
35	Forskjellen mellom festet og ikke festet topplokk . . . . .	50
36	Sammenstøt ved fall . . . . .	51
37	Valgt batteri . . . . .	52
38	Tversnittareal [ $mm^2$ ] vs. Spenningsfall [V] . . . . .	53
39	Spenningsfordeler . . . . .	53
40	Innhold ladstasjon . . . . .	54
41	Ladestasjon og kontaktpunkter . . . . .	55
42	Ferdig rigg, klar for test, fri luftstrøm foran og bak . . . . .	56
43	Motortest, børstemotor . . . . .	57
44	Sammenligning av propeller, 7.4V, "Turnigy F1104-4000KV 5.5g" . . . . .	58
45	Thrust-test av EMAX RS1106 - GF 3025x2 . . . . .	59
46	Første rigg for måling av vridningsmoment . . . . .	59
47	Endelig rigg for test av vridningsmoment og resultat . . . . .	60
48	Oppbygning av ROS nettverk for testing av drone . . . . .	60
49	0603 Resistor . . . . .	63
50	Utdrag fra AVR Microcontroller Hardware Design Considerations, kap 4.1.1Utdrag fra AVR Microcontroller Hardware Design Considerations, kap 4.1.1 . . . . .	67
51	Utdrag fra AVR's AN2519 . . . . .	67
52	Supply Board, ferdig etset fra Kina, med påført strømtrekk . . . . .	69
53	Inspeksjon av utlegg med gerber viewer . . . . .	70
54	Svikt i kvalitetsikring . . . . .	71
55	Plassering av komponenter på pcb . . . . .	71
56	Avlesning av enhet signatur i Atmel studio . . . . .	72
57	Feilmelding ved forsøk på å bygge can-bibliotek . . . . .	72
58	Fungerende kommunikasjon mellom PCAN-USB adapter og ATtiny85 . . . . .	73
59	Kontroll av pulstog-frekvens og duty cycle . . . . .	75
60	Oppkobling for CAN . . . . .	76
61	Ferdig oppkobling . . . . .	76
62	Kvantiseringsfeil . . . . .	77
63	Markører reflekterer kamerablitz . . . . .	78
64	Sporing av 3D-printet prototype i MotionLab . . . . .	78
65	Dronekropp med påførte krefter og momenter . . . . .	79
66	Dronens treghetsmomenter . . . . .	83
67	Blokkdiagram av dronens dynamikk . . . . .	84
68	Oppdeling av motormodell . . . . .	85
69	Thrust-kraft i newton som funksjon av pådrag . . . . .	85
70	Moment i newtonmeter som funksjon av pådrag . . . . .	86

71	Motorenens rotasjonshastighet (rpm) som funksjon av pådrag . . . . .	87
72	Dekoblet modell . . . . .	91
73	Vinkelposisjons-respons for konstant input på 0.0001 i pitch . . . . .	92
74	Posisjonsrespons for konstant input på 0.0001 i pitch . . . . .	93
75	Vinkelposisjons-respons for konstant input på 0.00001 i roll, 0.0003 i pitch og 0.001 i yaw . . . . .	93
76	Posisjonsrespons for konstant input på 0.00001 i roll, 0.0003 i pitch og 0.001 i yaw . . . . .	94
77	Signalmiksing for dronens regulering . . . . .	96
78	Bode plot av åpen løkke transferfunksjon mellom roll-pådrag og vinkelhastigheten $p$ . . . . .	97
79	Bode plot av lead-kompensator og PID-kontroller for vinkelhastighet $p$ . . . . .	98
80	Bode plot av åpen løkke transferfunksjon med kontrollere . . . . .	99
81	Vinkelhastighet step-respons med implementert kontroller . . . . .	99
82	Bode plot av åpen løkke transferfunksjon mellom roll-pådrag og vinkelposisjon $\phi$ . . . . .	100
83	Vinkelposisjons step-respons med implementerte kontrollere . . . . .	101
84	MPU 6050 node . . . . .	102
85	Vinkelposisjoner i radianer, estimert av komplementærfilter . . . . .	103
86	Illustrasjon av drone og IMU som punktpartikler . . . . .	104
87	Test av Simulink implementasjon av kompensering . . . . .	106
88	Drone i ladestasjon . . . . .	107
89	Resultat ESC CAN modul . . . . .	108
90	diskontinuerlig data fra qualisys system . . . . .	111
91	Sporing av markører i Z-akse . . . . .	111

## Liste over tabeller

1	Materialegenskaper . . . . .	12
2	Konstanter basert på grafer fra IPC-2221A . . . . .	17
3	Vekt komponenter . . . . .	52
4	Ytre dimensjoner, kretskort . . . . .	63
5	Dekobling kondensatorer . . . . .	64
6	Anslatt maksimalt strømtrekk . . . . .	65
7	Nødvendig sporbredde ved 1 tomme lengde og tillat temperaturstigning 10 °C . . . . .	68
8	Verdier grafisk dekoblet modell . . . . .	91

## 3 Introduksjon

### 3.1 Bakgrunn og motivasjon

MotionLab ved Universitetet i Agder består av to Stewart-plattformer av ulik størrelse, som blant annet kan benyttes til å simulere bølgebevegelser under marine operasjoner. Den er også utstyrt med et state-of-the-art motion capture system fra Qualisys [1]. Dette systemet benytter kameraer for å finne nøyaktig posisjon og orientering til merkede objekter i labvolumet.

Med dagens løsning er det problematisk å filme godt nok til å dokumentere et eksperiment tilstrekkelig. En løsning kan være å ta i bruk droner. Dronen kan hente informasjon om sin posisjon og navigere med bruk av Qualisys - motion capture system. Det bør bygges en solid plattform av software og hardware hvor dronen holder seg stabil i rommet før utviklingen av selve filmingen begynner.

For at det skal være mulig for enhver å filme når som helst uten pilotkurs (RO 1 eller høyere) er det ønskelig at dronene holdes utenfor reglementet i forskriften om luftfartøy som ikke har fører om bord [2].

Videre er det ytret et ønske fra Universitetet i Agder å styrke sin satsning på droner i fremtiden. Gjenbrukbarhet bør derfor prioriteres, slik at oppgaven i deler eller som helhet kan brukes i videre arbeid.

### 3.2 Oppgavebeskrivelse

Utvikling av drone med forenklet ladesystem, tiltenkt bruk i MotionLab.

GPS og magnetometer skal erstattes av posisjon- og orienteringsdata fra motion capture systemet, dermed må mulighet for trådløs innhenting av denne dataen foreligge.

Dronen skal lages med hensyn til masseproduksjon og må kunne opereres uten RO kurs, ihht. forskriften om luftfartøy som ikke har fører om bord (totalmasse under 250 gram) [2]. Dronen bør baseres på en modulær plattform, hvor egenskaper og konfigurasjoner kan implementeres eller endres i ettermiddag.

Prosjektet bør inneholde design og konstruksjon av skrog, sammensetning av arkitektur for software og hardware, samt testing av komplett system. For stabilisering bør det utarbeides en matematisk modell med tilhørende forenklinger, som muliggjør SISO tuning av kontrollere tilhørende de aktuelle frihetsgradene.

## 4 Konsept

Før design, vurderes en rekke konsepter. De presenteres her, med tilhørende fordeler og ulemper. Konseptene er delt opp i hovedgruppene listet under, og tar kortfattet for seg generell informasjon og/eller tenkt løsning. I slutten av dette delkapitelet er det lagt frem et konseptvalg som danner grunnlaget for designfasen.

### Hovedgrupper i konsept:

- Mekanisk konsept
  - Rammekonstruksjon
  - Utforming
  - Produksjonsmetode
  - Materialvalg
- Utstyr
  - Gimbal
  - Ladesystem og dokking
- Motor og Drivverk
- Systemarkitektur

### 4.1 Mekanisk konsept

I denne delen er det kun lagt fram ulike rammekonfigurasjoner, samt materialer. Disse to sees ikke i sammenheng da materiale er sterkt avhengig av rammens utforming og produksjonsmetode. Utforming og produksjonsmetode anses som en del av design/utviklingsfasen.

#### Partallbasert multikopter

Et multikopter er per definisjon et flyvende rotorfartøy med 2 eller flere motorer [3]. De mest brukte dronevariantene er quadkopter (fire motors), hexakopter (seks motors) og octokopter (åtte motors) [4]. Bikopter (to motors) blir sjeldent brukt da disse er vanskelige å stabilisere, dronen pitch må konstant reguleres. Begge motorene justeres med hver sin servo for å kunne stabilisere og styre. Bikopteret kan ha fordeler med at de vil ha en lavere vekt og er billigere mtp. konstruksjon. Den største forskjellen på de tre førstnevnte i forhold til antall motorer er utslaget det gjør på parametrene vekt, batteritid, størrelse, stabilitet, løftekraft og hastighet. På disse dronevariantene er antall motorer partallbasert, som gjør at det vil dannes symmetri mellom motorene som utligner rotasjonsmomentene hver propell danner. Prinsippet vil bli det samme hos alle disse dronene.

Å øke antall armer gir økt stabilitet, dronen kan da være i stand til å miste motorkraft fra flere motorer men samtidig klare å fly kontrollert. Med flere motorer øker løftakraften, og dronen vil da kunne løfte tyngre last, men også ha en høyere egenvekt. Å øke antall armer, øker også

antall motorer som samlet trekker mer strøm, men igjen så kan det da la seg gjøre å bruke mindre og billigere motorer, og eventuelt børstemotorer. Generelt vil mer armer gi mer vekt, det er stabilitetskravet som bestemmer om det er en god eller dårlig løsning.

## Oddetallsbassert multikopter

Oddetallsbaserte multikoptre vil ikke kunne bruke motorene sine rotasjonssymmetrier for å utligne rotasjonsmomentet som blir dannet om dronens z-akse (yaw). En av motorene må med denne løsning kunne tiltes for å kompansere for dette, og for å kunne kontrollere dronens yaw. De vanligste variantene er trikopter (tre motors) og pentakopter (fem motors). Oddetallsbasserte droner er generelt mye mindre foretrekket, på grunn av mer avansert styring og stabilitet enn partallbaserte droner

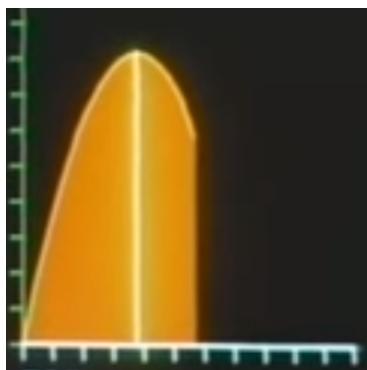
Andre oddetallsbasserte multikopter-løsninger kan eksempelvis være quadkopter bassert pentakopter. Der fire av motorene har symmetri og kompenserer for rotasjonsmomentet, og den femte brukes for å øke hastighet horisontalt.

## Helikopter

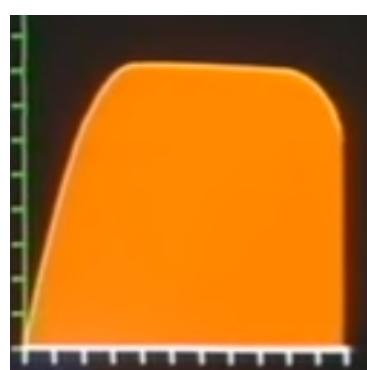
I denne fremleggingen er helikopter definert til å være et luftfartøy med en hovedrotor for løft og en halerotor til kompensering for rotasjonsmoment gitt av hovedmotoren. I tillegg til rotorene, brukes det generelt vinkling av hovedrotorens blader for å oppnå translasjon i planet. Vinklingen ses også i sammenheng med løft (i tillegg til hovedrotorens trutall), roll og pitch.

I den følgende teksten under er det tatt fram noen hovedpunkter rundt det å bygge en helikopter basert drone. Disse om handler både fordeler og ulemper.

Den regulerbare vinklingen av bladene gir mulighet for å optimalisere vinkling av bladene mot turtall på hovedrotor. Dette resulterer i høyere virkningsgrad over et større område sammenlignet med en propell med faste blader [5], noe som er en fordel med tanke på flytid, se figur 1. Samtidig innebærer systemet som vinkler bladene mye finmekanikk. I tillegg til at det må brukes en mer kompleks propell og motorer/servo-er som kan betjene vinklingen.



(a) Fast bladvinkel



(b) Variabel bladvinkel

Figur 1: [5] Virkninggrad til propell med variabel og fast bladvinkel som funksjon av rotasjonshastighet

Med tanke på ramme/kropp vil en helikopter konstruksjon kunne gi en kompakt og lett ramme. Dette fordi det kun er nødvendig med en "arm" ut til halerotoren, og at ramme ikke trenger å ha noe stort omfang volummessig, utover å gi plass til alle komponenter. Imidlertid tilkommer noe ekstra vekt som følge av at man må ha minimum to aktuatorer til å justere vinklingen av bladene. I tillegg til at store deler av helikopterkroppen normalt sett vil ligge under hovedrotoren og dermed ha negativ innvirkning luftstrømmen under propellen.

## Quadkopter

De siste årene har det blitt mer og mer droner i Norge. I slutten av november 2017 hadde 230 000 nordmenn en drone. [6] Når det prates om "drone" på denne måten menes det stort sett quadkopteret. Et quadkopter består av fire armer, en motor festet på hver av dem. De fire motorene utligner rotasjonsmomentet hver propell danner som fortalt i 4.1. Så for å kontrollere quadkopteret reguleres turtallet på motorene hver for seg. Det åpner for å endre yaw, pitch, roll og løft hver for seg eller sammen.

Vektmessig er ikke quadkopter vinneren blant dem vi har sammenlignet. Men med hensyn til designkravene til den fysiske konfigurasjonen nevnt i 3.2 kommer quadkopteret gunstig ut med tanke på et produksjonsvennlig design. Quadkopter ramme muliggjør mange utforminger som kan masseproduseres grunnet lite kompleks mekanikk. Noe monteringsarbeid må likevel påregnes.

## Material

Materialet dronen bygges av er viktig både i forhold til vekt og styrke. Droner blir stort sett bygget av materialer som karbonfiber, glassfiber, plastikk, aluminium eller isopor. Fellesnevner for gode drone materialer er høy stivhet og lav vekt. For å klare målene settes hovedfokuset på lav vekt. Styrke kommer i andre rekke, da dronens oppgave ikke innebærer akrobatikk eller tunge løft som krever overdreven styrke. Disse materialene er alle greie å håndtere med tanke på produksjon. Valg av materiale er sterkt avhengig av både rammekonfigurasjon utforming. Dermed tas materialvalg hensyn til i designfasen. Uansett materialvalg, planlegges det å lage prototypen med universitetets nye 3d-printer, som printer plast, forsterket med enten kevlars, glassfiber eller karbonfiber.

## 4.2 Utstyr

### Gimbal og kamera

For å kunne styre kameraet til å sikte på en ønsket posisjon og holde bildet stabilt under manøvrering trengs det en gimbal/aktuator. Det finnes mange varianter på markedet, men ytterst få av de kommersielle har lav nok vekt til bruk på droner under 300g. Regulering av rull og pitch ses på som tilstrekkelig, da yaw til en viss grad kan styres av selve dronen. De to vanligste aktuatorene for å styre en gimbal er børsteløse motorer og servoer.

Servo aktuering gir potensielt en veldig lett løsning da det er god tilgjengelighet av servoyer mellom 3-5g på markedet. Disse styres med pwm direkte fra mikrokontroller. Totalvekt med rammekonstruksjon og servomotorer er mulig å få helt ned mot 22g.

En stor ulempe ved å bruke tradisjonelle rc-servoer er treghet. De fleste rc-servoer opererer på 50 Hz. Det vil si at servoen må vente 20 ms mellom hver posisjons-endring. Dette vil være ugunstig når du ønsker et glatt og fint bilde. Rc-servoer vil også vibrere litt når de er spennningsatt, noe som heller ikke er gunstig når gimbal'ens oppgave i utgangspunkter er å sikre stabilt bilde.

Børsteløse motorer vil gi en mer responsiv og glattere styring av kameraene. De fleste gimbaler aktueres med børsteløse motorer med lav KV. Ulempen ved å bruke børsteløse motorer er i hovedsak vekt. Av tilgjengelige kommersielle motorer er 24g den letteste motoren funnet. To stk 5g servo motorer kontra to stk 24g børsteløse motorer vil skille 38g. Dette tilsvarer minst 15% av dronens totale vekt. Til tross for vektfordelen servomotorene innehar, vil bruk av børsteløse motorer være foretrukket. De vil sikre en presis og glatt bevegelse for kameraet, og dermed de beste bildene.

### Ladesystem og dokking

Droner generelt har lav batteritid og lang ladetid. Dette gjør at de ofte står på lading mer enn de flyr. Generelt sett er det tungvindt å lade dronen. Da batteriet manuelt må kobles fra dronen for så å koble batteriet til ladekabelen/ladestasjon. Vi vil se på mulighetene for å lage en dokking stasjon dronen flyr til, for å lade seg selv når den går tom for strøm. Følgende konsept er lagt fram:

### Stable i sylinder

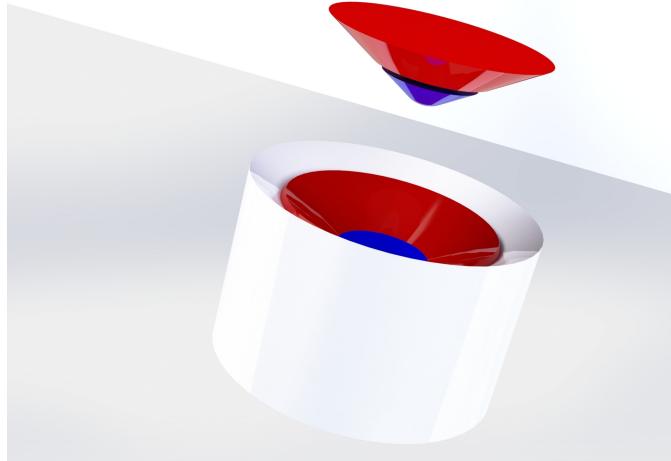
En løsning for å få et kompakt system for å lade flere droner er å stable dronene i en sylinder. Her må det lages en løsning for å skyve ut den nederste dronen som har ladet lengst. Dette er en god løsning, men er avhengig av rammevalget. At det er valgt et design som støtter at dronene kan stables i høyden og lades gjennom hverandre.

### Induktiv lading

Trådløs lading er et godt alternativ for å få automatikk i ladeprosessen. Med behovet for fysisk tilkobling fjernet gjenstår det bare å ha en presis nok navigasjon til å lande nærmere nok til å oppnå induksjonslading. Bakdelen er at løsningen blir avansert. Det er lite kommersielle induksjonsladere for droner. Det er stort sett brukt til tannbørster og mobiler, som lader med lavere spenning enn vi vil måtte gjøre. Resultatet blir at vi selv må lage hele induksjons ladesystemet fra bunn av, inkl. spoler. Det må også lages støtte-elektronikk for å håndtere egenfrekvens og litium batteriets ladeprosedyre. [7]

### Slepering

Ved å bruke slepering prinsippet, ser gruppen muligheten for å lage dronens ramme utformet som en kon i bunnen og lande den i en tilsvarende motsatt utformet docking stasjon. Den vil ha minus-polen i bunn og pluss-polen i topp av kon. Den vil da kunne lande i et landingsområde på 360 grader om z-aksen, som gjør at dronens yaw vinkel ved landing kan neglisjeres. Dette vil da være en løsning som gjør at vi ikke trenger å fysisk koble kabler sammen under lading, og som lar seg stackes i f.eks en matrise på en vegg.



Figur 2: Illustrasjon av docking stasjon med slepering prinsippet.

### 4.3 Motor og drivverk

Til motor kan børste- eller børsteløs motor velges. Den største forskjellen på de er at en børsteløs motor trenger en separat driver (ESC) for å regulere turtallet. Dette er ikke nødvendig på en børstemotor, der turtallet kan reguleres ved å endre spenningen direkte på coilen (ved bruk av f.eks en transistor).

Børsteløse motorer er dyrere enn børstemotor ved innkjøp, i tillegg til tilhørende driver som gjerne kan komme opp i samme pris og vekt som selve motoren. Til gjengjeld har de lengre levetid, siden de ikke har børster som slites ut av friksjon mot rotor. Lav friksjon gjør også at de genererer mindre varme.

Børstemotorer, er billigere enn børsteløs ved innkjøp (selges ofte i fire-pakninger). De trenger ikke egen driver, som gjør dem gunstige for lettvektsdroner. Ulempen med disse er at de har et lavere hastighetsområde pga. mekaniske begrensninger på børstene. Friksjon gjør at effekttapet blir større desto høyere hastigheten blir [8]. I tillegg er utvalget av lettvekts børstemotorer med trust over 25 gram begrenset. Et kjapt søk i nettbutikk for droner gir inntrykk av at droner med MTOM over omtrent 80 gram går over fra børste til børsteløse motorer.

For valg av motor er det faktorene trust, vekt og tilgjengelighet som blir satt i fokus. Motorene må være dimensjonert slik at de kan arbeide innenfor det området hvor trusten er lineæret avhengig av inputsignalet. Da ulineære systemer er ugreie å regulere. For å finne de optimale dimensjonene på motor som skal bli brukt, vil det i designfase bli testet forskjellige motorer med forskjellige propeller.

## 4.4 Systemarkitektur og styring

### Systemarkitektur

Oppsett av systemarkitekturen omhandler totalsystemet utgjort av hardware og software. Her er det mange mulige konfigurasjoner, under er tre av de mest aktuelle lagt frem. Det er kun lagt fram oppsett som kan bygges med lav vekt, da dette er kritisk for å oppnå en MTOM under 250 gram.

De tre konseptene under kan deles i to hovedgrupper, hvor enten et kort håndterer alt, eller flere kort jobber sammen og deler på oppgavene (hardware-modularitet). For løsningene med to kort, er det tatt utgangspunkt i å bruke en Raspberry Pi Zero w (wifi) (RPIzw) (blant annet med hensyn på wifi tilkobling) i kombinasjon med en ekstern mikrokontroller. Her er også andre kort aktuelle blant annet GUMSTIX OVERO, men disse konkurrerer ikke med RPIzw i pris.

Første konsept er å bruke RPIzw til å utføre all prosessering, og benytte en ekstern mikrokontroller som et rent "I/O kort". RPIzw vil da motta posisjons-signal fra Qualisys-systemet og data fra Inertial measurement unit (IMU). Den dataen vil også kunne brukes for algoritmer til banefølging og stabilisering av drone. Ekstern mikrokontroller vil kun motta informasjon fra RPIzw om pådrag den skal sende ut til hver enkelt aktuator. Med 2-akset gimbal og 4 børsteløse motorer vil man eksempelvis trenge 6 pwm utganger fra I/O kortet. Dette vil gi en begrenset modalitet hvor man kan simulere bestemte pådrag og teste motorer før algoritmer er klare.

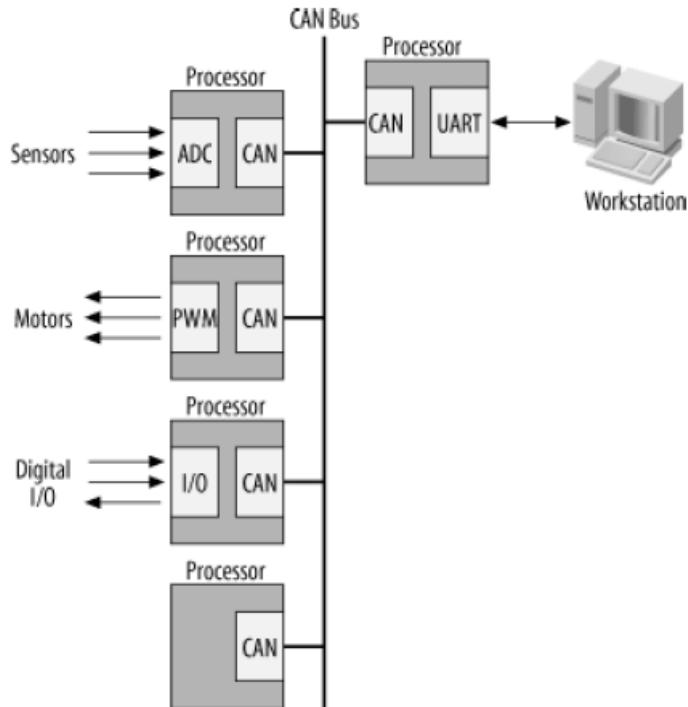
En annen mulighet er å bruke en ekstern mikrokontroller til å håndtere stabiliseringen og den essensielle flyvingen, mens banefølging og pådrag inn til mikrokontrolleren håndteres av en RPIzw. Dette gir god oppdeling av systemet, slik at stabilisering og navigasjon kan utvikles parallelt, man må i hovedsak kun ta hensyn til interface mellom de to kortene. Samtidig gir det mulighet for at pådrag fra RPIzw kan ignoreres av mikrokontroller om dette er kritisk for å holde dronen i luften, eller om RPIzw feiler. Denne løsningen gir også mulighet for å velge mellom å styre kamera-gimbal med enten mikrokontroleren eller RPIzw.

En tredje mulighet er å bruke et kretskort som håndterer alt, fra pådrag til motorer, til banefølging og navigasjon. Dette reduserer behovet for kabling til kommunikasjon og tilførsel fra batteri.

## 4.5 Kommunikasjon

Det er ønskelig med en kommunikasjonsplattform som tar høyde for modularitet og forskjellige grensesnitt. Det brukes sensorer og motordrivere som trenger å kommunisere over PWM og SPI eller I2C. CAN bus åpner for samspill på tvers av grensesnitt samtidig som det er robust mot støy og modulært.

CAN bus ble utviklet av bosch mot slutten av 1980-tallet for bilindustrien. De så den gang et økende antall sensorer og kompleksitet i ledningsnettet på biler, som krevde stor informasjonflyt mellom komponentene. Dette sørget for potensielle feilkilder og unødvendig store kabellengder. Det elektriske miljøet i bilindustrien er i tillegg fult av støykilder. Eksempelvis elektriske motorer, tenningsystem og radiofrekvenser. Løsningen på alle disse utfordringene ble CAN bus [9].



Figur 3: [9] CAN distribuert system

CAN plattformen gjør det mulig å implementere sanntids kommunikasjon med hastigheter opp mot 1 Mbps over et 2-lednings serielt nettverk. Dermed kan en med fire ledninger tilføre komponentene power og toveis kommunikasjon. Figur 3 illustrerer et modul basert CAN bus system som kommuniserer på tvers av grensesnitt via CAN bus.

## 4.6 Konseptvalg

Basert på konseptene presentert ovenfor ble følgende konsept valgt:

### Ramme

Quadkopter ramme med produksjonsvennlig design. Quadkopter konfigurasjon er den løsningen som gir minst mekanisk kompleksitet samtidig som stabiliteten er god, noe som både har betydning for pris og utviklingstid. En viktig faktor her er at i tillegg til at dronen skal designes og bygges, så skal den også kunne være operativ til testing og banefølging.

Selv utformingen av quadkopteret kommer i designfasen, og sees i sammenheng med valg av produksjonsmetode og materiale. Prototype lages med Universitetets Markforged X7 printer.

### Utstyr

Det vil bli satt av plass og vekt til kamera og gimbal i rammedesignet, men denne oppgaven vil ikke gå inn på design og utvikling av gimbal.

Til lading produseres det docking-stasjon som følger slepering prinsippet som framlagt i 4.2, for å forenkle ladeprosessen i kombinasjon med oppbevaringsmulighet, med en enkel konstruksjon. Det muliggjør at ladestasjon kan legges tilgjengelig for allmenheten, eksempelvis med veggmontering. Ladesstasjonens utforming må også hensyntas ved utvikling av rammen, slik at den er kompatibel med ladestasjonen.

### Motor og drivverk

Valg av propell og motor blir basert på testresultat av ulike kombinasjoner utført i designfasen. Her må det både tenkes på linearitet i arbeidsområdet (for reguleringens del) og strømforbruk. Batterivalg vil bli tatt ut i fra tilgjengelig vekt, samt at spesifikasjoner passer til motorene. Det må imidlertid legges inn en estimert størrelse på batteri for design av ramme.

### Systemarkitektur og styring

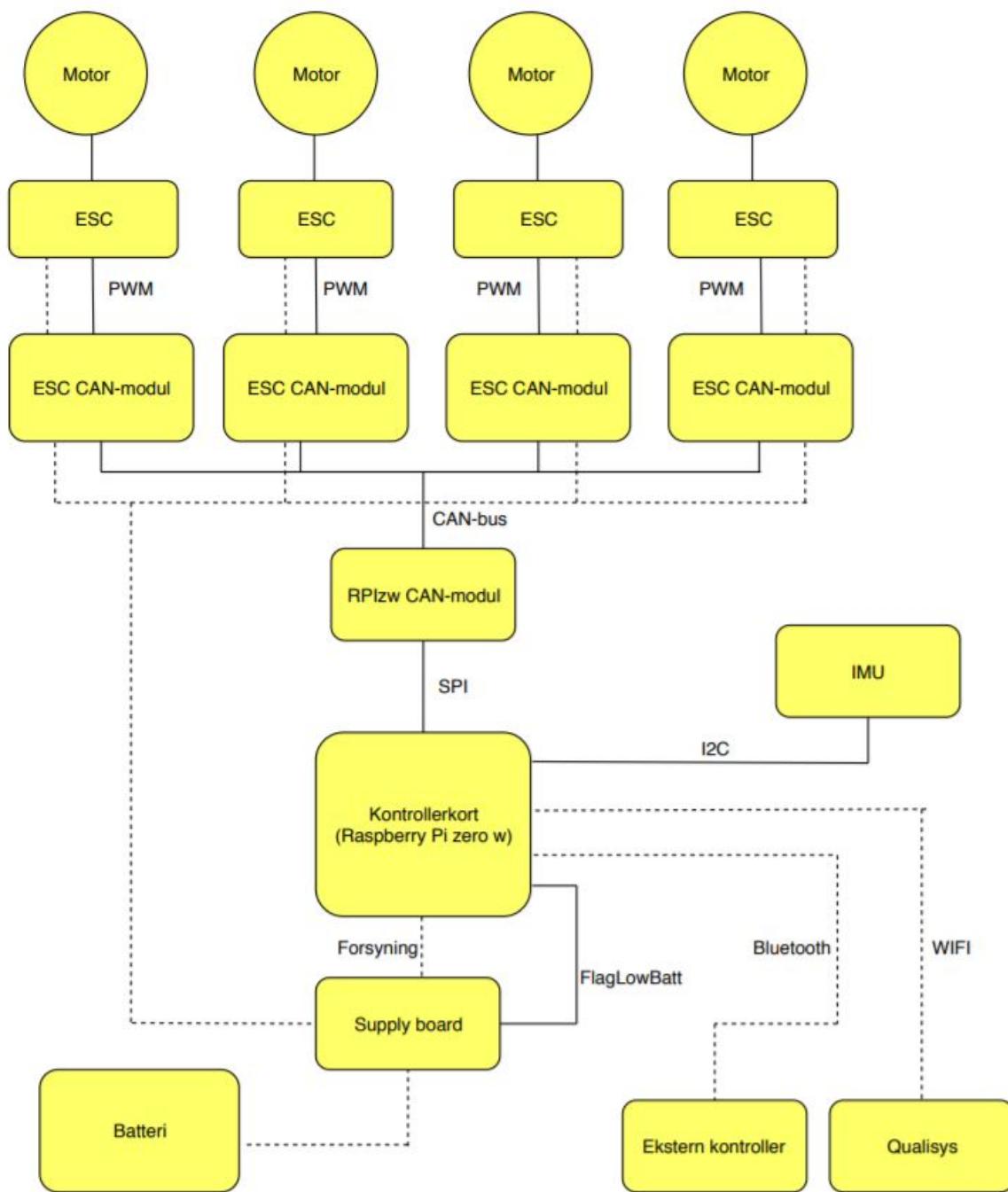
Det er valgt å bruke et kontrollerkort til å håndtere alle oppgaver, dette innebærer både banefølging, stabilisering, navigasjon, videolagring og eventuelt video-overføring. De to sistnevnte vil ikke implementeres i dette prosjektet, men muligheten for å implementere det skal ligge til rette. Ved å gjøre alle oppgaver på et kort, kan resursene som er tilgjengelige kortet utnyttes mest mulig. Som kontrollerkort er det valgt å bruke en RPIzw, hovedsaklig på grunn av pris, tilgjengelighet og vekt.

### Kommunikasjon

Det skal brukes CAN-bus til å kommunisere mellom RPIzw og hver motors ESC. I forbindelse med dette må det lages egne kretskort for håndtering av CAN-bus.

Sensormessig er det lagt opp til å bruke en IMU som minimum har akselerometer og gyroskop, for bedre fastsettelse av posisjon og orientering.

Programvare vil bli basert på et nettverk bygget i Robot Operating System (ROS), med en node hvor styring- og stabiliserings-algoritmen kjøres ved bruk av Simulink.



Figur 4: Systemskisse

Ut fra valgt konsept ble det utarbeidet en grafisk systemskisse som er vist over i figur 4.

## 5 Teori

### 5.1 Konstruksjon av deler og skrog

#### 5.1.1 3D-modellering og simulering

For å designe skroget og diverse komponenter ble 3D-CAD programmet SolidWorks benyttet. SolidWorks brukes i en rekke industrier, og i 2013 ble det registrert at over 2 millioner ingeniører i over 165 000 bedrifter på verdensbasis benyttet dette programmet [10]. Her tegnes alle deler som benyttes i dronen, og det gir et svært nøyaktig bilde av hvordan dronen vil se ut og hvordan den må bygges i fysisk form. Ved hjelp av simuleringsverktøyet kan en utføre diverse elasitistestester som Finite Element Method (FEM) for å se hvordan skroget vil oppføre seg under flygning, samt hvordan skroget best mulig bør utformes med fokus på stivhet, styrke og vekt.

Plasseringen til felles massemidtpunkt er viktig å lokalisere for å kunne utvikle en presis regulering. Om materiale med korrekt densitet tilegnes aktuelle deler i modellen, vil en enkelt kunne hente ut massemidtpunkt og treghetsmomenter fra Solidworks.

FEM-analyse innebærer at det numerisk beregnes endringer i en struktur, basert på påførte ytre påkjenninger. Strukturen blir delt opp i ett gitt antall elementer som blir beregnet hver for seg, for så å bli satt sammen til et større system av beregninger som til slutt gir et resultat av hvordan strukturen vil oppføre seg. Oppdelingen av elementer og noder blir gjort ved å danne en maske (mesh) over konstruksjonen. Det blir da hovedsakelig brukt CBM (curvature-based mesh) og SM (standard mesh). CBM fordeler et gitt antall element over et område. Den kan da dekke udetaljerte områder med store element og konsentrere flere små element inn mot mer detaljerte steder på konstruksjonen. Vinklene innad i elementene kan variere i størrelse, alt etter kompleksiteten på området det blir fordelt i. CBM lar deg velge største og laveste elementstørrelse, som da vil tilpasse seg konstruksjonen deretter. SM vil fordele element av en gitt størrelse og størst mulig vinkler innad i elementene, med en toleranse på hvor nøyaktig elementene skal fordeles. Dette kan da føre til at komplekse områder ikke blir nøyaktig målt om toleransen på elementene ikke er høy nok. Å dele strukturen opp i en slik maske av element og noder er metoden som brukes til å analysere både påkjenninger og forskyvninger i konstruksjonen [11] [12].

### 5.1.2 Material og 3D-print

Ved å bruke 3D-printer kan delene designet i SolidWorks printes ut direkte. 3D-print foregår ved at materialet som brukes blir bygget opp lag for lag. En får raskt ut ønsket del i fysisk form, og dette gjør 3D printing optimalt for prototyping. Universitetet har printerne CubePro, Dimension og Markforged tilgjengelig for prototyping. Til mindre komplekse konstruksjoner blir CubePro brukt, denne printer i materialet ABS-plast. For print av de mer komplekse skrogdelene til dronen egner Markforged printeren seg bedre, da den har avstandsmåling til tilbakekoblings-løkke og printer dermed veldig nøyaktig. Den printer i materialet Onyx, som er et nytt sterkt material tiltenkt som et alternativ til Nylon. Den har også mulighet for å printe med fiberforsterkning i kevlar, glassfiber eller karbonfiber.

Det ble testprintet 2 testprofiler, en på 1mm og en på 0.5mm. Resultatet sees i figur 5. Hensikten var å fysisk sjekke egenskapene til materialet i forhold til vår bruk. Disse hadde lignende former som armene på dronen, som kunne "klapses" sammen. For å teste printerens nøyaktighet ble det lagt til 2mm hull, som er samme dimensjon som motorfestene.



Figur 5: Testprofiler for materialtest

Tabell 1: Materialegenskaper

Material	Onyx	ABS	Nylon
Strekkfasthet (MPa)	36	47	54
Fleksibilitetstyrke (MPa)	81	66	32
Densitet(g/cm <sup>3</sup> )	1.18	1.07	1.10

Viktige faktorer ved valg av materiale er strekkfastheten, fleksibilitetstyrken og densiteten. Disse faktorene for materialene ABS, Onyx og Nylon er listet i tabell 1 ovenfor. Alle disse materialene har egenskaper som vil egne seg godt til bruk som materiale i skroget, da de har god styrke i forhold til sin lave densitet. Materialdata er hentet fra en forskningsrapport angående formstøping av ABS-plast [13], og fra Markforged sitt eget datablad for Onyx [14] og Nylon [15].

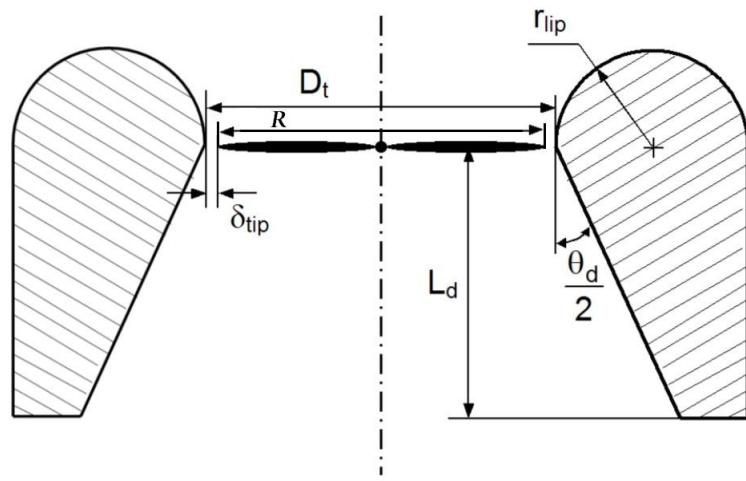
### 5.1.3 Vakuumstøping

Vakuumstøping er en metode som blir brukt ved utforming av deler av tynt metall eller plast. Metoden innebærer å lage et verktøy som er en negativ av delen som skal lages. Valgt material legges over og blir varmet opp til temperatur ved grensen til smeltepunktet. Materialet blir så sugd ned over negativen ved hjelp av en vakumpumpe og danner ønsket del. Etter at materialet har blitt avkjølt er delen praktisk talt et skall som kan løftes av negativen. Det kreves lite bearbeiding i form av kutting og sliping, før delen kan ansees som ferdig. Med tanke på at delen er et skall som løftes av verktøyet er det viktig at designet ikke legger materialet mer enn en  $90^\circ$  vinkel over formen slik at verktøyet kan tas ut.

Dette er en metode som fungerer bra med tanke på masseproduksjon, da et verktøy kan brukes til å produsere et stort antall deler. Eventuelt kan også flere verktøy lages for økt produksjon.

### 5.1.4 Duct design

Når skroget skal designes, kan ducter benyttes for å øke effektiviteten til propellene. Det innebærer at det monteres en sirkulær vegg rundt propellene, som skal bidra til en bedre utnyttelse av propellene sin thrust. Dette bidrar til at mer thrust fokuseres direkte nedover. Det oppstår også mindre turbulenser i luftstrømmen rundt rotoren, som fører til at den opplever mindre motstand under rotasjon. I et forskningsprosjekt utført ved University of Maryland ble det funnet mål på det som generelt sett skal være det optimale designet av en duct [16]. Illustrert i figur 6.



Figur 6: Tverrsnitt av duct designet ved University of Maryland

Alle mål er avhengig av  $D_t$  som en beregner ut fra valgt propellstørrelse  $R$ . Tilhørende figur 6 er vist under og basert på rapporten fra University of Maryland.

$$R = D_t - 2 \cdot \delta_{tip} \quad (5.1)$$

$$= D_t - 2 \cdot 0.001 \cdot D_t \quad (5.2)$$

$$= D_t(1 - 2 \cdot 0.001) \quad (5.3)$$

$$\Rightarrow D_t = \frac{R}{1 - 2 \cdot 0.001} \quad (5.4)$$

$\theta_d$  settes alltid lik  $10^\circ$ . Følgende ligninger blir benyttet for å finne resten av målene:

$$\delta_{tip} = 0.001 \cdot D_t \quad (5.5)$$

$$L_d = 0.5 \cdot D_t \leftrightarrow 0.72 \cdot D_t \quad (5.6)$$

$$r_{lip} = 0.13 \cdot D_t \quad (5.7)$$

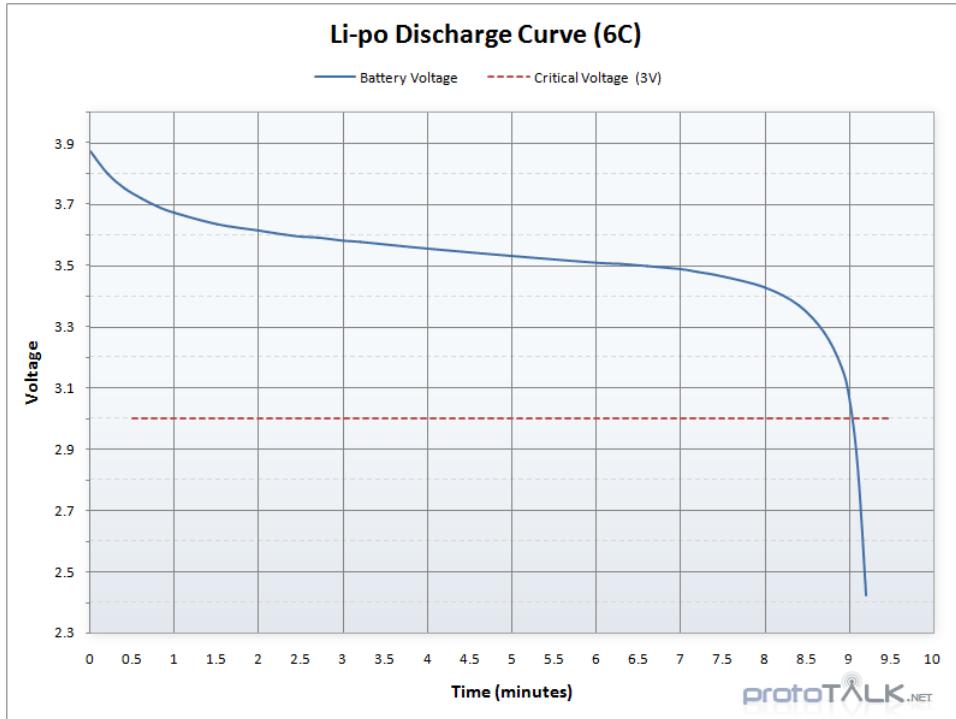
## 5.2 Strøm

### 5.2.1 Batteri

De aktuelle batteriene for droner er LiPo batterier. De består av en, eller flere LiPo celler. Batteri med flere celler refereres til som 2S, 3S, 4S osv.

En LiPo celle har nominell spenning på 3.7V. 3.7V er bestemt av industrien til å være den nominelle spenning. Fulladet har en celle 4.2V og laveste trygge ladning er 3.0V. LiPo batterier er godt egnet til bruk i droner. De har ikke en linær utladning, istedenfor holder de seg lenge på nominell spenning, men når de er utladet stuper de til bunns [17]. Kontroll på forbruk er derfor viktig.

I figur 7 ser en grafisk hvordan utladningen av en LiPo celle på 3.7V forekommer. Spenningen stuper raskt fra 4.2V ned til 3.7V, og holder seg jevnt synkende ned til 3.4V. Den stuper så raskt nedover til 2.4V. Dette viser da at nesten hele utladningsforløpet til cellen vil holde en stabil spenning.



Figur 7: [18] Spenningsfallet i en LiPo celle over tid

### Utladningsrate / C-rate

C-rate er en måleenhet for hvor mye strøm som kontinuerlig kan trekkes fra et batteri. For å beregne hvor mange ampere som kontinuerlig kan trekkes fra batteriet, gitt en C-rate må en også kjenne batteriets kapasitet. Eksempelvis ser vi i ligning 5.9 at et batteri med 20C og  $1600mAh$  kan en trygt trekke max  $32A$  av. Charge rate føres likt, den er ofte lavere og beskriver hvor hurtig batteriet kan lades.

$$I_{kont} = \text{C-rate} \times \text{"batterikapasitet [Ah]"} \quad (5.8)$$

$$\text{Eksempel: } I_{kont} = 20h^{-1} \cdot 1,6Ah = 32A \quad (5.9)$$

### 5.2.2 Ledninger

Ledninger er stort sett laget av kobber eller aluminium. Det er metaller som leder godt, men har høy vekt. For å kunne få vekt av ledninger til et minimum brukes følgende teori for å velge lavest mulig tverrsnittsareal for ledninger. [19].  $\rho$  er metallets resistivitet.

$$U = R \cdot I \quad (5.10)$$

$$R_{leder} = \frac{\rho \cdot l}{A_{tverrsnitt}} \quad (5.11)$$

$$\Delta U = R_{leder} \cdot I \quad (5.12)$$

Ligning (5.11) for resistans i leder, settes inn i ligning (5.12) og løses for tvernsittsarealet. Lengde l multipliseres med 2 for å ta hensyn til leder både til og fra:

$$A_{tverrsnitt} = \frac{\rho \cdot 2l \cdot I}{\Delta U} \quad (5.13)$$

Etter valg av tverrsnitt er strømtrekk og spenningsfall gjennom leder kjent. Dermed kan effekten som tapes til varme, beregnes ved hjelp av effektloven.

$$P = \Delta U \cdot I \quad (5.14)$$

## 5.3 Kretskort

Kretskortene er av eget design og er et resultat av krav med tanke på vekt og størrelse. Utstyr tilgjengelig på markedet for å sette opp CAN bus kommunikasjon er for stort og tungt til bruk innenfor kravene til vår konfigurasjon.

Til design av kretskort ble følgende matematikk og teori tatt i bruk.

### 5.3.1 Sporbredde strømbaner

For å beregne nødvendig sporbredde til strømbanene i kretskortene ble en sporbredde-kalkulator fra Qorvo brukt [20]. Den bygger på grafer fra IPC-2221A standarden, utdrag vist i figur 8. Og fungerer på følgende måte:

Først beregnes arealet:

$$A = \left( \frac{I}{k \cdot (T_{rise})^b} \right)^{\frac{1}{c}} \quad (5.15)$$

Kretskortene som designes er tolags, med det behøves bare teori for eksterne lag mot luft. Dermed brukes konstantene for eksterne lag mot luft i tabell 2 [20, 21]. IPC-2221A standarden bruker Amerikanske/Engelske benevninger, og konstantene er basert på grafer fra standarden.

Arealet kommer derfor ut i kvadrat mil (1 mil = 0,001 tomme).  $I$  settes inn som amper og  $T$  som  $^{\circ}\text{C}$  i ligning (5.15).

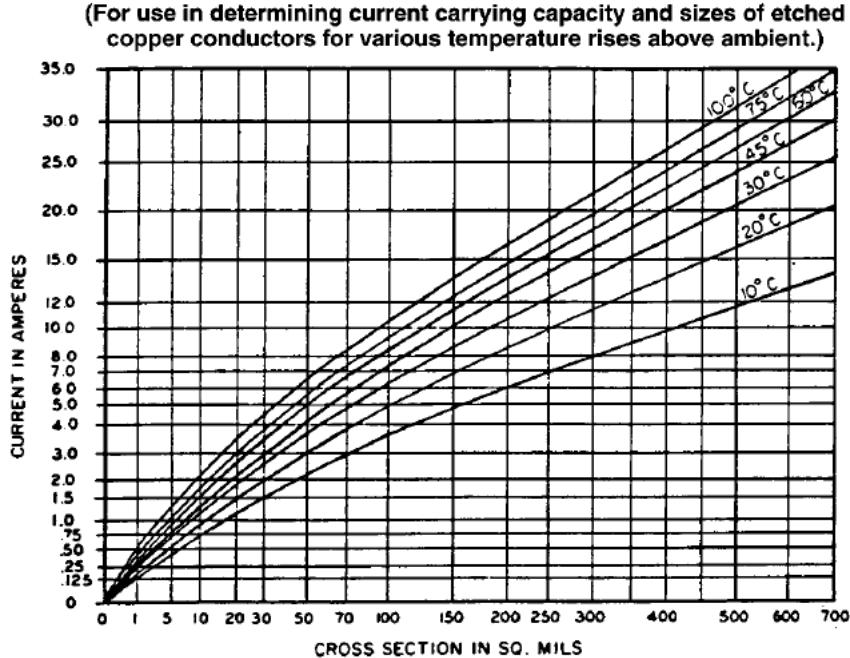
Tabell 2: Konstanter basert på grafer fra IPC-2221A

Konstant	Eksterne lag mot luft	Interne lag
k	0.048	0.024
b	0.44	0.44
c	0.725	0.725

Videre beregnes nødvendig bredde:

$$(\text{Sporbredde}) \quad W = \frac{A}{\text{Kobbertykkelse} \cdot 1,378} \quad (5.16)$$

Kobbertykkelse er gitt med vektbenevning, her som unse (oz). Den multipliseres med konstant 1,378 for å omgjøre fra vekt til tykkelse. 1,378 har benevning  $\frac{\text{mils}}{\text{oz}}$ . Dermed får vi ut nødvendig sporbredde i mil ved å løse ligning (5.16). Som er ok, da mil brukes i både designprogrammet Eagle og av PCB produsenter.

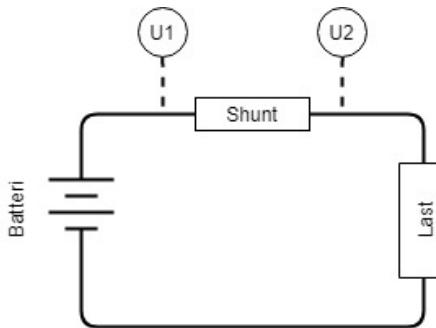


Figur 8: [21] Utdrag fra IPC-2221A - "External Conductors"

### 5.3.2 Beregning av strømforbruk

Kontroll på strømforbruks er viktig. For å måle strømforbruks brukes en shunt resistor i serie mellom batteri, og hele lasten til farkosten som vist i figur 9. Shunt resistoren bør ha en lav, men kjent resistans. Da kan Ohm's lov benyttes til å beregne strømmen, som i ligning 5.17, ved å måle spenningsfallet over shunt resistoren. Samtidig skal den ikke være en betydelig forbruker som tapper batteriet.

$$I_{shunt} = \frac{U_1 - U_2}{R_{shunt}} = \frac{\Delta U}{R_{shunt}} \quad (5.17)$$



Figur 9: Spenningsfall over shunt resistor

### Differensial forsterker

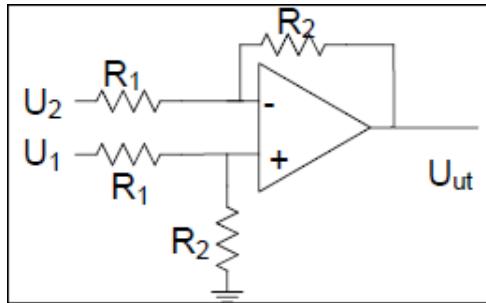
Spenningsfallet over shunt resistoren vil være veldig lite, grunnet dens lave motstand. Eksempelvis vil en 25A strøm gjennom en shunt med verdi  $0.001\Omega$  gi spenningsfall på 25mV

$$\Delta U = 0.001\Omega \cdot 25A = 25mV \quad (5.18)$$

For å effektivt ta i bruk store deler av oppløsningen til en 10-bits ADC vil det være lurt å benytte en OP-AMP til å lage en differensialforsterker som vist i figur 10. Det sees i ligning 5.19 at forsterkningen styres av forholdet mellom  $R_1$  og  $R_2$ . Resistor 1 og 2 må dermed velges med hensyn til hvilken oppløsning som skal benyttes for å lese av.

$$U_{ut} = \frac{R_2}{R_1} \cdot (U_1 - U_2) = \frac{R_2}{R_1} \cdot \Delta U \quad (5.19)$$

En differensialforsterker ser på spenningsene  $U_1$  og  $U_2$ . Grunnet operasjonsforsterkerenes egenskaper vil spenningen  $U_{ut}$  være spenningsdifferansen  $\Delta U$  mellom de to. Dette er meget gunstig, og tillater avlesning av forskjeller ved bruk av kun en pinne på en mikrokontroller. Det utelukker også utfordringen med at mikrokontrollere ofte kun kan lese mellom 0-5V på de analoge innngangene.



Figur 10: [22] Utdrag fra formelsamling MAS-218

## 5.4 Konfigurasjon av attiny mikrokontrollere

AVR har en rekke mikrokontrollere på markedet, hvor ATtiny er den serien med minst fysisk størrelse.

Konfigurasjon av mikrokontrolleren gjøres ved å modifisere bitverdier i bestemte register som er beskrevet i databladet.

Den viktigste oppgaven ATtiny skal utføre i denne rapporten er å generere et pulsbreddemodulert pulstog som skal sendes til motorenes ESC for regulering av turtall på motor. ATtiny85 sin 8-bit Timer 0 tilbyr flere moduser for å generere pulsbreddemodulert signal. Valgmulighetene er vist i figur 11 hvor det også er beskrevet krav til bit-verdier for WGM00, WGM01 og WGM02 i TCCR0A registeret [23].

**Table 11-5. Waveform Generation Mode Bit Description**

Mode	WGM 02	WGM 01	WGM 00	Timer/Counter Mode of Operation	TOP	Update of OCR <sub>x</sub> at	TOV Flag Set on
0	0	0	0	Normal	0xFF	Immediate	MAX <sup>(1)</sup>
1	0	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM <sup>(2)</sup>
2	0	1	0	CTC	OCRA	Immediate	MAX <sup>(1)</sup>
3	0	1	1	Fast PWM	0xFF	BOTTOM <sup>(2)</sup>	MAX <sup>(1)</sup>
4	1	0	0	Reserved	—	—	—
5	1	0	1	PWM, Phase Correct	OCRA	TOP	BOTTOM <sup>(2)</sup>
6	1	1	0	Reserved	—	—	—
7	1	1	1	Fast PWM	OCRA	BOTTOM <sup>(2)</sup>	TOP

Notes: 1. MAX = 0xFF

2. BOTTOM = 0x00

Figur 11: Oversikt over tilgjengelige pwm moduser for timer 0

"Clear Timer on Compare Match" (CTC) modus gir gode muligheter til å justere frekvens og duty cycle etter behov. CTC modus aktiveres ved å sette "WGM01" fra TCCR0A registeret lik "1" på bakgrunn av figur 11. Ved å bruke en interrupt trigget av timer 0, kan valgfri pinne på attiny85 velges som utgang for PWM.

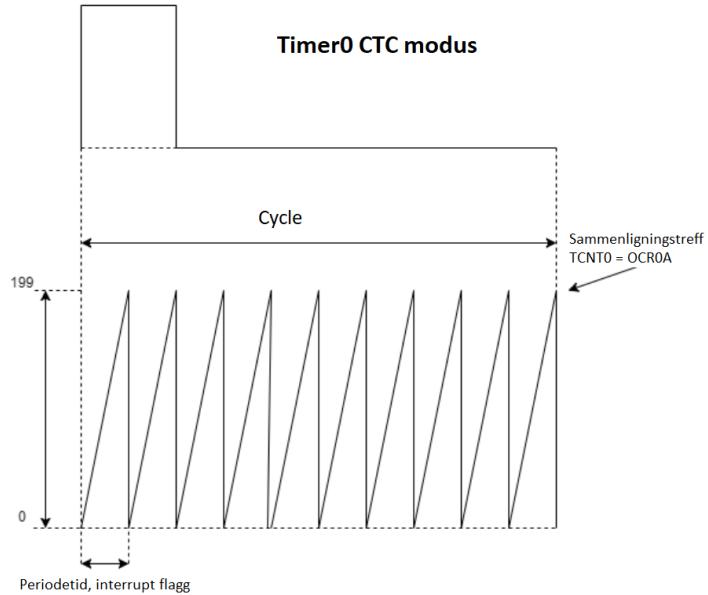
For å bruke "TIMER0\_COMPA\_vect" vektoren til å trigge interrupt, settes OCIE0A høy i TIMSK registeret. Vektoren trigges ved overflow på timer0.

TCNT0 er telleverdien til timer 0. I CTC modus resettes denne hver gang den får sammenligningstreff med OCR0A. Verdien på OCR0A kan justeres for å endre periodetiden på interrupt flagget. Om OCR0A settes til lavere verdi vil tiden mellom sammenligningstreff (TCNT0 = OCR0A) bli mindre, og frekvensen på interrupt flagg vil dermed øke.

Virkemåte for CTC modus er illustrert i figur 12 med  $OCR0A = 199$ . Hver gang interrupt trigges, kjøres en rutine som avgjør om PWM utgang skal være "0" eller "1" basert på ønsket duty cycle. I illustrasjonen er PWM utgangen lik "1" etter de to første sammenligningstreffene, og lik "0" de påfølgende 8 treffene. Dette resulterer i en duty cycle på 20 %.

Formel gitt av datablad for beregning av frekvens på interrupt flagg er vist i formel 5.20. Med utgangspunkt i denne kan verdi på  $OCR0A$  beregnes på bakgrunn av ønsket frekvens på interrupt flagg. Resulterende pulstog må samsvarer med ESC spesifikasjoner, der 0% pådrag =  $1000\mu s$  og 100% pådrag =  $2000\mu s$ .

Formel 5.21 viser mulig oppløsning i form av antall pådragsnivå, gitt av periodetiden på interrupt flagget( $T_{OC0A}$ ). Kortere periodetider muliggjør mer nyansert justering av hastighet.



Figur 12: Illustrasjon av CTC mode

$$f_{OC0A} = \frac{f_{clk\_I/O}}{N \cdot (1 + OCR0A)} \quad (5.20)$$

$$\text{Antall pådragsnivå} = \frac{\text{Max pulsbredde} - \text{Min pulsbredde}}{T_{OC0A}} \quad (5.21)$$

$f_{OC0A}$  = ønsket frekvens på interrupt flagg

$f_{clk\_I/O}$  = mikrokontrollerens klokkefrekvens

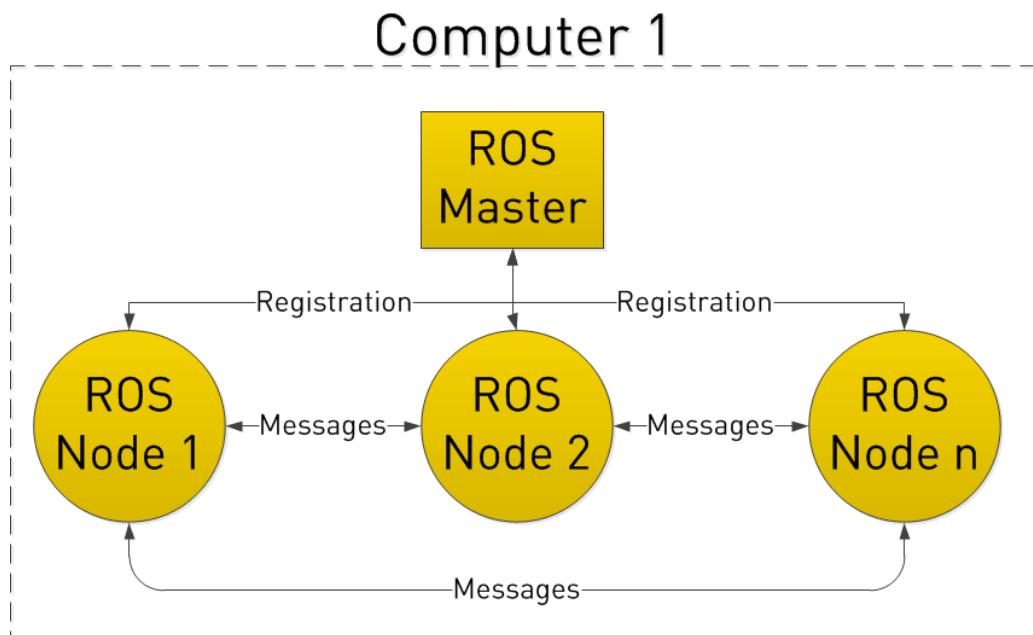
$N$  = Timer0 prescaler

$OCR0A$  = Verdi for sammenligningstreff

$T_{OC0A}$  = Periodetid interrupt flagg

## 5.5 Robot Operating System (ROS)

ROS er et nodebasert rammeverk for implementasjon av software. Noder skrevet i C++, python og generert fra Simulink kan enkelt kommunisere ved å publisere og lytte til "topics" som inneholder et sett med meldinger. ROS har innebygget funksjonalitet i Simulink ved hjelp av tillegspakken "Robotics system toolbox". Her finnes det blokker for å lytte og publisere på "topics", samt egne blokker for å kunne justere parametere i modellen ved hjelp av kommandoen "rosparam set". Dette muliggjør blant annet justering av PID kontrollere i ettertid, om Simulink allerede har generert ros noden. Virkemåte og kommunikasjon mellom ROS noder er illustert i figur 13



Figur 13: [24] Illustrasjon av ROS nettverk

## 5.6 Qualisys - motion capture

Motion capture system gir mulighet for bevegelses sporing av markører plassert i et område dekket av IR kamera. Markøren reflekterer IR lys fra kameraet som benyttes til å finne markørenes posisjonsvektorer i rommet. Teknologien er mye brukt i film- og spill bransjen så vel som innenfor ingeniørfag og sport.

Qualisys er leverandøren av Universitetets motion capture system i UiA MotionLab. Motion Lab UiA består av 17stk Oqus700+ kameraer [1].

## 5.7 Koordinatsystemer og referanserammer

For å kunne navigere, finne orientering og utelede dynamikk brukes ulike koordinatsystem/referanserammer (her også kalt aksesystem). Det finnes flere ulike typer med ulik oppbygning og anvendelses områder. Her vil vi gå gjennom systemene som er tilknyttet vårt bruksområde og som vi har anvendt gjennom oppgaven. Gjennom dette delkapittelet brukes referanse [25] (Unmanned Rotorcraft Systems kapittel 2) jevnlig. Følgende systemer er presentert:

1. Lokalt north-east-down (NED) koordinatsystem.
2. Lokalt egendefinert koordinatsystem.
3. Kroppssentrert north-east-down (NED) koordinatsystem.
4. Kroppsfast koordinatsystem.

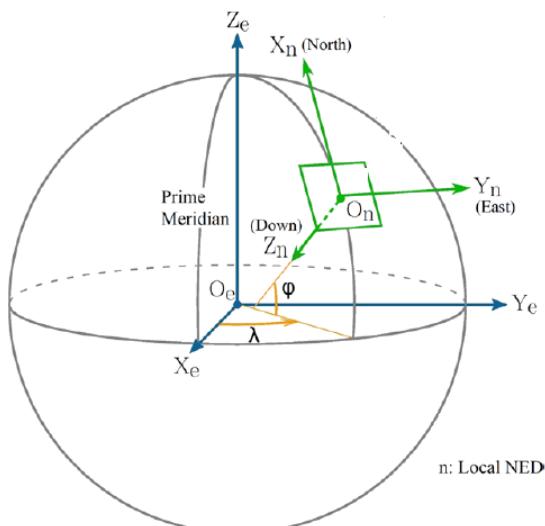
I hvert koordinatsystem vil det gjennom delkapittelet bli definert tre vektorer. Hvor  $P_\vartheta$  er posisjonen,  $V_\vartheta$  er hastigheten og  $a_\vartheta$  er akselrasjonen med hensyn på koordinatsystem  $\vartheta$ .  $\vartheta$  kan være "n,nv og b" som er notasjonen for henholdsvis lokalt NED, kroppssentrert NED og kroppsfast koordinatsystem.

### 5.7.1 Lokalt north-east-down (NED) koordinatsystem:

Aksesystemet er orientert på følgende måte (se også figur 14)

- Origo (notasjon  $O_n$ ) er et vilkårlig punkt festet til jordoverflaten.
- X-aksen (notasjon  $X_n$ ) peker mot jordas geografiske nordpol.
- Y-aksen (notasjon  $Y_n$ ) peker mot øst langs en tenkt linje parallelt med ekvator.
- Z-aksen (notasjon  $Z_n$ ) peker mot jordas senter, og står normalt på x- og y-aksen.

Dette er et kartesisk aksesystem som har sterkt tilknytning til navigasjon og styring av mindre UAV-er. Dette med grunn i at disse har begrenset rekkevidde, som vil si at jordkurvatur har liten betydning for navigasjonen [25].



Figur 14: [25] Lokalt NED koordinatsystemet i globalt perspektiv

### 5.7.2 Lokalt egendefinert koordinatsystem:

Ved navigering over svært små områder hvor man ikke er avhengig av GPS, men bruker andre løsninger som kjøretøysmontert eller eksternt monterte sensorsystemer, kan man navigere etter egendefinerte akser fastlagt i et vilkårlig punkt. I vårt tilfelle bruker vi MoCapS til å navigere, hvor koordinataksene bestemmes av brukeren.

For vår bruk legges z-aksen i retning mot jordsenteret slik at vårt koordinatsystem er identisk med et lokalt NED-system, sett bort i fra X- og Y-aksen ikke nødvendigvis sammenfaller med  $X_n$  og  $Y_n$ . Vi kan tenke oss at vi har en fiktiv nordpol med en tilhørende østlig retning. Dette er grunnlaget for at det videre i denne rapporten er brukt beskrivelsen "lokalt NED" eller "NED-rammen" selv om det egentlig er tenkt vårt lokalt egendefinerte koordinatsystem. Videre er det også bruk notasjonen fra lokalt NED-systemet, hvor vi har følgende vektorer:

$$P_n = \begin{bmatrix} X_n \\ Y_n \\ Z_n \end{bmatrix}, V_n = \begin{bmatrix} v_{x,n} \\ v_{y,n} \\ v_{z,n} \end{bmatrix}, a_n = \begin{bmatrix} a_{x,n} \\ a_{y,n} \\ a_{z,n} \end{bmatrix} \quad (5.22)$$

I forbindelse med oppsett av ligninger for dronens dynamikk blir også betegnelsen "den stillestående rammen" brukt. Her refereres det også til vår NED-ramme, selv om den prinsipielt sett ikke er en absolutt stillestående referanseramme. Det bemerkes at jordrotasjon har neglisjerbar effekt for vårt system. Det antas dermed at NED-rammen er en stillestående ramme.

### 5.7.3 Kroppssentrert north-east-down (NED) koordinatsystem

Det kroppssentrerte koordinatsystemet følger UAV-ens translasjon, med følgende akser og origo (se figur 15):

- Origó (notasjon  $O_{nv}$ ) plassert i UAV-ens massesenter.
- X-aksen (notasjon  $X_{nv}$ ) peker mot jordas geografiske nordpol.
- Y-aksen (notasjon  $Y_{nv}$ ) peker mot øst langs en tenkt linje parallelt med ekvator.
- Z-aksen (notasjon  $Z_{nv}$ ) peker mot jordas senter, og står normalt på x- og y-aksen.

Strengt tatt vil ikke aksernes retning fullstendig sammenfalle med akserne av det lokale NED-systemet, som følger av UAV-ens forflytning kombinert med jordkurvaturen. For vår bruk, innenfor et lite avgrenset område, gjør man en minimal feil ved å si at koordinatsystemene sammenfaller helt. Noe vi videre har antatt at koordinatsystemene gjør.

På samme måte som for det lokale NED koordinatsystemet er det også for det kroppssentrerte NED-systemet brukt en fiktiv nordpol. Dette er grunnlaget for at det videre i denne rapporten er brukt beskrivelsen «kroppssentrert NED» selv om det egentlig er tenkt UAV-ens koordinatsystem arrangert etter vårt egendefinerte lokale koordinatsystem.

I det kroppssentrerte koordinatsystemet har vi følgende vektorer:

$$P_{nv} = \begin{bmatrix} X_{nv} \\ Y_{nv} \\ Z_{nv} \end{bmatrix}, V_{nv} = \begin{bmatrix} v_{x,nv} \\ v_{y,nv} \\ v_{z,nv} \end{bmatrix}, a_{nv} = \begin{bmatrix} a_{x,nv} \\ a_{y,nv} \\ a_{z,nv} \end{bmatrix} \quad (5.23)$$

### 5.7.4 Kroppsfast koordinatsystem

Det kroppfaste koordinatsystemet (også kalt kroppsrammen) følger UAV-ens translasjon og rotasjon, og er fastlagt av følgende, se også figur 15:

- Origo (notasjon  $O_b$ ) plassert i UAV-ens massesenter.
- X-aksen (notasjon  $X_b$ ) peker forover langs dronens symmetriline i x-konfigurasjon.
- Y-aksen (notasjon  $Y_b$ ) peker fra origo mot UAV-ens styrbordside (sett ovenfra).
- Z-aksen (notasjon  $Z_b$ ) peker fra origo mot UAV-ens bunnpunkt, ortogonalt på X- og Y-aksen i henhold til høyrehåndsregelen.

I det kroppsfaste koordinatsystemet definerer vi følgende:

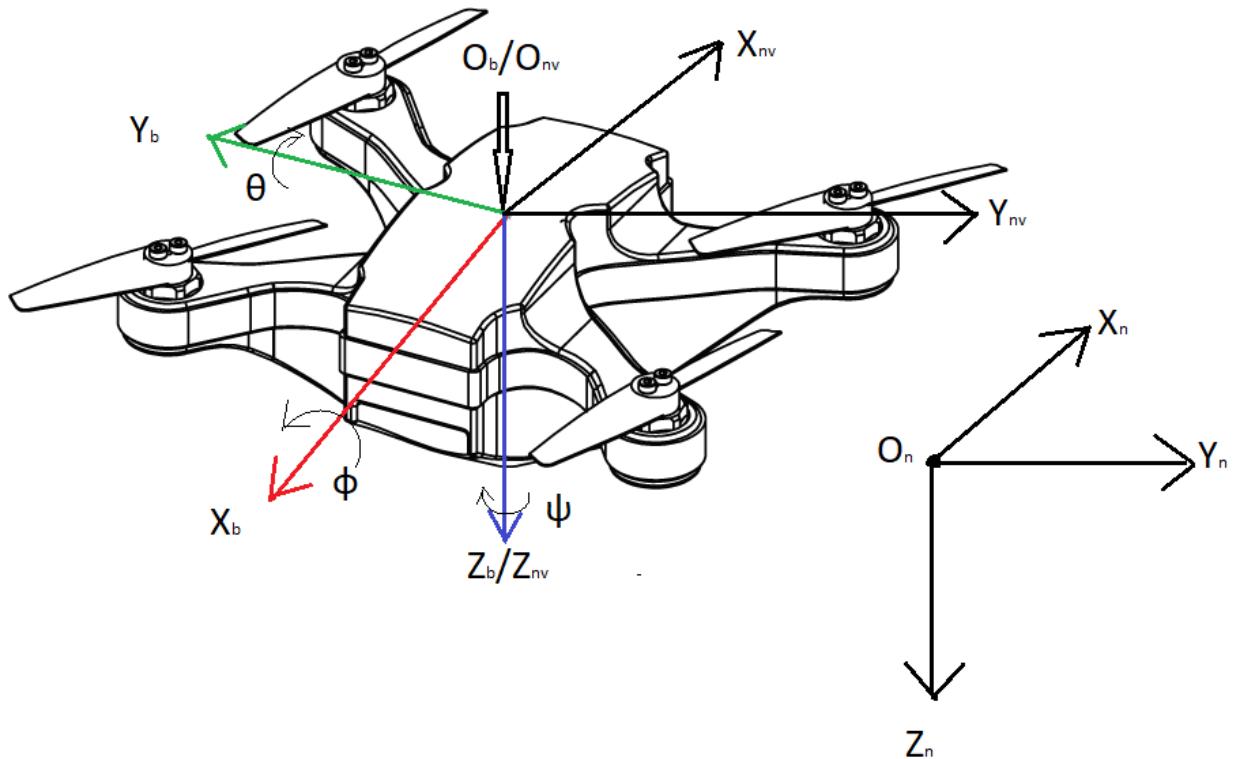
$$P_b = \begin{bmatrix} X_b \\ Y_b \\ Z_b \end{bmatrix}, V_b = \begin{bmatrix} v_{x,b} \\ v_{y,b} \\ v_{z,b} \end{bmatrix}, a_b = \begin{bmatrix} a_{x,b} \\ a_{y,b} \\ a_{z,b} \end{bmatrix} \quad (5.24)$$

I tillegg til translasjon vil også dette koordinatsystemet kunne rotere i forhold til den stillestående rammen. Rotasjonshastighetene om de ulike aksene i koordinatsystemet er definert:

- $p$  er rotasjonshastigheten om X-aksen.
- $q$  er rotasjonshastigheten om Y-aksen.
- $r$  er rotasjonshastigheten om Z-aksen.

Disse danner til sammen rotasjonshastighetsvektoren, hvor rotasjonshastigheten er gitt i forhold til den stillestående rammen:

$$\omega_b = \begin{bmatrix} p \\ q \\ r \end{bmatrix} \quad (5.25)$$



Figur 15: Kroppsfast, kroppssentrert NED og lokalt NED koordinatsystem

### 5.7.5 Oppsummering av koordinatsystemer og referanserrammer

Oppsummert er det valgt å bruke et kartesisk koordinatsystem, med referanseakser og origo som er fast i rommet (lokalt NED aksesystem). To aksesystem som er festet til dronekroppen, hvor det ene alltid står på linje med det lokale NED aksesystemet, mens det andre følger dronekroppegens rotasjon (kroppsfast). Det ses imidlertid at NED-rammens koordinataksler bestemmes av brukeren av systemet, det vil si at  $X_{n/vn}$  og  $Y_{n/vn}$  ikke nødvendigvis peker mot henholdsvis den geografiske nordpolen og parallelt med ekvator, men heller mot en fiktiv nordpol med en tilhørende østlig retning.

## 5.8 Orientering og rotasjon

For å kunne angi orientering i forhold til en referanseramme eller transformere fra et aksesystem til et annet, brukes ulike metoder. Blant disse er Euler-vinkler, rotasjonsmatriser og kvarternioner mye brukt. Eulervinkler og rotasjonsmatriser er kort forklart i denne delen, da disse er brukt til å danne grunnlaget for oppsettet av dynamikken til dronen.

### 5.8.1 Rotasjonsmatriser

Rotasjonsmatriser brukes for å endre orienteringen til en vektor i forhold til et koordinatsystem eller til å rottere selve koordinatsystemet. Ved å rottere koordinatsystemet og ikke vektoren kan vi finne lengde og retning på den samme vektoren sett i forhold til det nye koordinatsystemet [26]. Dette er en svært nyttig egenskap når vi ønsker å transformere mellom de ulike koordinatsystemene nevnt i 5.7.5.

En ekte ("proper") ortogonal rotasjonsmatrise er en  $n \times n$ -matrise med følgende egenskaper [27]:

- Determinanten av matisen er 1.
- $A^{-1} = A^T$ , hvor  $A^T$  er den transponerte av matrisen.
- $A^T A = I$ , hvor I er identitetsmatrisen.
- Produktet av to ortogonale matriser er en nye ortogonal matrise.
- Ved transformasjon bevares vektorlengdene og vinkel mellom dem [28], altså er det en ren rotasjon.

Det bemerkes at disse matrisene også har andre egenskaper, samt at det finnes andre typer rotasjonsmatriser. Det er i midlertid den overnevnte rotasjonsmatrisen som er brukt i dette prosjektet, spesielt på grunn av sistnevnte egenskapen.

### 5.8.2 Euler-vinkler

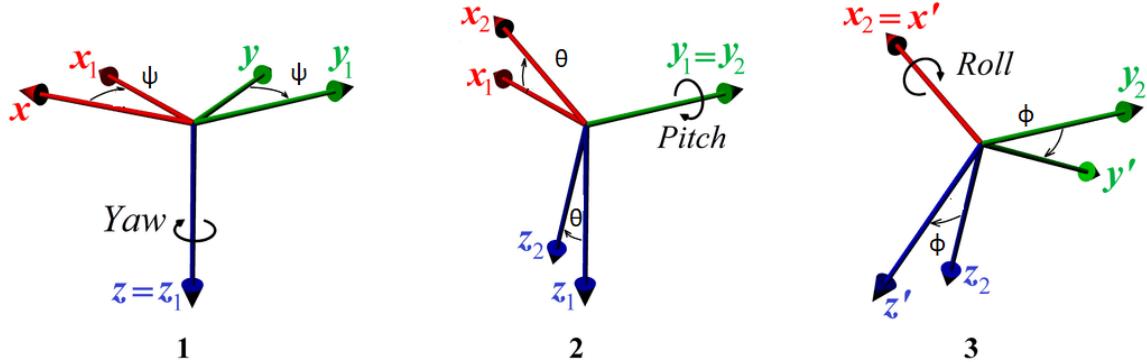
Eulervinkler er basert på at enhver orientering kan oppnås ved en sekvens av tre rotasjoner om en rammes 3 koordinataksjer. Her kan sekvensen bestå av 3 rotasjoner om to eller tre akser. Totalt er det 12 mulige sekvenser, hvor sekvenser bestående av rotasjon om to akser kalles ekte Euler-vinkler og rotasjon om tre akser kalles Tait-Bryan-vinkler [29].

Sekvensene er som følger:

- Ekte Euler-vinkler: (z-x-z, x-y-x, y-z-y, z-y-z, x-z-x, y-x-y).
- Tait-Bryan-vinkler: (x-y-z, y-z-x, z-x-y, x-z-y, z-y-x, y-x-z).

I denne rapporten vil z-y-x sekvensen (lest fra venstre mot høyre) bli brukt, dette med grunnlag i svakhetene Euler-vinkler har (diskutert senere i denne teksten under singularitet). Det skrives ofte Euler-vinkler selv om det strengt tatt refereres til Tait-Bryan-vinkler.

De ulike rotasjonene er definert  $\phi$ ,  $\theta$  og  $\psi$ , og er her beskrevet med utgangspunkt i figuren 16 og referanse [25].



Figur 16: [30] Rotasjonssekvens z-y-x

Tar vi for oss fire referanserammer i henhold til figuren over; *startrammen*, *ramme<sub>1</sub>*, *ramme<sub>2</sub>* og *sluttrammen'*. Kan  $\phi$ ,  $\theta$  og  $\psi$  forklares som følger:

- Yaw (notasjon  $\psi$ ) er rotasjonvinkelen fra *startrammen* til *ramme<sub>1</sub>*, denne rotasjonen går rundt *startrammens* z-akse. Denne vinkelen er den samme som vinkelen mellom *startrammens* x-akse og *sluttrammen'*s x-akse projisert ned i *startrammens* xy-plan.
- Pitch (notasjon  $\theta$ ) er rotasjonvinkelen fra *ramme<sub>1</sub>* til *ramme<sub>2</sub>*, denne rotasjonen går rundt *ramme<sub>1</sub>*'s y-akse. Denne vinkelen er den samme som vinkelen mellom *sluttrammen'*s x-akse og *startrammens* xy-plan.
- Roll (notasjon  $\phi$ ) er rotasjonvinkelen fra *ramme<sub>2</sub>* til *sluttrammen'*, denne rotasjonen går rundt *ramme<sub>2</sub>*'s x-akse.

Det bemerkes at alle koordinatsystemer her følger høyrehåndsreglene, og har positiv rotasjon mot klokken sett langs vektoren ned på origo.

### Singularitet

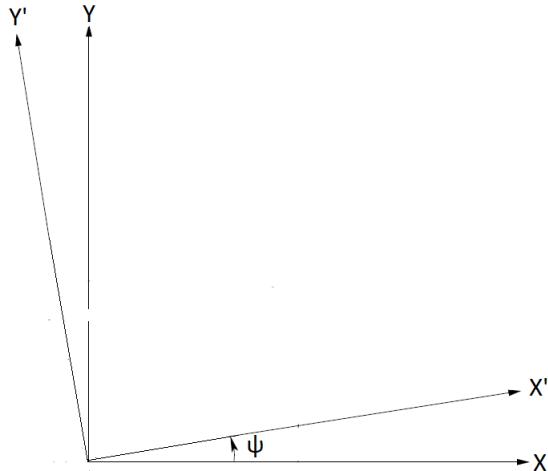
Selv om Euler-vinkler er en intuitiv måte å håndtere rotasjoner på, med minimalt antall parametere, har de allikevel en viktig begrensning. Denne begrensningen forekommer av singulære punkter hvor vi mister en frihetsgrad [31].

Bruker vi figur 16, hvor rotasjonssekvensen er z-y-x, vil vi ha et singulært punkt om vi "pitcher"  $\pm 90$  grader. Dette forekommer på følgende måte: Gjør først en vilkårlig rotasjon i Yaw (om *startrammens* z-akse), for så å "pitche"  $\pm 90$  grader ( $\theta = \pm 90$  grader). Som resultat av dette vil *ramme<sub>2</sub>*'s x-akse stå på linje med *startrammens* z-akse, som fører til at yaw- og roll-rotasjon går om samme akse. Vi har altså mistet en frihetsgrad.

Dette er et problem som det ikke er mulig å komme utenom ved bruk av Euler-vinkler, men det bemerkes at det finnes teknikker for å gå rundt problemet [32]. Fenomenet vil alltid oppstå i den andre rotasjonen i sekvensen, ved at rotasjonen fører til at første og siste rotasjonakse i sekvensen står på linje [31]. Man kan dermed unngå disse punktene i enkelte tilfeller, ved å velge en rekkefølge tilpasset sitt system. For dronen i dette prosjektet vil det aldri være nødvendig å "pitche"  $\pm 90$  grader, dette er da grunnlaget for å velge rotasjonsekvensen z-y-x.

## Utledningen av rotasjonsmatriser:

For den fullstendige utledningen er det brukt rotasjon om z-aksen. Det bemerkes at rotasjon om de to andre aksene framkommer ved bruk av samme framgangsmåte.



Figur 17: Rotasjon om z-aksen

Lengdene av vektorene i det nye koordinatsystemet er gitt av:

$$X' = X \cos(\psi) + Y \sin(\psi) \quad (5.26)$$

$$Y' = -X \sin(\psi) + Y \cos(\psi) \quad (5.27)$$

$$Z' = Z \quad (5.28)$$

På matriseform:

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \quad (5.29)$$

Fra dette ses det at rotasjonsmatrisen for rotasjon om Z-aksen er:

$$R_z = \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (5.30)$$

Ved bruk av samme fremgangsmåte framkommer rotasjonsmatrisene for henholdsvis rotasjon om y- og x-aksen:

$$R_y = \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \quad (5.31)$$

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{bmatrix} \quad (5.32)$$

For en rotasjonsmatrise vil den motsatte rotasjonen tilsvare den inverse matrisen. Dette kan brukes for å vise at  $A^{-1} = A^T$ . Om vi tar for oss rotasjon om z-aksen og roterer  $-\psi$  får vi:

$$R_z(-\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} = R_z(\psi)^T \quad (5.33)$$

### 5.8.3 Transformasjoner

I denne rapporten er de understående transformasjoner sammenlignet og bekreftet med andres resultater. Referanse [25] og [33] er hyppig brukt gjennom dette kapitellet.

#### Transformasjon mellom referanserammer

Den komplette transformasjonen mellom to referanserammene er en sammensetning av rotasjoner rundt 3 akser, som sammen danner en ny rotasjonsmatrice. Generelt for transformasjonene i denne rapporten vil det brukes bokstaven " $R$ " med to indekser, der den øvre viser til rammen det transformeres til, mens den nedre refererer til den rammen det transformeres fra.

Som tidligere nevnt vil det her brukes en z-y-x transformasjon. Matrisemultiplikasjonen foregår fra høyre mot venstre, slik at transformasjonen fra NED-rammen til kroppsrammen blir:

$$R_n^b = R_x R_y R_z = \begin{bmatrix} c(\theta)c(\psi) & c(\theta)s(\psi) & -s(\theta) \\ -c(\phi)s(\psi) + s(\phi)s(\theta)c(\psi) & c(\phi)c(\psi) + s(\phi)s(\theta)s(\psi) & s(\phi)c(\theta) \\ s(\phi)s(\psi) + c(\phi)s(\theta)c(\psi) & -s(\phi)c(\psi) + c(\phi)s(\theta)s(\psi) & c(\phi)c(\theta) \end{bmatrix} \quad (5.34)$$

Der  $R_x, R_y, R_z$  er rotasjonsmatrissene fra ligning 5.30, 5.31 og 5.32. Og "c" og "s" står for henholdsvis cosinus og sinus.

Som kjent er produktet av flere ortogonale matriser en ny ortogonal matrise, det vil si at den motsatte transformasjonen er den transponerte rotasjonsmatrisen. Tranformasjonen fra kropprammen til NED-rammen bli da som følger:

$$R_b^n = (R_b^n)^T \quad (5.35)$$

Strengt tatt kunne man her brukt den kroppssentrerte NED-rammen i stedet for den stillestående NED-rammen, men det er av ingen betydning da det kun translasjon som skiller dem.

Det legges merke til at om man skal gjøre transformasjonen fra kroppsrammen til NED-rammen sekvensvis, vil sekvensen bli x-y-z med motsatte/reverserte Euler-vinkler.

### Transformasjon av rotasjonshastigheter

I forbindele med den matematiske modellen er det nødvendig å ha en transformasjon fra rotasjonshastighetene i kroppsrammen til Euler-ratene i NED-rammen. For å komme fram til dette ser vi på rotasjonshastighetene i kroppsrammen gitt av Euler-ratene i NED-rammen.

Det bemerkes her at vektoren bestående av Euler-vinkler ikke er en ortogonal vektor, da rotasjonsaksene ikke står ortogonalt på hverandre. Det impliserer videre at Euler-rate-vektoren heller ikke er ortogonal [34], det medfører:

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} \neq \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \quad (5.36)$$

For denne forklaringen tar vi igjen utgangspunkt i figur 16. Hvor *startrammen* nå er NED-rammen, *sluttrammen'* er kroppsrammen, samt har vi to midlertidige rammer.

Den siste rotasjonen i transformasjonen fra NED-rammen til kroppsrammen går om  $x_b$ . Det vil si at roll-raten (endring av roll vinkel) gir et direkte bidrag til rotasjonshastighetsvektoren:

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix}_{\dot{\phi}} = \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} \quad (5.37)$$

Rotasjonshastigheten gitt av pitch-raten er mer komplisert. Pitch-raten er endringen av den andre rotasjonen vi gjorde i transformasjonen  $R_n^b$ . Pitch-raten må dermed transformeres til kroppsrammen fra *ramme*<sub>2</sub> for å gi riktig bidrag:

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix}_{\dot{\theta}} = R_2^b \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} \quad (5.38)$$

På samme måte er yaw-raten endringen av den første rotasjonen vi gjorde (med hensyn på tid), slik at denne må transformeres fra *ramme*<sub>1</sub> til kroppsrammen for å få riktig bidrag:

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix}_{\dot{\psi}} = R_2^b R_1^2 \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} \quad (5.39)$$

I disse ligningene er  $R_2^b = R_x$  fra ligning 5.32 og  $R_1^2 = R_y$  fra ligning 5.31

Sluttresultatet blir da som følger:

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + R_2^b \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + R_2^b R_1^2 \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\sin(\theta) \\ 0 & \cos(\phi) & \cos(\theta)\sin(\phi) \\ 0 & -\sin(\phi) & \cos(\theta)\cos(\phi) \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \quad (5.40)$$

For å finne transformasjonen fra rotasjonshastighet i kroppsrammen til Euler-rater i NED-rammen  $(R_\omega)_b^E$  må vi finne den inverse av rotasjonsmatrisen i ligningen over. Denne matrisen er ikke ortogonal ( $A^{-1} \neq A^T$ ) og vi må dermed ta den inverse på "vanlig" måte. Resultatet blir som følger:

$$(R_\omega)_b^E = ((R_\omega)_E^b)^{-1} = \begin{bmatrix} 1 & 0 & -\sin(\theta) \\ 0 & \cos(\phi) & \cos(\theta)\sin(\phi) \\ 0 & -\sin(\phi) & \cos(\theta)\cos(\phi) \end{bmatrix}^{-1} \quad (5.41)$$

$$(R_\omega)_b^E = \begin{bmatrix} 1 & \sin(\phi)\tan(\theta) & \cos(\phi)\tan(\theta) \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \frac{\sin(\phi)}{\cos(\theta)} & \frac{\cos(\phi)}{\cos(\theta)} \end{bmatrix} \quad (5.42)$$

Det legges merke til at i denne transformasjonen viser matematikken de singularitetene Euler-vinkler har. Setter vi  $\theta = \pm 90\text{grader}$  vil løsningen av matrisen være udefinert.

## 5.9 Dynamikk

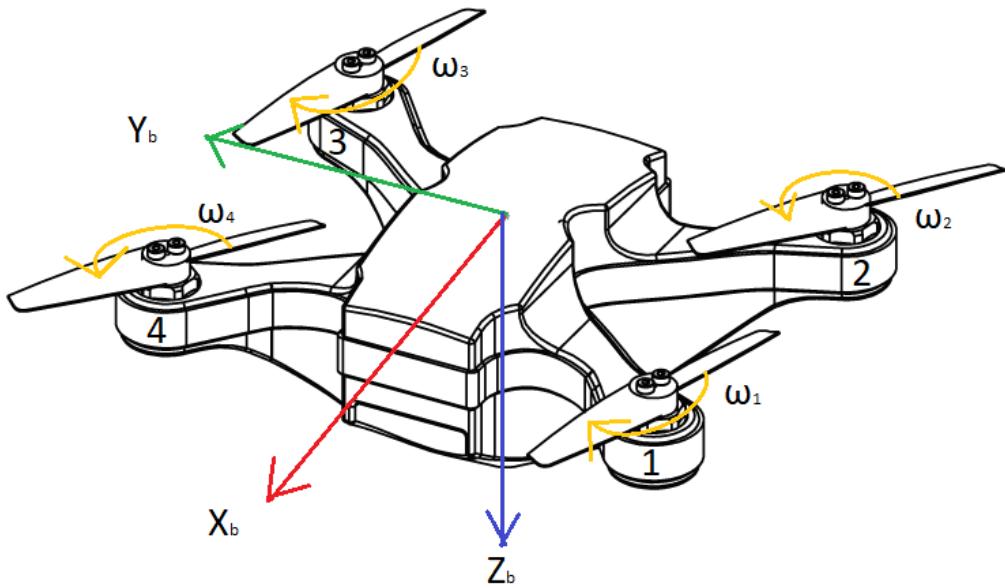
I dette kapittelet presenteres ligninger og sammenhenger som er vitale ved bygging av en matematisk dronemodell. Det er også lagt fram en konfigurasjon av dronen, med motor nummerering og dreieretning, denne konfigurasjonen gjelder gjennom hele denne rapporten (med mindre annet er spesifisert).

### 5.9.1 Dronens konfigurasjon

I dette prosjektet skal dronen flys i en x-konfigurasjon. Motor 1 til 4 defineres i henhold til figur 18.

Dronens motorer vil rotere som vist i figur 18. Sett ovenfra og ned langs positiv  $Z_b$  vil rotasjonen være som følger:

- motor1 = medurs (CW).
- motor2 = moturs (CCW).
- motor3 = medurs (CW).
- motor4 = moturs (CCW).



Figur 18: Motorplassering og rotasjonsretning

### 5.9.2 Roterende referanserammer

I dynamikk blir parametere gitt i en referanseramme ofte brukt til å beskrive oppførselen i andre rammer. I forbindelse med dette er det her lagt fram en kort beskrivelse av matematikken bak, følgende er en kort gjengivelse basert på utledninger gitt i referanse [35] og [36] kapittel 12.

Om vi tar for oss tre ortogonale vektorer som står fast i en referanseramme  $R_1$  som roterer med en vinkelhastighet  $\Omega$  (kan også transelere) i forhold til en globalt stillestående ramme  $R_2$ . Vektorene som står fast i  $R_1$  består av henholdsvis enhetsvektorene  $i$ ,  $j$  og  $k$ , og hvor lengden av vektorene er produktet av enhetsvektorene og konstantene  $c_{1,2,3}$ , får vi følgende:

$$C = ic_1 + jc_2 + kc_3 \quad (5.43)$$

Vi observerer nå endringen av  $C$  fra  $R_2$ . Her kan lengden av vektorene  $c_{1,2,3}$  kunne endres med tiden, men samtidig vil også vektorenes retning endres, altså retningen av enhetsvektorene. I henhold til derivasjonsreglene får vi:

$$\frac{d}{dt}C = \frac{d}{dt}i + \frac{d}{dt}c_1 + \frac{d}{dt}j + \frac{d}{dt}c_2 + \frac{d}{dt}k + \frac{d}{dt}c_3 \quad (5.44)$$

Om vi nå tenker på enhetsvektoren (konstant lengde) "i" som en posisjonsvektor fra  $R_1$  sitt origo til en partikkkel, ser vi at vi kan skrive endringen av "i" som den tangensielle hastigheten til partikkelen som da er gitt av:

$$\frac{d}{dt} i = vinkelhastigheten \times posisjonen = \Omega \times i \quad (5.45)$$

Dette kan gjøres på samme vis for de to resterende enhetsvektorene også. Dette fører da fram til følgende løsning i hendold til referanse [36]:

$$\frac{d}{dt} C = \frac{\partial}{\partial t} C + \Omega \times C \quad (5.46)$$

Hvor endringen av  $C$  sett fra  $R_2$  kan deles i to ledd, hvor første ledd er endringen av  $C$  som følger av endringen av vektorenes lengde  $c_{1,2,3}$  lokalt i  $R_1$ , og andre ledd er endringen av  $C$  som følger av endringen av vektorenes orientering i forhold til  $R_2$ .

For å gjøre denne ligningen mer lettlest er det videre brukt følgende notasjon:

$$\frac{dC}{dt}_{R_2} = \frac{dC}{dt}_{R_1} + \Omega \times C \quad (5.47)$$

Det bemerkes her at selv om vi nå har et uttrykk for endringen av  $C$  observert fra den stillestående rammen  $R_2$ , er dette ikke en geometrisk transformasjon. Avhengig av hvilken ramme man definerer vektorene innenfor, kan det være nødvendig å videre transformere  $C$  mellom  $R_1$  og  $R_2$ . Den overstående formelen konverterer vektorer fra en roterende referanseramme til en stillestående referanseramme.

### 5.9.3 Treghetsmoment og akser

Innen rotasjonsdynamikk har plassering og orientering av koordinatsystem betydning for ligningenes kompleksitet. Her er sammenhengen mellom treghetmoment-matrisen og koordinatsystemets akser viktig.

#### Treghetsmoment-matrisen

Tredimensjonal rotasjonsdynamikk regnes ofte i form av vektorer og matriser.

$$M = I\dot{\omega} \quad (5.48)$$

Hvor  $M$  er momentvektoren,  $\dot{\omega}$  er vinkelakselrasjonsvektoren og  $I$  er treghetsmoment-matrisen. Sistnevnte er en  $3 \times 3$ -matrise bestående av massetreghetsmomentene med hensyn på det aktuelle koordinatsystemet. Det kan bevises at matrisen har følgende oppbygning [35]:

$$I = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{yz} & I_{yy} & -I_{yz} \\ -I_{zx} & -I_{zy} & I_{zz} \end{bmatrix} \quad (5.49)$$

Hvor:

$$I_{xx} = \int (y^2 + z^2) dm \quad (5.50)$$

$$I_{yy} = \int (z^2 + x^2) dm \quad (5.51)$$

$$I_{zz} = \int (x^2 + y^2) dm \quad (5.52)$$

$$I_{xy} = I_{yx} = \int (xy) dm \quad (5.53)$$

$$I_{xz} = I_{zx} = \int (xz) dm \quad (5.54)$$

$$I_{yz} = I_{zy} = \int (yz) dm \quad (5.55)$$

Matrisen beskriver altså hvordan massen til et stivt legeme er fordelt med hensyn på det valgte koordinatsystemets plassering og orientering. Denne matrisen er gyldig for aksesystemer som står fast i legemet med origo i massesenteret. Det finnes også andre gyldighetstilfeller, men de er mindre aktuelle for dette prosjektet. For videre detaljer om utledning og egenskaper vises det videre til referanse [35].

## Prinsipielle akser

Kombinerer vi ligning 5.48 og 5.49, og skriver resultatet ut får vi:

$$\begin{bmatrix} M_x \\ M_y \\ M_z \end{bmatrix} = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{yz} & I_{yy} & -I_{yz} \\ -I_{zx} & -I_{zy} & I_{zz} \end{bmatrix} \begin{bmatrix} \dot{\omega}_x \\ \dot{\omega}_y \\ \dot{\omega}_z \end{bmatrix} \quad (5.56)$$

Fra ligningen over ser vi at for enkelte valg av koordinatsystem vil vinkelakselasjonen om to akser kunne gi et moment rundt den tredje aksen. Det ses samtidig at ligning 5.53, 5.54 og 5.55 er sterk knyttet til symmetri. Er massen symmetrisk fordelt vil disse leddene i matrisen bli null.

Det kan vises at for et vilkårlig punkt på et stift legeme vil det alltid finnes en eller flere orienteringer av koordinatsystemet, slik at bidraget fra ligning 5.53, 5.54 og 5.55 forsvinner og trehetsmoment-matrisen blir diagonal. Disse koordinataksene kalles prinsipielle akser [35]. Når koordinatene er prinsipielle akser blir matrisen som følger:

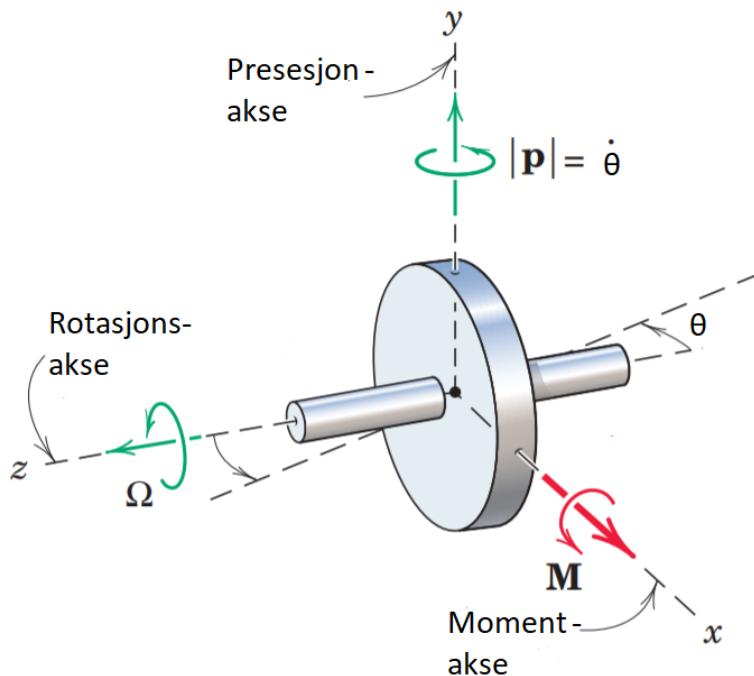
$$I = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \quad (5.57)$$

Dette er en egenskap som gjør matematiske utregninger enklere, samtidig som man kan dekoble ligningen for momentet om x, y og z (fullstendig eller i noen grad, avhengig av system). Svært fordelaktig med tanke på kompleksiteten og koblingene i et system.

Dette er egenskaper det er viktig å være klar over og ta hensyn til ved utforming av dronen og plassering av komponenter/masse.

### 5.9.4 Gyroskopisk momentbidrag

Gyroskopisk bevegelse/presesjon er et fenomen som oppstår når aksen et roterende legeme roterer om utsettes for et moment, hvor momentvektoren ikke står parallelt med aksen det roteres om. Det påførte momentet fører til at rotasjonsaksen også begynner å rotere som vist i figur 19.



Figur 19: [35] Illustrasjon av gyroskopisk bevegelse

Her roterer legemet med en vinkelhastighet  $\Omega$  om z-aksen. Når det nå påføres et moment  $M$  om x-aksen, vil rotasjonsaksen z, begynne å rotere om y-aksen med en vinkelhastighet  $p$ .

I det motsatte tilfellet, hvor vi tvinger en rotasjonsakse (z-aksen på figuren) til å rotere/endre vinkelposisjon om en annen akse (y-aksen på figuren), ser vi at det oppstår et moment. Dette er i henhold til følgende formel, vist i referanse [35], (kapittel 7):

$$M = p \times H \quad (5.58)$$

$$H = I\Omega \quad (5.59)$$

Hvor  $I$  er treghetsmatrisen og  $H$  er det angulære momentet gitt av  $I\Omega$ .

I overstående formler er det antatt at vinkelhastigheten  $p$  er vesentlig mindre enn vinkelhastigheten  $\Omega$ , slik at det angulære momentet gitt av den saktegående presesjonen kan neglisjeres. Det er også antatt at endringen i det angulære momentet er neglisjerbar, slik at første ledd av den generelle ligningen 5.47 faller bort og vi får ligningen over [35].

## 5.10 Regulering

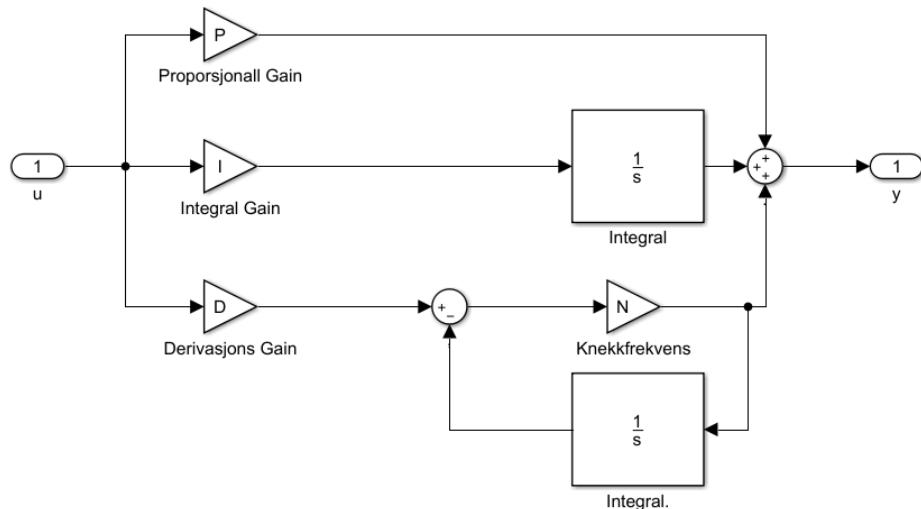
### Filtrering av PID-kontroller pådrag

I reguleringssammenheng brukes derivasjonsdelen av en PID-kontroller til å redusere oversving og innsvingnings-tiden. Det som imidlertid må hensyntas i denne sammenheng er følsomhet for støy og signaler med "skarpe kanter".

I en praktisk tilbakekobling med sensorer vil det alltid oppstå støy. Dette er typisk høyfrekvent støy som tilsier at den deriverte av signalet blir stor. Som videre fører til at vi får et "for stort" bidrag fra derivasjonsdelen av PID-kontrolleren [37].

Ved input med "skarpe kanter", eksempelvis step input, vil også den deriverte være stor, og bidraget fra derivasjonsdelen kan bli så stort at det har uønsket virkning i reguleringen [37].

I tilfeller der det er utelukket å kutte ut derivasjonsdelen, blir det ofte brukt et førsteordens lavpassfilter for å motvirke disse bivirkningene, med spesiell hensyn på støy. PID-kontroller med et slikt filter er vist i figur 20, med tilhørende ligning 5.60 hvor N representerer filterets knekkfrekvens i rad/s.



Figur 20: PID-kontroller med filtrert derivasjonsdel

$$\frac{y}{u} = P + I \frac{1}{s} + D s \frac{N}{s + N} \quad (5.60)$$

## 5.11 Motortest

### Motor

Test av motor er viktig for å kunne ta gode avgjørelser med hensyn til valg av motor og propell. Ved kjøp av motor oppgis ikke thrust, da den er avhengig av hvilken propell som velges. Stigning (pitch) og størrelse (diameter) på propell er parametere som spiller inn, og har stor betydning for dronens karakteristikk i luften. Dronens arbeidsoppgaver har også stor betydning for valg av motor og propell. Dronen vår skal utføre oppgaver i et rolig tempo. Dermed kan det sies at flyvetiden hovedsakelig vil bestå av hovring. Konfigurasjon av propell og motor kan dermed velges med hensyn til at dronen skal hovre på det turtall motoren trekker minst strøm. Dette er ønskelig for å oppnå lengst mulig flytid.

Samtidig er det ønskelig å utføre videre tester på valgt motor for å kartlegge styrke, linearitet, effektivitet, og moment.

### Propell

Ved avlesning av propell spesifikasjoner er det to format som brukes av produsentene:  $L \times P \times B$  eller  $LLPP \times B$ , hvor L er lengde, P er pitch og B er antall blad. I denne rapporten brukes sistnevnte. En "6045 × 2" har 6 tommer diameter, 4.5 tommer pitch og 2 blad. 4.5 tommer pitch betyr at om propellen teoretisk sett sammenlignes med en skrue, ville den skrudd seg 4.5 tommer innover på en runde.

### Test

Alle tester er forsøkt utført under samme forhold. Under hver test blir spenning og måling av strømtrekk notert.

God avstand mot vegg bak propell for å unngå metning, såkalt "ground effect" som kan forstyrre testresultatet.

Rigg for test er forsøkt bygget for å ha minimal friksjon, for å kunne neglisjere friksjon i kalkulasjoner. Utforming på testriggens arm er valgt med tanke på god luftstrøm forbi arm. Sirkulært tverrsnitt, ingen skarpe kanter. Samtidig i lignende dimensjon som forventes å ende opp med på dronen.

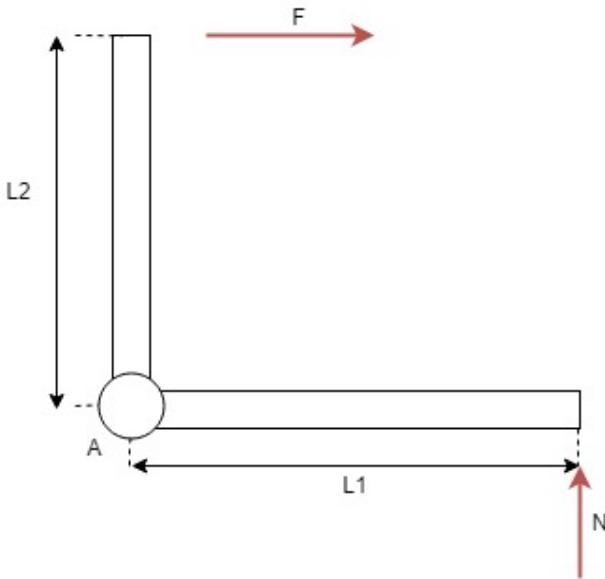
#### 5.11.1 Mekanikk

For å måle thrust, til de innkjøpte motorer og propeller ble en rigg laget. FLD for testrigg er vist i figur 21. Normalkraften  $N$  måles på en vekt, og thrustkraften  $F$  løses med ligning 5.62.

$$\sum^+ M_A = 0 \text{Nm} = L_1 \cdot N - L_2 \cdot F \quad (5.61)$$

$$F = \frac{L_1}{L_2} \cdot N \quad (5.62)$$

Det sees at forholdet mellom avlest og faktisk thrust er gitt av  $\frac{L_1}{L_2}$

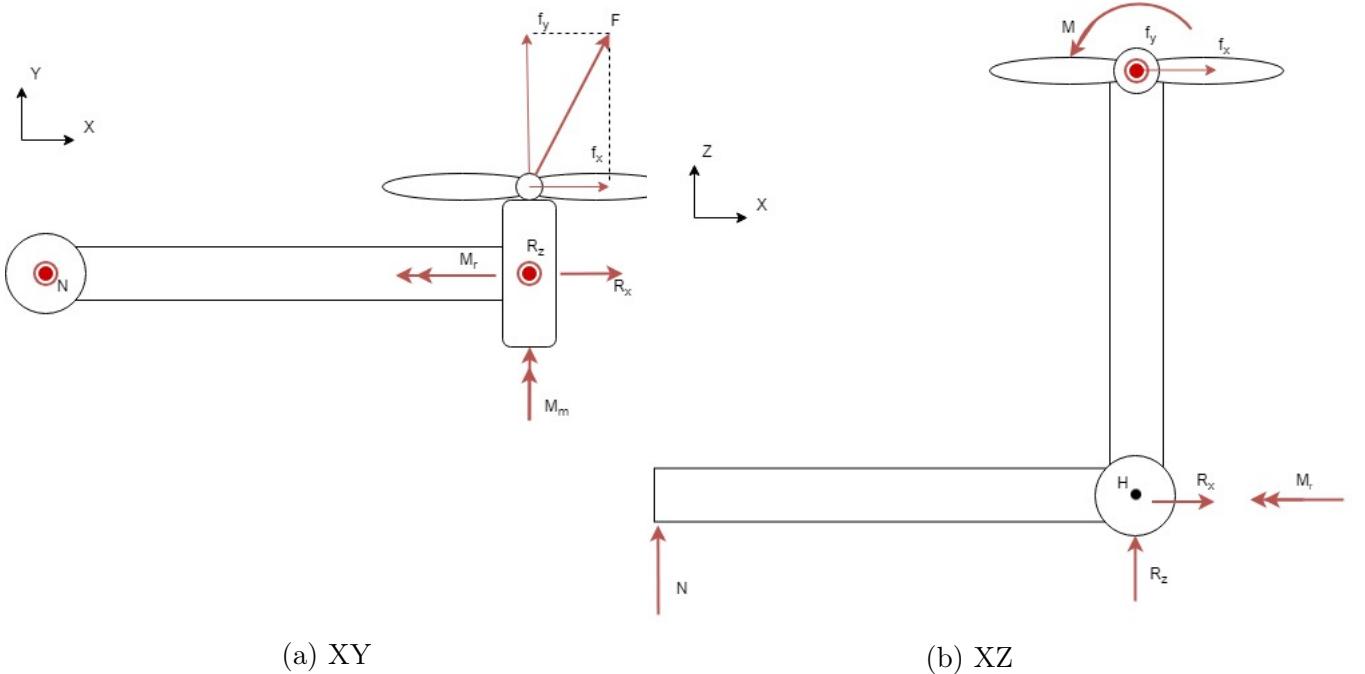


Figur 21: FLD - Motortest thrust

#### Test for å finne motorens vridningsmoment:

For å kunne regulere jaw korrekt trengs data på vridningsmoment generert fra motor. Første test ble rigget som vist i figur 22, den ga varierende resultat og dårlig repeterbarhet. Det var utfordrene å montere motoren som tegnet på figur, normalt mot riggen som var laget, altså parallelt med  $y$ -aksen. Derav oppstår en kraftkomponent  $F$ , med en vinkel ut fra  $y$ -akse, overdrevet vist i figur 22a. Ved å dekomponere  $F$  til  $f_x$  og  $f_y$ , sees det i figur 22b at  $f_x$  vil forstyrre normalkraften  $N$ . Som brukes for å lese av på vekt og løse for  $M$ . Ligninger for å finne  $M$  er ikke satt opp, figur 22 er laget for å illustrere kompleksiteten ved denne testen.

Det registreres også at  $f_y$  skaper et reaksjonsmoment  $M_r$  om hengslepunkt  $H$ . Det er uheldig, da det bidrar til å betydeliggjøre friksjon i hengslepunktet  $H$ , som også vil forstyrre  $N$ .

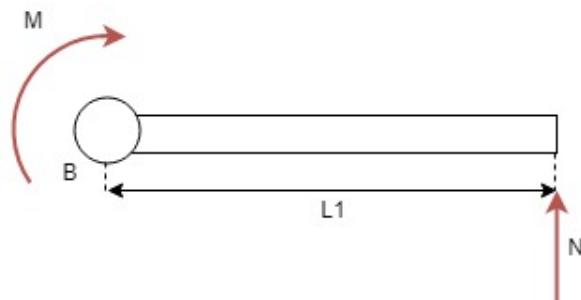


Figur 22: FLD, første test av vridningsmoment

For å unngå forstyrrelse fra andre kraft-komponenter, ble det valgt å lage en ny rigg som muliggjør mer direkte avlesning av momentet propellen genererer. Revidert utgave ble laget, FLD vist i figur 23. Her påføres momentet i samme akse som opplagring, og utelukker dermed bidraget fra andre kraftkomponenter som kan påvirke vridningsmomentet vi forsøker å måle. Momentet kan dermed noteres enkelt ved å lese av  $N$  på en vekt og løse ligning 5.64.

$$\sum^+ M_B = 0 \text{ Nm} = L_1 \cdot N - M \quad (5.63)$$

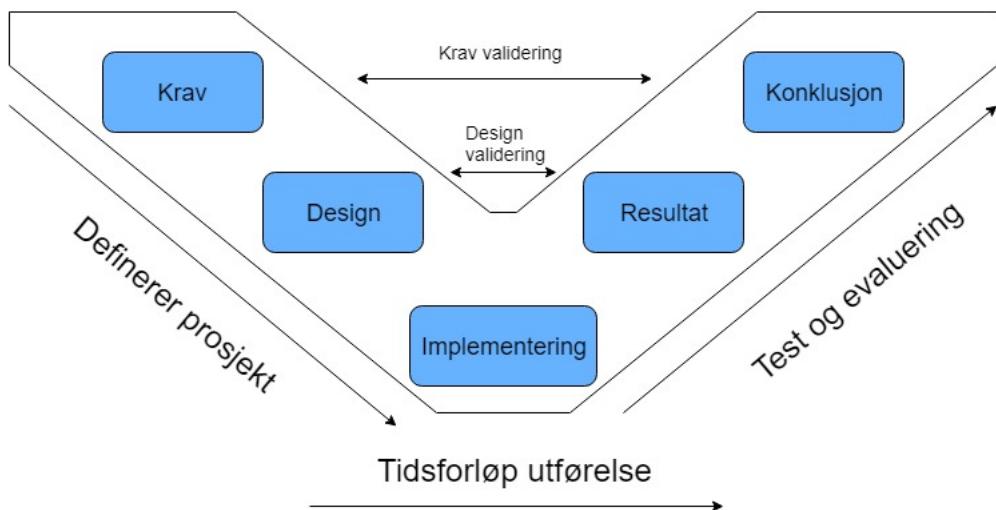
$$M = L_1 \cdot N \quad (5.64)$$



Figur 23: Motortest vridningsmoment

## 6 Metode

I prosjektet har vi benyttet oss av strategien som ligger bak V-modell prinsippet. Dette er en modell som skal være til hjelp for å jobbe systematisk gjennom et prosjekt. Den starter med fasene der en tar for seg konsept og teori, som videre utvikler seg til en designfase. Når konsept og design er fastsatt, vil det bli praktisert i en implementeringsfase der en utfører det som er blitt planlagt i de tidligere fasene i henhold til krav som er satt. Det vil så bli utført test av det som er blitt utarbeidet i implementeringsfasen, slik at en kan komme frem til et resultat og trekke en konklusjon ut fra det. En kan så sammenligne og validere hvordan krav, design, resultat og konklusjon henger sammen og hvordan sluttproduktet blir. Grafisk fremstilling av V-modellen er vist i figur 24.



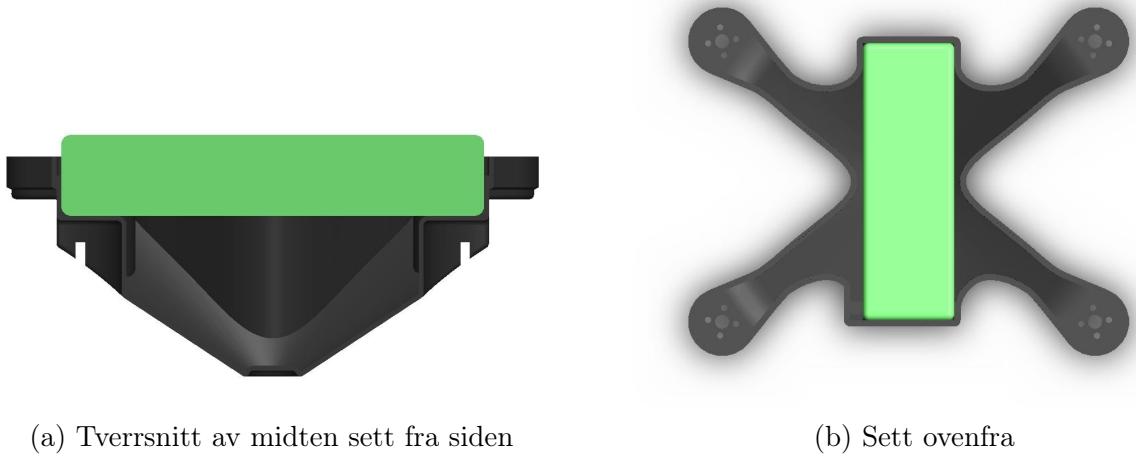
Figur 24: V-modell

### 6.1 Skrog

Ettersom oppgaven krever at dronen skal ha en totalmasse under 250 gram, ble designet av skroget en utfordring. Designet må også ta høyde for gunstige plasseringer av markørene benyttet av MoCapS. I tillegg skal den tilpasses en ladestasjonen der den kan lande og lade automatisk. Ut fra en oppsummering av massen til komponenter som blir benyttet i dronen, vist i tabell 3 (inkl valgt batteri), ble det satt av en masse på 40 gram til skroget.

Mange forskjellige design ble vurdert, der fokuset var å kunne "pakke inn" komponentene inni skroget, slik at det beskytter komponentene. På grunn av vektkravet måtte den bygges så kompakt som mulig med minst mulig materiale. Til slutt ble et endelig design funnet, bestående av to deler; toppdel og bunndel. Sammensatt utgjør dette et selvbærende skrog, som er stift i forhold til tykkelsen.

Batteriet utgjør en betydelig del av dronens masse, og er med på å dominere dronens tregheitsmoment. Med hensyn til tyngdepunktet ble det derfor naturlig å plassere batteriet i senter av dronen. Samtidig er batteriet kompakt og stift og kan nytties som en bærende del av skroget. Skroget ble dermed utformet rundt batteriet for å kunne utnytte batteriets stivhet og størrelse ved påkjenninger. I figur 25a og 25b ser vi at skroget er bygd inn mot batteriet.

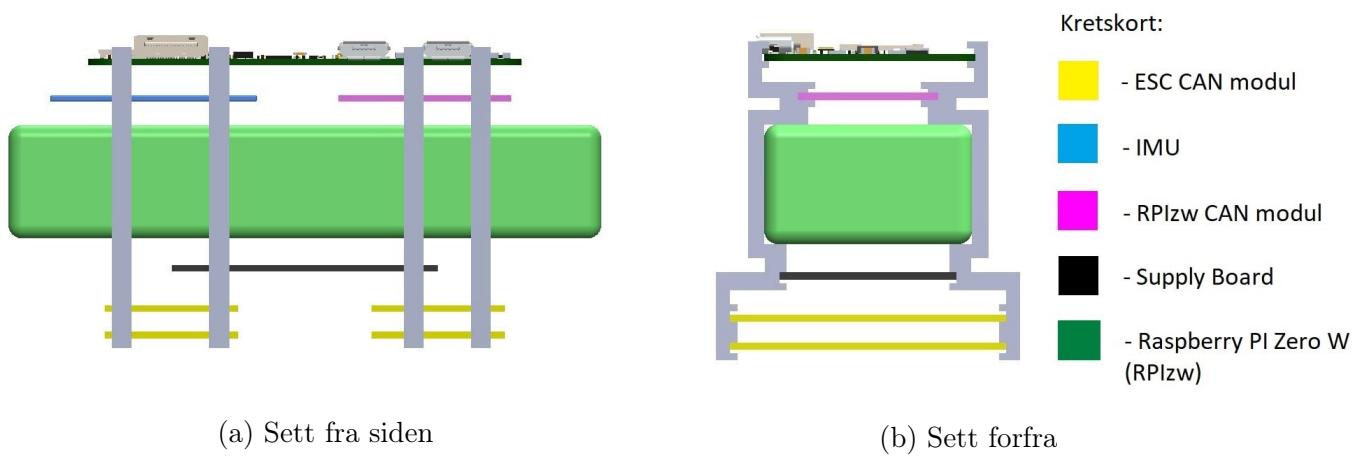


Figur 25: Bunndel utformet av med hensyn til batteri

Ut fra testprint i 5.1.2, ble det valgt veggykkelse 1mm for skroget. Hele skroget holder denne veggykkelsen for å kunne masseporduseres av plate med en tykkelse med bruk av vakuum-forming. Det ble utført FEM-analyse for å se om dette kunne være aktuelle dimensjoner å produsere skroget med. Simuleringer ble utført med hensyn til faktorene materiale, tykkelse og utforming. Resultater av analysen er vist i 6.2.1 og 6.2.2. Det vises ut fra disse resultatene at valgene innen nevnte faktorer gir gode egenskaper innen stivhet, vekt og styrke.

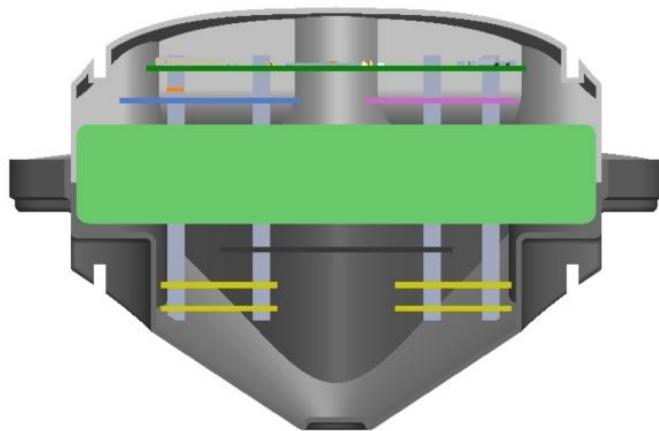
Området som er mest utsatt under flyvning er armene. De ble derfor designet for å ha høy stivhet i vertikal retning. Bakdelen med dette designet og at skroget er tynt, var at armene var svake for påkjenninger ved vridning/torsjon. Dette ble observert ved å bevege armene på printet ramme. For å kunne øke styrken i armene og samtidig beholde lik tykkelse på toppdel og bunndel, ble overlappen mellom disse laget stor. Slik at en større del av armene i praksis ble dobbelt så tykke som resten av skroget når det er sammensatt.

For at komponentene skulle bidra til en god plassering av felles massemidtpunkt ble de stablet både på oppside og underside av batteriet, og symmetrisk om midtdelen. Dette ivaretar også fordelene nevnt i 5.9.3 under prinsipielle akser. En festeanordning som festet komponentene i batteriet, og ikke i skroget var ønskelig. Dette hadde til hensikt å gjøre det mulig å enkelt kunne montere/demontere hele dronen i forhold til produksjon og inngrep i ettertid. 8 fester ble 3D-printet, tilpasset til hvert kretskort, og som ble festet på hver side av batteriet. For å holde det hele på plass ble åtte strikker som presset hver side inn mot kretskortene brukt for å låse fast fester mot kretskort.



Figur 26: Komponenter montert på batteri med komponentfester

Toppdelen av skroget er bygget så tett inntil batteri og komponenter at det også bidrar til å holde alt på plass. Angående montering av toppdel til bunndel, ble det i begge deler laget to spor til å feste strikk inn i for å låse dem fast. Dette på grunnlag av resultater fra simulering vist i figur 35. I tillegg var passformen mellom topp og bunn så god at friksjonen i mellom dem bidrar betydelig til å holde delene sammen. Figur 27 viser hvordan resultatet av festeanordningene ser ut på innsiden av skroget.



Figur 27: Tverrsnitt av ferdig montert skrog med batteri og komponenter



(a) Vekt på 35.5 gram



(b) Plassering av markører

Figur 28: Totalvekt av skrog og oppsett av markører til Motion Capture system

Ferdig 3D-printet prototype av skroget endte på 35.5 gram. Med komponentfestene i tillegg, på 0.6 gram hver, ble totalvekten på skroget med fester 39.4 gram. Like under målet som var satt til 40 gram. Tekniske tegninger av toppdel og bunndel er vist i appendiks C.1 og C.2.

Flatene som batteriet hviler på ble også designet med baktanke for muligheten til å senere kunne montere gimball med kamera. Den vil da kunne festes under disse flatene, uten å skape problemer for det kon formede designet. Dronen veier totalt med alle komponenter montert, i underkant 210 gram. Dermed vil det være mulig å tillate samlet vekt for gimball med kamera til 40 gram, og fortsatt være innenfor vektkravet på 250 gram.

Markørene for bruk i MotionLab er kuleformet og kan enkelt plasseres på dronen uten å komme i veien for propellene eller deres luftstrøm. Fokuset for markør plassering var at minst 3 kuler er synlig for kamera til enhver til, uansett orientering. En konfigurasjon er vist i figur 28b.

### 6.1.1 Duct

Når egen duct skulle designes ble det tatt utgangspunkt i en 3" (76.2mm) propell. Det ble utført test med duct i motor testriggen for å se hvor mye thrusten til motoren vi skulle bruke kunne økes med dette prinsippet. Følgende dimensjoner ble beregnet ut fra gitte formler i teoridelen til duct design 5.1.4:

$$D_t = \frac{76.2mm}{1 - 2 \cdot 0.001} = 76.35mm \quad (6.1)$$

$$\delta_{tip} = 0.001 \cdot 76.35mm = 0.076mm \quad (6.2)$$

$$L_d = 0.5 \cdot 76.35mm = 38.18mm \quad (6.3)$$

$$r_{lip} = 0.13 \cdot 76.35mm = 9.93mm \quad (6.4)$$

$$\theta_d = 10^\circ \quad (6.5)$$

Ducten ble laget ved hjelp av 3D-printing. Resultatet er vist i figur 29, der den er montert i testriggen. Teknisk tegning er vedlagt i appendiks C.7.

Ut fratestdata i figur 30 og 31, viser det seg at vi får opp mot 25% mer thrust med tilsvarende spenning som uten duct. En ser også at motorene trekker mindre strøm.

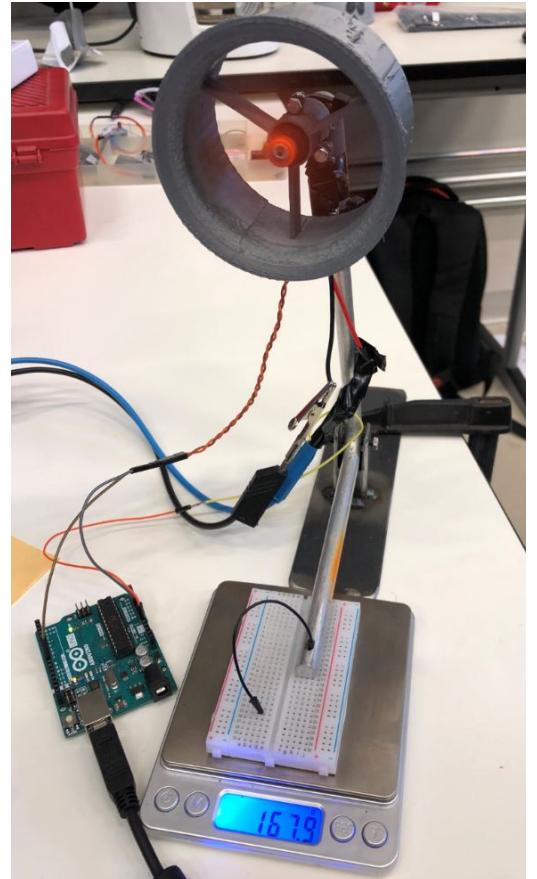
Ut fra CAD modell ble det estimert at en minimum fungerende duct med 1mm tykk sirkulær vegg og 2mm tykt motorfeste, ville veie i overkant av 17 gram. Dette med ABS-plast som materiale. Nøyaktig testdataene sees i figur 30 og 31, teksten forenkles med avrundede verdier.

Per motor vil vi med et gasspådrag på 80%, ha oppnådd en thrust på 86 gram. Den vil da thruste 20 gram mer enn uten duct, der thrusten er 66 gram. En må da opp til like under 80% thrust for å overgå en ekstra massen på 17 gram per duct.

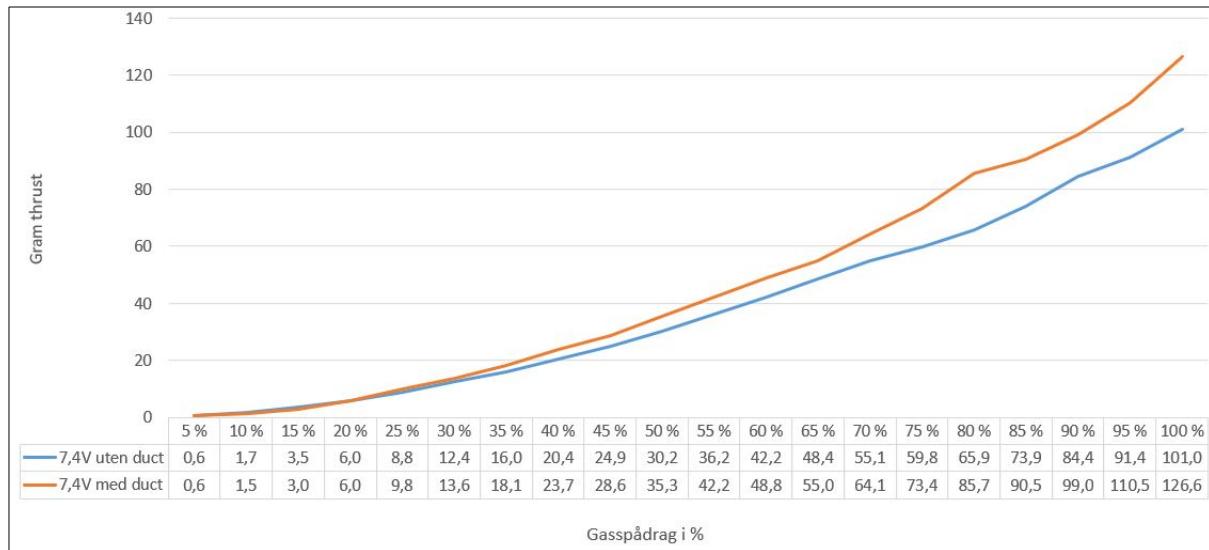
På 100% vil vi med duct ha oppnådd en thrust på 127 gram, og uten duct en thrust på 101 gram. Da vil vi, inkludert massen ducten utgjør, ha oppnådd en vunnet thrust på 9 gram. Da det er antatt at dronen skal hovre på rundt 40%, vil vunnet thrust her inkludert massen på duct være på 3 gram pr motor. Dette totalt på alle 4 motorene 13 gram, som ikke vil overgå den ekstra massen som totalt 4 ducter vil gi, som er på 56 gram.

Etter som strømforbruksgraden gradvis går ned jo høyere gasspådraget blir, ville dette i et ført til at flytiden kunne blitt forbedret. Vi kan se fra testdataene at ved hovring ved 40%, så ville strømforbruksgraden gått ned 0.06 ampere, til sammen med alle 4 motorene 0.24 ampere.

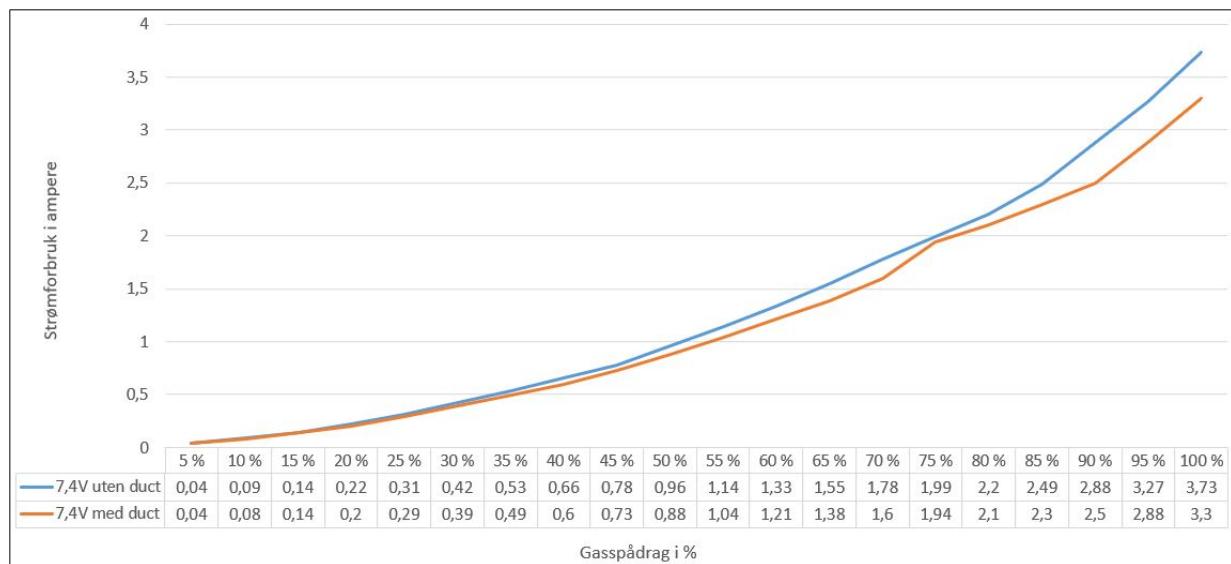
Størrelsen på ducten som er laget er liten, og i rapporten fra University of Maryland [16] ble testene utført i en vindtunell. Dette må bli tatt i betrakning, da dette kan ha påvirket resultatet. Det ble til slutt tatt en konklusjon på bakgrunn av den utførte testen, om å ikke benytte duct.



Figur 29: Duct montert i testbenk med motor og propell



Figur 30: Thrust



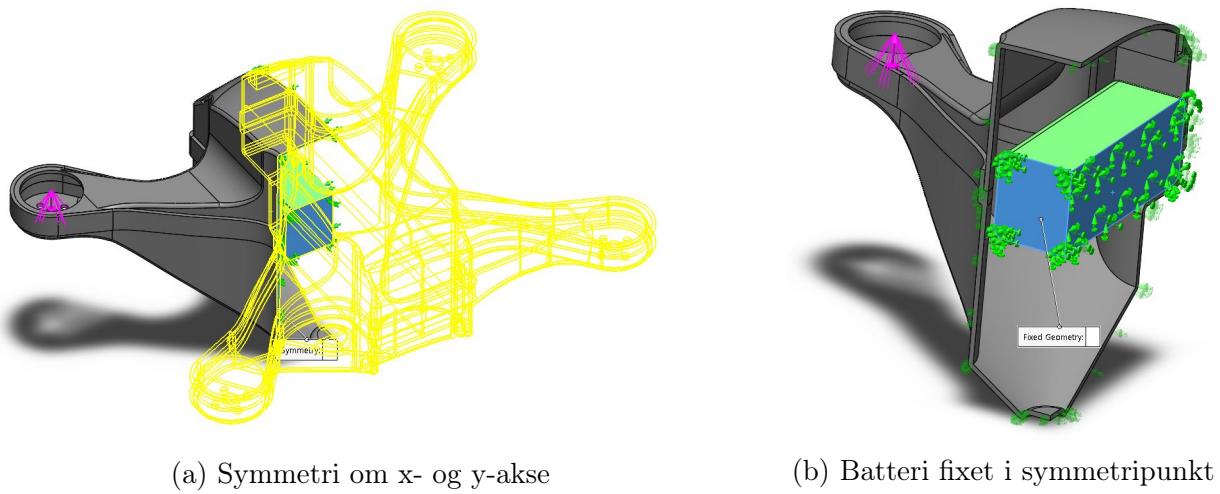
Figur 31: Strømforbruk

## 6.2 FEM-analyse

I FEM-analysen er hensikten å identifisere hvor elastisk skroget er, og hvor det vil oppleve størst påkjenninger ved flyvning. Med 1 mm tykkelse på godset, er det viktig at designet er designet for å tåle påkjenningene.

FEM-analyse vil gi indikasjon på om designet må endres, og hvor store disse endringene må være.

Grunnet simuleringsutfordringer blir symmetri fordelene utnyttet, vist i figur 32a, for å få en forenklet simuleringsprosessen. Dette gjør det mulig å gjennomføre simuleringer som gir et tilnærmet likt resultatet som ved å simulere hele dronen. Da dronen flyr, er det vanskelig å finne et fix-point. Det er enklere med fast monterte objekter. Batteriet, som største tyngste del, i senter av dronen ble valgt som fixed. Dermed se hvordan skroget beveger seg rundt det. Fixede flater er de blå flatene vist i figur 32b.



Figur 32: Symmetri og fix-point

For å kunne plassere kraften på hver arm på en realistisk måte ble "Remote Load" funksjonen brukt. I denne funksjonen defineres et punkt på rammen der en setter krefter på. Punktene ble skruenhullene der motorene er montert, og de ble definert ut fra et referansekoordinatsystem i senter av rammen, noe som i dette tilfellet også er symmetripunktet. Plasseringen av kraften er vist som rosa pil i figur 32a og 32b.

Hver motor har et max løft på 140 gram, noe som genererer en kraft tilsvarende 1.3734 N. Denne kraften vil da bli påført hver enkelt arm.

$$F = m \cdot a \quad (6.6)$$

$$F = 0.140\text{kg} \cdot 9.81 \frac{\text{m}}{\text{s}^2} \quad (6.7)$$

$$F = 1.3734\text{N} \quad (6.8)$$

Ut fra simulering med kraft og moment kan vi se hvilken retning det egner seg best at motorene roterer. Illustrasjon av rotasjonsretninger CW og CCW er vist i figur 18. Ved å rotere CW vil momentet bidra til å presse rammen inn mot batteriet, altså stive av hele konstruksjonen. Det motsatte vil skje om en roterer CCW, da dette vil bidra til å dra rammen ut fra batteriet, og forårsake en høyere forskyvning. For å finne motorens moment ble det tatt utgangspunkt i en motortest utført av Emax [38]. Ved test på 8V spenning, var motorens høyeste strømtrekk på 4.4A og en turtall på 27690 RPM. En kunne da beregne max moment pr motor:

$$T_{max} = \frac{P_{max}}{\omega_{max}} \quad (6.9)$$

$$T_{max} = \frac{U_{max} \cdot I_{max}}{\omega_{max}} \quad (6.10)$$

$$T_{max} = \frac{8V \cdot 4.4A}{27690 \cdot \frac{2\pi}{60}} \quad (6.11)$$

$$T_{max} = 0.01214 Nm \quad (6.12)$$

Ved oppdeling av noder og element ble det benyttet "curvature-based mesh", dette gir et høyere antall element og noder enn "standard mesh". Det vil da gi høy nøyaktighet av simuleringssresultatet. Forskjellen kan sees i figur 33, der antall element og noder er over dobbelt så høyt med en "curvature-based mesh". Tiden det tar å lage en mesh går også ned, men dette vil ikke ha noen betydning da tiden er så kort i utgangspunktet og konstruksjonen er såpass liten.

Mesh Details	
Study name	kraft og moment CW - standard mesh
Mesh type	Solid Mesh
Mesher Used	Standard mesh
Automatic Transition	Off
Include Mesh Auto Loops	Off
Jacobian points	4 points
Element size	3 mm
Tolerance	0.6 mm
Mesh quality	High
Total nodes	20725
Total elements	10789
Maximum Aspect Ratio	41.516
Percentage of elements with Aspect Ratio < 3	54.6
Percentage of elements with Aspect Ratio > 10	0.12
% of distorted elements (Jacobian)	0
Remesh failed parts with incompatible mesh	Off
Time to complete mesh(hh:mm:ss)	00:00:05

(a) Standard mesh

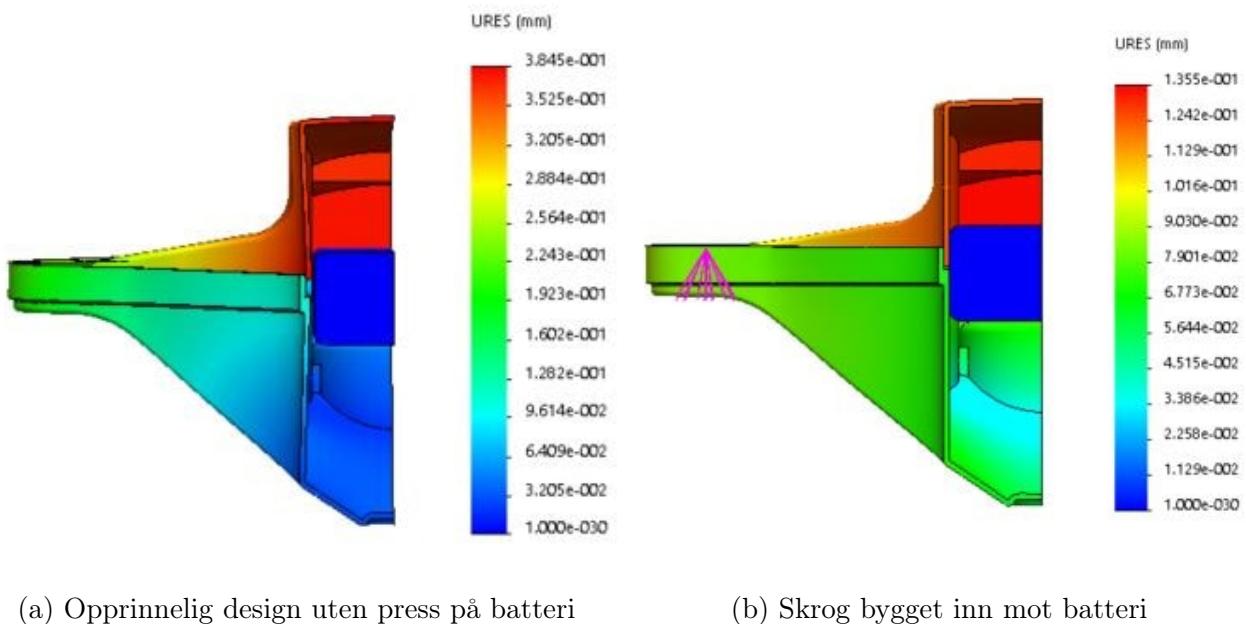
Mesh Details	
Study name	kraft og moment CW (-Default-)
Mesh type	Solid Mesh
Mesher Used	Curvature-based mesh
Jacobian points	4 points
Max Element Size	3 mm
Min Element Size	0.6 mm
Mesh quality	High
Total nodes	45060
Total elements	22663
Maximum Aspect Ratio	953.67
Percentage of elements with Aspect Ratio < 3	71.4
Percentage of elements with Aspect Ratio > 10	0.627
% of distorted elements (Jacobian)	0
Remesh failed parts with incompatible mesh	Off
Time to complete mesh(hh:mm:ss)	00:00:04

(b) Curvature-based mesh

Figur 33: Sammenligning av mesh

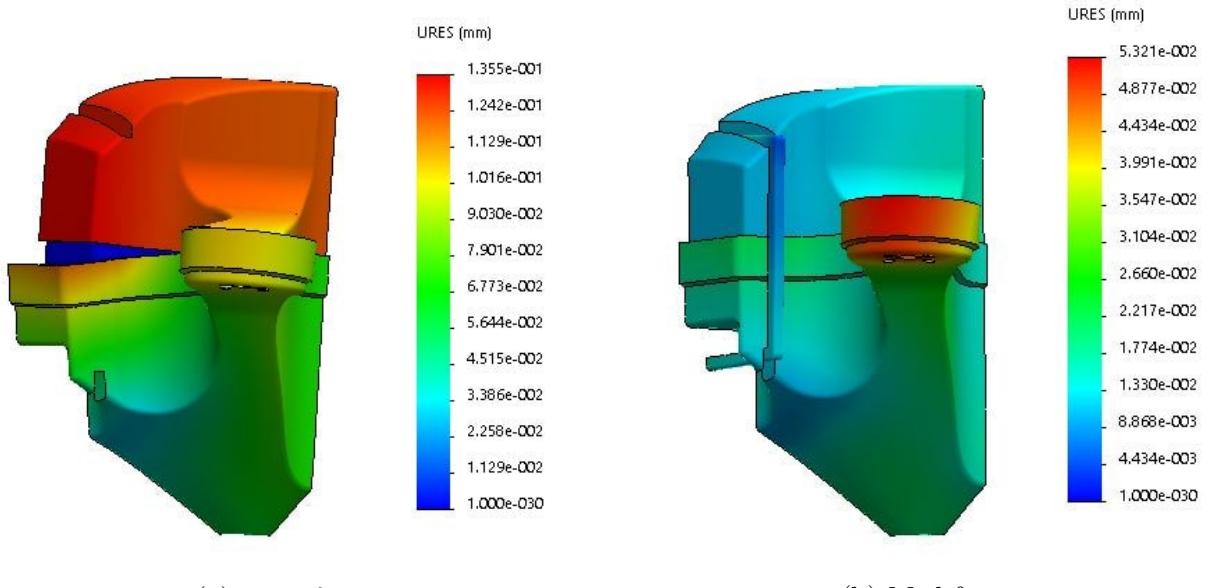
### **6.2.1 Forskyvning i skroget som følge av kraft og moment fra motor**

De tidligste designet av skroget hadde ikke innsnevninger mot batteriet. Ved hjelp av simuleringen kunne en se at store deler av kraften fra motorene ville skyve skroget inn mot batteriet. Det blir bare simulert med ABS-plast med 1mm tykkelse, da dette vil være det aktuelle spesifikasjonene med hensyn til produksjon. Ved å fokusere på et design som støtter seg mot batteriet, ble det tydelig forandring i kraftfordelingen.



Figur 34: Effekten av å bygge skroget inn mot batteriet

Ved å se på verdien vi får fra "URES" kan vi se på resultanten av forskyvningen i x, y og z retning. Vi ser i figur 34 at ved å endre plassering av batteri og snevre inn designet mot batteriet, kan det sees at største forskyvnings utslag gitt av bøyning i hver arm synker fra 0.38mm til 0.13mm forskyvning, noe som halverer høyeste forskyvningsverdi. For å se resultatet tydelig, har både 34a og 34b skalering 10:1. Siden forskyvningsverdien er så liten er det den forbedrede forskyvningsfordelingen som er fokuset her, som viser hvilken effekt kretene som oppstår fra motoren vil ha på skroget.



(a) Uten feste

(b) Med feste

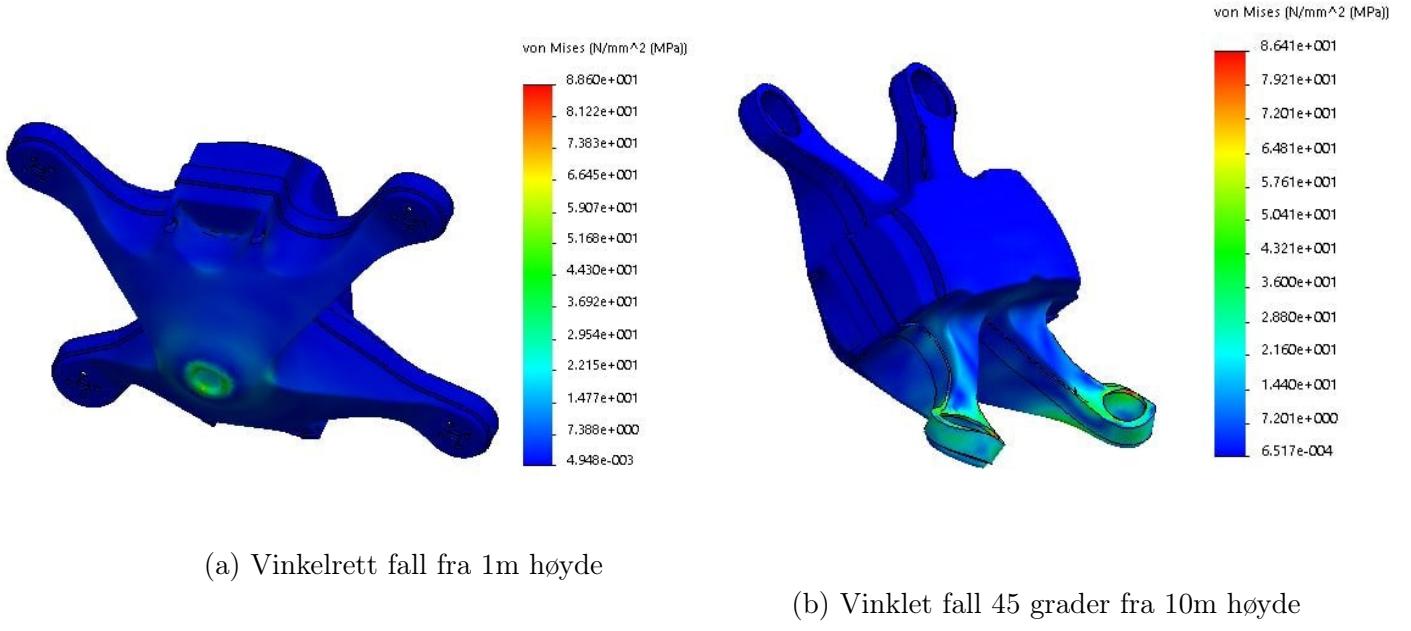
Figur 35: Forskjellen mellom festet og ikke festet topplokk

Ut i fra figur 35a vil topp og bunn bli skyvd fra hverandre i frem og bakdel. For å se resultatet tydelig har figur 35a og 35b skalering 50:1. Dette ble også testet i motsatt rotasjonsretning med samme resultat, vist i appendiks D.1. Å plassere festeanordningen her gjør at den bidrar til å stive av rammen i tillegg til å fungere som feste. Kreftene som oppstår fordeler seg ganske jevnt over topp og bunn, som vi ser i 35b. Materialet på festet er også simulert som ABS-plast, så i realiteten vil bruk av en stram gummistrikk ikke gi like godt resultat, men antas tilstrekkelig da forskyvningen er i utgangspunktet er liten.

Resultatet av endelig design og sammensatt ramme viser i følge simuleringene at de største delene av rammen vil oppleve et jevnt fordelt trykk og forflytning som ligger fra ca 0.008mm til 0.035mm, som er svært lavt. En ser at den største forflytning vil skje helt ute i armene, der vil den være på ca 0.05mm. Som nevnt er ikke størrelsen på forskyvningene fokuset, da de er tilnærmet neglisjerbare. Fokuset rettes mot hvordan kreftene de blir fordelt i rammen, da den er så tynn.

### 6.2.2 Sammenstøt

I og med at dronen er designet for bruk som bare innebærer hovring og svært begrensede bevegelser i luften, er den ikke designet for å tåle harde sammenstøt. Som vist i tabell 1 har ABS-plast en strekkfasthet på 47 MPa. Dette vil da være nesten 28 ganger mer enn høyest oppnådd von Mises verdi på 1.7 MPa, som kommer av simuleringer der maksimale kraftpåkjenninger fra motorene er påført rammen. Von Mises verdien er sammenlignet med strekkfastheten til materialene i tabell 1. Forskyvninger og spenninger fra flyvning er vedlagt i appendiks D.1.



Figur 36: Sammenstøt ved fall

Likevel om skroget ikke er designet for å tåle harde sammenstøt er det likevel interessant å simulere hvordan designet vil håndtere slike situasjoner. Derfor ble det simulert forskjellige scenarioer ved fritt fall. Det ble sett på påkjenningsene ved fall fra 1m- og 10m høyde. Her ble det simulert et fall fra vinkelrett hovring og ned i bakken, og et fall fra hovring der sammenstøtet treffer 2 av armene i en vinkel på 45 grader.

Når det skulle lages en maske på konstruksjonen var ikke CBM tilstrekkelig å bruke. Elementene som ble fordelt hadde ikke en sammenhengende kontakt mellom bunndel og toppdel. Dette kunne løses med å benytte SM, som fikk fordelt elementnettet slik at alle elementene hadde kontakt over hele skroget.

Om vi ser på simuleringen ved et fall på 1m, der den faller vinkelrett ned i bakken, vil bunnen ta opp største del av påkjenningsene. Punktet der jordingstilkoblingen til lading er, vil få påkjenninger helt opp mot 80 MPa, men jevnt over 45-50 MPa. Dette er på grensen til brudd. Ved sammenstøtet der dronen var vinklet 45 grader, oppnådde den høyeste påkjenninger ved ca 22 MPa. En ny simulering på 10m fall med samme vinkel, viser at armene får påkjenninger opp mot 86 MPa på enkelte punkt. Jevnt over rundt 40-50 MPa, noe som også er på grensen til brudd. Simuleringen ved 45 graders fall er ikke helt representativt til et virkelig fall, da den gir armene all påkjenningen. Ved et slikt fall vil massemidtpunktet som ligger i senter av dronen, føre til at skroget tiltar slik at bunnen treffer bakken. Den vil da bidra til å ta opp store deler av påkjenningsene. På tross av dette får vi et resultat som viser at armene vil tåle sammenstøtet, selv når de tar opp all påkjennning. Spenninger fra alle simulerte fall er vedlagt i appendiks D.2.

Dette var en simulering der kun toppdel, bunndel og batteri var inkludert, da dette er de bærende delene av dronen. Det ble lagt til en masse på batteriet som ga modellen en totalvekt på 260 gram. Den hadde da en sikkerhetsmargin på 10 gram over vektkravet.

## 6.3 Strøm

### 6.3.1 Batteri

Det var ønskelig å holde tilbake valg av batteri til alle andre komponenter var bestemt, grunnet muligheten til å fylle opp all resterende vekt innenfor 250 gram kravet med batteri. Dette for å oppnå lengst mulig flyvetid. Vekt på komponenter er vist i tabell 3. Vi må ta i betrakning at vekt på kabler og loddetinn vil komme i tillegg, noe som er vanskelig å estimere en vekt på.

Derimot måtte batteri bestemmes tidlig nok til å komme i gang med rammedesign, for å bygge rammen rundt batteriet. Det ble tidlig satt opp vekt tabell som en metode for å holde kontroll på anslått totalvekt, etter hvert som komponentene ble valgt ut. Tabell 3 er den endelige tabellen og den var klar tidlig nok til å kunne velge batteri. For å få lengst mulig flyvetid ble alle komponenter valgt med hensyn til vekt, og så ble resten av vekta tilgjengelig innenfor 250 gram brukt til batteri. Det valgte batteriet i figur 37 er et bra batteri mtp. vekt og størrelse kontra kapasitet. Det veier 81 gram og har målene  $89 \times 30 \times 16$  [mm]. Kapasiteten er på  $1600mAh$ , med 2S nominell spennin på 7.4V. C-raten er konstant på 20C og har en peak (10sek) på 30C. Ved valget av dette batteriet vil totalvekten være estimert til ca 236 gram, med en margin på 14 gram til kabler og loddetinn.



Figur 37: Valgt batteri

Tabell 3: Vekt komponenter

Komponenter	Vekt [gram]
Ramme + Komponentfester (8stk)	39.4
Gimball + kamera	40
Propeller (4stk)	4
Motor (4stk)	28
Skruer (24stk)	5.5
Raspberry PI Zero W	9
IMU	2.1
RPIzw CAN modul	2.3
ESC CAN modul (4stk)	10
Supply board	2.8
Motordrivere (4stk)	12
Total:	155.1

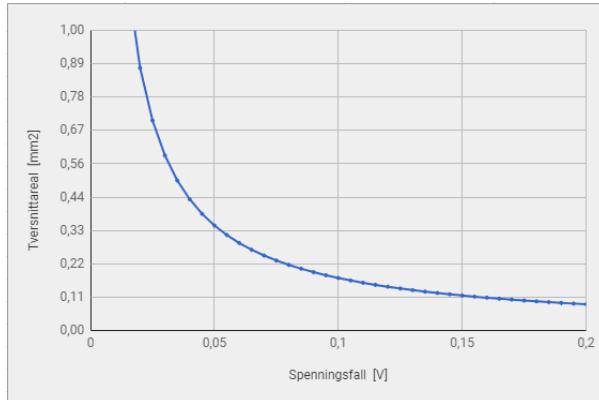
### 6.3.2 Ledere

Grunnet god tilgjengelighet ble det valgt å bruke kobberledere.

Kobber har resistivitet  $\rho = 0,0175 \left( \frac{\Omega \cdot mm^2}{m} \right)$  ved  $20^\circ C$  [19]. Dermed kan nødvendig tverrsnittarealet for kabler ut til motorene beregnes ved hjelp av ligning (5.13). Med  $\Delta U$  som variabel, max strømtrekk  $I$  til  $5A$  og kabellengde satt til  $0.05m$  kom grafen vist i figur 38. Hvor det tydelig kommer frem at for spenningsfall under  $0.05V$ , kreves det fort større tverrsnitt.

Det ble bestemt at for dette prosjekt tillates  $0.05V$  spenningsfall og ledninger med tverrsnittarealet ned til  $0.33mm^2$  kan brukes til motorene. Effektloven, ligning (5.14), gir da at det brukes  $0.25W$  per motor ved max strømtrekk  $5A$  for å lede strømmen til motoren.

Forøvrig gir strømtrekk på  $7A$  (som er max strømtrekk til ESC'ene) spenningsfall på  $0.1V$ . Det tilsvarer  $2.8W$  til varme.



Figur 38: Tversnittarealet [ $mm^2$ ] vs. Spenningsfall [V]

### 6.4 Ladestasjon

Ladestasjonen er også designet for å være enkel å masseprodusere. Prototypen består av totalt 4 deler og er laget for å stå på bakken. Når dronen står oppi den ser vi at propellene kan generere luftstrøm utenfor landingsområdet. Det gjør at om den eksempelvis henger på veggen med luft under seg vil en unngå mye potensiell luftmetning fra bakken ved landing. Den er laget av 3D-printer og plastrør. Plastrøret huser laderen, vist i figur 40b. Delene produsert av 3D-printer utformer toppen, med kon form som passer mot dronens bunndel. Tekniske tegninger av ladestasjonen er vedlagt i appendiks C.5 og C.6.

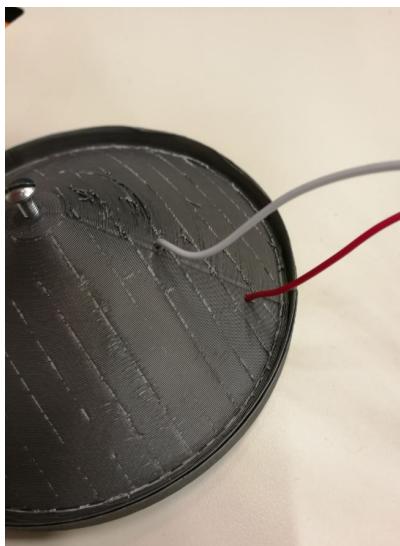


Figur 39: Spenningsfordeler

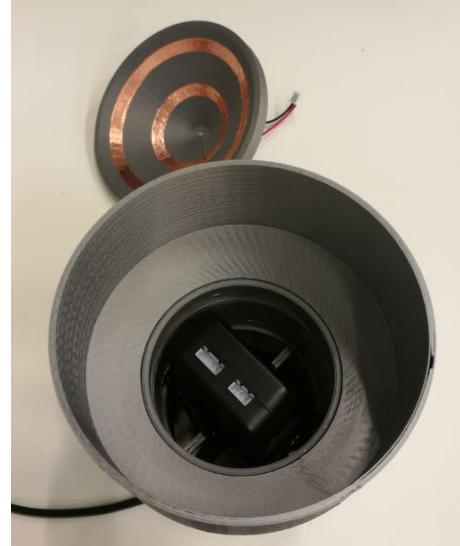
Laderen er en "Turnigy compact charger E3", som kan brukes på 2 og 3 cellers batteri. Den har innebygd spenningsomformer, kobles direkte til 230V. Vist i figur 39.

Laderingene ble konstruert etter at prototype av ladestasjonen ble printet ut. Dette ved hjelp av kobbertape som las over nedsenkede ringer i konen, der overflødig materiale ble skjært bort. Tilkobling mellom lader og laderingene er vist i figur 40a. Kablene ble loddet fast i laderingene.

Ladestasjonen er designet med hensyn til innendørs omgivelser, og inneholder derfor ingen form for vanntett tildekning mens dronen lader. Det er antatt at dronen ved hjelp av sin egen vekt klarer å plassere seg i riktig posisjon i ladestasjonen. Dette ble testet og fungerte bra. Landingsområdet i konen ble laget så stort som mulig, kun begrenset av dronens dimensjoner. Parametere som vinkelen på bunndelen til dronen, og høyden fra bunn opp til motorene spilte inn her. Landingsområdet endte med diameter på 118.6mm, høyde 34.7mm og vinkel 56.7°. Resultatet er vist i figur 41b.



(a) Tilkobling til laderinger



(b) Plassering av spenningsfordeler

Figur 40: Innhold ladstasjon

For å få kontakt mellom batteriet og laderingene ble det benyttet tynne blad av fjærstål fra en bladføler som kontaktpunkt i rammen på dronen. Disse ble plassert med hver sin posisjon i forhold til laderingene sin posisjon. De har en fjær-effekt, slik at de presser mot laderingene i ladestasjonen med hjelp av dronen sin vekt. De 3 kontaktpunktene er vist i figur 41a. Det er avgjørende at det ikke kan oppstå en kortslutning, så laderingene ble plassert med størst mulig mellomrom mellom hverandre. Rammen har ved jordingspunktet en utforming som gjør at stålbladet plassert her blir innfelt, og kan kun treffe jordingspunktet i ladestasjonen. I ladestasjonen består jordingspunktet av en leder, som treffer mot det innfelte stålbladet.

Ved endelig test ble det oppnådd tilstrekkelig kontakt på alle ladepunktene, slik at batteriet ladet om det skulle. Dette viste at prinsippet med laderinger som lademekanisme fungerte. For å illustrere en landing ble dronen blir sluppet ned i stasjonen.



(a) Plassering av kontaktpunkter i ramme



(b) Sammensatt ladestasjon

Figur 41: Ladestasjon og kontaktpunkter

## 6.5 Produksjon

Med tanke på masseproduksjon kontaktet gruppen Unica AS som driver med vakuumforming. De anslo pris på en eventuell masseproduksjon, og ga prisantydning på negativen beskrevet i 5.1.3, til 16 000 kr. Det er en engangskostnad, og den kan bli brukt til tusenvis av eksemplarer.

Ved produksjon av 5 eksemplarer vil hver del koste ca 800 kr, altså en total pris på 1600 kr for et komplett skrog. Ved masseproduksjon synker prisen derimot betraktelig, og en stor bestilling er nødvendig for at dette skal være en aktuell produksjonsmetode å velge. Ved produksjon av 1000 eksemplarer ble prisen betydelig lavere, der et komplett skrog vil ha en totalpris på 150

kr. Dette er priser eksl. moms [39] Gruppen har ikke sjekket opp priser i andre land en Norge, men er klar over muligheten for at det kan være enda billigere å bestille fra produksjonsland som eksempelvis Kina.

## 6.6 Motortest

Dokumentasjon av parameterer som treghetsmoment, elektrisk tidskonstant og thrust er ofte svært mangelfull ved kjøp av RC-elektronikk. Dette er noe av grunnlaget for at motortest prioriteres såpass høyt i denne oppgaven.

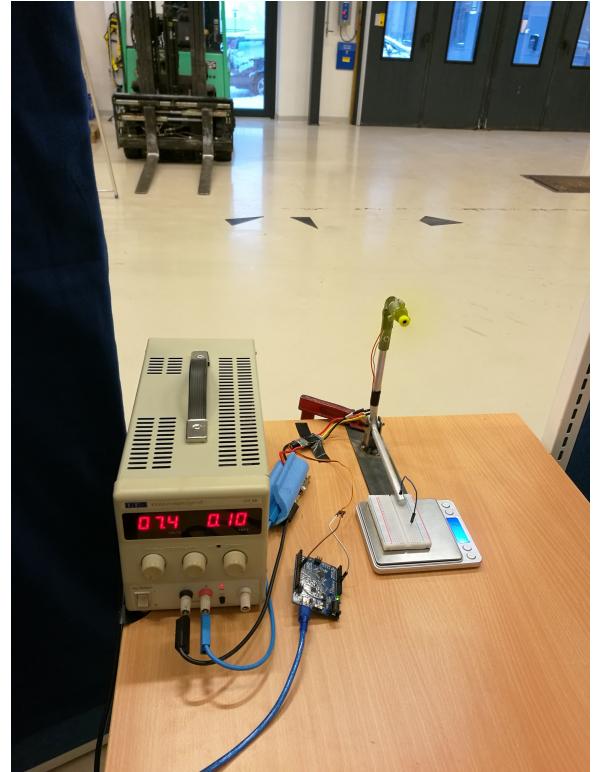
For gruppen var det viktig å fastsette motor- og propell valg på et tidlig tidspunkt da det har stor innvirkning på rammedesign og batterivalg. Det ble derfor kjøpt inn aktuelle motorer og propeller som tilsvarende ville være gode valg basert på leverandørens informasjon.

### 6.6.1 Thrust test

Motortest riggen med pålydende geometri som vist og forklart i 5.11 ble bygget. Med  $L_1 = 137,2\text{mm}$  og  $L_2 = 182,0\text{mm}$  løses forholdstallet mellom avlest og faktisk thrust Basert på ligning (5.62).

$$\frac{L_1}{L_2} = \frac{137,2}{182,0} = 0,7538 \quad (6.13)$$

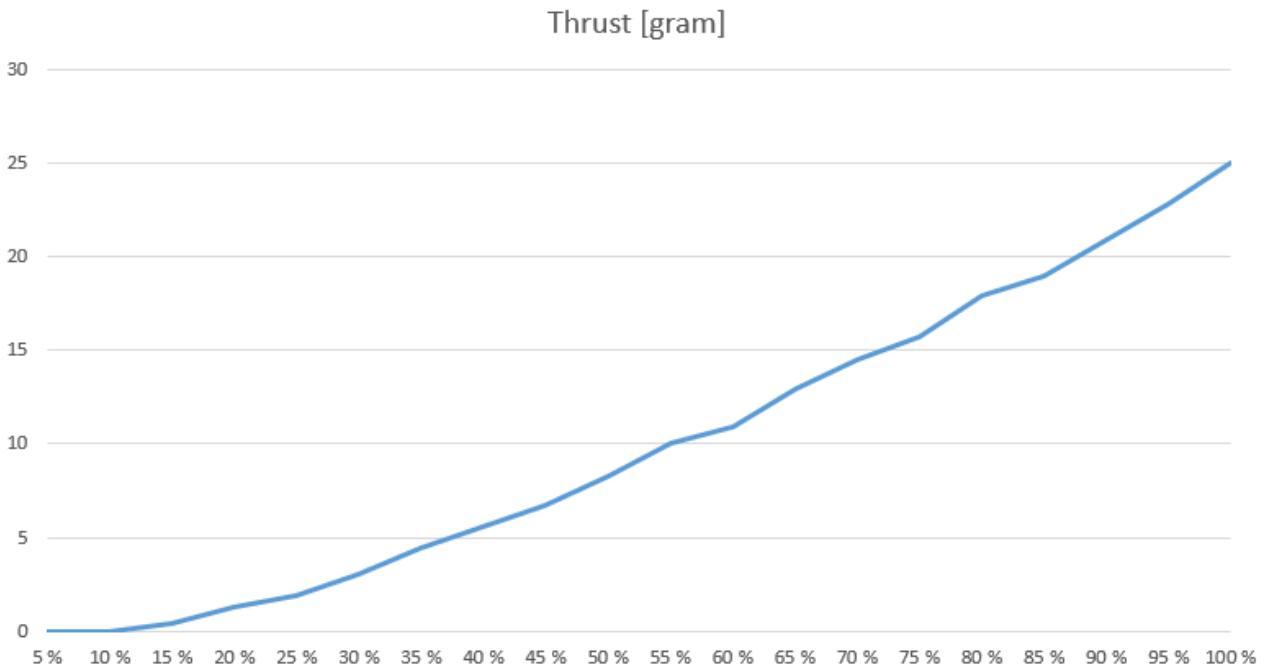
Under test leses N kraft av ligning (5.62) av på vekten, og føres i Excel hvor det multipliseres med forholdstallet og vi får F. Test er utført med 20 steps fra 0 til 100 % pådrag. Ferdig bygget rigg, ferdig oppkoblet, klar for test sees i figur 42.



Figur 42: Ferdig rigg, klar for test, fri luftstrøm foran og bak

## Børstemotor

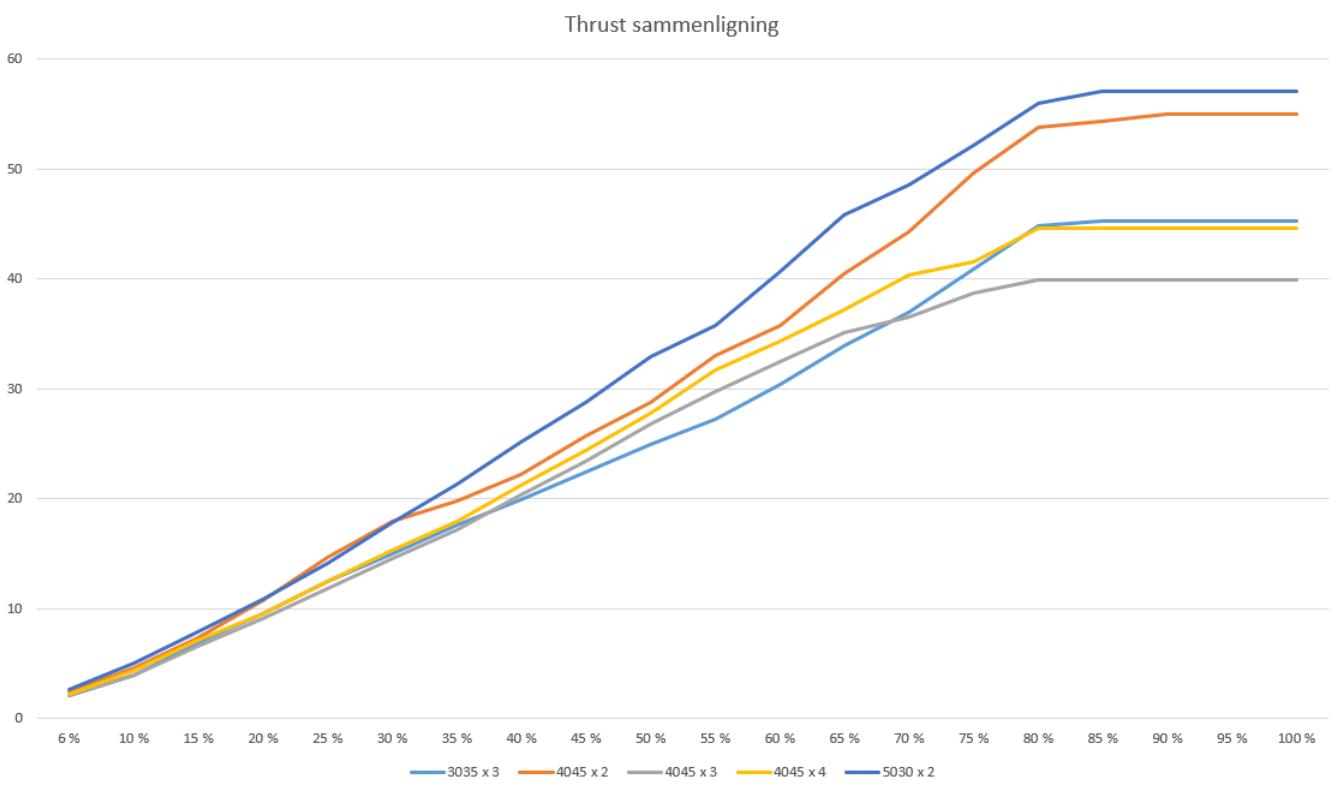
Gruppen har etter konseptvurdering liten tiltro til børstemotorer. Som funnet i konsept-del 4.3 er det begrenset utvalg av børstemotorer med peak thrust over 25 gram. Det ble likevel kjøpt inn en børstemotor, den sterkest tilgjengelige, for å gjøre egen test og vurdering. Dimensjon  $20mm \times 8mm$  testet med en  $2030 \times 2$  propell. Disse motorene er meget gunstige med tanke på pris, vekt og enkel justering av pådrag uten behov for ESC. Testresultatet i figur 43 under viser at 25 gram peak er en sann verdi fra leverandør. X-aksen indikerer % pådrag.



Figur 43: Motortest, børstemotor

## Børsteløs motor

Videre ble en 21watt børsteløs motor fra Turnigy kjøpt [40]. Det er den minste, letteste motoren tilgjengelig i hobbyking sin nettbutikk som gruppen har tro på at kan være sterk nok. Leverandør rater den til 70 gram peak thrust med en 3020 propell. Gruppen ønsket over 70g peak thrust, så konfigurasjoner med større propeller/høyere pitch ble derfor valgt til testen. Grafen i figur 44 viser testresultat med de forskjellige propellene som ble valgt til test. I forhold til leverandørens anslag på 70 gram thrust er resultatene skuffende, med tanke på at den er testet med propeller som burde gitt mer thrust på samme turtall.

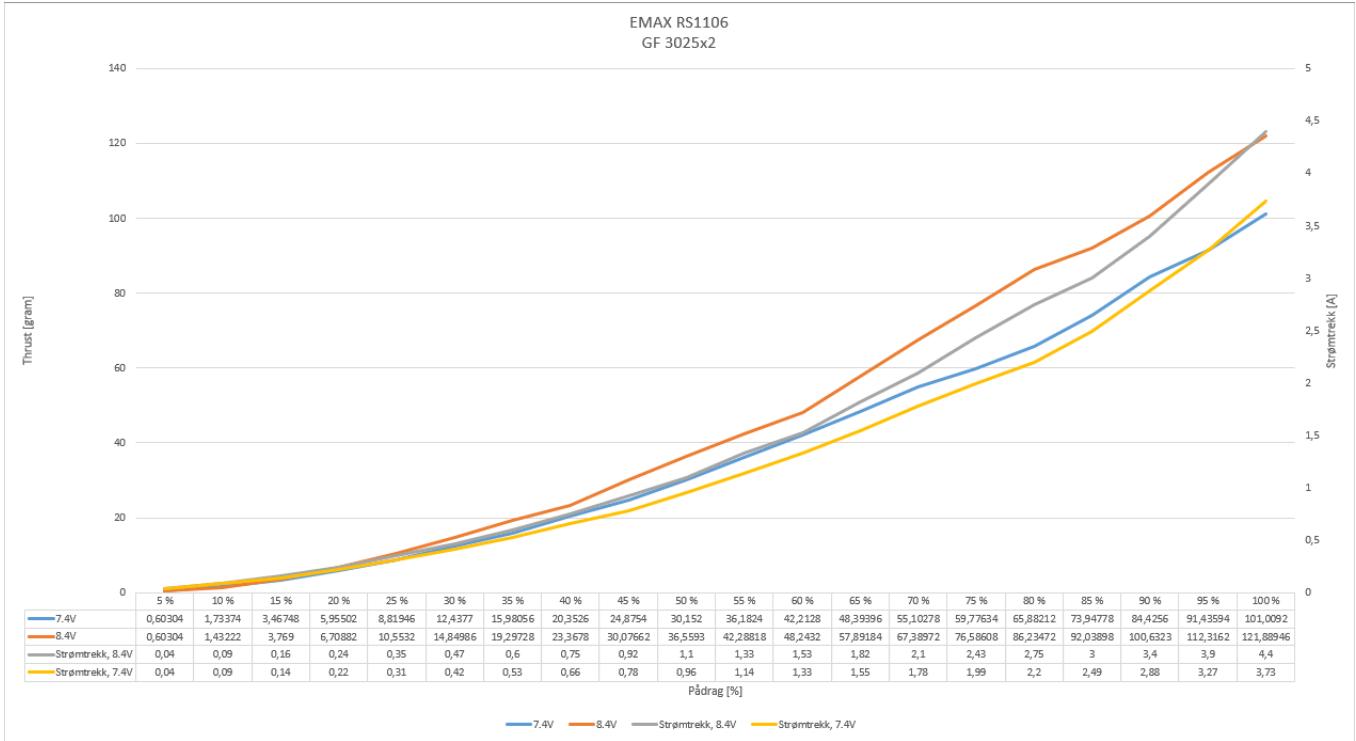


Figur 44: Sammenligning av propeller, 7.4V, "Turnigy F1104-4000KV 5.5g"

Thrust krav for å hovre finnes ved å ta  $250\text{gram} \div 4\text{motorer} = 62.5$ . Motoren i testen klarer ikke dette, i tillegg er det ønskelig å kunne hovre på omtrent 50% pådrag. Konklusjon etter test blir dermed at denne motoren ikke er aktuell å bruke i prosjektet.

Etter litt leting kom gruppen over en motor som ble kjøpt inn av flere grunner. EMAX RS1106, som påstås å være den sterkeste motoren for 2- og 3 tommers propeller. I tillegg har både leverandør og eksterne utført seriøse tester med flere forskjellige propeller [41, 42]. Det forenkler da arbeidet med å velge riktig propell på første forsøk. GF3025 ble kjøpt på bakgrunn av testdata fra leverandør, for så å teste videre selv.

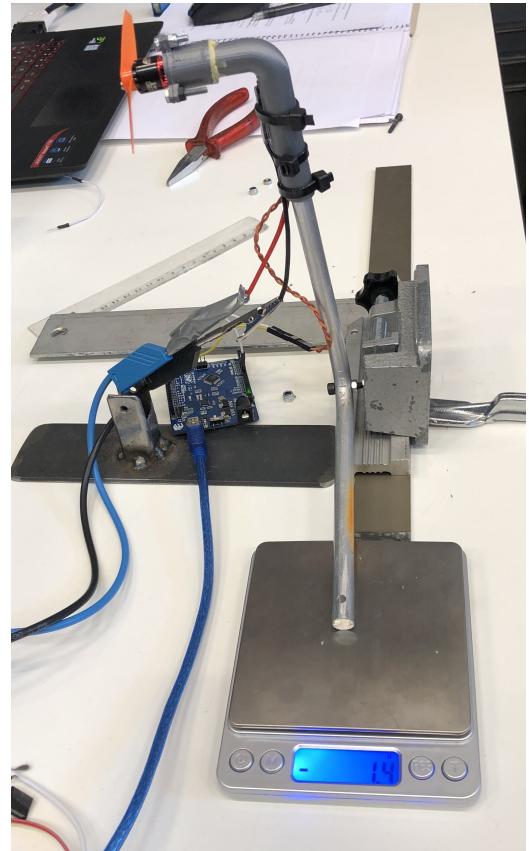
I figur 45 sees testresultat av EMAX motoren. Den er testet med 7.4 og 8.4 volt, og det er lest av strømtrekk for begge. 7.4 volt er nominell spenning på vårt aktuelle 2 cellers batteri, mens 8.4 V er spenning når batteriet er fulladet. Strømtrekk leses av på høyre y-akse, og thrust på venstre. Det registreres at den thruster over 100 gram både for 7.4V og 8.4V ved 100% pådrag, som totalt gir over 400 gram løft. I likhet med leverandøren får også vi ut maksimalt strømtrekk til å være 4.4A. Det registreres også at den klarer å hovre (62.5g) rundt 65-70%, hvor den trekker i underkant av 2A. Det kan tenkes at det finnes tap i motortest riggen og at tallene i realiteten er bedre. Tallene vil ikke kunne bli verre og gruppen sier seg fornøyd med motoren, og kan gå videre til test av vridningsmoment.



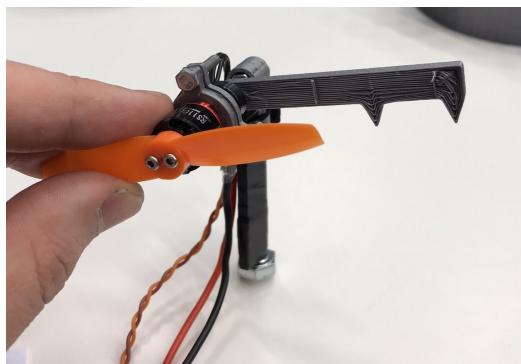
Figur 45: Thrust-test av EMAX RS1106 - GF 3025x2

### 6.6.2 Test av vridningsmoment

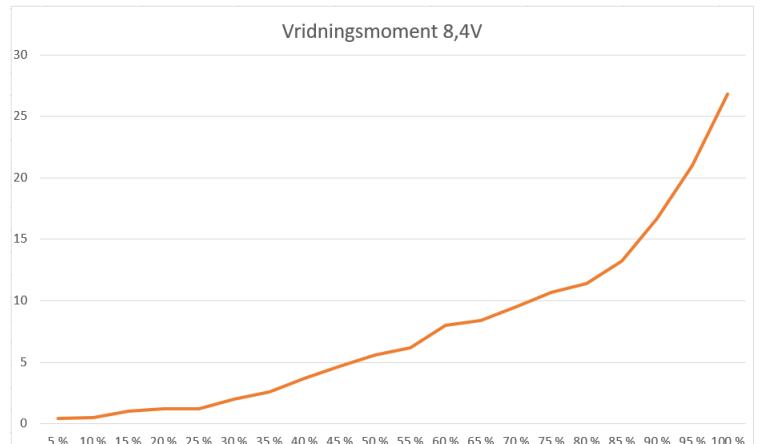
Test av vridningsmoment ble ikke utført før motor var bestemt. Da disse dataene ikke er kristiske for selve motorvalget med tanke på direkte thrust, men for å kunne motvirke vridningsmomentet som påføres rundt yaw-aksen. I 5.11 ble det lagt frem to modeller for måling av vridningsmoment. I figur 46 sees den første modellen. Som sagt var det ikke enkelt å få gode målinger med denne. Repeterbarheten var dårlig, og resultatene forskjellige for hver test. Den nye riggen er vist under i figur 47a, med resultat. I figur 47b sees det at ved 100% pådrag dannes et vridningsmoment på over 25gram. Dette er viktig data som brukes videre i drogens motormodell.



Figur 46: Første rigg for måling av vridningsmoment



(a) Endelig rigg for måling av vridningsmoment

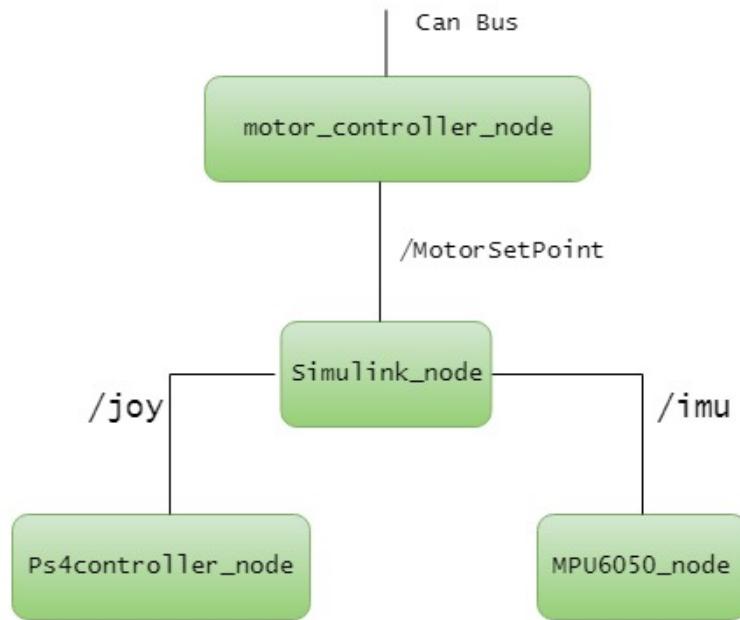


(b) Resultat av test

Figur 47: Endelig rigg for test av vridningsmoment og resultat

## 6.7 Robot Operating System (ROS)

ROS ble benyttet aktivt gjennom hele oppgaven og gjorde eksempelvis implementasjon av stabiliseringalgoritme modellert i Simulink enkel. Nettverk brukt ved avsluttende testing er vist i figur 48. Under følger en kort oppsummering av nodenes viktigste oppgaver.



Figur 48: Oppbygning av ROS nettverk for testing av drone

### **motor\_controller\_node**

Tar imot informasjon om ønsket pådrag på motorer fra Simulink\_node på emnet "\MotorSetpoint". Hver gang noden mottar nye pådragsverdier, flyttes verdiene over i fire CAN bus meldinger med unik meldings-ID tilhørende hver enkelt motor. Meldingene sendes deretter ut på CAN bus nettverket. Fullstendig kode sees i appendiks B.2.

### **Simulink\_node**

Generert node fra Simulink som inneholder PID-kontrollere, signalmiksing og tilbakekoblinger ved bruk av IMU for stabilisering av dronen. Noden lytter på "\imu" og "\joy". På disse emnene publiseres henholdsvis aksellerasjon- og vinkelhastighetsvektor fra MPU6050\_node, samt posisjon på analoge stikker på ps4 kontrolleren.

Som nevnt i 5.5 er "rosparam set" blokkene benyttet for å endre verdier på forskjellige offset og forsterkinger etter noden er generert.

Publiserer emnet "\MotorSetpoint" som sendes til motor\_controller\_node og kringastes dermed videre på CAN bus nettverket.

### **MPU6050\_node**

Noden mottar data fra MPU6050, og publiserer de på emnet "\imu". Noden er basert på mpu6050 ROS node fra matpalm [43]. Modifikasjoner som er utført er beskrevet i detalj i 6.16.1. Fullstendig kode sees i appendix B.3.

### **Ps4controller\_node**

Publiserer på emnet "\joy" som sendes til Simulink\_node. Noden er basert på ps4-ros node fra solbach [44]. Bufferstørrelsen er endret fra 1000 til 1 for å sikre at siste verdi alltid er den som publiseres. Fullstendig kode sees i appendiks B.4.

## 6.8 Kretskort

### 6.8.1 Egenskaper

Før valg av komponenter og design av skjematikk med utlegg kan begynne må kretskortenes egenskaper fastsettes.

#### RPIzw CAN modul

RPIzw er konfigurasjonens master. Den tar imot sensordata, og styrer pådrag til motorene deretter. Den må derfor formidle beskjeder ut til CAN bus nettet. RPIzw CAN modul sin egenskap er derfor å kommunisere med RPIzw over SPI og gjøre om til CAN.

#### ESC CAN modul

- Kommunisere over CAN med RPIzw.
- Omgjøre CAN melding til PWM signal og sende til ESC.
- Ta imot batteristrøm som egen forsyning, sende videre til ESC.
- Mulighet for  $120\Omega$  termineringsmotstand til CAN nett.
- Mulighet for programmering av mikrokontroller.

#### Supply Board

- Ta imot hovedstrøm fra batteri, max 25A.
- Fordel ut strøm til 4 motorer, max 5A per.
- Måle totalt strømforbruk.
- Kunne varsle RPIzw ved lavt batterinivå.
- Mulighet for programmering.
- Forsyne seg selv og RPIzw med 5V.

### 6.8.2 Design

#### Ytre dimensjoner

Fra rammedesignet ble det utarbeidet krav til max størrelse på kretskort. Ettersom rammedesignet er styrt av vektkrav og laget så kompakt som mulig ble det en styrende faktor for ytre dimensjoner for kretskortene. Et kompromiss mellom å få plass i rammen og å få plass til alle komponentene ble funnet. Vist under, i tabell 4. Med ytre dimensjoner fastsatt kunne de modelleres opp i CAD. For å lage ordning for innfestning og sikre plass i skroget, vist i figur 26.

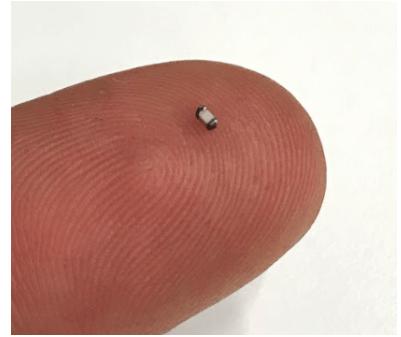
Tabell 4: Ytre dimensjoner, kretskort

Kretskort:	Lengde	Bredde	Høyde m.komponenter
ESC CAN-Module	20 mm	39 mm	5 mm
Supply Board	25 mm	40 mm	5 mm
RPIzw CAN module	20 mm	26 mm	5 mm

## Dimensjonering

Det var usikkerhet rundt bruken av Universitetets reflow ovn. Hvor godt den ville fungere og hvor høy kvalitet ferdig produkt ville ha. Derfor var det ønskelig å beholde muligheten for håndloddning på prototypen. Utlegg ble forsøkt i Eagle, med motstander og kondensatorer i størrelse 1206, da disse er greie å håndloddne. Det viste seg å bli vanskelig å holde seg innenfor de ytre dimensjonene satt for kortene med 1206. 0603 ble derfor valgt, de er vanskeligere, men også mulige å håndloddne.

Av mikrokontrollere ble SOIC pakker med synlige ben valgt. QFN pakker uten ben skal også være mulige å håndloddne, men gruppen valgte å ikke gå for disse på prototypen.



Figur 49: 0603 Resistor

### 6.8.3 Valg av komponenter

#### Mikrokontroller

ATtiny85 ble valgt til ESC CAN modulene. Den har tilstrekkelig med pinner til SPI fra CAN-tranciever og ledig pin til å sette ut PWM til ESC. Den har heller ikke flere pinner en dette. Så for å utføre sin oppgave brukes alle pinner.

ATtiny84 ble valgt til Supply Board da det er ønskelig å ha mulighet for flere funksjoner på dette kortet en på ESC CAN modul, som bare skal virke til sitt formål. Styre en transistor og tenne LED lys kan være av ønskede funksjoner. Da kan kortet enkelt utvides til at også dette kortet støtter CAN bus på et senere tidspunkt. Siden ATtiny84 klarer det, og litt mer. I tillegg har den støtte for ADC til måling av strømforbruk, med 10-bits oppløsning. For så å sette flagg høyt, som lavt batterinivå-varsle til RPIzw.

ATtiny84 er en litt større utgave av 85 og den ble valgt av den grunnen å ha litt mer åpent for å gjøre mer på et senere tidspunkt.

Både ATtiny84 og 85 har støtte for avr-libc [45].

#### CAN bus hardware og krystall

Gruppen har i tidligere prosjekter brukt MCP 2551 og 2515 til å sette opp CAN bus kommunikasjon, med gode resultat. Disse ble valgt også til dette prosjektet, men nå i SOIC pakken. Det fungerte den gang bra med 8 MHz krystall under testing av, og 8Mhz ble derfor valgt også i denne kretsen, med anbefalt kondensator på 18pF [46]. Grunnet kompatibilitet med biblioteket som vi bruker for å sette opp CAN; "avr-can-lib" så ble det senere byttet til 16Mhz krystall [47].

## Dekoplingkondensatorer og Pull-up resistor

Decoupling techniques fra "Analog Devices" anbefaler en stor kondensator ( $10\mu F$  -  $100\mu F$ ) ikke lenger unna en 2 tommer fra chipen. Den skal fungere som et strømreservoar til de øyeblikkelige strømkravene lokalt i kretsen, slik at strømmen ikke må trekkes gjennom induktansen i strømbanen. De anbefaler i tillegg en mindre kondensator ( $0.01\mu F$  -  $0.1\mu F$ ) så nerme som mulig power pin, for fjerne høyfrekvent støy bort fra chipen [48].

Konfigurasjonen her overstiger aldri 2 tommer i ytre mål, i tillegg til at inputstrøm fra batteriet antas å være stabil, blir første anbefaling ikke fulgt. Derimot de-kobles spenningsregulatorene både på input og output pin, iht. sine respektive datablader [49, 50], for å oppnå så stabil strømforsyning som mulig.

Mindre kondensatorer settes så nerme som mulig, mellom power-pin og ground på chipene. Verdier velges iht. datablad. I datablad for ATtiny 84 og 85 er det ikke anbefalt størrelse til dekoblingskondensator. Anbefaling fra "Analog Devices" følges. Størrelser er listet i tabell 5 under.

Tabell 5: Dekobling kondensatorer

Enhet	Dekoblings kondensator
ATtiny 84	$0.1 \mu F$
ATtiny 85	$0.1 \mu F$
MCP 2515	$0.1 \mu F$
MCP 2551	$0.1 \mu F$
MCP 1703T - 250mA	$1 \mu F$
LP38691 - 500mA	$1 \mu F$
Oscillator - 16Mhz	$18pF$

Det er anbefalt å bruke minst  $4.7k\Omega$  pull-up resistor på reset pin iht. Microchip's AN2519 [51]. Valgte å heller bruke  $30k\Omega$  pull-up resistor på reset pin iht. ATtiny84 og 85 datablad [23, 52].

## Powersupply

Alle powersupply'er får sin input gjennom en "rectifier diode" som kan føre 30A i "riktig" retning og lukker ved strøm i "feil" retning [53]. For å beskytte programmereren slik at den aldri blir til å forsyne strøm ut på kretsen.

CAN bus kortet til RPIzw, og RPIzw får power fra "Supply Board". Dermed trengst bare to spenningsregulatorer. En til "ESC CAN module" og en til "Supply Board". For å bestemme supply til disse må det kartlegges hvor mye som kreves for å drive kretsen. Til eksempel trekker RPIzw i gjennomsnitt 160mA, og maksimalt 350mA under stress. Ved boot trekker den 200mA [54]. I tabell 6 er det verste tenkelige tilfelle av maksimalt strømtrekk stilt opp, basert på enhetenes respektive datablader [23, 52, 55–57]. Resultatet er riktige iht. datablad, men villedene for konfigurasjonen på disse kortene. Strømtrekk for ATtiny 84 og 85 er ikke 200mA, da ingen av pinnene blir brukt til å trekke strøm (max 40mA per I/O pin), det føres bare signal. De anslås derfor til å trekke max 100mA i denne konfigurasjon. RPIzw skal heller ikke trekke strøm fra sine pinner, dermed antas strømtrekk ved boot til å være av det høyeste for RPIzw.

Ny utregning gir følgende total strømtrekk, 185mA for "ESC CAN module" og 345mA for "Supply Board". Spenningsregulatorer ble derav bestemt til henholdsvis 250mA og 500mA. Input spenning fra 2.7V til 16V for ESC CAN module sin og 2.7V til 10V for Supply Board [49, 50].

Tabell 6: Anslatt maksimalt strømtrekk

Supply Board		ESC CAN module	
Enhet	Strømtrekk	Enhet	Strømtrekk
ATtiny 84	200mA	ATtiny 85	200mA
MCP2551	75mA	MCP2551	75mA
MCP2515	10mA	MCP2515	10mA
LM321	0.43mA		
RPIzw	350mA		
		TOT	285mA
TOT	635.43mA		

## Shunt

Shunt resistoren med verdi på  $0.001\Omega$  hadde god tilgjengelighet og ble valgt. Gruppen utforsket ideen å bruke en transistor til å bytte mellom å styre strømmen gjennom shunt, og direkte. Tanken om at det er mulig å spare energi ved å styre strømmen gjennom shunt 10% av tiden for å gjøre en diskret tilnærming til strømforbruk var interessant. Gruppen gikk bort i fra dette, fordi de aktuelle transistorene hadde en såpass høy "on resistance" at effekt-fortjenesten ville blitt minimal.

## Differensialforsterker

Shunt resistoren er så liten at det kan bli problemer å benytte en 10-bits ADC.

ATtiny84 sin 10 bit ADC gir at 0V relativ til referansespenning gir bitverdi på 0, mens 5V gir bitverdi 1023. ATtiny84 anbefaler ikke referansespenning høyere enn  $V_{cc} - 1\text{ Volt}$ , derav kan ikke batterispenningen (8.4V) benyttes som referanse på ADC. Antatt max strøm gjennom shunt er 25A. Eksempelet i 5.3 viser i ligning (5.18) at max strøm gjennom shunt-resistor tilsvarer et spenningsfall på 0.025V. Som igjen tilsvarer 0.5% av oppløsingen på 1024.

$$\frac{1024}{5V} \cdot 0.025V = 5.12 \quad (6.14)$$

$$\frac{5.12}{1024} \cdot 100 = 0.5\% \quad (6.15)$$

For å løse disse problemene benyttes en OP-AMP for å lese av ferdig forsterket spenningsfall med en pin på ATtiny84. LM321 SOT-32 fra Texas Instrument ble valgt, og koblet som vist i figur 10. Det er ønskelig å sette forsterkning slik at omrent 75% av oppløsningen benyttes. For å gjøre rom for at det skal kunne brukes større motorer som trekker mer.

Regner baklengs gjennom ligning 6.15 og 6.14 og får at forsterkning f må bli 150.

$$\frac{x}{1024} \cdot 100 = 75\% \quad \Rightarrow \quad x = 768 \quad (6.16)$$

$$\frac{1024}{5V} \cdot 0.025V \cdot f = x \quad \Rightarrow \quad f = 150 \quad (6.17)$$

Forsterkning settes til 150. Maximal output strøm fra batteri av C-raten er 32 Ampere. Som over shunt evt. ville gitt et spenningsfall på 0.032V. Ved å sette det spenningsfallet inn i ligning 6.14 med forsterkning på 150 får en ut 983.04. Som tilsvarer 96% av oppløsningen til 10-bit ADC'en. Dermed er supply kortet kompatibelt med hele batteriets C-rate.

Fra 5.19 sees det at forsterkning  $f$  er styrt av forholdet mellom  $R_2$  og  $R_1$ . Ønsket forsterkning er funnet, dermed kan resistorverdi for OP-AMP beregnes:

$$f = \frac{R_2}{R_1} = 150 \quad (6.18)$$

$R_1$  velges til  $10k\Omega$ . Ligning 6.18 kan dermed løses for  $R_2$ :

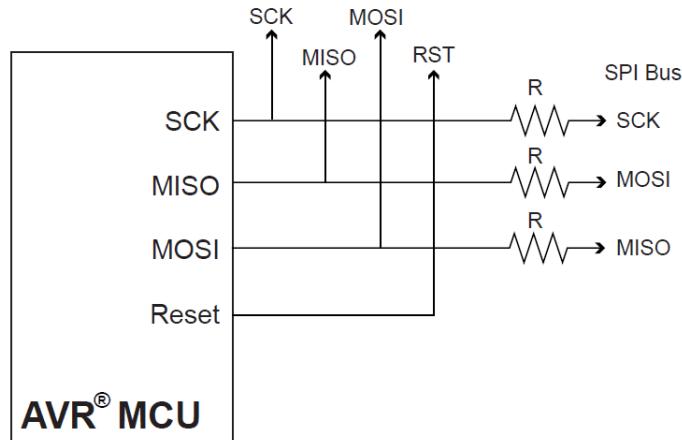
$$R_2 = 150 \cdot 10k\Omega = 1500\Omega \quad (6.19)$$

### Delt bruk av SPI programmerings linjer

Når flere enheter blir koblet til ISP linjene, må programmereren beskyttes fra enhver annen enhet enn AVR enheten, som kanskje vil prøve å drive linjen. Å koble resistorer i serie på SPI linjene som vist i figur 50 er den enkleste måten å oppnå dette på. AVR's "Microcontroller Hardware Design Considerations" (AN2519) anbefaler  $330\Omega$  resistorer, for å begrense input strømmen til 10mA for en supply volt (VCC) på 3.3V. [51]

For vår bruk, med 5V VCC kommer det frem av Ohm's lov i ligning 6.20 at det kreves  $500\Omega$  resistorer for å tilfredsstille anbefalt input strøm begrensing fra AVR.

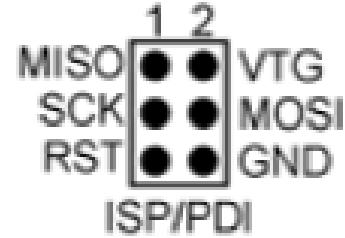
$$R = \frac{U}{I} = \frac{5V}{10 \cdot 10^{-3}A} = 500\Omega \quad (6.20)$$



Figur 50: [51] Utdrag fra AVR Microcontroller Hardware Design Considerations, kap 4.1.1

### Programmeringspinner

Videre ble en standard 6-pins kontakt fra Samtec valgt, med hensyn til pris, størrelse og enkelhet. Den ble koblet etter anvisning i AVR's AN2519 for å være kompatibel med ISP programmerere. Som vist i figur 51 for SPI grensesnitt mot ISP, for å enkelt kunne programmere kortene med en ICE programmerer [51].



Figur 51: [51] Utdrag fra AVR's AN2519

### 6.8.4 Eagle

#### Skjematikk

Skjematikk ble tegnet i Eagle og koblet i henhold til komponentenes respektive datablader. Deler av skjematikken ble utledet ved hjelp av prototyping i lab.

Skjematikk for alle kretskortene er vedlagt i Appendiks A.

#### Utlegg

Til de valgte komponentene er det brukt Eagle-biblioteker med footprints til utlegget som følger dokumentasjonen på komponenten. Ved valg av komponenter fra Farnell og Elfa distrelec ble komponenter med Eagle-bibliotek prioritert for å spare tid og øke kvaliteten på utlegget. Jobben med å legge utlegg for kretskortet ble da mye enklere, og "footprintet" mer troverdig, en om alle footprints skulle tegnes selv.

Mye tid har gått med på å få til et optimalisert design, hvor kondensatorer står plassert så nærmest mulig sine respektive powerpins, med direkte dekobling til jord. Samtidig som det skal være mulig å koble seg innpå kretskortet, med taktisk plasserte ledningskontakter/loddepunkt. Samtidig som det må være plass til alle strømbaner, uten å gå skrått inn på footprints eller ha skarpe 90° svinger.

Størrelse på strømbaner er kalkulert med en nettbasert sporbredde kalkulator fra Qorvo som nevnt i 5.3.1. Resultat for aktuelle strømtrekk er listet i tabell 7.

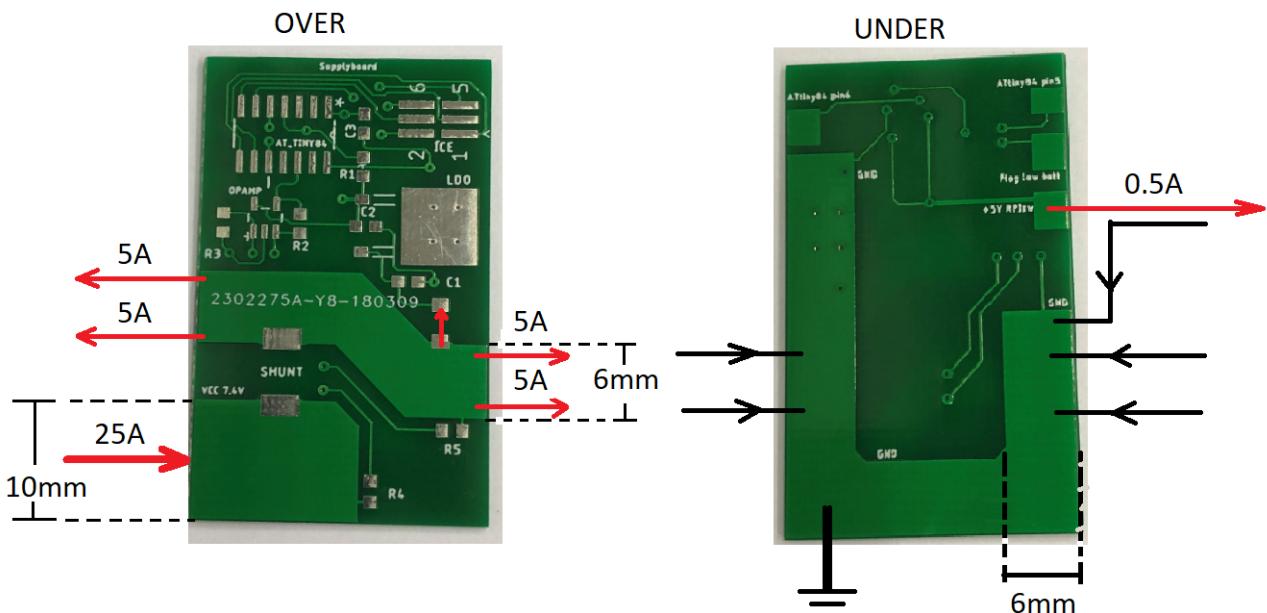
Tabell 7: Nødvendig sporbredde ved 1 tomme lengde og tillat temperaturstigning 10 °C

Strømtrekk	Tykkelse ved 1 $\frac{oz}{ft^2}$	Tykkelse ved 2 $\frac{oz}{ft^2}$
0.5 A	0.115 mm	0.054 mm
5 A	2.77 mm	1.38 mm
12.5 A	9.8 mm	4.57 mm
25 A	25.5 mm	11.9

25 amper er antatt absolutt max trekk gjennom SupplyBoard og det sees at for å håndtere den mengden kreves en sporbredde på 25.5mm. Legges det derimot inn tillatt temperaturøkning på 45°C holder det med 10 mm. Som er den nødvendige bredden for å håndtere 12.5A. 12.5A er en god antagelse på kontinuerlig strømtrekk.

Fra motortesten av EMAX i figur 45, så vi at for å hovre med thrust i overkant av 250 gram, trekkes det i underkant av 2 ampere per motor. Det tilsvarer totalt 8 ampere med alle motorene. Vi setter så av 2 amper til strømtrekk fra kretskortene. En sporbredde på 10mm gir da 2.5 amper til overs, som tilsvarer en sikkerhetfaktor på 25% for kontinuerlig strømtrekk. Det er greit, da tåler strømbananen pådrag fra motorene, og vi tillater noe temperatur økning ved peak strømtrekk. Peak strømtrekk vil bare skje i korte perioder og temperaturen antas å ikke stige betraktelig. Videre tas det også med i betrakningen at ledning inn her skal loddes på. Noe som øker tykkelsen. I tabell 7 ser vi at allerede med en tykkelse på 2  $\frac{oz}{ft^2}$  (75  $\mu m$ ) halveres nødvendig sporbredde.

Resultat av det ferdige designet av Supply Board er vist i figur 52. Der sees ferdig etset pcb fra Kina. Strømmen som trekkes er illustrert med røde piler for + og sort for  $\div$ , peak strøm og sporbredde til strømbaner er ført på.



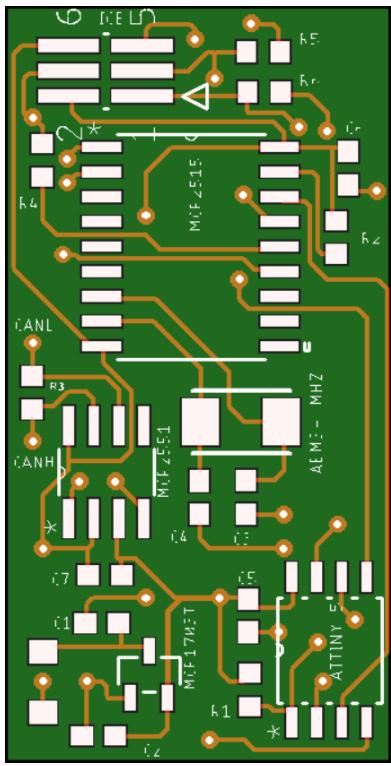
Figur 52: Supply Board, ferdig etset fra Kina, med påført strømtrekk

Utleget ble laget med et strengt sett med DRC-regler. Basert på fabrikkens begrensninger [58]. Det vil si at en ikke får lov av programmet til å gjøre feil, som f.eks å legge strømbaner for nærme et lodddepunkt. Det kan spare en for mye arbeid, eller at en bestiller noe som leverandøren ikke klarer å lage og får et kort med masse kortslutninger i verste fall.

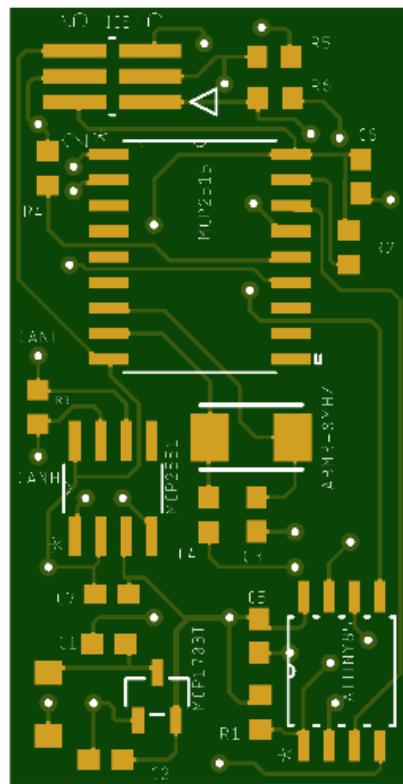
Vanlig tykkelse på kretskort er 1.6mm, og er den tykkelsen Universitetet kan stille med. Gruppen valgte 1mm tykkelse som et kompromiss mellom å ha litt tykkelse til å håndtere varmen fra loddingen og å ha et vektredusert kort.

### Gerber og bestilling

Universitetet har utstyr for å frese ut egne pcb-kort selv. Våre krav til størrelse gjorde derimot at vi heller ønsket å bestille kort fra Kina. Firmaet heter JLCPCB, en av Kinas største bedrifter for prototyper av PCB. For å bestille krever de å få tilsendt en "gerber" fil på utleget. Gerber lages i Eagle og lastes opp på nettsiden til leverandør. Som en del av kvalitetsikringen, for å unngå feilkoblinger og andre feil med utleget, brukes egen programvare for å inspisere gerber filene. Gruppen sökte om tilgang til programvaren til Macaos og fikk ta i bruk den. Det viste seg derimot at JLCPCB stiller med online programvare for å inspisere gerber filer. Den ble valgt å bruke i tillegg til Macaos, da tilliten til firmaets egen "gerber viewer" var større. Det er naturlig å anta at resultatet vil ende opp som i pcb-leverandørens programvare. Kontra å bruke en ekstern programvare som Macaos.



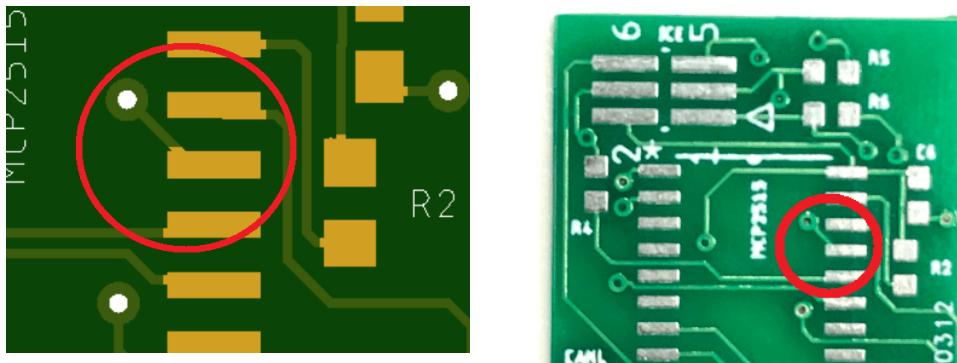
(a) Macaos programvare



(b) JLCPCB online programvare

Figur 53: Inspeksjon av utlegg med gerber viewer

Til tross for at alle kort var kontrollert i gerber-viewer ble en uplanlagt skarp inngang mot et loddepunkt oppdaget når kortene ankom fra Kina. Når kobberbaner strekkes i utlegget ønskes det ikke å gå skrått inn på footprints. Dette fordi en skaper kanter som er vanskelige å fysisk klare å lage for fabrikken og med det øker sjansen for å få en feil på kortet. Uhell som dette kan være vanskelige å oppdage i software'en som brukes for å designe, da det er lett å se seg blind. Dette poengterer viktigheten av kvalitetskontroll før PCB design sendes til produksjon. Dette er ikke et reelt problem for oppgaven, da fabrikken klarte å produsere skråbanen og kortet fungerer utmerket. Det kommenteres likevel i rapporten for å understreke fordelene med å benytte kvalitetskontroll. Komplekse design har lav terskel for glipper og uten gerber viewer ville kortet hatt enda flere.



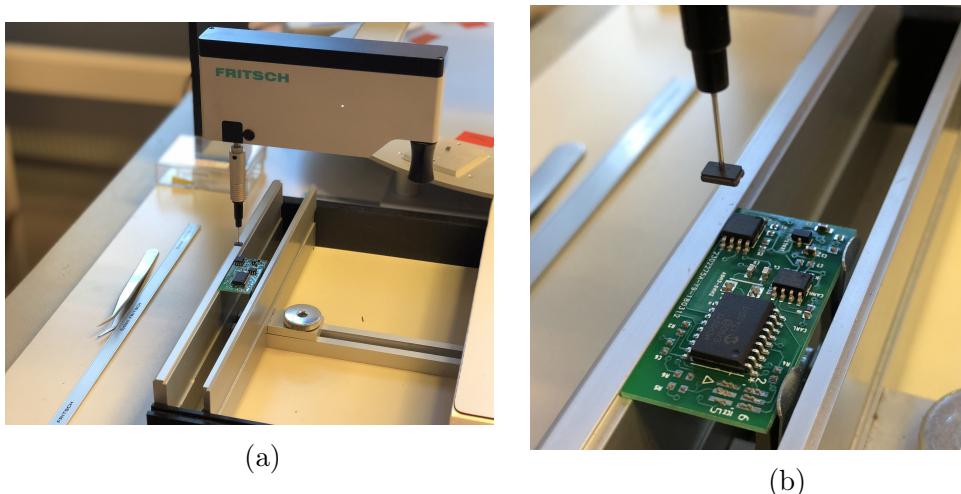
(a) Tabbe synlig i JLCPCB's online viewer

(b) Tabbe på kort

Figur 54: Svikt i kvalitetsikring

### 6.8.5 Sammenstilling og testing

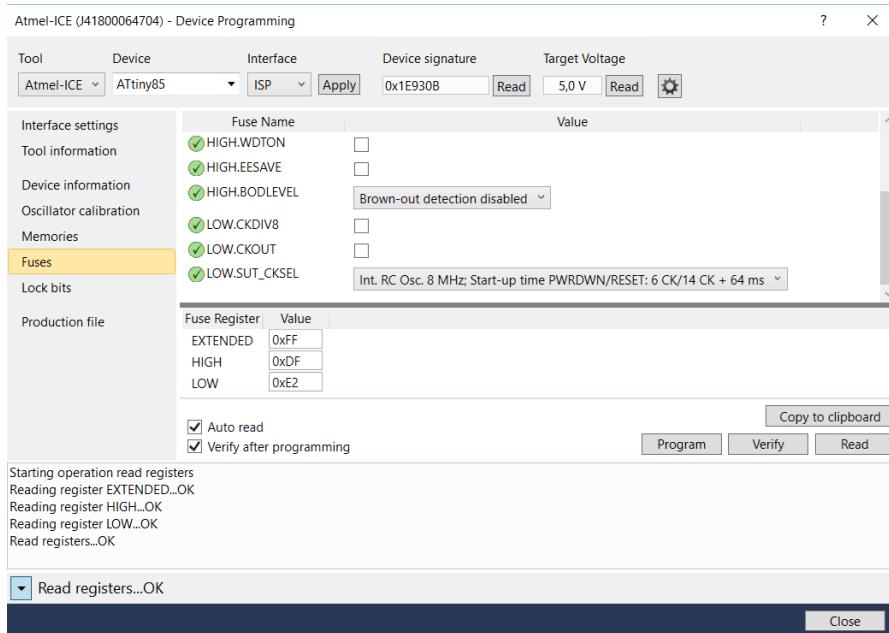
Med komponenter ned til 0603 størrelse er lodding for hånd utfordrene. Universitetets elektro lab har utstyr til å håndtere prototyping av kretskort med disse størrelsene. Først legges det ut pasta på pcb-brettet ved hjelp av pedalstyrt trykkluftspumpe, så plasseres komponentene med et verktøy som bruker vakuum for å holde komponentene, og slipper når de presset mot pcb-brettet. Når alle komponentene er plassert riktig plasseres kortet i en reflow ovn som baker kortet ved å gjennomføre en temperatur sekvens opp til loddepastaens flyttemperatur. Da trekker den seg på plass mot loddepunktene.



Figur 55: Plassering av komponenter på pcb

Etter sammenstilling måtte kortene testes. Kretsen ble målt over for å sjekke for kortslutning, og videre ble spenning koblet på og vi leste av 5V etter regulator. Enhets signatur ble så avlest med ønsket resultat som vist i figur 56. Dersom kommunikasjonen fra ATtiny85 til atmel ICE hadde vært koblet feil, ville vi ikke vært i stand til å lese av denne signaturen. Dette

bekrefter at SPI-kommunikasjon for programmering fungerer, samt at ATtiny85 håndterte den høye temperaturen i reflow ovnen.



Figur 56: Avlesning av enhet signatur i Atmel studio

## 6.9 Programmering

### 6.9.1 Oppsett av CAN bibliotek

Noe mer av grunnlaget bak valget av ATtiny84/85 som mikrokontollere var biblioteket "avr-can-lib" [47] laget for bruk i "Eurobot" konkurransen av Roboterclub Aachen. Biblioteket støtter tilsvynelatende CAN-bus kommunikasjon for ATtiny84/85 ved bruk av mcp 2515 og mcp 2551 CAN-tranciever/kontroller.

Biblioteket er kompatibelt med flere typer mikrokontrollere i avr-familien. En "config.h" fil må modifiseres for kompatibilitet med ATtiny85. Her deklarerer type CAN kontroller, valgt pin for chip select og eventuell støtte for utvidet CAN ID. I bibliotekets "makefile" blir mikrokontroller og klokkefrekvens deklarert. Komplett config.h og makefile kan sees i digitale vedlegg.

"WinAVR" var anbefalt som verktøy til å bygge selve biblioteket. Windows 10 og "WinAVR" viste seg å ha kompatibilitets-problemer, så etter råd fra veileder valgte vi heller å benytte innebygget funksjonalitet i Ubuntu for å bygge biblioteket. Her bygges biblioteket ved å navigere til aktuell mappe fra terminalen, for deretter å utføre kommandoen "make lib". Dette skal generere en .a fil som kan linkes til ønsket kompilator før krysskompilering til eksempelvis ATtiny85.

Ved første forsøk på å bygge biblioteket i Ubuntu dukket feilmeldingene i figur 57 opp. Alle feilmeldingene omhandler "spi.h" filen. ATtiny85

```
spi.h: In function 'spi_start':
spi.h:75: error: 'SPDR' undeclared (first use in this function)
spi.h:75: error: (Each undeclared identifier is reported only once
spi.h:75: error: for each function it appears in.)
spi.h: In function 'spi_wait':
spi.h:80: error: 'SPSR' undeclared (first use in this function)
spi.h:80: error: 'SPIF' undeclared (first use in this function)
spi.h:83: error: 'SPDR' undeclared (first use in this function)
```

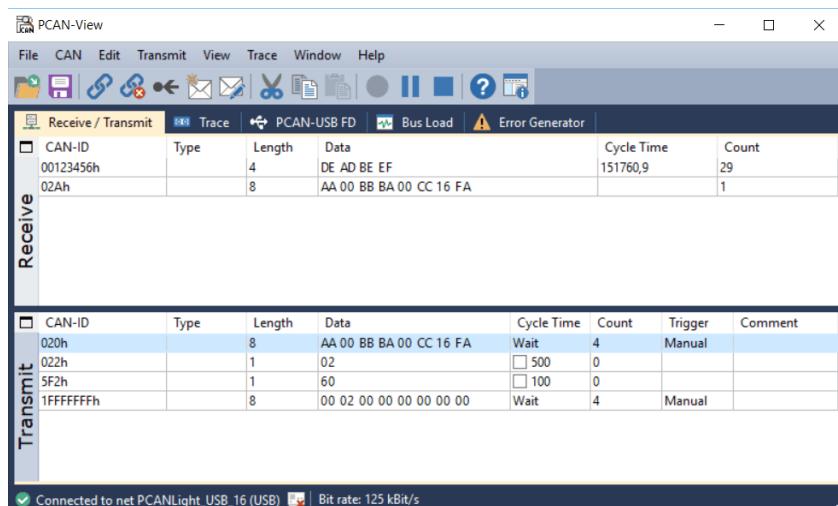
har ikke hardware SPI støtte, men har et USI grensesnitt som gir støtte for "software-SPI" med noen få linjer kode til hjelp. "avr-can-lib" har en metode for "software-SPI" innebygget som defineres om man velger ATTiny85 som mikrokontroller, men det var tydelig noe rundt denne definisjonen som ikke fungerte som det skulle.

Feilkilden viste seg å være at "spi.h" var inkludert i "mcp2515\_private.h" før "USE\_SOFTWARE\_SPI" var definert. Problemet ble løst ved å flytte inkluderingen av "spi.h" lenger ned i programmet, og dette førte til at biblioteket kunne bygges uten feilmeldinger. Denne feilen skyldes antakeligvis at skaperne av biblioteket i hovedsak hadde brukt en ATmega32 som har hardware SPI støtte, og dermed utført mangelfull testing ved implementasjon av støtte for software-SPI mikrokontrollere.

"libcan.a" filen som ble generert kunne linkes til et "Atmel studio" prosjekt ved å klikke "project » options » toolchain » libraries" og velge filplassering for .a filen.

Et enkelt eksempelprogram for sending og mottagelse av CAN-meldinger ble kompilert og lastet over til ATTiny85 med suksess. Et PCAN-USB adapter fra PEAK systems med tilhørende PCAN-view software ble brukt for å kontrollere innkommende og sende utgående CAN meldinger. Her kunne vi se at meldingene fra eksempelprogrammet ikke ble mottatt, og om vi prøvde å sende en CAN melding genererte dette feilmeldingen "bus inactive".

Aktuering av en enkel led ble brukt for å undersøke om hele programmet kjørte uten å henge seg opp i noen uendelige løkker. Feilsøking viste at et interrupt fra mcp2515 var definert til å bli mottatt på PB2, samme pinne som brukes til SCK i usi/spi grensesnittet. Implementasjonen av interrupt ble fjernet, og biblioteket måtte bygges på nytt. Endringen resulterte i at ATTiny85 veldig godt kunne sende og motta CAN meldinger. Første suksessfulle meldinger mottatt og sendt mellom ATTiny85 og PCAN-USB adapter er vist i figur 58



Figur 58: Fungerende kommunikasjon mellom PCAN-USB adapter og ATTiny85

## 6.9.2 Programmering

Som beskrevet i 6.8.1 skal CAN modulen motta CAN melding fra RP1zw og sende ut et korresponderende PWM signal til motorenens ESC. Turnigy Multistar ESC'en som er valgt, støtter pulstog-frekvensens opp til 480Hz [59]. 250 Hz ble valgt som ønsket pulstog-frekvens. Dette er godt innenfor spesifikasjonene til motorenens ESC samtidig som det er raskt nok til ikke å svekke regulering og stabilisering.

CTC modus som er beskrevet i 5.4 ble tatt i bruk. Hastighet signalet ATTiny-mikrokontrolleren mottar via CAN bus vil være 0-100 med step på 1. Dermed bør det forsøkes å opprettholde denne oppløsningen ved å velge periodetid på interrupt flagg samsvarende til 100 pådragsnivå. Utregning av nødvendig periodetid basert på formel 5.21 er vist under.

$$\text{Antall pådragsnivå} = \frac{\text{Max pulsbredde} - \text{Min pulsbredde}}{T_{OC0A}} \quad (6.21)$$

$$T_{OC0A} = \frac{\text{Max pulsbredde} - \text{Min pulsbredde}}{\text{Antall pådragsnivå}} \quad (6.22)$$

$$T_{OC0A} = \frac{2000\mu s - 1000\mu s}{100} \quad (6.23)$$

$$T_{OC0A} = 10\mu s \quad (6.24)$$

Mikrokontrollerens klokkefrekvens er originalt 8 MHz, men det er satt en prescaler på 1/8 fra fabrikk som gir den en klokkefrekvens på 1 MHz. For å få høyest mulig verdi på OCR0A ble prescaleren fjernet ved å endre argumentet som blir brukt av Atmel ICE ved opplastning av program fra "-Ulfuse:w:0x62" til "-U lfuse:w:0xe2". Beregningene i ligning 6.24 gir periodetid  $10\mu s$  som tilsvarer  $100\ 000 Hz$ . Timer 0 har en prescaler for justering av verdi på OCR0A om det kreves lengre periodetider. Tilgjengelig verdier er 1, 8, 64, 256 og 1024. Da  $T_{OC0A}$  har frekvens 80 ganger lavere en systemklokke (8MHz) er kun prescaler 1, 8 og 64 aktuelle. Beregninger ved bruk av forskjellige prescalers basert på formel 5.20 er vist under.

$$f_{OC0A} = \frac{f_{clk\_I/O}}{N \cdot (1 + OCR0A)} = T_{OC0A}^{-1} \quad (6.25)$$

$$OCR0A = \frac{f_{clk\_I/O}}{N \cdot f_{OC0A}} - 1 \quad (6.26)$$

$$OCR0A = \frac{8\ 000\ 000}{1 \cdot (10\mu s)^{-1}} - 1 = 79 \quad (6.27)$$

$$OCR0A = \frac{8\ 000\ 000}{8 \cdot (10\mu s)^{-1}} - 1 = 9 \quad (6.28)$$

$$OCR0A = \frac{8\ 000\ 000}{64 \cdot (10\mu s)^{-1}} - 1 = 0,25 \quad (6.29)$$

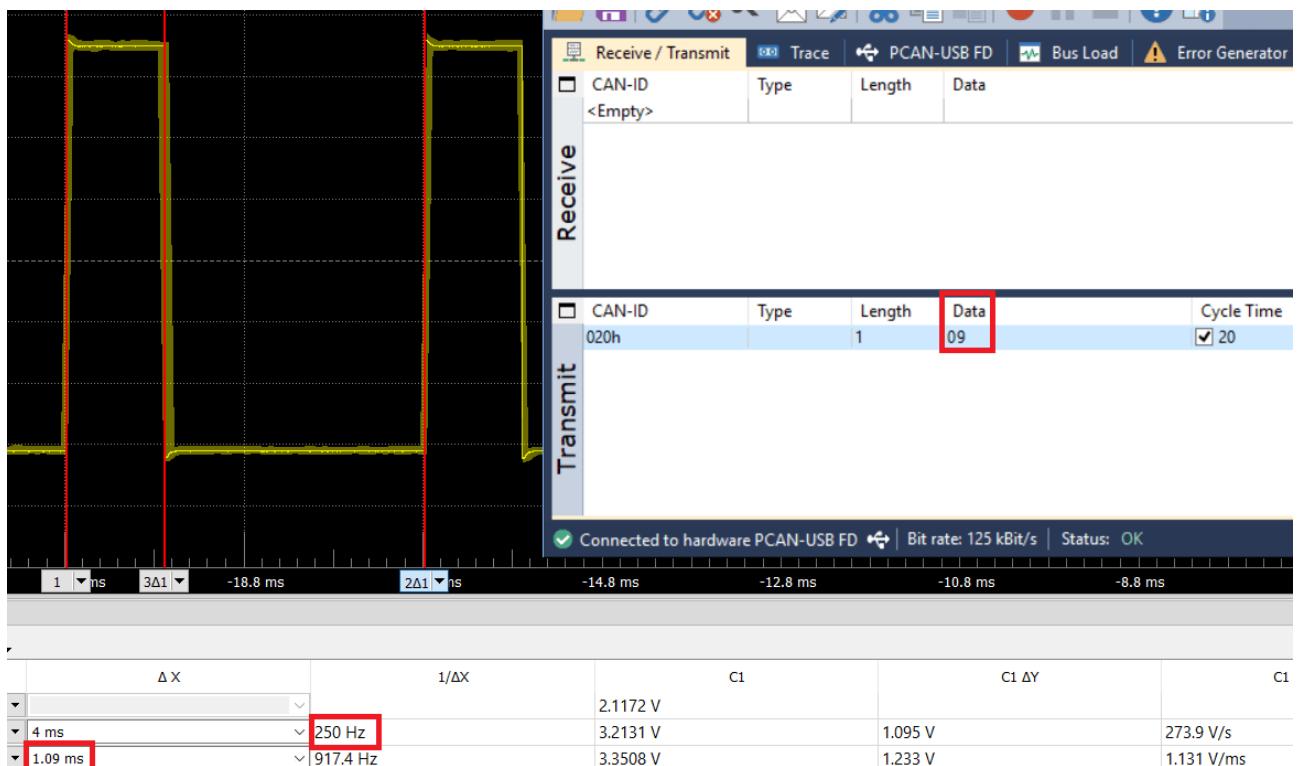
Timer0 er en 8-bit timer. Med konfigurasjon hvor prosessor kjøres uten prescaler, inkrementeres timer0 for hvert "tikk" fra systemklokken frem til den når sin toppverdi, som er  $2^8$  (255). Da OCR0A er verdien Timer0 sammenlignes med for å time pwm pulstoget, er det ønskelig med OCR0A så nærmest 255 som mulig. Flere tikk fra systemklokke mellom hver interrupt sikrer at mikrokontrolleren rekker å fullføre alle sine oppgaver før neste interrupt inntreffer. På bakgrunn av ligningene 6.26 - 6.29 velges da prescaler 1, og 79 brukes som verdi for OCR0A. Dette for å oppfylle ønsket oppløsning på 100 pådragsnivå.

Da interrupten trigger med periodetid  $T_{OC0A}$  ( $10\mu s$ ) kunne antall interrupts enkelt beregnes for å oppnå ønsket pulstogfrekvens på 250 Hz.

$$\text{Antall interrupts} = \frac{T_{pulstog}}{T_{OC0A}} = \frac{(250\text{Hz})^{-1}}{10\mu s} = \frac{4000\mu s}{10\mu s} = 400 \quad (6.30)$$

En variabel "interruptTeller" inkrementeres i interrupt rutinen. Når denne har telt 400 ganger resettes telleren, og generering av neste pulstog begynner. "interruptTeller" inkrementeres med fast rate opp til 400 uavhengig av setpunkt mottatt fra CAN melding, for å sikre en robust frekvens på 250Hz.

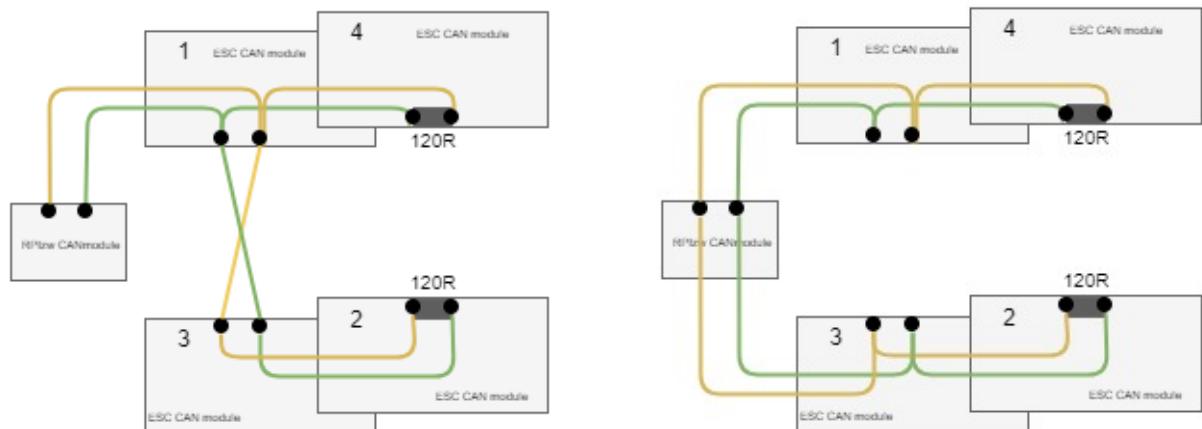
For å kontrollere at beregningene ga ønsket resultat ble det koblet til et oscilloskop. En CAN melding med data "09" ble sendt. Dette tilsvarer 9% pådrag, og skal generere pulstog som er "høy"  $1090\mu s$  og "lav" de resterende  $2910\mu s$ . Testen ga tilfredstillende resultat, og er vist i figur 59



Figur 59: Kontroll av pulstog-frekvens og duty cycle

### 6.9.3 Oppkobling av CAN

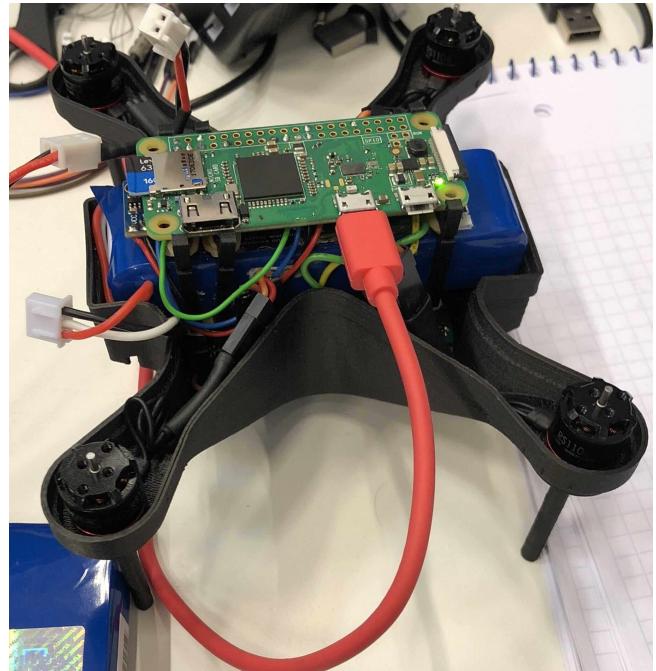
Kobling av CAN nettet kunne gjøres på to måter som vist i fig 60. Termineringsmotsand er lagt på ESC CAN module. For å spare mest mulig vekt av ledning ble konfigurasjonen vist i figur 60a. Da RPizw er montert over batteri har den det lengste ledning strekket og det er da naturlig å bare trekke en avstikker opp dit, fremfor å la den være "senter" i loopen.



Figur 60: Oppkobling for CAN

### 6.9.4 Funksjonstesting

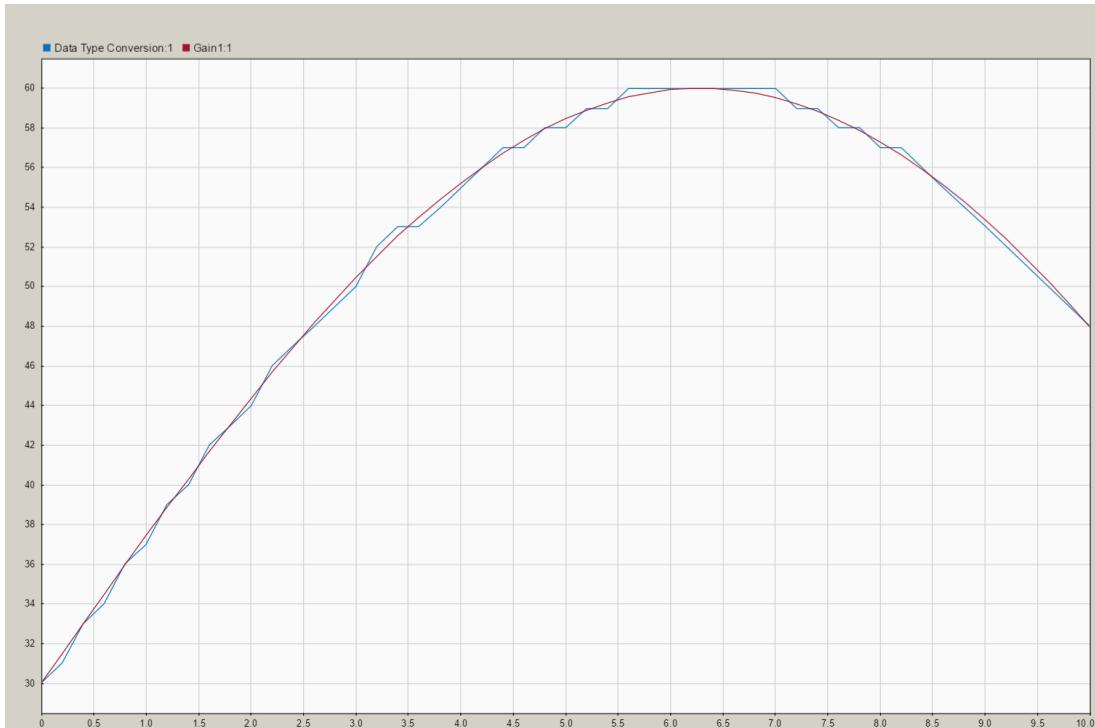
For en komplett test av funksjonaliteten til kretskortene ble kretskortene, batteri, RPizw og motorer med tilhørende ESC koblet sammen. Motor 1-4 (se figur 18) ble tildelt CAN-ID 0x20-0x23. Hele systemet fungerte på en tilfredstillende måte når hver enkelt motor ble forsøkt aktuert fra terminalen på RPizw. Dette ble gjort ved å sende fire forskjellige CAN meldinger med samme pådragskommando, men variere meldings-ID fra 0x20-0x23. Testen bekreftet at hele systemet fungerte som ønsket da alt av hardware og software må fungere for å oppnå ovennevnt resultat.



Figur 61: Ferdig oppkobling

Fra Simulink ble det generert en ROS-node som gir pådrag til motorene tilsvarende en sinusbølge som varierer fra 0-60, for å se hvordan systemet responderte på et varierende motorpådrag. Med stoppeklokke kunne det valideres at motoren fulgte den genererte sinusbølgjen, men det kunne tydelig høres at motorpådraget ikke fulgte en jevn kurve.

ROS-emnet motor-pådragene blir publisert på inneholder fire uint32 setpunkt, et til hver motor. Desimaltall hadde blitt valgt til å rundest av til nærmeste integer, og det var dette som genererte den hørbare kvantiseringsfeilen. Problemet kan enkelt løses ved å gange motor-settpunktet med 100 før konverteringen til uint32, for deretter å dele på samme faktor på ESC CAN module sin attiny mikrokontroller. Dette sørger for at avrundingsfeilen som gjøres ved konvertering er mindre med en faktor på  $10^{-2}$  og forskjellen vises i figur 62



Figur 62: Kvantiseringsfeil

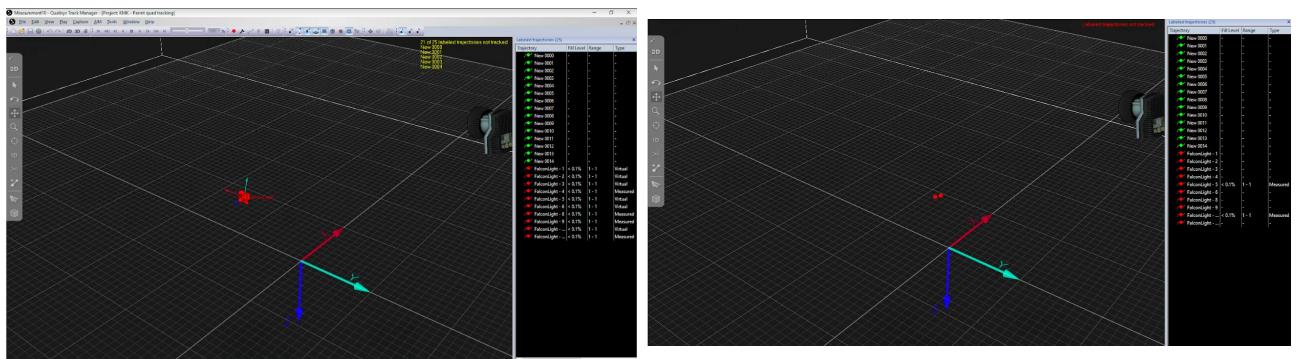
## 6.10 Qualisys

For å spore dronen i motion capture ble den utstyrt med 9 markører som vist i figur 63. De er som vist plassert symmetrisk på skroget for opprettholde vektbalanse og dermed kunne neglisjere treghetsmomentbidraget de tilfører. For å definere et nytt legeme i Qualisys software legges markørenes koordinater inn relativ dronens massemidtpunkt. Enkelte markører må legges inn som "virtual" for at systemet skal ha kontroll på dronens orientasjon.



Figur 63: Markører reflekterer kamerablitz

Oppnak av posisjons innhenting ble forsøkt med begrenset suksess. Systemet mister til stadighet kontroll på en eller markører, noe som fører til at posisjon og orientering ikke kan hentes ut til en hver tid. Slike diskontinuiteter er svært uheldig om dataen skal brukes til flyvning. Til vårt formål kreves en jevn strøm med troverdig data, publisert med en høy rate.



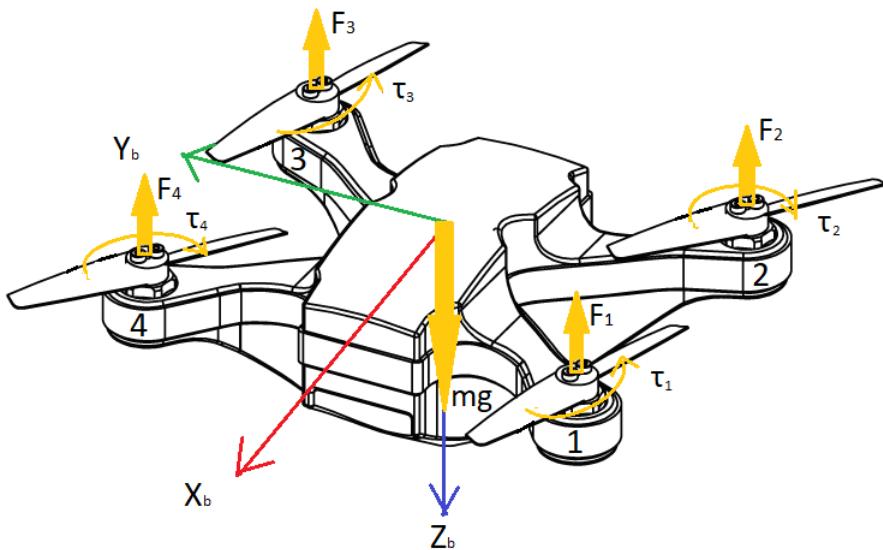
(a) Suksessfull innhenting av posisjon og orientering av drone

(b) Systemet oppfatter kun 2 av 9 markører

Figur 64: Sporing av 3D-printet prototype i MotionLab

## 6.11 Matematisk dronemodell

I dette kapittelet utledes ligningene som danner den matematiske dronemodellen. For å bestemme hvordan dronen oppfører seg ved flyvning må (tilnærmet) alle krefter og momenter som påvirker den være med i modellen. For å finne ut hvordan krefter og momenter i kroppsrammen påvirker dronens flyvning, bruker vi teorien om roterende referanserammer i delkapittel 5.9.2, i tillegg til resten av kapittel 5.9. I slutten av dette kapitellet er det kort lagt fram hvilke påvirkninger som ikke er tatt med i modellen.



Figur 65: Dronekropp med påførte krefter og momenter

### 6.11.1 Kraftligning

Her vil vi først se på sammenhengen mellom kreftene med utgangspunkt i massemidtpunktet til dronen  $O_b$ . Fra Newtons lover har vi:

$$F = \frac{d}{dt}(mv) \quad (6.31)$$

For vår drone er massen konstant og vi kan skrive:

$$F = m \frac{d}{dt}(v) = ma \quad (6.32)$$

Det er dermed interessant å se på akselerasjonen til  $O_b$ . I ligningene under vil  $V_b$  være hastigheten til dronen med hensyn på kroppsrammens akser. Om vi observerer akselerasjonen av dronekroppen fra den stillestående rammen, hvor  $\omega_b$  er rotasjonen av dronekroppen rundt dens egne akser med hensyn på sistnevnte ramme, får vi følgende, i henhold til beskrivelse i delkapittel 5.9.2:

$$\left[ \frac{d}{dt} V_b \right]_n = \left[ \frac{d}{dt} V_b \right]_b + \omega_b \times V_b \quad (6.33)$$

Det bemerkes at bemerkelsen n, her ikke betyr med hensyn på NED-rammens akser, men med at vi observerer fra den stillestående ramme. Om vi nå ser på summen av krefter i kroppsrammen, observert fra den stillestående-framen får vi:

$$\sum F_b = m \left[ \frac{d}{dt} V_b \right]_n = m \left( \left[ \frac{d}{dt} V_b \right]_b + \omega_b \times V_b \right) \quad (6.34)$$

Sluttresultatet, med tanke på de krefter som virker i kroppsrammen blir da:

$$m a_b = m \left[ \frac{d}{dt} V_b \right]_b = \sum F_b - m(\omega_b \times V_b) \quad (6.35)$$

Som sett av figur 65 vil kraftvektorene generert av propellene følge dronekroppen, de er altså kroppsfaste, mens kraften fra jorda på dronen vil alltid peke ned. Den fullstendige ligningen blir da:

$$m \left[ \frac{d}{dt} V_b \right]_b = \begin{bmatrix} 0 \\ 0 \\ -F_1 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -F_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -F_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -F_4 \end{bmatrix} + R_n^b \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} - m(\omega_b \times V_b) \quad (6.36)$$

### 6.11.2 Momentligning

På samme måte som for kraftligningen tar vi her utgangspunkt i massemidtpunktet til dronen  $O_b$ . I tillegg vil også kroppsrammen her være en roterende referanseramme og vi må bruke teorien fra delkapittel 5.9.2.

Som kjent er momentet  $M$  gitt av den deriverte av det angulære momentet med hensyn på en stillestående ramme:

$$M = \dot{L} \quad (6.37)$$

Hvor  $L$  er det angulære momentet. Når vi som sagt tar utgangspunkt i massemidtpunktet, blir det angulære momentet gitt av ligningen:

$$L = I\omega \quad (6.38)$$

$I$  er treghetsmoment-matrisen og  $\omega$  er rotasjonshastigheten til den roterende referanserammen sett fra en stillestående referanseramme. Det bemerkes også at treghetsmomentet om de ulike aksene i kroppsrammen er konstant, altså er  $I$  konstant:

$$M = I\dot{\omega} \quad (6.39)$$

Tar man summen av momentene som virker på dronekroppen, sett i forhold til en stillestående ramme, vil vi ved bruk av 5.9.2 få følgende:

$$\sum M_b = \left[ I \frac{d}{dt} \omega_b \right]_n = \left[ I \frac{d}{dt} \omega_b \right]_b + \omega_b \times (I\omega_b) \quad (6.40)$$

Her ser vi at endringen i angulært moment, sett fra den stillestående rammen, tilsvarer det angulære momentet observert av en observatør i kroppsrammen, i tillegg til det siste ledet i ligningen som skyldes at at vektorene som det angulære momentet utgjør også endrer orientering i forhold til den stillestående rammen.

Løser vi for det angulære momentet sett fra kroppsrammen får vi:

$$I \left[ \frac{d}{dt} \omega_b \right]_b = \sum M_b - \omega_b \times (I\omega_b) \quad (6.41)$$

Sluttført kan dette skrives som:

$$I \left[ \frac{d}{dt} \omega_b \right]_b = \vec{r}_1 \times \begin{bmatrix} 0 \\ 0 \\ -F_1 \end{bmatrix} + \vec{r}_2 \times \begin{bmatrix} 0 \\ 0 \\ -F_2 \end{bmatrix} + \vec{r}_3 \times \begin{bmatrix} 0 \\ 0 \\ -F_3 \end{bmatrix} + \vec{r}_4 \times \begin{bmatrix} 0 \\ 0 \\ -F_4 \end{bmatrix} + \tau_b - \omega_b \times (I\omega_b) \quad (6.42)$$

Hvor  $\vec{r}_{1\dots 4}$  er posisjonsvektorene ut til de fire motorene, med utgangspunkt i massesenteret/ $O_b$ .  $\tau_b$  er momentene motorene genererer i kroppsrammen.

Tar vi videre også med de gyroskopiske momentbidragene forårsaket av motorene får vi:

$$I \left[ \frac{d}{dt} \omega_b \right]_b = \vec{r}_1 \times \begin{bmatrix} 0 \\ 0 \\ -F_1 \end{bmatrix} + \vec{r}_2 \times \begin{bmatrix} 0 \\ 0 \\ -F_2 \end{bmatrix} + \vec{r}_3 \times \begin{bmatrix} 0 \\ 0 \\ -F_3 \end{bmatrix} + \vec{r}_4 \times \begin{bmatrix} 0 \\ 0 \\ -F_4 \end{bmatrix} + \tau_b + M_{gyro} - \omega_b \times (I\omega_b) \quad (6.43)$$

### 6.11.3 Gyroskopsisk moment

I ligning 6.43 er det gyroskopiske momentbidraget tatt med som resultat av de fire motorenes rotasjon, samt dronekroppens egen rotasjon.

Under tiltenkt banefølgning i labvolumet vil endringen av motorhastighet foregå relativt sakte, slik at antagelse om at endring i angulært moment er lik null, vil være en god tilnærming. Samtidig vil også motorenes vinkelhastighet være vesentlig større enn dronekroppens vinkelhastigheter, noe som også er i henhold til antagelse i delkapittel 5.9.4.

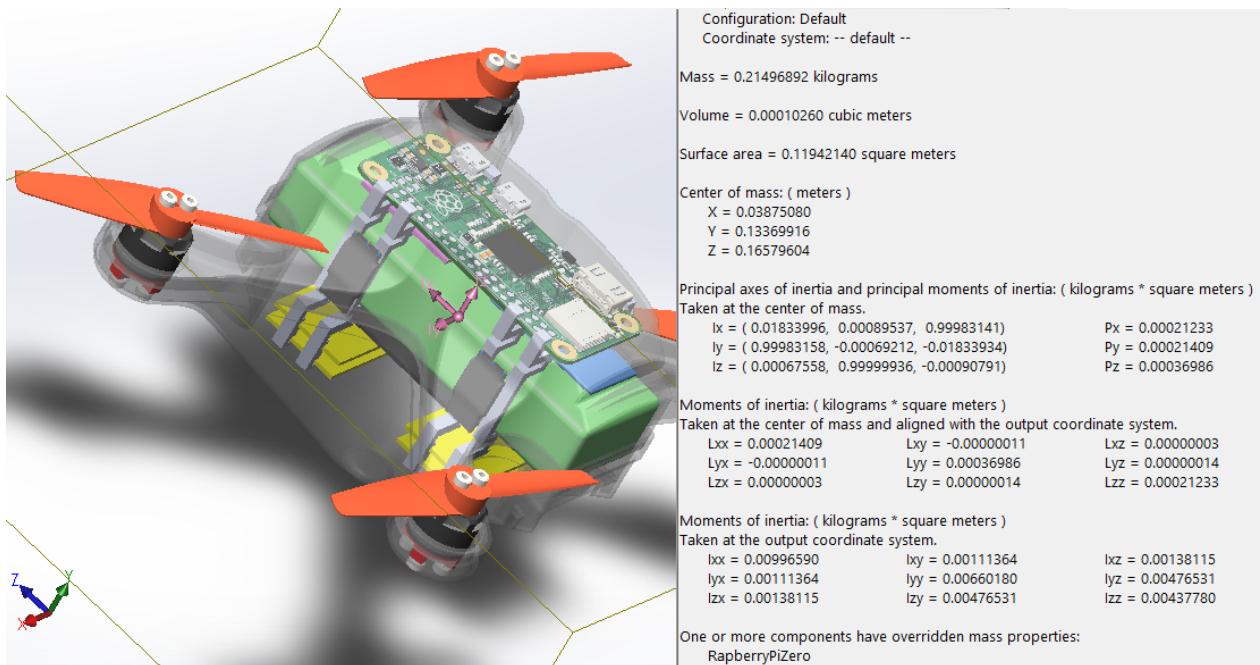
Med utgangspunkt i delkapittel 5.9.4 og figur 18, kan vi utlede det gyroskopiske momentbidraget ( $M_{gyro}$ ) motorene gir i kroppsrammen. Dette resulterer i følgende:

$$M_{gyro} = \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ I_{mp} * (2\pi/60) * (\omega_1 - \omega_2 + \omega_3 - \omega_4) \end{bmatrix} \quad (6.44)$$

Hvor  $\omega_{1\dots 4}$  er motorenes rotasjonshastighet i rpm.

#### 6.11.4 Trehetsmoment-matrisen

Ved bruk av vår SolidWorks-modell av dronen kan vi legge inn materiale og egenvekt på komponenter. Her foreligger også muligheten til å hente ut treghetsmomentene for den aktuelle modellen.



Figur 66: Dronens treghetsmomenter

Ser vi på treghetsmoment-matrisen (som tar utgangspunkt i massesenteret og koordinataksene i nedre venstre hjørne) i figur 66, kan vi konkludere følgende: Største verdi utenfor matrisens diagonal utgjør 0,65% av minste verdi på diagonalen. Det vil si at vi gjør marginale feil ved å si at koordinataskene er prinsipielle akser. I dette aksesystemet er y-aksen derfinert i motsatt retning av  $z_b$ , x-aksen i motsatt retning av  $y_b$  og z-aksen i samme retning som  $x_b$ . Dette impliserer at våre koordinatasker også er prinsipielle akser. Transformerer verdiene over til kroppsrammene akser:

$$I = \begin{bmatrix} 0,00021233 & 0 & 0 \\ 0 & 0,00021409 & 0 \\ 0 & 0 & 0,00036986 \end{bmatrix} \quad (6.45)$$

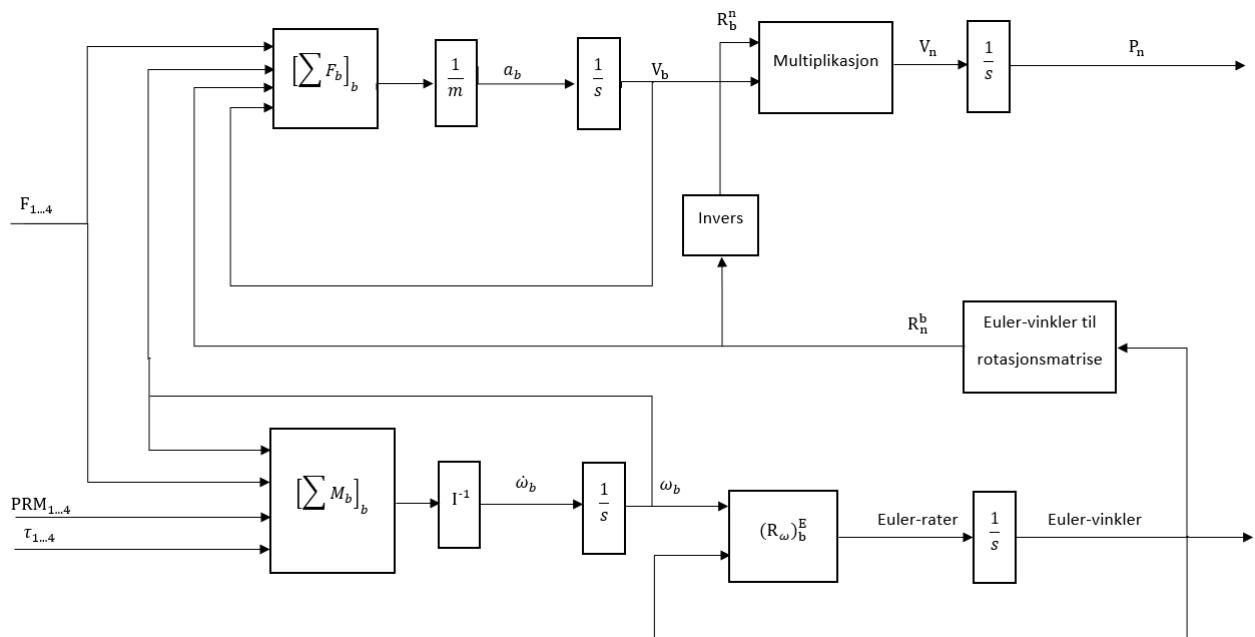
For vår dronemodell vil matrisen over bli brukt. Det bemerkes imidlertid at disse verdiene ikke vil være helt nøyaktige sammenlignet med den fysiske modellen da den ikke tar høyde for blant annet ledningsnettet.

### 6.11.5 Andre system påvirkninger og forstyrrelser

For dette prosjektet er dronen tiltenkt flyvning i lav hastighet i et lukket innendørs miljø. For flyvning i lav hastighet er det her antatt at luftmotstand og aerodynamiske effekter har neglisjerbar effekt på drones oppførsel. Samtidig er støypåvirkninger i form av vind utelukket.

Videre er det antatt at skroget er uendelig stift, samt at motorene er montert perfekt, slik at alle kraftvektorene er parallelle med dronekroppens z-akse ( $z_b$ ). Det vil si at det ikke er tatt hensyn til utbøyninger og vridninger ved flyvning i denne modellen, da det i 6.2.1 kan sees fra simuleringer at utslaget fra påkjenningene er svært små. Dette ses i sammenheng med at dronekroppen er designet slik at disse effektene skal være minimale og neglisjerbare.

### 6.11.6 Grafisk dronemodell

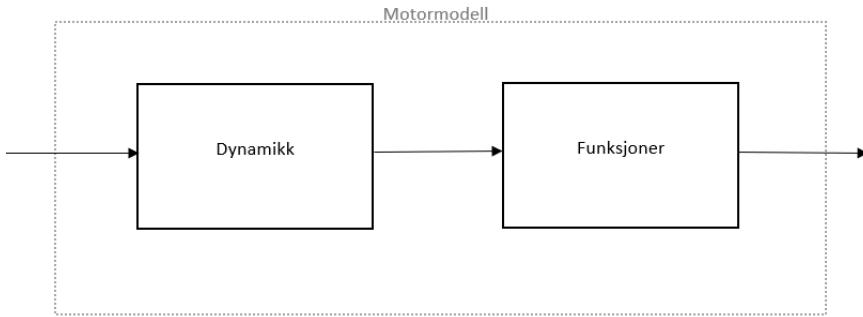


Figur 67: Blokkskjema av dronens dynamikk

Figuren over viser et blokkskjema av dronens dynamikk. I denne modellen inneholder sumasjonsblokkene henholdsvis ligning 6.36 og 6.43, hvor blokkenes output tilsvarer ligningenes venstre side. Simulink-modellen av dronens dynamikk er basert på dette diagrammet, og er vedlagt i appendiks. Modellen er teste og verifisert mot Matlabs egen modell, outputene fra de to modellene er identisk ved simulering.

## 6.12 Motormodell

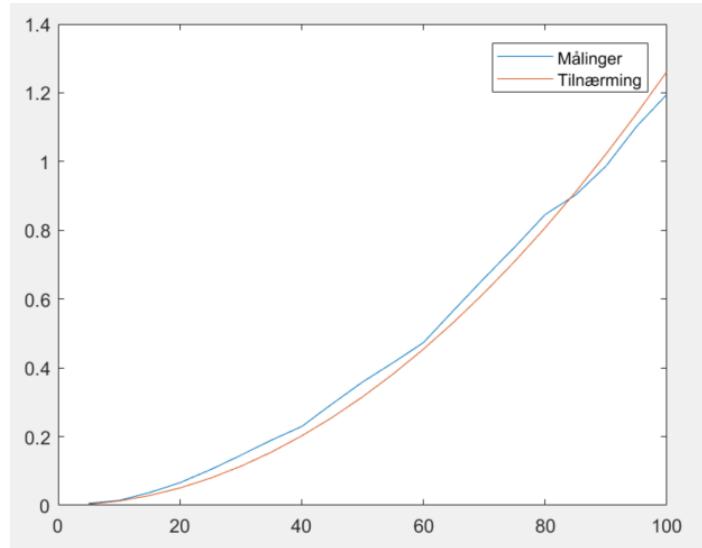
For vår motormodell har vi valgt å dele den inn i to hoveddeler, der den ene beskriver motorenens dynamikk, mens den andre angir thrust-kraft, moment og turtall som funksjonen av pådraget inn på ESC-en. Dette er en løsning som også er anvendt i andre lignede prosjekter [60], til forskjell fra oss, modelleres funksjonene her med utgangspunkt motorenes turtall.



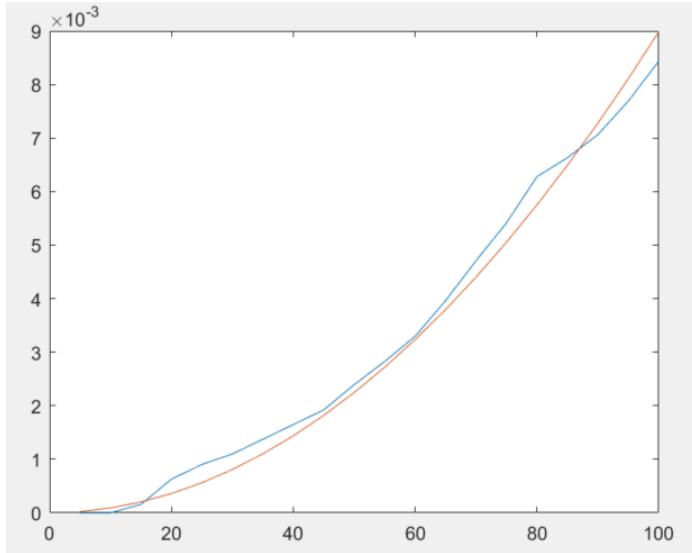
Figur 68: Oppdeling av motormodell

### 6.12.1 Moment- og thrust-kraft- funksjon

I de testene som er gjort er det plotet en funksjon, hvor output er thrust-kraften/momentet motoren gir som funksjon av input i form at pådrag fra 0-100%. I figurene under er målingene vist sammen med den tilnærmingen som er gjort. Det vises her at disse målingene har god overensstemmelse med en eksponensialfunksjonene i ligning 6.46 og 6.47



Figur 69: Thrust-kraft i newton som funksjon av pådrag



Figur 70: Moment i newtonmeter som funksjon av pådrag

De tilnærmede funksjonene er:

$$F = Thrust - kraft = 0,0001262 \cdot (\text{Pådrag})^2 \quad (6.46)$$

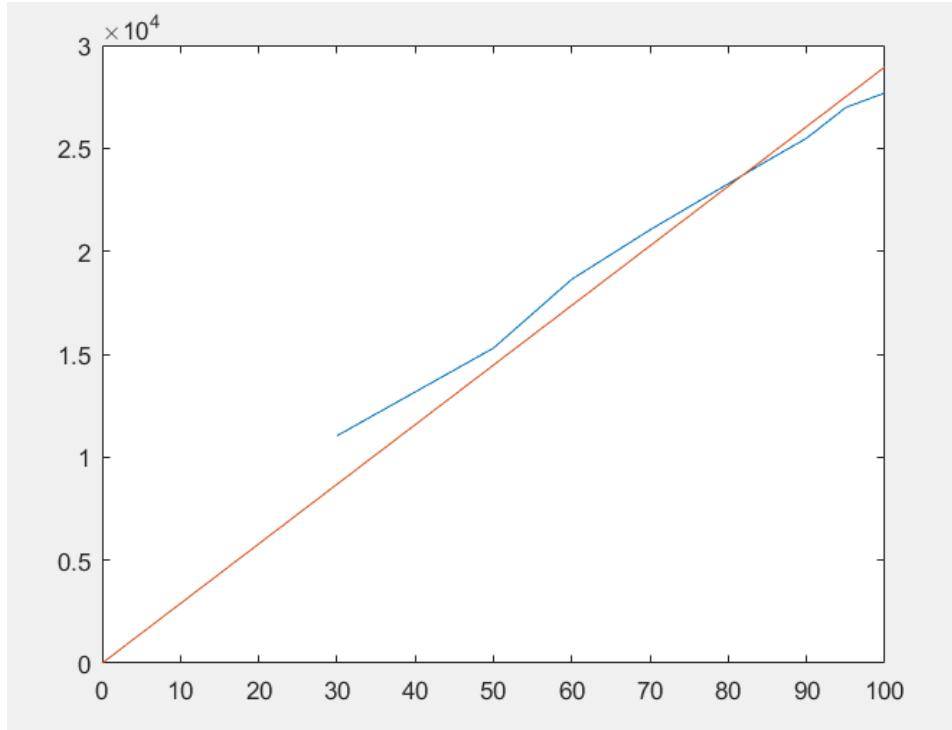
$$\tau_n = Moment = 8,985 \cdot 10^{-7} \cdot (\text{Pådrag})^2 \quad (6.47)$$

I praksis vil pådraget kunne være mellom 0 og 100. Det er dermed også lagt inn begrensninger på dette i motormodellen, slik at vi også i en simulering vil kunne ha begrenset kraft og moment fra motorene. Denne begrensningen er også nyttig med tanke på programvaren, da vi har definert hvilke verdiområde vi jobber innenfor.

For det gyroskopiske momentbidraget er vi også avhengig av å kunne hente ut motorenes turtall fra motormodellen. I mangel på utstyr for å kunne mål dette, er det her brukt målingsdata fra referanse [41], hvor vår aktuelle motor er testet med en lignende propell. Fra disse dataene har vi tilnærmet rotasjonshastigheten til følgende funksjon:

$$\omega_n[\text{rpm}] = 289.5 \cdot \text{Pådrag} \quad (6.48)$$

Denne er vist sammen med dataene i figuren under:



Figur 71: Motorens rotasjonshastighet (rpm) som funksjon av pådrag

### 6.12.2 Motordynamikk

I generell mangel på spesifikasjoner for RC-elektronikk har det vært vanskelig å kunne sette opp en modell for utregning av motorens transferfunksjon. Det har også vært mangel på utstyr for å kunne teste seg fram til dette. Transferfunksjoner tilknyttet motorer tilnærmes ofte et førsteordens system med den mekaniske tidskonstanten, hvor den elektriske tidskonstanten negligeres da den som regel er mye mindre en den mekaniske.

For vår modell er det gjort en tilnærming mot referanse [60], hvor det er brukt en førsteordens transferfunksjon med tidskonstant på 0.076 sekunder for å modellere motorenes dynamikk. Disse motorene har et treghetsmoment som er mer en 15 ganger større en vårt treghetsmoment med propell. Dette forholdet kan ikke brukes direkte på tidskonstanten da momentet for disse motorene naturligvis også er større. Ut ifra dette antar vi at en konservativ tidskonstant for våre motorer er 0.04 sekunder.

$$Transferfunksjon_{motor} = \frac{1}{\tau_{motor}s + 1} = \frac{1}{0.04s + 1} = \frac{25}{s + 25} \quad (6.49)$$

### 6.12.3 Motormodell i Simulink

Motormodellen er bygd opp med grunnlag i ligningene 6.46, 6.47, 6.48 og 6.49. Denne modellen tar inn fire motorkommandoer og leverer ut thrustkraft, moment og turtallet for hver motor. Modellen er lagd slik at de enkelte funksjonene står hver for seg, slik at de enkelt kan endres

uavhengig av hverandre. Det er som nevnt også lagt inn pådragsbegrensninger i modellen for å representer de begrensningene motorene har i praksis. Det er her viktig å være klar over at begrensninger i pådrag fører til ulinearitet i systemet når vi kommer over grensene for metning. Implementasjonen i Simulink kan ses i appendiks.

#### 6.12.4 Linearisering av Motormodell

I forbindelse med linearisering av systemer er det vanlig å linearisere rundt et arbeidspunkt. For dronen vår vil det typisk være ved hovring. Dette gir en god tilnærming nært arbeidspunktet, med økende avvik ved større avstand. I denne delen vil vi bruke en alternativ metode som opphever eksponensialfunksjonene og gjør thrust-kraften og momentet lineært avhengig av pådraget.

Fra vår motormodell vet vi at kraft og moment er eksponensialfunksjoner av pådraget, samt at motorenes turtall er tilnærmet lineært med pådraget. Det vil si at vi kan gjøre kraft og moment lineært avhengig av pådraget ved å introdusere en pådragsfunksjon som opphver/kansellerer eksponensialfunksjonene. Det nevnes her at ved bruk av denne løsningen vil ikke lengre motorenes turtall være lineært avhengig av pådraget, men samtidig har det gyroskopiske momentbidraget lite innvirkning sammenlignet med de resterende kreftene og momentene som virker på dronen (framkommer av testing og lav roll- og pitch-rate). Dette er noe vi dermed ikke tar hensyn til.

$$\text{pådrag}_{ut} = \sqrt{100 \cdot \text{pådrag}_{inn}} \quad (6.50)$$

I ligning 6.50 er  $\text{pådrag}_{ut}$  pådraget ut av funksjonen som går videre til motormodellen. Introdusere vi denne funksjonen før motormodellen får vi følgende:

$$F = \text{Thrustkraft} = 0,01262 \cdot \text{Pådrag}_{inn} \quad (6.51)$$

$$\tau_n = \text{Moment} = 8,985 \cdot 10^{-5} \cdot \text{Pådrag}_{inn} \quad (6.52)$$

$$\omega_n[\text{rpm}] = 289,5 \cdot \sqrt{100 \cdot \text{Pådrag}_{inn}} \quad (6.53)$$

I Simulink-implementasjonen av denne lineariserende funksjonen er den plassert i en egen blokk kalt "Motor function compensator". Oppbyggingen er vist i appendiks. Det bemerkes at denne blokken er en del av dronens regulering/stabilisering, og er brukt i Matlab-noden i ROS.

Når denne blokken er i bruk vil motormodellens transferfunksjoner (kombinert med denne blokken) bli følgende:

$$TF_{\text{Thrustkraft}} = 0,01262 \cdot \frac{25}{s + 25} \quad (6.54)$$

$$TF_{\text{moment}} = 8,985 \cdot 10^{-5} \cdot \frac{25}{s + 25} \quad (6.55)$$

## 6.13 Forenklet dronemodell

I dette kapitellet forenkles den koblede matematiske dronemodellen som framkommer av kapittel 6.11 ned til en dekoblet modell, hvor hvert separate input har en "egen" output. Hensikten med dette er å forenkle prosessen med å sette opp kontrollere for dronen regulering og stabilisering.

For dette kapitellet tar vi også utgangspunkt i at motormodellen er linearisert i henhold til 6.12.4

### 6.13.1 Dekobling

For å dekoble modellen gjør vi en del antagelser og forenklinger, dette begrenser anvendelsesområdet til den forenklede modellen.

For den dekoblede modellen tas det utgangspunkt i at dronen hovere/står i  $X_nY_n$ -planet og at hver enkelt av de fire frihetsgradene som styres direkte har sitt eget input.

#### Momentligning

Fra ligning 6.43 har vi:

$$I \left[ \frac{d}{dt} \omega_b \right]_b = \vec{r}_1 \times \begin{bmatrix} 0 \\ 0 \\ -F_1 \end{bmatrix} + \vec{r}_2 \times \begin{bmatrix} 0 \\ 0 \\ -F_2 \end{bmatrix} + \vec{r}_3 \times \begin{bmatrix} 0 \\ 0 \\ -F_3 \end{bmatrix} + \vec{r}_4 \times \begin{bmatrix} 0 \\ 0 \\ -F_4 \end{bmatrix} + \tau_b + M_{gyro} - \omega_b \times (I\omega_b) \quad (6.56)$$

Dronen skal brukes for flyvning i lav hastighet og dermed små vinkelutslag, det antas dermed her at vinkelhastighetene er så små at siste ledd i ligningen kan neglisjeres. I tillegg ser vi bort fra det gyroskopiske momentbidraget.

Løser vi nå ut kryssproduktene og separerer ligningene for moment om  $x_b$ ,  $y_b$  og  $z_b$  får vi:

$$\dot{p} = \frac{-r_{y1}F_1 - r_{y2}F_2 - r_{y3}F_3 - r_{y4}F_4}{I_{xx}} \quad (6.57)$$

$$\dot{q} = \frac{r_{x1}F_1 + r_{x2}F_2 + r_{x3}F_3 + r_{x4}F_4}{I_{yy}} \quad (6.58)$$

$$\dot{r} = \tau_{b_z} \quad (6.59)$$

Fra oppbygningen av signalmixingen (kapittel 6.15) som setter sammen motorpådragene fra de ulike kontrollerne, vet vi at pådraget går like mye ned på to motorer som det går opp på de andre to når vi prøver å rotere dronen. Det vil si at kreftene og momentene er like store i utslag begge veier. Det presiseres her at det ikke lengre er totale krefter vi ser på, men separate for hver enkelt input. Samtidig vet vi at dronen er symmetrisk om massesenteret slik at alle posisjonsvektorene ut til motorene har samme komponenter bare ulikt fortegn. Vi får følgende ligninger

$$\dot{p} = \frac{4 \cdot r_y \cdot F_\phi}{I_{xx}} \quad (6.60)$$

$$\dot{q} = \frac{4 \cdot r_x \cdot F_\theta}{I_{yy}} \quad (6.61)$$

$$\dot{r} = \frac{4 \cdot \tau}{I_{yy}} \quad (6.62)$$

## Kraftligning

Fra ligning 6.36:

$$m \left[ \frac{d}{dt} V_b \right]_b = \begin{bmatrix} 0 \\ 0 \\ -F_1 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -F_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -F_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -F_4 \end{bmatrix} + R_n^b \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} - m(\omega_b \times V_b) \quad (6.63)$$

Med samme antagelser som for momentligningen, lav hastighet og små vinkelutslag, antas det at siste ledd i ligningen over er neglisjerbart.

For det dekoblede systemet tar vi utgangspunkt i ren hovring for krefter i  $z_b$ -retning slik at vi får:

$$a_{z,n} = a_{z,b} = \frac{4F_z}{m} + g \quad (6.64)$$

Videre vil akselrasjonen i y-retning være avhengig av dronens orientering:

$$m \cdot a_{y,b} = R_n^b \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} = \sin(\phi)\cos(\theta)mg \quad (6.65)$$

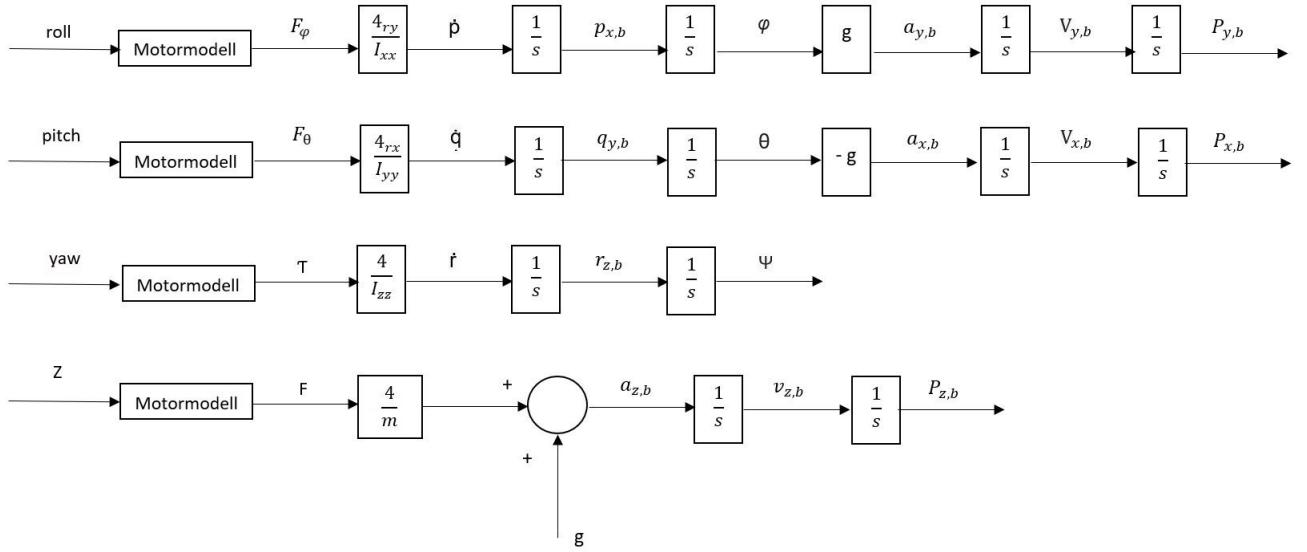
Om vi nå antar at vi skal kjøre i kun y-retning, vil akselrasjonen være avhengig av dronens roll-vinkel. Samtidig bruker vi antagelse om små vinkelutslag og setter  $\sin(\phi) \approx \phi$  (gjør mindre en ti prosent feil med 12 grader vinkel). Dette gir

$$a_{y,b} = \phi g \quad (6.66)$$

På samme måte for akselrasjon i x-retning får vi:

$$a_{x,b} = -\theta g \quad (6.67)$$

### 6.13.2 Grafisk forenklet modell



Figur 72: Dekoblet modell

Tabell 8: Verdier grafisk dekoblet modell

	$I_{xx}$	$I_{yy}$	$I_{zz}$	$r_x$	$r_y$	$r_z$	$g$	$m$
Verdi	0.00021233 kgm <sup>2</sup>	0.00021409 kgm <sup>2</sup>	0.00036986 kgm <sup>2</sup>	0.045m	0.06m	0.015	9.81m/s <sup>2</sup>	0.209kg

Fra modellen over ser vi at vinkelakselasjonen om kroppsrammens akser er integrert opp og brukt som vinklene  $\phi$ ,  $\theta$  og  $\psi$  videre i beregningene av akselasjon. Dette er ikke i henhold til hvordan Euler-vinklene er definert, men dette vil fungere greit for å se på systemets oppførsel ved et input av gangen. Startbetingelsene vil her være at dronen står horisontalt slik at  $\phi$  og  $\theta$  representerer vinkelene mellom dronekroppen og NED-rammens horisontale plan. Mens  $\psi$  angir dronens retning (heading) i forhold til NED-rammen.

Dette oppsette gir muligheten til å transformere dronens flyveretning fra kroppsrammen til NED-rammen ved bruk yaw-vinkelen. Denne transformasjonen er vist ligningene under.

$$X_n = X_b \cos(\psi) - Y_b \sin(\psi) \quad (6.68)$$

$$Y_n = X_b \sin(\psi) + Y_b \cos(\psi) \quad (6.69)$$

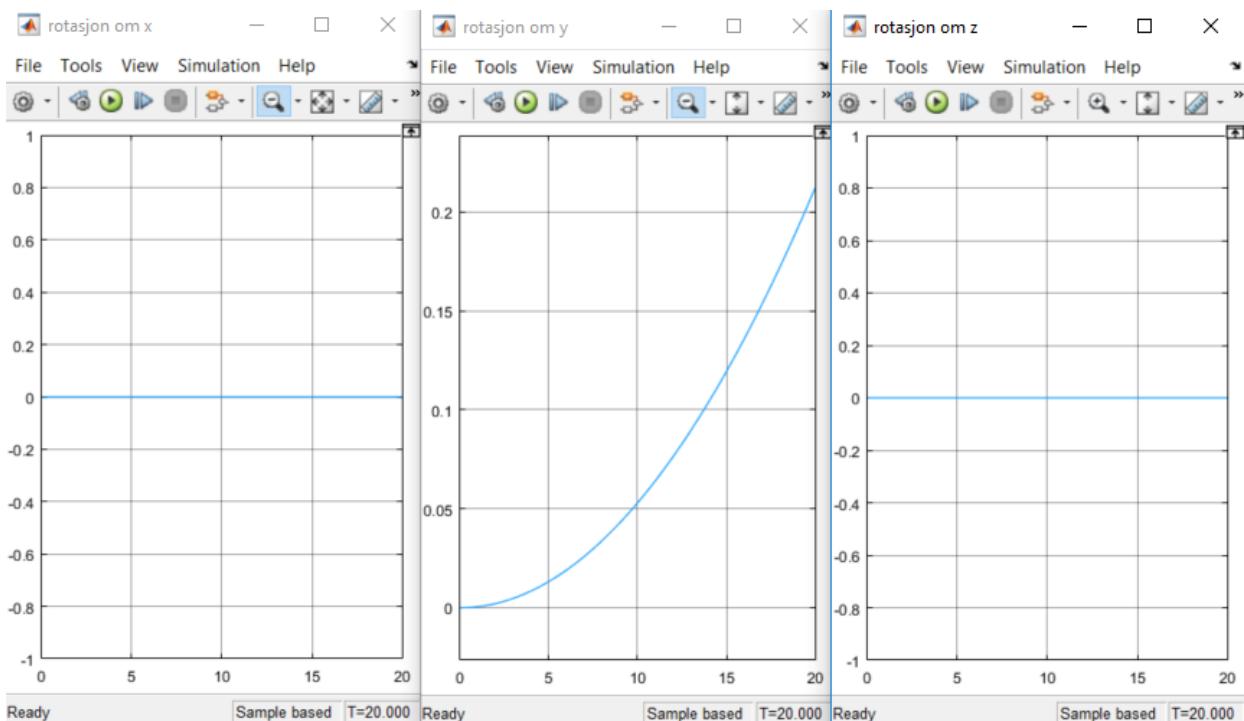
Dette gjør at man kan se på hver enkelt delsystem som separate som i figur 72. Hvor den endelige outputen av systemet er en kombinasjon av alle outputene fra hvert delsystem. Den dekoblede modellen/systemet består av SISO-systemer og vi kan dermed designe kontrollere for et delsystem av gangen.

## 6.14 Sammenligning av modeller

### Sammenligning av koblet og dekoblet modell

For å verifisere at den dekoblede modellen er en tilnærmet representasjon av den koblede modellen er det her lagt fram en sammenligning, hvor ulike input er sendt inn i de to ulike modellene. I henhold til 6.13 vil det være mest aktuelt å se på vinkeldreininger under 0.2 rad (gjelder ikke yaw) for å holde feilen knyttet til posisjon, hastighet og akselrasjon innenfor moderate grenser.

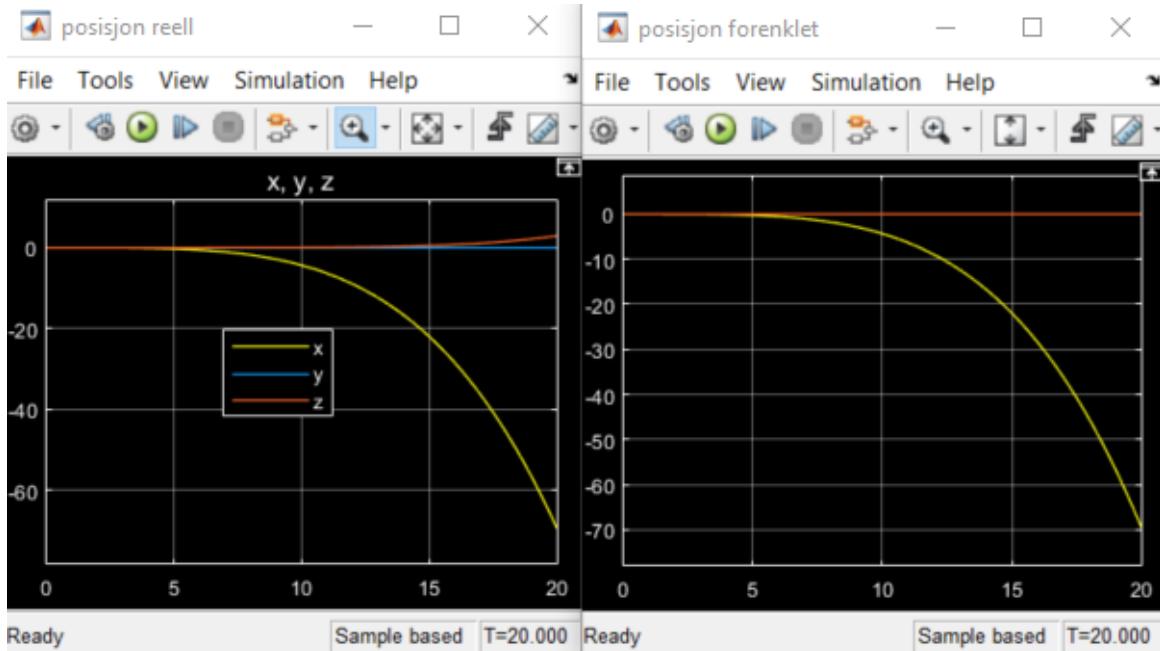
#### Enkelt-input



Figur 73: Vinkelposisjons-respons for konstant input på 0.0001 i pitch

Overstående figur viser output av modellene i form av vinkelposisjon i samme graf. I denne testen hovrer dronen, samt får den et konstant pitch input. Det er kun en linje som vises da modellenes output er identisk. Det samme resultatet framkommer om man tar utgangspunkt i hovring, og kjører separate tester for roll og yaw. Altså vil man ved hovring med rotasjon om kun en akse oppnå samme output av begge modeller. Dette er i henhold til de forventninger diskutert i siste del av 6.13

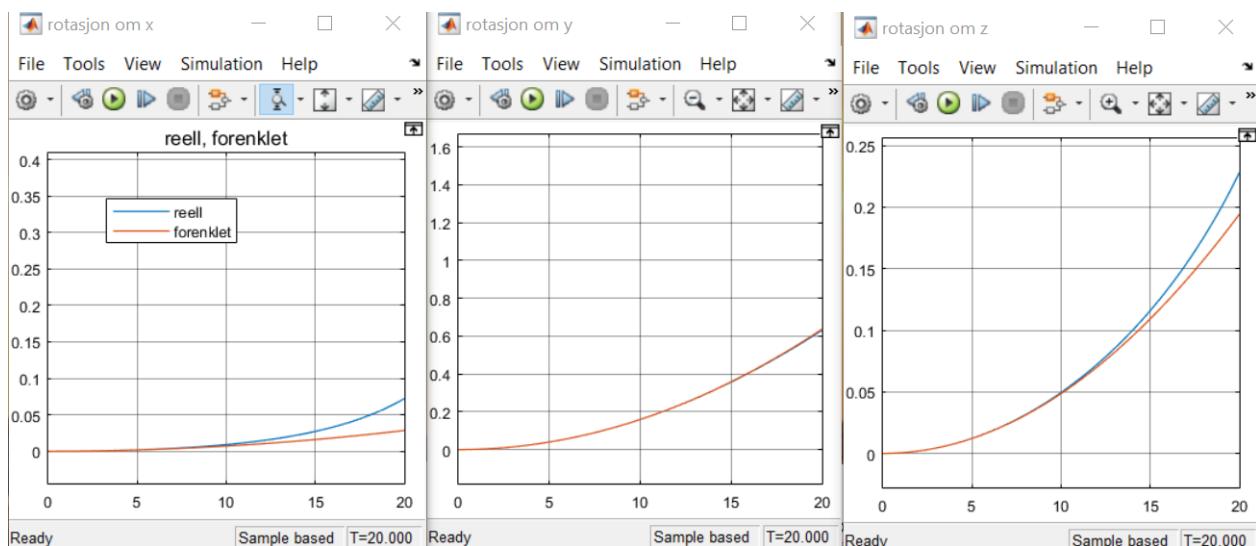
I figur 74 er modellresponsene i form av posisjon lagt fram for samme input som nevnt i avsnittet over. Det ses her at det er god overensstemmelse, noe som hører sammen med at vi holder oss innenfor 0.2 radianer i pitch. Den lille endringen i z posisjon skyldes at i den koblede modellen vil kraftvektoren i negativ Z-retning bli noe mindre på grunn av dronens helning, og dronen vil dermed miste høyde (simulering uten regulering av  $z_n$ ).



Figur 74: Posisjonsrespons for konstant input på 0.0001 i pitch

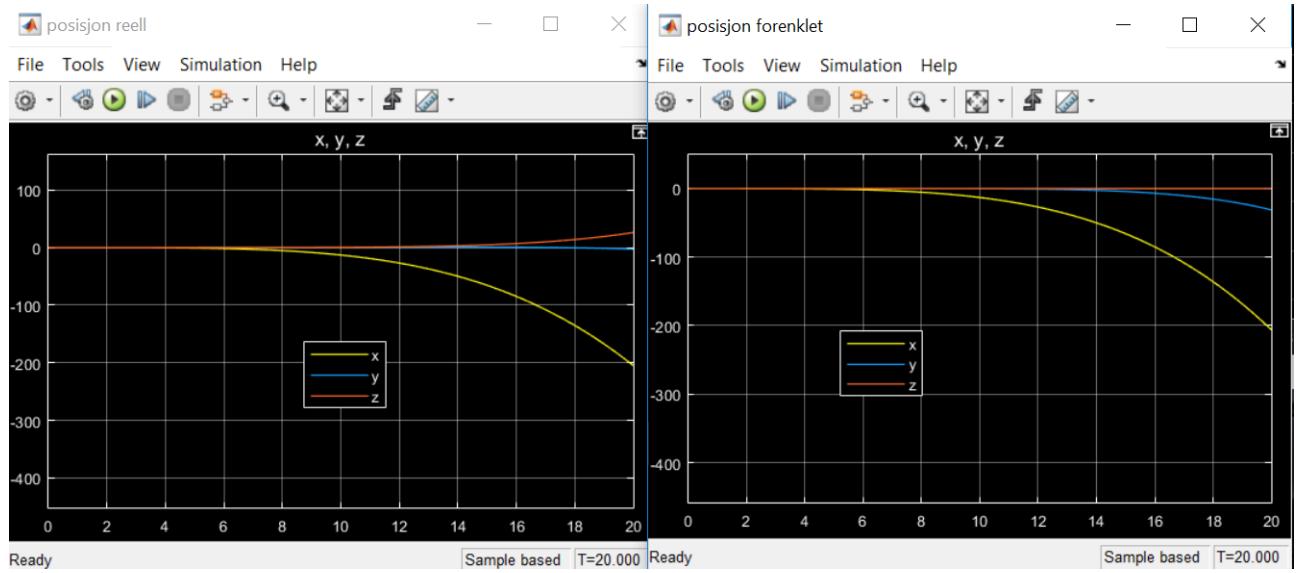
## Multi-input

I figur 75 er modellene utsatt for et konstant input på 0.00001 i roll, 0.001 i yaw og 0.0003 i pitch (med utgangspunkt at dronen hovrer). Her kan man se forskjell mellom output fra den dekoblede- og den koblede- modellen. Dette skyldes de forenklinger som er gjort mellom de to modellene. I tillegg til at den koblede modellen er basert på eulervinkler, mens i den dekoblede modellen er pitch og roll, vinkler mellom det horisontale plan gitt av  $X_{nv}Y_{nv}$  og det kroppsfaste koordinatsystemet.



Figur 75: Vinkelposisjons-respons for konstant input på 0.00001 i roll, 0.0003 i pitch og 0.001 i yaw

Om figur 76 tas i betrakting, kommer det fram at responsen fra den dekoblede modellen er en representativ tilnærming til den koblede modellen, også med tanke på posisjon. Det må imidlertid påregnes noe avvik på samme måte som for vinkelposisjon.



Figur 76: Posisjonsrespons for konstant input på 0.00001 i roll, 0.0003 i pitch og 0.001 i yaw

Overstående viser at den forenklede modellen er representativ for den koblede, for ulike input. Ved bruk av den forenklede modellen vil modellering av systemkontrollere forenkles kraftig, men det må understrekkes at for ekstreme kombinasjoner av ulike input vil vi kunne få store avvik mellom modellene. Dette er grunnlaget for at kontrollere utviklet med den forenklede modellen, bør testes i den koblede for å se den virkelige oppførselen.

## 6.15 Regulering og stabilisering

For vår drone som er et quadcopter er det hovedsaklig 4 frihetsgrader man kan styredirekte. Det er rotasjon om alle 3 aksene til dronen, samt dens flyhøyde  $z_n$ . De to siste frihetsgradene som er bevegelse i x- og y- retning, er indirekte styrt av dronens rotasjonsfrihetsgrader.

For å holde dronen stabil og samtidig kunne styre disse frihetsgradene må det implementeres kontrollere. I dette kapitellet er det kort vist hvordan vi har gått fram for å lage kontrollere. Teknikkene som er brukt er beskrevet i referanse [61] kapittel 11. Det er gjennom kapittelet tatt med de aktuelle bode plotene som viser virkningen av kontrollerne.

Det er hovedsaklig brukt teknikker for å lage lag-, lead- og laglead-kompensatorer. Resultatet av dette er videre regnet om til PID-kontrollere, som er implementert i reguleringen. Dette på grunn av at PID-kontrollere har svært enkel oppbygning med få separate parametre, og er dermed mer egnet for testing og finjustering.

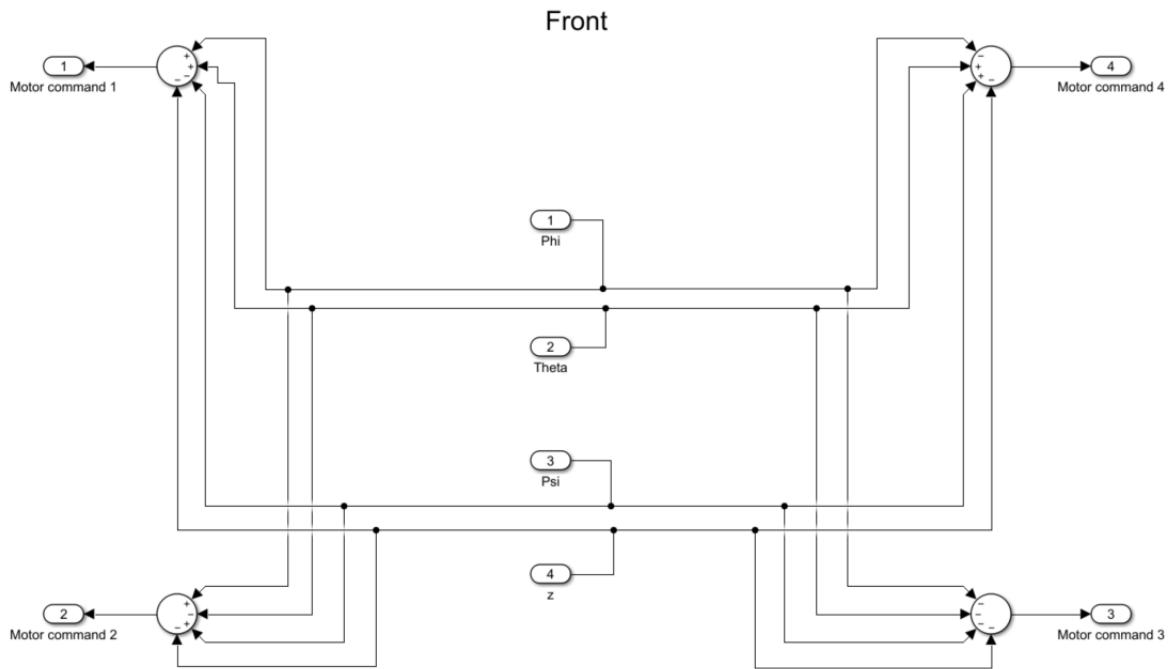
Først og fremst er det vinkelposisjonen vi ønsker å kontrollere for å sikre en stabil flyvning. Fra andre reguleringssmodeller er det brukt ulike løsninger for å gjøre dette. Ser vi på et tilvarende prosjekt i referanse [60] er det brukt en enkel tilbakekobling med en PID-kontroller for å styre vinkelposisjonen. En alternativ løsning er ArduPilots oppsett med to tilbakekoblinger, en løkke for vinkelhastighetsstyring ved hjelp av en PID-kontroller og en ytre løkke for vinkelposisjonsstyring med en enkel P-kontroller i tillegg til en foroverkobling [62].

For vår regulering har vi brukt to tilbakekoblingsløkker, en for vinkelhastighetsstyring og en for vinkelposisjonsstyring med hver sin PID-kontroller. Hensikten med dette er å gi mulighet til å kunne fly ved bruk av kun vinkelhastigheter til å begynne med. Dette for å kunne finjustere disse parametrerne ved flyvning med fysisk modell, før man tar for seg vinkelposisjonssløyfene. Bakdelen med dette er at det blir et behov for å lage flere kontrollere for å styre vinkelposisjonen.

Kontrollerne som er designet er tenkt som et utgangspunkt å justere rundt ved oppstart av flyvning/testing. De krav som er satt for kontrollerene ved design, er hovedsaklig brukt for å holde kontroll på de enkelte løkkenes oppførsel. I tillegg til å ha en forventet respons å sammenligne med. Generelt er lite oversving med påfølgende høy fasemargin ønskelig for å motvirke blant annet potensielle til oscillasjoner.

### 6.15.1 Signalmiksing

For å sette sammen kontrollpådragene fra de ulike kontrollerne, til pådrag ut til hver enkelt motor, er det satt opp en signalmiksing. For den kobledes modellen er dette kun fire summeringer av de ulike kontrollpådragene. Modellen fra Simulink er vist under, hvor inputene  $\phi, \theta, \psi$  og  $z$  representerer de forskjellige kontrollpådragene:



Figur 77: Signalmiksing for dronens regulering

### 6.15.2 Kontroller design

De kontrollerne som er lagt frem i denne delen er designet ved hjelp av den forenklede modellen for så å bli testet i den koblede modellen etterpå. Det tas utgangspunkt i at linearisering fra 6.12.4 er i bruk.

Framgangsmetoden er lik for kontrollerdesignet for de ulike frihetsgradene. Det er derfor kun vist framgangen for en av dem her (for roll). Her brukes åpen-løkke bode plot for å se på den lukkede løkkens egenskaper i henhold til [61].

### 6.15.3 Simulator

Med utgangspunkt i dynamikken fra den koblede modellen lagt fram i 6.11, motormodellen i 6.12, samt signalmiksingen og kontrollstrategien i dette kapitellet er det bygget en simulator for flyvning av dronen. Her er det brukt animasjonsblokker fra Simulink biblioteket for å visualisere flyvningen. Denne modellen ble mye brukt under utvikling av den matematiske modellen, samt ved sammensetning av system, for å sammenligne oppførsel med de valgt input. Modellen kan ses i appendiks.

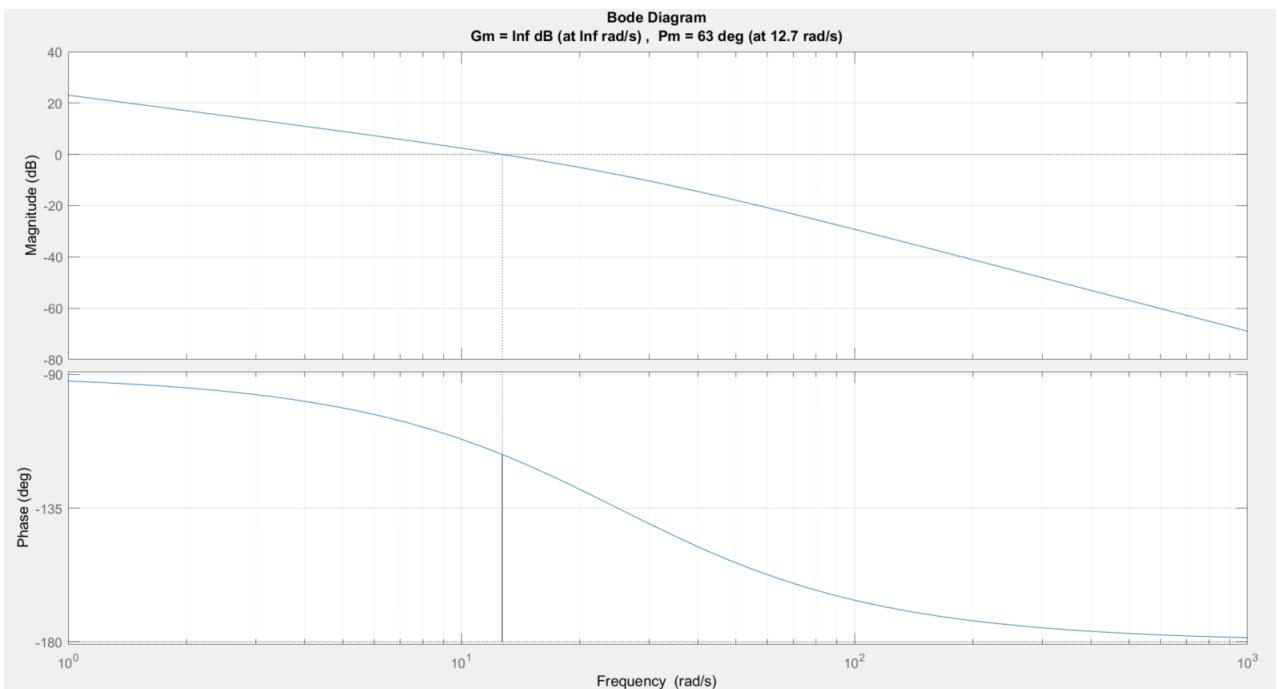
Det er også bygd opp en simuleringmodell for å sammenligne det koblede og det dekoblede modellen tilknyttet 6.14. Dette tilknyttet sammenligning og verifisering arbeid. Denne kan ses i appendiks

## Hastighetssløyfe

Med utgangspunkt i figur 72 og transferfunksjonen i ligning 6.54, vil åpen løkke transferfunksjonen mellom roll-pådrag og vinkelhastighet  $p$  bli:

$$TF_p = \frac{p}{roll_{\text{pådrag}}} = \frac{14.264}{s} \frac{25}{s + 25} \quad (6.70)$$

Med følgende bode plot:



Figur 78: Bode plot av åpen løkke transferfunksjon mellom roll-pådrag og vinkelhastigheten  $p$

Fra transferfunksjonen ser vi at vi har et type 1 system, det vil si at vi vil ha 0 i stasjonæravik for en steg input og et konstant avvik ved en rampe input.

Med den naturlige forsterkningen som transferfunksjonen har vil vi ha et stasjonæraviket på 0.7% for en ramp input med stigning på 0.1. Ut ifra dette velger vi å ikke øke forsterkningen ytterligere med tanke på stasjonæravik. Imidlertid er det ønskelig å øke hastigheten til systemet, samtidig som vi ivaretar en god stabilitet. For dette kan vi lage en lead-kompensator/PD-kontroller med hensyn på å innfri disse ønskene.

For posisjonssløyfene for roll og pitch ønsker vi en topptid på 0.2 sekunder. Da den indre hastighetsløkken må være raskere, setter vi krav om topptid til 0.07 sekunder, med en oversving på mindre enn 2%.

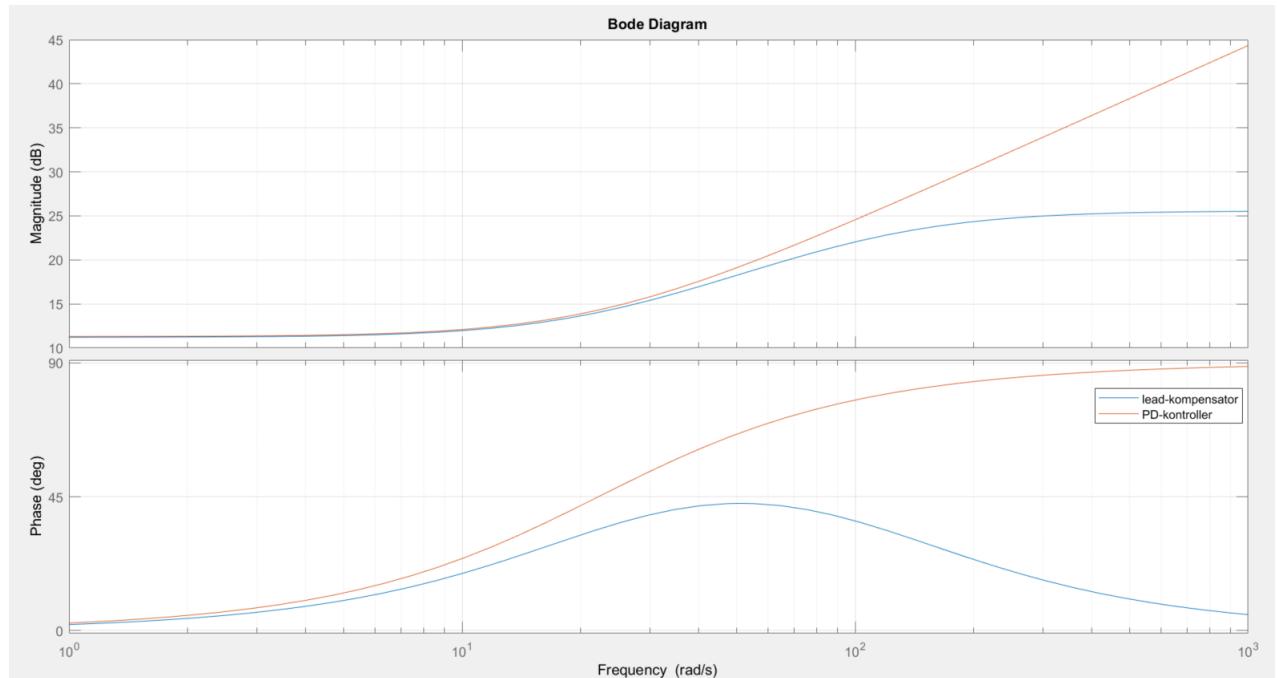
Dette innebærer følgende krav: Fasemargin må være større eller lik  $68^\circ$  og båndbredden må være større enn 65 rad/s.

For designet av kontrollerene antas det at krysningen av 0db linjen i bode plotet er 0.8 ganger kryssfrekvensen (notasjon  $\omega_c$ ). Dette gir følgende kontrollere:

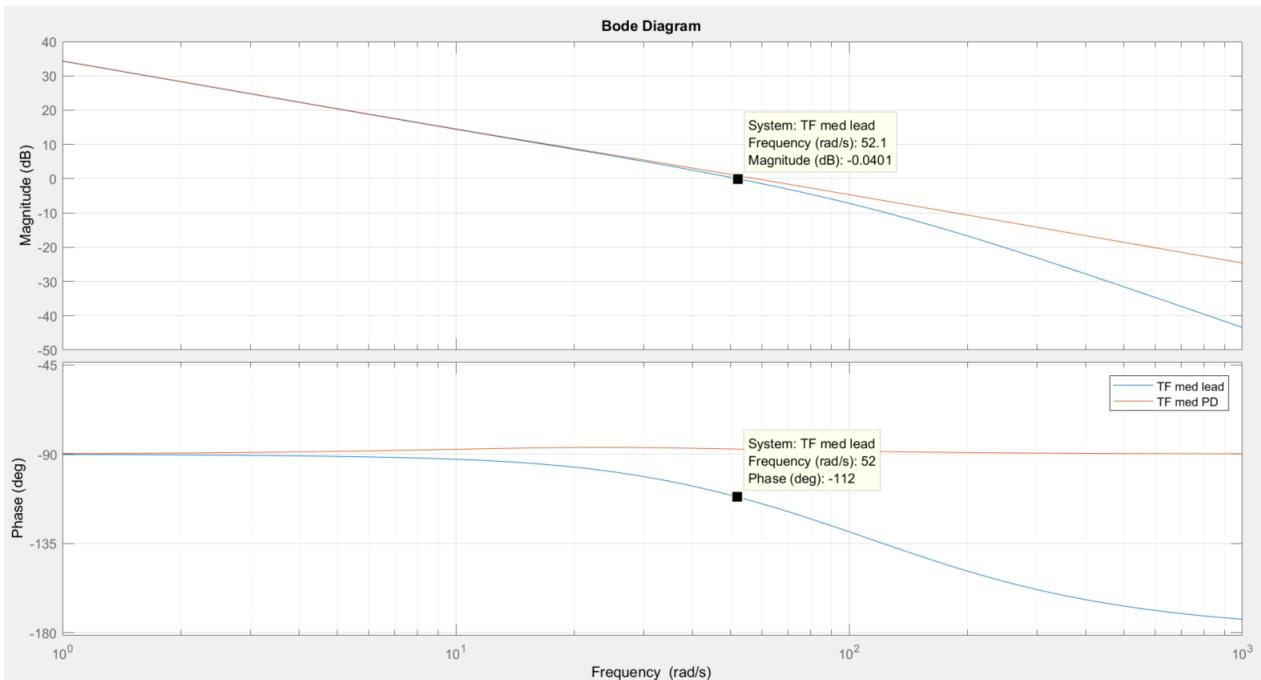
$$G_{lead} = \frac{19s + 425}{s + 117} \quad (6.71)$$

$$G_{PD} = s0.165 + 3.67 \quad (6.72)$$

Disse kontrollerne er vist separat i figur 79, og sammen med system i figur 80

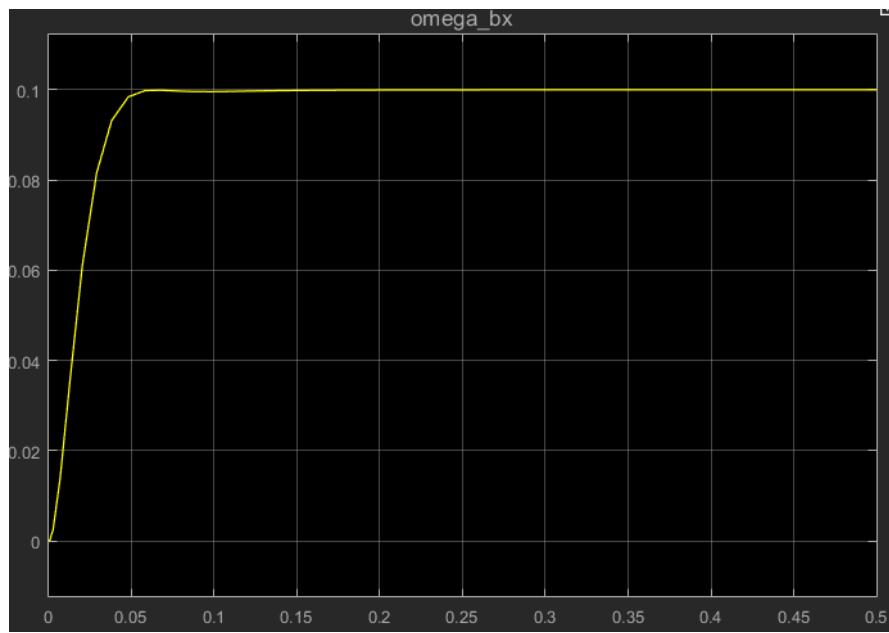


Figur 79: Bode plot av lead-kompensator og PID-kontroller for vinkelhastighet p



Figur 80: Bode plot av åpen løkke transferfunksjon med kontrollere

Fra bode plotet over ses det at kravet vi har satt til fasemargin er overholdt. For et åpen løkke bode plot leses den lukkede løkkens båndbredde ut på -7db, noe som vil si at vi er langt over minstekravet her. Step-responsen til hastighetssløyfen er vist i figuren under, og bekrefter at responsen er som ønskelig.

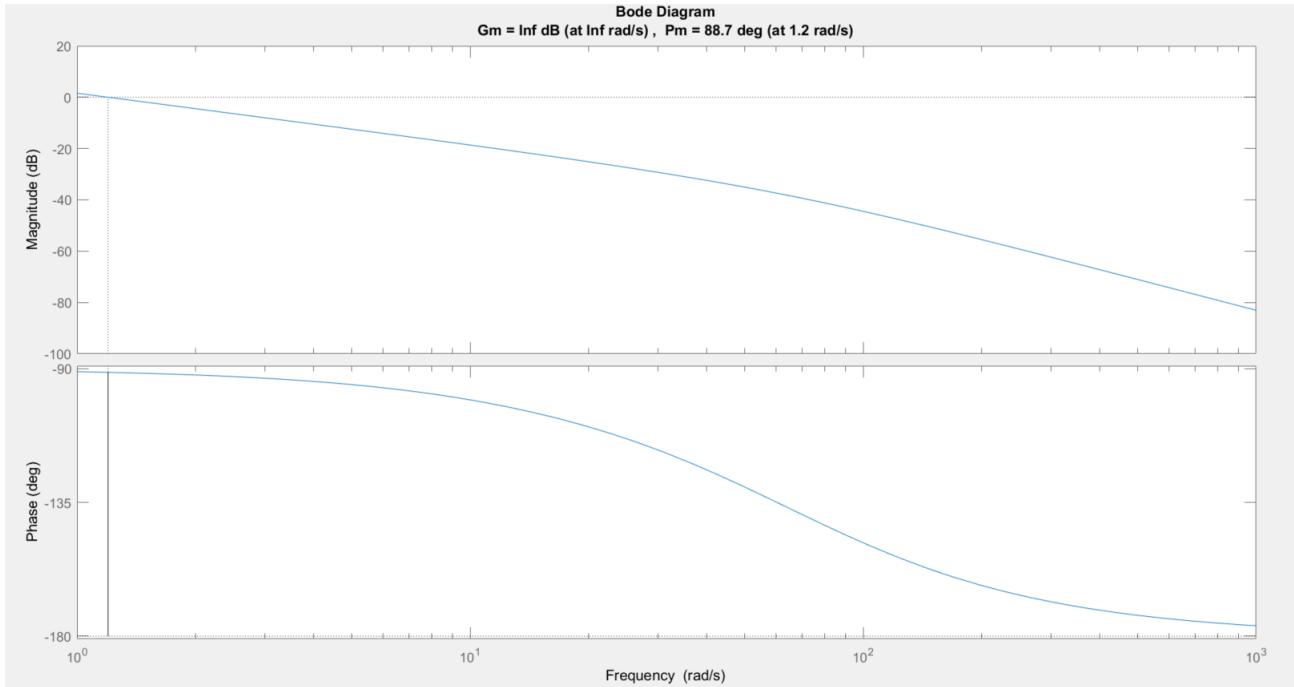


Figur 81: Vinkelhastighet step-respons med implementert kontroller

## Vinkelposisjonssløyfe

Om vi implementerer PD-kontrolleren i ligning 6.72 og lukker hastighetssløyfen ved hjelp av blokkreduksjon får vi følgende åpen løkke transferfunksjon for vinkelposisjonen:

$$TF_\phi = \frac{\phi}{roll_{pd़ag}} = \frac{58.8s + 1309}{s(s^2 + 83.8s + 1309)} \quad (6.73)$$



Figur 82: Bode plot av åpen løkke transferfunksjon mellom roll-pådrag og vinkelposisjon  $\phi$

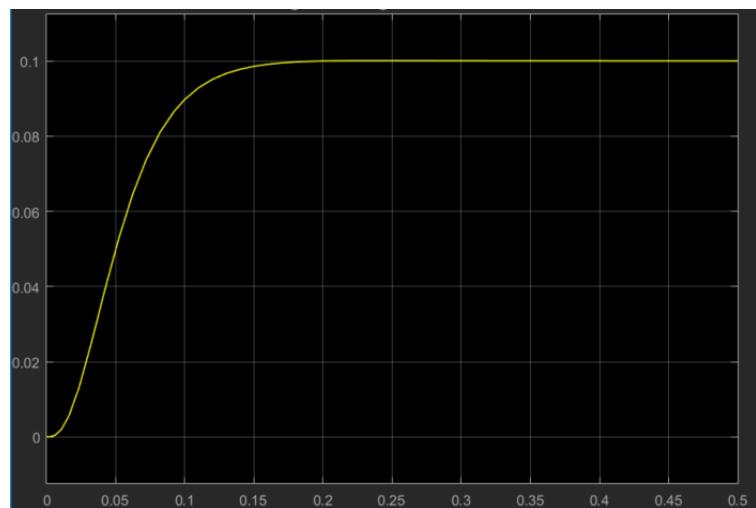
Som nevnt ønsket vi en topptid på 0.2 sekunder for vinkelposisjons-step-responsen. Dette tilsvarer en båndbredde på 22rad/s i lukket løkke. Dette tilsvarer en kryssfrekvens på 17.6 rad/s. Dette er også et type 1 system, slik at vi vil ha et stasjonæravvik for et rampe input. Vi setter krav til mindre en 5% avvik ved rampe med stigning 0.1. Dette tilsvarer her at vi må ha en kryssfrekvens på 20rad/s, og dette er dermed styrende.

På grunn av bode plotets utforming er det mulig tilfredsstille krav til avvik, samtidig som man forstatt har en god fasemargin, ved bruk av en ren P-kontroller. Alternativt kan det brukes en PI- eller PID-regulator.

Bruker en P-kontroller med forsteking 17.8

$$G_P = 17.8 \quad (6.74)$$

Step-responsen til kaskadekoblingen bestående av vinkelhastighetsløkke og vinkelposisjonløkken er vist i figuren under, for en step på 0.1 radian. Her ser vi at det endelige resultatet innfør det ønsker vi hadde for step-responsen.



Figur 83: Vinkelposisjons step-respons med implementerte kontrollere

Etter å ha designet ferdig kontrollerne er de sammen testet i simulatoren. Hvor de har vist seg å fungere også i det koblede systemet ved kombinasjon av ulike input.

Dette må imidlertid tas hensyn til at stabilitetsmarginene vil kunne være mindre i den koblede modellen og i praksis, sammenlignet med den dekoblede modellen.

## 6.16 IMU

### 6.16.1 ROS node

Det ble valgt en en MPU6050 fra Invensense. Det finnes ROS noder laget for denne IMU'en på github, og noden som ble utviklet av gruppen er basert på matpalm sin "ros-mpu6050-node" [43]. Noden er originalt satt opp til å kjøre på 10 Hz med FIFO register aktivert for å sikre at all data blir registrert. For vår applikasjon er det nødvendig med høyere sample rate og vi ønsker alltid å motta nyeste sett med data.

Dette kan muliggjøres ved å endre noen av registrene beskrevet i manualen fra Invensense [63]. FIFO registeret fungerer som en buffer på IMU'en og kan holde på opptil 1000 sett med data om IMU'en sampler raskere enn den sender data. En modifikasjon av "0x6B" registeret deaktiverte FIFO for å sikre at reguleringsløyfer alltid mottar nyeste data som er tilgjengelig.

ROS emnet "imu" inneholder seks float64 variabler hvor lineær akselerasjon- og vinkelhastighetvektorer blir publisert. Publiseringssraten til noden kan settes ved et parameter ROS, men en bedre tilnærming vil være å la IMU'en styre raten. Dette kan gjøres ved å overvåke INT\_STATUS registeret. Med riktig konfigurasjon har registeret verdien "1" hver gang ny data er klar, og kan dermed brukes til å time publisering av data på ROS emnet. En "sample rate divider" bestemmer hvor ofte MPU6050 skal sample data. Ligning for beregning av verdi på denne er gitt av manual, og under vises utregning for 100Hz sample rate.

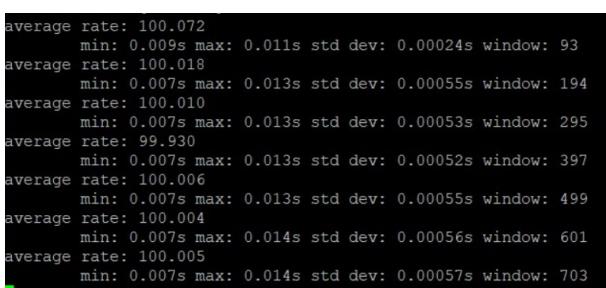
$$\text{Sample Rate} = \frac{\text{Gyroscope output rate}}{1 + \text{SMPLRT\_DIV}} \quad (6.75)$$

$$\text{SMPLRT\_DIV} = \frac{\text{Gyroscope output rate}}{\text{Sample Rate}} - 1 = \frac{8 \cdot 10^3 \text{s}^{-1}}{100 \text{s}^{-1}} - 1 = 79 = 0x4F \quad (6.76)$$

"Gyroscope output rate" er oppgitt til 8kHz i datablad. Utdrag fra kode samt test av publiseringssrate er vist i figur 84a. Det sees at Ros noden publiserer på ønskede 100Hz. Ved å lytte på "imu" emnet kunne det også sees at alle data sett var unike, noe som tyder på at IMU'en ikke publiserer før den har mottat ny data. Høppigheten en ROS node publiserer meldinger kan sjekkes med kommandoen "rostopic hz". Resultatet er vist i figur 84b. Fullstendig kode kan sees i appendiks B.3

```

48 // Publish in loop.
49 while(ros::ok()) {
50     sensor_msgs::Imu msg;
51     const int interruptStatus = wiringPiI2CReadReg16 (fd, INT_STATUS) ;
52     const bool dataIsReady = interruptStatus > 0; //& (1U << DATA_RDY_INT);
53
54     if( !dataIsReady )
55     {
56         continue;
57     }
58 }
```



The terminal window shows the following output:

```

average rate: 100.072
min: 0.009s max: 0.011s std dev: 0.00024s window: 93
average rate: 100.018
min: 0.007s max: 0.013s std dev: 0.00055s window: 194
average rate: 100.010
min: 0.007s max: 0.013s std dev: 0.00053s window: 295
average rate: 99.930
min: 0.007s max: 0.013s std dev: 0.00052s window: 397
average rate: 100.006
min: 0.007s max: 0.013s std dev: 0.00055s window: 499
average rate: 100.004
min: 0.007s max: 0.014s std dev: 0.00056s window: 601
average rate: 100.005
min: 0.007s max: 0.014s std dev: 0.00057s window: 703
```

(a) Utdrag av kode

(b) Publiseringssrate MPU6050 node

Figur 84: MPU 6050 node

### 6.16.2 Komplementærfilter

Gjennom Matlab har droneprodusenten Parrot publisert representativ "flight controller" modell, tiltenkt bruk på deres kompatible droner. Komplementærfilteret de har benyttet for å estimere vinkelposisjon, er forsøkt implementert i vår Simulink ROS node. Dette for å se om filteret kunne tilpasses vårt system med en MPU6050. Første test viste betydelig drift i vinkelposisjon rundt alle akser. Filteret som er hentet fra Parrot modellen har mange variable fordelt over flere flere matlab script. Blant variablene var en matrise med kalibreringsdata. Gruppen forsøkte å sette dronen på et underlag som var så plant som mulig for å kunne hente ut representative kalibreringsdata for IMU'en som var montert i dronen. Data som publiseres på ROS emner kan printes i matlab vennlig format i terminalen ved å bruke kommandoen "rostopic echo -p". Dette tillot oss å ta gjennomsnittsmålinger av 958 aksellerasjon- og vinkelhastighetsmålinger publisert på "\imu" emnet. Dataen som ble funnet erstattet gammel kalibreringsdata som lå inne i modell fra Parrot. Dette forbedret resultatet til filteret betraktelig og vinkelposisjonsmålingene var tilsynelatende stabile og kan sees i figur 85. Resultatet vi endte opp med var såpass lovende at det ble med videre i testingen.

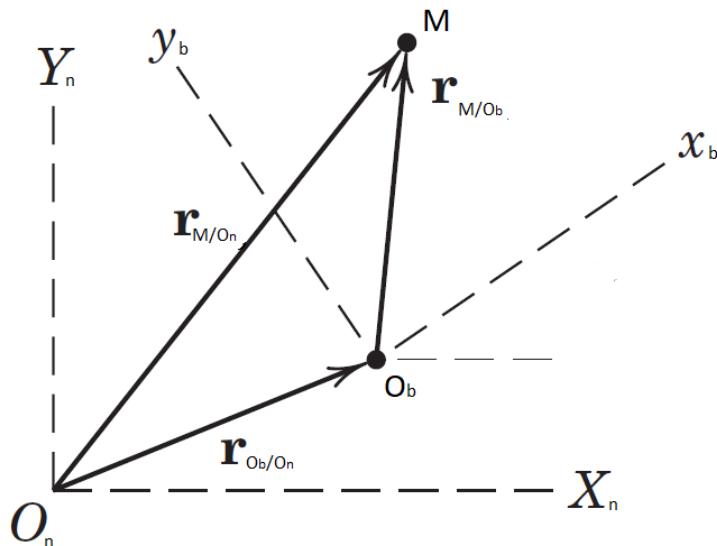
```
---
x: -0.0118338540196
y: -0.00048796919873
z: 0.000300902582239
---
x: -0.0331375040114
y: -0.00048796919873
z: 0.000300902582239
---
x: -0.0282494854182
y: -9.08330257516e-05
z: 0.00203531957231
---
x: -0.00120737729594
y: -9.08330257516e-05
z: 0.00203531957231
---
x: 0.00774271227419
y: 0.00295387743972
z: 0.000834569276776
---
x: -0.00342370476574
y: 0.00295387743972
z: 0.000834569276776
---
x: -0.00349221471697
y: -0.00128224154469
z: -0.00076643092325
---
x: -0.000785514246672
y: -0.00128224154469
z: -0.00076643092325
---
x: -0.00584948342294
y: 0.00308625632897
z: 0.000967986008618
---
x: -0.000635667704046
y: 0.00308625632897
z: 0.000967986008618
---
x: 0.00465630739927
y: -0.00194413517602
z: 3.40692022291e-05
---
x: 0.00122171221301
y: -0.00194413517602
z: 3.40692022291e-05
---
```

Figur 85: Vinkelposisjoner i radianer, estimert av komplementærfilter

### 6.16.3 Kompensering for IMU plassering og orientering

Som sett av figur 26 og 27 er IMU-en ikke plassert i dronens massesenter. I henhold til 5.9 vil IMU-en dermed oppleve en annen akselrasjon enn dronens massesenter. I tillegg, av praktiske årsaker, ligger heller ikke IMU-en slik at dens aksesystemer stemmer overens med det aksesystemet vi har definert for dronen. Ut ifra dette er det da nødvendig å kompensere for plassering og orientering, for å kunne prosessere de dataene vi får ut av IMU-en. Vi er her interessert i akselrasjonen til massepunktet og rotasjonshastighetene fra gyroskopene, med hensyn på det aksesystemet vi har definert for dronen.

I dette delkapittelet er matematikken bak kompenseringen lagt fram. Da en slik kompensering ikke ligger ferdig i Simulink, har vi selv laget en blokk som gjør denne jobben. Denne er testet mot Simulinks ferdige IMU-blokk for å verifisere at den fungere.



Figur 86: Illustrasjon av drone og IMU som punktpartikler

### Matematisk kompensering

Her er det matematiske grunnlaget for transformasjonen/omregningene, som er nødvendig for å kompenserer for plassering og orientering av IMU-en, lagt fram. Gjennom utledningen vises det til figur 86.

Posisjonsvektorene med hensyn på NED-rammen er:

$$r_{IMU} = r_{M/O_n} = r_{Ob/O_n} + r_{M/Ob} \quad (6.77)$$

Hvor  $r_{Ob/O_n}$  er vektoren fra NED-rammens origo til dronens massesenter  $O_b$  og  $r_{M/Ob}$  er vektoren fra dronens massesenter til IMU-ens origo. Bruker nå teorien fra roterende referanserammer til å finne et utrykk for akselrasjonen. Legger merke til at det er kun siste vektor i ligningen over som er i en roterende referanseramme. Akselrasjonen til IMU-en (punktet M) blir som følger:

$$\begin{aligned} \frac{d^2}{dt^2}(r_{M/On}) = a_{IMU} &= \frac{d^2}{dt^2}(r_{Ob/On}) + \frac{d^2}{dt^2}(r_{M/Ob}) + \frac{d}{dt}\omega_b \times r_{M/Ob} \\ &\quad + 2(\omega_b \times \frac{d}{dt}(r_{M/Ob})) + \omega_b \times (\omega_b \times r_{M/Ob}) \end{aligned} \quad (6.78)$$

Fra plasseringen vet vi at  $r_{M/Ob}$  er konstant slik at etter å ha snudd ligningen har vi:

$$a_{Ob} = \frac{d^2}{dt^2}(r_{Ob/On}) = a_{IMU} - \frac{d}{dt}\omega_b \times r_{M/Ob} - \omega_b \times (\omega_b \times r_{M/Ob}) \quad (6.79)$$

Dette er nå et uttrykk for akselrasjon av massemidtpunktet gitt av IMU-ens målte akselrasjon, samt vinkelhastighetene gitt av gyroskopene. For gyroskopene trengs ingen omregning, kun en transformasjon mellom aksesystemer. Dette på grunn av at vinkelhastigheten er den samme over hele legemet ved stift legemedynamikk. Det ses samtidig at akselerometerorienteringen også må transformeres til riktig aksesystem.

$$a_{IMU_b} = R_{IMU}^b a_{IMU} \quad (6.80)$$

$$\omega_b = R_{IMU}^b \omega_{IMU} \quad (6.81)$$

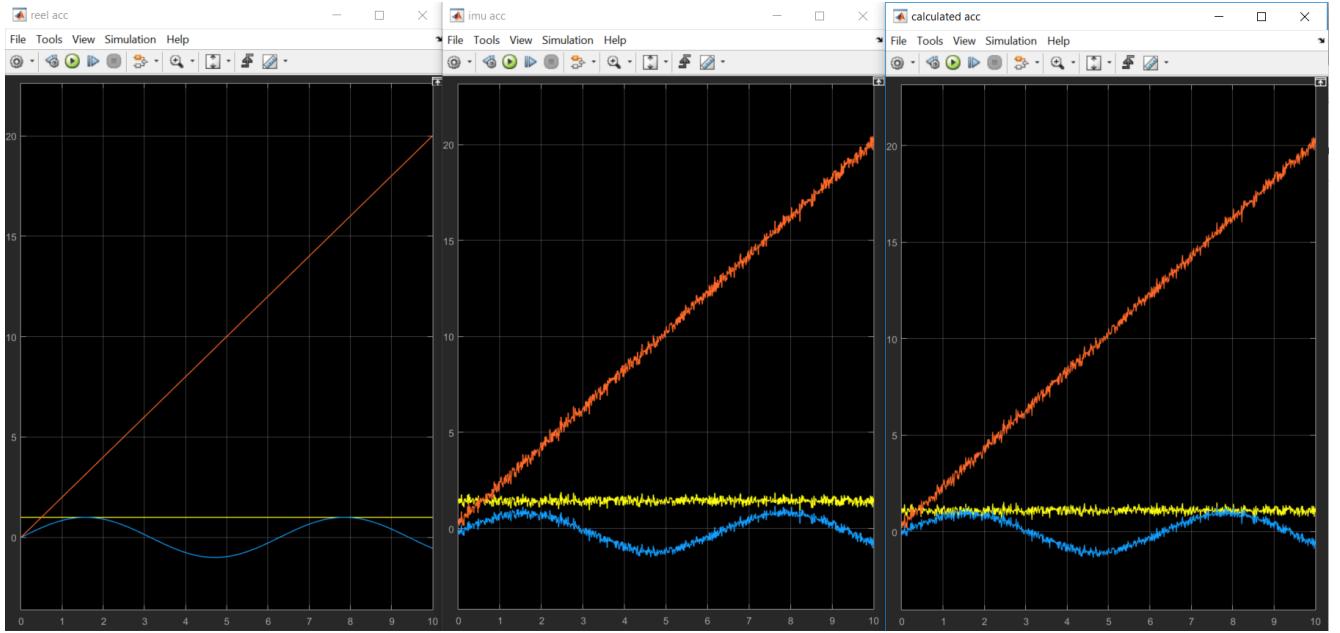
Sluttresultatet blir da ligning 6.81 og følgende ligning for akselrasjonen:

$$a_{Ob} = R_{IMU}^b a_{IMU} - \frac{d}{dt}(R_{IMU}^b \omega_b) \times r_{M/Ob} - R_{IMU}^b \omega_b \times (R_{IMU}^b \omega_b \times r_{M/Ob}) \quad (6.82)$$

Dette resultatet er sammenlignet og bekreftet med resultat funnet i referanse [64]. Det bemerkes her at det ikke er lagt inn kompensering for at måleaksene (koordinataskene) i IMU-en aldri vil være perfekt ortogonale.

### Simulink implementasjon

For å enkelt kunne bygge opp og ta i bruk denne matematikken ble modellen i appendiks laget. Denne ble videre testet mot Simulinks egen IMU-blokk for å verifisere funksjonaliteten, med tanke på avstand fra massesenter og orientering.



Figur 87: Test av Simulink implementasjon av kompensering

I figur 87 er målingene til venstre akselrasjonen til massesenteret som input til IMU-blokken i Simulink. Midtre målinger er akselrasjonen IMU-en måler og målingene til høyre er målinger etter kompensering. Det bemerkes at der lagt til støy i IMU-blokken. Utøver dette ser vi at det er overensstemmelse mellom input og kompenserte målinger, selv om akselrasjonen IMU-en måler avviker fra de to. For denne testen er avstanden mellom massesenteret og IMU-ens origo satt til de aktuelle verdiene for vår drone. Ved større avstander vil IMU-ens målte akselrasjon avvike enda mer i forhold til input og de kompenserte målingene.

Det bemerkes at kompensering/transformasjonen av IMU-ens akser til kroppsrammens akser ble testet i praksis for å verifisere at transformasjonen var riktig.

## 7 Resultater

Dronen som er laget i dette prosjektet er tilpasset bruk i Universitetet i Agder sin MotionLab. 3D-printet prototype er blitt utstyrt med 9 markører og videre testet og sporet i motion capture systemet. Sammenstilt prototype veier 209 gram uten markører og tilfredstiller med det vektkravet som ble satt til 250 gram. Dette åpner for å ettermontere gimbal med kamera. Under avsluttende testing av systemet verifiseres det med synlig regulering av pådrag, at samtlige komponenter og software som er utviklet fungerer sammen. I egenutviklet simulator, basert på dronens dynamikk er det testet og verifisert at den dekoblede modellene er representativ. Dette muliggjør SISO tuning, som i denne oppgaven er brukt til å lage kontrollere.

Det egendesignede skroget er godt egnet til vakuumstøping som er en god metode for masseproduksjon. Til tross for at vakuumstøping krever samme veggtykkelse over hele skroget, har det gjennom FEM-analyser og revidering av design, endt opp med tilstrekkelig stivhet rundt aktuelle akser. Dette har tillatt oss å neglisjere dronekroppens utbøyninger og deformasjoner i arbeid med matematisk modell.

Bakgrunnen for utformingen av skrogets bunndel, er ønsket om forenklede muligheter for lading. Det er utviklet en dockingstasjon som har en invers form av den koniske bunndelen av skroget. En løsning med sleperinger er benyttet for å kunne lande og lade dronen uavhengig av orientering i det horisontale plan.

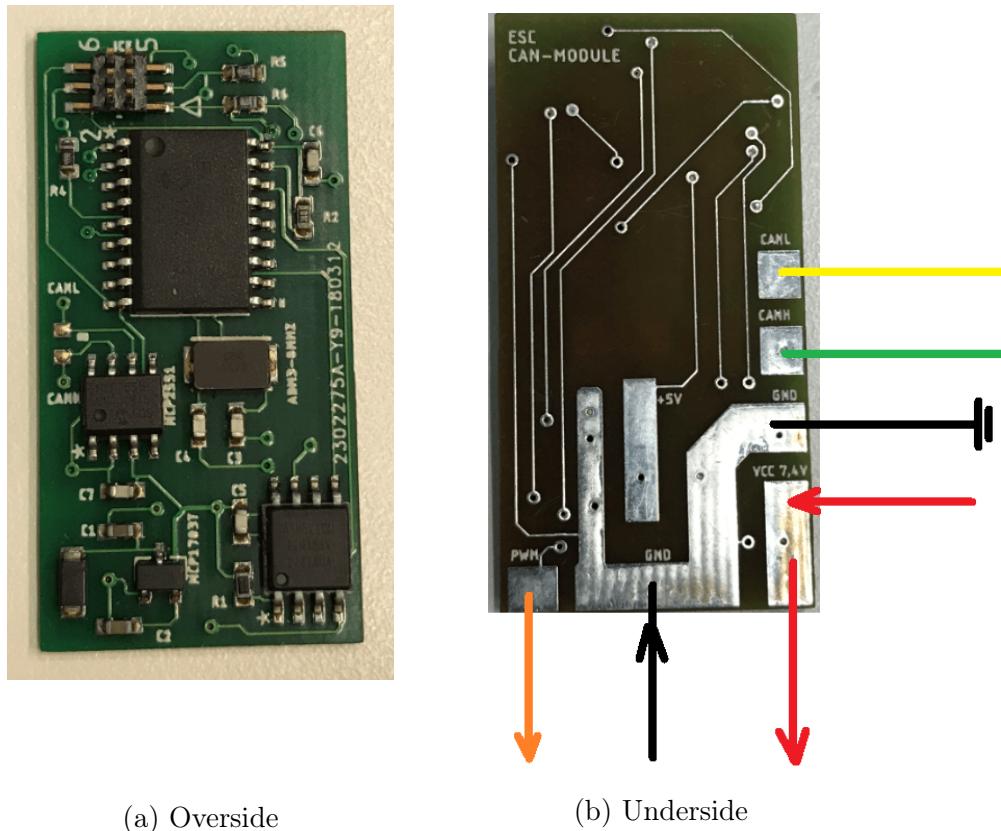


Figur 88: Drone i ladestasjon

Det er designet, utviklet og programmert tre typer kretskort. Alle kretskortene fungerer godt, både med tanke på software og hardware. Kortenes overflatemonterte mikrokontroller har tilknyttede programmeringsspinner, og kan enkelt programmeres ved hjelp av Atmel ICE eller annen ISP programmerer.

Supply Board og RPIzw CAN modul er designet for sine formål til dette prosjektet. De kan også nytties i andre applikasjoner, men de er ikke designet for å passe en hvilken som helst konfigurasjon uten å måtte gjøre endringer. ESC CAN modul har derimot blitt sterkt påvirket av ønsket om høy modularitet. Kortet er kompatibelt med både 2S og 3S batterier (7,4V-16V) [49]. Strømbanene, som tar inn strøm og sender videre til ESC er i utgangspunktet dimensjonert til å håndtere minst 5A. Dersom en tillater en temperatur stigning på 35°C kan de teoretisk sett belastes med 15A uten problem. Kortet mottar CAN bus meldinger med unik meldings-id som inneholder informasjon om motorenes pådrag, og genererer et korrespondere 1000-2000  $\mu$ s pulstog. Tanken bak er at kortet skal kunne tas rett ut av esken og brukes i en vilkårlig applikasjon der en ønsker å introdusere CAN bus for å kontrollere rc servoer eller børsteløse motorer.

Kortene har blitt brukt mye gjennom testperioden uten problemer og fremstår svært robuste. Ferdig resultat av ESC CAN modul er presentert under i figur 89.



Figur 89: Resultat ESC CAN modul

Bruken av ROS gjennom oppgaven har tillat oss å modularisere sofware plattformen, og dele denne opp i noder med respektive arbeidsoppgaver. Input og output fra virtuelle noder har blitt brukt for å teste systemet underveis, og blitt nyttiggjort for å verifisere funksjonaliteten til noder før hele systemet er ferdig. Innebygget ROS-funksjonalitet i Simulink er blitt utnyttet for blant annet å kunne sette PID parameter "live" under testing. Mulighet til å lytte på ROS emner har forenklet feilsøking underveis da all data som overføres mellom nodene kan overvåkes og logges.

Uthenting av posisjon og orientering fra MoCapS ga dårlige resultater. Systemet klarte ikke å følge merkede objekter i rommet på tilfredstilende måte, og mistet til stadighet helt eller delvis kontroll på flere av markørene over lengre perioder. Veileder har hatt en dialog med leverandør av systemet, uten å finne noen vedvarende løsninger. Med grunnlag i dette har MoCapS blitt nedprioritet i oppgaven til tross for at Qualisys oppgir god kompatibilitet med ROS og Simulink.

PID kontrollere laget med hensyn på den dekoblede matematiske modellen er testet og verifisert i den koblede modellen for se at de også her gir stabil flyvning. Tanken er å bruke disse kontrollerne ved implementasjon i den fysiske modellen, for videre å finjustere parameterne under praktisk testing.

Under avsluttende testing kunne det visuelt bekreftes at regulering og signalmiksing hadde ønsket virkemåte. Det ble først forsøkt å styre drone med vinkelposisjon, men drifting fra komplimentærfilter gjorde dette vanskelig. Videre testing ble forsøkt med vinkelrate styring. Her ble det oppnådd en stabil yaw- og roll-rate. Problemer med oppdateringsfrekvens fra tilkoblet ps4 kontroll gjorde tuning av pitch-rate vanskelig.

## 8 Diskusjon

### 8.1 Skrog

For å nevne noen forbedringspunkter som på skroget til diskusjon kan det være interessant å hente fram igjen duct prinsippet. Det må tas i betrakning at ved bruk av ducter måtte et alternativt skrogdesign med ducter innfelt blitt utarbeidet. Et kjapt forslag er å hente inspirasjon fra en UFO. Det ville redusert den ekstra massen ductene introduserer, som hadde gjort at en større andel av ekstra thrusten som blir generert hadde blitt nyttegjort til mer enn bare løfte ductenes egenvekt. Årsaker som kan ha begrenset ducten sitt potensiale, kan være den ujevne overflaten som 3D-printen gir. Den kan skape friksjon i luftstrømmen som kan forårsake metning. Dette kunne blitt forbedret ved å lage ducten i et material med en jevnere overflate, eksempelvis vakuumformet ABS-plast.

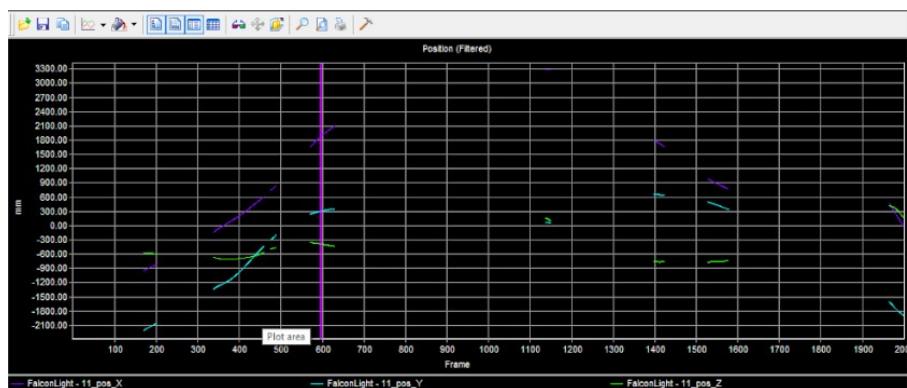
Videre kan det sies litt om størrelse. En oppskalering i dimensjon kunne vrt gunstig for sammenstillingen. Dersom en eksempelvis skulle gått bort fra lodding som kontakt-metode og heller brukt kontakter, kreves litt mer plass. Mer plass ville igjen fort gått utover vekt kravet. Det kan diskuteres for videre arbeid hva som er mest interessant av å implementere en gimball+kamera pakke på 40 gram, eller å tillate 40 ekstra gram til videre design av oppdatert versjon av dronen. Et oppdatert skrogdesign med tillgang på litt mer vekt har potensial til å bli meget optimal.

Gruppen opplevde under testing at de innkjøpte motor-driverene genererer en del varme som ikke blir ventilert ut. Det utviklet seg ikke til å bli et problem, men for sikkerhetssyld bør kjølekanaler for å kanalisere noe av luftstrømmen fra propellene gjennom skroget bli implementert i oppdatert design. Her er det igjen mulighet for å spare inn vekt, ved å kutte ut materiale fra konstruksjonen for å konstruere kjølekanaler. Med kjølekanaler bør det legges opp til full gjennomtrekk for kjøling av hele elektronikk pakken.

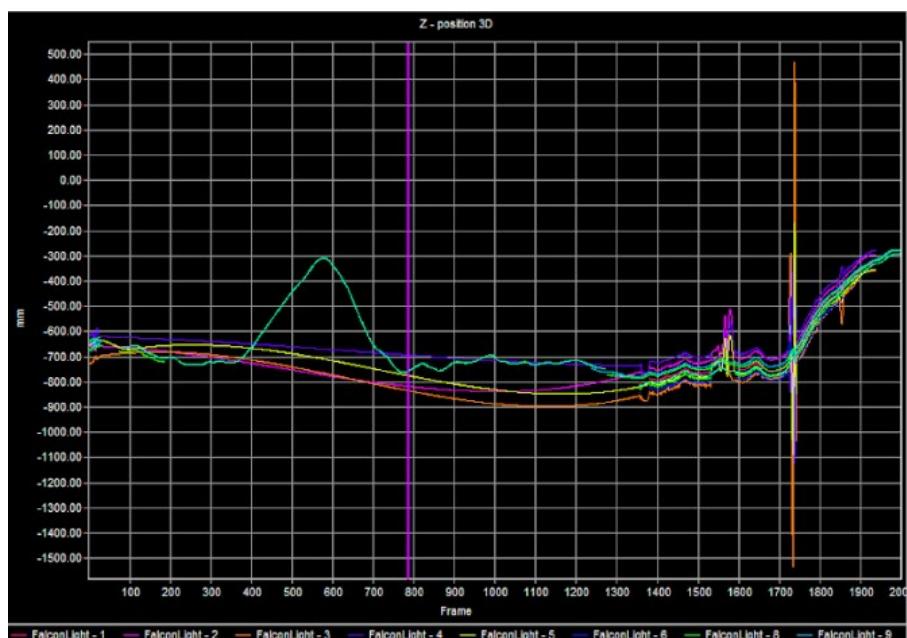
## 8.2 Qualisys

Posisjon og orientering fra Qualisys systemet skulle i utgangspunktet streames til gruppen, og oppgaven var kun tiltenkt å motta disse datene. Vi hadde ikke tilgang til systemet underveis, uten veileder tilstede, noe som gjorde feilsøking vanskelig. Veileder har vært i kontakt med leverandør og brukt mye tid på å konfigurerer systemet, uten å oppnå nødvendig resultat. Systemet bør antakeligvis konfigureres på nytt fra bunnen av i korrespondanse med leverandør. Kameraene bør da deles inn i riktige blitz-grupper for å unngå at enkelte kameraer oppfatter blitzingen fra andre kamera som en forstyrrelse. Leverandør har også uttrykt bekymring for sollyset som slipper inn gjennom vinduene i MotionLab. Slik systemet står idag er det mange potensielle feilkilder som kan forsake målinger.

Sporing av drone som definert legeme i X, Y og Z er vist i figur 90. Det sees tydelig at systemet ikke er i stand til å få tak orientering i alle tidsrammer.



Figur 90: diskontinuert data fra qualisys system



Figur 91: Sporing av markører i Z-akse

## 8.3 Kretskort

Kretskortene som er laget er prototyper, designet er dermed ikke hugget i stein. Som beskrevet innledningsvis i seksjon 6, må det etter resultat gjøres en design validering. Før en slipper delene videre til konklusjon og validerer kravene før en evt. setter igang masseproduksjon. Det som først må sies om kretskortene er at de har fungert meget godt, og har opprettholdt de krav som er stilt til dem i seksjon (6.8.1) for å kunne fly. De har også fremstått som meget robuste gjennom dronens testfase med tanke på både ESD og behandlingen de gjennomgår i en slik testperiode.

Noen punkter med hensyn til design kan likevel kommenteres. Nå som det er kartlagt og testet hvor godt reflow ovnen til Universitetet fungerer er det helt klart at ved en ny versjon bør QFN-pakken velges. Det ville redusert størrelsen betraktelig, særlig for MCP2515 som er den største av komponentene.

Opprinnelig var tanken at ESC CAN modul skulle plasseres i armene til dronen. Det viste seg å bli vanskelig å få fikk tidlig en plass under batteriet. Å vurdere firelags kort med komponenter plassert på begge sider kan minimalisere kortet nok til å muliggjøre plassering ute i armene, tett mot ESC. Kortene kunne da blitt laget små nok til å også tillate plass til kontakter for å stikke ledningene rett inn, fremfor loddning.

Software messig kan det nevnes at å ha en ekstern krystall til ATtiny85 mikrokontrolleren åpner for raskere interrupt rutiner, noe som gir høyere oppløsning på pulstoget og muligheter for flere pådragsnivåer.

Det viste seg at bruken av et såpass stort og komplekst bibliotek ga visse utfordringer og krevde en tilpasning for å fungere med eget designet system. Når all konfigurasjon var fullført fungerte det utmerket og ga oss et CAN bus grensesnitt med en lett og liten mikrokontroller

## 8.4 Modeller og regulering

Fra testing er det vist at reguleringspådrag virker, her er kontrollerne og reguleringssystemet brukt fra simuleringsmodellen vår. Kontrollerene er her bygd opp rundt SISO-tuning som forenkler prosessen med å beregne kontrollerparametere. Alle regulering har her tatt utgangspunkt i kontinuerlige signaler. Det har imidlertid har det vist seg at diskret regulering ville vært mere egnet for vårt system med hensyn på systemrate. For vår oppgave ble de kontinuerlige PID-kontrollere/kompensatorer transformert til diskret regulering gjennom en transformasjon i MATLAB. For videre arbeid burde det heller vært tatt utgangspunkt i å også lage simulator med tanke på diskret regulering.

Simulatoren vi endte opp med er en relativt enkel modell som ikke tar hensyn til sensor dynamikk. Det ble også brukt antagelser og data fra andre prosjekter for å modellere motordynamikken, i mangel på måleutstyr til å kjøre step- eller frekvens-analyse. For å kunne oppnå en mer presis modell burde ovennevnte dynamikk vært modellert inn. I tillegg til mulighet for å legge inn støy og forstyrrelser.

## 8.5 Testing

Det ble utført tester som viste at vinkelposisjon og vinkelrate fungerte i lukket sløyfe med IMU. Ved tilting av drone var det godt synlig at riktige motorer økte/reduserte turtall for å forsøke å holde dronen stabil.

Tilbakekobling med vinkelposisjon fra komplementærfilter ga lovende utslag på motorer under håndholdte tester. I videre testing viste det seg at filteret driftet i for stor grad til at det kunne brukes i videre testing. Filteret er som nevnt i oppgaven basert på utgitt modell fra Parrot. Det inneholder mange variabler som det trengs bedre kjennskap til for å gjøre en tilpasning til vår drone som er robust nok.

Under tester på bakken ble det da valgt tilbakekobling med vinkelrate. Innledningsvis roterte dronen ukontrollert rundt yaw-aksen. Dette ble korrigert med å legge inn en korrigerende offset i modellen. Dronen var nå stabil i yaw og roll. Når en skal fly farkoster på vinkelrate, kreves en hurtig og presis kommunikasjon med kontroller/radio. Under testing ble det brukt en ps4 kontroller med tilhørende ROS node. Kontrolleren er kompatibel med RPIzw innebygde bluetooth modul, men underveis opplevde gruppen sporadiske utfall av signal og generelt lave publiseringssrater fra kontroll. Dette gjorde det vanskelig utføre tilstrekkelig tuning av offset og forsterkning tilhørende pitch rate.

P, I og D parameterne til kontrollere brukt i praktiske tester er bestemt ved hjelp av forenklet matematisk modell og virket å ha fornuftige verdi gjennom testing.

## 9 Konklusjon

I dette prosjektet har hovedfokuset vært på å designe og utvikle en modulbasert drone. Dette er gjort gjennom å optimalisere skroget med tanke på designparametere som vekt, stivhet, masseproduserbarhet og forenklet lading. Videre er det designet, produsert og programmert egne kretskort for å modularisere elektronikken ved bruk av CAN bus. I tillegg er det blitt bygd opp en simulator for å kunne simulere dronens oppførsel ved bruk av den valgte reguleringstrategien. For å sette sammen hele prosjektet i software er ROS brukt, noe som også bidrar til modularitet i software.

Grunnet tekniske problemer med qualisys har ikke flyvning i samspill med motion capture blitt gjennomført. Før videre arbeid i forbindelse med dette systemet utføres, bør konfigurasjon av kameraer og tilhørende software gjennomgås og kvalitetssikres av leverandør. Problemer med komplimentærfilter og ps4-kontroller har gjort flyvning utenom systemet vanskelig.

Omfanget av å designe og bygge opp en drone med tilhørende software og hardware fra bunnen av er en stor oppgave. En drone har høy terskel for å oppnå en stabil flyvning da det stiller krav til at alle delsystemer fungerer i takt med en høy oppdateringsfrekvens.

Mindre droner stiller høyere krav til nøyaktighet og respons i reguleringssystemet. Det hadde dermed vært en stor fordel å gjøre en frekvensanalyse av motorene for å kunne modellere disse nøyaktig.

Gjennom denne rapporten er utviklingsarbeidet og grunnlaget bak de valgte løsninger godt dokumentert. Vi er fornøyd med de resultater som er oppnådd, på tross av overnevnte utfordringer. Resultatene danner et godt grunnlag for videre arbeid, ved bruk av hele eller deler av dette prosjektet.

## 10 Forkortelser, variabler og definisjoner

### Forkortelser

**ABS** Acrylonitrile Butadiene Styrene. 4, 39, 43, 44

**ADC** Analog to Digital Converter. 10, 57

**CAD** Computer-Aided Design. 3, 39

**CBM** Curvature-based mesh. 3, 45

**CCW** Counterclockwise. 42, 124

**CW** Clockwise. 42, 124

**DRC** Design Rule Checking. 61

**FEM** Finite Element Method. 3, 36, 41

**FLD** Fritt Legeme Diagram. 32

**ISP** In-System Programming. 58

**LiPo** Lithium Polymer. 6, 7

**MoCapS** Motion Capture System. 16

**OP-AMP** Operational Amplifier. 10, 58

**PCB** Printet Circuit Board. 9, 61

**QFN** Quad Flat No-leads package. 55, 103

**RO** RPAS-operatør (Remotely Piloted Aircraft Systems). 1

**SISO** Single-input single-output system. 1

**SM** Standard mesh. 3, 45

**SOIC** Small Outline Integrated Circuit. 55, 56

## Referanser

- [1] "Motion compensation in the norwegian motion laboratory | qualisys," <http://www.qualisys.com/stories/motion-compensation-in-the-norwegian-motion-laboratory/>.
- [2] "Forskrift om luftfartøy som ikke har fører om bord mv - lovdata, [online]," [https://lovdata.no/dokument/SF/forskrift/2015-11-30-1404#KAPITTEL\\_2](https://lovdata.no/dokument/SF/forskrift/2015-11-30-1404#KAPITTEL_2), 11 2015.
- [3] "Multirotor - wikipedia," <https://en.wikipedia.org/wiki/Multirotor>, 01 2018.
- [4] M. Gortolev, "Quadcopter vs hexacopter vs octocopter: Pros and cons [online]," <http://dronebly.com/quadcopter-vs-hexacopter-vs-octocopter-the-pros-and-cons>, 11 2014.
- [5] "The propeller explained - dokumentar, john gau," <https://www.youtube.com/watch?v=0bP2MH3LqvI>.
- [6] "- 230.000 nordmenn eier en drone - uas norway," <https://www.uasnorway.no/tall-fra-drone-norge/>.
- [7] M. B. at Nuts & Volts Magazine, "Build your own induction charger - nuts & volts magazine - for the electronics hobbyist," [http://www.nutsvolts.com/magazine/article/august2013\\_Bates](http://www.nutsvolts.com/magazine/article/august2013_Bates), 08 2013.
- [8] "Brushed vs brushless motors - think rc," <http://www.thinkrc.com/faq/brushless-motors.php>.
- [9] J. Catsoulis, *Designing Embedded Hardware, 2nd Edition*. O'Reilly, May 2005, ISBN: 978-0-596-00755-3.
- [10] "Solidworks - wikipedia," <https://en.wikipedia.org/wiki/SolidWorks>, 03 2018.
- [11] "Finite element method - wikipedia," [https://en.wikipedia.org/wiki/Finite\\_element\\_method#/The\\_structure\\_of\\_finite\\_element\\_methods](https://en.wikipedia.org/wiki/Finite_element_method#/The_structure_of_finite_element_methods), 02 2018.
- [12] H. R. S. ENGINEERING, "Curvature based mesh advantages in solidworks simulation," <https://hawkridgesys.com/blog/curvature-based-mesh-advantages>.
- [13] B. Ozcelik, A. Ozbay, and E. Demirbas, "Influence of injection parameters and mold materials on mechanical properties of abs in plastic injection molding," *International Communications in Heat and Mass Transfer*, vol. 37, no. 9, pp. 1359 – 1365, 2010. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0735193310001636>
- [14] markforged.com, "Onyx – markforged," <https://support.markforged.com/hc/en-us/articles/209934486-Onyx>.
- [15] ——, "Nylon – markforged," <https://support.markforged.com/hc/en-us/articles/209934406>.
- [16] I. Chopra, "Hover and wind-tunnel testing of shrouded rotors for improved micro air vehicle design," <https://drum.lib.umd.edu/bitstream/handle/1903/8752/umiumd5771.pdf?sequence=1>, 2008.

- [17] B. Schneider, “A guide to understanding lipo batteries,” <https://rogershobbycenter.com/lipoguide/>, 03 2017.
- [18] N. Seidle, “Battery technologies - learn.sparkfun.com,” <https://learn.sparkfun.com/tutorials/battery-technologies/lithium-polymer>, 02 2013.
- [19] Øyvind Nilsen, *Elektroteknikk i praksis, 1. utgave / 1.opplag.* Elforlaget, 2000, ISBN: 82-7345-324-3.
- [20] “Trace width calculator - qorvo,” <https://www.qorvo.com/design-hub/design-tools/trace-width-calculator>.
- [21] “*Generic Standard on Printed Board Design - IPC-2221A*,” Mai 2003.
- [22] M. Ottestad, “Formelsamling mas-200,” upublisert.
- [23] Atmel, “Atmel attiny25, attiny45, attiny85 datasheet,” [http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-2586-AVR-8-bit-Microcontroller-ATtiny25-ATtiny45-ATtiny85\\_Datasheet.pdf](http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-2586-AVR-8-bit-Microcontroller-ATtiny25-ATtiny45-ATtiny85_Datasheet.pdf), Jan 2013.
- [24] “Ros 101: Intro to the robot operating system | robohub,” <http://robohub.org/ros-101-intro-to-the-robot-operating-system/>.
- [25] G. Cai, B. M. Chen, and T. H. Lee, *Unmanned Rotorcraft Systems*. Springer, 2011, ISBN: 82-7345-324-3.
- [26] “Rotation matrix for rotations around x-axis - matlab rotx - mathworks nordic,” <https://se.mathworks.com/help/phased/ref/rotx.html>.
- [27] “Rotation matrix – from wolfram mathworld,” <http://mathworld.wolfram.com/RotationMatrix.html>.
- [28] “Orthogonal transformation – from wolfram mathworld,” <http://mathworld.wolfram.com/OrthogonalTransformation.html>.
- [29] J. Diebel, “Representing attitude: Euler angles, unit quaternions, and rotation vectors,” <https://www.swarthmore.edu/NatSci/mzucker1/papers/diebel2006attitude.pdf>, 10 2006.
- [30] A. Janota, V. Šimák, D. Nemec, and J. Hrbček, “Improving the precision and speed of euler angles computation from low-cost rotation sensor data,” <http://www.mdpi.com/1424-8220/15/3/7016/htm>.
- [31] M. H. Ang and V. D. Tourassis, “Singularities of euler and roll-pitch-yaw representations,” file:///C:/Users/J%C3%BCrgen%20Benum/Downloads/TM-55.pdf, 10 1986.
- [32] P. Singla, D. Mortari, and J. L. Junkins, “How to avoid singularity when using euler angles,” <http://lairs.eng.buffalo.edu/pdffiles/pconf/C10.pdf>.
- [33] A. Noureldin, T. B. Karamat, and J. Georgy, *Fundamentals of Inertial Navigation, Satellite-based Positioning and their Integration*. Springer, 2013, ISBN: 978-3-642-30465-1.
- [34] R. Stengel, “Aircraft equations of motion - flight path computation - 2016.pptx,” <https://www.princeton.edu/~stengel/MAE331Lecture9.pdf>.

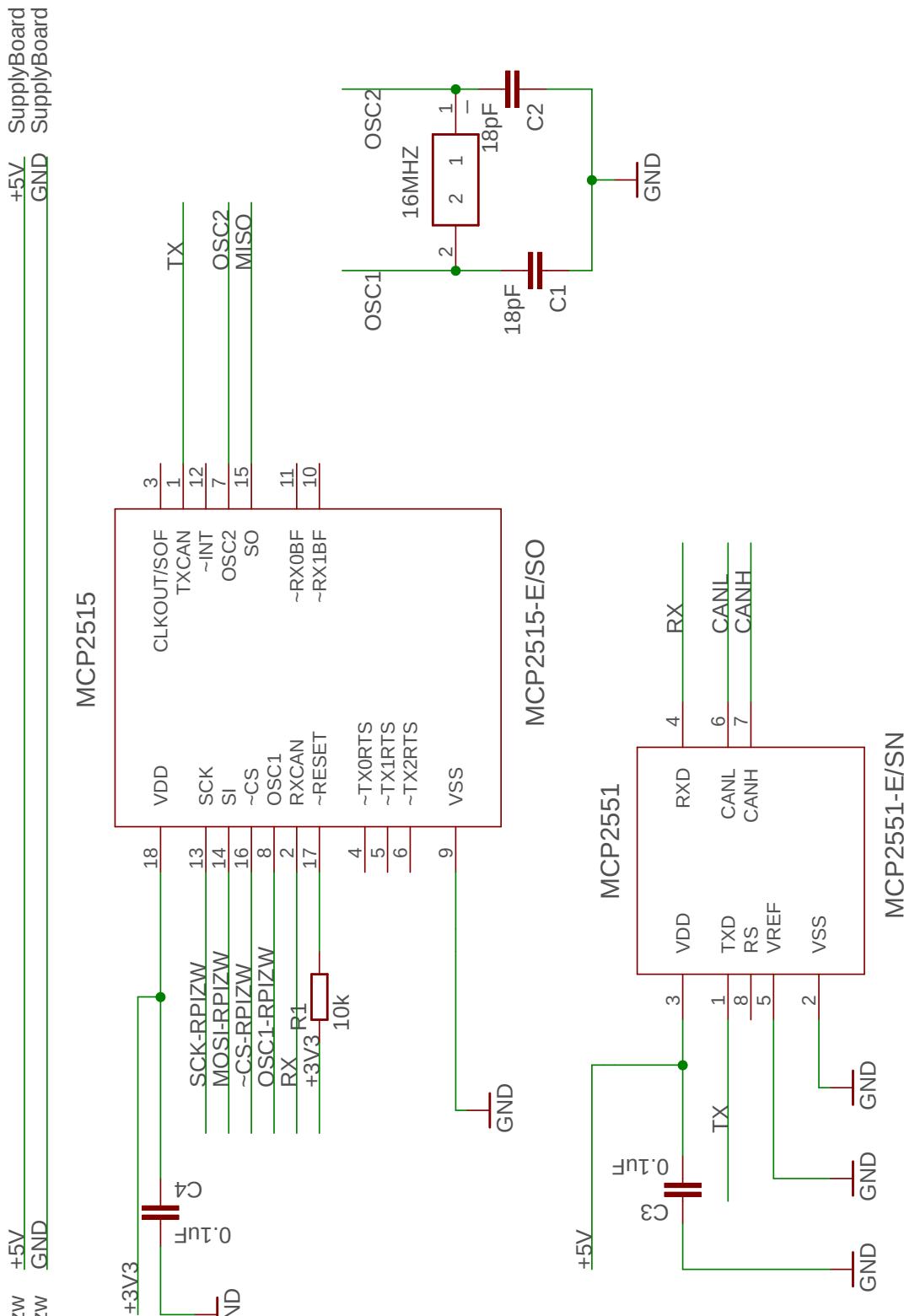
- [35] J. L. Meriam and L. G. Kraige, *Engineering: Mechanics Dynamics, seventh edition*. Wiley, 2013, ISBN: 978-1-118-08345-1.
- [36] J. L. Synge and B. A. Griffith, “Principles of mechanics, second edition,” 1949.
- [37] K. M. Passino and N. Quijano, “Proportional-integral-derivative control with derivative filtering and integral anti-windup for a dc servo,” <http://www2.ece.ohio-state.edu/~passino/lab3prelab.pdf>, 2002.
- [38] emax usa.com, “Emax rs1106 micro brushless motor (1 pcs) - emax usa - fpv racing products,” <http://emax-usa.com/emax-rs1106-micro-brushless-motor-1-pcs/>.
- [39] Unica AS, E-post, 17.04 2018.
- [40] HobbyKing, “Brushless motor f1104-4000kv cw,” [https://hobbyking.com/en\\_us/turnigy-f1104-4000kv-cw.html](https://hobbyking.com/en_us/turnigy-f1104-4000kv-cw.html).
- [41] “Thrust-test: Emax rs1106 4500kv,” <http://fishpepper.de/2017/11/29/thrust-test-emax-rs1106-4500kv/>, 2017.
- [42] “Emax rs1106 micro brushless motor (1 pcs) - emax usa - fpv racing products,” <http://emax-usa.com/emax-rs1106-micro-brushless-motor-1-pcs/>.
- [43] “matpalm/ros-mpu6050-node: raspberry pi c++ ros mpu6050 imu node,” <https://github.com/matpalm/ros-mpu6050-node>.
- [44] “solbach/ps4-ros: [ros] sony playstation 4 dualshock®4 node joy\_msg to twist\_msg,” <https://github.com/solbach/ps4-ros>.
- [45] “avr-libc: Avr libc,” <http://nongnu.org/avr-libc/user-manual/>, feb 2016.
- [46] ABRACON, “Abm3-pg2.pdf,” [http://www.farnell.com/datasheets/1754353.pdf?\\_ga=2.141071291.841878703.1525335446-196449175.1519034580](http://www.farnell.com/datasheets/1754353.pdf?_ga=2.141071291.841878703.1525335446-196449175.1519034580).
- [47] “dergraaf/avr-can-lib: Universal can library for avrs supporting at90can, mcp2515 and sja1000,” <https://github.com/dergraaf/avr-can-lib>.
- [48] Analog Devices, “Mt-101: Decoupling techniques,” <http://www.analog.com/media/en/training-seminars/tutorials/MT-101.pdf>, 2009.
- [49] Microchip, “250 ma, 16v, low quiescent current ldo regulator,” [http://www.farnell.com/datasheets/1268705.pdf?\\_ga=2.186539680.841878703.1525335446-196449175.1519034580](http://www.farnell.com/datasheets/1268705.pdf?_ga=2.186539680.841878703.1525335446-196449175.1519034580), 03 2011.
- [50] Texas Instruments, “Lp38691/93 500-ma low dropout cmos linear regs stable datasheet (rev. o),” <http://www.ti.com/lit/ds/symlink/lp38691.pdf>, 12 2015.
- [51] “Avr® microcontroller hardware design considerations,” <http://ww1.microchip.com/downloads/en/AppNotes/AN2519-AVR-Microcontroller-Hardware-Design-Considerations-00002519E.pdf>, 02 2018.
- [52] Atmel, “Attiny24/44/84,” <http://ww1.microchip.com/downloads/en/DeviceDoc/doc8006.pdf>, Jan 2010.

- [53] “S1al series\_o15.xls,” [http://www.farnell.com/datasheets/2357978.pdf?\\_ga=2.190720683.1400860144.1526310395-196449175.1519034580](http://www.farnell.com/datasheets/2357978.pdf?_ga=2.190720683.1400860144.1526310395-196449175.1519034580).
- [54] “Raspberry pi faqs - frequently asked questions,” <https://www.raspberrypi.org/help/faqs/#powerReqs>.
- [55] Texas Instruments, “Lm321 low power single op amp datasheet (rev. c),” <http://www.ti.com/lit/ds/symlink/lm321.pdf>, des 2014.
- [56] Microchip, “Mcp2551 high-speed can transceiver,” <http://ww1.microchip.com/downloads/en/DeviceDoc/20001667G.pdf>, 12 2016.
- [57] ——, “Mcp2515 - stand-alone can controller with spi interface,” [http://www.farnell.com/datasheets/1669372.pdf?\\_ga=2.82418591.841878703.1525335446-196449175.1519034580](http://www.farnell.com/datasheets/1669372.pdf?_ga=2.82418591.841878703.1525335446-196449175.1519034580), 12 2016.
- [58] “Jlcpcb - capabilities,” <https://jlcpcb.com/capabilities/Capabilities>.
- [59] “Turnigy multistar bl-arm 32bit 7a race spec esc 1~2s (opto),” [https://hobbyking.com/en\\_us/turnigy-multistar-blheli-arm-32bit-7a-0-8g-race-spec-esc-1-2s-opto.html](https://hobbyking.com/en_us/turnigy-multistar-blheli-arm-32bit-7a-0-8g-race-spec-esc-1-2s-opto.html).
- [60] “Quad-sim/quadcopter dynamic modeling and simulation,” <https://github.com/dch33/Quad-Sim/tree/master/Quadcopter%20Dynamic%20Modeling%20and%20Simulation/Simulation%20Files>.
- [61] N. S. Nise, *Control Systems Engineering, sixth edition*. Wiley, 2011, ISBN: 978-0-470-64612-0.
- [62] “Copter attitude control — dev documentation,” <http://ardupilot.org/dev/docs/apmcopter-programming-attitude-control-2.html>.
- [63] Invensense, *MPU-6050 Register Map and Descriptions*, <https://www.invensense.com/wp-content/uploads/2015/02/MPU-6000-Register-Map1.pdf>.
- [64] Øyvind Magnussen, M. Ottestad, and G. Hovland, “Calibration procedure for an inertial measurement unit using a 6-degree-of-freedom hexapod,” <https://home.uia.no/geirh/PDF/C52.pdf>.

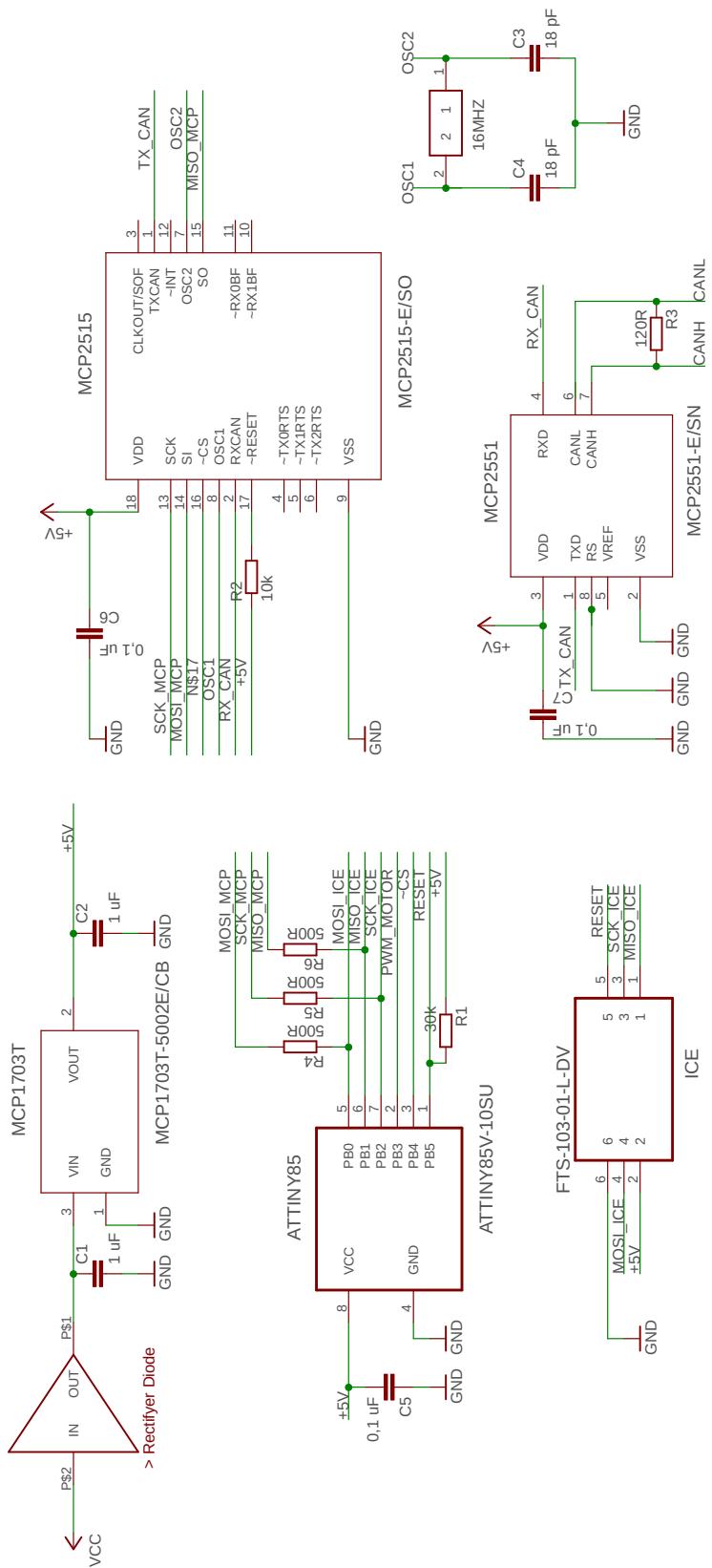
# Appendiks

## A Skjematiskk

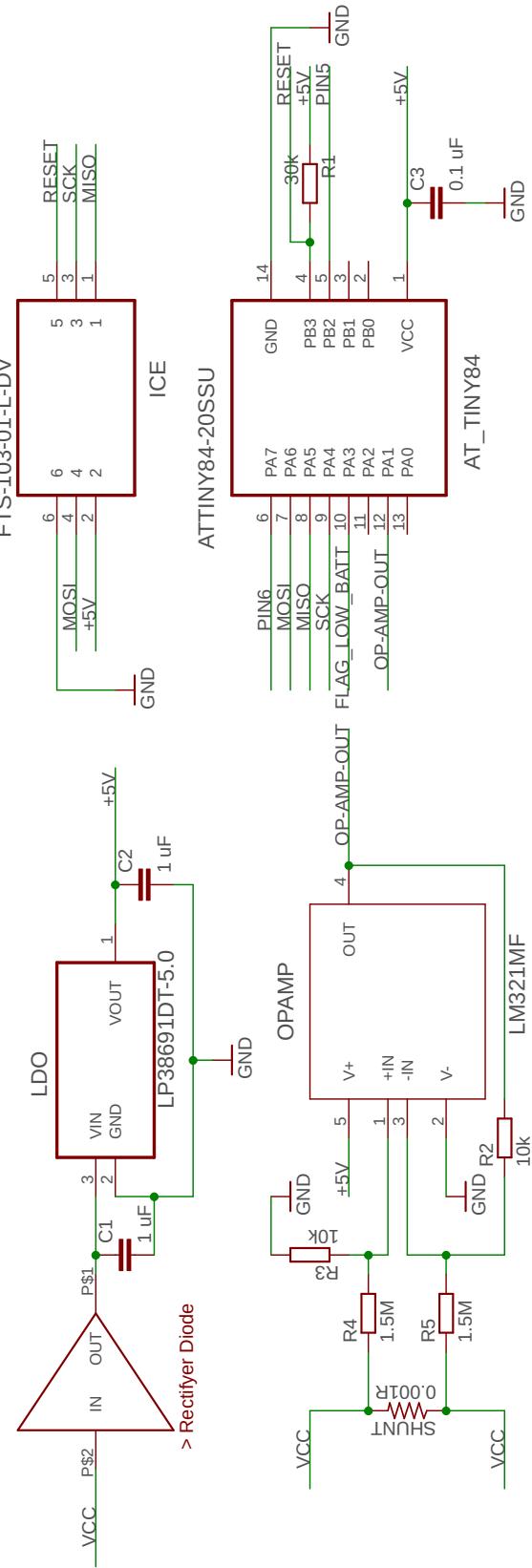
### A.1 RPIzw CAN module



## A.2 ESC CAN module



### A.3 Supply Board



## B Kildekode

### B.1 ESC CAN modul

```
1
2
3 /*
4  * Motor1.c
5  *
6  * Created: 26.03.2018 14:50:46
7  * Author : ocvaa
8 */
9
10 // coding: utf-8
11
12 #include <avr/io.h>
13 #include <avr/pgmspace.h>
14 #include <avr/interrupt.h>
15
16 #include "can.h"
17
18 const uint8_t can_filter [] PROGMEM =      // Tar kun i mot meldinger med Id = 0
19     x20
20 {
21     // Gruppe 0
22     MCP2515_FILTER(0x20),           // Filter 0
23     MCP2515_FILTER(0x20),           // Filter 1
24
25     // Gruppe 1
26     MCP2515_FILTER(0x20),           // Filter 2
27     MCP2515_FILTER(0x20),           // Filter 3
28     MCP2515_FILTER(0x20),           // Filter 4
29     MCP2515_FILTER(0x20),           // Filter 5
30
31     MCP2515_FILTER(0xFF),          // Maske 0 (for gruppe 0) 0xFF gir deg tilgang til
32     alle filter
33     MCP2515_FILTER(0xFF),          // Maske 1 (for gruppe 1)
34 };
35
36 //***** Global variabels *****
37 uint16_t interruptTeller = 0;
38 uint8_t styrke;
39 uint8_t maksStyrke = 100;
40 int watchDogTeller = 0;
41 uint8_t watchdogDelaySec = 1;
42 uint16_t pwmFrekvens = 250;
43
44 ISR(TIMER0_COMPA_vect)
45 {
46     if (interruptTeller <= 100)    // Setter utgang hoy de forste 1000 us uavhengig
47         verdi pa styrke.
48     {
49         PORTB |= 0b00001000;      //PB3 hoy
50     }
51 }
```

```

48     else if (interruptTeller >= 100 + styrke) //Legger pa nodvendig periodetid pa
        pulstog basert pa "styrke" sin verdi. Ved styrke = 100 gir hoy utgang i 2000
        us .
49     {
50         PORTB &= 0b11110111;      //PB3 lav
51     }
52
53     interruptTeller++;
54
55     if (interruptTeller > 399) //Etter 400 interrupts resettes interrupt teller ,
        dette gir pulstog-frekvens pa 250Hz
56     {
57         interruptTeller = 0;
58         watchDogTeller++; //watchDogTeller inkrementeres for hver pwm syklus , og
        resettes hver gang en CAN-melding mottas.
59     }
60 }
61
62
63 int main(void)
64 {
65     //Settter PB3 som utgang
66     DDRB |= 0b00001000;
67     //Sikrer at TCCR0A/B har default verdier
68     TCCR0A = 0;
69     TCCR0B = 0;
70     //(1 << WGM01) --> CTC Mode
71     TCCR0A |= (1 << WGM01);
72     //Prescaler 1
73     TCCR0B |= (1 << CS00);
74     //Aktiverer interrupt flagget ved overflow pa timer0
75     TIMSK |= (1 << OCIE0A);

76
77     //Verdi som sammenlignes med TCNT0 og genererer overflow interrupt om TCNT0 >
        OCR0A. OCR0A = 79 => 10000Hz/10us interrupt
78     OCR0A = 79;
79     //Aktiverer Global interrupt
80     sei();
81
82     // Initialize MCP2515
83     can_init(BITRATE_125_KBPS);
84
85     // Load filters and masks
86     can_static_filter(can_filter);
87
88     while (1)
89     {
90         // Sjekker etter meldinger
91         if (can_check_message())
92         {
93             can_t msg;
94
95             // Leser melding
96             if (can_get_message(&msg))
97             {
98                 styrke = msg.data[0];

```

```

99     watchDogTeller = 0;
100    if (styrke > maksStyrke)
101    {
102        //Om det skrives verdi hoyere enn 100, settes styrke til 100
103        styrke = maksStyrke;
104    }
105    }
106    if (watchDogTeller > watchdogDelaySec * pwmFrekvens) //WatchdogTeller
107        inkrementeres for hver pwm syklus, og resettes hver gang en CAN-melding
108        mottas.
109        {
110            //pwmFrekvens = 250Hz gir 250 pwm sykluser per
111            sekund.
112            styrke = 0; //WatchDogDelay i sekunder kan da ganges inn
113            for a velge antall sekunder-
114        }
115    }
116    return 0;
117 }
```

## B.2 motor\_controller ROS node

```
1 #!/usr/bin/python
2
3 import rospy
4 from sensor_msgs.msg import RegionOfInterest
5
6 import can
7
8 bus = can.interface.Bus(bustype='socketcan', channel='can0', bitrate=500000)
9
10 motor1 = can.Message(arbitration_id=0x20, data=[0], extended_id=False)
11
12 motor2 = can.Message(arbitration_id=0x21, data=[0], extended_id=False)
13
14 motor3 = can.Message(arbitration_id=0x22, data=[0], extended_id=False)
15
16 motor4 = can.Message(arbitration_id=0x23, data=[0], extended_id=False)
17
18 def callback(data):
19     motor1.data[0] = data.x_offset
20     motor2.data[0] = data.y_offset
21     motor3.data[0] = data.height
22     motor4.data[0] = data.width
23
24     bus.send(motor1)
25     bus.send(motor2)
26     bus.send(motor3)
27     bus.send(motor4)
28
29
30
31
32 def listener():
33
34     # In ROS, nodes are uniquely named. If two nodes with the same
35     # name are launched, the previous one is kicked off. The
36     # anonymous=True flag means that rospy will choose a unique
37     # name for our 'listener' node so that multiple listeners can
38     # run simultaneously.
39     rospy.init_node('listener', anonymous=True)
40
41     rospy.Subscriber('MotorSetpoint', RegionOfInterest, callback)
42
43     # spin() simply keeps python from exiting until this node is stopped
44     rospy.spin()
45
46 if __name__ == '__main__':
47     listener()
```

### B.3 MPU6050 ROS node

```
1 #include <ros/ros.h>
2 #include <sensor_msgs/Imu.h>
3 #include <wiringPi.h>
4 #include <wiringPiI2C.h>
5
6 using namespace std;
7
8 const int I2C_ADDR = 0x68;
9 const int PWR_MGMT_1 = 0x6B; //original value 0x6B
10 const int USER_CTRL = 0x6A; //fifo register disasble
11 const int SMPRT_DIV = 0x19; //sample rate divider register
12 const int INT_ENABLE = 0x38; //interrupt enable
13 const int INT_STATUS = 0x39; //interrup data ready bit
14 const int DATA_RDY_INT = 0; //bit0 in INT_STATUS
15
16
17 float read_word_2c(int fd, int addr) {
18     int high = wiringPiI2CReadReg8(fd, addr);
19     int low = wiringPiI2CReadReg8(fd, addr+1);
20     int val = (high << 8) + low;
21     return float((val >= 0x8000) ? -((65535 - val) + 1) : val);
22 }
23
24 int main(int argc, char **argv) {
25
26     // Connect to device.
27     int fd = wiringPiI2CSetup(I2C_ADDR);
28     if (fd == -1) {
29         printf("no i2c device found?\n");
30         return -1;
31     }
32
33     wiringPiI2CWriteReg16(fd, USER_CTRL, 0); //FIFO register disable
34     wiringPiI2CWriteReg16(fd, SMPRT_DIV, 0x4F); //4F tilsvarer 100Hz sample
35     wiringPiI2CWriteReg16(fd, INT_ENABLE, 0x01); //enabler data ready interrupt(
36     DATA_RDY_EN)
37
38     // Device starts in sleep mode so wake it up.
39     wiringPiI2CWriteReg16(fd, PWR_MGMT_1, 0);
40
41
42     // Start ROS node stuff.
43     ros::init(argc, argv, "mpu6050");
44     ros::NodeHandle node;
45     ros::Publisher pub = node.advertise<sensor_msgs::Imu>("imu", 1);
46
47     // Publish in loop.
48     while(ros::ok()) {
49         sensor_msgs::Imu msg;
50         const int interruptStatus = wiringPiI2CReadReg16(fd, INT_STATUS) ;
51         const bool dataIsReady = interruptStatus > 0; //& (1U << DATA_RDY_INT) ;
52     }
```

```

53
54     if( !dataIsReady )
55     {
56         continue;           //Hopper til starten av while-lokken om ikke ny data er klar
57     }
58
59     msg.header.stamp = ros::Time::now();
60     msg.header.frame_id = '0'; // no frame
61
62
63     // Read gyroscope values.
64     // At default sensitivity of 250deg/s we need to scale by 131.
65     msg.angular_velocity.x = read_word_2c(fd, 0x43) * 3.14159265359 / (131*180);
66     msg.angular_velocity.y = read_word_2c(fd, 0x45) * 3.14159265359 / (131*180);
67     msg.angular_velocity.z = read_word_2c(fd, 0x47) * 3.14159265359 / (131*180);
68
69     // Read accelerometer values.
70     // At default sensitivity of 2g we need to scale by 16384.
71     // Note: at "level" x = y = 0 but z = 1 (i.e. gravity)
72     // But! Imu msg docs say acceleration should be in m/s^2 so need to *9.807
73     const float la_rescale = 16384.0 / 9.807;
74     msg.linear_acceleration.x = read_word_2c(fd, 0x3b) / la_rescale;
75     msg.linear_acceleration.y = read_word_2c(fd, 0x3d) / la_rescale;
76     msg.linear_acceleration.z = read_word_2c(fd, 0x3f) / la_rescale;
77
78     // Pub & sleep.
79     pub.publish(msg);
80     ros::spinOnce();
81     //rate.sleep();
82 }
83 return 0;
84 }
```

## B.4 PS4controller ROS NODE

```
1 #include <string>
2 #include <ros/ros.h>
3 #include <sensor_msgs/Joy.h>
4 #include <geometry_msgs/Twist.h>
5
6 class PS4_ROS {
7 public:
8
9 /**
10 * @brief      { PS4 to TWIST MESSAGES }
11 *
12 */
13
14 PS4_ROS(ros::NodeHandle &n) {
15     // get ros param
16     ros::NodeHandle private_nh("~");
17     private_nh.param("scale_linear", this->scale_linear, 1.0);
18     private_nh.param("scale_angular", this->scale_angular, 1.0);
19     private_nh.param<std::string>("pub_topic", this->pubName, "/searchbot/
p3at/vel_cmd");
20
21     this->chat = n.advertise<geometry_msgs::Twist>(pubName, 1);
22     this->sub = n.subscribe<sensor_msgs::Joy>("/joy", 1, &PS4_ROS::
23 subscribePS4, this);
24
25     /* set calibration counter to zero */
26     this->calib1 = 0;
27     this->calib2 = 0;
28     this->calib = false;
29
30     this->maxVel = this->scale_linear;
31     this->maxVelR = this->scale_linear * -1;
32
33     ROS_INFO("maxVelR: %f", this->maxVelR);
34
35     ROS_INFO("scale_linear set to: %f", this->scale_linear);
36     ROS_INFO("scale_angular set to: %f", this->scale_angular);
37     ROS_INFO("PS4_ROS initialized");
38 }
39 ~PS4_ROS() {
40     // std::cout << "Destroy the pointer" << std::endl;
41 }
42
43 void run() {
44     if(this->calib) {
45         this->publishTwistMsg();
46     }
47 }
48
49 void prepareData()
50 {
51     // Normalize velocity between ]-1.0, 1.0[
```

```

52     this->send_l2 = (-0.5 * this->l2) + 0.5;
53     this->send_r2 = (this->r2 - 1.0) * 0.5;
54
55     // Apply rosparam "scale_linear"
56     this->send_l2 = this->scale_linear * this->send_l2;
57     this->send_r2 = this->scale_linear * this->send_r2;
58
59     // Apply rosparam "scale_angular"
60     this->send_leftStickX = this->scale_angular * this->leftStickX;
61 }
62
63 void publishTwistMsg() {
64     geometry_msgs::Twist msg;
65     msg.linear.x = 0.0;
66     msg.linear.y = 0.0;
67     msg.linear.z = 0.0;
68
69     msg.angular.x = 0.0;
70     msg.angular.y = 0.0;
71     msg.angular.z = 0.0;
72
73     prepareData();
74
75     //printRaw();
76     if (!this->buttonTouch) {
77         if (((this->send_l2 >= 0.1) && (this->send_l2 <= maxVel)) {
78             msg.linear.x = this->send_l2;
79         } else if (((this->send_r2 <= 0.0) && (this->send_r2 >= maxVelR))) {
80             msg.linear.x = this->send_r2;
81         }
82         msg.angular.z = this->send_leftStickX;
83     }
84     else{
85         //ROS_WARN("SENDING EMERGENCY STOP");
86
87         /* To Do */
88     }
89
90     this->chat.publish(msg);
91 }
92
93 void subscribePS4(const sensor_msgs::Joy::ConstPtr &joy) {
94     this->buttonSq = joy->buttons[0];
95     this->buttonX = joy->buttons[1];
96     this->buttonO = joy->buttons[2];
97     this->buttonTr = joy->buttons[3];
98     this->buttonTouch = joy->buttons[13];
99     this->l1 = joy->buttons[4];
100    this->r1 = joy->buttons[5];
101
102    this->arrowsX = joy->axes[9];
103    this->arrowsY = joy->axes[10];
104    this->l2 = joy->axes[3];
105    this->r2 = joy->axes[4];
106    this->leftStickX = joy->axes[0];
107    this->leftStickY = joy->axes[1];

```

```

108     this->rightStickX = joy->axes[2];
109     this->rightStickY = joy->axes[5];
110
111     //printRaw();
112
113 }
114
115 bool calibrate()
116 {
117     double progress = ((double) this->calib1 / this->calib_duration) * 100;
118     if( (this->l2 == -1.0) && (this->r2 == -1.0) )
119     {
120         ROS_WARN("Press L2 and R2 to calibrate: %i%%", (int) progress);
121         this->calib1++;
122         this->calib2++;
123     }
124     else{
125         this->calib1 = 0;
126         this->calib2 = 0;
127     }
128
129     if( (this->calib1 > this->calib_duration) && (this->calib2 > this->
130         calib_duration))
131     {
132         this->calib = true;
133         return true;
134     }
135     else
136         return false;
137 }
138
139 bool waitForRelease()
140 {
141     if( (this->l2 == 1.0) && (this->r2 == 1.0) )
142     {
143         return true;
144     }
145     else{
146         return false;
147     }
148 }
149
150 void printSend()
151 {
152     ROS_INFO("#####");
153     ROS_INFO("Send L2: %f", this->send_l2);
154     ROS_INFO("Send R2: %f", this->send_r2);
155     ROS_INFO("##### \n");
156 }
157
158 void printRaw()
159 {
160
161     ROS_INFO("#####");
162     ROS_INFO("Squared Button pressed: %i", this->buttonSq);

```

```

163     ROS_INFO("X Button pressed: %i", this->buttonX);
164     ROS_INFO("O Button pressed: %i", this->buttonO);
165     ROS_INFO("Triangel Button pressed: %i", this->buttonTr);
166     ROS_INFO("Left/Right Button pressed: %i", this->arrowsX);
167     ROS_INFO("Down/Up Button pressed: %i", this->arrowsY);
168     ROS_INFO("Touch Button pressed: %i", this->buttonTouch);
169     ROS_INFO("L1: %i", this->l1);
170     ROS_INFO("R1: %i", this->r1);
171     ROS_INFO("L2: %f", this->l2);
172     ROS_INFO("R2: %f", this->r2);
173     ROS_INFO("Left Stick Y: %f", this->leftStickY);
174     ROS_INFO("Left Stick X: %f", this->leftStickX);
175     ROS_INFO("Right Stick Y: %f", this->rightStickY);
176     ROS_INFO("Right Stick X: %f", this->rightStickX);
177     ROS_INFO("# ##### \n");
178 }
179
180 private:
181     ros::Publisher chat;
182     ros::Subscriber sub;
183
184     /* calibration variables */
185     int calib_duration = 20; // 1/10 sec
186     int calib1, calib2;
187     bool calib;
188
189     /* raw data */
190     double leftStickY, leftStickX, rightStickY, rightStickX, l2, r2;
191     int arrowsX, arrowsY, buttonSq, buttonX, buttonO, buttonTr,
192         buttonTouch, l1, r1;
193
194     /* rosparams */
195     double scale_linear, scale_angular;
196     std::string pubName;
197
198     double maxVel, maxVelR;
199
200     /* send data */
201     double send_leftStickX, send_l2, send_r2;
202
203 };
204
205 int main(int argc, char **argv) {
206     ros::init(argc, argv, "PS4_ROS");
207     ros::NodeHandle n;
208
209     // create ps4_ros object
210     PS4_ROS *ps4_ros = new PS4_ROS(n);
211
212     // calibrate
213     ROS_WARN("Press L2 and R2 to calibrate");
214     bool ready = false;
215     while (!ready)
216     {
217         ready = ps4_ros->calibrate();
218         ros::spinOnce();

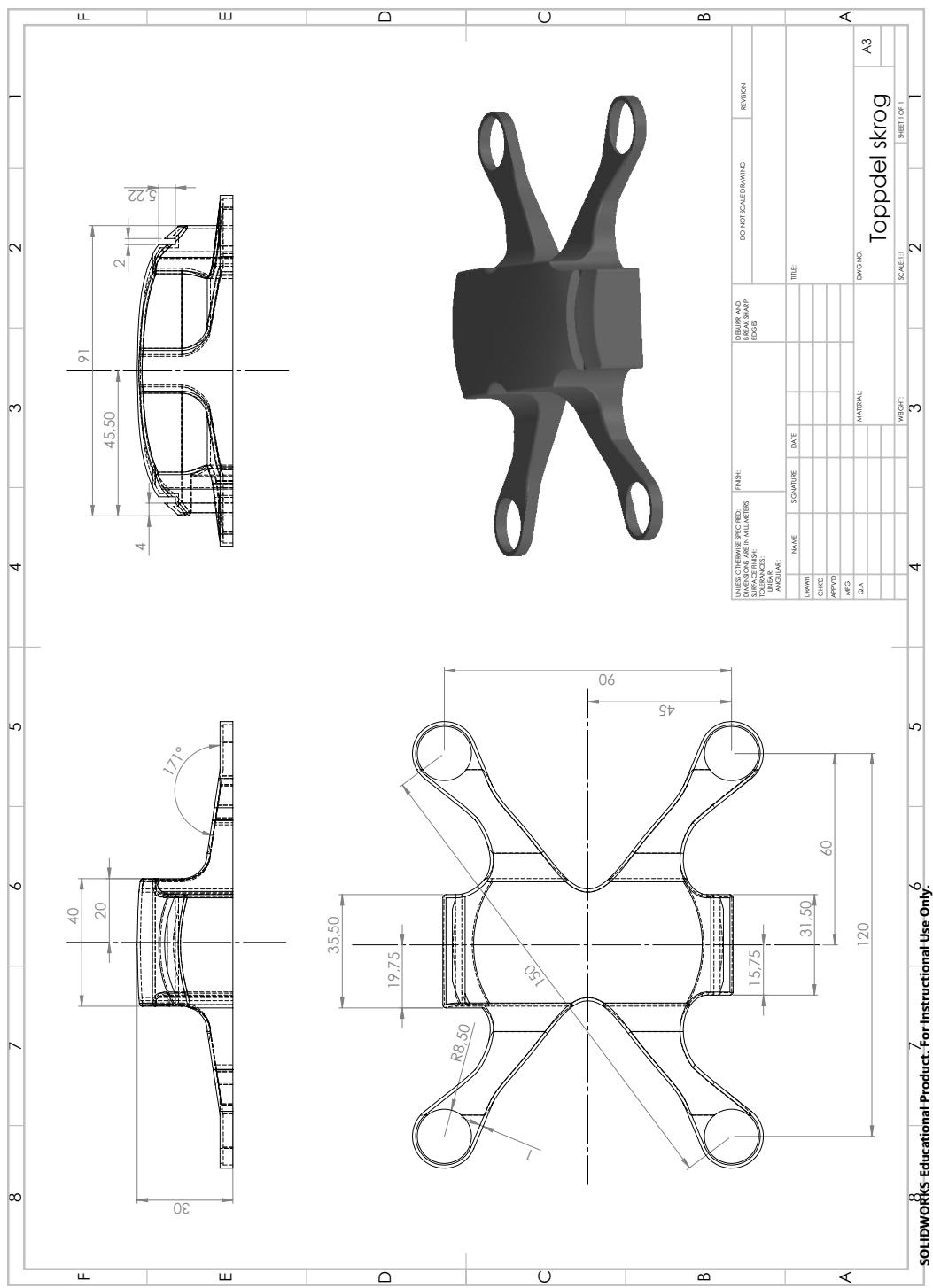
```

```

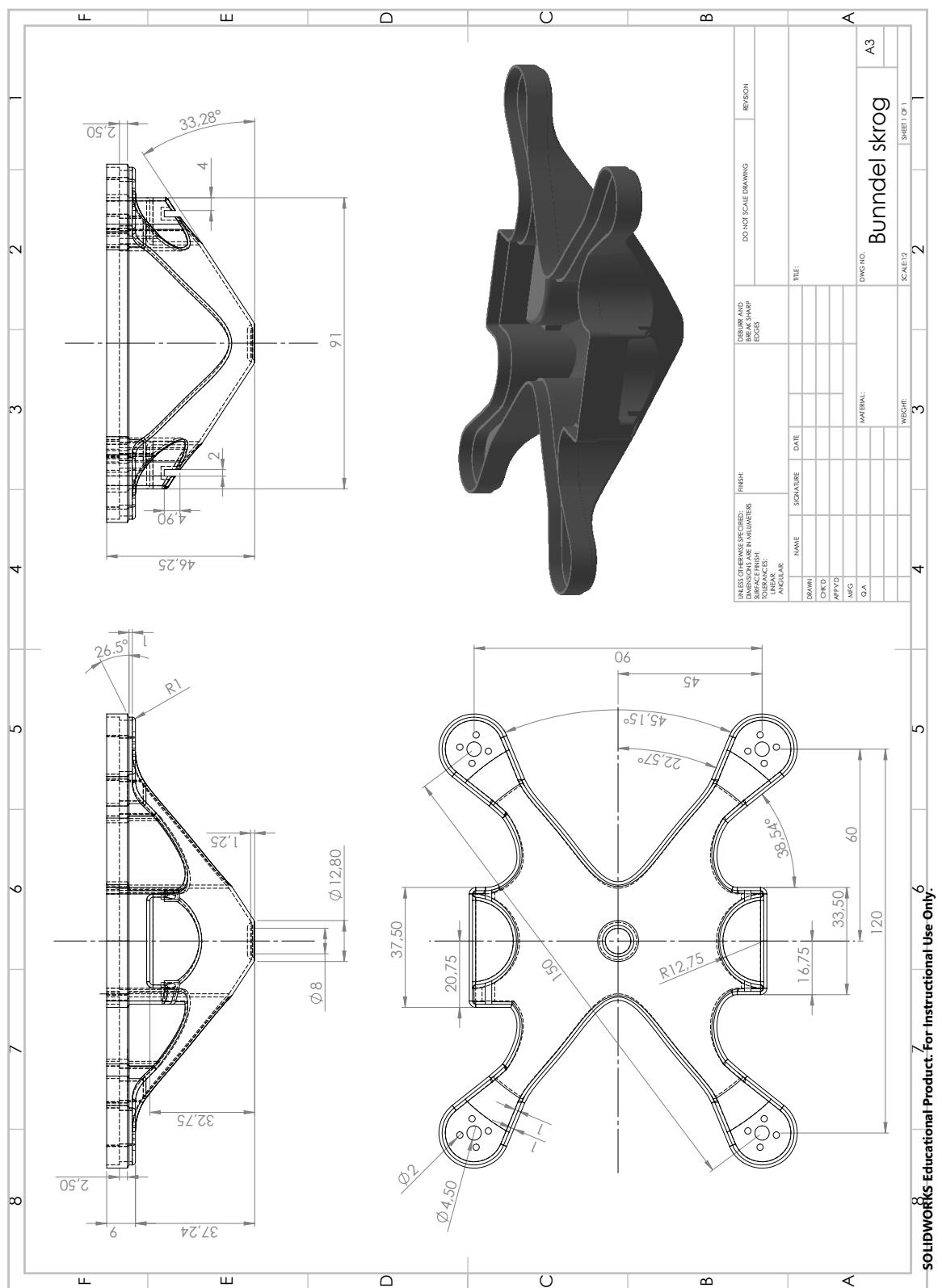
219     ros::Duration(0.1).sleep();
220 }
221
222 ROS_WARN("Release L2 and R2");
223 ros::Duration(2.0).sleep();
224 ready = false;
225 while (!ready)
226 {
227     ready = ps4_ros->waitForRelease();
228     ros::spinOnce();
229     ros::Duration(0.1).sleep();
230 }
231 ROS_INFO("Calibrated - Ready to use");
232
233 ros::Rate loop_rate(10);
234 while (ros::ok())
235 {
236     ps4_ros->run();
237     ros::spinOnce();
238     loop_rate.sleep();
239 }
240
241 delete ps4_ros;
242 return 0;
243 }
```

## C Tekniske tegninger

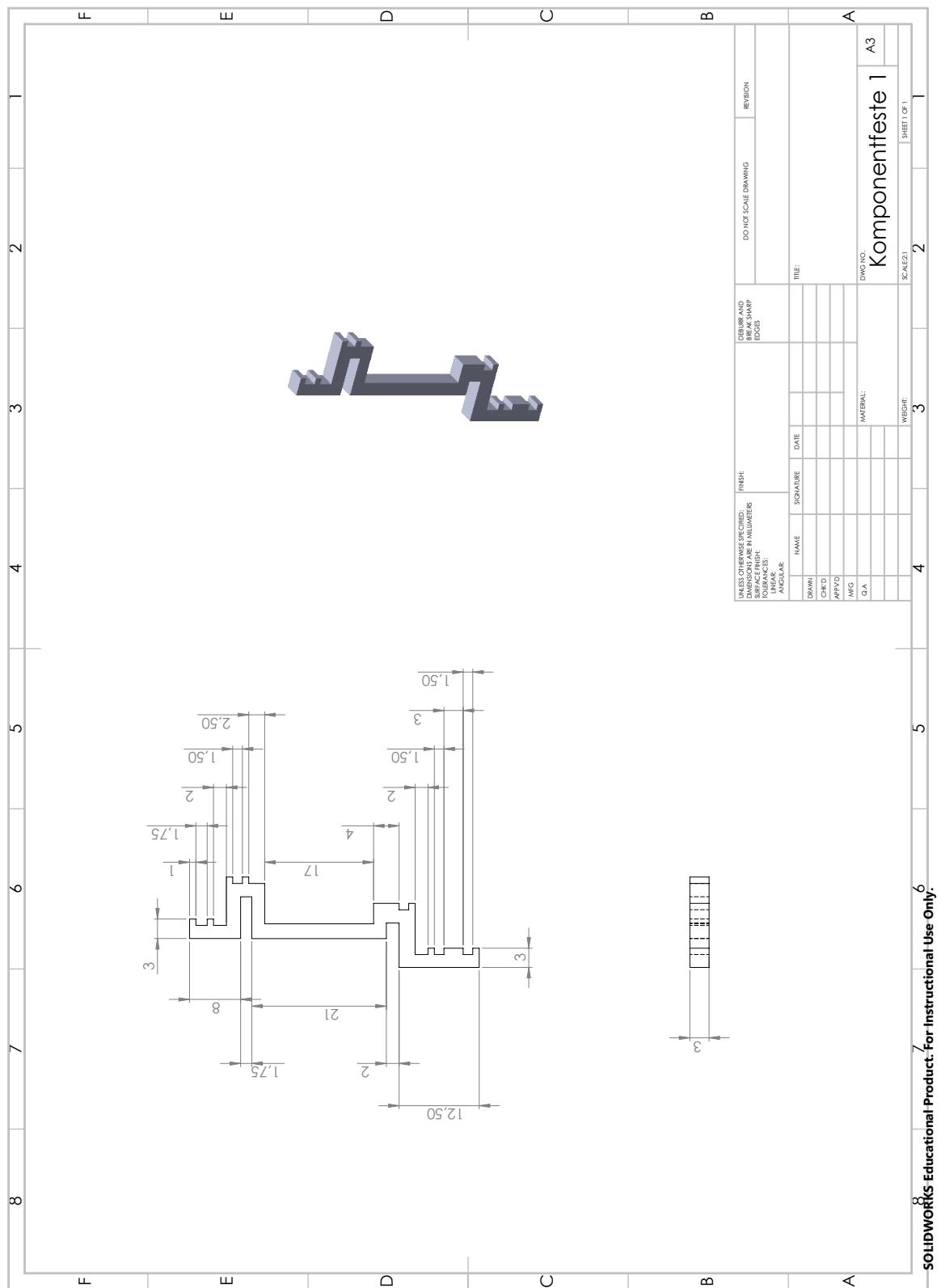
### C.1 Toppdel skrog



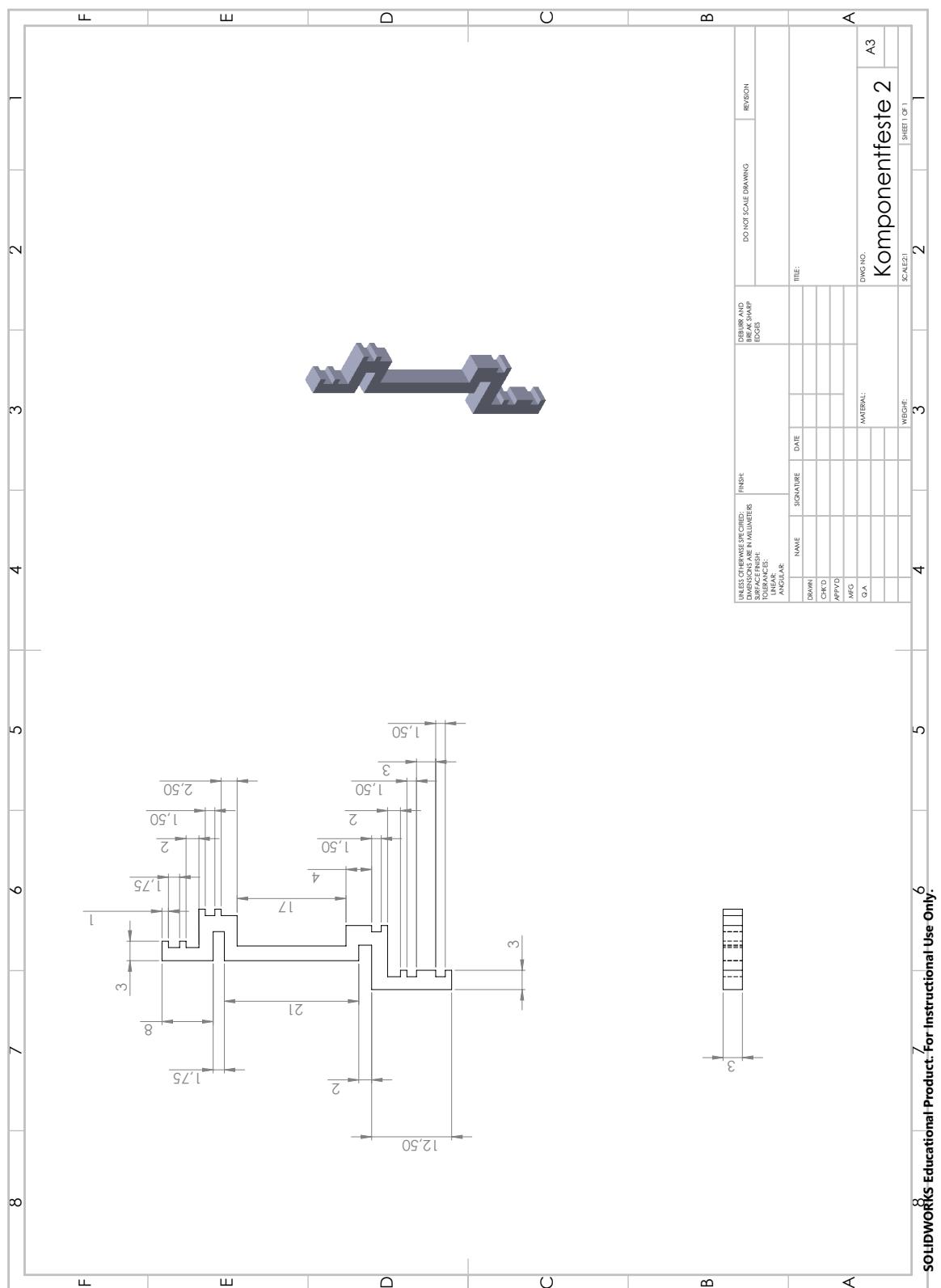
## C.2 Bunndel skrog



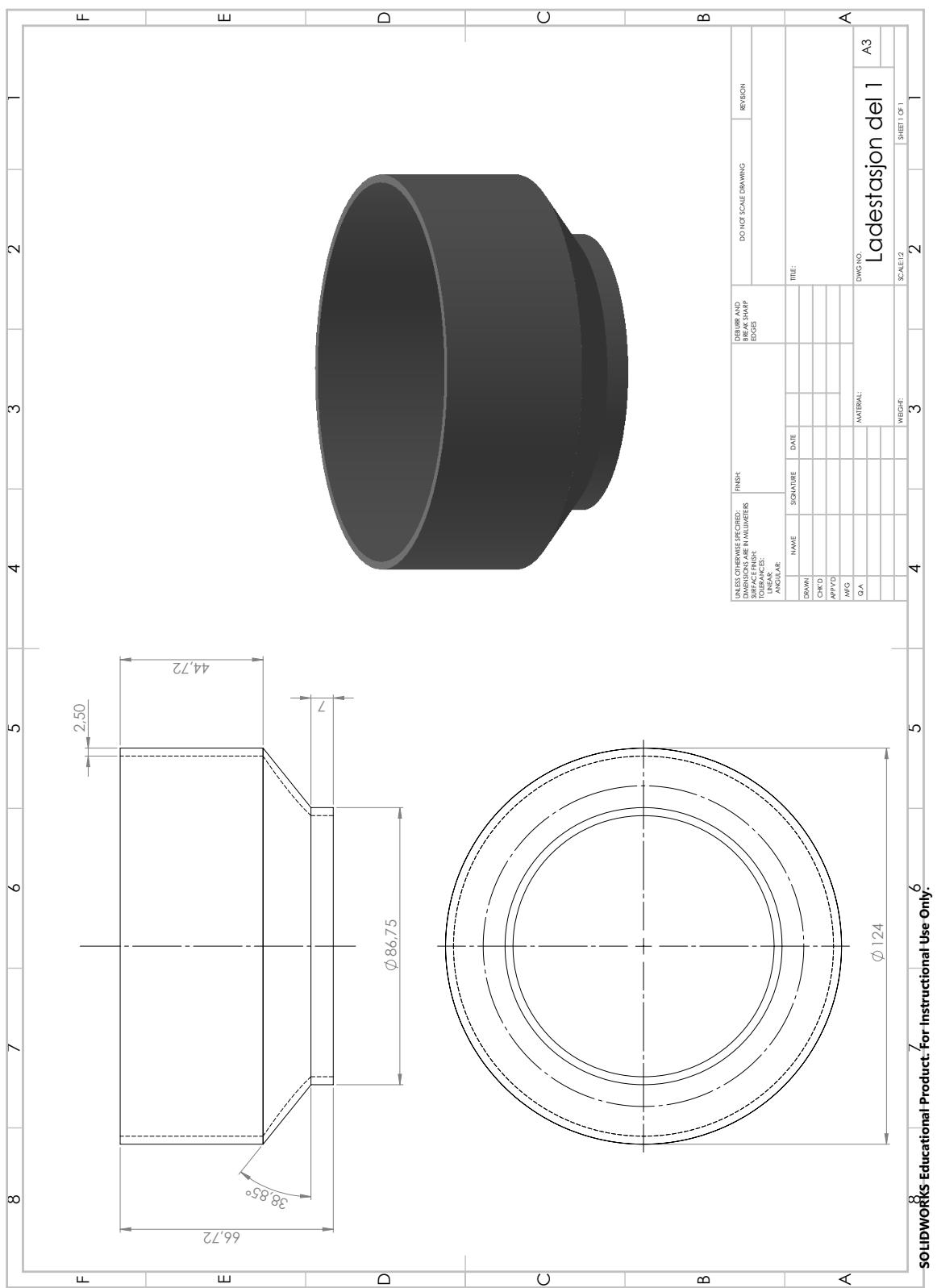
### C.3 Komponentfeste 1



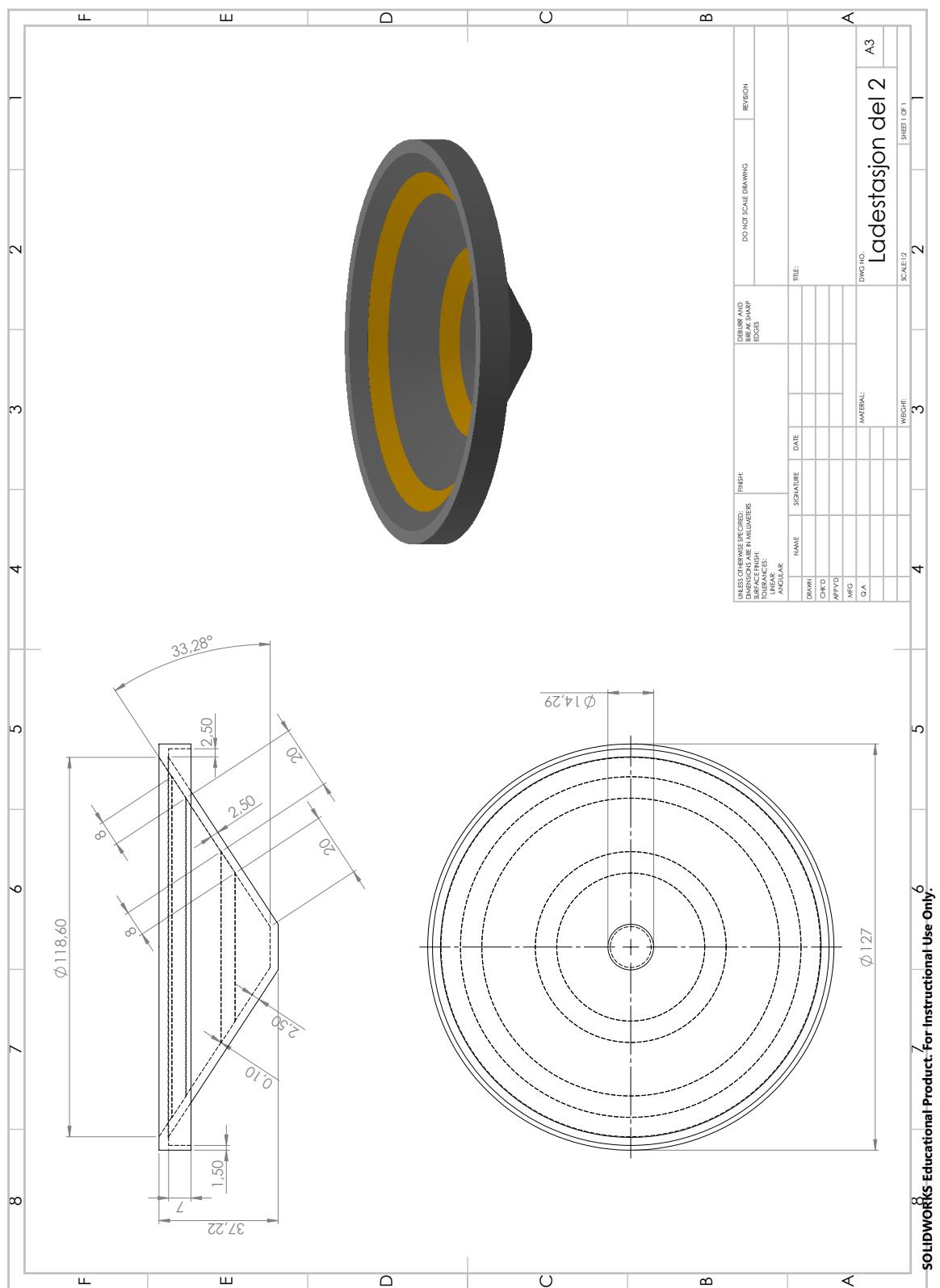
## C.4 Komponentfeste 2



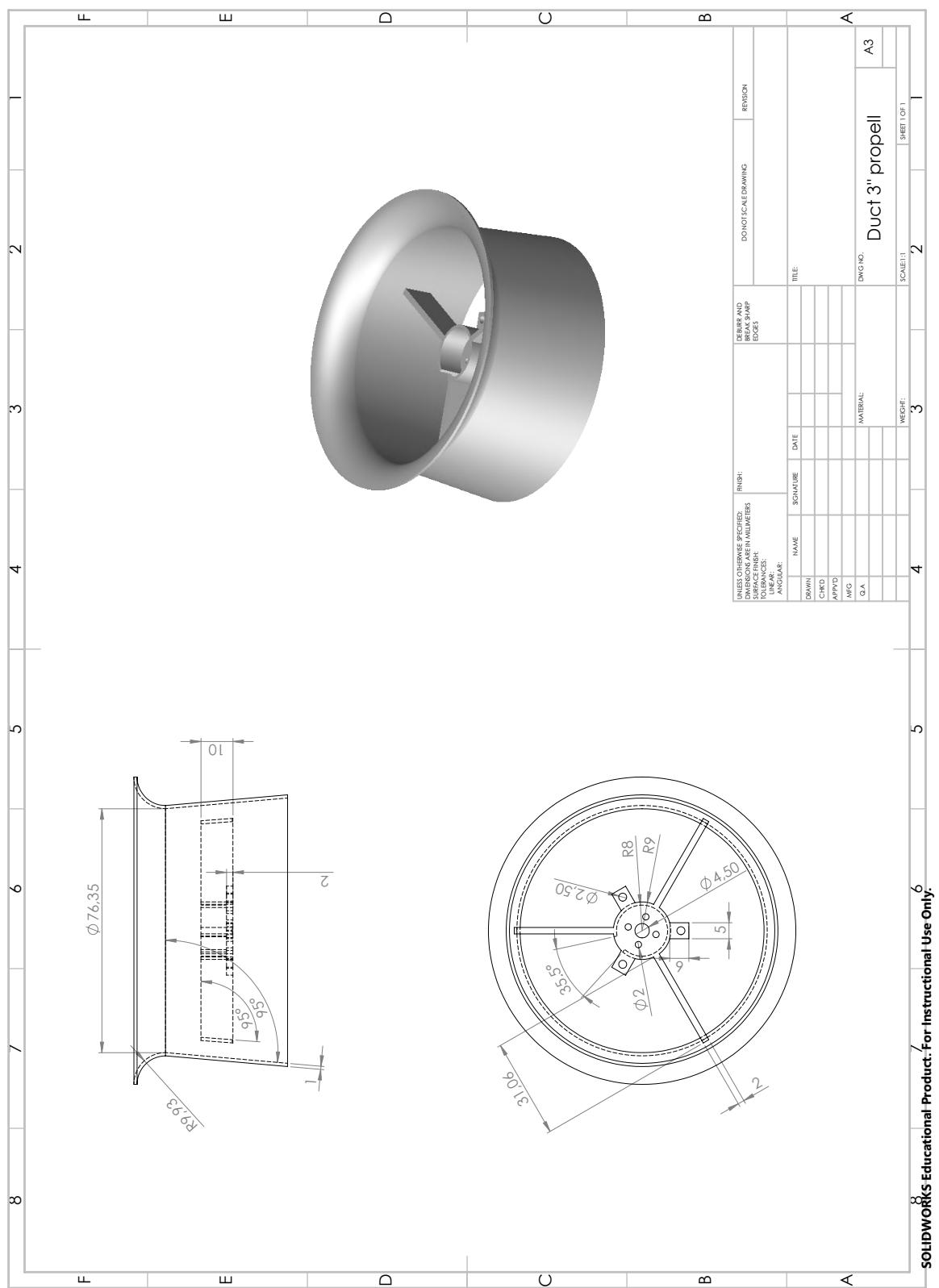
## C.5 Ladestasjon del 1



## C.6 Ladestasjon del 2

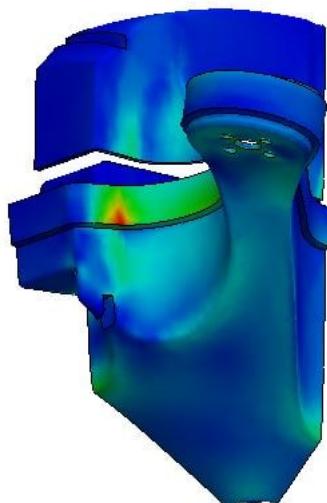


## C.7 Duct 3" (76.2mm)

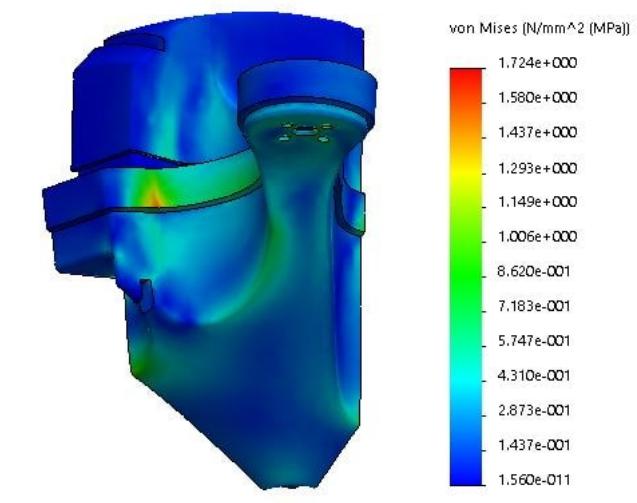


## D FEM-analyse

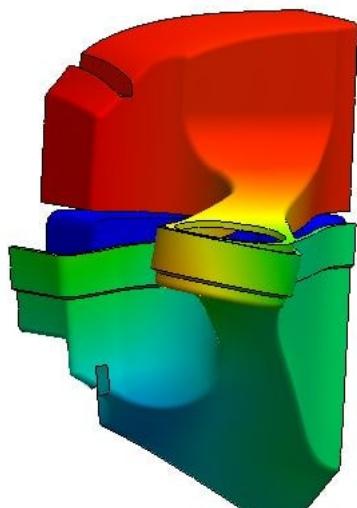
### D.1 Spenninger og forskyvninger som følge av kraft og moment fra motor. Skalering 50:1



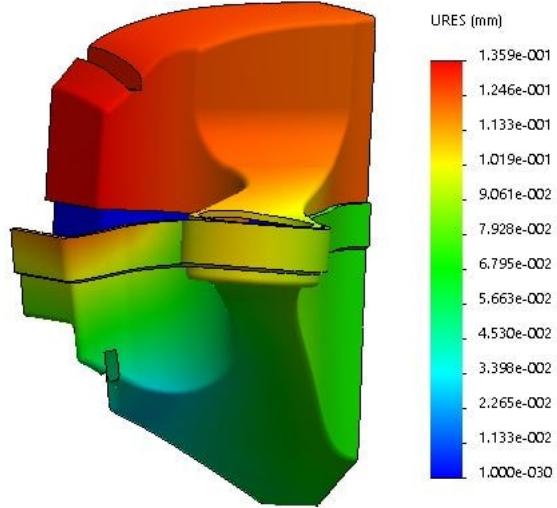
(a) Spenninger CCW



(b) Spenninger CW

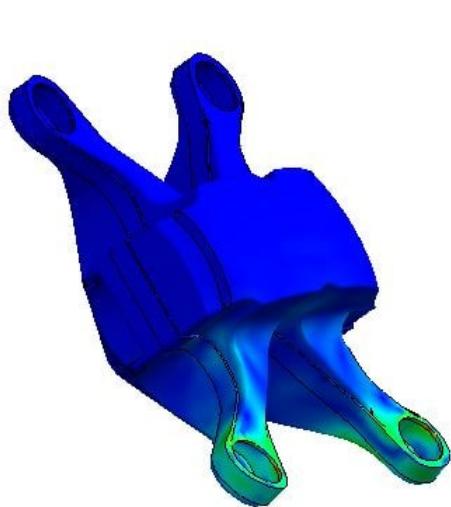


(c) Forskyvninger CCW

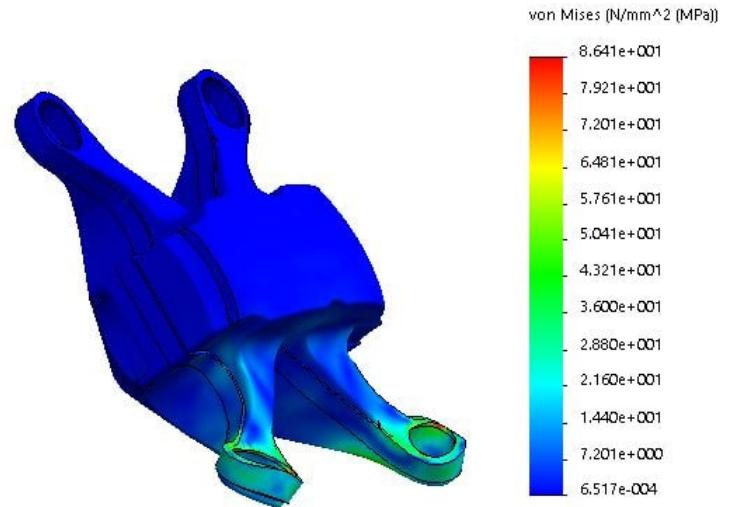


(d) Forskyvninger CW

## D.2 Spenninger fra fall. Skalering 5:1



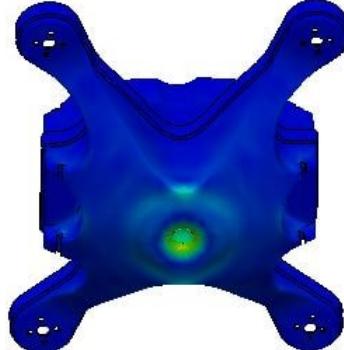
(a) Spenninger ved 1m fall, 45 grader



(b) Spenninger ved 10m fall, 45 grader



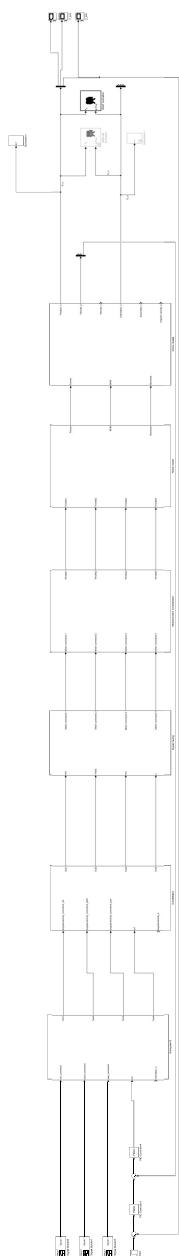
(c) Spenninger ved 1m fall



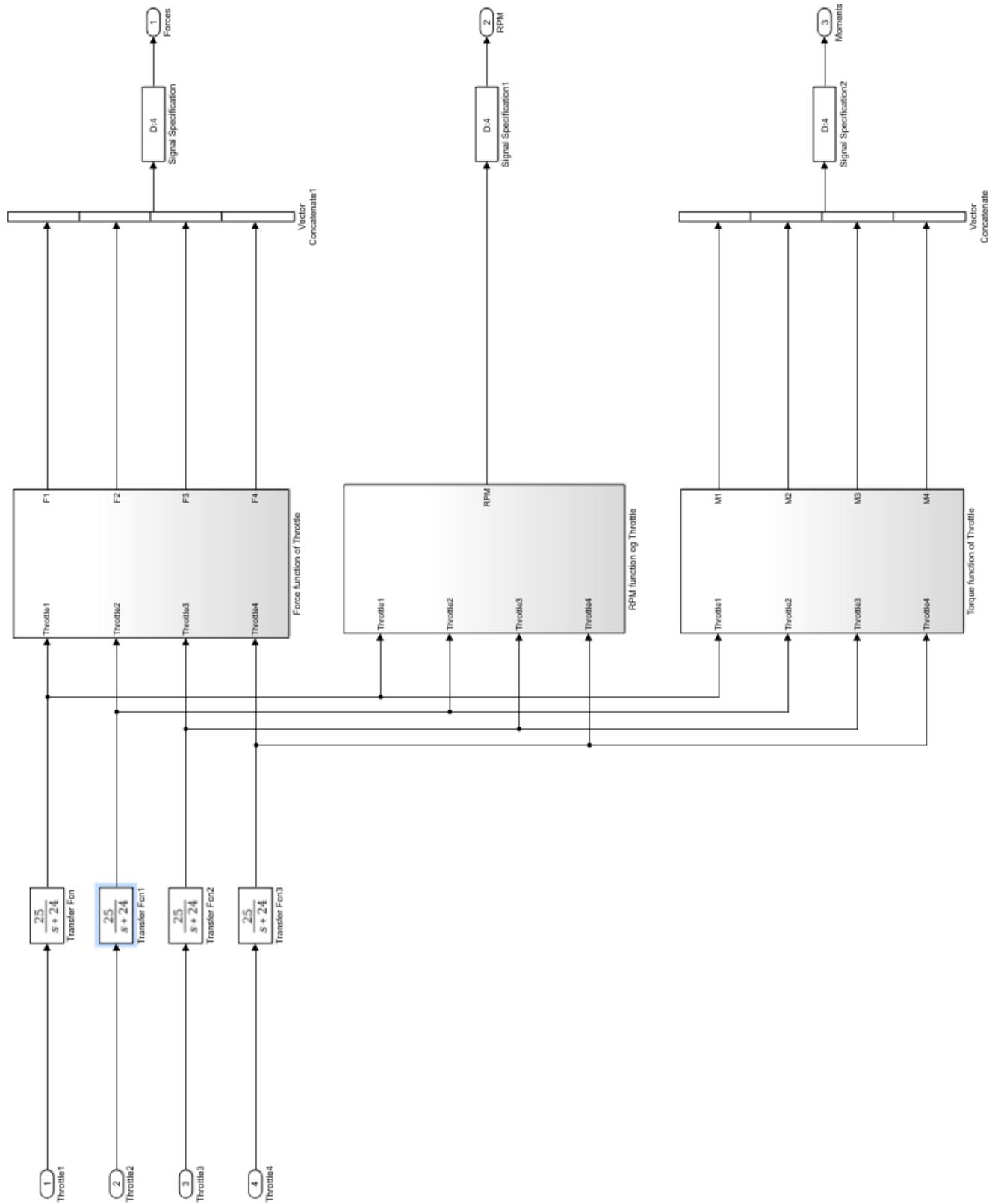
(d) Spenninger ved 10m fall

## E Simulator

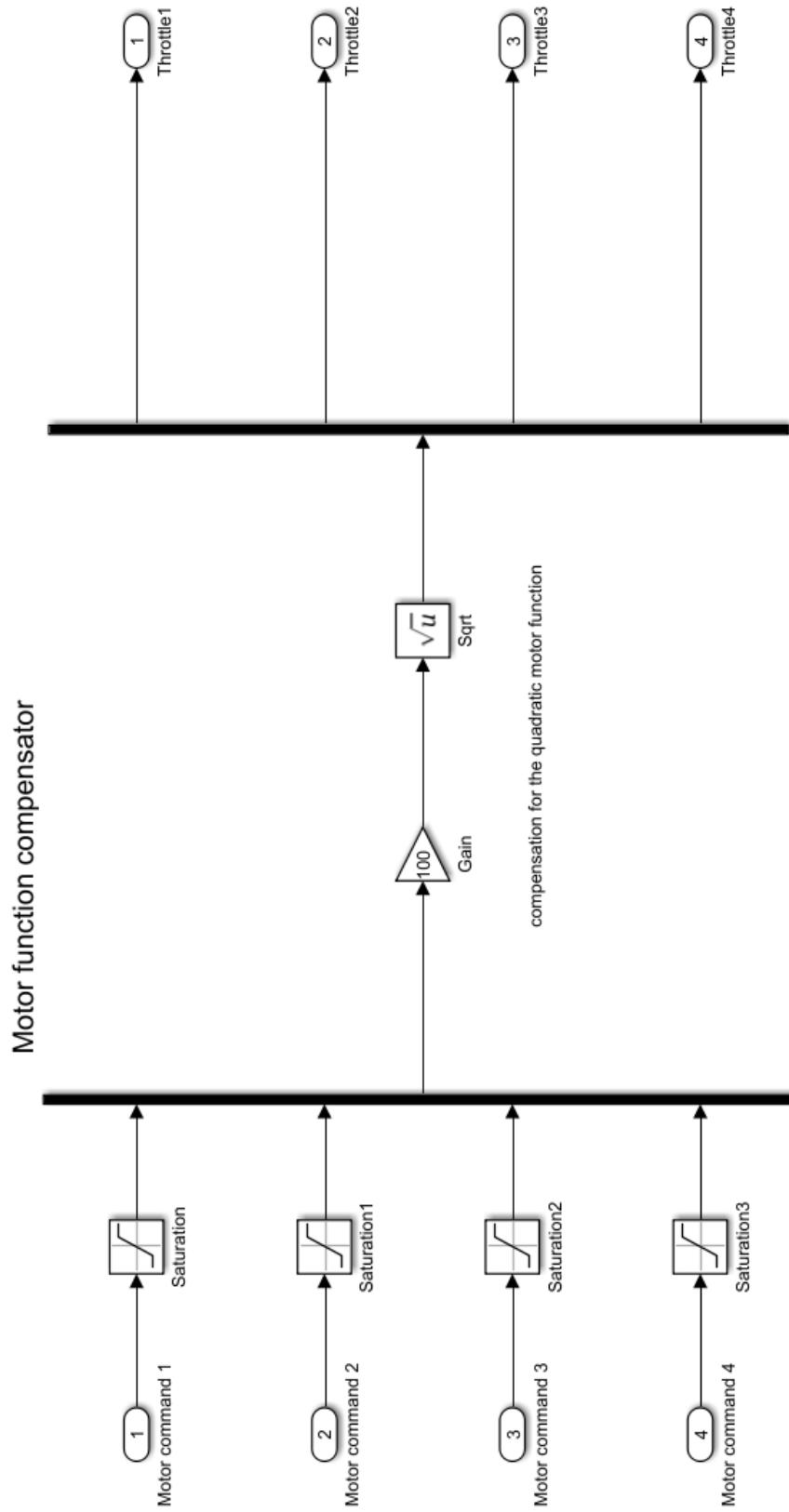
### E.1 Komplett modell av drone



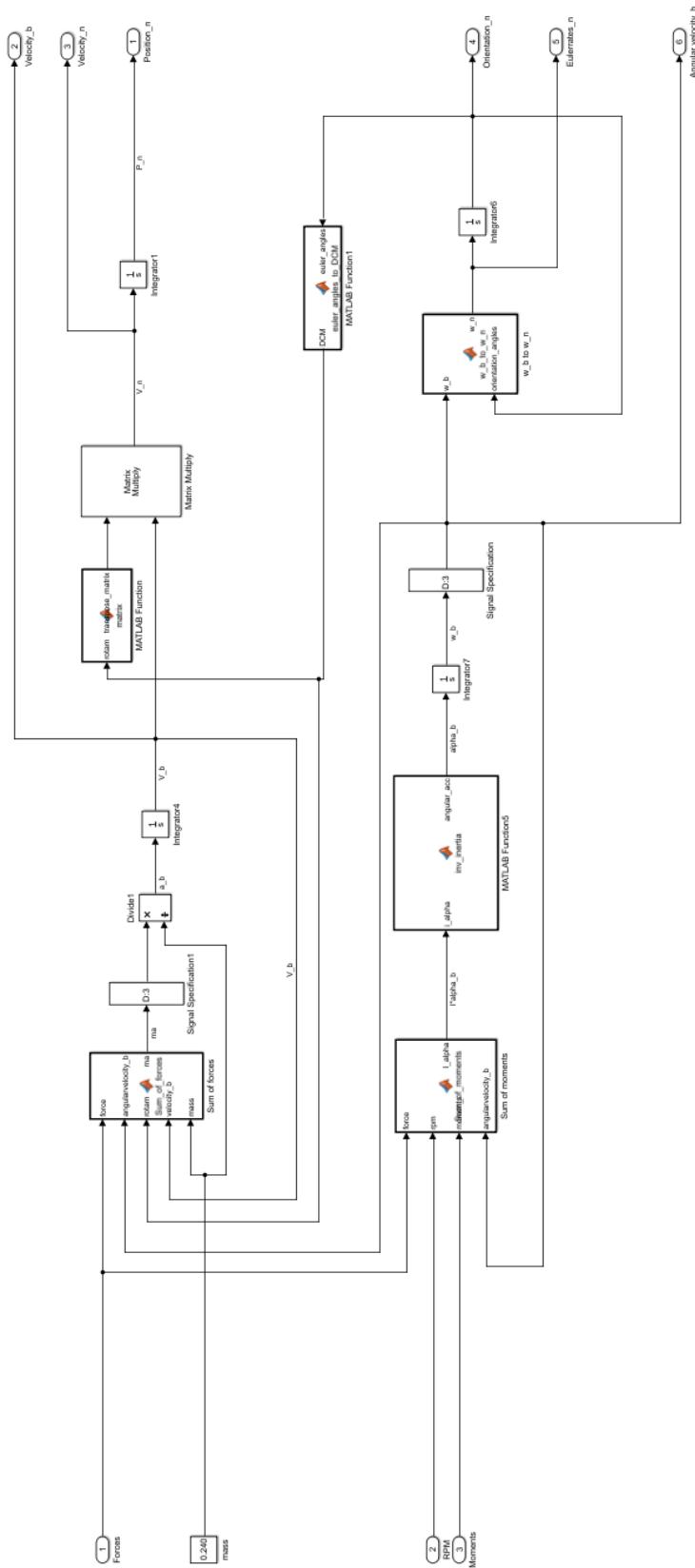
## E.2 Motormodel



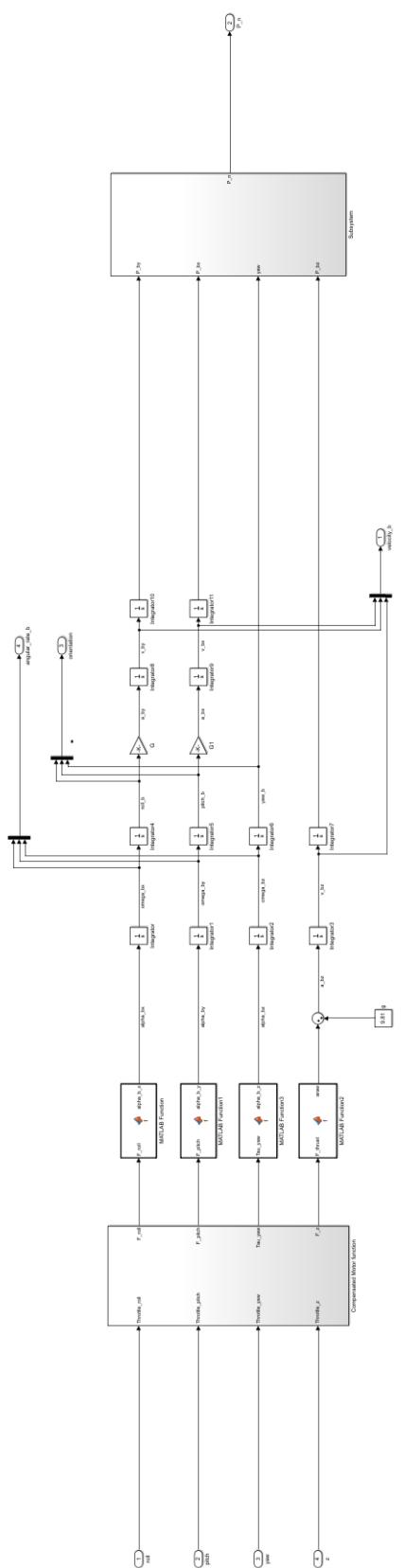
### E.3 Kompensator



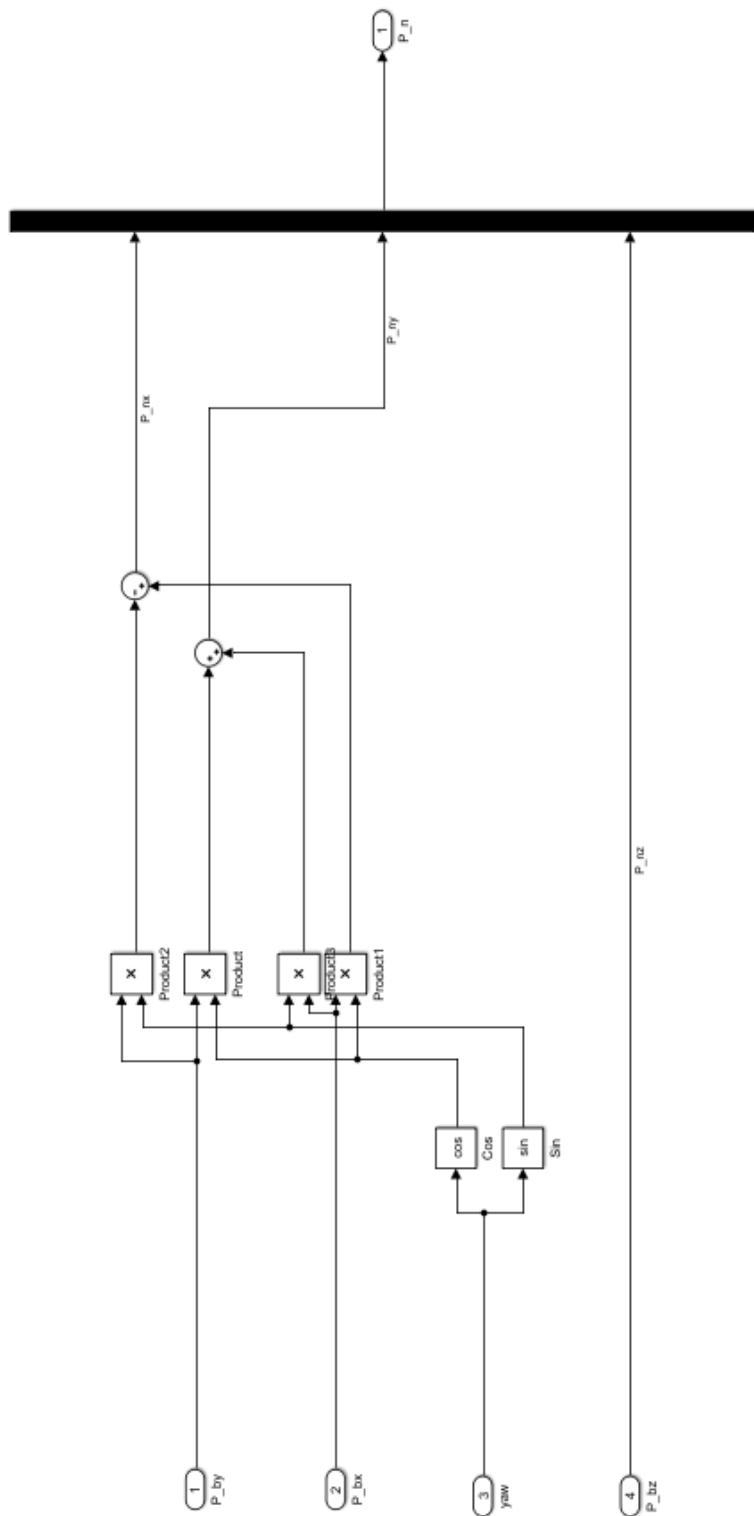
## E.4 Matematisk modell



## E.5 Dekoblet modell



## E.6 Transformasjon forenklet modell



## E.7 Sum av krefter

```
1 function ma = Sum_of_forces( force , angularvelocity_b , rotam , velocity_b , mass )
2
3 %sum of forces
4 %force is a vector of thrustforces
5 %rotam is the rotation matrix
6
7 %p = angularvelocity(1)
8 %q = angularvelocity(2)
9 %r = angularvelocity(3)
10
11 %%forces bodyframe
12 f1_vector = [    0;
13                 0;
14                 -force(1) ];
15
16 f2_vector = [    0;
17                 0;
18                 -force(2) ];
19
20 f3_vector = [    0;
21                 0;
22                 -force(3) ];
23
24 f4_vector = [    0;
25                 0;
26                 -force(4) ];
27
28 gravityForce_n = [    0;
29                     0;
30                     9.81*mass ];
31
32 ma = (f1_vector + f2_vector + f3_vector + f4_vector) + ((rotam) *gravityForce_n)
      - ((cross(angularvelocity_b , velocity_b ))*mass);
```

## E.8 Sum av moment

```
1 function I_alpha = Sum_of_moments(force ,rpm ,moments ,angularvelocity _ b )
2
3 %sum of moments
4 %force is thrustforce-vector
5 %anglesvelocity is a vector describing the rate of rotation
6
7 %% variables
8
9 %distance from cg in bodyframe
10 cg_motor_distace_x = 0.045;
11 cg_motor_distace_y = 0.060;
12 cg_motor_distace_z = 0.015;
13
14 %inertia of drone
15 inertia_xx = 0.00021233;
16 inertia_yy = 0.00021409;
17 inertia_zz = 0.00036986;
18
19 %inertia of motor+prop
20 %2 x prop-inertia
21 motor_inertia = 0.00000035;
22
23
24 %p = angularvelocity (1)
25 %q = angularvelocity (2)
26 %r = angularvelocity (3)
27
28 %% vectors
29
30 tau_moment1 = [ 0;
31                 0;
32                 -moments(1) ];
33
34 tau_moment2 = [ 0;
35                 0;
36                 moments(2) ];
37
38 tau_moment3 = [ 0;
39                 0;
40                 -moments(3) ];
41
42 tau_moment4 = [ 0;
43                 0;
44                 moments(4) ];
45
46 r1_vector = [    cg_motor_distace_x;
47                 -cg_motor_distace_y;
48                 cg_motor_distace_z ];
49
50 r2_vector = [    -cg_motor_distace_x;
51                 -cg_motor_distace_y;
52                 cg_motor_distace_z ];
```

```

54 r3_vector = [ -cg_motor_distace_x;
55                 cg_motor_distace_y;
56                 cg_motor_distace_z ];
57
58 r4_vector = [ cg_motor_distace_x;
59                 cg_motor_distace_y;
60                 cg_motor_distace_z ];
61
62 f1_vector = [ 0;
63                 0;
64                 -force(1) ];
65
66 f2_vector = [ 0;
67                 0;
68                 -force(2) ];
69
70 f3_vector = [ 0;
71                 0;
72                 -force(3) ];
73
74 f4_vector = [ 0;
75                 0;
76                 -force(4) ];
77
78 inertia_matrix = [ inertia_xx      0      0;
79                         0      inertia_yy      0;
80                         0          0      inertia_zz ];
81
82 %angular momentum of motor+prop in bodyframe
83
84 angular_momentum_motor_b = [
85                               0;
86                               0;
87                               motor_inertia*(rpm(1) - rpm(2) + rpm(3) - rpm
88                               (4))*(2*pi)/60) ];
89
90 %% calculations
91
92 I_alpha = cross(r1_vector,f1_vector) + cross(r2_vector,f2_vector)

```