Universitetet i Stavanger

DET TEKNISK-NATURVITENSKAPELIGE FAKULTET

BACHELOROPPGAVE

Studieprogram/spesialisering: Bachelor i ingeniørfag / Automatisering og elektronikkdesign	Vårsemesteret 2022 Åpen
Forfattere: Hanne Lovise Berger Malin Harr Overland	Hanne L. Berger Malin Harr Overland
Fagansvarlig: Morten Tengesdal Veileder(e): Morten Tengesdal	
Tittel på bacheloroppgaven: Utvikling av smart flyter Engelsk tittel: Development of intelligent float	
Studiepoeng: 2x20	
Emneord: AUV, ROV, UiS Subsea, MATE, batteriforsyning, oppdriftsmotor,	Sidetall: 161 + vedlegg/annet: 96 Stavanger 15. mai 2022

Sammendrag

UiS Subsea er en studentorganisasjon, som ble opprettet i 2013. Hvert år, bygges det et nytt undervannsfartøy for å delta i en konkurranse, arrangert av Marine Advanced Technology Education (MATE) i USA. Årets lag, består av 18 studenter fra studieretningene elektro, data og maskin, fordelt på åtte ulike bacheloroppgaver. For å delta i årets MATE-konkurranse, må det bygges to undervannsfartøy, en ROV (Remotely Operated Vehicle) og en AUV (Autonomous Underwater Vehicle).

Denne rapporten tar for seg utviklingen av en smart flyter, altså årets AUV. Denne skal lages for å kunne utføre en spesifikk oppgave i konkurransen, der den blir satt ut av ROV-en før den skal synke ned og flyte opp to ganger. Det er første gang en flyter blir laget i UiS Subsea, og prosjektet er derfor preget av mye planlegging, og vurdering av ulike løsninger.

For å utvikle en smart flyter, har vi designet et kretskort, som fungerer som et grensesnitt mellom alle de ulike delene som inngår i flyteren. Dette gjelder både mikrokontroller, sensorer som er brukt, pumpe, ventil og strømforsyning.

Andre deler av oppgaven har vært å finne aktuelle komponenter, utforme oppsettet inni flyteren og lage maskin- og programvare. Det var også et krav fra konkurransen at flyteren skal være batteridrevet. Det har derfor blitt gjort en batterivurdering, der det har blitt tatt hensyn til kravene om maksimalt strømtrekk og spenning.

Gjennom oppgaven, har vi hatt stort læringsutbytte, ikke bare faglig, men også i forhold til det å jobbe med et stort tverrfaglig prosjekt, hvor man må samarbeide med flere grupper. Siden flyteren er uavhengig av ROV-en, har vi stort sett kun samarbeidet med en av maskingruppene, men det har vært fint å ha hele Subsea-laget å kunne be om hjelp eller støtte.

Flyteren er testet både tørt og i vann, og den oppfyller kravene som ble satt i behovs- og funksjonsspesifikasjonen. Fremover mot konkurransen, skal det testes videre med hvordan ROV-en skal holde flyteren, og hvordan de kommuniserer.

Forord

Det har vært veldig lærerikt og interessant å jobbe med dette prosjektet. Å jobbe med et prosjekt helt fra planleggingsfasen, til et ferdig produkt, har vært veldig spennende, og vi har også fått jobbet med studenter fra andre studieretninger.

Vi vil gjerne takke Jon Fidjeland og Romuald Karnol Bernacki for god hjelp med komponenter og lån av utstyr. Vi vil også takke Kristian Thorsen for langt og utfyllende svar angående spenningsregulatorer.

Sist, men ikke minst, vil vi takke vår veileder, Morten Tengesdal, for gode tilbakemeldinger og veiledning gjennom hele semesteret, og UiS Subsea, for muligheten til å delta i dette prosjektet.

Ordforklaring

ADC	En modul som konverterer et analogt signal til et digitalt signal (**A**nalog-to-**D**igital **C**onverter)
API	Programmeringsgrensesnitt (**A**pplication **P**rogramming **I**nterface)
AUV	Autonom undervannsfarkost (**A**utonomous **U**nderwater **V**ehicle)
CSB	**C**hip **S**elect **B**it
EMI	Elektromagnetisk forstyrrelse (**E**lectro **M**agnetic **I**nterference)
ESD	Elektrostatisk utladning (**E**lectro **S**tatic **D**ischarge)
FCCM	**F**orced **C**ontinous **C**onduction **M**ode
HAL	Et programmerings-"lag" med funksjoner som utfører direkte operasjoner på mikrokontrolleren (**H**ardware **A**bstraction **L**ayer)
HSE	**H**igh **S**peed **E**xternal
HSI	**H**igh **S**peed **I**nternal
I^2C	En seriell databuss som brukes til kommunikasjon mellom enheter (**I**nter-**I**ntegrated **C**ircuit)
KVL	Regel om spenning i en krets (**K**irchhoff's **V**oltage **L**aw)
LSb	**L**east **S**ignificant bit
LSE	**L**ow **S**peed **E**xternal
MATE	**M**arine **A**dvanced **T**echnology **E**ducation
Mils	Måleenhet. 1 mil = $1{\cdot}10^{-3}$ tommer
MSb	**M**ost **S**ignificant bit
PCB	**P**rinted **C**ircuit **B**oard
PLLCLK	**P**hase-**L**ocked **L**oop Clock
PROM	**P**rogrammable **R**ead-**O**nly **M**emory
PSM	**P**ulse-**S**kip **M**ode
PWM	**P**ulse **W**idth **M**odulation
ROV	Fjernstyrt undervannsfarkost (**R**emotely **O**perated **V**ehicle)
SCL	**S**erial **C**lock
SDA	**S**erial **D**ata
SSR	Et rele som slås på eller av når det påføres en ekstern spenning (**S**olid **S**tate **R**elay)
SWD	Programmeringsprotokoll (**S**erial **W**ire **D**ebug)
SYSCLK	**S**ystem **C**lock
TVS	En transient-undertrykkelsesdiode (**T**ransient-**V**oltage-**S**uppression-**D**iode)
USM	**U**ltrasonic **M**ode

Innhold

1 Innledning

Innhold

Denne bacheloroppgaven er en del av et større prosjekt, som baserer seg på design, utvikling og bygging av undervannsfarkoster. Utgangspunktet for prosjektet er deltakelse i den internasjonale konkurransen, *MATE ROV Competition* [19]. Kapittelet vil presentere organisasjonen UiS Subsea, hvilke typer undervannsfarkoster som skal utvikles og oppgavene de designes etter. Deretter vil det overordnede systemet presenteres med inndeling av ansvarsområder for de ulike bachelorgruppene. Avslutningsvis gis et mer detaljert innblikk i systemet denne bacheloroppgaven omhandler.

Introduksjonskapittelet er skrevet av gruppen med ansvar for styrings- og reguleringssystemet, og er delt med alle gruppene i prosjektet. Kapittelet er felles for alle gruppene ettersom det presenterer prosjektet i sin helhet.

1.1 UiS Subsea

UiS Subsea er en studentorganisasjon ved Universitetet i Stavanger, som siden 2013 har engasjert studenter i undervannsteknologi. UiS Subsea har som mål å gi studentene erfaring med å jobbe i et større prosjekt på tvers av fagområder.

Organisasjonen har gjennom flere år bygget ROV-er for å delta i MATE-konkurransen. Dette gir grunnlag for et prosjekt som krever at de involverte løser utfordrende tverrfaglige oppgaver. Formålet med dette er å skape et miljø hvor studentene får utviklet sine tekniske ferdigheter, samt evne til å samarbeide med andre studenter. UiS Subsea er en organisasjon som muliggjør et tett samarbeid mellom studenter og næringslivet. En rekke bedrifter er involvert og svært interessert i dette prosjektet, og bidrar med utstyr og andre ressurser gjennom sponsoravtaler. For å dyrke dette samarbeidet, vil det i 2022, for første gang, arrangeres *Subsea-dagen*. Her vil næringslivet inviteres til universitetet for å presentere sine bedrifter.

I tidligere år har det manglet kontinuitet i organisasjonen, da den hvert år har blitt overtatt av et helt nytt kull med bachelorstudenter. I fjor ble organisasjonen restrukturert ved at styret ble adskilt fra selve prosjektet, med formål om å muliggjøre drift uavhengig av prosjektdeltakere. Dette åpner også mer opp for deltakelse fra studenter i andre perioder av utdanningsløpet. Fjorårets leder og nestleder har valgt å sitte en periode til, for å redusere belastningen på bachelorstudentene, og for å videreføre tidligere erfaringer.

Nytt for årets prosjekt, er at også en bachelorgruppe av økonomistudenter innehar en rolle i driften av organisasjonen. Dette bidrar med kunnskap innenfor økonomi og markedsføring i styret.

Årets styre består av følgende roller med tilhørende innehavere:

- **Styreleder:** Daniel Vasshus
- **Nestleder:** Geir Arne Solland Kindingstad
- **Økonomiansvarlig:** Sina Brunes
- **Markedsføring:** Sanna Sørskår og Åse Jortveit Sagebakken
- **Sponsoransvarlig:** Maren Lovise Jåsund og Otto Nessa Ljosdal
- **Styremedlem:** Tage Mellemstrand

Grunnet restruktureringen, er det opprettet en egen prosjektledelse med ansvar for det tekniske. Prosjektledelsen fungerer som et bindeledd mellom organisasjonen og prosjektdeltakerene.

Prosjektledelsen med roller og innehavere er:

- **Prosjektleder:** Tomas Royal Choat
- **Teknisk ansvarlig programvare:** Christoffer Næss
- **Teknisk ansvarlig maskinvare:** Mats Røste

1.2 Farkostene

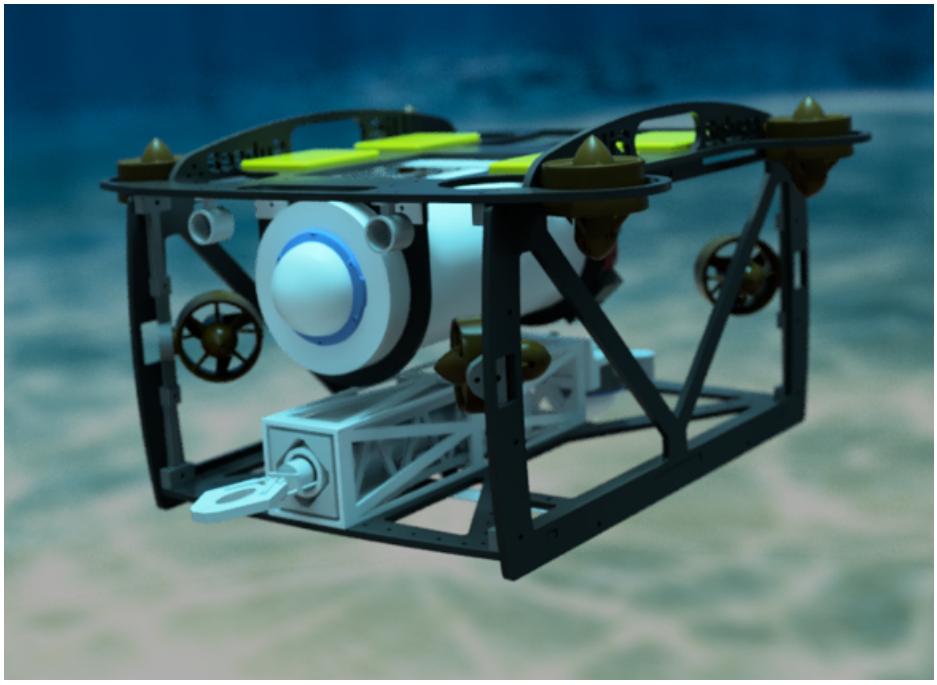
1.2.1 ROV

En ***R****emotely* ***O****perated Underwater* ***V****ehicle* (ROV) er, som navnet indikerer, en fjernstyrt undervannsfarkost. ROV-en er et hjelpemiddel som muliggjør undervannsoperasjoner uten behov for dykkere. Utviklingen av denne typen undervannsroboter har åpnet dørene for en rekke operasjoner under vann som ikke var mulig før, da man ikke lenger trenger å risikere menneskeliv. Dette utvider rekkevidden under vann, og forenkler operasjonene, siden de kan gjennomføres i mindre skala. Nyere ROV-er er også mulig å operere fra land, som reduserer kostnader av å utføre operasjoner offshore.

Utviklingen av ROV-er har på grunn av offshore-næringen vokst kraftig siden midten av 1980-tallet [46], da dette medførte utplassering av store installasjoner og behov for operasjoner på dybder man ikke kunne nå med dykkere. Eksempler på dette fremheves av årets konkurranse, med oppgaver som inspeksjon av fiskemerder og havvindmøller.

1.2.2 Fenris

ROV-en som utvikles i år, *Fenris*, er en frittsvømmende ROV med navlestreng, vist i figur 1.

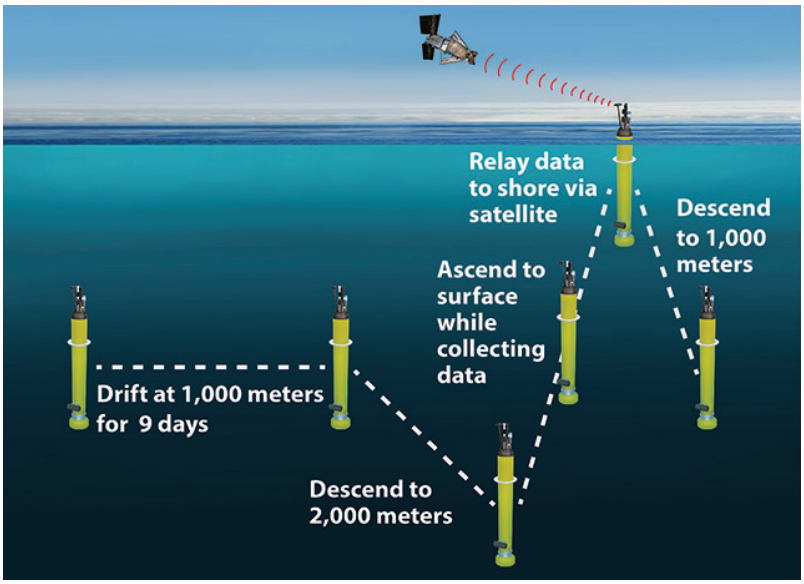


Figur 1: 3D-modell av Fenris.

Fenris styres fra et operatørgrensesnitt på land, og kommuniserer gjennom navlestrengen. Navlestrengen utgjør også kraftforsyningen til ROV-en. Målet er at ROV-en skal være i stand til å nå en dybde på 50 meter. Fenris er designet etter oppgavene i *MATE ROV Competition*, og er derfor utstyrt med en manipulatorarm for å løse disse. For å tilrettelegge for videreutvikling på ROV-en, er den designet modulært, slik at alt enkelt kan byttes ut. Dette gjør at Fenris kan brukes som en base for eventuell fremtidig utvikling.

1.2.3 Flyter

Nytt for årets prosjekt er at det skal bygges en *flyter*, inspirert av *float*-teknologi utviklet av forskningsprosjektet GO-BGC [11]. En flyter er en autonom undervannsfarkost som har i hovedoppgave å overvåke områder under vann. For å utføre dette, er den utstyrt med biologiske og kjemiske sensorer. Den har også en innebygd mekanisme som endrer volumet til farkosten, og dermed kan oppdriften reguleres. Et eksempel på syklusen til en flyter er vist i figur 2.

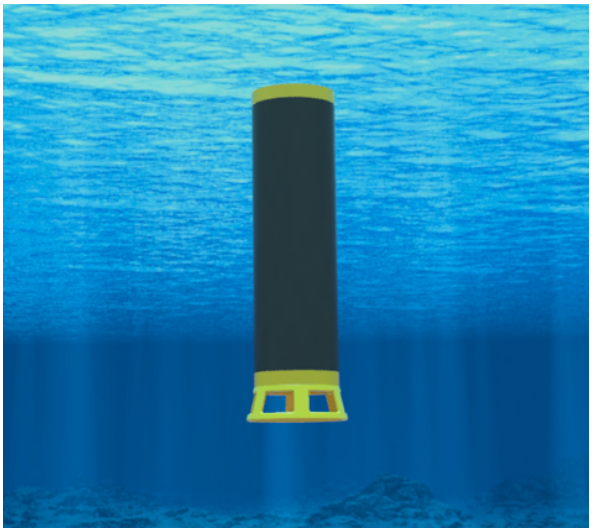


Figur 2: Eksempel på syklusen til en flyter. Hentet fra [23].

Flyteren kan dykke ned til flere tusen meters dybde, og returnere til overflaten på egenhånd. Når flyteren når overflaten, sendes innhentet data via satellitt, og slik skal den kunne fortsette ubemannet i opptil 5 år.

1.2.4 Frøya

Flyteren som utvikles i år, *Frøya*, er vist i figur 3.



Figur 3: 3D-modell av Frøya.

Dette er en prototype av en flyter, hvor fokuset er rettet mot den dynamiske oppdriftsmekanismen, og den er ikke utstyrt med biologiske eller kjemiske sensorer. Dybden den befinner seg på måles med en trykksensor, og reguleres gjennom en oppdriftsmotor. Oppdriftsmotoren er realisert ved hjelp av en pumpe som forflytter luft ut i en ballong på undersiden av flyteren. Av sikkerhetsmessige årsaker, må batteriene i flyteren være alkaliske. Dette begrenser hvor lenge den kan operere. Målet er at flyteren skal være i stand til å gjennomføre 2 vertikale profiler innenfor en tidsramme, etter at den er blitt plassert i vannet av ROV-en. I konkurransen tilsvarer en vertikal profil et dykk fra overflaten, ned til bunnen og tilbake.

1.3 MATE – Marine Advanced Tehchnology Education

Informasjonen under er hentet fra arrangørens hjemmeside [18].

Prosjektet gjennomføres, som tidligere nevnt, med hovedfokus rettet mot en internasjonal konkurranse innen utvikling av robotiserte undervannsfarkoster. Arrangøren bak konkurransen er forskningssenteret *The Marine Advanced Tehchnology Education (MATE) Center*, som er et samarbeid mellom flere organisasjoner i USA, med et felles mål om å forbedre utdanning innen marin teknikk. Senterets logo er vist i figur 4. Forskningssenteret jobber i tett samarbeid med flere skoler og universiteter, men også med bedrifter. Gjennom dette samarbeidet kan de tilby utdanning av studenter, basert på etablerte retningslinjer fra bedriftene, og sørge for direkte kontakt mellom student og arbeidsgiver under utdanningen. Et av hovedformålene med The MATE Center er å gi studenter et læringsutbytte, som senere kan være med på å utvikle hele næringen. I den sammenheng arrangeres konkurransen *MATE ROV Competition* hvert år. UiS Subsea stiller i *utforskerklassen*, som er klassen med høyest vanskelighetsgrad.

Figur 4: The MATE Center sin logo. Hentet fra hjemmesiden deres.

MATE ROV Competition

Informasjon om konkurransen er hentet fra konkurransemanualen [19]. Konkurransen har også sin egen logo, vist i figuren under.

Figur 5: *MATE ROV Competition* logo. Hentet fra konkurransens nettside.

Årets utgave av konkurransen vender oppmerksomheten mot FNs havforskningstiår (2021-2030), som er et initiativ fra FN med bakgrunn i de 17 bærekraftsmålene. Motivasjonen bak MATEs engasjement er å snu den nedadgående trenden til havets helse, og i den forbindelse vil de utfordre studenter globalt til å komme med nyskapende løsninger på problemet. Oppgaven går ut på å bygge en ROV som bidrar i arbeidet mot klimaendringer, legger til rette for ren energi, sørger for mat til en voksende populasjon, overvåker havets helse og bevarer maritim historie. MATE konstaterer at gjennom dette arbeid vil man skape "havet vi trenger, for fremtiden vi ønsker".

Poenggivning

Under konkurransen deles det ut poeng i tre hovedkategorier. I del 1 gjennomgår man en produktdemonstrasjon. Her kontrolleres vekt og fysiske mål. I denne delen skal også ROV-en og flyteren gjennom de praktiske oppgavene som tester deres operative egenskaper. I konkurransen er det tre separate oppgaver man skal løse, hvor hver oppgave har en tidsramme på 15 minutter. I tillegg belønnes struktur og effektivitet under demonstrasjonen. Man får 1 poeng ekstra for hvert minutt, og 0.01 poeng for hvert sekund, man fullfører før den avsatte tiden i hver oppgave. I del 2 får man poeng for teknisk dokumentasjon av farkostene, samt kommunikasjon og markedsføring fra organisasjonens (UiS Subsea) side. I del 3 får man poeng for å ta hensyn til sikkerhet. Dette innebærer skriftlig sikkerhetsdokumentasjon, men også det å vise aktsomhet under operasjoner med ROV-en.

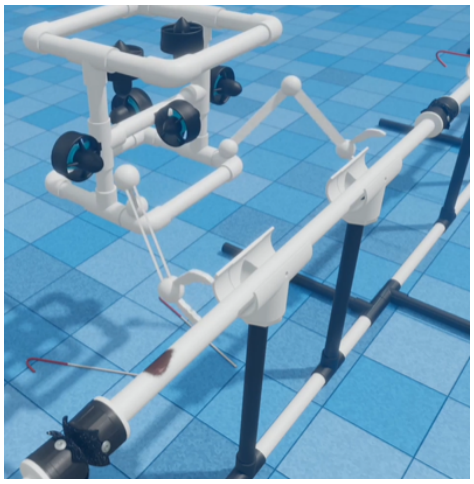
Under gjennomgås de praktiske oppgavene, med tilhørende poengfordeling. Oppgaver, poengfordeling og restriksjoner er gjengitt fra konkurransemanualen [20].

Oppgave 1: Maritim fornybar energi
FNs bærekraftsmål:
#7 Ren energi for alle
#12 Ansvarlig forbruk og produksjon

Oppgaven er utformet for å representere arbeidsoppgaver som er nødvendige for å utføre vedlikehold på offshore havvind. Man skal blant annet reparere en skadet strømkabel, erstatte en ødelagt oppdriftsmodul og fjerne et fiskegarn som har satt seg fast på en av vindturbinenes understell. Utover dette skal det også plasseres ut en hydrofon for å overvåke tilstedeværelsen av sjøpattedyr. Til slutt skal ROV-en kjøre autonomt inn i en dokking-stasjon. Alle arbeidsoppgavene er simulert ved hjelp av konstruksjoner laget av PVC-rør. Under er oppgavebeskrivelsene gjengitt, med tilhørende poengfordeling.

1.1 Erstatte en skadet del av en strømkabel

Figur 6 viser hvordan deler av oppgaven kan løses.

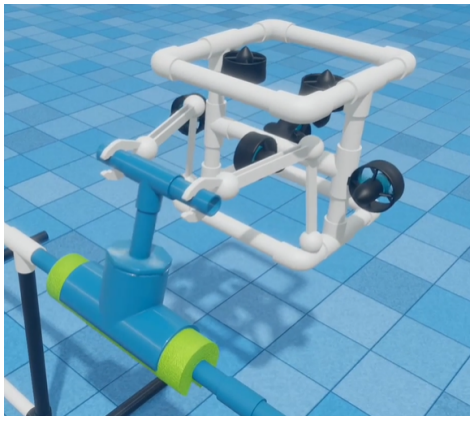


Figur 6: Fjerning av skadet del av kabel. Skaden er markert med en brun flekk. Låsepinnen vises rett til venstre for den brune flekken på røret. Hentet fra [22].

- Utføre en visuell inspeksjon av kabelen: **5 poeng**.
- Kutte kabelen på begge sider av den skadede delen. Simulert ved å dra ut en låsepinne på hver side: **10 poeng**.
- Fjerne den skadede delen av kabelen (PVC-rør), og bringe den til overflaten: **5 poeng**.
- Installere en ny kabelseksjon (PVC-rør), og feste denne med tilhørende festemekanisme: **5 poeng** for hver ende av røret, totalt **10 poeng**.

1.2 Erstatte en skadet oppdriftsmodul på en strømkabel tilhørende en havvindmølle

Figur 7 viser hvordan deler av oppgaven kan løses.

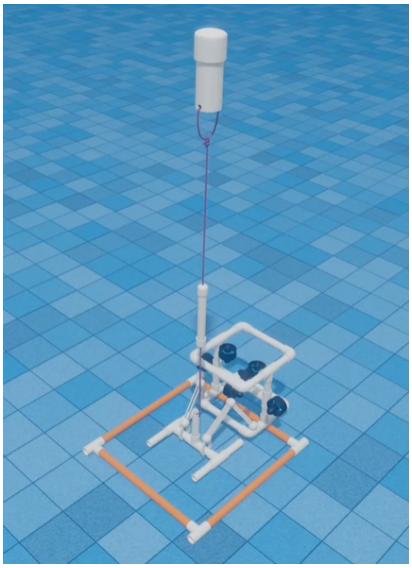


Figur 7: Fjerning av skadet oppdriftsmodul. Den nye oppdriftsmodulen er lik den som fjernes, men har borrelås som lar den festes på røret. Hentet fra [22].

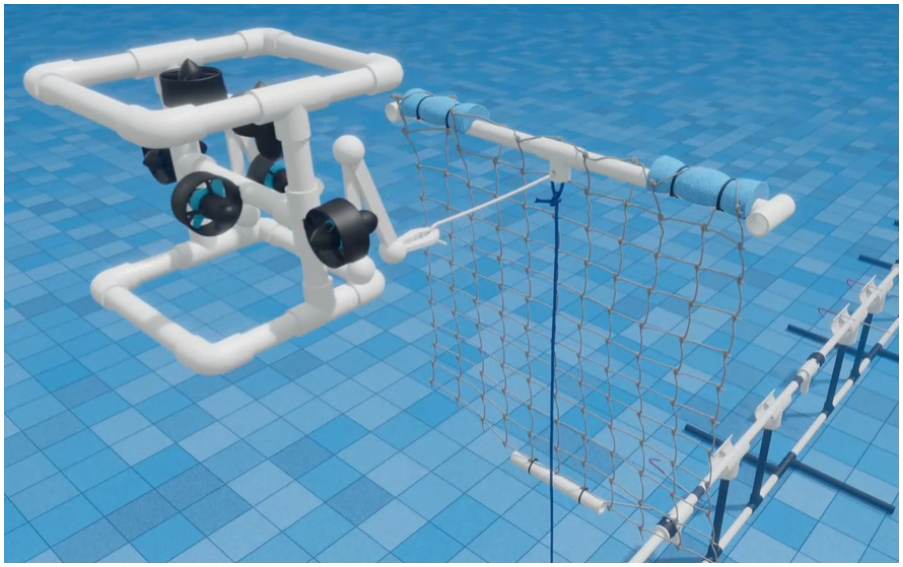
- Fjerne den skadede oppdriftsmodulen ved å rotere festet 180 grader: **5 poeng**.
- Frakte den skadede oppdriftsmodulen til kanten av bassenget, slik at den kan plukkes opp fra land: **5 poeng**.
- Plassere en ny oppdriftsmodul på kabelen: **5 poeng**.
- Feste oppdriftsmodulen ved hjelp av borrelås: **5 poeng**.

1.3 Overvåke miljøet

Figur 8a viser hvordan første del av oppgaven kan løses. Figur 8b viser hvordan andre del av oppgaven kan løses.



(a) Utplassering av hydrofon. Området på 40 cm × 40 cm er representert av de oransje PVC-rørene. Hentet fra [22].



(b) Fjerning av fiskegarn. Garnet vil flyte opp til vannoverflaten når låsespinnen dras ut. Hentet fra [22].

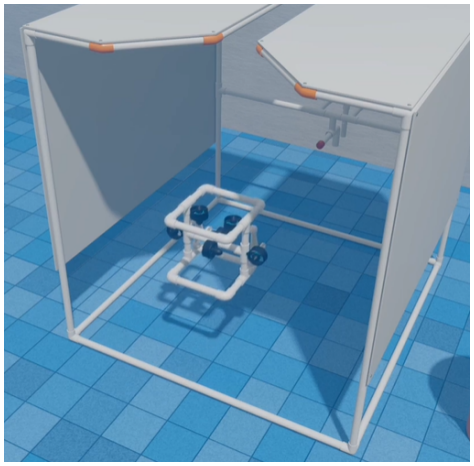
Figur 8

- Utplassere en hydrofon, simulert av et PVC-rør, for å oppdage og registrere tilstedeværelsen av sjøpattedyr.
  - Plassere hydrofonen i et angitt område på 40 $cm \times 40\ cm$: **5 poeng**.
  - Plukke opp hydrofonen etter 5 minutter i vannet og frakte den til bassengkanten: **5 poeng**.
- Fjerne et fiskegarn som er fanget på vindturbinens understell.
  - Dra ut en låsepinne som holder en flytende ramme med fiskegarn under vann: **10 poeng**.

– Fjerne fiskegarnet fra vannet ved å frakte det til bassengkanten, og plukke det opp fra land: **10 poeng**.

1.4 Styre ROV-en inn i en dokking-stasjon

Figur 9 viser hvordan oppgaven kan løses.



Figur 9: Parkering av ROV i dokking-stasjon. Hentet fra [22].

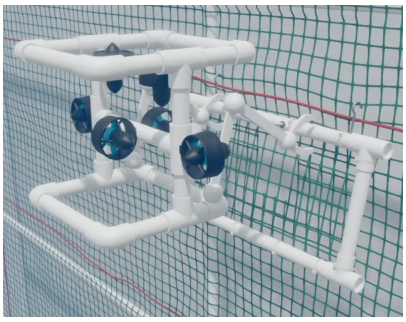
- Styre ROV-en autonomt inn i dokking-stasjonen: **15 poeng**.
- Styre ROV-en manuelt inn i dokking-stasjonen: **5 poeng**.

Oppgave 2: Offshore akvakultur og blå karbon
FNs bærekraftsmål:
#2 Utrydde sult
#13 Stoppe klimaendringene
#14 Liv under vann

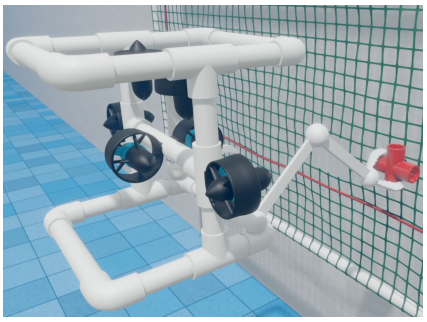
Oppgaven er utformet for å representere arbeidsoppgaver som er nødvendige for å utføre vedlikehold på en fiskemerd, og å bidra til en sunn havhelse. Man skal blant annet reparere en skadet del av en fiskemerd, og fjerne uønsket algevekst. Oppgaven innebærer også å autonomt skille døde fisk fra levende, og å kunne måle størrelsen på fisken. Utover dette skal dødt sjøgress fjernes, og det skal videre plantes nytt. Alle arbeidsoppgavene er simulert ved hjelp av konstruksjoner laget av PVC-rør. Under er oppgavebeskrivelsene gjengitt, med tilhørende poengfordeling.

2.1 Inspisere en offshore akvakultur fiskemerd

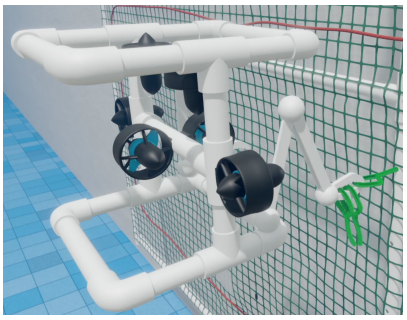
Figur 10 viser hvordan noen av deloppgavene kan løses.



(a) Reparasjon av skadet del av nettet.



(b) Fjerne innkapslende marin vekst.



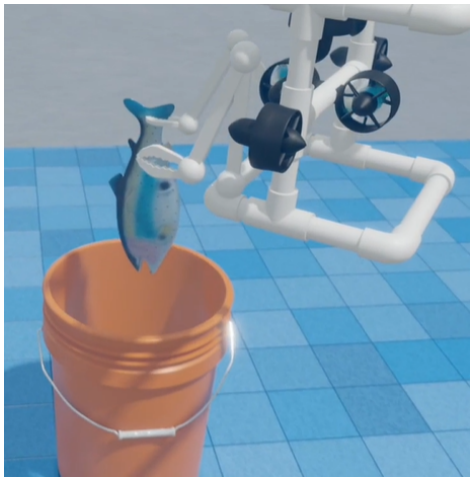
(c) Fjerne algevekst.

Figur 10: Alle hentet fra [22].

- Inspisere nettet for å identifisere skadede områder.
  - Kjøre en transektlinje for å identifisere skadede områder: Autonomt: **25 poeng**. Manuelt: **10 poeng**.
  - Identifisere og telle skadede områder av nettet: **5 poeng**.
- Reparere en skadet del av nettet: **10 poeng**.
- Fjerne marin vekst.
  - Fjerne innkapslende marin vekst, simulert med et PVC-kryss: **5 poeng**.
  - Fjerne algevekst, simulert med 3 piperensere: **5 poeng**.

2.2 Opprettholde et sunt miljø

Oppgaven som innebærer oppsamling av død fisk er vist i figur 11.



Figur 11: Plassering av død fisk i oppsamlingsrør. Hentet fra [22].

- Håndtere fiskedød ved å fjerne død fisk, simulert av en gummifisk, fra bunnen av bassenget.
  - Lage et program som kan skille døde fra levende fisk. Man får utdelt en video, og programmet skal merke døde fisk med røde rammer: **10 poeng**.
  - Plukke opp en død fisk: **5 poeng**.
  - Legge fisken i et oppsamlingsrør, simulert av en bøtte: **5 poeng**.

2.3 Måle fiskestørrelser

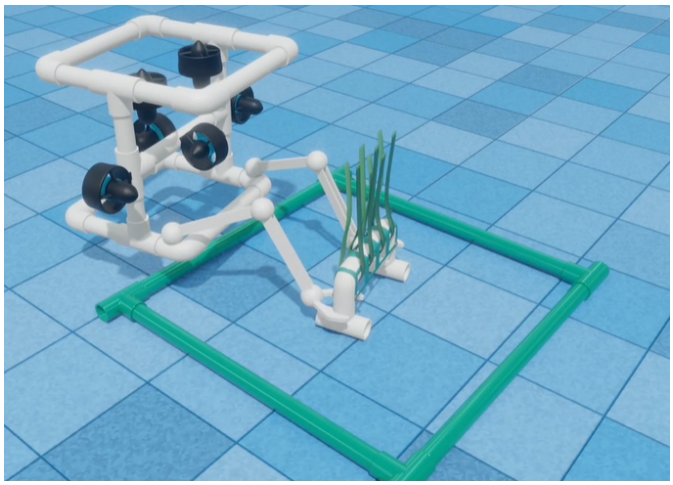
Et eksempel på hvordan man kan løse denne oppgaven er vist i figur 12.

Figur 12: Måling av fiskelengde. Hentet fra [22].

- Beregne gjennomsnittsstørrelsen på en fiskekohort (tre gummifisker av ulik lengde) med maksimalt 2 cm feilmargin: **15 poeng**.
- Beregne biomassen av fiskekohorten: **5 poeng**.

2.4 Dyrke sjøgress

Planting av nytt sjøgress er vist i figur 13.



Figur 13: Planting av nytt sjøgress. Sjøgresset som skal lukes er simulert av en lignende konstruksjon, men denne skal fjernes fra det markerte området. Hentet fra [22].

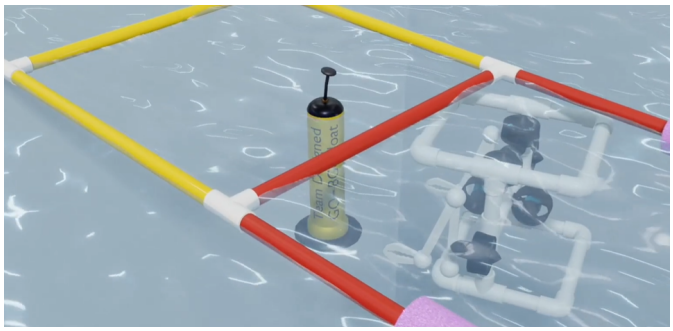
- Luke et eksisterende sjøgressbed: **5 poeng**.
- Plante et nytt sjøgressbed: **5 poeng**.

Oppgave 3: Da og nå – *Endurance22* og *MATE Floats*!
FNs bærekraftsmål:
#13 Stoppe klimaendringene

Første del av oppgaven representerer det å hente inn en "GO-BGC"-flyter, for så å plassere ut vår egenproduserte flyter i et angitt område. Flyteren vår skal deretter gjennomføre 2 vertikale profiler, altså traversere dybden av bassenget to ganger. Andre del av oppgaven handler om å kartlegge posisjonen til vraket av skipet *Endurance*, som sank i Antarktis. Deretter skal det autonomt lages en fotomosaikk av vraket, og videre skal lengden av det måles. Både "GO-BGC"-flyteren og vraket av Endurance er simulert ved hjelp av PVC-rør. Under er oppgavebeskrivelsene gjengitt, med tilhørende poengfordeling.

3.1 MATE Floats!

Utplassering av flyter i et angitt område er vist i figur 14.



Figur 14: Plassering av selvlagd flyter i angitt område. Hentet fra [22].

- Hente inn en *GO-BGC* flyter.
  - Beregne hvor flyteren dukker opp fra vannet: **5 poeng**.
  - Plukke opp flyteren, simulert av et PVC-rør: **10 poeng**.
- Design og konstruksjon av en operativ flyter for vertikal profilering.
  - Bygge en flyter før konkurransen starter: **5 poeng**.
  - Plassere flyteren i et angitt område ved hjelp av ROV-en: **5 poeng**.
  - Flyteren fullfører vertikal profilering.
    - To ganger: **25 poeng**.
    - Én gang: **15 poeng**.

3.2 Endurance22

Kartlegging av området der vraket av *Endurance* ble funnet, se figur 15.

Figur 15: Området der vraket av *Endurance* er plassert. Vraket er simulert med den brune PVC-strukturen. Hentet fra [21].

- Finne og kartlegge posisjonen til *Endurance*.
  - Kjøre en transektlinje over området der vraket er. Vraket er simulert av PVC-rør: **10 poeng**.
  - Indikere på et kart hvor vraket er: **5 poeng**.
- Lage en fotomosaikk av vraket.
  - Ta bilder av alle delene av vraket: **5 poeng**.
  - Autonomt sette bildene sammen til en fotomosaikk: **20 poeng**.
  - Manuelt sette bildene sammen til en fotomosaikk: **10 poeng**.
- Måle lengden av vraket fra baug til akterende.
  - Innen 10 cm av den sanne lengden: **10 poeng**.
  - Innen 10.1 til 20 cm av den sanne lengden: **5 poeng**.

Videre vil ROV-en veies, og belønnes med 10 ekstrapoeng dersom egenvekt i luft er mindre enn 20 kg, og 5 ekstrapoeng med egenvekt under 28 kg. Det deles ikke ut poeng dersom ROV-en er tyngre enn 28 kg, og veies den til over 35 kg resulterer det i diskvalifikasjon. Nytt for året er at det ikke tas størrelsesmål av farkosten. Likevel begrenses størrelsen på ROV-en av dokking-stasjonen på én kubikkmeter.

Oversikt over poengfordeling

Det gås ikke inn på detaljer rundt poenggivning i forhold til dokumentasjon og sikkerhet, da det ikke har betydning for den tekniske delen av prosjektet. Tabellen under viser en oppsummering av den totale poengfordelingen i konkurransen.

Produktdemonstrasjoner	
Oppgave 1-3	300 + tidsbonus
Vektrestriksjoner	10
Organisatorisk effektivitet	10
Prosjektering og kommunikasjon	
Teknisk dokumentasjon	100
Produktpresentasjon	100
Markedsføring	50
Bedriftens spesifikasjonsark	20
Bedriftsansvar	20
Bruk av VR-utstyr	25
Sikkerhet	
Gjennomgang av sikkerhet og dokumentasjon	20
Sikkerhetsinspeksjon	30
Sikker jobbanalyse	10
Totalt	695 + tidsbonus

Begrensninger og restriksjoner

Fysiske begrensninger for ROV-en omfatter ikke flyteren, da denne regnes som en egen enhet.

I utforskerklassen spesifiseres det at deltakere skal designe en ROV, uten eksterne komponenter, for å løse de praktiske oppgavene. Alle deler og komponenter skal være festet i ROV-ens ramme. Thrusterene skal ha beskyttelse som ikke tillater fingre å berøre propellene. Farkosten skal operere i kloret ferskvann med temperaturer i området 15 ^{o}C - 30 ^{o}C. Det nevnes også at uforutsette strømninger i vannet kan oppstå på grunn av bassengets filtreringssystem. Bassenget oppgis til å være maksimalt 6 meter dypt, og alle oppgaver utføres innenfor 10 meter fra bassengkanten. Stasjonen for skjermer og styresystem vil maksimalt være plassert 3 meter fra bassenget. Lengde på navlestreng må altså beregnes etter disse målene. Vinsj eller kran for sjøsetting er ikke tillatt. Dette utføres manuelt.

Arrangøren stiller med kraftforsyning, og dette er den eneste forsyningen man har tilgang til. Ekstra batterier og lignende tillates ikke. Kraftforsyningen leverer en nominell spenning på 48 VDC. I år er også deltakere pålagt å bruke en av MATE sine elektriske sikringer på 20 A, 25 A eller 30 A. Dette begrenser altså effektforbruket til 1440 W. All spenningsregulering skal foregå i elektronikkhuset til ROV-en, og kan ikke overgå 48 VDC. I tillegg er det oppgitt at koblinger og motorer som er eksponert i vann skal vanntettes, og ha en minimum resistans på 10 MΩ til vannet.

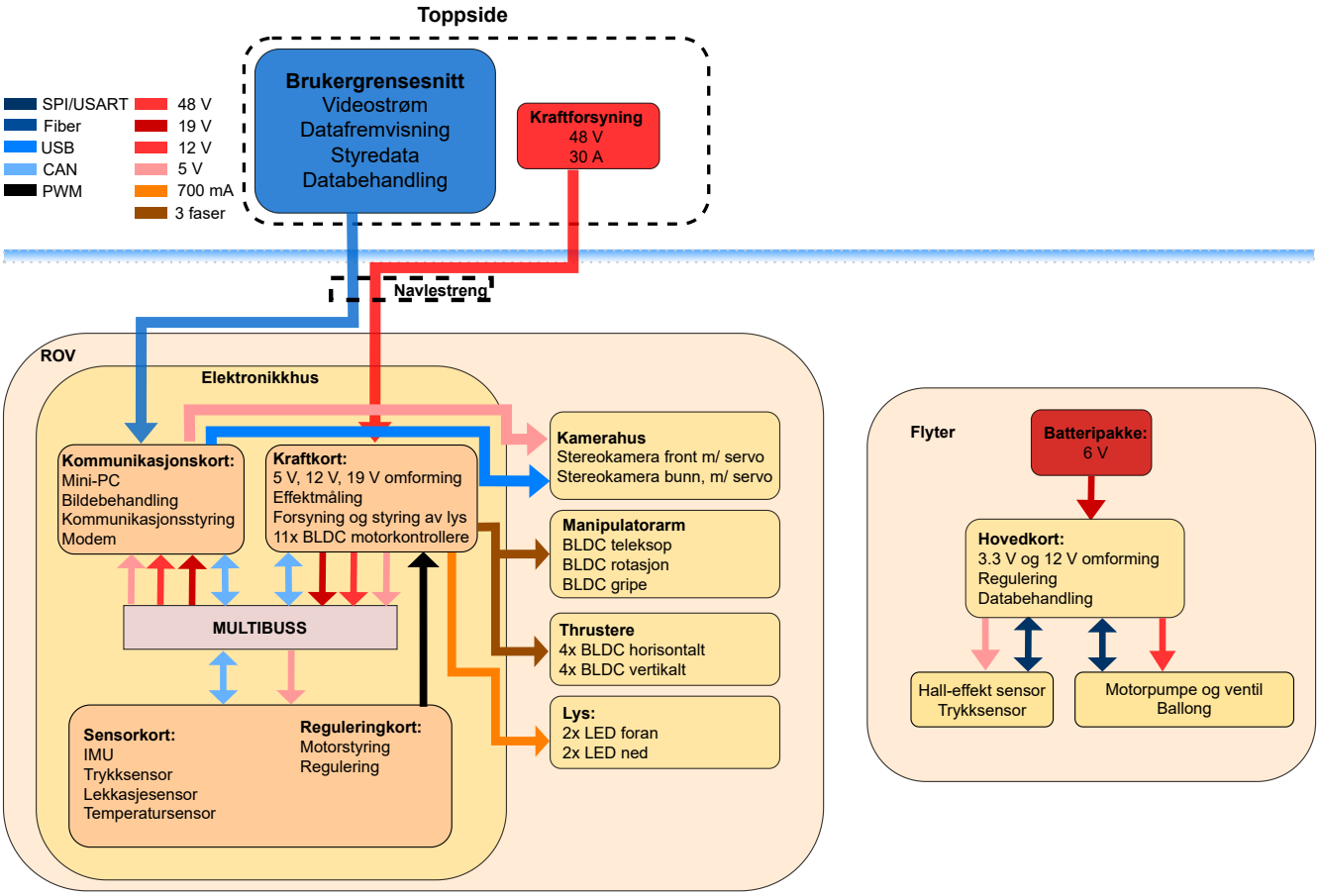
Flyteren skal være bygget fullstendig adskilt fra ROV-en, og skal fungere som en egen enhet. Størrelsen på flyteren er begrenset til 1 m i høyde, og 18 cm i diameter. Batterier ombord brukes som kraftforsyning. Her angir arrangøren en maksimal spenning på 12 VDC, og maksimalt strømtrekk på 6 A. Det kreves i tillegg en sikring på 7.5 A, som maksimalt kan plasseres 5 cm fra positiv pol på batteripakken.

1.4 Overordnet system

Figur 16 viser et blokkskjema av det overordnede systemet. ROV-en styres fra et brukergrensesnitt på toppsiden. Herfra sendes det styredata, i tillegg til at det vises videostrøm og status fra ROV-en. Kraftforsyning og fiberkommunikasjon fra toppsiden går gjennom en navlestreng. Kraftforsyningen går direkte inn på kraftkortet. Her foregår omforming til nødvendige spenningsnivåer, og distribuering av disse. Lys, motorer og motorkontrollere forsynes direkte fra kraftkortet. Fiberkommunikasjonen går inn på en mini-PC, som behandler data og organiserer kommunikasjonen i systemet. Kommunikasjonssignalene går inn på et kommunikasjonskort, via USB, og sendes videre over den lokale CAN-bussen. Mini-PC-en er også koblet direkte til kameraene, og driver kontinuerlig bildebehandling. Både kommunikasjon og kraftforsyning sendes ut til det interne systemet i elektronikkhuset, via en felles multibuss. Det siste kortet er et kombinert sensor- og reguleringskort. Sensordelen av kortet leser og behandler data fra de ulike sensorene i systemet. Reguleringsdelen sørger for styring av thrustere og manipulatormotorer basert på styredata fra toppsiden. I tillegg utføres regulering av thrustere fra sensordataen. Pådragsdata sendes videre via motorkontrollerne på kraftkortet, og ut til aktuelle motorer.

Flyteren drives av en batteripakke på 6 V, som går direkte inn på et hovedkort. Her omformes spenningen fra 6 V til både 3.3 og 12 V. I tillegg gjennomføres databehandling og regulering i mikrokontrolleren på kortet. Sensordataen hentes fra to eksterne sensorkort, som inkluderer én Hall-effekt-sensor, og én trykksensor. Mikrokontrolleren sender også styredata til motorpumpa og luftventilen, som sammen med

en ballong utgjør oppdriftsmotoren.



Figur 16: Overodnet blokkskjema av ROV-en.

1.5 Prosjektinndeling

Prosjektet er delt opp i flere oppgaver, hvor hver bachelorgruppe er tildelt et eget ansvarsområde. Totalt utgjør dette 9 bacheloroppgaver, hvor 5 av disse er innen elektro, 2 innen maskin, 1 innen data og 1 innen økonomi. Nedenfor presenteres hver oppgave i korte trekk, med ansvarsområde og gruppemedlemmer.

1.5.1 Elektro

Kraftforsyningsmodul – Carl Henrik Preber Ettesvoll, Nicolai Jensen Narvesen og Jon Arve Andersen:

Oppgaven handler om å utvikle en kraftforsyningsmodul til ROV-en. I konkurransen skal ROV-en få elektrisk forsyning fra land via en navlestreng. Konkurransemanualen oppgir at det blir tilført 48 V spenning, og at strømbegrensningen er på 30 A. Kortet skal sørge for at all elektronikk blir tilført riktig

spenningsnivå, og at effektforbruk overvåkes der det er nødvendig. Det skal også kunne prioritere kritiske prosesser, ved hjelp av digitalt styrte sikringskretser. Kraftkortet må ha rask responstid, kombinert med et robust design.

Maskinsyn og kommunikasjon – Christoffer Næss, Mats Røste og Tage Mellemstrand:

Oppgaven er todelt. Maskinsyn-delen går ut på å hente ut informasjon fra stereokameraene. Dette innebærer blant annet avstands- og størrelsesmåling, gjenkjenning av objekter og nødvendig informasjon for å løse de autonome oppgavene. Kommunikasjonsdelen går ut på å lage et robust kommunikasjonssystem i samarbeid med brukergrensesnittgruppen. Systemet skal kunne motta kommandoer fra toppsiden, behandle disse og sende dem videre til mikrokontrollere, som styrer de lokale prosessene i ROV-en. I tillegg skal systemet hente bilder fra kameraer, og sender dem tilbake til toppsiden, med så lite forsinkelse som mulig.

Sensorkort – Jørgen Hemnes Johannessen:

Oppgaven handler om å utvikle et sensorsystem til ROV-en. Dette innebærer å hente inn rådata fra sensorer, behandle dem og distribuere dem videre over kommunikasjonssystemet. Sensordata som behøves til reguleringen, er vinkelhastighet og -posisjon, samt dybden ROV-en befinnner seg på. Det må i tillegg være mulig å detektere kritiske feil, som for eksempel vannlekkasje eller overoppheting. For å realisere systemet, må det utvikles et kretskort med en mikrokontroller som sørger for tilfredsstillende databehandling. Vinkeldata leses fra en *Inertial Measurement Unit* (IMU) som settes på kortet. Resterende data leses fra eksterne sensorer. Dette innebærer trykk- og temperaturmåling, i tillegg til lekkasjedeteksjon.

Styrings- og reguleringssystem – Tomas Royal Choat, Kristian Birkeland og Otto Nessa Ljosdal:

Oppgaven går ut på å utvikle et motorsystem for ROV-en. Dette består av thrustere for manøvrering, samt elektriske motorer for bevegelse av manipulatorarmen. Det skal utvikles et robust styringssystem for alle motorer. Dette må kunne motta manuell styredata fra toppsiden, eller autonom styredata fra mini-PC-en, og generere motorpådrag etter ønsket oppførsel. For å forenkle manøvreringen, skal det i tillegg utvikles et reguleringssystem, bestående av tre PID-regulatorer. Disse skal sørge for at ROV-en automatisk kan holdes ved en ønsket dybde, og at den ligger stabilt i vannet. Sistnevnte oppnås ved å motvirke rotasjon fremover, bakover og sideveis.

Utvikling av smart flyter – Malin Harr Overland og Hanne Lovise Berger:

Oppgaven går ut på å utvikle en trådløs flyter, som automatisk kan manøvreres i vertikal retning. Manøvreringen realiseres ved hjelp av et styresystem, og en oppdriftsmotor. Motoren består av en pumpe og en solenoidventil, som endrer oppdriften ved å forflytte luft fra innsiden av flyteren, og ut i en ballong. For at styresystemet skal vite når motoren må aktiveres, kreves et sensorsystem. Dette inkluderer en trykkmåler for avlesing av dybdeendring, og en Hall effect-sensor for å detektere om flyteren holdes av ROV-en. Flyteren skal i tillegg drives på batterier, og systemet må derfor tilpasse spenningsnivå til de ulike formålene. Dette gjøres på et hovedkort, som også sørger for tilfredsstillende styring og databehandling.

1.5.2 Maskin

Design og montering av ROV og flyter - Christine Nordal og Sandra Nygård:

Oppgaven går ut på å designe og bygge rammen til ROV-en, samt den indre og ytre konstruksjonen til flyteren, ved å følge produktutviklingsprosessen. Den indre konstruksjonen til flyteren inkluderer oppdriftssystemet. Fokuset ved designet er å sikre at farkostene er godt egnet til å utføre oppgavene i konkurransen. I år rettes det også et økt fokus mot bærekraft og miljø. Det vil derfor bli lagt vekt på konseptet *Design for environment* (DFE) i utviklingsprosessen. Målet er å redusere miljøpåvirkningen i utviklingen av produktene. Effektiv bruk av DfE kan også bidra til reduserte kostnader og produksjonstid, i tillegg til å øke kvaliteten på produktet. Oppgaven inneholder også materialvalg, dimensjonering, strukturell analyse, flytanalyse og beregning av oppdrift og stabilitet. Dette for å sikre at produktene er stabile, har riktig oppdrift og er lette å manøvrere.

Design og produksjon av manipulator – Henrik Welde og Sindre Rød Torsteinsen:

Oppgaven går ut på å designe en manipulator som er i stand til å gripe tak i objekter. Fokuset for designet er å gjøre den så enkel som mulig, uten at dette går på bekostning av funksjonaliteten. Dette vil forenkle demontering, i tilfelle vedlikehold må utføres. Samtidig skal det tilstrebes å produsere armen så lett som mulig, da redusert vekt vil bidra til både bedre manøvreringsegenskaper, og poeng i konkurransen. Klypen må være i stand til å gripe objekter av forskjellige størrelser og former. I tillegg skal den ha mulighet for rotasjon og teleskopbevegelse.

1.5.3 Data

Operatørgrensesnitt og kommunikasjon – Vebjørn Lia Riiser og Åse Jortveit Sagebakken:

Oppgaven som skal løses, er å utvikle et system for å styre og overvåke ROV-en. Det må derfor implementeres et system for å sende kommandoer og styringsdata fra toppsiden til mini-PC-en i ROV-en. Dette gjøres i samarbeid med maskinsyn-gruppen. I tillegg må det designes et brukergrensesnitt som presenterer videostrøm og sensordata i sanntid. Hovedfokuset for grensesnittet er brukervennlighet, som innebærer fremvisning av nødvendig informasjon på en oversiktlig måte.

1.5.4 Økonomi

Endringsprosess i UiS Subsea – Maren Lovise Jåsund, Sina Brunes og Sanna Sørskår:

UiS Subsea er i år utvidet ved å involvere en gruppe fra økonomi og administrasjon. Målene fremover er blant annet å utvide organisasjonen til å involvere flere fagfelt, og ha flere pågående prosjekter samtidig. Ved hjelp av John P. Kotters 8-stegs modell, skal gruppen se på hvordan UiS Subsea på best mulig måte kan fortsette endringsprosessen av organisasjonen. Hovedsakelig innebærer dette organisering av de økonomiske og administrative oppgavene. UiS Subsea er hovedkilden. For å få et større perspektiv på økonomi og administrasjon, intervjues i tillegg tre andre bedrifter som organiserer disse oppgavene på forskjellige måter. Innhentet informasjon legger grunnlaget for videre analyser av hvordan UiS Subsea på

best mulig måte kan organiseres med flere medlemmer og prosjekter. De strategiske analysene som skal gjennomføres, er $SWOT$-analyse og strategilerret. Analysemetodene vil gi god innsikt i bedriftene, og et godt grunnlag for å sammenligne dem.

1.6 Utvikling av smart flyter

Fokuset for denne oppgaven er å utvikle en flyter som er uavhengig av ROV-en. Utviklingen tar utgangspunkt i kravene som er gitt i konkurransen. Dette innebærer å lage en oppdriftsmotor, finne passende komponenter til denne, velge passende sensorer til å kunne gjennomføre to profiler automatisk, og å designe et passende strømforsyningssystem.

For at alle delene i systemet skal fungere sammen, er det nødvendig å designe et kretskort som kan fungere som et grensesnitt til å kommunisere mellom alle delene. Kretskortet må inneholde en mikrokontroller som kan programmeres til utføre sekvensstyring av flyteren.

2 Systemkonstruksjon

Innhold

I dette kapittelet presenteres det hvordan oppgaven er ønsket løst, forslag til hvordan den skal løses, og spesifikasjoner til hva som trengs for å løse oppgaven. Systemkonstruksjon er hele prosessen i oppgaven, fra å definere hva som skal lages, til ferdig produkt. Dette blir gjerne delt inn i flere underdeler. Først kommer planleggingsfasen med tre underdeler:

- Behovsspesifikasjon
- Funksjonsspesifikasjon
- Systemdesign

Deretter følger realiseringen av prosjektet, også i tre deler:

- Maskinvare
- Programvare
- Testing av systemet

Til slutt må systemet evalueres og testes, og eventuelle endringer må gjøres. Planleggingsfasen dekkes i dette kapitellet, realisering og evaluering kommer senere i egne kapitler.

2.1 Behovsspesifikasjon

Behovsspesifikasjonen tar for seg behovet for oppgaven. Det vil si hva som trengs, hvilket problem som skal løses, og eventuelt hva som er ønskelig. Behovene settes ofte av kunden, og i dette tilfellet settes de av konkurransen *MATE ROV Competition*. Behovsspesifikasjonen skal være løsningsnøytral og fokusere på hva som trengs, ikke hvordan det skal løses.

Behov:

- Flyteren skal være maksimalt en meter høy og ha en diameter på maksimalt 18 cm.
- Flyteren skal kunne utføre en vertikal profil, som vil si å synke til den treffer bunnen, for så å flyte opp igjen til overflaten.

- Flyteren skal bli plassert ut og deretter utføre to vertikale profiler.
- Flyteren skal ikke være koblet fysisk til land eller til ROV-en.
- Flyteren må drives av batteri, med maksimalverdier på 12 V og 6 A.
- Flyteren må klare seg ned til 4 m dybde.

Ønsker:

- Flyteren kan detektere når den har nådd bunnen og overflaten.
- Hastigheten og posisjonen til flyteren kan lagres og hentes ut etterpå.
- Flyteren har regulering som gjør at den kan stoppes på en gitt dybde i vannet.
- Det er ønskelig at flyteren kan gå ned til 10 m dybde.

2.2 Funksjonsspesifikasjon

Funksjonsspesifikasjonen skal ta for seg hovedfunksjonene som systemet må eller bør inneholde, med forslag til løsning. Denne brukes som en idemyldring, så alle løsningene som skrives her, blir ikke nødvendigvis brukt i det endelige produktet. Dette skrives likevel om i rapporten, siden denne skal dekke alt som det er blitt brukt tid på, og ikke bare det endelige produktet.

- Flyteren skal ha en av/på vippebryter, montert på utsiden av flyteren, som kobler fra batteriet når den er avskrudd.
- For at flyteren skal skjønne at den er plassert ut av ROV-en, ble det vurdert to muligheter:
  - flyteren har trådløs tilkobling til en kontroll på land.
  - flyteren blir holdt av ROV-en, og det brukes en Hall effect-sensor på flyteren, samt en magnet på ROV-en for å detektere et magnetfelt når flyteren blir holdt.
- For vertikal profilering av flyteren, skal det brukes en oppdriftsmotor som flytter luft fra innsiden av flyteren til en ekstern ballong, plassert på utsiden av flyteren.
- Ved bruk av luft, kan det brukes to ulike alternativer:
  - en pumpe som pumper luft fra flyteren og inn i den eksterne ballongen, samt en solenoidventil som slipper lufta tilbake inn i flyteren igjen.
  - en lineæraktuator som, ved hjelp av et stempel, presser luft fra et eget kammer inne i flyteren og ut i ballongen, og som suger lufta tilbake inn i kammeret igjen.
- Batterier som kan brukes er spesifisert av *MATE ROV Competition*, og er følgende: AAA, AA, A, A23, C, D eller 9V alkaline batterier.

- Batteriet må ha en sikring, på maksimalt 7.5 A, plassert maksimalt 5 cm unna batteriets positive pol.
- For å detektere at flyteren har nådd bunn eller overflate, kan ulike sensorer brukes. Forslag til disse er gyroskop, trykksensor, akselerometer og ultralydsensor.
- For styring og lagring av data, skal det brukes en mikrokontroller.
- For styring av pumpe og ventil, skal det designes et eget kraftforsyningskort med krafttransistorer, eller releer, som får av/på signal fra mikrokontrolleren. På dette kretskortet må en også ha nødvendige spenningsregulatorer og batteriene tilkoblet.
- Ved design av kretskort, bør en legge til overspenningsvern for å beskytte diverse komponenter.
- Kretskortet må også inneholde grensesnitt for sensorene som skal brukes.

2.2.1 Budsjett

Siste del av funksjonsspesifikasjonen er å bestemme et budsjett. Budsjettet på det elektriske systemet til flyteren er satt til 8000 kroner av UiS Subsea. I tabell 1 er et estimert budsjett vist.

Utstyr	Type	Leverandør	Pris pr.	Antall	Frakt	Pris totalt
Mikrokontrollerkort	ARM MINI-M4 STM32	RS Components	195	1		195
Plattformgrensesnitt og Breakout board		JLCPCB	700	1	417	1117
Komponenter *Plattformgrensesnitt*		Digikey				1890
Pumpe	Vakumpumpe	Supermagneter	629	1	76	705
Ventil	3/2 Compact Solenoid Valve	RS Components	387	1	110	497
Trykksensor	MS5803-05BA	RS Components	195	1		195
Trykksensorhylse	Potted Cable Penetrator 6 mm	Blue Robotics	59	1		59
Hall effect-sensor	DRV5053	Digikey	17	1		17
Batteri	BATT ALKALINE D	Digikey	14	30		420
Batteriholder	BATT HOLDER D	Digikey	44	2		88
Støpemiddel	Power Epoxy Universal	LOCTITE	130	1		130
SUM TOTALT						**5313**

Tabell 1: Estimert budsjett

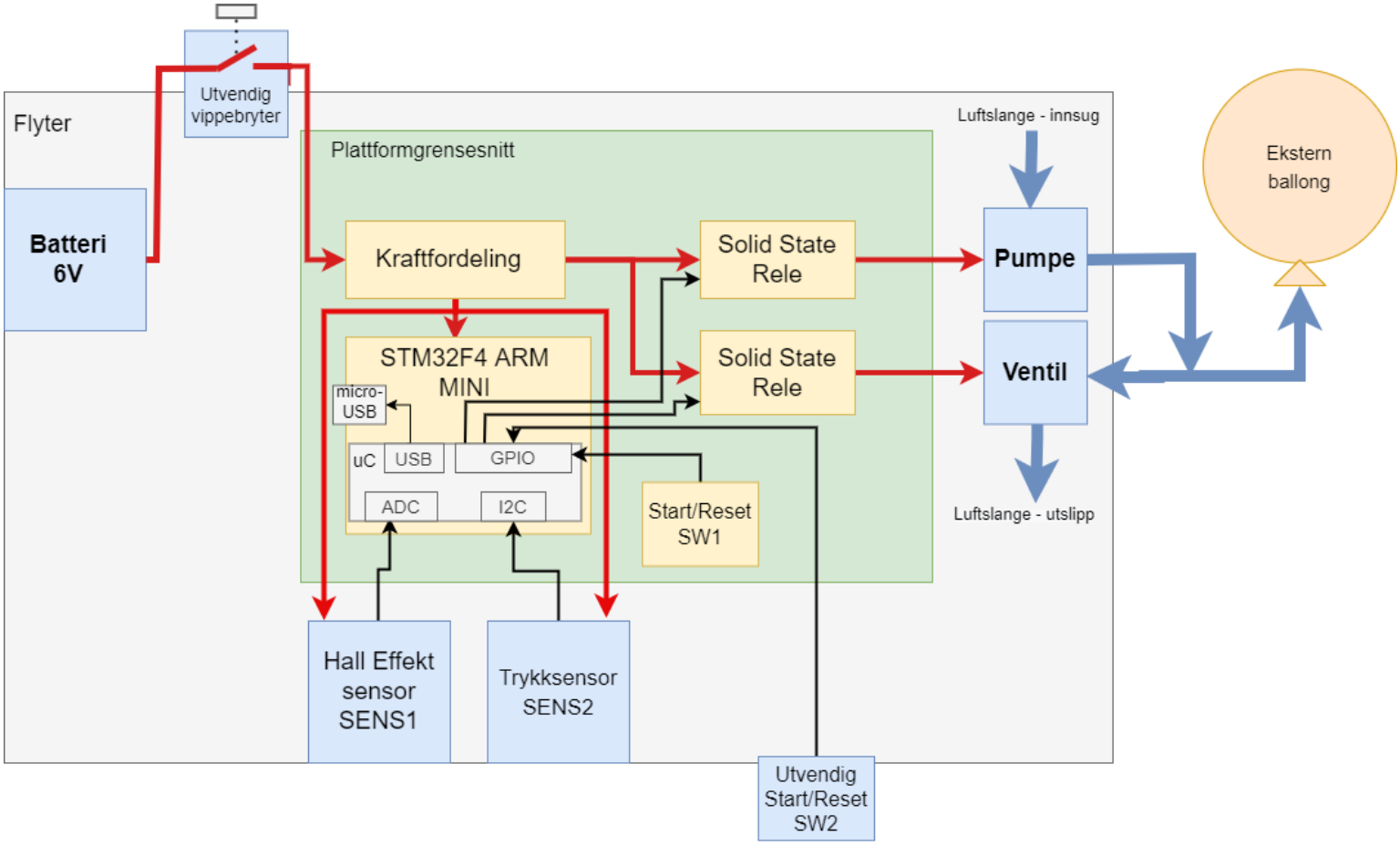
Ser her at vi ligger under utdelt budsjett med 2687 kroner. Noen av delene vi brukte til flyteren hadde UiS Subsea allerede liggende. Noen komponenter ble bestilt opp av andre grupper, mens andre komponenter har vi kjøpt for dem.

2.3 Systemdesign

Dette delkapittelet går mer teknisk inn på maskin- og programvaren. Aller først må det defineres hvilke tekniske krav som stilles til systemet:

- Systemet skal operere med maksimalt 12 V og 6 A.
- For å detektere plassering i vannet, brukes en trykksensor med I^2C-kommunikasjonsprotokoll.
- For å detektere at ROV-en har sluppet flyteren, brukes det en analog Hall-effect-sensor, som bruker ADC-modulen til mikrokontrolleren.
- Mikrokontrolleren som brukes er *STM32F415RG*, montert på plattformkortet *STM32F4 ARM MINI*.
- For programmering av mikrokontrolleren brukes et *STM32 Nucleo-64 Board* med *SWD*-programmeringsprotokoll.
- Programvaren som brukes for programmering er *STM32CubeIDE* med programmeringsspråket C.
- For lesing av verdier, ved testing og utvikling, skal plattformkortet kobles til PC via USB, og bruke seriell kommunikasjon for overføring av data. Mottak av data skjer i en programvare, skrevet i *Python*, på PC.
- Mikrokontrollerens moduler som skal tas i bruk er:
  - Ekstern klokke
  - ADC-modul
  - I^2C-modul
  - GPIOA
  - GPIOC
  - USB-modul

Et overordnet blokkskjema over hele systemet er vist i figur 17.



Figur 17: Blokkskjema av hele systemet.

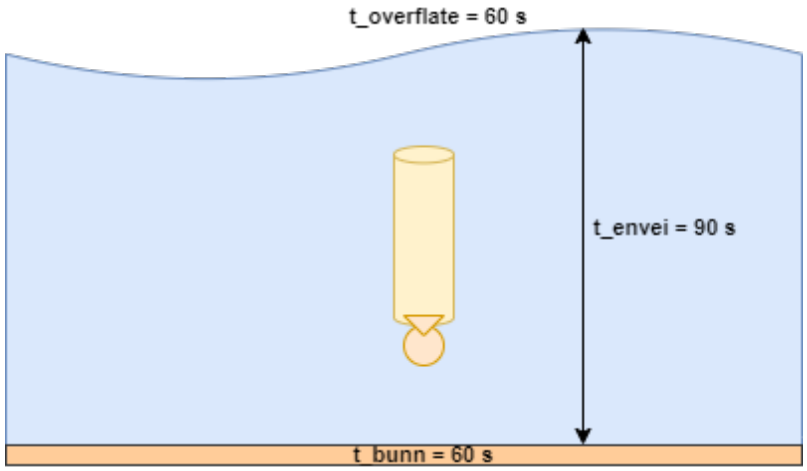
3 Fysiske egenskaper

Innhold

I dette kapittelet skal det lages en dynamisk modell av flyteren. Dette er for å kunne forutse tiden det tar fra flyteren blir satt ut i vann, til den har sunket til bunn, og til den er på vannoverflaten igjen. Her er det flere variabler en må ta hensyn til. Blant disse er oppdriften i vann, med tanke på vekt og størrelse av flyteren, samt hvor dypt bassenget er, og trykket i vannet.

Aller først må det bestemmes hvor lang tid flyteren kan bruke på en profil. I konkurransen er det satt av 15 minutter for hver oppgave. Ettersom det er flere deloppgaver, settes det av ti minutter for å gjennomføre de to profilene, altså fem minutter per profil. Det antas at flyteren trenger ett minutt i overflaten og på bunnen, for å detektere at den skal snu og blåse opp/tømme ballongen. Flyteren skal kunne utføre to profiler, dermed blir tiden den kan bruke på å synke ned eller flyte opp 1.5 minutt, som vist i ligning 1. Dette er også illustrert i figur 18.

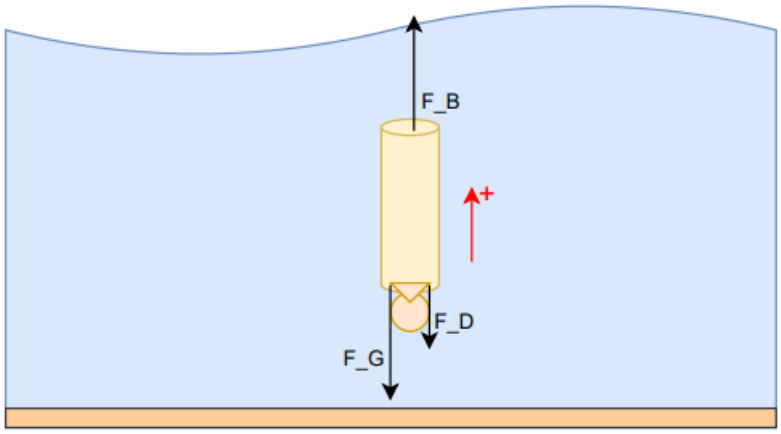
$$t_{envei} = \frac{10 - 1 \cdot 4}{4} = 1.5\ min = 90\ s \tag{1}$$



Figur 18: Illustrasjon av en profil med tider.

3.1 Litt om oppdrift

Når flyteren gjennomfører en profil i vannet, er det flere ulike krefter som virker på den. Fra fysikken er det ulike formler man kan bruke for å finne forholdet mellom disse kreftene. Alle kreftene er illustrert i figur 19.



Figur 19: Kreftene som virker på flyteren med positiv retning oppover.

Den første av kreftene som virker på flyteren er gravitasjonskraften, $\overrightarrow{F_G}$. Denne avhenger av vekten til flyteren, og kan regnes ut som i formel 2, hvor m_f er den totale vekten til flyteren i kg, og g er jordens gravitasjonskraft.

$$F_G = m_f g \tag{2}$$

Oppdriftskraften, $\overrightarrow{F_B}$ (B for Buoyancy på engelsk), virker i motsatt retning av gravitasjonskraften. $\overrightarrow{F_B}$ er lik vekten til vannet som blir forflyttet. Formelen for denne kraften er gitt i formel 3, og hentet fra [48]. Her er m_w vekten til vannet som blir forflyttet. Fra [48] har man at $m_w = V_w \rho_w$, hvor V_w er volumet til vannet som blir forflyttet og ρ_w er tettheten til vannet. Volumet av vannet som blir forflyttet av flyteren er likt volumet av flyteren, V_{tot}, som gir oss siste del av formel 3.

$$\begin{aligned} F_B &= m_w g \\ &= V_w \rho_w g \\ &= V_{tot} \rho_w g \end{aligned} \tag{3}$$

Den siste kraften som virker på flyteren er dragkraften, $\overrightarrow{F_D}$. Ettersom farten til flyteren økes gradvis når den synker eller flyter opp, vil dragkraften øke, og alltid virke i motsatt retning av bevegelsen til flyteren. Dette kan ses i illustrasjonen i figur 19. Den røde pilen viser positiv retning, det vil si at når flyteren flyter oppover, virker dragkraften i motsatt retning. Dette skjer fordi, når flyteren synker, beveger vannet seg oppover, og vil dermed prøve å "dra med seg" flyteren oppover. Dragkraften er gitt i formel 4, og er hentet fra [43]. De ulike variablene er listet opp under. Dragkoeffisienten avhenger av flyterens form og flate på de sidene som kommer i "motstand" med vannet, altså toppen og bunnen av flyteren. Ettersom

flyteren kan tilnærmes formen til en sylinder med en halv, hul, halvkule under, går man ut ifra tabellen i [43], og velger $c_d = 2.30$.

- c_d - dragkoeffisient, 2.30 (enhetsløs)
- ρ_w - vannets tetthet, 1000 kg/m^3
- v - farten [m/s]
- A - arealet til overflaten tilhørende dragkoeffisienten, 0.0123 m^2

$$F_D = \frac{1}{2} c_d \rho_w v^2 A \tag{4}$$

Formel 4 skrives om for enkelthets skyld til senere bruk, som vist i formel 5, hvor C_D er gitt som i formel 6. Dragkraftens retning er, som tidligere nevnt, avhengig av om flyteren er på vei nedover eller oppover [1]. Med denne omskrivningen, hvor man tar absoluttverdien av farten v multiplisert med den faktiske verdien, inkludert fortegn, ivaretar man dragkraftens retning.

$$\begin{aligned} F_D &= C_D v^2 \\ &= C_D |v| v \end{aligned} \tag{5}$$

$$\begin{aligned} C_D &= \frac{1}{2} c_d \rho_w A \\ &= \frac{1}{2} \cdot 1.17 \cdot 1000 \cdot 0.0123 \\ &= 7.2 \end{aligned} \tag{6}$$

Når man har bestemt kreftene som virker på flyteren, kan man finne den totale kraften som påvirker flyteren, ved å ta summen av kreftene F_B, F_G og F_D og bruke Newtons 2.lov, som vist i formel 7. Her er det tenkt at flyteren beveger seg oppover (i positiv retning), derfor blir gravitasjonskraften og dragkraften negativ.

$$\begin{aligned} \sum F &= F_B - F_G - F_D \\ &= g\rho_w V_{tot} - m_f g - C_D |v| v \\ &= ma \end{aligned} \tag{7}$$

I ligning 7 er både akselerasjon a, og fart v, variabler. Siden disse henger sammen med posisjonen x, som i ligning 8 og 9, kan de settes inn i ligning 7, og skrives om for å finne en andreordens differensialligning, vist i 10.

$$v = \int a \; dt \tag{8}$$

$$x = \int v \; dt \tag{9}$$

$$g\rho_w V_{tot} - m_f g - C_D |\dot{x}| \dot{x} = m\ddot{x} \tag{10}$$

Når flyteren skal synke, vil ballongen være tom, og det totale volumet V_{tot} er derfor kun volumet til selve flyteren. For at flyteren skal synke, må kreftene nedover, $\overrightarrow{F_G}$, være større enn oppdriftskraften, $\overrightarrow{F_B}$. Dette justeres ved å dimensjonere vekten på flyteren, til å oppfylle kravet vist i 11.

$$\overrightarrow{F_G} > \overrightarrow{F_B} \tag{11}$$

Når dette er oppfylt vil flyteren synke til bunnen. For å få den til å flyte opp, må oppdriftskraften kunne endres til den blir større enn gravitasjonskraften. Det er dette ballongen skal brukes til. Ved å øke volumet i ballongen, vil oppdriftskraften også økes (se ligning 3). Når oppdriftskraften blir større enn gravitasjonskraften, vil flyteren begynne å flyte oppover.

Det ble funnet, under vanntest, at når flyteren er 44 cm høy, har en diameter på 12.5 cm og ballongen er tom, må vekten være 5.769 kg for at flyteren skal ha negativ oppdrift. Dette blir, sammen med formlene som er utledet, brukt videre for å lage den dynamiske modellen av flyteren.

3.2 Dynamisk modell av flyter

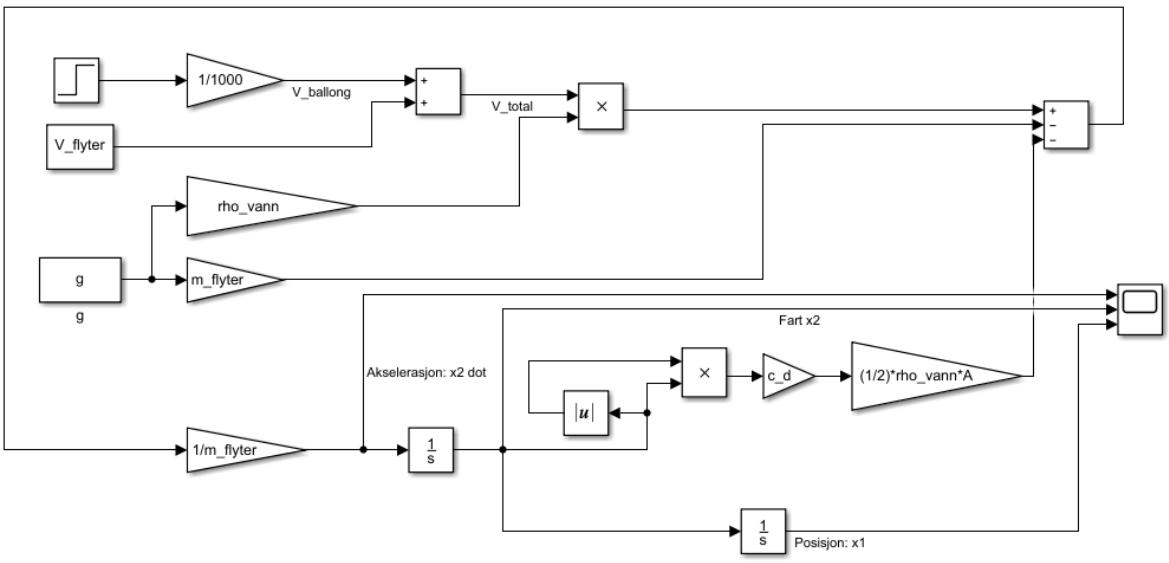
Ettersom det nå er funnet en andreordens differensialligning (10), kan det lages en tilstandsrommodell for å regne ut farten og akselerasjonen til flyteren. En tilstandsrommodell er en strukturert måte å skrive en differensialligning for et system på [12].

Det innføres derfor et par nye variabler, x_1 for posisjonen x, og x_2 for farten v. Tilstandsrommodellen blir dermed som vist i ligning 12 og 13. Her har man snudd om på ligning 10, for å finne $\ddot{x}$, som er akselerasjonen.

$$\dot{x_1} = \dot{x} = x_2 \tag{12}$$

$$\dot{x_2} = \ddot{x} = \frac{1}{m_f}(g\rho_w V_{tot} - m_f g - C_D|x_2|x_2) \tag{13}$$

Ut fra denne tilstandsrommodellen kan det lages et blokkskjema i *Simulink*, som vist i figur 20.



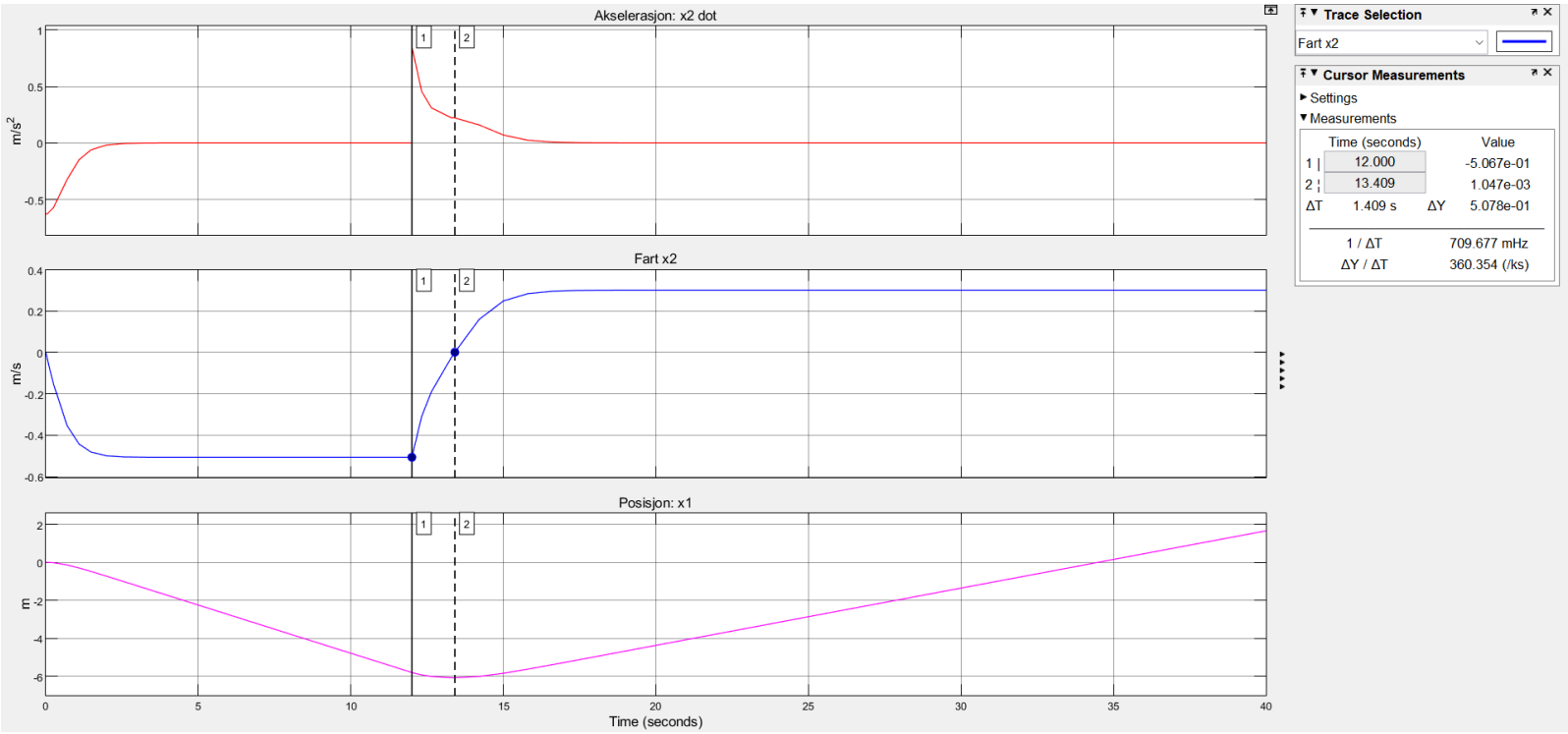
Figur 20: Blokkskjema laget fra tilstandsrommodellen.

Her tas alle variablene inn fra et skript i *MATLAB*, vist i kodeutsnitt 1.

```
%% Konstanter for flyter:
r = 0.125/2; % m
h = 0.44; % m
m_flyter = 5.769; % kg
g = 9.81; % m/s^2
rho_vann = 1000; % kg/m^3
A = pi*r^2; % m^2
c_d = 2.3;
%% Volum av float:
V_flyter = pi*r^2*h; % m^3
```

Kode 1: Konstantene til flyteren settes.

Resultatet sendes til et skop i *Simulink*-modellen, og grafene fra skopet er vist i figur 21. For å simulere volumet i ballongen, er det lagt inn en steg-funksjon som starter med verdi null, og blir satt til 0.5 (liter) etter tolv sekunder. Dermed viser grafene hvordan flyteren først synker (fordi ballongen er tom), og deretter stiger idet ballongen fylles. Her er det ikke lagt inn en begrensning for hvor dypt flyteren kan synke, og den fortsetter derfor nedover helt til ballongen fylles.



Figur 21: Resulterende graf fra tilstandsrommodellen.

Den øverste delfiguren viser akselerasjonen. Denne starter som en negativ verdi og øker opp til null. Dette er fordi farten, som er i den midterste delfiguren, synker fra start. Når farten synker, er akselerasjonen negativ, og når farten øker, er akselerasjonen positiv. Dette kan også vises matematisk som i ligning 8. Dersom man snur om på denne, får man at akselerasjon er den deriverte av farten. Da følger det at, dersom farten synker, er akselerasjonen negativ.

I $t = 12\ s$ ser man en spiker i akselerasjonen. Dette skjer på grunn av steg-funksjonen, som øker volumet i ballongen. Farten begynner dermed å stige, som gjør at akselerasjonen blir positiv. Farten stabiliserer seg rett etter det har gått 15 sekunder. Man ser også at akselerasjonen synker ved samme tidspunkt, og blir null.

Når farten er negativ, som i starten av figuren, betyr det at flyteren beveger seg i negativ retning, altså nedover. Man kan se i figuren at når ballongen får et volum (ved 12 sekunder), begynner farten å stige i verdi, men det tar litt tid før den blir positiv. Dette tidsrommet er markert med de to markørene som man kan se tilhørende verdier til, til høyre i figuren. Dette er rett og slett fordi flyteren i den dynamiske modellen vår trenger litt tid på å først senke hastigheten, til den står i ro, før den begynner å stige. Dette kan også ses i den nederste delfiguren, som viser posisjonen. Når ballongen får et volum i $t = 12\ s$ begynner grafen å slake ut, og i $t = 13.409\ s$ er bunnpunktet til posisjonen, før den begynner å stige. Det er i dette tidspunktet at verdien til farten blir positiv.

I praksis vil ikke flyteren endre retning på samme måte. I konkurransen er bassenget maksimalt 4 m dypt. Flyteren vil derfor stoppe idet den treffer bunnen, og trenger dermed ikke det samme tidsrommet til å senke hastigheten før den snur.

Ut ifra den dynamiske modellen, kan man konkludere med at flyteren er godt innenfor tiden den kan bruke på å utføre en profil. Her bruker den ≈ 34 sekunder fra den begynner å synke, til den er ved overflaten igjen. Dersom det legges til ett minutt for deteksjon av både nådd bunn og overflate, slik som diskutert i innledningen til kapittel 3, vil to profiler ta fire minutter og åtte sekund. Dette er godt innenfor kravet som er satt til å være ti minutter.

4 Maskinvare

Innhold

I dette kapittelet skal metode for å lage en oppdriftsmotor gjennomgås og begrunnes, samt valg av sensorer og batterivurdering. Det blir tatt i betrakning flere måter å løse de ulike deloppgavene på, samt vurdering av det man endte opp med å bruke. Gjennomgang av kretskort kommer i et eget kapittel: 5 “Maskinvare: Kretskortdesign”.

4.1 Vurdering av oppdriftsmotor

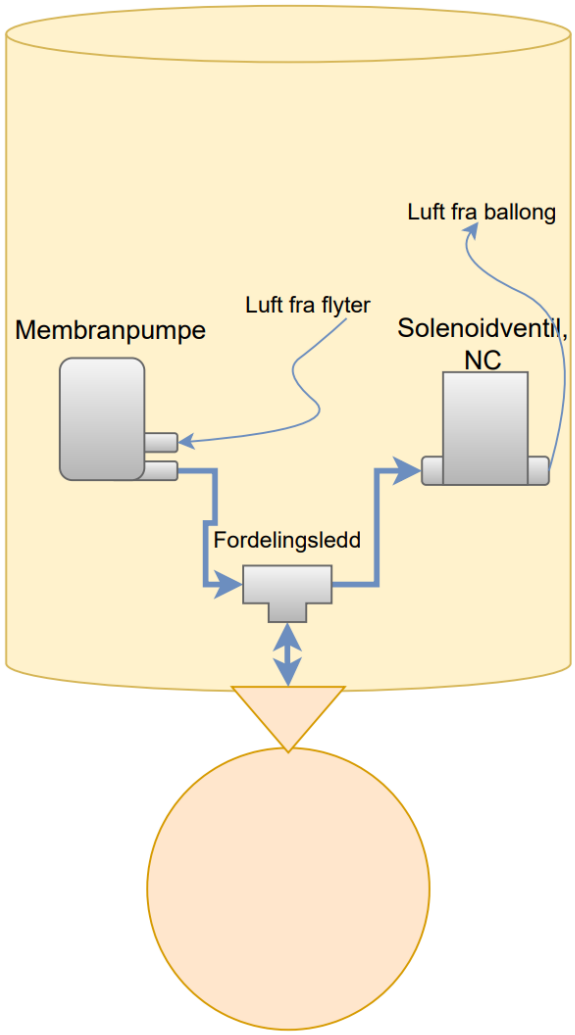
For å lage en oppdriftsmotor, ble det i kapittel 2.2, besluttet å bruke luft som medie, samt en ekstern ballong til å flytte luften ut i. Målet er å bruke luften inne i flyteren til å styre volumet til ballongen. Luften kan styres på to aktuelle måter:

1. Luften som allerede befinner seg inne i flyteren kan pumpes ut i ballongen ved hjelp av en luftpumpe/vakuumpumpe, og slippes inn i flyterens kammer igjen ved hjelp av en solenoidventil.
2. Luften kan isoleres i en egen sylinder inne i flyteren, og dyttes ut i ballongen ved hjelp av en lineæraktuator og et stempel, for så å “suges” inn igjen i sylinderen.

Videre skal det ses på fordeler, ulemper og hvilke komponenter som kan være aktuelle for begge metodene. Ut ifra dette vil vi kunne ta et valg om hvilken metode som egner seg best til vårt bruk ved å ta i betrakning batteribruk, størrelse, budsjett og fysikk.

4.1.1 Bruk av pumpe og solenoidventil

Prinsippet rundt det å bruke pumpe og ventil til å styre luften, går ut på at flyterens kammer har atmosfæretrykk. Dersom luften inne i kammeret pumpes ut i en ekstern ballong, vil trykket inne i kammeret synke, og bli lavere enn trykket i ballongen (forutsatt at volumet til flyterens kammer forblir det samme). Dersom en solenoidventil åpner for luften til å passere fra ballongen og inn i flyterens kammer igjen, vil luften inne i ballongen "dyttes" tilbake til kammeret. Dette skjer fordi trykket inne i kammeret er lavere enn trykket i ballongen, i tillegg til at trykket på utsiden av ballongen (trykket i vannet) er høyere enn trykket inne i ballongen. Figur 22 viser en illustrasjon av konseptet for bruk av pumpe og ventil til oppdriftsmotor.



Figur 22: Illustrasjon av bruk av pumpe og ventil som oppdriftsmotor.

Fordelen med bruk av pumpe, for å flytte luften som allerede er inne i flyteren, er at det er mer plasseffektivt ved design av flyteren. Dette fordi det ikke trengs et eget innvendig kammer, som har nødvendig volum, for å blåse opp ballongen.

Ulempen er at det trengs to komponenter for å gjøre det på denne måten, en pumpe og en ventil. Det vil si at ved bruk av en normalt lukket ventil, vil denne trekke strøm for å holde seg åpen mens flyteren synker ned, for å slippe ut luften fra ballongen. Deretter vil pumpa trekke strøm når ballongen pumpes opp igjen, for å få flyteren til å flyte oppover. Med andre ord, vil det alltid være et større strømtrekk under operasjon og batteriet vil dermed tappes raskere.

For valg av pumpe og ventil, må det tas hensyn til hvor mye trykk som trengs for å pumpe opp ballongen under vann. Formelen for trykk er vist i formel 14, hvor p er utregnet trykk, p_0 er trykk ved overflaten, ρ_w er tettheten til vannet, g er tyngdekraften og d er dybden i vannet målt i meter.

$$p = p_0 + \rho_w g d \tag{14}$$

Fra konkurransemanualen [20], vet man at bassenget ikke er dypere enn fire meter, men det er ønskelig at flyteren skal klare seg helt ned til ti meter. Trykket under vann på ti meters dybde blir som vist i ligning 15.

$$\begin{aligned} p &= p_0 + \rho_w g d \\ &= 100000 + 1000 \cdot 9.81 \cdot 10 \\ &= 198100\ Pa \\ &= 1.98\ bar \end{aligned} \tag{15}$$

Pumpa og ventilen må kunne operere med et høyere trykk enn det som er utenfor ballongen, altså her 1.98 bar. I tabell 2 er det vist en oversikt over aktuelle pumper med spesifikasjoner. Det maksimale arbeidstrykket i tabellen er relativt trykk, altså i tillegg til atmosfæretrykket. Det vil si at den øverste pumpa, i realiteten, fungerer på opptil 2 bar.

Pumpe				
Produktnavn	**Maksimalt arbeidstrykk**	**Strømtrekk**	**Spenning**	**Pris**
Elektrisk membranpumpe *RS PRO*	1 bar	190 mA	1.5 V - 4.5 V	731,10 kr
Vakuumpumpe *AIR PUMP SERIES D2028*	2.2 bar	1 A	12 V	629 kr

Tabell 2: Oversikt over ulike pumper.

I tabell 3 er det vist en oversikt over aktuelle solenoidventiler med spesifikasjoner. Det samme gjelder for maksimalt arbeidstrykk her, som for pumpene.

Ventil				
Produktnavn	**Maksimalt arbeidstrykk**	**Strømtrekk**	**Spenning**	**Pris**
3-veis/2-posisjons solenoidventil *VDW250-6G-1-M5-Q*	10 bar	250 mA	12 V	482,94 kr
2-veis solenoidventil *VX212AZ1D*	10 bar	375 mA	12 V	385,39 kr

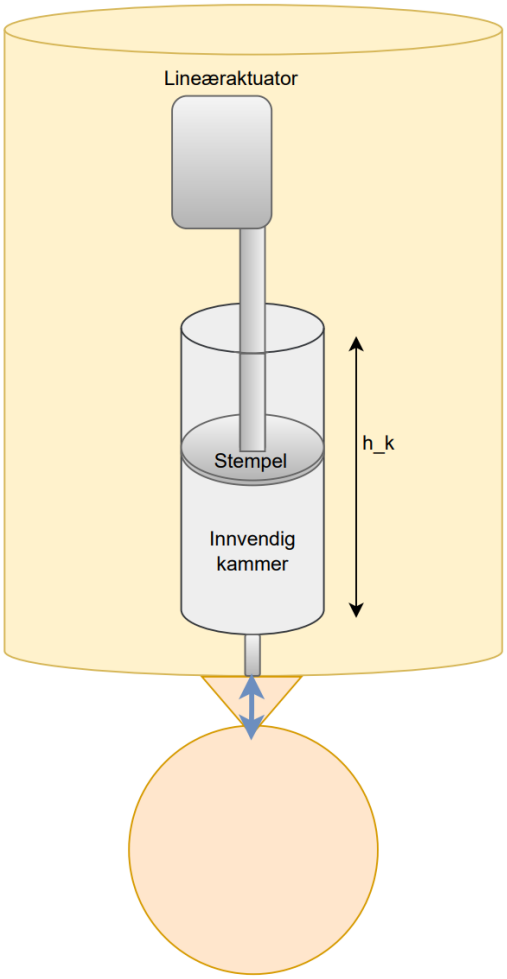
Tabell 3: Oversikt over ulike solenoid ventiler.

Ut ifra tabellene over er det funnet gode kandidater, med lavt strømtrekk, som tilfredsstiller kravet til maksimalt 12 V som forsyningsspenning og 6 A som strømtrekk.

4.1.2 Bruk av lineæraktuator

Fordelen med bruk av en lineæraktuator er at det, i motsetning til pumpe og ventil, kun trengs en komponent for å dytte luft inn i ballongen, for så å suge den ut igjen. Denne vil dermed i teorien kunne være mer energieffektiv, ettersom den drives av et av/på signal. Dermed vil den kun trekke strøm i en retning, som er når ballongen skal fylles med luft for at flyteren skal flyte opp. Når den ikke lenger får spenning, går aktuatoren inn igjen og suger med seg luften tilbake uten å trekke strøm.

Ulempen her, er at et eget kammer for luft plassert inne i flyteren vil kreve mer plass, kontra å bruke luften som allerede ligger i flyteren. Videre skal man også se på om det, i teorien, i det hele tatt er mulig å finne en lineæraktuator som trekker lite strøm, men som samtidig kan dytte kraftig nok. Figur 23 viser en illustrasjon av konseptet for bruk av lineæraktuator.



Figur 23: Illustrasjon av bruk av lineæraktuator som oppdriftsmotor.

I tabell 4 er en oversikt over to lineæraktuatorer med spesifikasjoner, hvorav den øverste er mest aktuell med tanke på lavt strømtrekk.

Lineæraktuator					
Produktnavn	**Maksimal arbeidskraft**	**Slaglengde**	**Strømtrekk**	**Spenning**	**Pris**
Actuonix elektrisk lineæraktuator	80 N	30 mm	185 mA	12 V	1508,81 kr
RS PRO elektrisk lineæraktuator	2500 N	300 mm	4.6 A	12 V	2172,79 kr

Tabell 4: Oversikt over ulike aktuatorer.

For valg av aktuator, må man finne nødvendig kraft aktuatoren må kunne dytte med, for å overvinne det maksimale trykket i vannet. Man må også finne krav for slaglengden på aktuatoren, som tilsvarer høyden h_k på det innvendige kammeret.

Man starter med å se på dimensjonene til flyteren. Det er ønskelig å dimensjonere flyteren mindre enn maksimumskravene, for å gjøre det enklere for ROV-en å gripe tak i, og flytte på flyteren. Størrelse på flyteren er satt av maskingruppa til å være 44 cm i høyde og 12.5 cm i diameter. Fra dette kan man sette størrelsen på det innvendige kammeret til å være maksimalt 30 cm høyt og 5 cm i diameter. Dette er et grovt estimat for vurderingen sin del. Dersom det blir tatt i betraktning hvor stor plass batteriet, kretskortet og ballasten, som også skal inn i flyteren, kommer til å ta, må sannsynligvis flyteren dimensjoneres større for denne størrelsen på det innvendige kammeret. Det maksimale volumet på det innvendige kammeret blir som vist i formel 16, med variablene listet opp under.

- V_{kammer} - volumet til det innvendige kammeret, $[m^3]$
- $A_{stempel}$ - arealet til stempelet, $[m^2]$
- h_k - høyden til det innvendige kammeret, [m]
- r_k - radiusen til det innvendige kammeret, [m]

$$\begin{aligned} V_{kammer,maks} &= A_{stempel,maks} \cdot h_{k,maks} \\ &= \pi r_k^2 \cdot h_{k,maks} \\ &= \pi(0.025\ m)^2 \cdot 0.3\ m \\ &= 5.8 \cdot 10^{-4}\ m^3 \\ &= 0.58\ L \end{aligned} \tag{16}$$

Det kan antas at volumet til det innvendige kammeret må være minst like stort som det volumet ballongen trenger, for å få ønsket oppdrift på flyteren. Med dimensjonene satt av maskingruppa, og som ble brukt i den dynamiske modellen (kapittel 3.2), ble det funnet at ballongen trenger et volum på rundt 5 dL. Det vil si at volumet til det innvendige kammeret, V_{kammer}, må være minst $5\ dL = 0.5\ L = 5 \cdot 10^{-4}\ m^3$. Dette er et minimumsvolum på kammeret, så det kan være større og heller begrenses ved at lineæraktuatoren ikke bruker hele slaglengden. Vi ser her at volumet, beregnet i ligning 16, er stort nok til å kunne fylle ballongen tilstrekkelig.

Frem til nå, har man kun sett på maksimal høyde på det innvendige kammeret. Ettersom det er satt et maksimumskrav på arealet til stempelet, finnes det også et minimumskrav på høyden. Dette finner man ved å sette inn maksimalt areal (som er gitt ved $\pi \cdot 0.025^2$) og minimalt volum i formel 16, og løse med hensyn på h_k, som vist i formel 17.

$$\begin{aligned} V_{kammer} &= A_{stempel} \cdot h_k \\ \Rightarrow h_{k,min} &= \frac{V_{kammer,min}}{A_{stempel,maks}} \\ \Rightarrow h_{k,min} &= \frac{5 \cdot 10^{-4}}{\pi(0.025\ m)^2} \\ h_{k,min} &= 0.255\ m = 25.5\ cm \end{aligned} \tag{17}$$

For å få ønsket volum i ballongen, med hensyn til plassbegrensningene, er det endelige kravet til høyden på det innvendige kammeret, som gitt i 18.

$$25.5\ cm \leq h_k \leq 30\ cm \tag{18}$$

Ved å se i oversikten over aktuelle aktuatorer, i tabell 4, ser man at slaglengden til den første er 30 mm, som tilsvarer 3 cm. Denne faller dermed ut av vurderingen ettersom den ikke oppnår kravet om at slaglengden må være minimum 25.5 cm. Den andre aktuatoren har en slaglengde på 300 mm = 30 cm, som er maksimumskravet til godkjent slaglengde. Det som gjenstår å finne ut, er den nødvendige kraften aktuatoren må kunne dytte med, for å overvinne det maksimale trykket i vannet. Fra forrige delkapittel, 4.1.1, fant man ut at det trykket som skal overvinnes er på 1.98 bar, altså 198000 Pa. For å gjøre trykk om til kraft, brukes formel 19, hentet fra [48]. Her er F_A kraften aktuatoren må kunne dytte med, og p_w er det maksimale trykket i vannet som aktuatoren må kunne overvinne. Enheten til p_w er Pascal, og kan skrives om til $\frac{N}{m^2}$ [48]. For å kunne bruke volumet og høyden til kammeret, snur man om på formel 17, og løser for $A_{stempel}$. Setter deretter dette inn i formelen for kraft, som vist i formel (19).

$$\begin{aligned} F_A &= p_w \cdot A_{stempel} \\ &= p_w \cdot \frac{V_{kammer}}{h_k} \end{aligned} \tag{19}$$

Ettersom aktuatoren, som er til vurdering, har en slaglengde på 300 mm, settes $h_k = 0.3\ m$. Kraften som er nødvendig regnes ut i formel 20.

$$\begin{aligned} F_A &= p_w \cdot \frac{V_{kammer,min}}{h_{k,maks}} \\ &= 198000\ Pa \cdot \frac{5 \cdot 10^{-4}}{0.3\ m} \\ &= 330\ N \end{aligned} \tag{20}$$

Her ser man at aktuatoren, fra tabell 4, har et maksimalt arbeidstrykk på 2500 N, som er langt over det som er nødvendig. Denne vil derfor kunne fungere.

4.1.3 Konklusjon

Sammenligner man resultatene fra vurderingene, ved bruk av pumpe og ventil og lineæraktuatoren, kan det konkluderes med at, selv om en lineæraktuator i teorien så ut til å virke enklest, ikke er det beste valget, dersom en tar hensyn til størrelse og energiforbruk.

Den siste lineæraktuatoren man så på, måtte ha et innvendig kammer på 300 mm, altså 30 cm. Dette vil ta betydelig større plass innvendig i flyteren enn om man bruker pumpe og ventil.

Sammenligner man strømforbruk, har lineæraktuatoren et maksimalt strømtrekk på 4.6 A, mens pumpa og ventilen totalt har 1.25 A. Det blir dermed naturlig at man velger det som er mest plass- og energieffektivt, altså pumpe og solenoidventil.

4.2 Vurdering av sensorer for dybdemåling

Da sensor for dybdemåling skulle velges, gikk vi ut fra tre ulike alternativer: akselerometer, trykk-, og ultralydsensor. Her skal det gås gjennom egenskaper til de forskjellige typene, samt konkluderes med hvilken som er mest hensiktsmessig å bruke. Sensoren som velges, må muliggjøre målingene som er spesifisert i kapittel 2. Mye av teorien bak de ulike sensorene er hentet fra [29].

4.2.1 Akselerometer

For å måle akselerasjonen til et objekt, bruker man et akselerometer. Akselerometeret er bygget opp som et skall med en masse inni. Massen er festet på en fjær, som har en fjærkonstant K. Når objektet blir utsatt for en akselerasjon, vil massen presse fjæren sammen med en viss distanse. Fjæren vil deretter presse tilbake med en kraft, F_s, som vist i ligning 21. Her er K fjærkonstanten, og x er distansen fjæren blir presset sammen. Akselerasjonen har også en kraft som virker på massen, F_a, som i ligning 22. Her er M massen og $\ddot{x}$ er akselerasjonen.

$$F_s = Kx \tag{21}$$

$$F_a = M\ddot{x} \tag{22}$$

For å unngå oscillasjoner i målingene, er det lagt inn en demper i akselerometeret. Denne lager en dempekraft, F_d, som vist i ligning 23. Her er B dempningsfaktoren og $\dot{x}$ farten til massen.

$$F_d = B\dot{x} \tag{23}$$

Når disse tre kreftene virker på massen, som vist i figur 24, kan de settes sammen som i ligning 24. Denne kan skrives om til en tilstandsrommodell, som i ligning 25 og 26. I tilstandsrommodellen er x_1 posisjonen og x_2 farten. Dermed følger det at $\dot{x_2}$ er akselerasjonen.

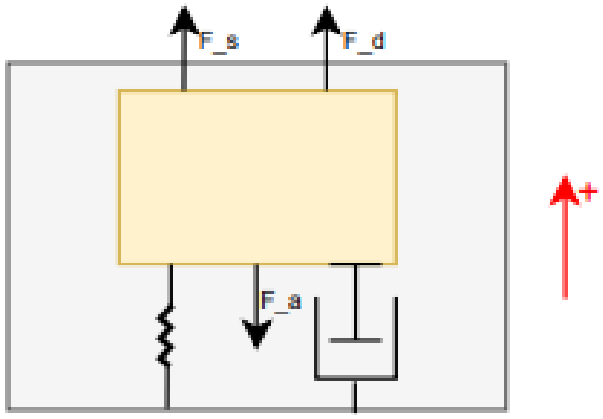
$$\sum F = F_s + F_d - F_a$$

$$\Rightarrow F_s + F_d = F_a \tag{24}$$

$$\Rightarrow Kx + B\dot{x} = M\ddot{x}$$

$$\dot{x_1} = \dot{x} = x_2 \tag{25}$$

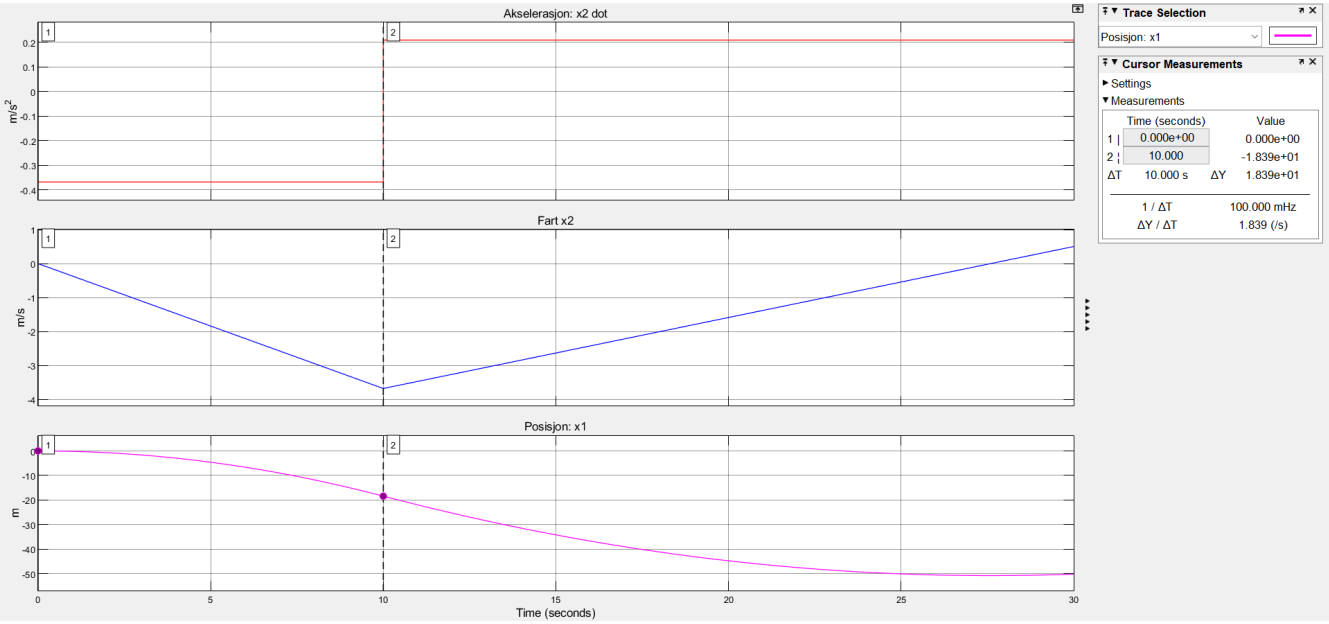
$$\dot{x_2} = \ddot{x} = \frac{Kx_1 + Bx_2}{M} \tag{26}$$



Figur 24: Massen inni et akselerometer med påvirkende krefter.

Det som skjer når akselerometeret blir utsatt for en akselerasjon, er at massen blir forflyttet med en distanse. Denne distansen er det som blir fysisk målt, og satt inn i ligning 26. Akselerometeret gir deretter ut den resulterende verdien derfra.

Hensikten med sensoren i vårt system, er at den skal detektere når flyteren har nådd bassengbunnen, eller vannoverflaten. Med den dynamiske modellen fra kapittel 3.2, kan det vises at dersom det ikke er en dragkraft som virker på flyteren i vannet, er akselerasjonen konstant når flyteren beveger seg gjennom vannet. Grafen for dette er vist i figur 25.



Figur 25: Dynamisk modell av flyteren uten dragkraft. I tidsrommet $0 < t < 10$ beveger flyteren seg nedover og i $t = 10$ får ballongen et volum og flyteren begynner å skifte retning.

Når flyteren treffer bunnen, blir derimot farten lik null, og akselerasjonen vil dermed også bli null. Dersom dette hadde vært tilfellet i realiteten, kunne et akselerometer blitt brukt til å detektere at flyteren har nådd bunn og overflate.

I realiteten, derimot, finnes det en dragkraft som virker på flyteren. Med dragkraften tatt i betrakning, stabiliserer farten seg underveis, som fører til at akselerasjonen etterhvert blir null, før flyteren når bunnen/overflaten. Dette kan observeres i figur 21. Derfor vil ikke et akselerometer fungere til sin hensikt. Man må derfor vurdere de to andre alternativene til sensoren.

4.2.2 Trykksensor

En trykksensor kan brukes for å måle trykket ved et objekt. Trykk kan defineres på tre forskjellige måter: absolutt, relativt og differansetrykk. Absolutt trykk er differansen mellom målt trykk og nullverdien av trykk. Relativt trykk er differansen mellom målt trykk og en atmosfæres trykk. En atmosfæres trykk er typisk satt til 1.013 bar, men den varierer med både høydemeter og værforhold. Absolutt og relativt trykk er relatert som vist i ligning 27.

$$Absolutt\ trykk = Relativt\ trykk + Atmosfærisk\ trykk \tag{27}$$

Differansetrykk er differansen mellom to verdier av absolutt trykk. Dette kan, for eksempel, være trykket på utsiden og innsiden av flyteren. Til vårt formål er ikke dette ideelt, ettersom trykket på innsiden av

flyteren vil endre seg underveis mens luften pumpes inn og ut.

Ved å måle trykket på utsiden av flyteren, kan det også regnes ut hvor langt under vannoverflaten den befinner seg. Ved å snu om på ligning 14, får man en formel for distansen, vist i ligning 28.

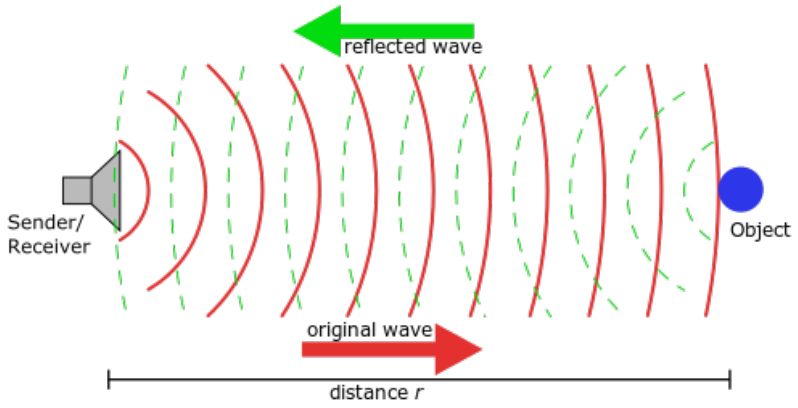
$$d = \frac{p - p_0}{\rho g} \tag{28}$$

Det er ulike måter en trykksensor kan fungere på. En av de mest vanlige, er en masse som blir forskjøvet når den opplever en trykkendring og at denne bevegelsen blir målt av en forskyvningstransduser. Den andre metoden fungerer ganske likt, men istedenfor å måle forskyvningen direkte måles endringen i kapasitans mellom de to platene som blir forskjøvet.

Ettersom trykket slutter å endre seg når flyteren er i bunnen eller overflaten av bassenget, kan man bruke en trykksensor til å måle når den har nådd disse posisjonene. Et krav for at trykksensoren skal fungere, er at den må plasseres på utsiden av flyteren. Dette kan enkelt løses ved å plassere den i topp- eller bunnlokket. En trykksensor vil også kunne gi oss et godt mål på hvor dypt i bassenget flyteren er. Dette kan være relevant dersom en ønsker å oppnå regulering av hvor dypt i vannet flyteren skal være.

4.2.3 Ultralydsensor

En ultralydsensor beregner avstand ved å sende ut lydbølger, og registrere tiden det tar før de reflekteres, og kommer tilbake. Dette er illustrert i figur 26.



Figur 26: Illustrasjon av en ultralydsensor. Hentet fra [45].

Denne må plasseres i bunnen av flyteren, for å måle avstanden fra flyteren til bunnen av bassenget. For å finne avstanden brukes formel 29. Her er d avstanden fra flyteren til bunnen av bassenget, t er tiden lydbølgene bruker fra de blir sendt ut til de blir mottatt, og v_0 er lydhastigheten under vann.

$$d = \frac{t \cdot v_0}{2} \tag{29}$$

For å bruke ultralydsensor i vårt system, må denne, som tidligere nevnt, være plassert i bunnen av flyteren. Problemet med dette, er at ballongen også må være plassert på bunnen av flyteren. Denne vil dermed komme i veien for ultralydsensoren. Med en ultralydsensor må man i tillegg passe på at det ikke er andre lydsensorer i nærheten som sender ut signal på samme frekvens, og dermed forstyrrer signalet.

4.2.4 Konklusjon

Etter å ha sammenlignet de ulike typene sensorer, valgte vi å bruke en trykksensor. Den har minst utfordringer av de tre alternativene, og kan lett brukes til både deteksjon av overflate/bunn, i tillegg til å måle hvor dypt i vannet flyteren befinner seg.

I tabell 5, er sensorene som ble vurdert av sensorgruppa i fjorårets Subsea-gruppe, vist. Under kommentarkolonnen har de skrevet det de ser som den største ulempen med hver av sensorene.

Produktnavn	Type	Måleområde	Oppløsning	Målefeil	Pris	Kommentar
Bar30	Absolutt	0 til 300 m	0.20 til 1.57 mbar	± 2 m	800 kr	For stor målefeil
Bar02	Absolutt	0 til 10 m	0.016 til 0.11 mbar	± 4 cm	900 kr	For lite måleområde
PS30	Absolutt	0 til 100 m	0.0025 mbar	0.05 cm	21000 kr	For dyr
PX3AG1BH010	Differanse	0 til 100 m	Analog spenning	± 1 m	500 kr	Differansetype
MS5803-14BA	Absolutt	0 til 140 m	0.2 til 1.0 mbar	± 20 cm	130 kr	Må lage sensorhus

Tabell 5: Ulike trykksensorer. Hentet fra [26].

I år har vi valgt en sensor som ikke er blitt brukt i UiS Subsea tidligere. Den heter *MS5803-05BA*, produseres av *TE Connectivity*, og skal også brukes på årets ROV. Databladet er vist i [6], der finner man verdier for måleområdet. Dette er oppgitt i bar og regnes om til meter, som vist i ligning 30 og 31. Her må trykket, p, gjøres om fra bar til pascal ved å multipliseres med 100000. Variablene er listet opp under, der både $p_{nedregrense}$ og $p_{øvregrense}$, som henholdsvis er det nedre og øvre trykket sensoren kan måle, er tatt fra *PRESSURE OUTPUT CHARACTERISTICS* på side fire i databladet.

- $p_{nedregrense}$ - det minste trykket trykksensoren tåler, 0 bar
- $p_{øvregrense}$ - det største trykket trykksensoren tåler, 5 bar
- ρ - vannets tetthet, 1000 kg/m^3
- g - jordens gravitasjonskraft, 9.81 m/s^2

$$
\begin{aligned}
p_{nedregrense} &= \rho \cdot g \cdot h \\
\Rightarrow h &= \frac{p_{nedregrense}}{\rho \cdot g} \\
&= \frac{0 \cdot 100000}{1000 \cdot 9.81} \\
&= 0\ m
\end{aligned}
\tag{30}
$$

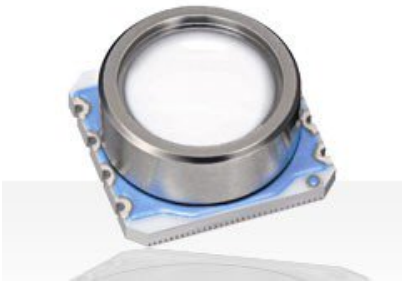
$$
\begin{aligned}
p_{øvregrense} &= \rho \cdot g \cdot h \\
\Rightarrow h &= \frac{p_{øvregrense}}{\rho \cdot g} \\
&= \frac{5 \cdot 100000}{1000 \cdot 9.81} \\
&\approx 50m
\end{aligned}
\tag{31}
$$

Målefeilen regnes ut på samme måte. Den er vist, sammen med flere verdier, i tabell 6.

Spesifikasjon	***MS5803-05BA***
Driftspenning	1.8-3.6 V
Måleområde	0 til 50 m
Målefeil	± 4 cm/± 80 cm
Oppløsning	0.036 til 0.195 mbar
Grensesnitt	I2C/SPI
Pris	195
Kommentar	Må støpes

Tabell 6: Verdier for MS5803-05BA.

Sensoren er vist i figur 27, og må støpes inn i en hylse, slik som den i figur 28, for at flyteren skal være vanntett.



Figur 27: Trykksensoren MS5803-05BA. Bildet er hentet fra [6].

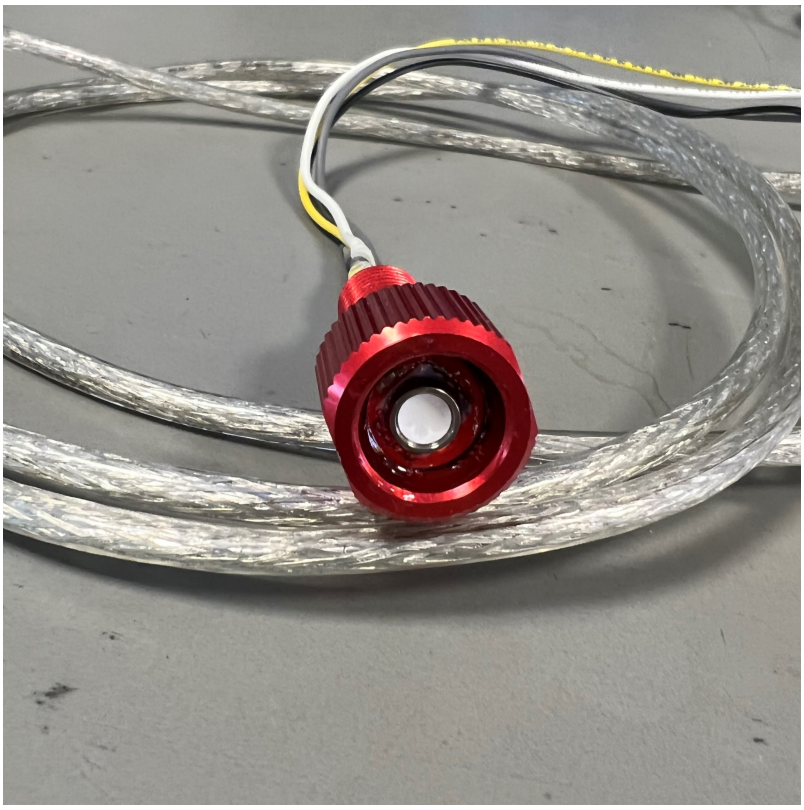
Figur 28: Hylsen trykksensoren støpes inn i. Bildet er hentet fra[3].

I og med at flyteren skal designes til å kunne gå ned til ti meter, er måleområdet på denne sensoren mer enn stort nok. Trykksensoren skal kun måle om flyteren har nådd bunnen eller overflaten, det er dermed ikke er nødvendig å kunne regne ut nøyaktig posisjon i vannet. Oppgitt målfeil fra databladet, er relativt stor i forhold til bruksområdet flyteren skal være i. Det blir gjort en grundigere vurdering av målefeilen til trykksensoren i kapittel 7.4 med grunnlag fra testrapportene i vedlegg B.5 og B.7. Trykksensoren er ellers relativt billig, og eneste ulempe er at den må støpes inn i hylsen for å bli vanntett.

Støping av *MS5803-05BA*

Trykksensoren må, som tidligere nevnt, støpes for å bli vanntett. Siden vi bruker samme sensor som årets ROV, valgte vi å bruke sensorkretsen som er utviklet av Jørgen Hemnes Johannessen fra sensorgruppa. Denne er skrevet mer om i kapittel 5.4.

For at støpingen skal fungere som ønsket, må den ikke bare være vanntett, men også kunne utsettes for et bestemt trykk uten å bli ødelagt. Det ble diskutert to ulike løsninger. Den ene løsningen var et støpemiddel som kom i form av gele når det stivnet [30]. Dette ble testet på en trykksensor i en hylse, men støpemiddelet var mykere enn vi først så for oss. Med tanke på at det skal tåle et bestemt trykk, valgte vi å teste ut alternativ nummer to. Det andre alternativet var å bruke epoxy-lim, og den ferdigstøpte sensoren er vist i figur 29.



Figur 29: Trykksensoren støpt inn i hylsen med epoxy-lim.

Epoxy-lim er et to-komponents lim som gjennomgår en kjemisk reaksjon, og dermed går fra flytende form, til gele, til fast form. Det ble brukt en sprøyte for å dytte limet helt ned i hylsen. Deretter ble sensorkretsen dyttet på plass, og til slutt ble det fylt på med lim over sensorkretsen. Her var det viktig å ikke få lim på toppen av trykksensoren, da dette ville ødelagt sensoren.

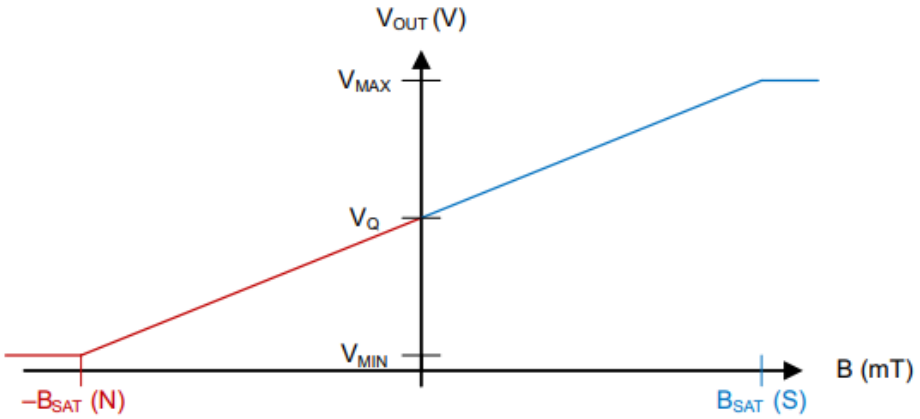
4.3 Deteksjon av sluppet flyter

For å detektere at flyteren er sluppet, og skal begynne med en profil, ble det besluttet å bruke en Hall effect-sensor. En Hall effect-sensor måler styrken av magnetfelt. ROV-en skal ha en påmontert magnet, som kommer i nærheten av sensoren når ROV-en holder flyteren. Signalet fra sensoren skal altså kun brukes som et av/på signal, og derfor er ikke nøyaktighet av målingene kritisk her. Det er likevel viktig at signalet er stabilt nok til å ikke gi falske av/på signaler. Hall effect-sensoren skal plasseres på innsiden av flyteren, og blir dermed liggende i nærheten av vakuumpumpa. Ettersom vakuumpumpa får et roterende magnetisk felt rundt seg når den kjører, er det ønskelig å verifisere om sensoren kan brukes til vårt formål eller ikke.

4.3.1 Måleprinsipp

I en Hall effect-sensor er det en tynn stripe med metall, som blir påført en strøm [44]. Når det kommer et magnetfelt vinkelrett på retningen til strømmen, vil det oppstå en differanse i spenningen på de to endene av metallstripen. Det er denne differansen som blir målt og sendt ut fra utgangen av sensoren som et analogt signal. Denne differansen er proporsjonal med styrken på magnetfeltet.

Til flyteren, valgte vi den analoge Hall effect-sensoren *DRV5053*, med datablad vist i [16]. Figur 30 viser karakteristikken over måleområdet på sensoren.



Figur 30: Utgangsspenningen med nord- og sørpolen av en magnet mot sensoren. Hentet fra [16].

Hall effect-sensoren drives med en inngangsspenning på 3.3 V, og gir ut et analogt spenningssignal fra 0.2 V til 1.8 V. En utgangsspenning på 1 V tilsvarer at den ikke måler et magnetfelt (0 mT). Spenninger over eller under 1 V, tilsvarer at sensoren måler et magnetfelt med henholdsvis sør- eller nordpolen nære sensoren, som illustrert i figur 30.

4.3.2 Støy fra vakuumpumpe

Det er utført en test for å undersøke hvor mye pumpa støyer, i form av magnetisk felt, og om det vil være mulig å detektere magneten, montert på ROV-en, til tross for den eventuelle støyen. Testrapporten er vist i vedlegg B.6.

I testen ble det funnet at når pumpa ikke kjører, er det snakk om et så lite magnetfelt at Hall effect-sensoren ikke plukker det opp, med mindre den er i fysisk kontakt med pumpa. Når pumpa kjører, ble det derimot målt betydelig mer støy, og dette varierer over hele måleområdet til sensoren. Støyen plukkes opp, selv om sensoren er 40 cm fra pumpa.

Videre i testen, ble magnetfeltet fra magneten målt samtidig som pumpa kjørte. Det ble funnet at støyen da varierte rundt en DC-komponent, altså magneten. Det ble derfor konkludert med at det skal bli tatt i bruk et digitalt filter, samt å legge inn en forrigling, som forhindrer systemet i å lete etter et magnetfelt mens pumpa kjører. Dette forklares nærmere i kapittel 5.3.

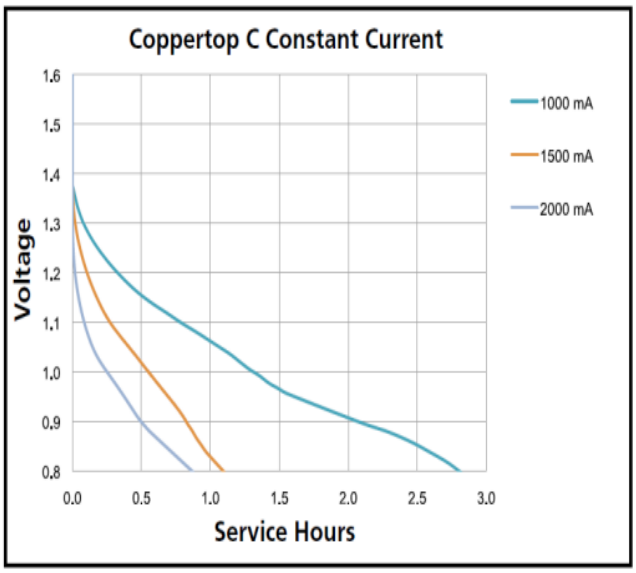
4.4 Batterivurdering

Fjorårets gruppe som designet Mikro-ROV-en, har utført en batterivurdering for denne [27]. I dette delkapittelet tas det utgangspunkt i deres batterivurdering. I følge konkurransens regler er det et fåtall av batterier som er lovlige å bruke. Det er gitt at man kun har lov til å bruke ikke-oppladbare, alkaliske batterier. I tabell 7 er de lovlige alkaliske batteriene listet opp, sammen med batterienes spesifikasjoner. Strømleveringsevnen [mAh] tilsier hvor mange ampere batteriet kan levere i løpet av en time.

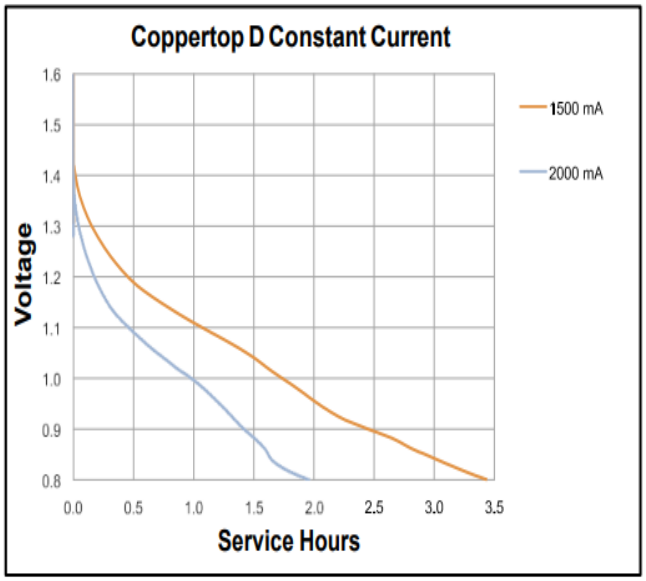
Batteritype	**Spenning [V]**	**Strømleveringsevne [mAh]**	**Max test datablad[mA]**	**Vekt [g]**
AAA	1.5	≈450	500	11
AA	1.5	≈1000	1000	24
A23	12.0	45	15	8
C	1.5	≈1600	2000	69
D	1.5	≈3800	2000	139
9V	9	≈175	250	46

Tabell 7: Batterispesifikasjoner. Hentet fra [27], og oppdatert.

Etter å ha sett gjennom databladene til de ulike batteriene, er det type C og D som skiller seg ut ved at de er testet med høyest strømtrekk. På samme måte som mikro-ROV gruppa har gjort, i [27], sammenlignes spenningsfallet til disse to batteriene i figur 31 og 32.



Figur 31: Utladningskurve til C-type batteri, hentet fra [9].



Figur 32: Utladningskurve til D-type batteri, hentet fra [8].

Her ser man at C-typen har et hurtigere spenningsfall enn D-typen. Dette tilsier at D-typen har høyere kapasitet med samme strømtrekk. Det konkluderes derfor med at D-typen er mest aktuell å bruke. Videre bestemmes det hvor mange som skal kobles i serie og parallell, for å oppnå et ønskelig resultat. Dette gjøres ved å estimere maksimalt strømtrekk til det hele sammensatte systemet. En oversikt over komponentene som trekker strøm, er vist i tabell 8. Merk at her er kun hovedkomponenter tatt med, altså komponenter som har et betydelig strømtrekk. Spenningsregulatorene, som også trekker strøm, har et strømtrekk på mindre enn 1 mA, og det regnes her som neglisjerbart.

Komponent	**Maximalt strømtrekk [mA]**	**Tilførselsspenning [V]**	**Effektforbruk [W]**
Boost-regulator (U1)			
Vakuumpumpe	1000	12	12
3-ports solenoidventil	230	12	2.76
Lineærregulator (U4)			
ARM MINI M4	240	3.3	0.792
Hall effect-sensor (SENS1)	2.7	3.3	$8.9{\cdot}10^{-3}$
Trykksensor (SENS2)	1.4	3.3	$4.6{\cdot}10^{-3}$
Solid state rele (U2 og U3)	2 x 30	1.65	0.099

Tabell 8: Oversikt over maksimalt strømtrekk.

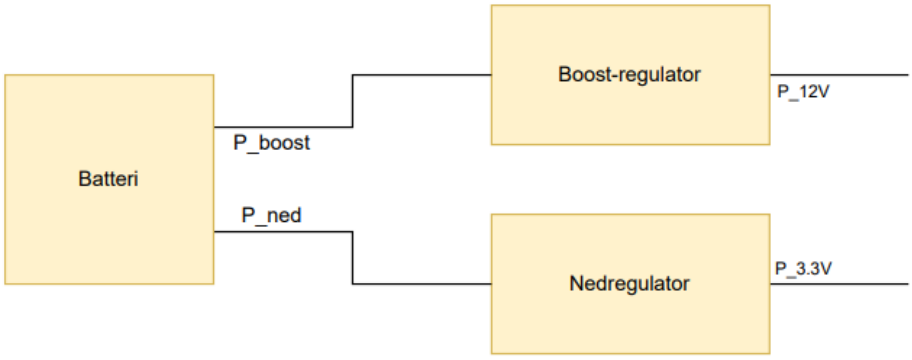
Som vist i tabell 8, trenger komponentene både 12 V og 3.3 V spenningstilførsel. Her må man finne en løsning for å kunne forsyne pumpe og solenoidventil med 12 V, samt de resterende komponentene med 3.3 V. En måte dette kan gjøres, er ved å koble nok D-type batterier i serie, til å få en spenning

på 12 V, og deretter bruke en nedregulator, til å regulere spenningen ned til 3.3 V. Ulempen er at etterhvert, som batteriet brukes, vil spenningen på batteriet reduseres, og 12 V tilførsel vil dermed ikke være opprettholdbart. Det vil i tillegg kreve $\frac{12\ V}{1.5\ V} = 8$ batterier i serie for å få til.

Et bedre alternativ er å koble fire D-type batterier i serie. Da får man $1.5\ V \cdot 4 = 6\ V$. Videre kan man bruke en boost-regulator til å regulere spenningen opp, fra 6 V til 12 V, samt en nedregulator i parallell til å regulere spenningen ned, fra 6 V til 3.3 V. Fordelen er at man vil kunne opprettholde en stabil utgangsspenning på 12 V, også etterhvert som spenningen på batteriet reduseres. Videre skal dette alternativet grundigere analyseres med hensyn til strømtrekk og batterikapasitet.

4.4.1 D-type batteri med spenningsregulering

Som nevnt innledningsvis, skal det vurderes å bruke en boost-regulator. Denne skal regulere spenningen fra batteriet opp til 12 V, for å forsyne både pumpe og ventil. I parallell med denne skal man se på bruk av en nedregulator. Denne skal regulere spenningen fra batteriet ned til 3.3 V, for å forsyne mikrokontroller, sensorer og resten av kretsen. En oversikt, i form av et blokkskjema av koblingen, er vist i figur 33.



Figur 33: Blokkskjema over batteri og kraftforsyning.

Dersom fire batterier seriekobles, får man $V_{batt,max} = 1.5\ V \cdot 4 = 6\ V$ fra fulladet batteri, og $V_{batt,min} = 0.9\ V \cdot 4 = 3.6\ V$ fra utladet batteri. Dette er dersom man går ut fra at et 1.5 V D-type batteri anses som "tomt" når det er utladet til 0.9 V. Det vil si at spenningen på inngangen til spenningsregulatorene, vil variere mellom 3.6 V og 6 V. For å beregne maksimalt strømtrekk fra batteriet, må man finne maksimalt effektforbruk på utgangen av boost- og nedregulatoren. Dette er vist i ligning 32 og 33. Her er $P_{12\ V}$, $I_{tot,12\ V}$ og $V_{12\ V}$ henholdsvis den totale effekten, det maksimale strømtrekket og spenningen på utgangen av boost-regulatoren.

$$P_{12\ V} = I_{tot,12\ V} \cdot V_{12\ V} = (1 + 0.23) \cdot 12 = 14.76\ W \approx 15\ W \tag{32}$$

$$P_{3.3\ V} = I_{tot,3.3\ V} \cdot V_{3.3\ V} = \frac{(240 + 2.7 + 1.4)}{1000} \cdot 3.3 + 0.099 = 0.9\ W \tag{33}$$

En boost-regulator som kan brukes, er *MEZD4150A-B*, og ettersom denne er en svitsjet regulator, har den en virkningsgrad (η), altså et effekttap. *MEZD4150A-B* har en virkningsgrad på 95%. For å vite hvor mye effekt som blir trukket fra batteriet, må effekttapet legges til, som vist i ligning 34. P_{boost} er effekten som blir trukket fra batteriet til boost-regulatoren, og η er virkningsgraden til boost-regulatoren i prosent.

$$\begin{aligned} P_{boost} &= P_{12\ V} + \frac{(100-\eta)}{100} \cdot P_{12\ V} \\ &= 15 + \frac{(100-95)}{100} \cdot 15 \\ &= 15.75\ W \end{aligned} \tag{34}$$

Nedregulatoren som vurderes, er en lineærregulator. Når denne regulerer ned batterispenningen fra 6 V til 3.3 V, får man et effekttap. Lineærregulatoren som er valgt, forklares nærmere i kapittel 5.1.2. Virkningsgraden denne har, regnes ut som vist i ligning 35. Videre kan effekttapet i lineærregulatoren regnes ut, som vist i ligning 36, ved bruk av virkningsgraden.

$$\begin{aligned} \eta &= \frac{V_{3.3}}{V_{batt,maks}} \cdot 100 \\ &= \frac{3.3}{6} \cdot 100 \\ &= 55\% \end{aligned} \tag{35}$$

$$\begin{aligned} P_{ned} &= P_{3.3\ V} + \frac{(100-\eta)}{100} \cdot P_{3.3\ V} \\ &= 0.9 + \frac{(100-55)}{100} \cdot 0.9 \\ &= 1.30\ W \end{aligned} \tag{36}$$

Dersom man bruker en boost-regulator, med en virkningsgrad på 95%, og en lineærregulator, vil den totale effekten, trukket fra batteriet, være summen av effekten på inngangen til både boost- og lineærregulatoren, vist i ligning 37.

$$P_{TOT} = P_{boost} + P_{ned} = 15.75\ W + 1.30\ W = 17.05\ W \approx 17\ W \tag{37}$$

Det maksimale strømtrekket fra batteriet er, ifølge konkurransemanualen, begrenset til å være maksimalt 6 A. Det betyr at vi ønsker å forsikre oss mot at systemet skal kunne kreve et høyere strømtrekk enn 6 A. Etterhvert som batteriet tappes, vil spenningen i batteriet reduseres. Når dette skjer, vil strømtrekket øke, for å kompensere for den reduserte spenningen, for å opprettholde den totale effekten som blir brukt. Med et maksimalt effektforbruk på systemet, regner man ut øvre og nedre strømtrekk når batteriet er fulladet ($V_{batt,max}$), og utladet ($V_{batt,min}$), i ligning 38 og 39.

$$\begin{aligned} I_{min} &= \frac{P_{TOT}}{V_{batt,max}} \\ &= \frac{17\ W}{6\ V} \\ &= 2.83\ A \end{aligned} \tag{38}$$

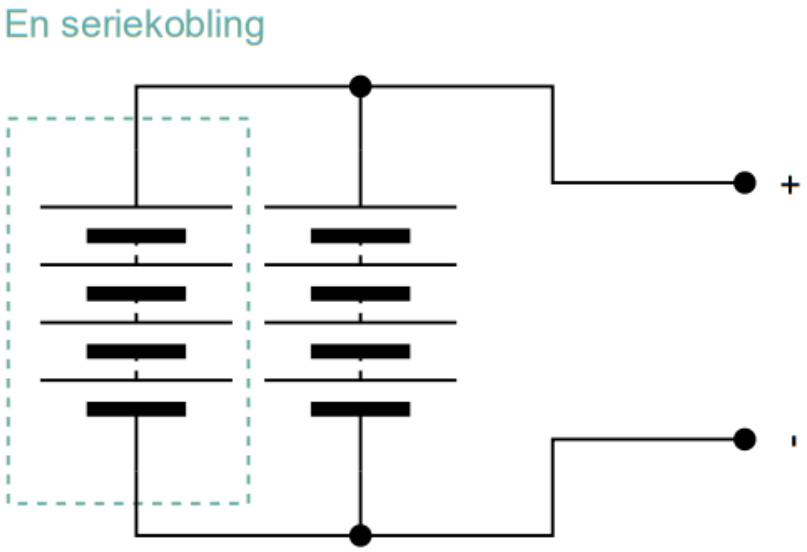
$$\begin{aligned} I_{max} &= \frac{P_{TOT}}{V_{batt,min}} \\ &= \frac{17\ W}{3.6\ V} \\ &= 4.72\ A \end{aligned} \tag{39}$$

Dette er, i praksis, ikke et reelt kontinuerlig strømtrekk fra batteriet, ettersom pumpa kun skal pumpe når flyteren skal flyte opp, og ventilen kun skal trekke strøm når flyteren skal synke. For å være på den sikre siden, dimensjonerer man kraftforsyningsdelen av systemet til å kunne forsyne alt på en gang.

Batterikapasiteten som er nødvendig kan nå bestemmes, ettersom man vet det totale effektforbruket og strømtrekket til systemet. Batterikapasiteten, i milliampere timer (mAh), som trengs for å kunne kjøre systemet i 15 minutter (som er tiden vi har på å utføre oppgaven i konkurransen), kan regnes ut som i ligning 40. Her er tiden t gitt i timer [h], hvor 15 minutter tilsvarer $t = \frac{15}{60} = 0.25h$. For å være på den sikre siden, tar man utgangspunkt i kontinuerlig maksimalt strømtrekk.

$$I \cdot t = 4720 \cdot 0.25\ h = 1180\ mAh \tag{40}$$

For å slippe å bytte batteri for hvert 15. minutt, ved kjøring av flyteren, er det ønskelig å ha høyere batterikapasitet enn det som kreves for å kjøre i 15 minutter. Ettersom strømleveringsevnen til D-typen er testet til å være omtrent 3800 mAh, med et strømtrekk på 2000 mA, kan det hende det er nok å kun koble fire batterier i serie, som vist i grønn stiplet firkant i figur 34.



Figur 34: Koblingsskjema over batterioppsettet til flyteren.

I testrapport B.1, er det utført en utladningstest på en seriekobling, med fire alkaliske D-type batterier, med pumpa som strømtrekk. Det gjennomsnittlige strømtrekket var på 800 mA, noe som er mer tilnærmet et reellt strømtrekk ved normal drift. I testrapporten, i vedlegg B.1, kan man se at selv med kun en seriekobling av batterier, holder spenningen seg over 5 V i over 30 minutter. Man kan derfor konkludere med at et batterioppsett bestående av to parallellkoblinger, med fire batterier seriekoblet, vil være tilfredsstillende til dette bruket. Ved å parallellkoble to seriekoblinger av fire alkaliske D-type batterier, som vist i figur 34 oppnår man et halvvert strømtrekk på hver side av parallellkoblingen. Batterikapasiteten vil dermed dobles.

5 Maskinvare: Kretskortdesign

Innhold

Kretskortet som er designet, har som hensikt å fungere som et grensesnitt mellom mikrokontrolleren, og de ulike delene som skal styres av denne. Kretskortet er derfor designet med spenningsregulatorer, for å forsyne pumpe, ventil, sensorer og mikrokontroller med riktig spenning. Hele skjemategningen er lagt ved i vedlegg A, og de ulike delene av dette skal gjennomgås her. Utlegget og produksjon skal også gjennomgås i dette kapittelet. For å designe kretskortet, er programmet *Altium Designer* brukt. Gerber-filene kan finnes i [2].

5.1 Kraftforsyning

Kraftforsyningen er delt i to deler. Systemets komponenter krever både 12 V og 3.3 V, som tidligere nevnt i kapittel 4.4. Man skal først se på komponenten som skal kunne forsyne systemet med 12 V, (boost-regulatoren U4), og deretter komponenten som forsyner den delen av systemet som trenger 3.3 V, (nedregulatoren U3).

5.1.1 Svitsjet boost-regulator

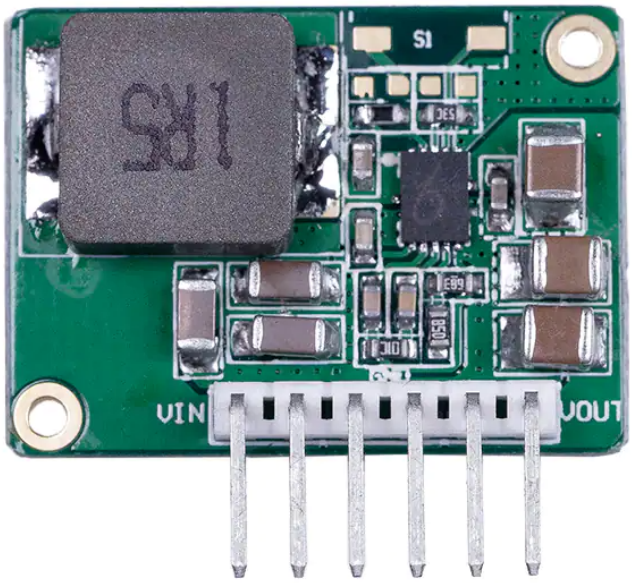
I kapittel 4.4, så man på hvor mye effekt boost-regulatoren trenger på inngangen, for å kunne forsyne systemet ved maksimalt strømtrekk. Man vil derfor se på hva den maksimale strømmen er, som kommer til å gå gjennom boost-regulatoren, ved minimal inngangsspenning fra batteriet. Dette er vist i ligning 41.

$$I_{boost,max} = \frac{P_{boost}}{V_{batt,min}} = \frac{16\ W}{3,6\ V} = 4.44\ A \tag{41}$$

Det vil si at man trenger en boost-regulatorkrets som tåler det maksimale strømtrekket på 4.44 A. Det ble sett på tre ulike alternativer til boost-regulator. Disse er listet opp nedenfor.

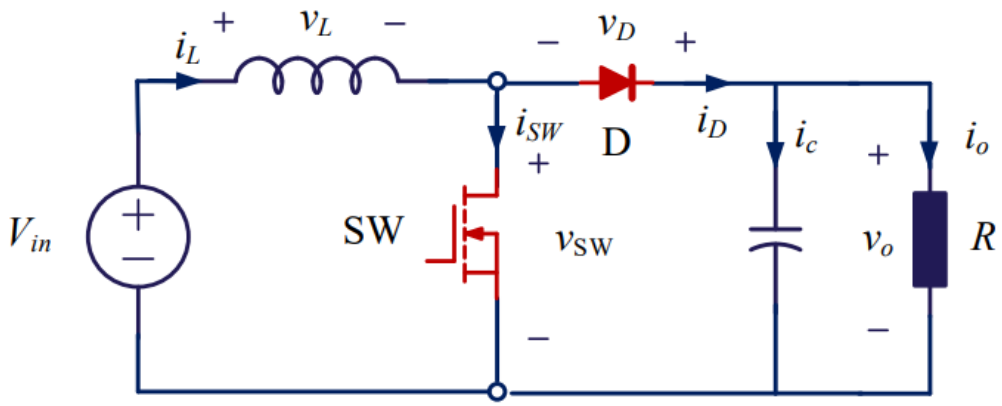
- *MEZD4150A-B*
- *LM2623*
- *LT1170*

Etter litt diskusjon, ble det bestemt å bruke *MEZD4150A-B*, ettersom *LM2623* ikke tåler høy nok strøm, og *LT1170* er mindre effektiv, i tillegg til å ha lavere svitsjefrekvens. *MEZD4150A-B* er vist i figur 35. Fordelen med denne, er at den er en ferdig designet regulatorkrets, som består av en svitsjet boost-regulator og er koblet med motstander, kondensatorer og spole. Man skal først se på hvordan en generell boost-regulator fungerer, før man går mer spesifikt inn på den regulatoren som skal brukes.



Figur 35: *Boost*-regulator *MEZD4150A-B*, hentet fra [38].

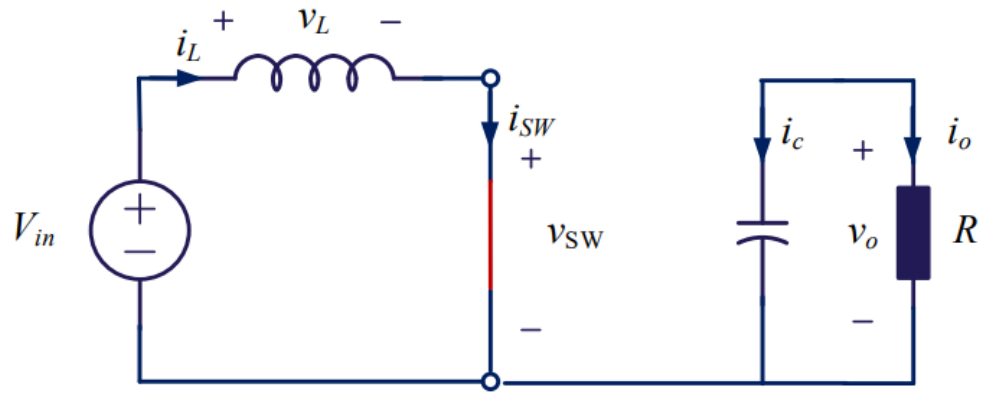
En boost-regulator er bygget opp, som vist i figur 36, av en spole, en transistor, en diode og en kondensator.



Figur 36: Kretsen for en boost-regulator. Hentet fra [49].

Når transistoren blir skrudd av og på, er regulatoren i to forskjellige tilstander [49]. Gjennom den følgende analysen, går man ut fra at transistoren blir skrudd på i tidspunkt $t = 0$, skrudd av i $t = DT$, og skrudd på igjen i $t = T$. Analysen ser på kretsen i stasjonærtilstand.

Når transistoren blir skrudd på, blir kretsen seende ut som i figur 37.

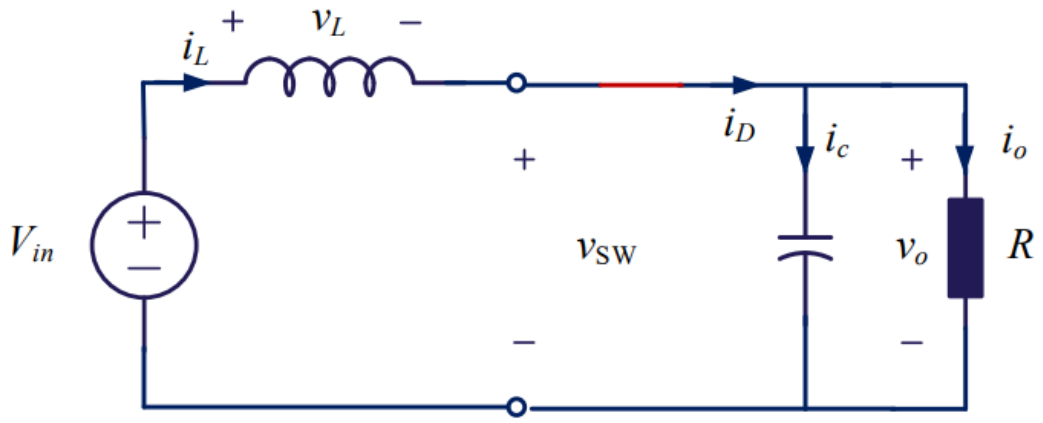


Figur 37: Boost-kretsen når transistoren er på. Hentet fra [49].

Da vil spenningen over spolen være den samme som inngangsspenningen, $v_L = v_{in}$. Når denne er konstant, vil dermed strømmen gjennom spolen øke lineært. Dette er vist i formel 42, ved å bruke KVL og standard kretsanalyse. Denne strømøkningen skjer altså fra $0 < t < DT$.

$$
\begin{aligned}
&- v_{in} + v_L = 0 \\
&\Rightarrow v_{in} = v_L = L\frac{di_L}{dt} \\
&\Rightarrow i_L(t) = \int_0^t \frac{v_{in}}{L} dt
\end{aligned}
\tag{42}
$$

Når transistoren blir skrudd av, ser kretsen ut som i figur 38.



Figur 38: Boost-kretsen når transistoren er av. Hentet fra [49].

Ved å bruke KVL, finner man her at $v_L = v_{in} - v_o$. Siden utgangsspenningen er større enn inngangsspenningen, er altså spenningen over spolen negativ. Strømmen gjennom spolen vil derfor synke lineært. Dette gjelder for tidsperioden $DT < t < T$.

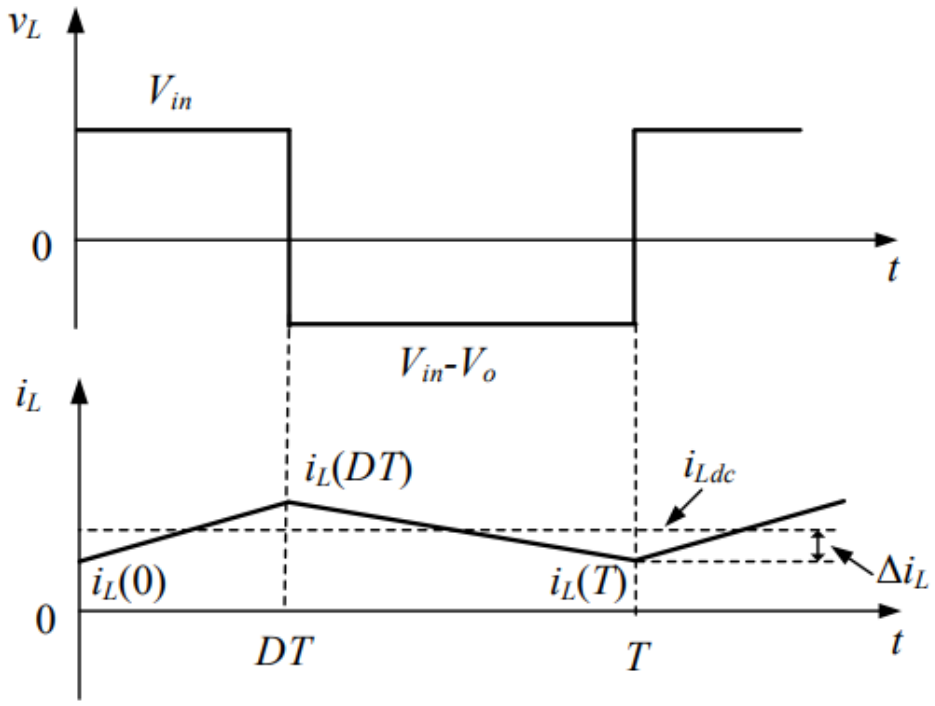
Strømmen gjennom en spole er definert som i ligning 43, hvor $i_L(0^+)$ er strømmen gjennom spolen rett etter $t = 0$.

$$
i_L(t) = \frac{1}{L}\int_0^t v_L dt + i_L(0^+) \tag{43}
$$

For at strømmen ikke skal stige eller minke i det uendelige må $i_L(T) = i_L(0^+)$. Dette kan settes inn i ligning 43, for å utlede utgangsspenningen som i ligning 44.

$$
\begin{aligned}
&i_L(T) = \frac{1}{L}\int_0^T v_L dt + i_L(0^+) \\
&\frac{1}{L}\int_0^T v_L dt = i_L(T) - i_L(0^+) \\
&\frac{1}{L}\int_0^T v_L dt = 0 \\
&\Rightarrow \int_0^T v_L dt = 0 \\
&\Rightarrow \int_0^{DT} v_{in} dt + \int_{DT}^T v_{in} - v_o dt = 0 \\
&\Rightarrow v_{in}DT + v_{in}T - v_oT - v_{in}DT + v_oDT = 0 \\
&\Rightarrow v_{in} = v_o\frac{T - DT}{T} \\
&\Rightarrow v_o = v_{in}\frac{T}{T - DT}
\end{aligned}
\tag{44}
$$

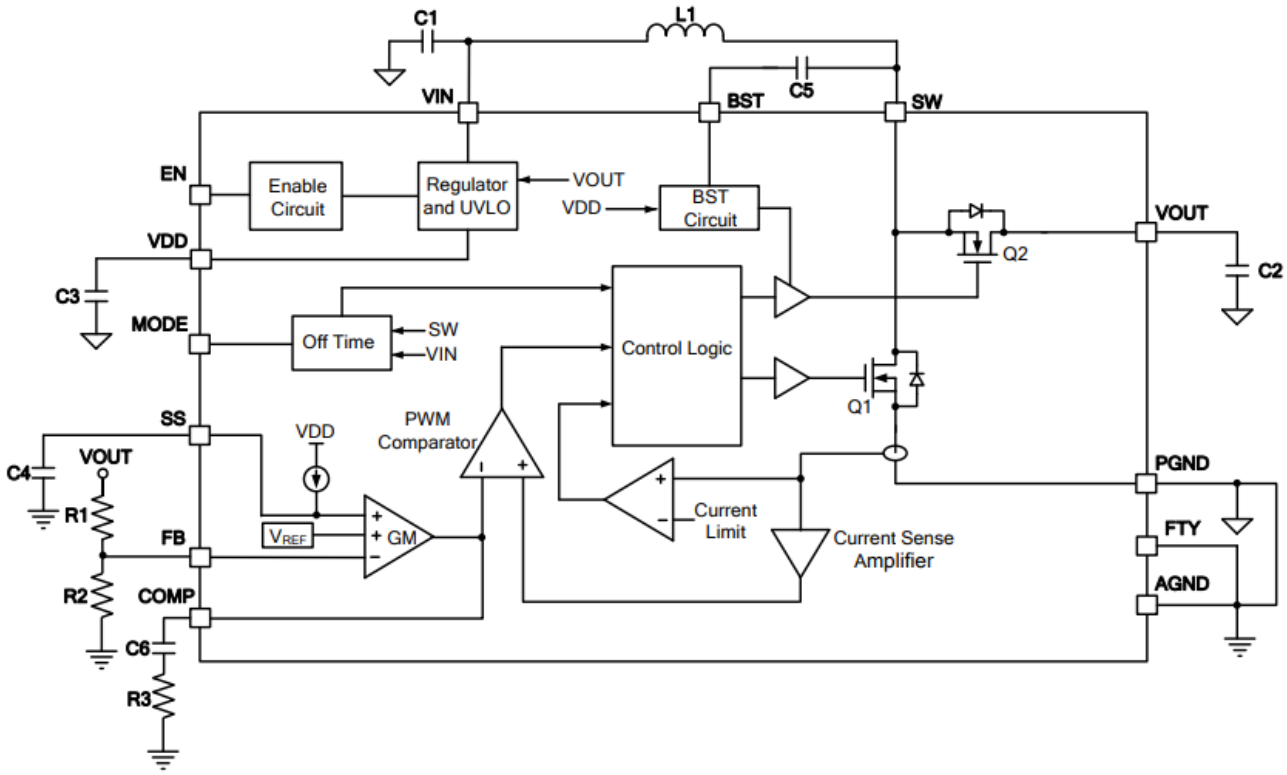
Her er altså DT tiden transistoren blir skrudd av og spenningen over spolen er positiv, mens T er tiden den blir skrudd på igjen. Dette er illustrert i tidsdiagrammet i figur 39.



Figur 39: Tidsdiagram som viser spolespenningen og -strømmen over tid. Hentet fra [49].

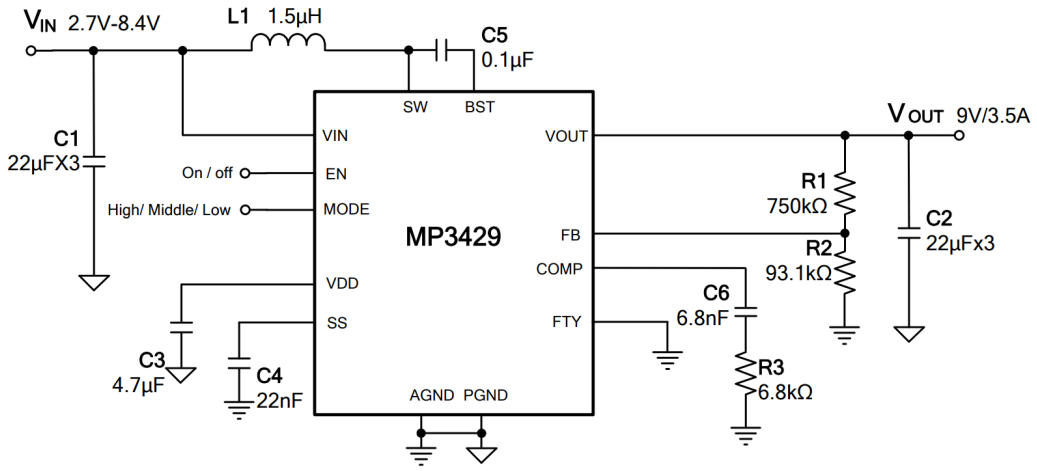
Fra formel 44, ser man at dersom man for eksempel setter DT til å være halvparten av T blir $v_o = 2 \cdot v_{in}$, og periodetiden kan derfor brukes til å justere utgangsspenningen som man ønsker.

På regulatorkretskortet, *MEZD4150A-B*, er det en påmontert boost-regulator, *MP3429*. Blokkdiagrammet til denne, med eksterne komponenter, er vist i figur 40. Her kan man se spolen, transistoren, dioden og kondensatoren som er beskrevet over, i tillegg til en del logikk som er med på å regulere spenningen.

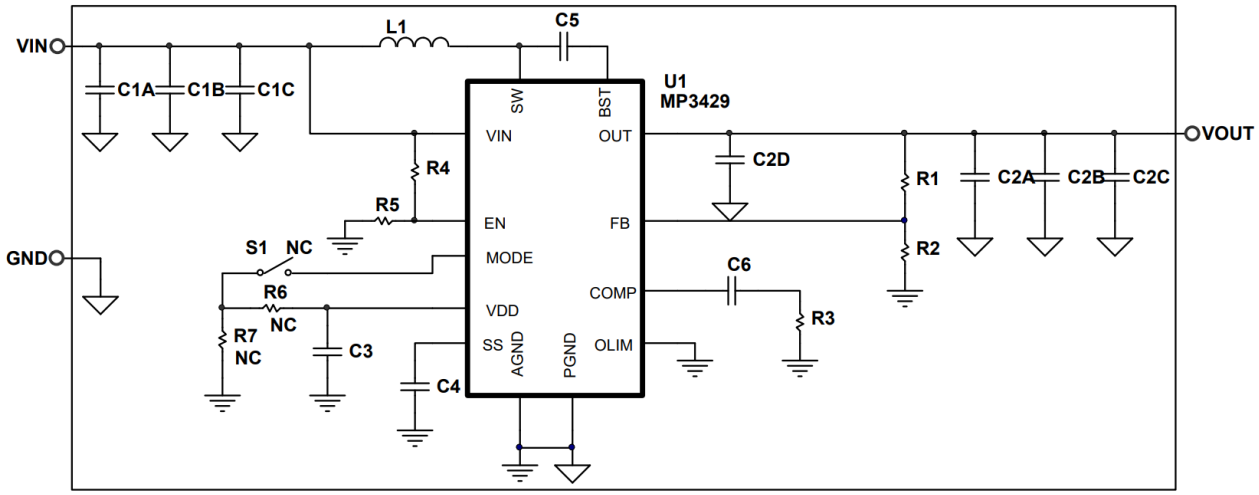


Figur 40: Blokkdiagram av regulatoren *MP3429*. Hentet fra [37].

Anbefalt oppsett for boost-regulatoren er angitt i databladet [37], og vist i figur 41. Oppsettet på den ferdige kretsen, *MEZD4150A-B*, er vist i figur 42, og man ser at disse oppsettene stemmer overens med hverandre.



Figur 41: Typisk oppsett for *MP3429*, hentet fra [37].



Figur 42: Fullstending skjema over *boost*-regulatorkretsen, hentet fra [38].

I figur 41 og 42 kan man se på inngangen, med navn *MODE*, at det er en åpen bryter. Denne er der for å kunne velge mellom ulike moder på boost-regulatoren. Det er tre ulike moder å velge mellom. Disse er listet opp nedenfor.

- *Forced Continious Conduction mode* (FCCM)
- *Pulse-Skip Mode* (PSM)
- *Ultrasonic Mode* (USM)

For å bruke FCCM, må mode-inngangen settes til inngangsspenningen, VIN. Her er det lavere rippel på VOUT, enn det er i PSM. For å bruke PSM, må mode-inngangen flyte. Her er det høyere effektivitet, men som sagt høyere rippel, og frekvensen kan bli lav og produsere støy. USM brukes ved å sette mode-inngangen til $0.2 < V_{MODE} < 0.7\ V$. Her unngår man den støyen som ofte oppstår i PSM [37].

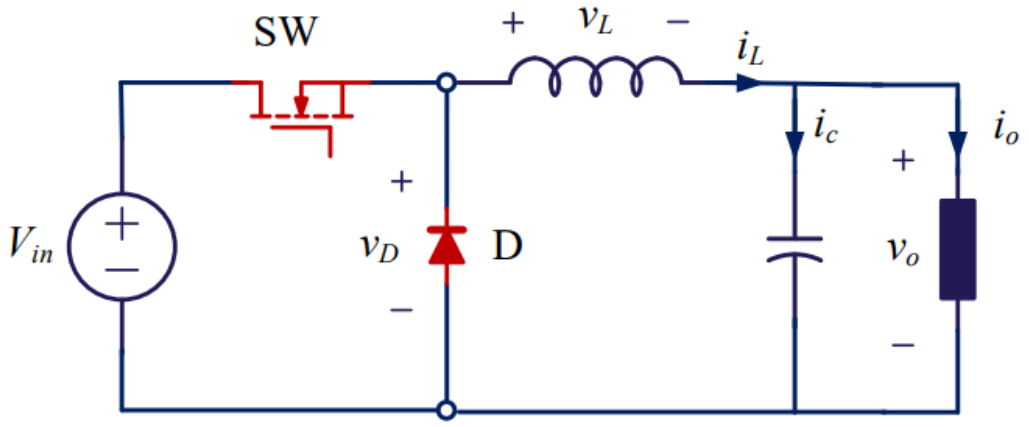
S1 er åpen, og dermed er regulatoren satt i PSM-modus. Her er det høyest effektivitet, og kan som sagt bli litt mer støy. Dette blir diskutert mer i kapittel 5.7.

5.1.2 Nedregulator

En regulator som regulerer ned spenningen kan være bygget opp på to ulike måter, enten svitsjet eller lineær. Her skal begge typene gjennomgås, før man ser på den som brukes i systemet vårt.

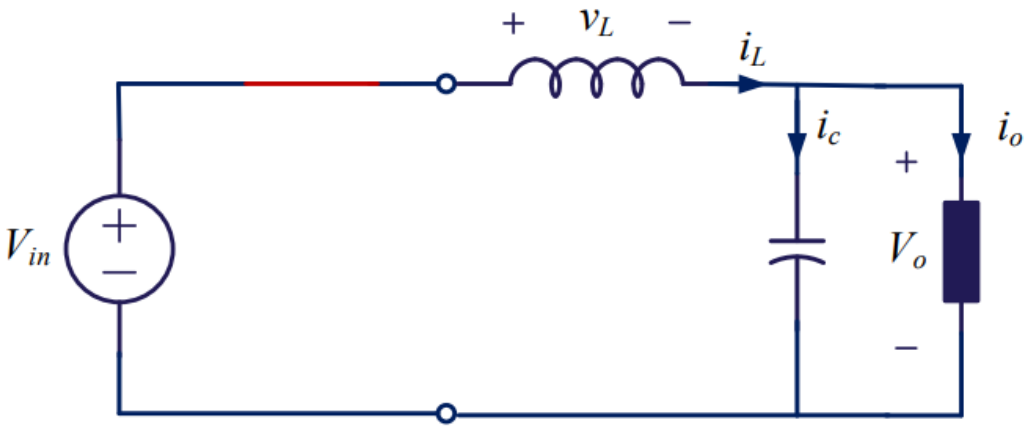
Svitsjet buck-regulator

En svitsjet buck-regulator er, som en boost-regulator, bygget opp av en spole, diode, kondensator og en transistor. Komponentene er koblet sammen, som vist i figur 43. Her vil det også være to ulike tilstander, når transistoren er på, og når den er av.



Figur 43: Kretsen for en buck-regulator. Hentet fra [49]

Når transistoren er på, vil dioden være av og kretsen blir seende ut som i figur 44.



Figur 44: Buck-kretsen når transistoren er på. Hentet fra [49].

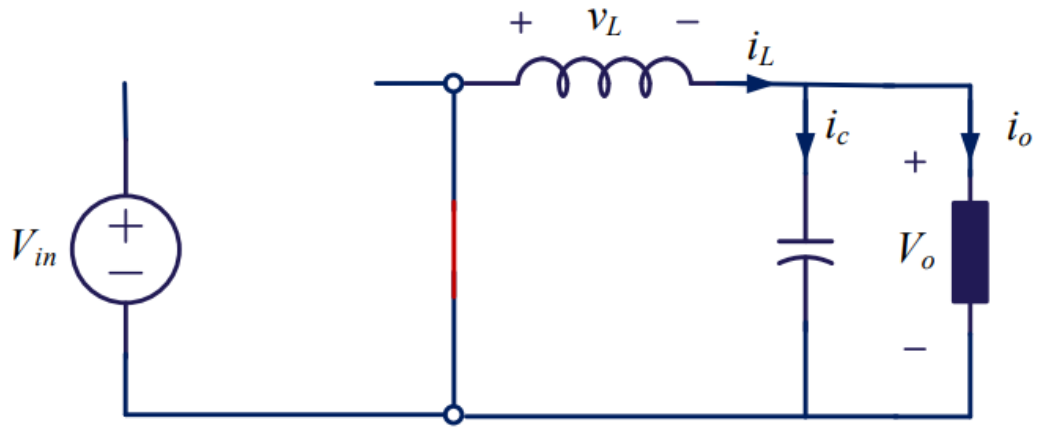
Ved å bruke KVL kan spenningen over spolen skrives som i ligning 45, og strømmen gjennom spolen blir som i 46.

$$\begin{aligned} v_{in} &= v_L + v_o \\ \Rightarrow v_L &= v_{in} - v_o \\ &= L\frac{di_L}{dt} \end{aligned} \tag{45}$$

$$i_L(t) = \int_0^t \frac{v_{in} - v_o}{L} dt \tag{46}$$

Siden både v_{in} og v_o er konstante, og $v_o < v_{in}$, blir det en lineær økning i strøm når transistoren er på.

Når transistoren skrus av blir kretsen seende ut som i figur 45.



Figur 45: Buck-kretsen når transistoren er av. Hentet fra [49].

Ved å bruke KVL her, finner man at spenningen over spolen blir som i ligning 47, og strømmen gjennom den blir som i 48.

$$v_L + v_o = 0$$
$$\Rightarrow v_L = -v_o = L\frac{di_L}{dt} \tag{47}$$

$$\frac{di_L}{dt} = \int_0^t \frac{-v_o}{L} dt \tag{48}$$

Her kan man se at så lenge utgangsspenningen er konstant vil strømmen gjennom spolen være lineært avtagende.

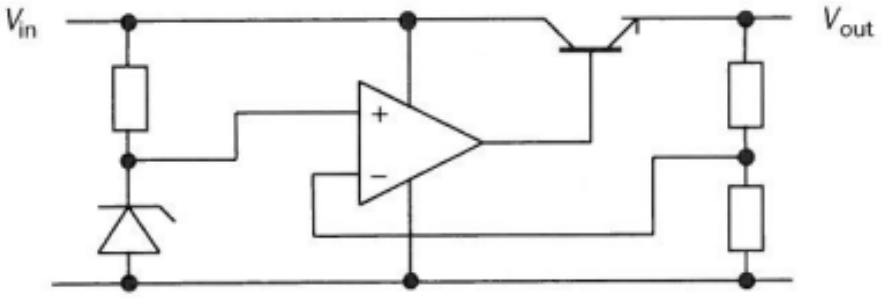
For at strømmen over spolen ikke skal stige eller minke i det uendelige, må også her (som for boost-regulatoren) $i_L(T) = i_L(0^+)$. DT er tidspunktet transistoren blir skrudd av, og T er tidspunktet den blir skrudd på igjen. Ligning 43 kan derfor brukes igjen og løses for utgangsspenningen, som i ligning 49.

$$\begin{aligned}
&\frac{1}{L}\int_0^T v_L dt = 0 \\
&\Rightarrow \int_0^T v_L dt = 0 \\
&\Rightarrow \int_0^{DT} v_{in} - v_o dt + \int_{DT}^T -v_o dt = 0 \\
&\Rightarrow v_{in}DT - v_oDT - v_oT + v_oDT = 0 \\
&\Rightarrow v_{in}DT - v_oT = 0 \\
&\Rightarrow v_o = \frac{v_{in}DT}{T}
\end{aligned} \tag{49}$$

Her ser man at dersom DT settes til halvparten av periodetiden, T, blir $v_o = \frac{1}{2}v_{in}$. Dermed kan periodetiden også her brukes til å sette hva utgangsspenningen skal bli.

Lineær regulator

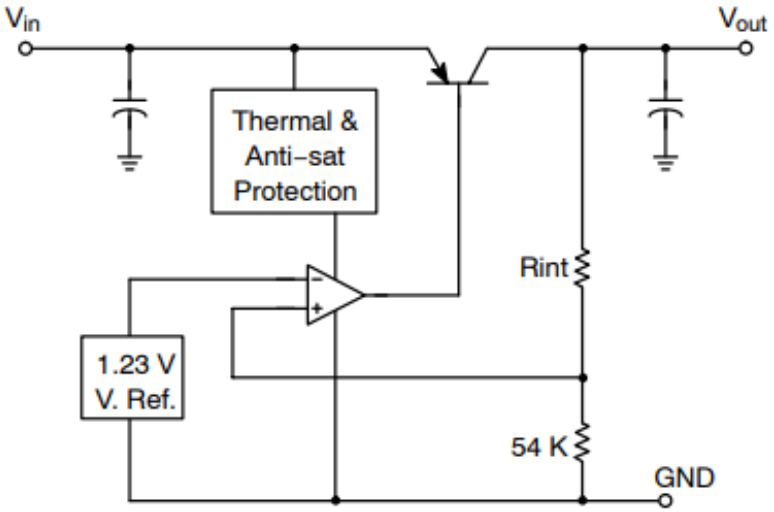
En lineær regulator er bygget opp som i figur 46, med en zenerdiode, operasjonsforsterker og en transistor.



Figur 46: Oppbyggningen av en lineær buck-regulator. Hentet fra [47].

Zenerdioden brukes til å sette en referansespenning inn på operasjonsforsterkeren. Det forsterkede signalet sendes deretter til transistoren. Kretsen fungerer på den måten at dersom V_{out} for eksempel stiger, sendes dette tilbake inn på operasjonsforsterkeren. Operasjonsforsterkeren vil tilstrebe at spenningsdifferansen, mellom inngangene dens, er lik null. Så når V_{out} stiger, vil den kompensere med å redusere spenningen ut til transistoren. Dermed reduseres også utgangsspenningen. Det omvendte vil skje dersom utgangsspenningen blir for lav. På denne måten holder buck-regulatoren spenningen stabil. Det er altså zenerdioden som bestemmer hva utgangsspenningen skal være, ved å sette referansespenningen inn på operasjonsforsterkeren, [47] og [13].

I et prosjekt i forrige semester brukte vi en lineær regulator for å regulere spenningen ned til 3.3 V. Siden vi har brukt denne før, og dermed vet hvordan den fungerer, samt hvilke tilleggskomponenter den trenger, valgte vi å bruke den igjen nå. Den aktuelle regulatoren var også tilgjengelig på laboratoriet på universitetet. I tillegg var det vanskelig å finne gode alternativer som var på lager i butikk og på nett. Denne regulatoren heter *MC33275* og blokkdiagrammet er vist i figur 47.

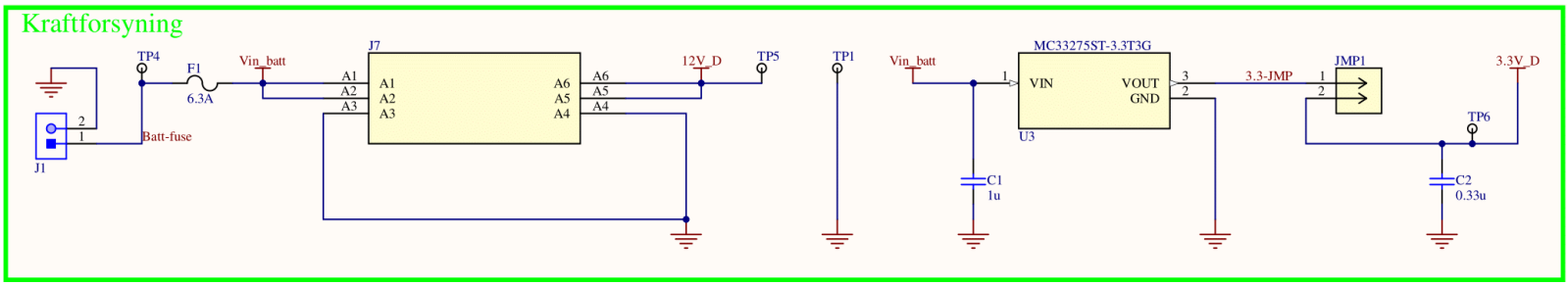


Figur 47: Blokkdiagram av regulatoren *MC33275*. Hentet fra [15].

Man kan se her at oppsettet er likt som i figur 46, med en bestemt referansespenning inn på operasjonsforsterkeren. Forskjellen her er at referansespenningen går inn på minusinngangen. Dette er fordi referansespenningen er mindre enn utgangsspenningen. *MC33275* kan bestilles med ulike pakninger, og ønsket utgangsspenning. Vi bruker den med fotavtrykk SOT-223 og utgangsspenning 3.3 V.

5.1.3 Skjemategning kraftforsyning

I figur 48 er skjemategningen for kraftforsyningen vist. Siden U4 er en ferdig krets, er det her satt inn en plugg (J7) for å koble den til vår krets.



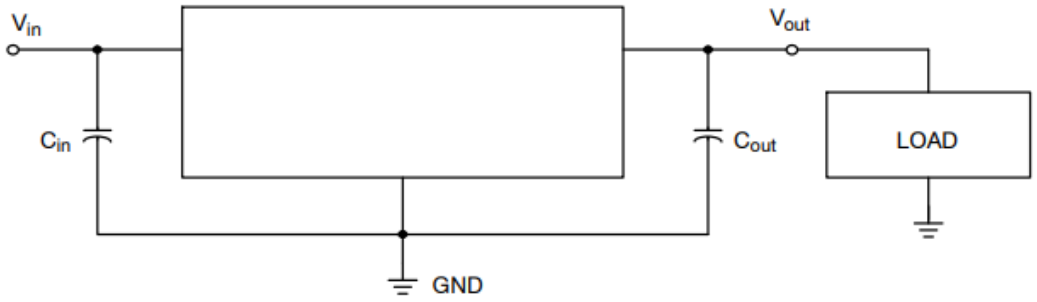
Figur 48: Kraftforsyningen i skjemategningen.

Foran J7, er det lagt til en sikringsholder, F1. I konkurransemanualen er det beskrevet ett krav om å ha en sikring, maksimalt 5 cm fra batteriets positive pol. Etterhvert som flyteren ble bygget, ble det klart at batteriets positive pol ble plassert lenger enn 5 cm fra krestkortet. Dermed ble sikringen på kretskortet byttet ut med en "in line"-sikring, som kobles direkte på lederen fra batteriets positive pol. Holderen vi kjøpte er vist i figur 49.

Figur 49: Sikringsholderen. Bildet er hentet fra [31].

Inni denne ble det plassert en glassikring på 5 A, ettersom det maksimale strømtrekket er regnet ut til å være på omtrent 4.72 A. Det ble det også satt en ledning over F1, slik at det er en direkte kobling.

Anbefalt oppsett for U3 er vist i figur 50. I kretsskjemaet i figur 48, er det brukt samme kondensatoroppsett, med avkoblingskondensatorer på inngang og utgang.



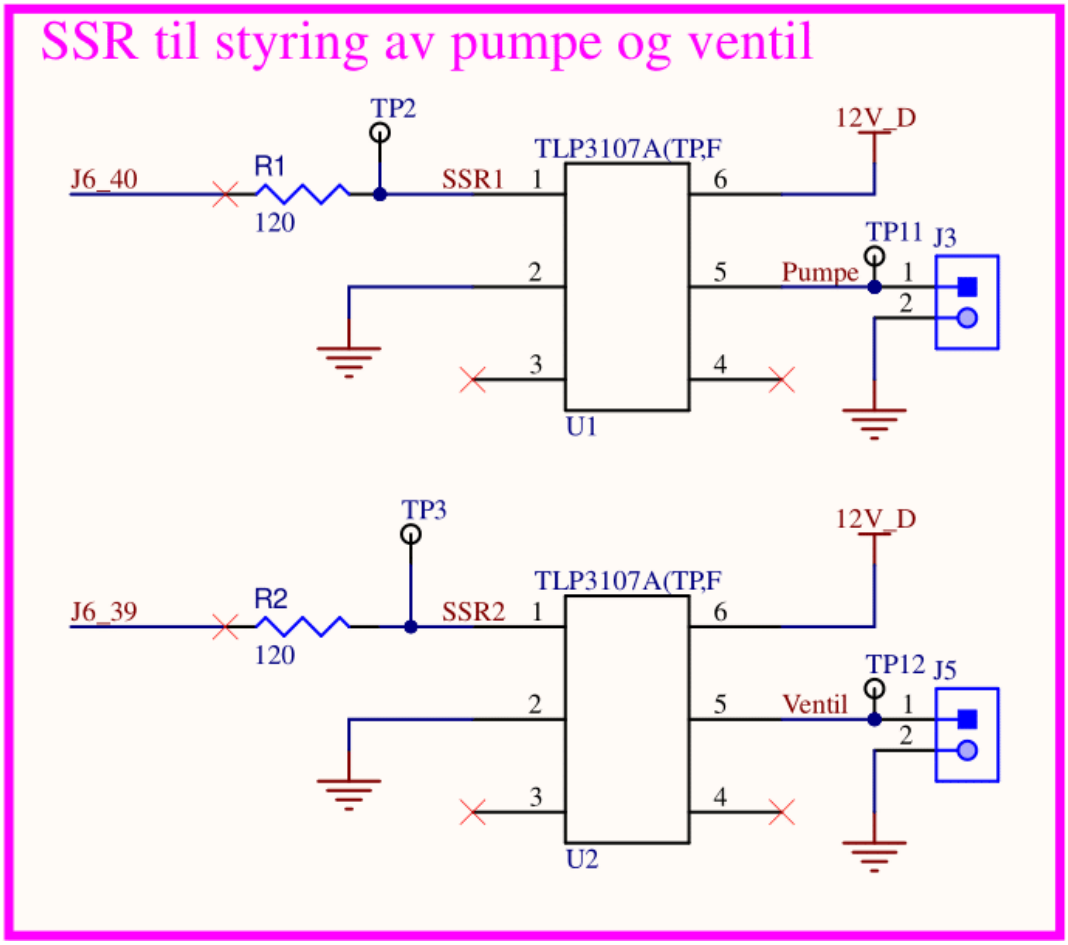
Figur 50: Anbefalt oppsett for buck-regulatoren-

I databladet [15] står det at for å ha en stabil utgangsspenning på 3.3 V, er det anbefalt med en utgangskondensator (C2) på 0.3 μF. Det står også beskrevet at inngangskondensatoren skal være på 1 μF, og at begge disse bør være montert så nære regulatoren som mulig.

Det er også lagt til en jumper på utgangen av lineærregulatoren, for å kunne isolere denne under programmering. Dette er på grunn av at under programmering blir kortet forsynt med 3.3 V direkte fra programmeringskortet *STM32 Nucleo-64*.

5.2 Pumpe- og ventilstyring

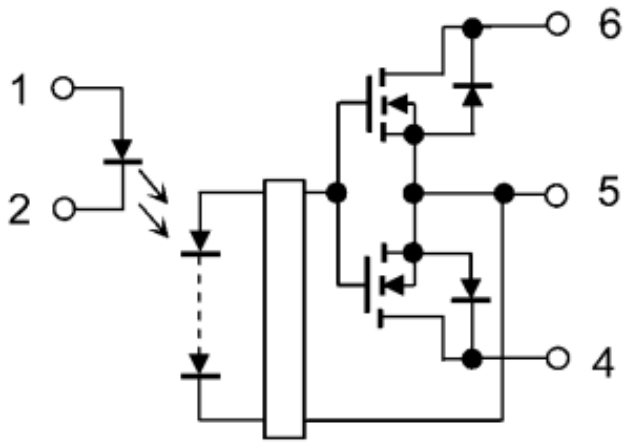
Både pumpa og ventilen vi har valgt, trenger et signal på 12 V for å starte. Dermed er det bruk for en komponent som forsyner disse med 12 V, når den får beskjed om det fra mikrokontrolleren. For å realisere dette har vi valgt å bruke to solid state-releer (U1 og U2). Disse er vist i figur 51 med tilhørende komponenter.



Figur 51: Solid state-releene i skjemategningen.

5.2.1 Solid State-rele

Solid state-releene som brukes heter *TLP3107A* og er kalt U1 og U2 i skjemategningen. *TLP3107A* bruker optokobling for å detektere at det er strømgjennomgang. Dette gjør den ved hjelp av en lysdiode og en fotodiode. Når lysdioden får strøm, vil den belyse fotodioden som blir ledende, og 12 V spenningen ledes gjennom releet. Kretsen inni releene er vist i figur 52.



Figur 52: Intern krets inni solid state-releene. Bildet er hentet fra [41].

Kan se her at lysdioden er koblet på pinne en og to. Spenningen fra mikrokontrolleren kommer altså inn på pinne en, går gjennom dioden, ut på pinne to og til jord. Pinne tre og fire brukes ikke. 12 volten som sendes gjennom releet sendes inn på pinne seks og ut gjennom pinne fem. Sammenligner man med kretsskjemaet i figur 51, er pinne fem koblet til pluggene som skal kobles videre til pumpe og ventil.

Fordelen med å bruke optokobling, er at det ikke blir en direkte kobling mellom inngangen og utgangen. Det vil si at det ikke er noen direkte kobling mellom 3.3 volten, som kommer inn på pinne en, og 12 volten, fra boost-regulatoren til pumpa/ventilen. Dette gir god beskyttelse hvis det, for eksempel, skulle skje noe galt med 12 volten, så den ikke kommer direkte inn på mikrokontrolleren.

Slik vi har koblet solid state-releene (figur 51) er bare en av tre måter man kan velge å koble opp solid state-releene. Alle tre mulighetene er vist i figur 53.

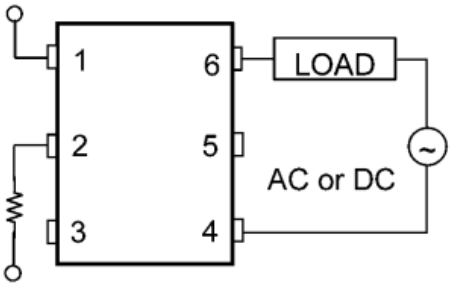


Fig. 12.2.1 A Connection

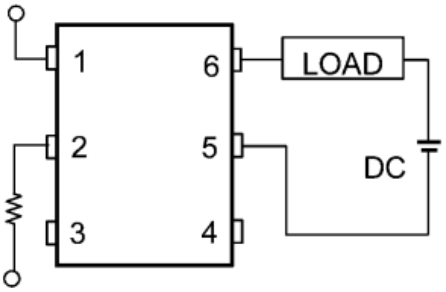


Fig. 12.2.2 B Connection

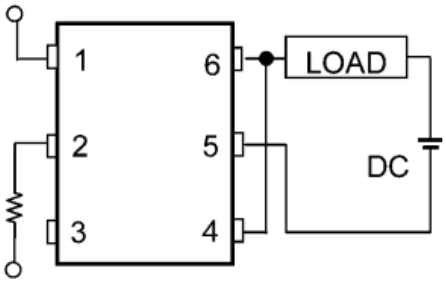


Fig. 12.2.3 C Connection

Figur 53: De ulike mulighetene for oppkobling av solid state-releene. Bildet er hentet fra [41].

Her bruker vi en oppkobling som er nesten lik oppkobling B. Forskjellen er at fra utgang seks, kommer 12 V-spenningen først, og deretter lasten (pumpa og ventilen) på utgang fem. Vår oppkobling kan sees i figur 51. Grunnen til at vi bruker B, er at vi ikke trenger AC som oppkobling A gir muligheten til, og oppkobling C er laget for et mye høyere strømtrekk enn oppkobling B (figur 12.1.8 i databladet: [41]).

Spenningen mikrokontrolleren gir ut, er på 3.3 V, men i databladet står det at dioden på solid state-releet kun skal ha 1.65 V [41]. Derfor er det dimensjonert på en motstand, for å redusere spenningen ned til ønsket verdi. Maksimal tillatt strøm gjennom dioden, er 30 mA. Dermed kan motstanden regnes ut som i ligning 50.

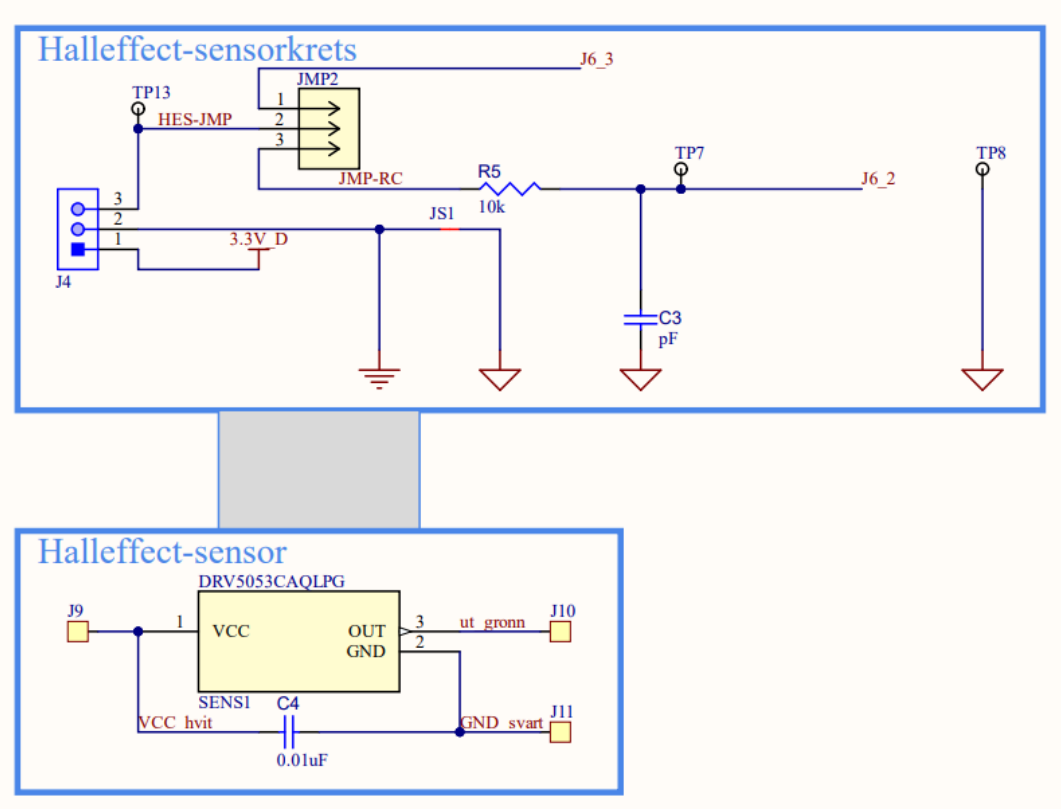
$$R = \frac{U}{I} = \frac{3.3 - 1.65}{30mA} = 55\Omega \tag{50}$$

For å ikke maksimere strømmen dioden kan håndtere, rundes det opp til 120 Ω på denne motstanden. Det vil dermed gå $I = \frac{U}{R} = \frac{3{,}3-1{,}65}{120} = 13.8\ mA$ i dioden. Verdien 1.65 V, er hentet fra tabell 7 i databladet [41]. Denne kan variere med tanke på strømmen, men ved å overdimensjonere motstanden til 120 Ω, sikrer man at strømmen ikke vil bli større enn det dioden tåler. Solid state-releene skal dermed fungere til sin hensikt, og kan brukes til å sende av/på-signal til pumpe og ventil gjennom pluggene, J3 og J5.

5.3 Hall effect-sensorkrets

Den analoge delen av kretskortet skal ta inn signaler fra den analoge Hall effect-sensoren, som ble beskrevet i kapittel 4.3.

Ettersom Hall effect-sensoren må plasseres der hvor magneten til ROV-en treffer flyteren, er det designet et "breakout board"[1] til denne, som kobles til kretskortet via en plugg (J4). Kretsskjemaet for breakout board-et er vist i nederste del av figur 54 og øverste del av figuren viser den analoge delen av kretskortet *Plattformgrensesnitt*.



Figur 54: Logikken rundt Hall effect-sensoren i skjemategningen.

På breakout board-et er det en avkoblingskondensator, selve sensoren, samt tilkoblingspunkter for ledninger. I databladet [16] står det at det må plasseres en keramisk kondensator så nære sensoren som mulig. Denne er derfor plassert på breakout board-et. Den har en anbefalt verdi på 0.01 μF, fra databladet, og har derfor fått denne verdien.

Delen på *Plattformgrensesnitt*, består av et 1.ordens RC-filter. Design-prosedyren til RC-filteret, er beskrevet i databladet, og blir gjennomgått i det neste delkapittelet. Det er også lagt inn en jumper, der

[1] Et breakout board er et eget lite kretskort som kommer utenom hovedkortet.

man kan koble fra RC-filteret, og erstatte dette med en ren bane inn på mikrokontrolleren. På denne måten er det mulighet for å endre filteret, med eksterne komponenter, eller bruke det ufiltrerte signalet.

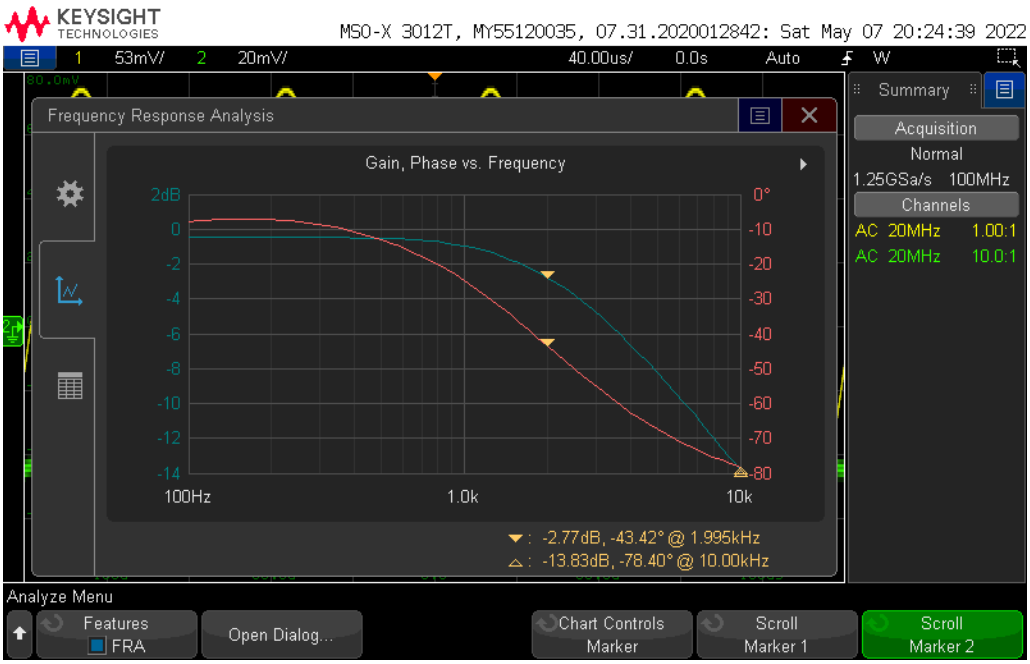
5.3.1 Analog signalbehandling

I og med at Hall effect-sensoren kun skal brukes til å fange opp om flyteren holdes av ROV-en eller ikke, trengs det ikke å detektere særlig høye frekvenser. Det er valgt å sette en grense på $f_{BW} = 1\ kHz$, slik at sensoren ikke fanger opp endringer i magnetfeltet som skjer raskere enn 1 ms. Dette vil også hjelpe med å filtrere bort støy fra for eksempel pumpa. For å realisere dette brukes det et 1. ordens RC-filter.

For å slippe ønskede frekvenser gjennom RC-filteret, må man sette en knekkfrekvens, f_b. Det vil si den høyeste frekvensen som er tillat å slippe gjennom filteret. Knekkfrekvensen har en standard demping på $-3dB$, og alt over denne frekvensen blir dempet med -20 dB per dekade. For å sikre at frekvensen $f_{BW} = 1\ kHz$ kommer gjennom filteret, settes knekkfrekvensen til $f_b = 2 \cdot f_{BW} = 2\ kHz$. Deretter må man finne passende verdier til motstanden og kondensatoren. Fra databladet til Hall effect-sensoren [16], finner man formelen vist i ligning 51, hvor R5 og C3 er motstanden og kondensatoren i RC-filteret. Her er R5 valgt til å være 10 $k\Omega$, som spesifisert i databladet.

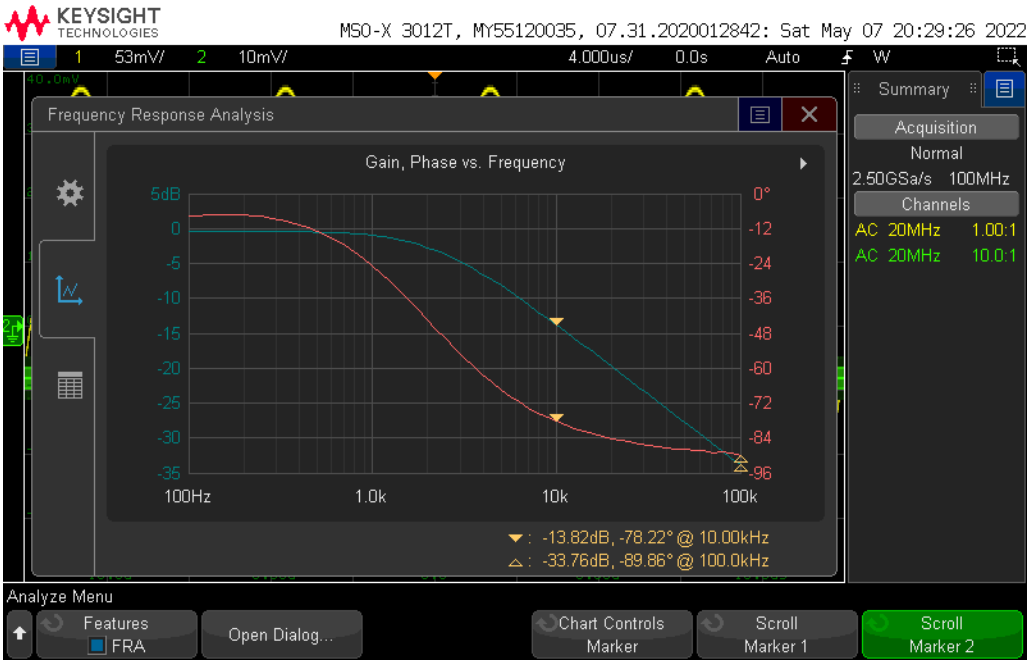
$$\begin{aligned} 2f_{BW} &< \frac{1}{2\pi \cdot R5 \cdot C3} \\ C3 &< \frac{1}{2\pi \cdot R5 \cdot 2f_{BW}} \\ C3 &< \frac{1}{2\pi \cdot 10\ k\Omega \cdot 2\ kHz} \\ C3 &< 7958\ pF \\ \Rightarrow C3 &= 7500\ pF \end{aligned} \tag{51}$$

For å oppnå denne knekkfrekvensen, må altså kondensatoren være mindre enn 7958 pF, det rundes derfor ned til 7500 pF. Bodeplottet for dette 1. ordens lavpassfilteret er vist i figur 55.



Figur 55: Bodeplott med knekkfrekvens $f_b = 1\ kHz$.

Her kan man se at i $f = 2\ kHz$, blir signalet dempet med -3dB og alle frekvenser over det blir dempet. Figur 56 viser bodeplott av det samme filteret, men med litt lengre x-akse.

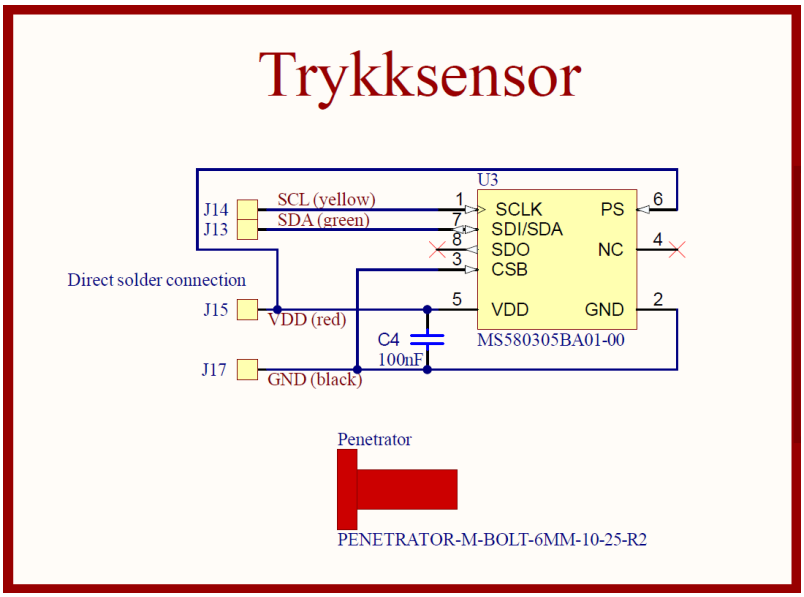


Figur 56: Bodeplott med en markert dekade.

Her er en dekade markert med markørene. Man kan se at på en dekade blir frekvensen dempet med -20 dB, som forventet.

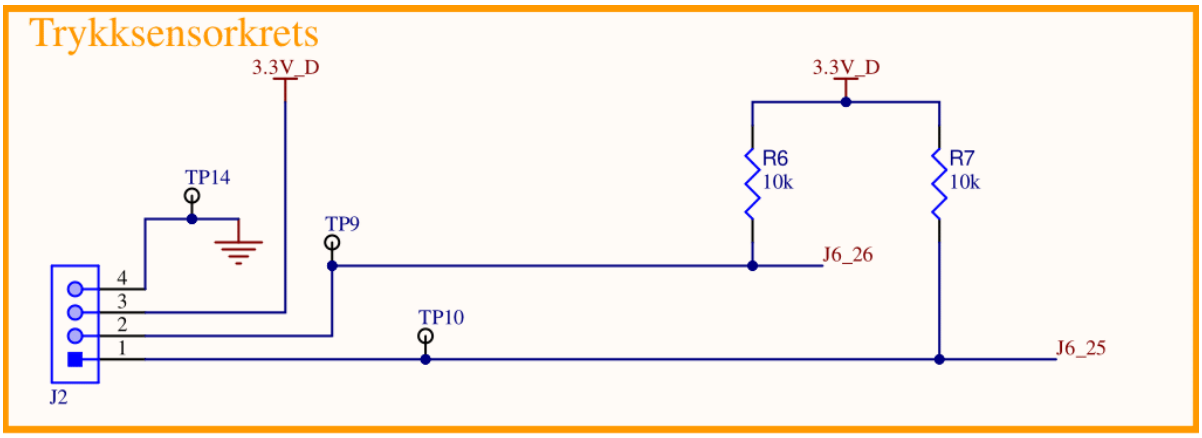
5.4 Trykksensorkrets

Trykksensoren kan, som Hall effect-sensoren, ikke være plassert direkte på *Plattformgrensesnitt*. Denne må være på utsiden av flyteren. Det ble derfor designet et eget breakout board til denne også. Skjemategningen for dette er vist i figur 57. Dette ble designet av Jørgen Hemnes Johannessen fra sensorgruppa på årets ROV, da ROV-en skal ha samme trykksensor.



Figur 57: Skjemategningen for breakout board-et til trykksensoren. Designet av sensorgruppa.

På dette breakout board-et er det en avkoblingskondensator, selve sensoren, samt tilkoblingspunkter til ledninger. Kondensatoren er anbefalt, i databladet [6], til verdien $C4 = 100\ nF$, når man bruker kommunikasjonsprotokollen I^2C. I figur 58, er resten av skjemategningen rundt trykksensoren vist.

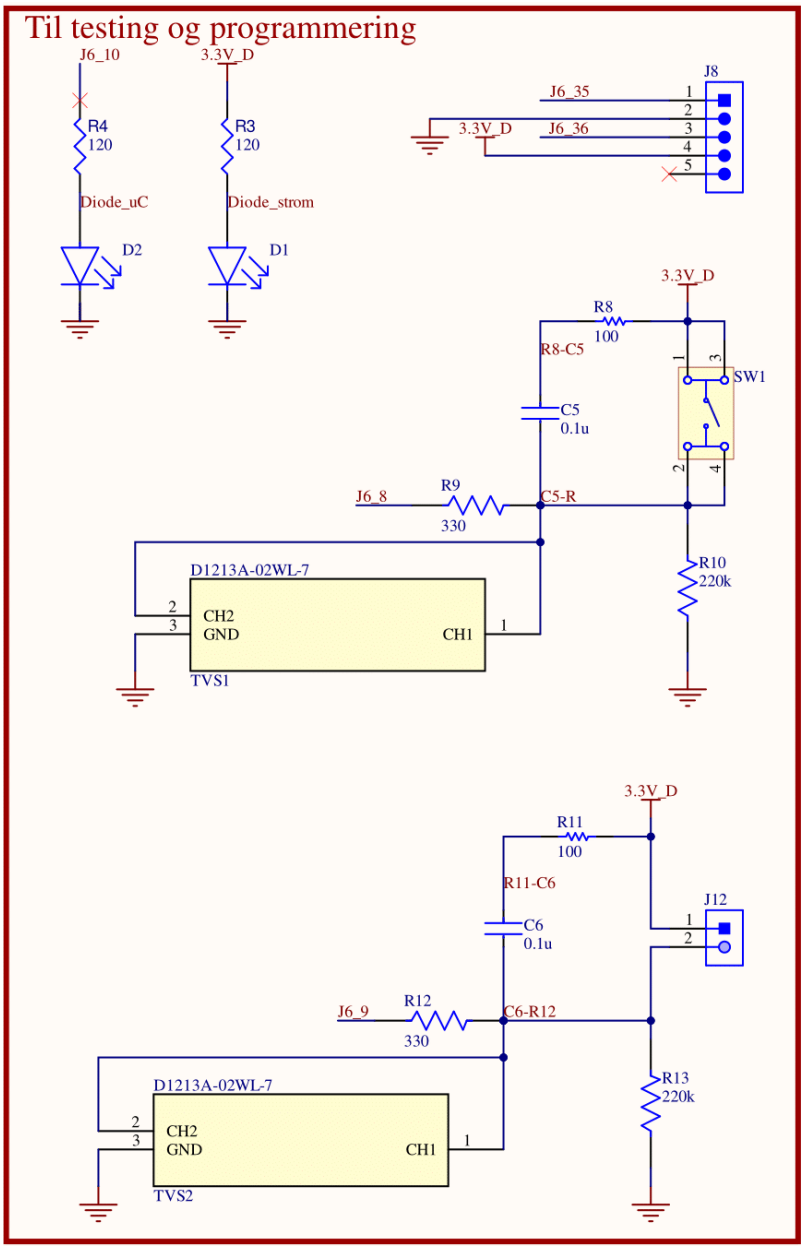


Figur 58: Skjemategningen for logikken rundt trykksensoren.

Her blir data sendt til og fra trykksensoren, via datalinjene SDA og SCL, som er koblet til hver sin pinne på mikrokontrolleren. Det er også lagt inn to pull up-motstander. Disse er spesifisert i databladet, til 10 kΩ, når man skal bruke et I^2C-grensesnitt.

5.5 Testing og programmering

Den siste delen av skjemategningen er vist i figur 59. Her er det lagt opp til to brytere, SW1 og SW2. SW1 er plassert på kretskortet, og SW2 kan kobles til via plugg J12.



Figur 59: Skjemategning av bryterkrets og tilleggslogikk.

Det er også tilkobling for programmering, via pluggen J8, og to dioder, D1 og D2. D1 er koblet direkte

på 3.3 V spenningen, for å indikere at systemet er spenningssatt. D2 er koblet til mikrokontrolleren og kan dermed programmeres til ønsket funksjon.

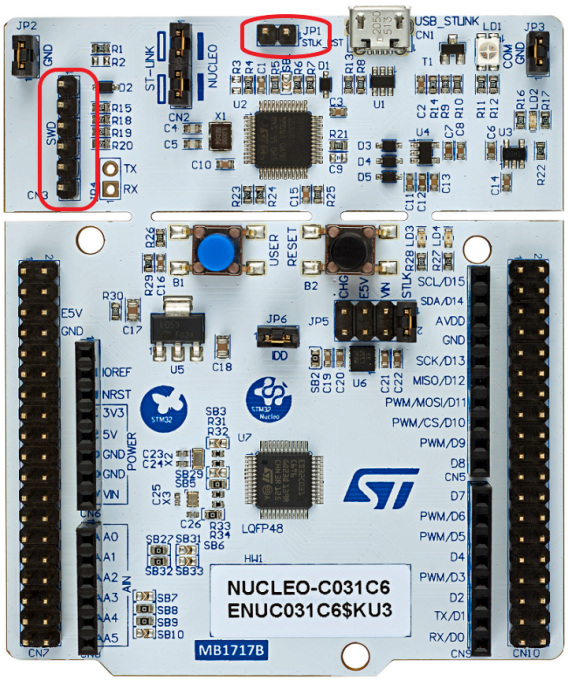
5.5.1 Bryterkrets

Bryteren SW1 er den samme som finnes på kretskortet *STM32F3DISCOVERY*, og databladet kan ses i [32]. Kretsen rundt bryteren er spesifisert i databladet, foruten TVS-dioden. Bryterkretsen inneholder et RC-filter, som filtrerer bort avprelling ved trykk av bryteren. Dette sørger for at det ikke er nødvendig å programmere ekstra filtrering ved brytertrykk.

De to bryterne, SW1 og SW2, er designet med samme bryterkrets, men SW2 er ikke direkte koblet på kortet. Her har vi satt på en plugg (J12) slik at bryteren kan være på utsiden av flyteren. TVS1 og TVS2 er overspenningsvern for bryterne. Grunnen til at vi har valgt å ta med disse overspenningsvernene, er nærmere beskrevet i kapittel 5.7.

5.5.2 Tilkobling for programmering

Til å programmere mikrokontrolleren, brukes et *STM32 Nucleo-64*-kort, vist i figur 60. Dette gjøres ved SWD-protokoll (Serial Wire Debug) som er en programmeringsprotokoll, for å få tilgang til *"debugger"*-grensesnittet i ARM-mikroprosessoren.



Figur 60: *STM32 Nucleo-64 Board*, bilde hentet fra [35].

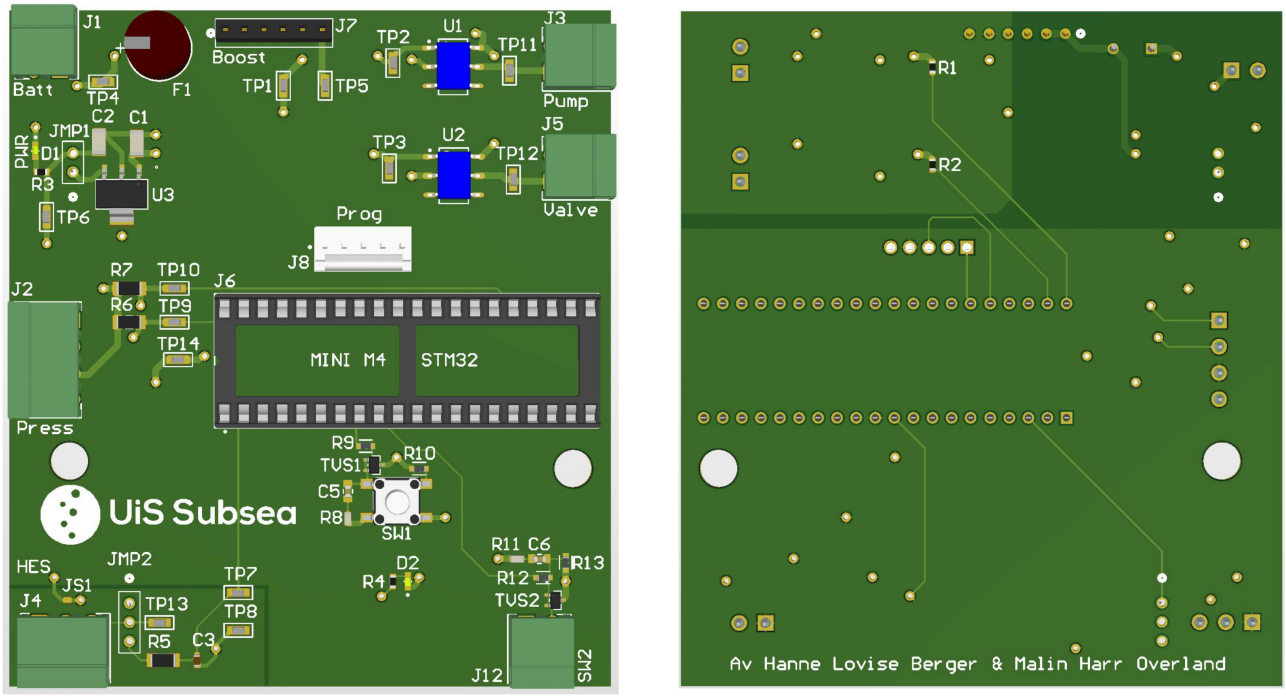
På kretskortet som er designet, kan programmeringspluggen, J8, kobles til *STM32 Nucleo-64*-kortet som vist i tabell 9. Koblingspunktene på *STM32 Nucleo-64*-kortet er markert i rødt i figur 60.

STM32 Nucleo-64	**Plattformgrensesnitt**
SWD_2	J8_3
SWD_3	J8_2
SWD_4	J8_1
JP1_1	J8_4

Tabell 9: Koblingspunkt for programmering av mikrokontroller.

5.6 Utlegg

Når komponentene skal plasseres på utlegget er det flere hensyn som må tas. Noen av komponentene må plasseres nære hverandre, det må tas hensyn til støy, analoge signaler bør skilles fra digitale og banebredden må tilpasses for strøm, for å minimere effekttap. 3D-modellen av utlegget for kretskortet er vist i figur 61, med oversiden til venstre og undersiden til høyre.



Figur 61: 3D-modell av kretskortet.

Arbeidet med å designe utlegget ble mye enklere i og med at det ble lagd, og tatt i bruk, 3D-modell av komponentene og kretskortet. Det er dermed enklere å se hvordan plasseringene passer i forhold til realiteten.

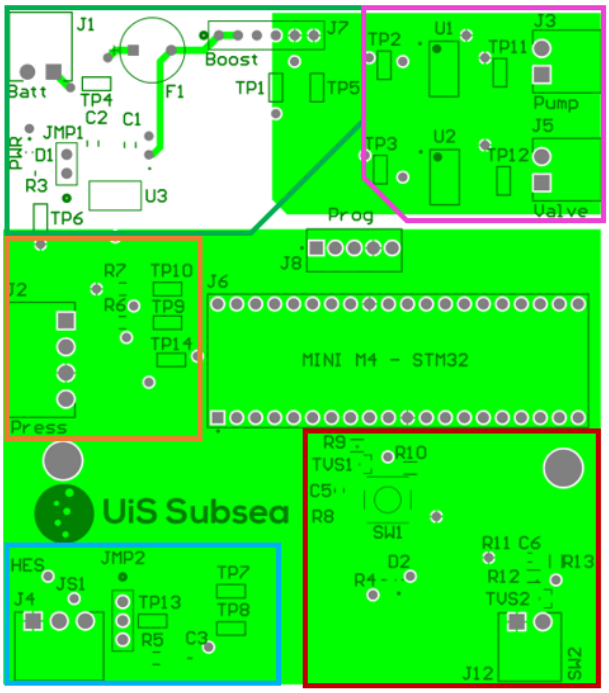
5.6.1 Kretskortlag og plassering av komponenter

Kretskortet er laget med fire lag (Layers på engelsk). Dette er for å kunne isolere forsyningsspenning og jord på hvert sitt lag, og for å kunne fordele de resterende signalbanene i topp- og bunnlag. Tykkelsen på lagene ble satt til 0.035 mm, for lag en og fire, og 0.0175 mm, for lag to og tre. Dette er standarden til JLCPCB (der vi bestilte kortene) og ble derfor tatt i bruk for hele UiS Subsea. Lagene med spesifikasjoner er vist i figur 62, som er tatt fra *Altium Designer*.

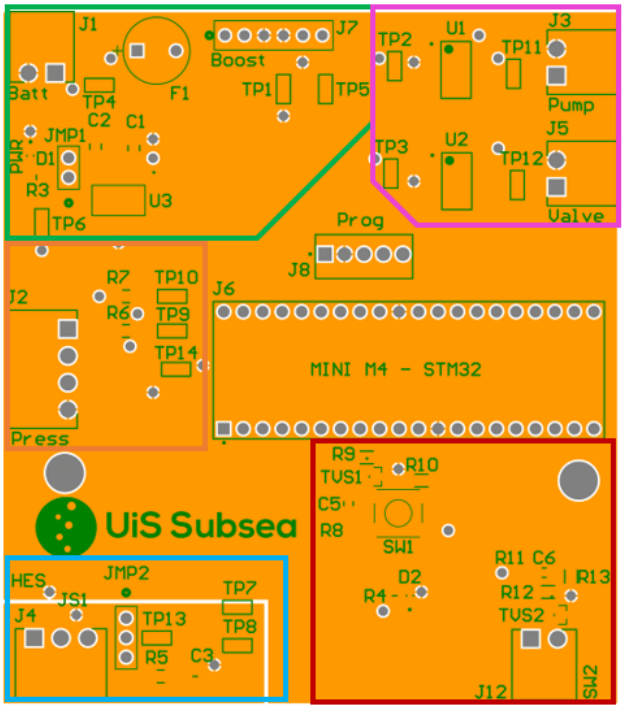
#	Name	Material	Type	Weight	Thickness	Dk	Df
	Top Overlay		Overlay				
	Top Solder	Solder Resist	Solder Mask		0.0127mm	3.8	
1	Top Layer		Signal	1oz	0.035mm		
	Dielectric 1	2313	Prepreg		0.1mm	4.05	
2	Mid Layer 1		Signal	1/2oz	0.0175mm		
	Core	FR-4	Core		1.265mm	4.5	
3	Mid Layer 2		Signal	1/2oz	0.0175mm		
	Dielectric 2	2313	Prepreg		0.1mm	4.05	
4	Bottom Layer		Signal	1oz	0.035mm		
	Bottom Solder	Solder Resist	Solder Mask		0.0127mm	3.8	
	Bottom Overlay		Overlay				

Figur 62: Alle lagene på kretskortet med spesifikasjoner.

Lag to (Mid Layer 1) og tre (Mid Layer 2) er lagene for henholdsvis kraftforsyning og jord. I figur 63, kan man se forsyningslaget i lysegrønt. Silketrykket er lagt på i mørkegrønt, samt fargede markeringer med samme fargekoder som er brukt i skjemategningen. Hele skjemategningen kan sees i vedlegg A. I figur 64 er jordlaget vist i oransje, igjen med silketrykk og inndelinger som i skjemategningen.



Figur 63: Forsyningslag vist i lysegrønt og silketrykk i mørkegrønt.

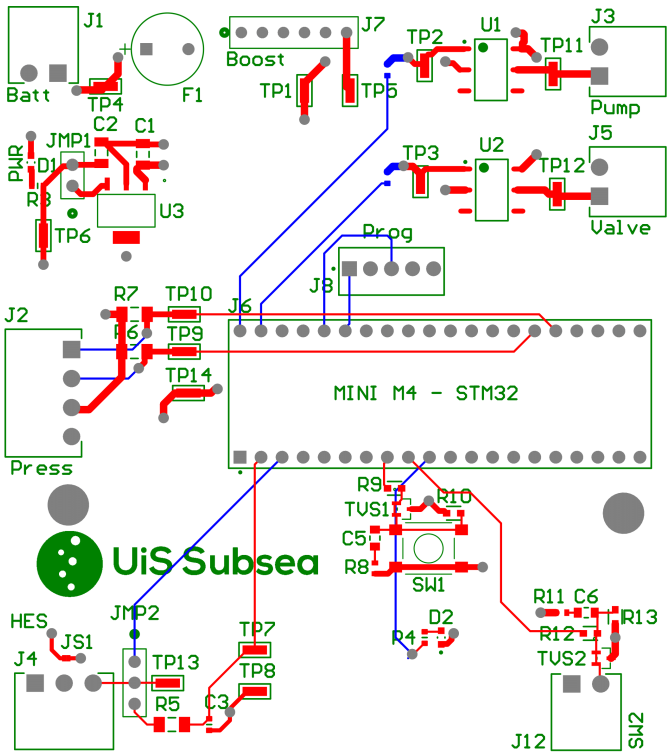


Figur 64: Jordlag vist i oransje og silketrykk i mørkegrønt.

Forsyningslaget, i figur 63, er delt i to. Alt som går på 12 V er isolert fra det som går på 3.3 V. Det er pumpa og ventilen som går på 12 V, de er plassert øverst i høyre hjørne, i den rosa firkanten. Det største lysegrønne området er 3.3 V. Her finner man trykksensorkretsen i den oransje boksen, Hall effect-sensorkretsen i den blå, bryterkretsene i den røde, og over den er mikrokontrollerkortet og tilkobling for programmering. Den grønne firkanten inneholder batteritilkoblingen, og spenningsregulatorene.

Jordlaget, i figur 64, er også separert. De to jordtypene som er separert, er digital og analog jord. Den analoge delen av kretskortet er plassert nederst i den lyseblå boksen, mens resten av laget er digital jord. Det går en bane i topplaget mellom disse to jordlagene, for å koble de sammen. Her er det også plassert et fotavtrykk, JS1, som kan ses i figur 61. Dette er et fotavtrykk uten komponent på, fordi *Altium Designer* ikke tillater å koble sammen to ulike nett direkte. Både forsyningslaget og jordlaget, holdes av til kraftforsyning og jord, og brukes ikke til å rute andre signalbaner.

Topp- og bunnlaget brukes til å rute de resterende banene. Dette er vist i figur 65, hvor banene som ligger i topplaget er i røde, og banene i bunnlaget vises i blått. Det er kun to komponenter som er plassert i bunnlaget. Dette er motstandene som hører til solid state-releene, R1 og R2.



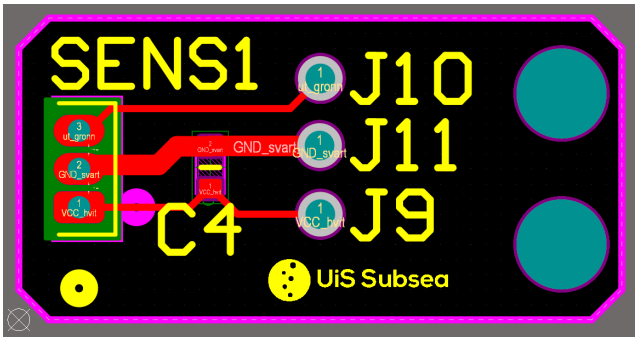
Figur 65: Topp- og bunnlaget. De røde banene ligger i topplaget og de blå ligger i bunnlaget.

To komponenter som var viktige å plassere riktig, var TVS-diodene. For at disse skulle fungere til sin hensikt, måtte de plasseres mellom bryteren og inngangen på mikrokontrolleren. Det er også plassert en via i nærheten, med kontakt til jord, slik at overspenning raskt blir sendt til jord. Disse viaene kan lettest ses i figur 61, til høyre for TVS1, og ovenfor TVS2.

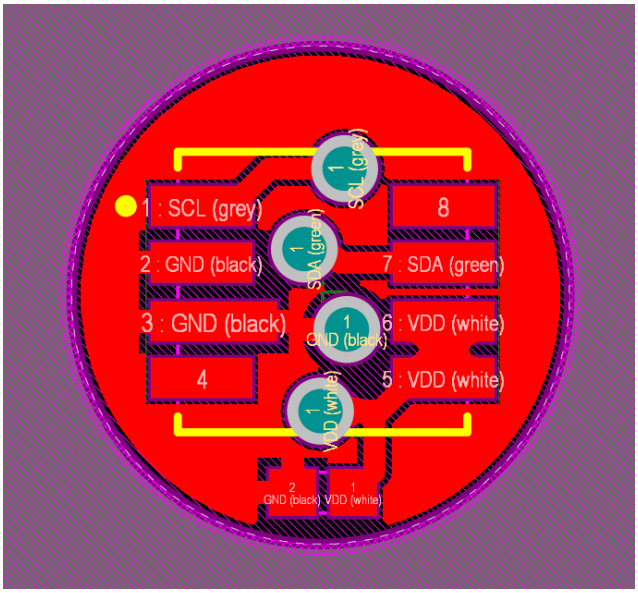
Ellers er komponentene og banene plassert, og rutet, for å oppnå kortest mulig bane fra komponent til mikrokontroller. Pinnene på mikrokontrolleren ble konfigurert samtidig som utlegget ble designet. Man kan i ettertid se forbedringspotensiale i ruting og komponentplassering, dersom dette hadde vært nøyere planlagt. Dette forklares nærmere i kapittel 8.

5.6.2 Breakout board-ene

Som nevnt i kapittel 5.3 og 5.4, ble det designet egne breakout board for både Hall effect-sensoren og trykksensoren. Breakout board-et for trykksensoren er designet av Jørgen Hemnes Johannessen i sensorgruppa. Disse er vist i figur 66 og 67. Begge kortene er to-lags, men har kun komponenter plassert på toppsiden.



Figur 66: Utlegget til Hall effect-sensoren.



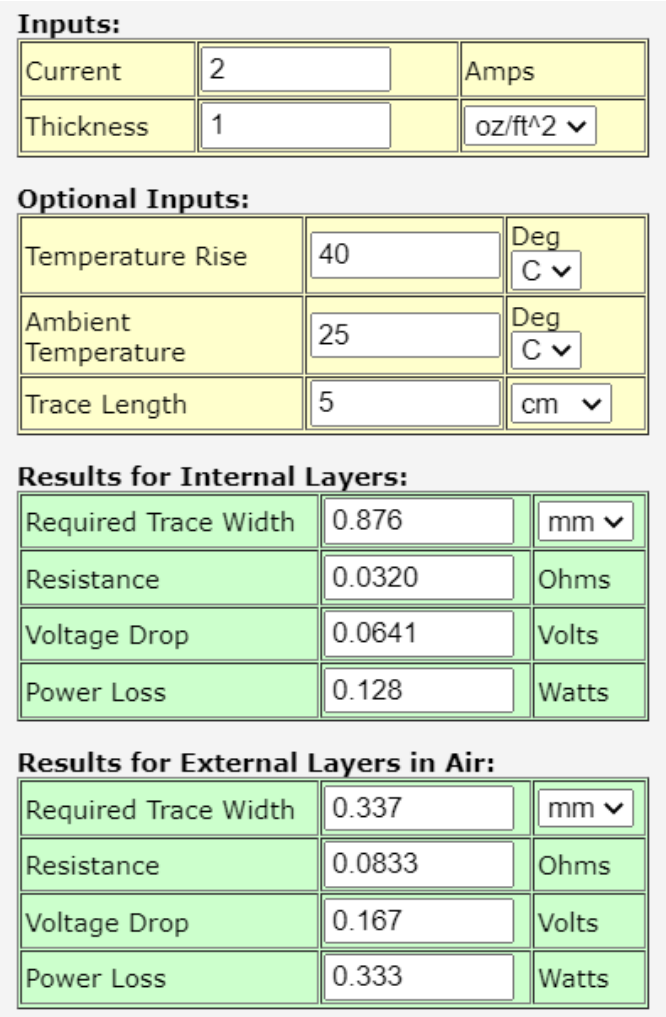
Figur 67: Utlegget til trykksensoren. Vi har fått figuren fra sensorgruppa.

Begge sensorene har spesifisert i databladene ([16] og [6]) at det skal være en kondensator så nære sensoren som mulig, disse er derfor plassert på breakout board-ene.

5.6.3 Banebredde

For å bestemme banebredde, ble det brukt en banebreddekalkulator fra *Advanced Circuits* sine nettsider [5], samt en standard utarbeidet av IPC [2] [17]. I kalkulatoren, la man inn verdiene som vist i figur 68.

[2]IPC er en organisasjon som utarbeider standarder innenfor elektronikkdesign



Inputs:

Current	2	Amps
Thickness	1	oz/ft^2

Optional Inputs:

Temperature Rise	40	Deg C
Ambient Temperature	25	Deg C
Trace Length	5	cm

Results for Internal Layers:

Required Trace Width	0.876	mm
Resistance	0.0320	Ohms
Voltage Drop	0.0641	Volts
Power Loss	0.128	Watts

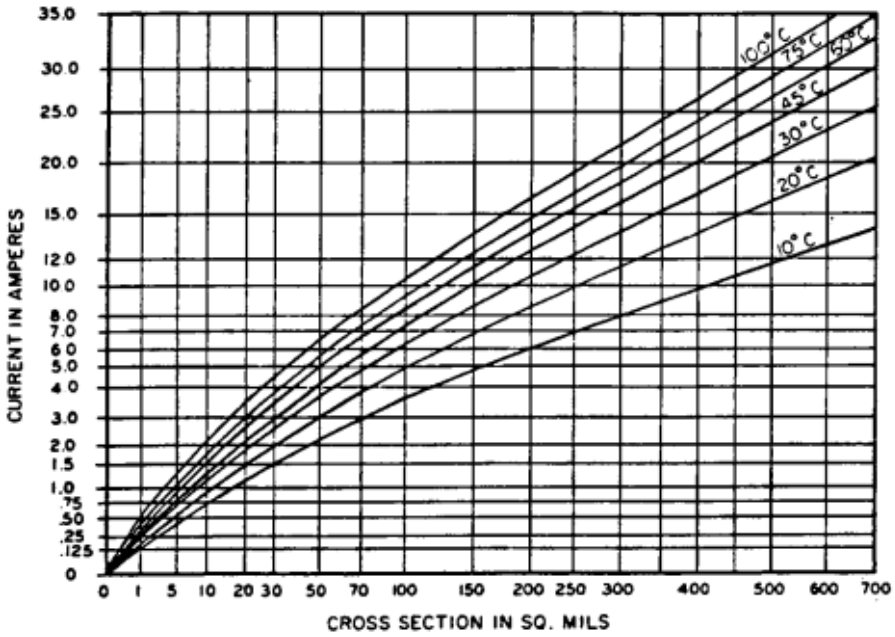
Results for External Layers in Air:

Required Trace Width	0.337	mm
Resistance	0.0833	Ohms
Voltage Drop	0.167	Volts
Power Loss	0.333	Watts

Figur 68: Banebreddekalkulator.

Her er verdiene for strømmen satt til en mer realistisk verdi enn det som ble funnet i kapittel 4.4. Banelengden er satt litt høyere enn hva den vil være i realiteten, for å være på den sikre siden. For temperaturen, har man satt en omtrentlig verdi for hvor mye banene kan tillates å øke med. Dermed regnes det ut at banene på de indre lagene bør ha en bredde på minimum 0.876 mm (34.488 mils), og banene i topp- og bunnlaget bør ha en bredde på 0.337 mm (13.268 mils).

I standarden fra IPC, finner man grafen vist i figur 69, som viser forholdet mellom strøm og banebredde ved ulike temperaturer.



Figur 69: Graf med banebredde mot strømtrekk. Hentet fra [17].

De forskjellige kurvene viser til hvor mye temperaturen øker i banene. Følger kurven med 45°C, og finner hvor den møter linjen for 2.0 A. Finner da at anbefalt banebredde fra standarden er 20 $mils = 0.508\ mm$.

Ut fra de to kildene, ble det bestemt å bruke tre ulike banebredder. Banebreddene er større enn det som ble anbefalt, i både kalkulatoren og IPC standarden, med unntak av signalbanene til og fra mikrokontroller.

- 0.9 mm (35.433 mils) → de banene med størst strømmer, som for eksempel ut fra batteriene og frem til pumpa og ventilen.
- 0.254 mm (10 mils) → signalbanene til og fra mikrokontrolleren.
- 0.7 mm (27.559 mils) → resterende baner, alle disse ligger i topplaget.

5.7 Utlegg: Støyimmunitet

Dette delkapittelet presenterer de ulike typene støy, samt hvilke tiltak som er gjort på kretskortet for å isolere mot det. Det er to ulike former for støy her:

1. ESD (Electro Static Discharge)
2. EMI (Electromagnetic Interference)

Under følger en gjennomgang av de to støytypene.

5.7.1 ESD

ESD er statisk elektrisitet, som blir overført fra en gjenstand til en annen. I denne sammenhengen, oppstår det når en person tar på kretskortet, for eksempel under lodding, eller ved et knappetrykk. Dersom spenningen fra en person "smitter" over på elektronikken, kan den bli ødelagt dersom man ikke har gjort gode nok tiltak. Det er tatt i bruk to ulike tiltak for å beskytte mot ESD. Disse gjennomgås her.

Jordingsarmbånd: Under lodding ble det brukt jordingsarmbånd. Dette beskytter kretskortet fra spenninger ved berøring, ved at man er koblet til jord, på samme måte som komponenten som håndteres.

TVS-dioder: På kretskortet er det to brytere koblet til, en som er montert direkte på kretskortet, og en som kobles til via en plugg. På disse er det lagt til to TVS-dioder. En TVS-diode fungerer slik at når den blir utsatt for ESD, vil den få en spenning over seg og lede strømmen direkte til jord.

Trykk- og Hall effect-sensoren blir koblet til kretskortet med to plugger. Her kunne det vært aktuelt med TVS-dioder for å beskytte mot ESD. Dette er tatt til vurdering, men ettersom sensorene har en intern ESD-beskyttelse på 4 kV for trykksensoren, og 2.5 kV for Hall effect-sensoren, er ikke dette nødvendig. Sensorene skal ikke kobles inn og ut under normal drift. Det er dermed mulig å bruke jordingsarmbånd når de monteres på laboratoriet.

5.7.2 EMI

EMI er støy som kommer fra en ekstern kilde, altså all støy som ikke er en indre egenskap. Det er fem forskjellige typer EMI [39], som skal gjennomgås her sammen med tiltakene som er gjort mot dem.

Induktans i ledere:

I en leder vil det alltid være en induktans, en såkalt parasittinduktans. Når det kommer strømendringer i lederne, vil det bli en spenningsendring over induktansen. Dette kan føre til forstyrrelser i andre spenninger, eller signaler.

For å motvirke denne parasittinduktansen, har man hatt fokus på effektiv baneruting. Dette vil si å ha så korte baner som mulig, samt å plassere komponenter logisk, slik at det blir mulig. Det er også brukt avkoblingskondensatorer på, for eksempel, U3 og Hall effect-sensoren. Dette hjelper ved å glatte ut rippelspenninger fra strømendringene.

Induktiv kobling mellom ledere:

Denne typen støy er det som oppstår mellom to signalsløyfer dersom de ligger for nære hverandre. I vårt system er det som utgjør størst risiko her, de to spenningsregulatorene, og aller mest den svitsjede boost-regulatoren.

Boost-regulatoren er plassert på et eget kretskort, som er koblet vinkelrett på vårt kretskort, og er dermed godt isolert fra de andre komponentene. Begge regulatorene, samt strømforsyningen, er plassert langt unna den analoge delen av kretskortet, for å ta hensyn til støy. I tillegg er det prøvd å plassere komponenter, som er koblet sammen, så nære hverandre som mulig, slik at sløyfearealet blir mindre. Et annet tiltak som er gjort, er å bruke polygoner i utlegget. Dette er et større felt som er tilkoblet samme nett. Vi har fire forskjellige polygoner på vårt kretskort. Blant disse har man GND_D, AGND, 12V og 3.3V. Dette er også beskrevet i kapittel 5.6.1.

Kapasitiv kobling mellom ledere:

Kapasitiv kobling mellom ledere, blir kalt både signalsmitte og interferens. Interferens er når et signal smitter over fra en bane, til en annen, og dermed blir oppfattet som støy. Et hovedtiltak som er gjort for å unngå dette, er å sette regler for avstanden mellom ledere. Dette er gjort i *Design Rule Check* (DRC) i *Altium Designer*, og er vist i figur 70.

	Track	SMD Pad	TH Pad	Via	Copper	Text
Track	0					
SMD Pad	0	0.254				
TH Pad	0.254	0.254	0.254			
Via	0.254	0.254	0.254	0.254		
Copper	0.254	0.254	0.254	0.254	0	
Text	0.254	0.254	0.254	0.254	0	0
Hole	0	0	0	0	0	0

Figur 70: Reglene satt i *Design Rule Check*(DRC) i *Altium Designer*.

Tabellen i figuren er satt opp på en slik måte at tallene er avstanden som skal være mellom de to komponentene, som krysser i den gjeldende ruta. Som man kan se i figuren, er avstanden satt til 0.254 mm [3]. Dermed gir *Altium Designer* en varsling dersom disse reglene ikke blir overholdt.

[3] 0.254 mm er standardverdi i *ALtium Designer*

Innstrålt elektromagnetisk støy:

Innstrålt elektromagnetisk støy kan innebære både støy fra en kilde som har elektromagnetisk induksjon og elektromagnetiske pulser. De vanligste kildene til slik støy, er for eksempel trådløs- og radiokommunikasjon, motordrivere og brytere. I vårt system tilsvarer dette bryterne, pumpa og ventilen.

Systemet har to brytere. En for å kunne skru av og på batteriforsyningen, og en for å starte/resette flyteren under testing/kjøring. Begge disse er montert på utsiden av flyteren. Dette er, i seg selv, et tiltak mot den elektromagnetiske støyen. Kretskortet er dermed godt beskyttet mot eventuelle gnister som oppstår ved innkobling av batteri, ettersom dette oppstår langt unna kretskortet.

Når pumpa og ventilen kjører, kan det føre til innstrålt elektromagnetisk støy. For å beskytte mot dette er det 3D-printet et brett som står montert i midten, mellom pumpa/ventilen og kretskortet. Dette er en del av designet til maskingruppa som har ansvar for design og montering av flyteren og ROV-en.

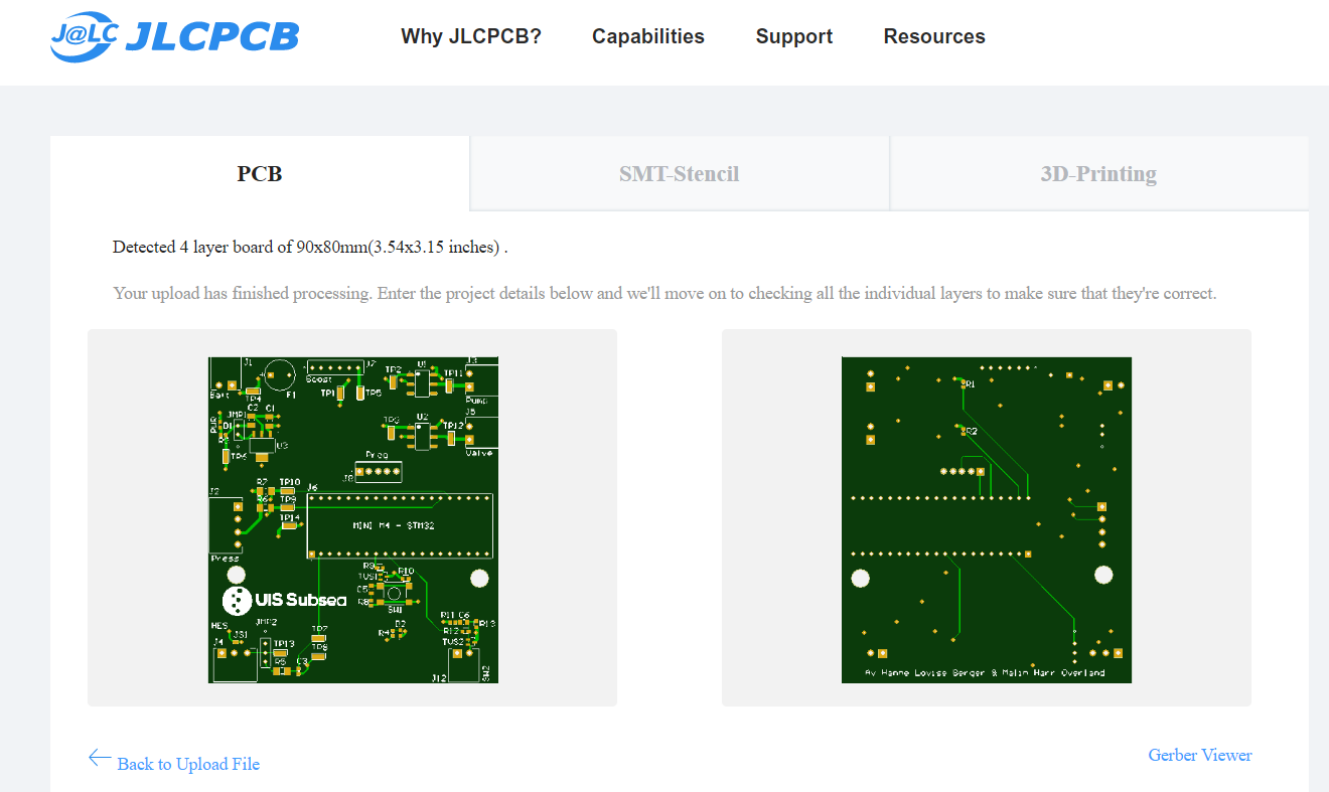
Jordstøy:

I en jordbane vil det alltid være en motstand. Det vil også alltid være en forskjell i potensialet mellom jordnett. Dette kan skape såkalt jordstøy. Det er primært sikret mot jordstøy ved å bruke flere jordnett. På kretskortet er det laget to forskjellige jordnett. Disse heter GND_D, for digital jord, og AGND, for analog jord.

I kretsen vår, er det i tillegg to komponenter som har galvanisk skille, nemlig solid state-releene. Dette kan ses i figur 52, hvor det vises at det ikke er kobling mellom lys- og fotodiodene. I praksis vil det likevel ikke være et galavanisk skille i kretsen vår. Dette fordi både 3.3 V-spenningen, som kommer inn på releene fra mikrokontrolleren, og 12 V-spenningen, som kommer fra boost-regulatoren, er koblet sammen ved batteriet. Dermed blir det ikke et skille i kretsen, selv om det er det inne i solid state-releene.

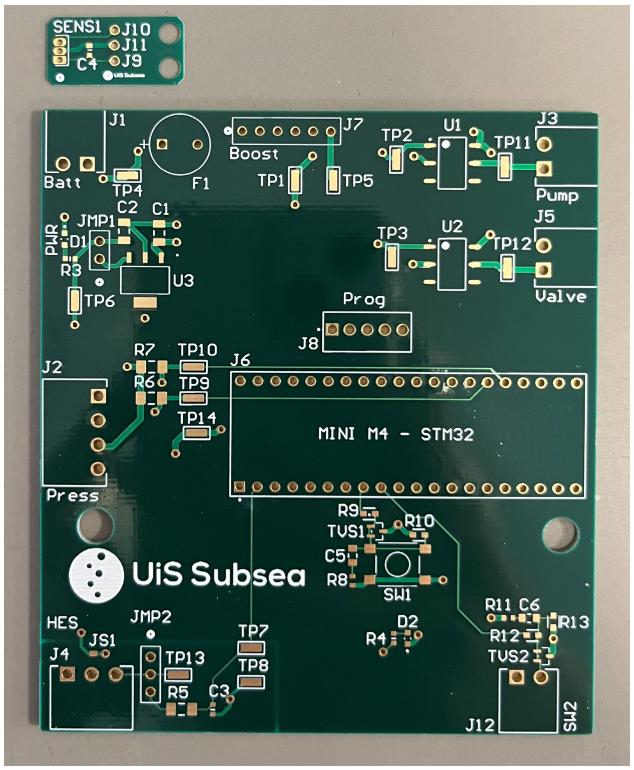
5.8 Produksjon

I år valgte samtlige grupper i UiS Subsea å bestille inn kretskortene, men å lodde på komponenter selv. Kretskortene ble bestilt fra leverandøren *JLCPCB*. Før man kan bestille, må det lages såkalte gerber-filer. Dette ble gjort som beskrevet i brukermanualen for *Altium Designer* i [28]. Disse gerber-filene ble lastet opp på nettsidene til *JLCPCB*. Da de var ferdig opplastet, så siden ut som i figur 71.



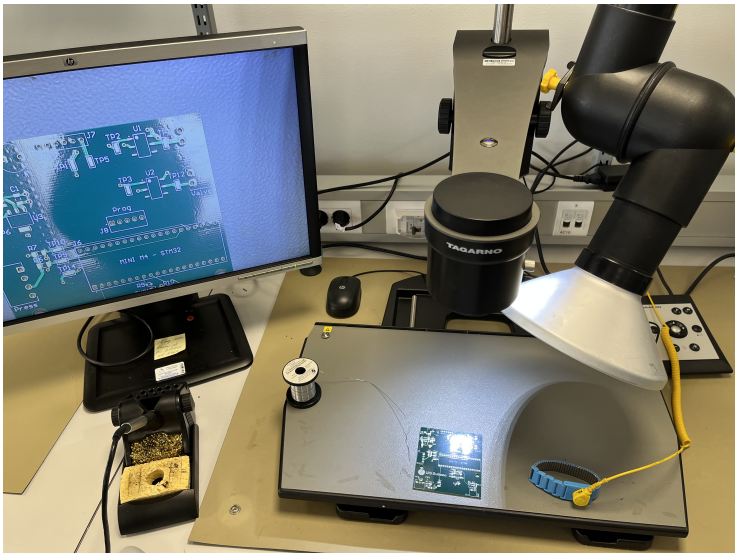
Figur 71: Inne på nettsiden til *JLCPCB* etter gerber-filene er lastet opp.

Her fikk man altså opp et bilde av både topp- og bunnside, som gjør det lett å dobbeltsjekke at alt er slik det skal være. Videre nedover på nettsiden, ble det spesifisert flere detaljer på kortet (som farge og vekt) før bestillingen ble sendt inn. I figur 72, er både hovedkortet og breakout board-et til Hall effect-sensoren vist slik de var da vi fikk de fra *JLCPCB*.



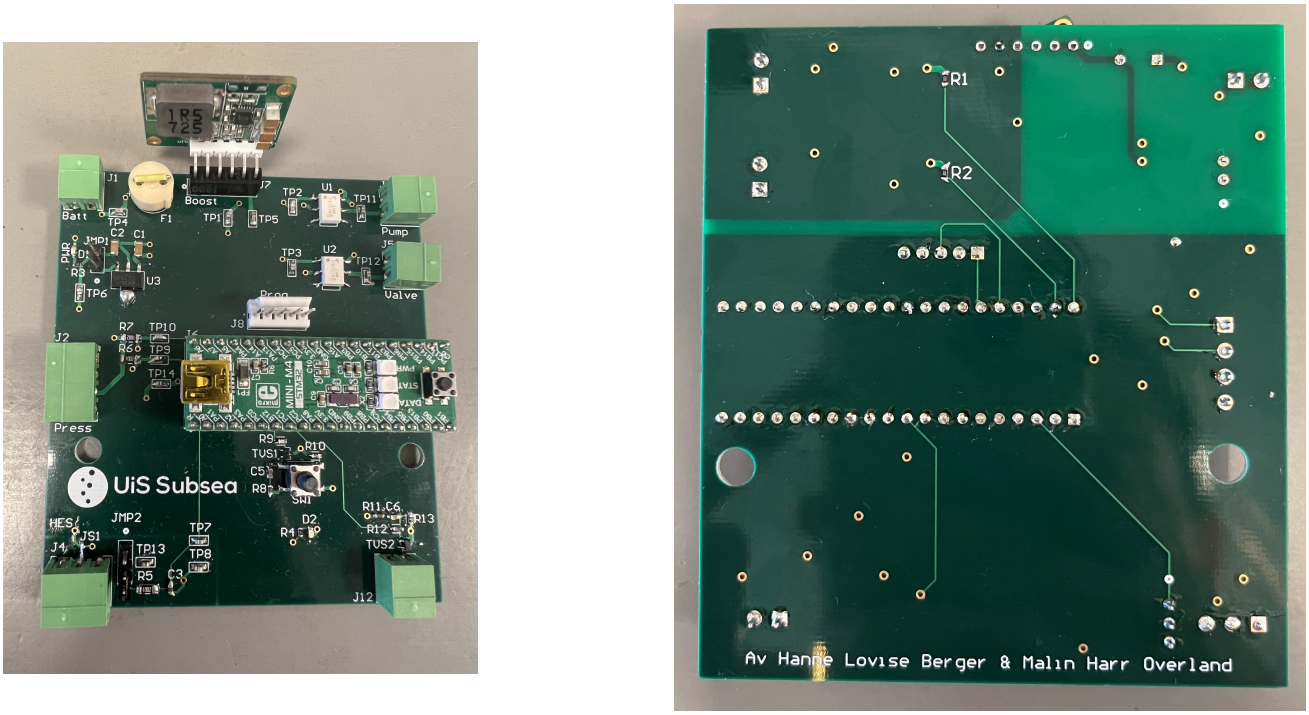
Figur 72: Ferdig produsert kort uten komponenter loddet på.

Deretter gjenstod det å lodde på komponenter. Dette ble gjort for hånd, med utstyret som er tilgjengelig på laboratoriet ved Universitetet i Stavanger, vist i figur 73.



Figur 73: Oppsettet for lodding av kretskort.

Loddeprosessen ble utført i flere steg, og kretskortet ble testet underveis. For eksempel ble kraftforsyningen loddet og testet først. Dette kan leses mer om i testrapport B.3. Etter lodding, er kretskortet ferdig, og det endelige resultatet er vist i figur 74.



Figur 74: Det ferdig loddede kortet, med toppsiden til venstre og undersiden til høyre.

Breakout board-ene til både trykk- og Hall effect-sensoren, ble også bestilt i *JLCPCB* og produsert på samme måte.

6 Programvare

Innhold

I dette kapittelet skal programvaren til systemet gjennomgås. Dette innebærer programstrukturen, programkoden for styresystemet, hvordan mikrokontrolleren er konfigurert, og til slutt litt om de ulike kommunikasjonsprotokollene som er brukt. Programvaren til systemet er laget i *STM32CubeIDE*, og kan finnes i [2].

6.1 Programstruktur

Programstrukturen er laget for at koden skal være så leselig som mulig. Det er lagt fokus på å gi funksjoner og variabler rasjonelle og meningsfulle navn for å oppnå dette, samt å samle funksjoner og initialisering av variaber i egne filer for å øke leseligheten til hovedprogrammet. I det neste delkapittelet skal filhierarkiet gjennomgås.

6.1.1 Filhierarki

I strukturen vist nedenfor, er en oversikt over hierarkiet i prosjektet, som er laget i *STM32CubeIDE*. Strukturen inneholder kun de filene i prosjektet som er laget og endret på, og som dermed er aktuelle å dokumentere. Under mappen `Inc`, ligger alle hjelpefilene (*.h*). Dette er filer som inneholder deklarering og initialisering av globale og lokale variabler. Under mappen `Src`, ligger kildefiler (*.c*), som inneholder utførbar programkode. Mappen `USB_DEVICE` inneholder filen `usbc_cdc_if.c`, som brukes for seriell kommunikasjon. Dette blir gjennomgått i kapittel 6.4.3. Strukturen og det overordnede innholdet i kildefilene skal gjennomgås her.

```
UiS Subsea Float
├── Core
│   ├── Inc
│   │   ├── *** Hjelpefiler ***
│   │   ├── extern_variables.h
│   │   ├── main.h
│   │   ├── trykksensor.h
│   │   └── variables.h
│   └── Src
│       ├── *** Kildefiler ***
│       ├── functions.c
│       ├── main.c
│       ├── stm32f4xx_it.c
│       └── trykksensor.c
├── USB_DEVICE
│   └── App
│       ├── *** Seriell kommunikasjon ***
│       └── usbd_cdc_if.c
└── UiS Subsea Float.ioc
```

main.c
`main.c`-filen er hvor hovedprogrammet utføres. Denne filen tar inn egendefinerte variabler fra hjelpefilene `variables.h` og `trykksensor.h`. Hovedprogrammet bruker andre funksjoner, som er skrevet i andre kildekodefiler, som `functions.c` og `trykksensor.c`.

stm32f4xx_it.c
Denne filen inneholder funksjoner som fungerer som avbrudd. Av disse, er det funksjonen `SysTick_Handler` som brukes. Dette er en avbruddsmetode som kjøres, hvert millisekund, av mikrokontrolleren. Her kalles funksjoner som sjekker om brytere er trykket inn, i tillegg til å ha diverse tellere som brukes i hovedprogrammet. Dette blir gjennomgått nærmere i kapittel 6.2.2.

functions.c
`functions.c`-filen består av diverse funksjoner som brukes både i `main.c`-filen, og i `SysTick_Handler`. Denne filen tar inn globale variabler fra `extern_variables.h`-filen, slik at disse kan brukes videre i andre kildekoder. For at funksjonene kan kalles på i andre kildekodefiler, må de funksjonene som brukes, initialiseres i den kildekodefilen de blir brukt i.

trykksensor.c
I denne filen er driverkoden for trykksensoren skrevet. Denne filen inneholder funksjoner som initialiserer, leser kalibrasjonsdata og leser av trykket fra trykksensoren. Disse funksjonene blir brukt i hovedprogrammet i `main.c`. Hvordan funksjonene er laget og brukes, blir nærmere gjennomgått i kapittel 6.4.2, som tar for seg I^2C-protokollen til trykksensoren, samt i kapittel 6.2.1 som tar for seg hovedprogrammet til systemet.

6.1.2 Globale variabler

Globale variabler blir deklarert i `extern_variables.h`-filen med bruk av skrivemåten `extern uint8_t`. Disse variablene kan bli brukt i alle kildekodefilene som tar inn `extern_variables.h`-filen med `#include` kommandoen, med unntak av hovedkildekoden `main.c`.

De globale variablene kan derimot kun brukes i hovedkildekoden dersom de, i tillegg, deklareres i `variables.h`-filen, ettersom det er denne hjelpefilen som tas inn her.

I `variables.h`-filen er det blant annet initialisert variabler som brukes som konstanter. Disse listes opp nedenfor med beskrivelse av bruksområde.

```
uint8_t Test_Mode = 1;
```

Setter hvilket `case`-nummer sekvenskjøringen av systemet skal starte på. Beskrives i kapittel 6.2.1.

```
uint32_t differanse_margin = 800;
```

Avgjør hvor stor differanse trykkmålingene kan ha, for at det skal telles som samme trykk.

6.2 Programkode for styresystem

Frem til nå har man sett på filstrukturen, og hva de ulike filene inneholder. Hvordan programkoden for styresystemet fungerer og er skrevet, skal nå gjennomgås. Hovedprogrammet skal først gjennomgås, deretter blir avbruddsmetoden (`SysTick_Handler`) gjennomgått, og til slutt blir de ulike funksjonene, som blir kalt i hovedprogrammet, forklart.

6.2.1 Hovedprogram

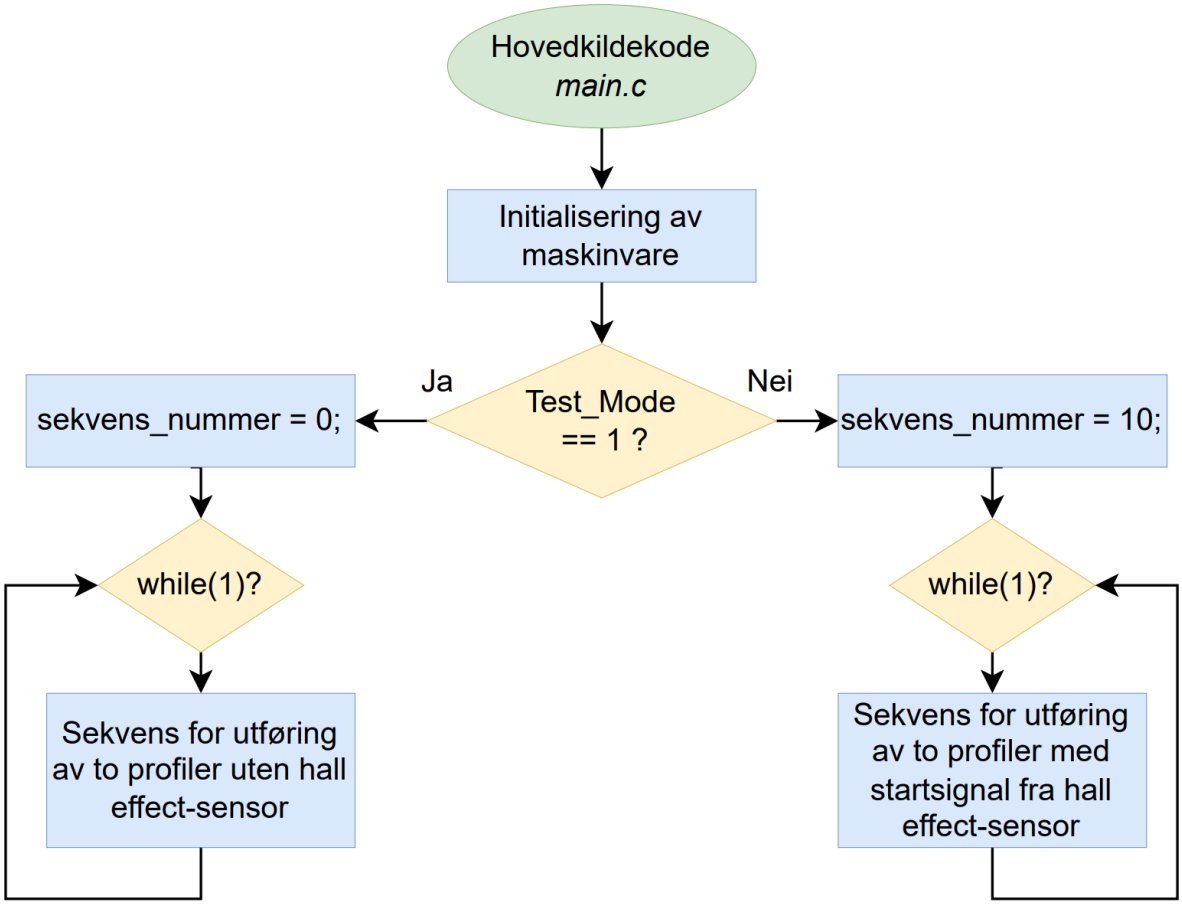
Hovedprogrammet er delt opp i to deler:

1. Testmodus
2. Konkurransemodus

Testmodus er aktivert når variabelen `Test_Mode` er satt til 1. Denne er laget for å enklere kunne teste flyteren i bassenget, uavhengig av ROV-en og et magnetfelt som viser at en er tilkoblet. Systemet kan da startes manuelt med trykkbryteren på bunnen av flyteren (SW2), eller med trykkbryteren på kretskortet (SW1), og kjører deretter i gang med utføring av to profiler med en gang.

Konkurransemodus er aktivert når variabelen `Test_Mode` er satt til 0, og er det programmet som skal kjøres under konkurransen. Som tidligere nevnt i kapittel 2.2, er det spesifisert i konkurransemanualen at flyteren er suksessfult satt ut av ROV-en, når ROV-en ikke lenger er i kontakt med flyteren. Derfor starter programmet med å vente på et detektert magnetfelt fra ROV-en, som videre gir startsignal for utføring av

to profiler, først når det er detektert at magnetfeltet er borte igjen. Konkurransemodus fungerer altså likt som testmodus, med unntak av at sekvensen ikke starter før magnetfeltet er oppdaget og så er borte igjen. I konkurransemodus, blir bryterne (SW1 og SW2) brukt som resett-brytere, i motsetning til i testmodus, hvor de blir brukt som startknapp. Et forenklet flytskjema for hovedkildekoden i `main.c`-filen, er vist i figur 75.



Figur 75: Overordnet flytskjema for hovedkildekoden.

Før `while`-løkka, blir det avgjort om systemet er satt i `Test_Mode` eller ikke. Deretter kjøres den koden som er spesifisert, altså testmodus (til venstre i figur 75), eller konkurransemodus (til høyre i figur 75). Koden kjøres så lenge mikrokontrolleren har strøm. For å avslutte kjøringen, kobles batteriene fra med vippebryteren på bunnen av flyteren, som forklart i kapittel 4.4.

Variabelen `Test_Mode` endres manuelt i `variables.h`-filen. Den eneste måten å bytte mellom de to modusene på, er å koble på programmeringskortet igjen, som forklart i kapittel 5.5.2, og endre variabelen `Test_Mode` til ønsket modus (0 eller 1) i *STM32CubeIDE*, for så å laste opp programmet til mikrokontrol-

leren på ny.

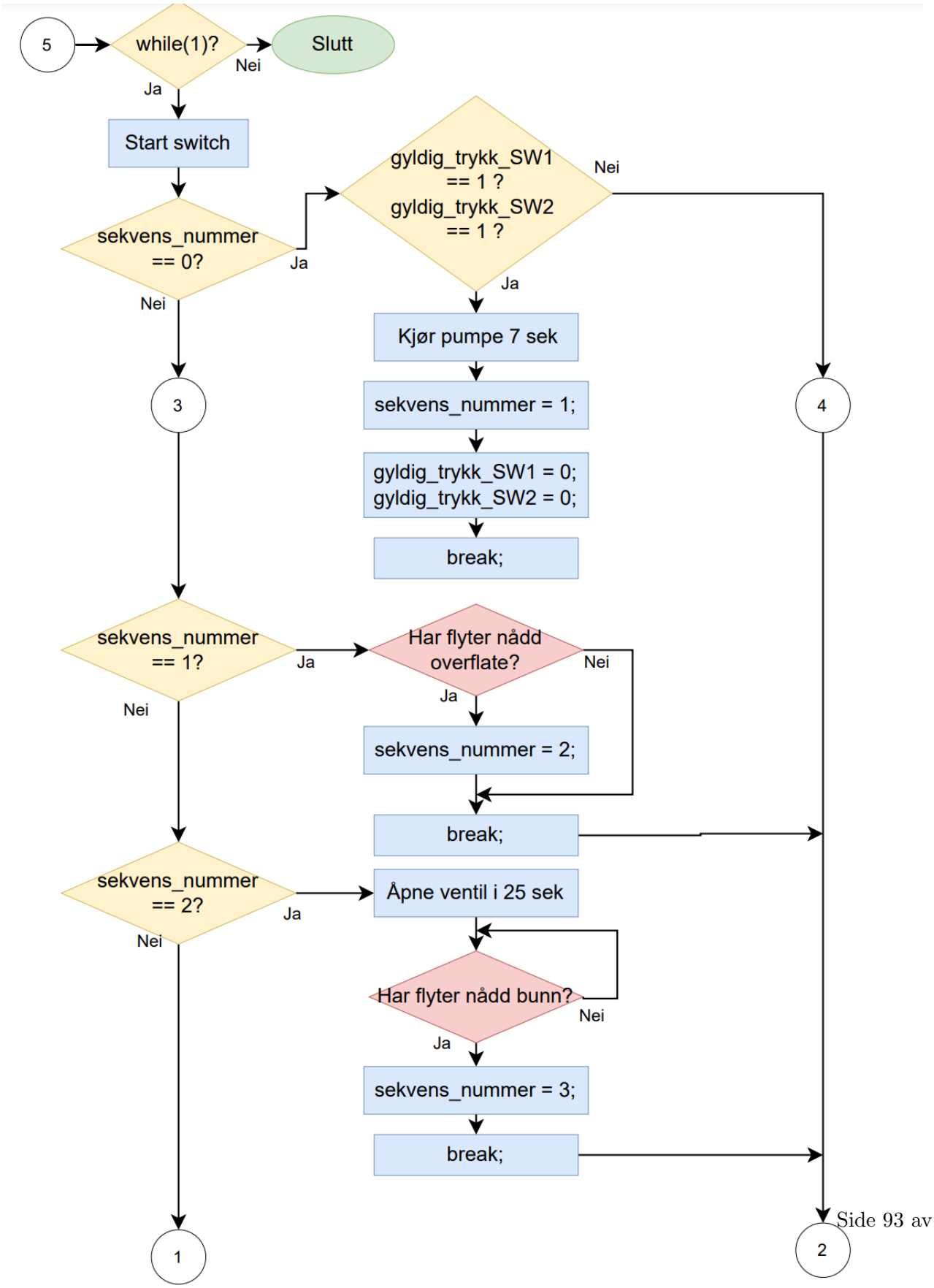
Det ble tidligere, i kapittel 2.1, forklart at flyteren skal utføre to profiler. Hovedprogrammet blir dermed kjørt sekvensielt. For å gjøre dette, er `switch/case`-funksjonen brukt. Denne funksjonen tester en variabel, `sekvens_nummer`, og går inn i den casen som har tilsvarende `case`-nummer som variabelen. Dette skal gjennomgås nøyere i forklaring av de to modusene.

Testmodus

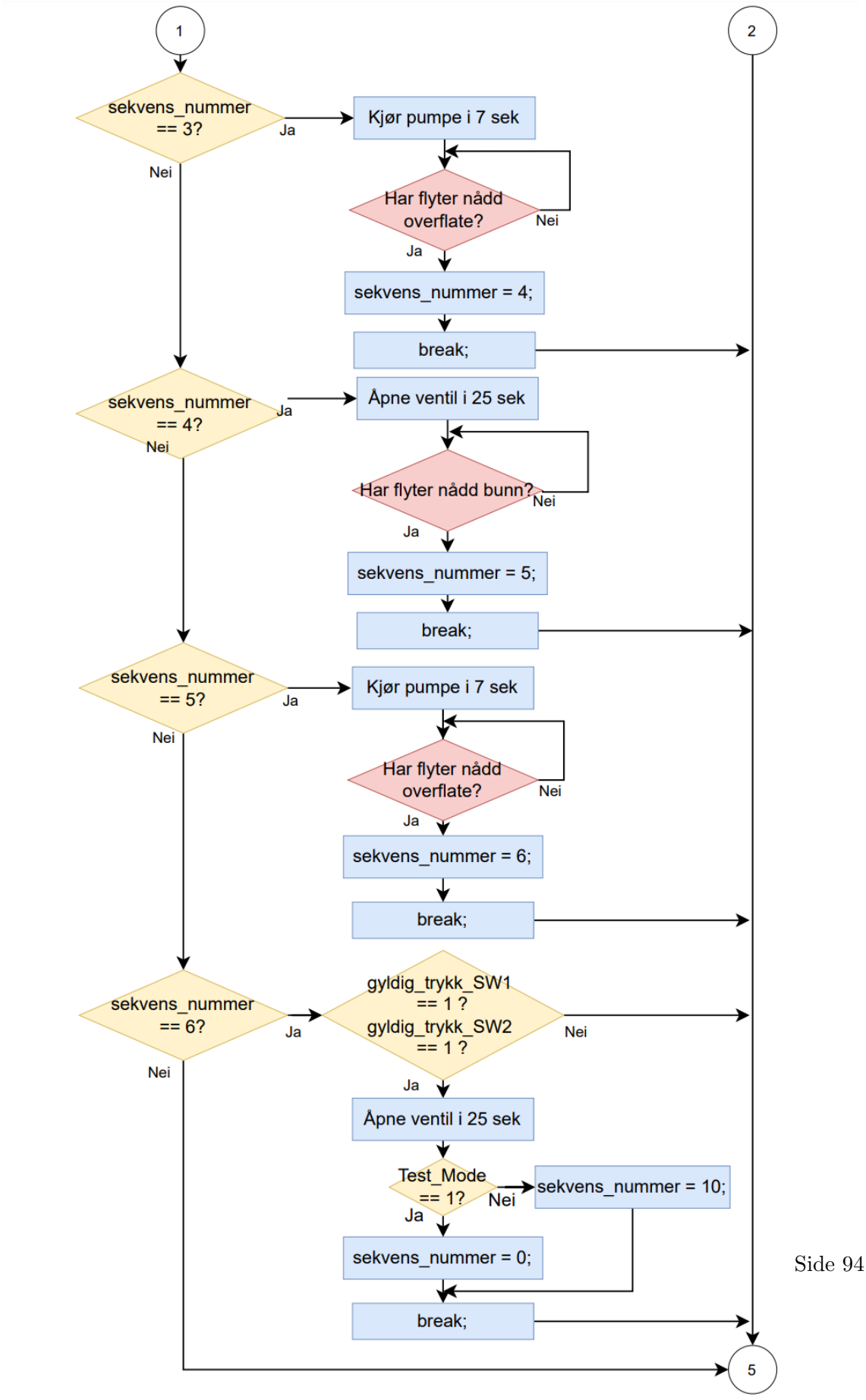
Dersom `Test_Mode` er satt til 1, vil variabelen `sekvens_nummer` være lik 0, og `switch/case`-funksjonen starter på `case 0`. Nedenfor er sekvenskjøringen for testmodus oppsummert.

`case 0:`	Dersom startknapp (SW1 eller SW2) er trykket, pump opp ballong for å få positiv oppdrift for å kunne starte på overflaten.
`case 1:`	Vent til flyteren har nådd overflaten, og til pumpa er av, med å starte med en profil.
`case 2:`	Synk til bunnen for første gang.
`case 3:`	Flyt opp til overflaten for første gang. Nå er en profil utført.
`case 4:`	Synk til bunnen igjen for andre gang.
`case 5:`	Flyt opp til overflaten igjen for andre gang. Nå er to profiler utført.
`case 6:`	Dersom startknapp (SW1 eller SW2) er trykket, åpne ventilen slik at ballongen tømmes før en eventuell ny sekvens startes. Returner til `case 0`.

Figur 76 og 77 viser et flytskjema over sekvenskjøringen i testmodus. Fra punkt 3 og 4 er det felles kode for begge modusene.



Figur 76: Flytskjema over sekvenskjøringen i testmodus og delvis konkurransemodus (fra punkt 3 og 4).

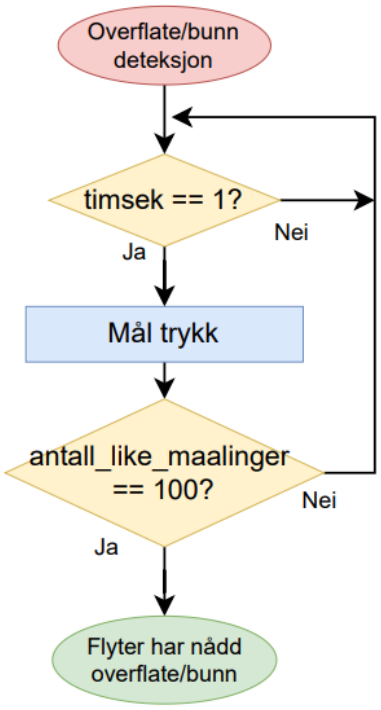


Figur 77: Flytskjema over sekvenskjøringen i både testmodus og konkurransemodus.

Programkoden forutsetter at systemet blir spenningssatt med en tom ballong, ettersom det første som skjer, når SW1 eller SW2 blir trykket, er at pumpa blåser opp ballongen. Etter flyteren har utført sine to profiler, blir ballongen værende oppblåst, for å kunne ligge flytende, klar til å bli hentet opp av vannet. Programkoden er nå i `case 6`, og venter på et nytt knappetrykk som åpner ventilen og tømmer ballongen. Deretter er systemet klart til ny start med tom ballong i `case 0`.

Start og åpning av henholdsvis pumpe og ventil, skjer i hovedprogrammet `main.c`. Videre er det lagt inn timere i avbruddsmetoden som stopper/lukker pumpe og ventil. Dette gjennomgås i kapittel 6.2.2, sammen med brytersjekk.

Underveis i sekvenskjøringen, blir det testet om flyteren har nådd bunn eller overflate, markert i rødt i figur 76 og 77. Denne testen er lik i alle tilfeller. Det er derfor laget som et eget lite flytskjema for å spare plass, og øke leseligheten. Flytskjemaet for denne testen er vist i figur 78.



Figur 78: Flytskjema for sjekk om flyter har nådd bunn eller overflate.

I avbruddsmetoden `SysTick_Handler`, blir `timsek` satt til 1 hvert 100 ms. Denne variabelen brukes for å måle trykket hvert 100 ms, altså hvert halve sekund. Deretter sjekkes det om det er målt 100 like målinger etter hverandre[4]. Variabelen `antall_like_maalinger` økes med 1 for hvert målt trykk som er innenfor

[4]En måling er lik den forrige målingen dersom absoluttverdien av differansen mellom dem er under en bestemt verdi. Verdien er definert i variabelen `differanse_margin`, vist i kapittel 6.1.2

den bestemte verdien fra det forrige målte trykket. Når `antall_like_maalinger` er 100, betyr det at flyteren har nådd bunn eller overflate, og kan gå videre til neste sekvens. I hovedkildekoden, utføres denne testen med funksjonen `Skal_FLyter_Snu()`. Denne blir mer detaljert gjennomgått i kapittel 6.2.3.

Konkurransemodus

Dersom `Test_Mode` er satt til 0, vil variabelen `sekvens_nummer` være lik 10. `Switch/case`-funksjonen starter dermed på `case 10`. Sekvenskjøringen i konkurransemodus er lik som i testmodus, når det gjelder utføring av to profiler. Forskjellen mellom disse to modusene, er hvilket startsignal som starter utførelsen av profilene, og hva bryterknappene brukes til. Nedenfor er sekvenskjøringen for konkurransemodus oppsummert, i kronologisk rekkefølge.

`case 10:` Resett system før start. Åpne ventilen i tilfelle ballongen allerede er oppblåst.

`case 11:` Når ventilen er lukket igjen, startes pumpa for å få positiv oppdrift før start.

`case 12:` Vent på et magnetfelt.

`case 13:` Magnetfelt er detektert, vent til det er borte.

`case 1:` Vent til flyteren har nådd overflaten med å starte med en profil.

`case 2:` Synk til bunnen for første gang.

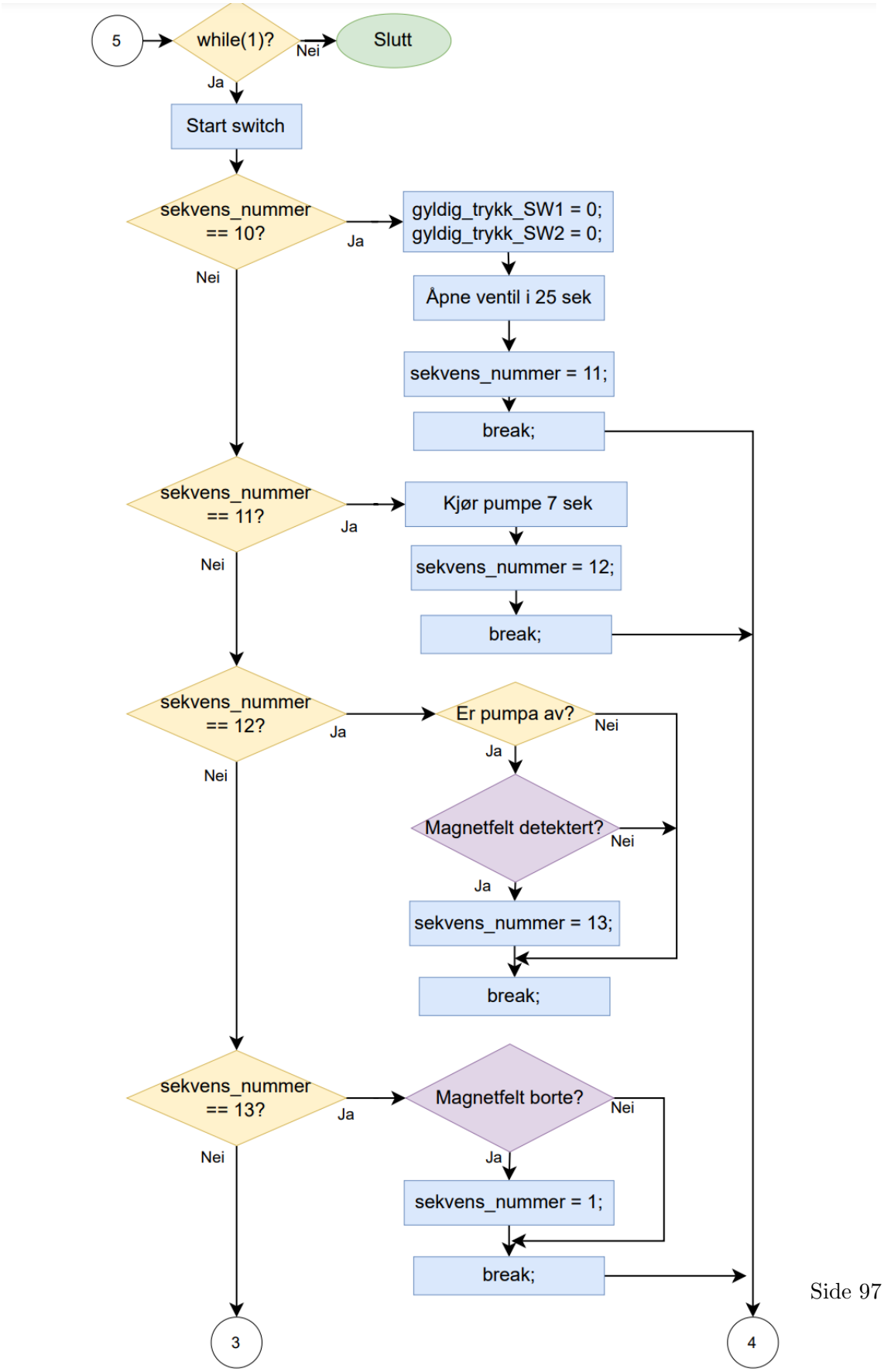
`case 3:` Flyt opp til overflaten for første gang. Nå er en profil utført.

`case 4:` Synk til bunnen for andre gang.

`case 5:` Flyt opp til overflaten for andre gang. Nå er to profiler utført. Returner til `case 12`.

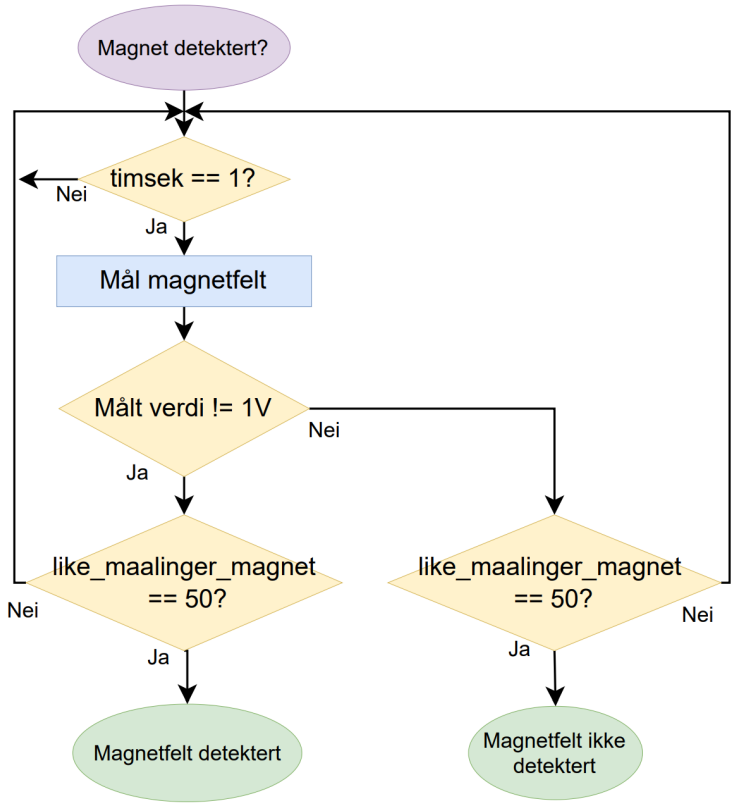
Under sekvenskjøring i konkurransemodus, kan bryterne (SW1 eller SW2) trykkes inn når som helst for å resette systemet. En vil da gå tilbake til `case 10`, som åpner ventilen og tømmer ballongen for eventuell luft, før den pumpes opp igjen, og begynner på ny utføring av to profiler.

Flytskjemaet, vist i figur 79, viser første del av sekvensen med startsignal fra Hall effect-sensoren, som detekterer magnetfelt. Deretter fortsetter flytskjemaet, for konkurransemodus, fra punkt 3 og 4 i figur 76, og til slutt i figur 77.



Figur 79: Flytskjema over sekvenskjøringen i konkurransemodus.

Før start av profilutføring, letes det etter et magnetfelt, markert i lilla i figur 79. Her letes det etter et oppdaget magnetfelt for første gang i `case 12`. Deretter ventes det til magnetfeltet er borte igjen i `case 13`. I praksis, tilsvarer dette når ROV-en tar tak i flyteren, for så å slippe den. Disse to testene er like, og vist i eget flytskjema i figur 80.

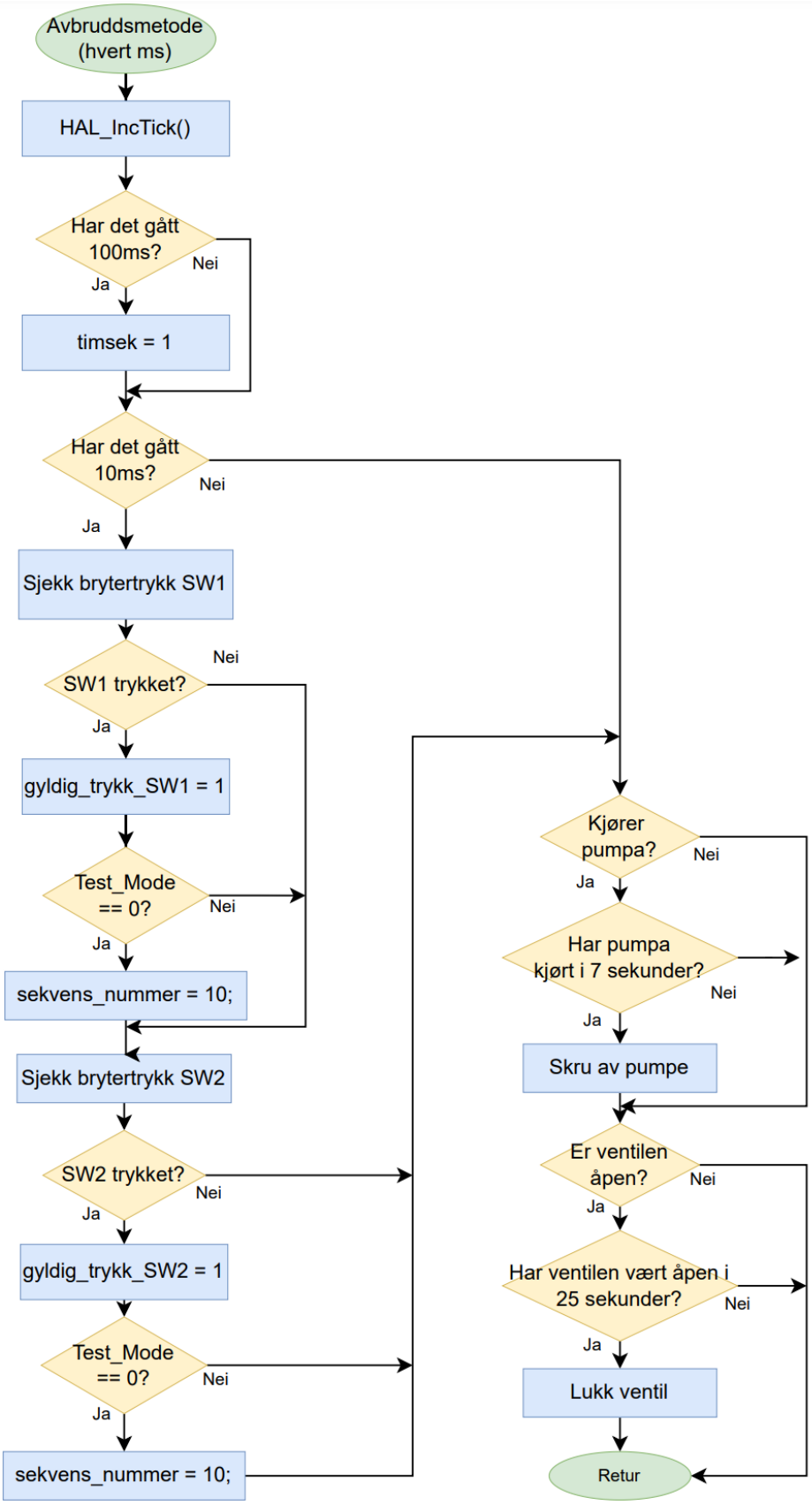


Figur 80: Flytskjema for sjekk av detektert magnetfelt.

For måling av magnetfelt brukes variabelen `timsek`, på samme måte som for måling av trykk. Det blir altså målt et magnetfelt hvert 100 ms. Det målte magnetfeltet blir målt i spenning, og gjort om til en digitalverdi i ADC-omformingen. Dette blir gjennomgått i mer detalj i kapittel 6.4.1. Ettersom Hall effect-sensoren gir ut 1 V dersom det ikke er noe magnetfelt i nærheten, sjekkes det, med en bestemt margin, om målt verdi tilsier at det er et magnetfelt i nærheten eller ikke.

6.2.2 Avbruddsmetode

Avbruddsmetoden som er tatt i bruk er `SysTick_Handler` fra `stm32f4xx_it.c`-filen. Denne kjøres hvert ms, og sjekker brytertrykk, i tillegg til å fungere som en timer for når pumpe og ventil skal skrus av, samt når trykksensoren og Hall effect-sensoren skal leses av. Flytskjema for avbruddsmetoden er vist i figur 81.



Figur 81: Flytskjema for avbruddsmetoden.

Avbruddsmetoden består av fire tellere som har hver sine funksjoner.

1. `count`
2. `TickTeller`
3. `pump_timer`
4. `valve_timer`

Den første telleren, `count`, teller til 100 før den setter variabelen `timsek` = 1. Denne brukes til å lese av trykk- og Hall effect-sensoren hvert 100 ms.

Den andre telleren, `TickTeller`, teller opp til 10, før den kaller på funksjonene `Sjekk_SW1()` og `Sjekk_SW2()`. Denne sjekker om en av bryterne, SW1 eller SW2, er blitt trykket inn. Det vil si at bryterne blir sjekket hvert tiende millisekund. Disse funksjonene blir mer detaljert forklart i kapittel 6.2.3.

Den tredje telleren, `pump_timer`, teller opp til 7000 dersom pumpa kjører. Deretter kalles funksjonen `Pump_Off()`, som skrur av GPIO-pinnen til mikrokontrolleren, som er koblet til solid state-releet som går til pumpa. Det vil si at når pumpa skrus på i `main.c`, starter `pump_timer` å telle, og skrur pumpa av etter syv sekunder.

Den fjerde telleren, `valve_timer`, virker likt som `pump_timer`, men reagerer på ventilen. Når ventilen åpnes i `main.c`, starter `valve_timer` å telle opp til 25000, som tilsvarer 25 sekunder. Deretter kalles funksjonen `Valve_Close()`, som lukker ventilen, på samme måte som `Pump_Off` skrur av pumpa.

6.2.3 Funksjoner

Funksjonene som blir brukt i `main.c` og `SysTick_Handler`, er samlet i `functions.c`. Her finner man funksjoner for å sette og skru av diverse LED-lys, skru av og på pumpe, åpne og lukke ventil, sjekke brytertrykk, og funksjonen for å teste om flyteren har nådd bunn eller overflate. Disse funksjonene skal nå gjennomgås i mer detalj.

Skru av og på programmerbare LED-lys

Kretskortet *Plattformgrensesnitt*, har fem LED-lys, inkludert de tre LED-lysene på plattformkortet, *STM32F4 ARM MINI*, der mikrokontrolleren er montert på. Blant disse er det D2, STAT og DATA-lyset som er programmerbare. D1 og PWR er koblet direkte på strømforsyningen, og indikerer at kretskortene, *Plattformgrensesnitt* og *STM32F4 ARM MINI*, er tilkoblet strøm. For å øke leseligheten til programkoden i `main.c`, er det laget ulike funksjoner for hvert av de programmerbare LED-lysene:

- `Set_D2(uint8_t bitstatus)`
- `Set_DATA_LED(uint8_t bitstatus)`
- `Set_STAT_LED(uint8_t bitstatus)`

Alle funksjonene tar inn en variabel, `bitstatus`, som settes ved kall på funksjonen. Denne kan settes lik 1, for å skru på et LED-lys, eller til 0, for å skru av et LED-lys. Ettersom alle funksjonene fungerer likt, vises kun `Set_D2()` her, i kode 2.

```
1 void Set_D2(uint8_t bitstatus){
2     if (bitstatus == 1){
3         HAL_GPIO_WritePin(D2_GPIO_Port, D2_Pin, GPIO_PIN_SET);
4     }
5     if (bitstatus == 0){
6         HAL_GPIO_WritePin(D2_GPIO_Port, D2_Pin, GPIO_PIN_RESET);
7     }
8 }
```

Kode 2: Funksjon for å skru av eller på LED-dioden D2 på *Plattformgrensesnitt*.

For å sette en GPIO-utgang, brukes funksjonen `HAL_GPIO_WritePin()`, fra HAL API-et, som er funnet på side 414 i [34]. Hva HAL er, og hvordan det brukes, blir forklart i kapittel 6.3. Denne funksjonen tar inn GPIO-port, GPIO-pinne og hvilken status pinnen skal settes til. Som vist i kodeutsnitt 2, ser man at dersom `bitstatus` er 1, er statusen pinnen skal settes til `GPIO_PIN_SET`.

Skru av og på utgang til pumpe og ventil

Utgangen til pumpa og ventilen, settes høy og lav på samme måte som LED-lysene. Funksjonene som gjør dette er også delt opp, for å øke leseligheten i programkoden:

- `Pump_On(void)`
- `Pump_Off(void)`
- `Valve_Open(void)`
- `Valve_Close(void)`

Funksjonene for pumpa, `Pump_On` og `Pump_Off`, bruker variabelen `Pump_Status`, og setter denne til 0 og 1 når pumpa skrus henholdsvis av og på, som vist i kodeutsnitt 3.

```
void Pump_On(void){
    HAL_GPIO_WritePin(SSR_Pump_GPIO_Port, SSR_Pump_Pin, GPIO_PIN_SET);
    Pump_Status = 1;
}

void Pump_Off(void){
    HAL_GPIO_WritePin(SSR_Pump_GPIO_Port, SSR_Pump_Pin, GPIO_PIN_RESET);
    Pump_Status = 0;
}
```

Kode 3: Funksjon for å skru av og på pumpa.

På samme måte bruker funksjonene for ventilen, `Valve_Open` og `Valve_CLose`, variabelen `Valve_Status` til samme formål. Det er på denne måten avbruddsmetoden, `SysTick_Handler`, kan se om utgangene til pumpa og ventilen er aktive eller ikke.

Sjekk av brytertrykk

Funksjonene for sjekk av brytertrykk, er delvis hentet fra prosjektet vårt i datamaskinkonstruksjon, høsten 2021 [28], med unntak av å bruke HAL API-et til å lese GPIO-inngangene. Funksjonen for sjekk av SW1, er vist i kode 4.

```
void Sjekk_SW1(void) {
    if( HAL_GPIO_ReadPin(SW1_GPIO_Port, SW1_Pin) == 0) {
        if(SW1_nettopp_sluppet){

            SW1_nettopp_sluppet = 0;
            gyldig_trykk_SW1 = 1;
        }
    }
    else {
        SW1_nettopp_sluppet = 1;
    }
}
```

Kode 4: Funksjon for å sjekke om bryter SW1 er trykket inn.

Ved sjekk av brytertrykk, har det ikke vært nødvendig å lage et digitalt filter, som filtrerer bort bryterprell. Dette er på grunn av RC-filteret som er koblet til bryterkretsen, og forklart i kapittel 5.5.1.

Funksjonen `HAL_GPIO_ReadPin()` som brukes, er også funnet på side 414 i [34]. Denne funksjonen tar inn GPIO-port og GPIO-pinne som parametre, og leser av spesifisert GPIO-pinne. Avhengig av om avlest GPIO-inngang er høy eller lav, returnerer funksjonen henholdsvis `GPIO_PIN_SET`, som er lik 1, eller `GPIO_PIN_RESET`, som er lik 0, i variabelen `bitstatus`. Kodeutsnittet for denne funksjonen, er vist i kode 5.

```
1  GPIO_PinState HAL_GPIO_ReadPin(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin)
2  {
3    GPIO_PinState bitstatus;
4
5    /* Check the parameters */
6    assert_param(IS_GPIO_PIN(GPIO_Pin));
7
8    if((GPIOx->IDR & GPIO_Pin) != (uint32_t)GPIO_PIN_RESET)
9    {
10     bitstatus = GPIO_PIN_SET;
11   }
12   else
13   {
14     bitstatus = GPIO_PIN_RESET;
15   }
16   return bitstatus;
17 }
```

Kode 5: Funksjon fra HAL API-et, funnet i `stm32f4xx_hal_gpio.c`-filen og dokumentert i [34].

Dersom returnert verdi er 0, vil det si at GPIO-inngangen er logisk lav, og motsatt. I følge bryterkretsen vår, som er vist i kapittel 5.5.1, er GPIO-inngangen logisk høy når bryteren er åpen, og blir dratt til jord (logisk lav) når byteren trykkes inn. Derfor sjekkes det i kode 4, linje 2, om returnert verdi fra `HAL_GPIO_ReadPin()` er lik 0, da dette tilsvarer et brytertrykk for vår krets. Videre, settes variabelen `gyldig_trykk_SW1` til 1, og kan brukes i hovedprogrammet.

Sjekk av om flyter har nådd bunn eller overflate

Funksjonen `Skal_Flyter_Snu()`, vist i kode 6, sjekker hver måling av trykk, fra trykksensoren, og sammenligner den nye målingen med den forrige målte verdien. Dette er for å sjekke om flyteren har nådd bunn eller overflate, som medfører at trykket slutter å endre seg.

```
void Skal_Flyter_Snu(uint32_t maalt_trykk, uint8_t nytt_sekvens_nummer ){
    int32_t differanse;

    forrige_trykk = nytt_trykk;
    nytt_trykk = maalt_trykk;

    // Når differanse i nåtrykk og forrigetrykk <= 800: antall like målinger ++
    differanse = nytt_trykk - forrige_trykk;
    if(differanse < 0){
        differanse = -differanse;
    }

    if(differanse <= differanse_margin ){
        antall_like_maalinger ++;
        if(antall_like_maalinger == 100){
            antall_like_maalinger = 0;
            sekvens_nummer = nytt_sekvens_nummer;
        }
    }
    else{
        antall_like_maalinger = 0;
    }
}
```

Kode 6: Funksjon for å sjekke om flyter har nådd bunn eller overflate

Funksjonen tar inn det nye målte trykket, `maalt_trykk`, og det nye `case`-nummeret, `nytt_sekvens_nummer`, som skal bli satt når flyteren har nådd bunn eller overflate.

Funksjonen lagrer det forrige trykket i en egen variabel, `forrige_trykk`, før den lagrer det nye målte trykket, i variabelen `nytt_trykk`. Deretter regnes differansen mellom de to trykkene ut, som blir sammenlignet med en bestemt margin, `differanse_margin`. Dersom absoluttverdien til den utregnede differansen er innenfor differansemarginen, blir det nye målte trykket regnet som samme trykk som forrige verdi, og telleren `antall_like_maalinger` inkrementeres med en. Når telleren har telt 100 like målinger etter hverandre, blir variabelen `sekvens_nummer` satt til det nye `case`-nummeret, som sendes inn ved kall på funksjonen.

6.3 Konfigurasjon av mikrokontrolleren

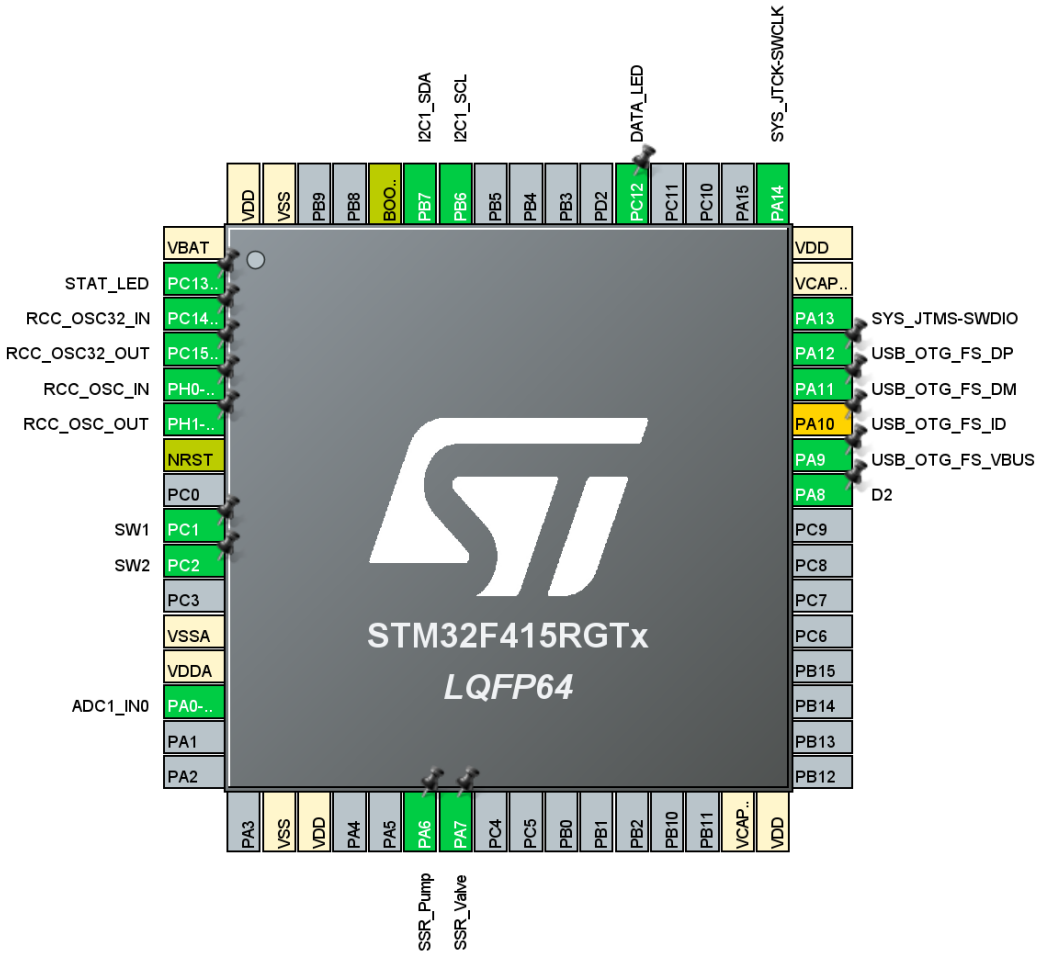
Til å konfigurere mikrokontrolleren, brukes *STM32Cube HAL* (Hardware Abstraction Layer). Dette er et abstraksjonslag med innebygde funksjoner, som utfører operasjoner på mikrokontrollerens registre og perifermoduler, ved å skrive direkte til registrene for oss. De modulene i mikrokontrolleren som skal konfigureres er, som nevnt i kapittel 2, GPIO-modulene, mikrokontrollerens interne klokke, ADC-modulen, I^2C-modulen og mikrokontrollerens USB-modul.

6.3.1 Konfigurasjon av pinnefunksjoner

Figur 82, viser en oversikt over hvordan pinnene til mikrokontrolleren er konfigurert. Denne oversikten finnes i *UiS Subsea Float.ioc*-filen, som er nederst i filstrukturen i kapittel 6.1.1. For å bestemme hvilken pin som skulle konfigureres til hva, ble skjemategningen fra databladet til plattformkortet, *STM32F4 ARM MINI*, brukt [24]. Skjemategningen er vist i figur 83 og blir gjennomgått senere i delkapittelet. Tabell 10 viser en oversikt over koblingene fra mikrokontrolleren til komponentene på *Plattformgrensesnitt*. Hvordan man bruker HAL til å konfigurere de ulike modulene, skal gjennomgås nå.

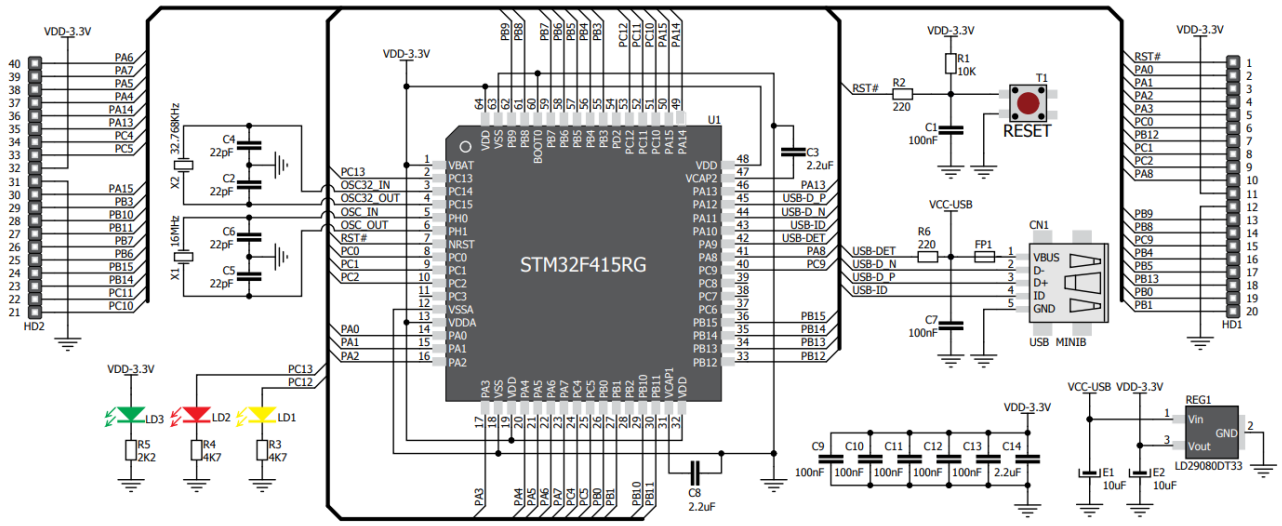
GPIO Pin	Plattform Pin	Komponent
PC1	J6_8	SW1
PC2	J6_9	SW2
PA6	J6_40	SSR for pumpe (U)
PA7	J6_39	SSR for ventil (U)
PA8	J6_10	D2
PA0 (ADC1)	J6_2	Hall-effect sensor (SENS1)
PB7 (I2C_SDA)	J6_26	Trykksensor (SENS2)
PB6 (I2C_SCL)	J6_25	Trykksensor (SENS2)

Tabell 10: Oversikt over hvilke pinner komponentene er koblet til på plattformkortet *STM32F4 ARM MINI*, og hvilken GPIO de tilhører.



Figur 82: Konfigurasjon av pinnefunksjoner.

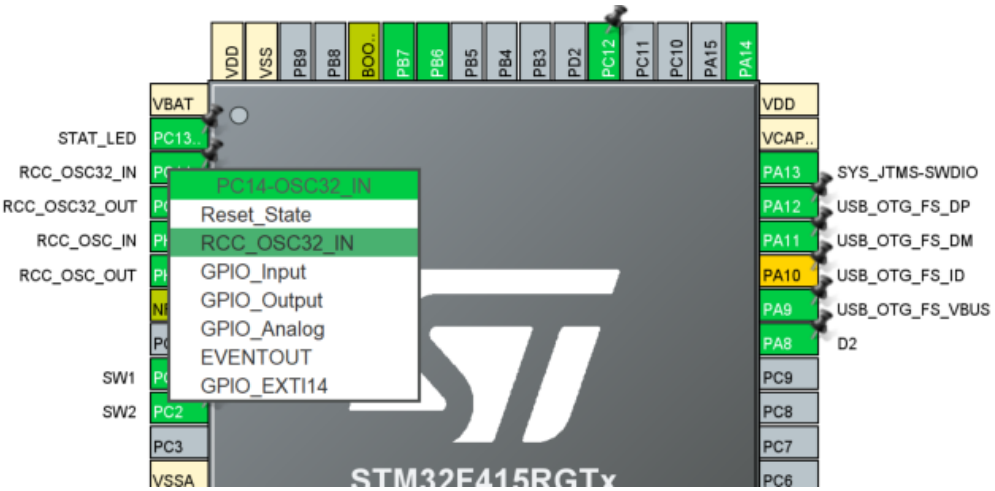
Ved å studere skjemategningen i figur 83, ser man at noen av pinnene til mikrokontrolleren allerede er koblet til egne komponenter. Blant annet tre LED-dioder (LD1, LD2 og LD3), en USB-tilkobling (CN1), to klokkegeneratorer (X1 og X2), en resettbryter (T1), samt diverse tilkoblinger til strømforsyning og jord. Dette er tilkoblinger som må konfigureres i mikrokontrolleren manuelt, ettersom kretskortet *STM32F4 ARM MINI* ikke er produsert av *STMicroelectronics* selv, men av *MikroElektronika*.



Figur 83: Kretsskjema for plattformkortet, hentet fra [24].

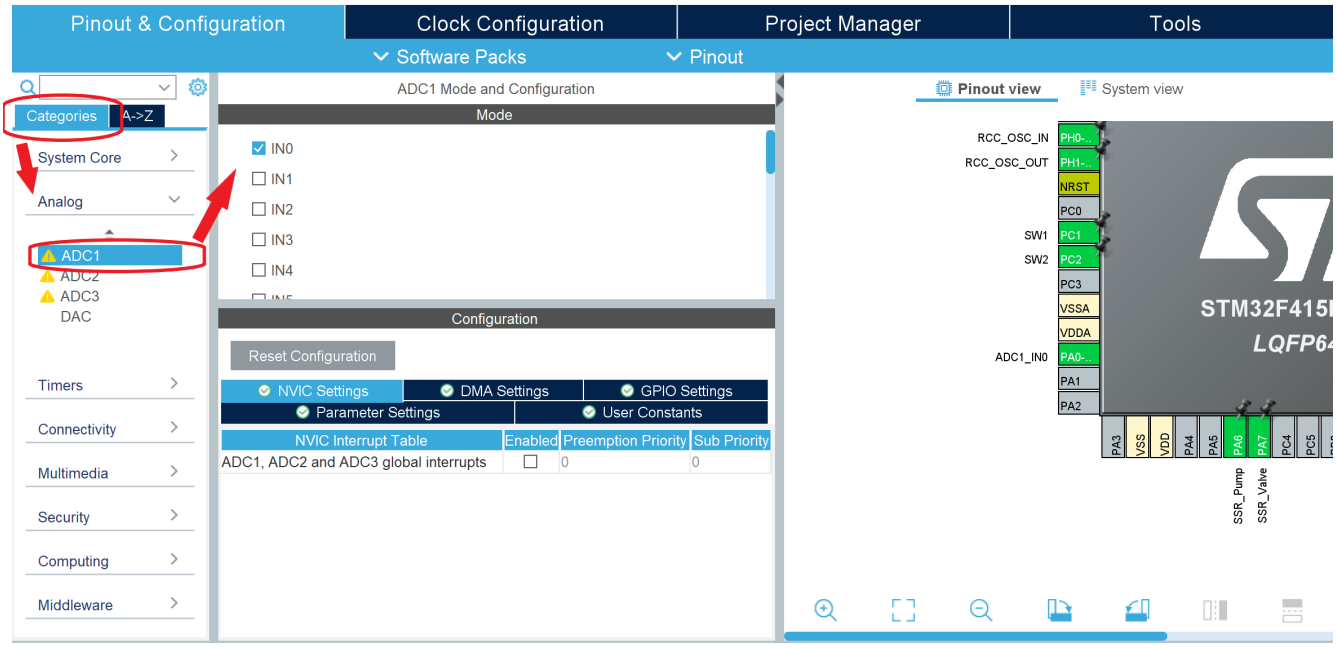
Man kan lese av, i skjemategningen, hvilken GPIO-pinne de ulike komponentene er koblet til, og kan velge samme GPIO-pinne i *Pinout & Configuration*-fanen i *STM32CubeIDE* (figur 82). Ved å venstreklikke på den GPIO-pinnen som skal konfigureres, kan ønsket konfigurasjon velges. GPIO-pinnene som skal brukes som inngang/utgang (for eksempel resettbryteren og LED-diodene), konfigureres til henholdsvis "GPIO_Input/GPIO_Output". Ved konfigurering av innganger og utganger, til kretskortet *Plattformgrensesnitt*, er det viktig å se i skjemategningen at de pinnene som velges her, er tilgjengelige og ledige. Man kan for eksempel se, i skjemategningen, at PC6, PC7 og PC8, ikke er koblet til pinnene på HD1 og HD2, som går videre til *Plattformgrensesnitt*.

Pinnene som skal ha spesialfunksjoner, som for eksempel klokkesignalet, USB-modulen, ADC-modulen og I^2C-modulen, konfigureres på samme måte, men her velges den spesialfunksjonen pinnen skal ha. For eksempel, er klokkegeneratorene koblet til PC14, PC15, PH0 og PH1. Venstreklikker man på PC14, i figur 82, får man valgene vist i figur 84. Her ser man at navnet på linjen, som går inn på PC14 i skjemategningen fra figur 83, stemmer overens med funksjonen, *RCC_OSC32_IN*, som kan velges i figur 84.



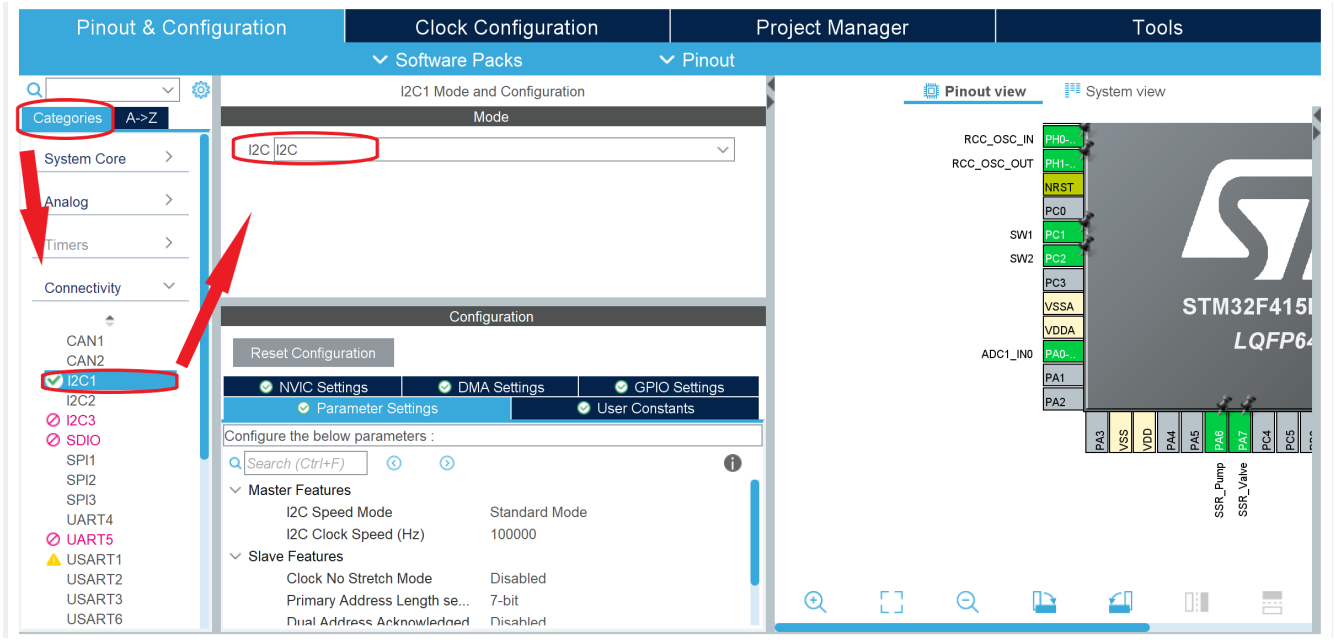
Figur 84: Konfigurasjon av klokkegeneratorene. Spesialfunksjonen tilgjengelig kommer automatisk opp.

På samme måte velges resten av spesialfunksjonene som tilhører de ulike pinnene. Når det gjelder ADC- og I^2C-modulen, kan man under *Categories* til venstre i *Pinout & Configuration*-fanen, konfigurere disse enkelt slik at de velger pinner på mikrokontrolleren selv. Dette er vist i figur 85, for ADC-modulen, og i figur 86, for I^2C-modulen.



Figur 85: Konfigurasjon av en ADC-modul.

For å konfigurere ADC-modulen, kan man klikke på *Analog*, og velge ADC1, som vist i figur 85, for så å huke av IN0. Når dette er gjort, vil den tilhørende pinnen konfigureres, og bli grønn, som vist i figur 82. Hvordan ADC-funksjonaliteten blir behandlet, blir gjennomgått i kapittel 6.4.1.

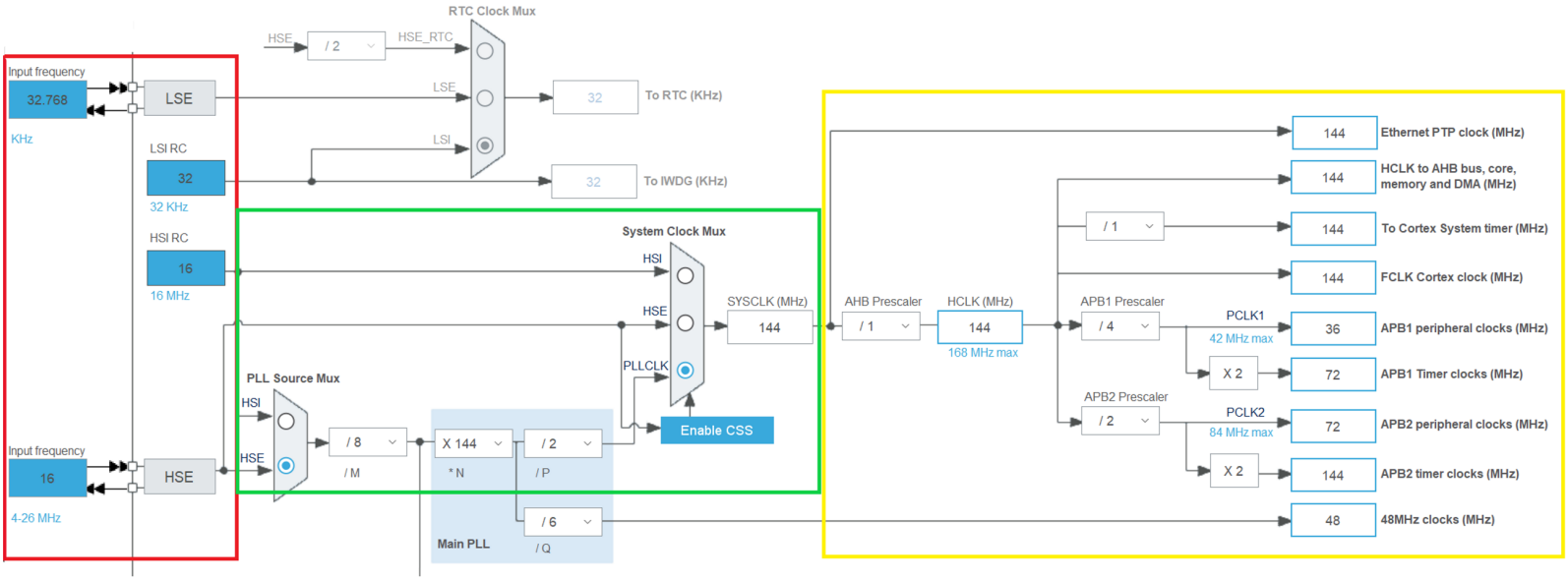


Figur 86: Konfigurasjon av en I^2C-modul.

På samme måte kan I^2C-modulen konfigureres. Under *Categories*, kan man under *Connectivity* velge I2C1, og så I2C i nedtrekksmenyen, som vist i figur 86. Når dette er gjort, vil de tilhørende pinnene konfigureres, som i figur 82 (PB6 og PB7), og bli grønne.

6.3.2 Klokkekonfigurasjon

Figur 87 viser klokkekonfigurasjonen. Rammet inn i rødt, har man fire klokkesignaler som innganger. Den vertikale linjen, i midten av den røde firkanten, markerer eksterne og interne klokkesignal i mikroprosessoren. **LSE** står for "*Low Speed External*", og **HSE** står for "*High Speed External*".



Figur 87: Oversikt over klokkekonfigurasjonen.

Systemklokken som prosessoren kjøres på vises i **SYSCLK**, helt til høyre i den grønne boksen. For å velge hvilken frekvens systemklokken til mikroprosessoren skal kjøre med, kan man velge den interne (HSI), den eksterne (HSE) eller **PLLCLK** (*"Phase-locked loop clock"*). Ved å velge **PLLCLK**, kan man generere et eget hurtigere klokkesignal.

For at denne skal bli konfigurert til 144 MHz, er inngangsfrekvensen til det eksterne **HSE**-klokkesignalet satt til 16 MHz, som videre inn i den grønne boksen blir dividert og multiplisert til 144 MHz, som vist i ligning 52.

$$\frac{HSE}{M} \cdot \frac{N}{P} = \frac{16}{8} \cdot \frac{144}{2} = 144\ MHz \tag{52}$$

Videre i den gule boksen, markert i figur 87, ser man, helt til høyre, klokkesignal til perifermodulene og timere som er i bruk. I databladet til mikrokontrolleren *STM32F415* [33], ser man (på side 20) et blokkskjema over de ulike modulene og bussene. Her står spesifikasjonene på maksimal frekvens for de ulike klokkesignalene til høyre.

6.4 Kommunikasjonsprotokoller

I dette delkapittelet, gjennomgås de ulike kommunikasjonsprotokollene som er brukt. Dette innebærer ADC-protokollen for å lese av Hall effect-sensoren, I^2C-protokollen for å lese av og skrive til trykksensoren, og til slutt litt om USB-kommunikasjonen mellom mikrokontroller og PC, som er brukt til testing.

6.4.1 ADC-protokoll for Hall effect-sensor

For å lese av verdiene til den analoge Hall effect-sensoren, brukes ADC-modulen for å konvertere analog-signal til digitalsignal. ADC-modulen til mikrokontrolleren har en oppløsning på 12 bit. For å vite hvor stort område dette tilsvarer i digital verdi, regnes 12 bit om til digital verdi i ligning 53.

$$2^{12} - 1 = 4096 - 1 = 4095 \tag{53}$$

Man har altså et område fra 0 til 4095 på den digitale utgangsverdien. Inngangen til mikrokontrolleren, har et spenningsområde fra 0 V til 3.3 V. Man kan dermed regne ut forholdet mellom inngangen og utgangen på ADC-modulen, som vist i utregning 54.

$$\begin{aligned} \text{Oppløsning} &= \frac{\text{Spenningsområde ADC}}{2^{\text{antall bit ADC}}} \\ &= \frac{3.3\ V}{2^{12}} \\ &= \frac{3.3\ V}{4096} \\ &= 8.05 \cdot 10^{-4} \approx 0.8\ mV \end{aligned} \tag{54}$$

Man vet dermed at for omtrent hver 0.8 mV, vil den digitale verdien øke med en, frem til det gis 3.3 V på inngangen til ADC-modulen. Da vil den digitale verdien være på 4095. Hall effect-sensorens måleområde er gitt i databladet [16], og er på $0.2\ V - 1.8\ V$, som vil si at hele måleområdet til ADC-modulen ikke blir brukt.

Hall effect-sensorens digitale verdi uten et detektert magnetfelt, $D_{1\ V}$, som tilsvarer 1 V i analog verdi, vil derfor være som vist i ligning 55.

$$\begin{aligned} D_{1\ V} &= \frac{1.0\ V}{3.3\ V} \cdot 4096 \\ &= 1241.21 \end{aligned} \tag{55}$$

Videre kan man også regne ut den minimale og maksimale digitale verdien som kommer som et resultat av måleområdet til Hall effect-sensoren, vist i ligning 56 og 57.

$$\begin{aligned} D_{0\ V} &= \frac{0.2\ V}{3.3\ V} \cdot 4096 \\ &= 248.24 \end{aligned} \tag{56}$$

$$\begin{aligned} D_{1.8\ V} &= \frac{1.8\ V}{3.3\ V} \cdot 4096 \\ &= 2234.18 \end{aligned} \tag{57}$$

Disse verdiene brukes til å verifisere det analoge signalet, under test av Hall effect-sensoren. Utregningen blir også brukt til å sette grenser for hva som gjelder som et magnetfelt, og ikke.

For å skrive kode for avlesing av ADC-modulen, er koden fra video [14], brukt til inspirasjon. Det er laget en funksjon, `MAGNETSENSOR_LesVerdi()`, i `functions.c`-filen, som kalles på i `main.c` når Hall effect-sensoren skal avleses. Koden for kall av funksjonen, er vist i kodeutsnitt 7.

```
1   magnetsens = MAGNETSENSOR_LesVerdi(&hadc1);
```

Kode 7: Kall på funksjon for lesing av Hall effect-sensor i `main.c`.

Her sendes structen `hadc1` som inneholder konfigurasjonsinformasjonen til ADC1, som er den spesifiserte ADC-modulen som skal brukes inn, for videre bruk i HAL API-et sin funksjon.

Kodeutsnitt 8, viser funksjonen som leser av Hall effect-sensoren. Denne finnes i `stm32f4xx_hal_adc.c`-filen.

```
1  uint16_t MAGNETSENSOR_LesVerdi(ADC_HandleTypeDef *hadc1){
2      uint16_t raaverdi;
3
4      HAL_ADC_Start(hadc1);
5      raaverdi = HAL_ADC_GetValue(hadc1);
6      return raaverdi;
7  }
```

Kode 8: Funksjon for lesing av Hall effect-sensoren gjennom ADC-modulen på mikrokontrolleren.

Den første funksjonen fra HAL API-et som brukes, `HAL_ADC_Start()`, aktiverer ADC-modulen og starter konvertering av de verdiene som blir tatt inn. Den andre funksjonen fra HAL API-et som brukes, `HAL_ADC_GetValue()`, henter ut den konverterte verdien fra dataregisteret det ble lagret i, av den første funksjonen. Deretter returneres den konverterte verdien og blir lagret i variabelen `magnetsens`.

Det siste som kan gjøres med denne råverdien, for å øke leseligheten i koden, er å gjøre om den digitale verdien til mV, som vist i ligning 58.

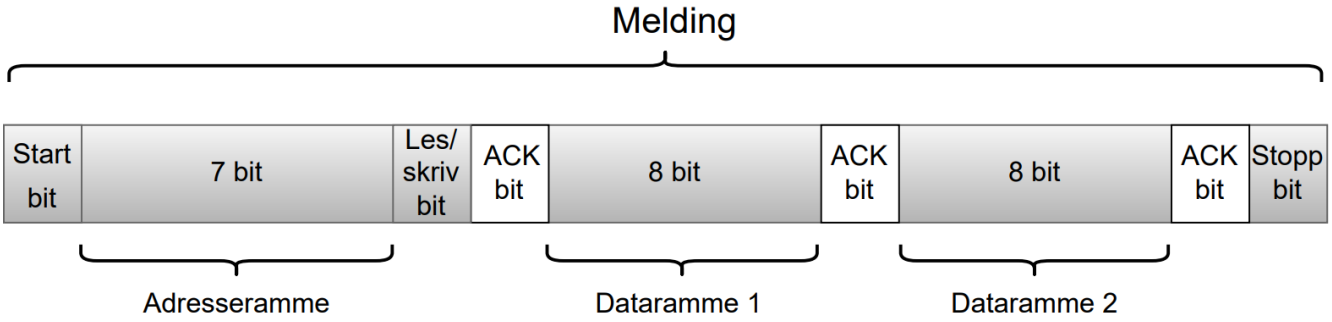
$$Verdi\ i\ mV = \frac{Digital\ verdi}{4096} \cdot 3300\ mV \tag{58}$$

På denne måten vil man kunne legge inn grenser for verdiene ved å referere til mV, istedenfor en digital verdi. For å sikre full presisjon ved å gjøre dette i *STM32CubeIDE*, må den nye variabelen være av datatype `float`, slik at desimaltall ikke blir rundet opp til heltall. Dette er vist i kapittel 7.5. Her har

man ikke regnet det om til mV i *STM32CubeIDE*, men i *Python*, etter at dataen er mottatt, ved bruk av seriell kommunikasjon.

6.4.2 I^2C-protokoll for trykksensor

For å kommunisere med trykksensoren, brukes det I^2C-protokoll (Inter-Integrated Circuit på engelsk). Det er en synkron kommunikasjonsprotokoll mellom to enheter, en master og en slave. Kommunikasjonsprotokollen går på to linjer, SDA (Serial Data) og SCL (Serial Clock). SDA er linjen som sender og mottar data. SCL er linjen med klokkesignalet som er styrt av masteren. Med I^2C-protokoll, sendes data i form av meldinger. Meldingene er delt opp i "rammer" med data, som vist i figur 88. Hver melding har en adresseramme, som inneholder den binære adressen til slaven, og en eller flere datarammer, som inneholder dataen som skal overføres. Hver melding inneholder også et startbit, stoppbit, les-/skrivbit og acknowledge-/not acknowledge-bit [4].



Figur 88: En typisk melding med I^2C-protokoll.

En melding starter alltid med et startbit, og deretter følger adresserammen, som forteller slaven at det er den som skal motta dataen som sendes. På slutten av adresserammen, er det inkludert et les/skriv bit, som forteller slaven om masteren vil skrive data til slaven, eller lese data fra slaven. Etter denne første adresserammen er sendt til slaven, sender slaven et bit tilbake til masteren som bekrefter eller avkrefter kommunikasjon. Dette er acknowlegde-bitet. Når masteren detekterer et acknowledge-bit fra slaven, er den første datarammen klar til å bli sendt. Datarammen består alltid av åtte bit, og er sendt med MSb (Most Significant bit) først. For hver dataramme mottatt, blir det øyeblikkelig sendt et acknowlegde-bit tilbake.

I dette systemet, er trykksensoren slave, mens mikrokontrolleren er master. Det må lages en driver til trykksensoren som sender meldinger (som forklart over) til trykksensoren og som lagrer mottatt data, som her er målt trykk. Dette gjennomgås i neste delkapittel.

Driver til trykksensor

Driveren til trykksensoren er hovedsaklig laget av oss, men med bidrag fra Jørgen Hemnes Johannessen fra sensorgruppa. Databladet til trykksensoren [6] inneholder trykksensorens adresse, som er den første adresserammen som skal sendes i en melding. Slaveadressen består av syv bit, og er avhengig om Chip Select-pinnen (CSB) er satt høy eller lav, vist i figur 89. Slaveadressens LSb (Least Significant Bit) er den inverterte av Chip Select-pinnen.

Pin CSB	Address (7 bits)
High	0x76 (1110110 b)
Low	0x77 (1110111 b)

Figur 89: Slaveadresse med høy og lav Chip Select-pin. Hentet fra [6].

I vårt system er CSB satt lav, dermed er trykksensorens adresse, i heksadesimaltall, 0x77. Som tidligere nevnt, består adressen av syv bit, men adresserammen som sendes til trykksensoren er på åtte bit. Det siste bitet er les/skriv bit (1/0), avhengig av om masteren vil lese fra eller skrive til trykksensoren. Ettersom slaveadressen kun består av syv bit, og det skal legges til et ekstra les/skriv bit på slutten av adressen (LSb), har man, i koden, utført en bitvis skiftoperasjon[5] til høyre på adressen, som legger til en 0 på slutten av adressen. Adressen til trykksensoren er definert i `trykksensor.h`-filen, som vist i kode 9. Deretter vil HAL-funksjonene, som brukes senere til å sende kommando til trykksensoren, automatisk sette LSb til 1 eller 0, basert på hvilken kommando som sendes til trykksensoren.

```
1 // DEFINES
2 #define TRYKKSENSOR_DEVICE          (0x77 << 1)
```

Kode 9: Definering av adressen til trykksensoren, både for lesing og skriving.

Databladet [6] beskriver også de ulike kommandoene som kan sendes til trykksensoren. Kommandoene fra databladet gjengis her:

- *Reset*
- *Read PROM (128 bit of calibration words)*
- *D1 conversion*
- *D2 conversion*
- *Read ADC result (24 bit pressure / temperature)*

[5] En bitvis skiftoperajson flytter hvert bit en plass til høyre eller venstre, avhengig av spesifisert retning.

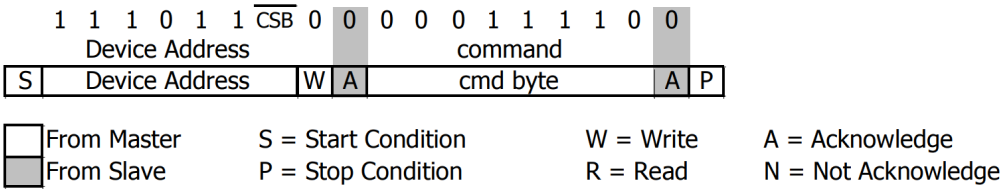
Reset-kommandoen resetter SDA-linjen slik at denne, dersom den venter på et acknowledge-bit, åpnes igjen for nye kommandoer. Read PROM-kommandoen blir brukt til å lese av fabrikkalibreringsdataen, som er blant annet konstantene C1 til C6. D1 og D2 conversion-kommandoen, skrives til trykksensoren for å gi beskjed om hvilken data man vil ha ut, og hvilken oppløsning man vil ha dataen ut i. Dette må sendes før Read ADC-kommando sendes. Vi kommer tilbake til hvordan dette er utført i praksis, når koden beskrives.

Hver kommando består også av åtte bit, og blir sendt som en dataramme etter adresserammen er sendt, og et acknowledge-bit er mottatt. Tabellen under, i figur 90, viser en oversikt over adressene, i heksadesimaltall, til hver kommando. Alle kommandoadressene er definert som egne konstanter i `trykksensor.h`-filen, som er vedlagt. Her ser man også at ulik oppløsning av D1 og D2 har egne adresser.

	Command byte								**hex value**
Bit number	0	1	2	3	4	5	6	7	
Bit name	PR M	COV	-	Typ	Ad2/ Os2	Ad1/ Os1	Ad0/ Os0	Stop	
Command									
Reset	0	0	0	1	1	1	1	0	0x1E
Convert D1 (OSR=256)	0	1	0	0	0	0	0	0	0x40
Convert D1 (OSR=512)	0	1	0	0	0	0	1	0	0x42
Convert D1 (OSR=1024)	0	1	0	0	0	1	0	0	0x44
Convert D1 (OSR=2048)	0	1	0	0	0	1	1	0	0x46
Convert D1 (OSR=4096)	0	1	0	0	1	0	0	0	0x48
Convert D2 (OSR=256)	0	1	0	1	0	0	0	0	0x50
Convert D2 (OSR=512)	0	1	0	1	0	0	1	0	0x52
Convert D2 (OSR=1024)	0	1	0	1	0	1	0	0	0x54
Convert D2 (OSR=2048)	0	1	0	1	0	1	1	0	0x56
Convert D2 (OSR=4096)	0	1	0	1	1	0	0	0	0x58
ADC Read	0	0	0	0	0	0	0	0	0x00
PROM Read	1	0	1	0	Ad2	Ad1	Ad0	0	0xA0 to 0xAE

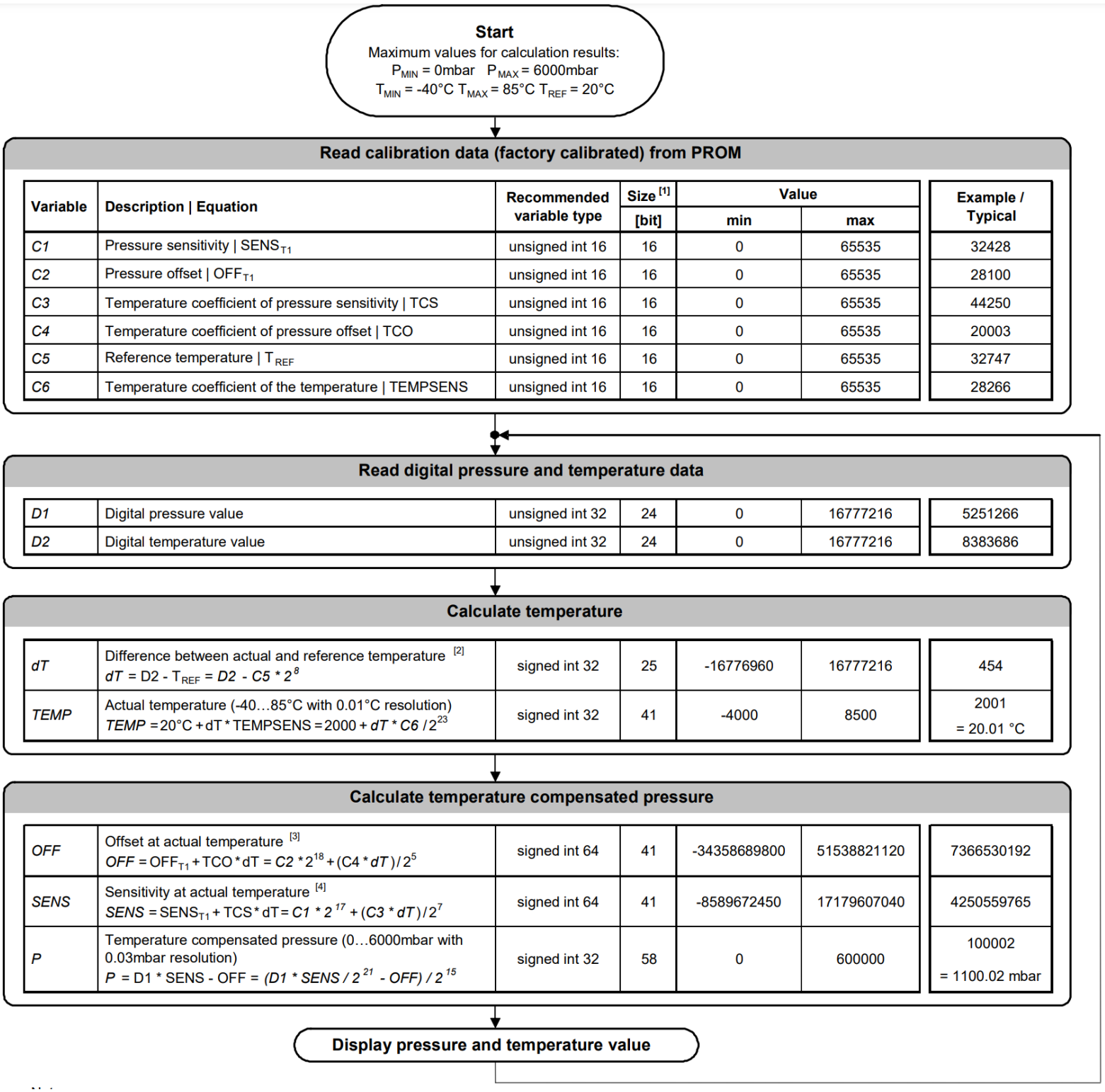
Figur 90: Oversikt over adressene til de ulike kommandoene som kan sendes til trykksensoren. Hentet fra [6].

For å lese trykket målt fra trykksensoren, skal kommandoene sendes i en spesifikk rekkefølge, som også er spesifisert i databladet [6]. Ved initialisering av trykksensoren, sendes det først en Reset-kommando for å resette SDA-linjen, i tilfelle denne venter på et acknowledge-signal som ville blokkert for andre kommandoer å komme gjennom. Reset-sekvensen er vist i figur 91.



Figur 91: Oversikt over Reset-sekvens. Hentet fra [6].

Databladet [6] har et flytskjema for lesing av konstanter og variabler, samt formler for utregning og omgjøring av trykk. Flytskjemaet er vist i figur 92, og skal videre gjennomgås steg for steg.



Variable	Description | Equation	Recommended variable type	Size [1] [bit]	Value min	Value max	Example / Typical
C1	Pressure sensitivity | $SENS_{T1}$	unsigned int 16	16	0	65535	32428
C2	Pressure offset | OFF_{T1}	unsigned int 16	16	0	65535	28100
C3	Temperature coefficient of pressure sensitivity | TCS	unsigned int 16	16	0	65535	44250
C4	Temperature coefficient of pressure offset | TCO	unsigned int 16	16	0	65535	20003
C5	Reference temperature | T_{REF}	unsigned int 16	16	0	65535	32747
C6	Temperature coefficient of the temperature | TEMPSENS	unsigned int 16	16	0	65535	28266
D1	Digital pressure value	unsigned int 32	24	0	16777216	5251266
D2	Digital temperature value	unsigned int 32	24	0	16777216	8383686
dT	Difference between actual and reference temperature [2] $dT = D2 - T_{REF} = D2 - C5 * 2^8$	signed int 32	25	-16776960	16777216	454
TEMP	Actual temperature (-40…85°C with 0.01°C resolution) $TEMP = 20°C + dT * TEMPSENS = 2000 + dT * C6 / 2^{23}$	signed int 32	41	-4000	8500	2001 = 20.01 °C
OFF	Offset at actual temperature [3] $OFF = OFF_{T1} + TCO * dT = C2 * 2^{18} + (C4 * dT) / 2^5$	signed int 64	41	-34358689800	51538821120	7366530192
SENS	Sensitivity at actual temperature [4] $SENS = SENS_{T1} + TCS * dT = C1 * 2^{17} + (C3 * dT) / 2^7$	signed int 64	41	-8589672450	17179607040	4250559765
P	Temperature compensated pressure (0…6000mbar with 0.03mbar resolution) $P = D1 * SENS - OFF = (D1 * SENS / 2^{21} - OFF) / 2^{15}$	signed int 32	58	0	600000	100002 = 1100.02 mbar

Figur 92: Flytskjema for lesing av konstanter, variabler og utregning av trykk. Hentet fra [6].

Etter Reset-kommandoen er sendt, kan konstantene C1 til C6 leses av. Dette skjer i funksjonen `TRYKKSENSOR_LesKalibrering()`, som er i `trykksensor.c`-filen. Alle konstantene leses av på samme måte, så derfor vises kun en avlesing her. Første del av funksjonen `TRYKKSENSOR_LesKalibrering()`, er vist i kode 10.

```
trykk_status stat;
uint8_t SENS_T1_buf[2];            // pressure sensitivity buffer
// [...]

// Les pressure sensitivity
stat = TRYKKSENSOR_LesRegistrene(dev, TRYKKSENSOR_PROM_C1, (uint8_t *)SENS_T1_buf, 2);

if(stat != TRYKK_SUKSESS){
    trykkfeil += 1;
}

// [...]

// Samle leste verdier til 16-bits variabler.
dev->SENS_T1 = (SENS_T1_buf[0]<<8|SENS_T1_buf[1]);
```

Kode 10: Kode for lesing av konstanten C1 som ligger i PROM-registeret i trykksensoren.

Her initialiseres først en 8-bits array, *SENS_T1_buf*, med en lengde på to bytes. Denne skal fungere som en buffer for dataen som leses inn fra trykksensoren. I funksjonen `TRYKKSENSOR_LesRegistrene()`, tas kommandoadressen inn til å lese konstanten C1, `TRYKKSENSOR_PROM_C1`, samt bufferen som dataen som skal leses, blir lagt inn i. Denne funksjonen er vist i kode 11.

```
trykk_status TRYKKSENSOR_LesRegistrene(TRYKKSENSOR *dev, uint8_t reg, uint8_t *data,
uint8_t lengde){
HAL_StatusTypeDef status;

status = HAL_I2C_Mem_Read(dev->i2cHandle, TRYKKSENSOR_DEVICE, reg, I2C_MEMADD_SIZE_8BIT,
data, lengde, HAL_MAX_DELAY);

if(status != HAL_OK) return TRYKK_FEIL;

return TRYKK_SUKSESS;
}
```

Kode 11: Funksjonen som kalles ved lesing av registerene i trykksensoren.

Denne funksjonen bruker HAL API-et sin innebygde funksjon, `HAL_I2C_Mem_Read`, til å sende både slave-adressen, lesbit (som settes automatisk av funksjonen, basert på om det er en les- eller skrivkommando), og kommandoadressen til trykksensoren.

Videre, i kode 10, legges dataen, som foreløpig er samlet i en 2 x 8-bits array, inn i en 16-bits variabel `SENS_T1`, ved hjelp av bitvise skiftoperasjoner. På samme måte, leses resten av variablene C2 til C6 av.

Nå som fabrikkalibreringsdataen er avlest, er neste steg å lese av trykket i ADC-registeret. Som tidligere nevnt, må man sende en skrivkommando til trykksensoren om hva man vil lese av, og i hvilken oppløsning. Det er ønskelig å lese av både trykk (D1) og temperatur (D2), ettersom man kan bruke temperaturen etterhvert for å regne ut et mer nøyaktig trykk. Disse avleses på samme måte, derfor går man kun gjennom avlesningen av trykket her. I kode 12 er koden for avlesning av trykksensoren vist.

```
uint8_t D1_buf[3];            // Digital pressure value buffer
uint8_t trykkbuf[1];
// [...]

trykkbuf[0] = TRYKKSENSOR_D1_256;

// [...]

// Les Digital pressure value og legg det i D1buf
stat = TRYKKSENSOR_SkrivRegister(dev, (uint8_t *)trykkbuf);

if(stat != TRYKK_SUKSESS){
    trykkfeil += 1;
}
HAL_Delay(10);

stat = TRYKKSENSOR_LesRegistrene( dev, TRYKKSENSOR_ADC_READ, (uint8_t *)D1_buf, 3 );

if(stat != TRYKK_SUKSESS){
    trykkfeil += 1;
}

// [...]

// Samle leste verdier til 32-bits variabler.
dev->D1 = (D1_buf[0]<<16|D1_buf[1]<<8|D1_buf[2]);
// [...]
```

Kode 12: Kode for avlesning av trykk.

Her skrives først en D1 conversion-kommando til trykksensoren, med en oppløsning på 256. Dette er den laveste oppløsningen trykksensoren har, men som gir raskest konverteringstid. Deretter er det lagt inn en tidsforsinkelse, som skal kompensere for konverteringstiden. Trykksensoren har nå fått beskjed om å konvertere dataen i ADC-registeret om til trykk med oppløsning på 256. Dermed sendes det en ny kommando, `TRYKKSENSOR_ADC_READ`, som leser av ADC-registeret. Dataene som kommer tilbake, 8 bit av gangen, lagres i arrayet `D1_buf`. Til slutt, legges verdiene i `D1_buf` sammen, ved hjelp av skiftoperasjoner til en 32-bits variabel `D1`.

På samme måte, leses temperaturen av, men ved bruk av D2 istedenfor D1. Deretter er man klar til å følge resten av flytskjemaet, i figur 92, og regne ut temperaturkompensert trykk. Dette gjøres i flere steg, som vist i kode 13.

```
// Regn ut temperaturen
dev->dT = dev->D2 - (dev->TREF *pow(2,8));         // Formel fra datablad.
dev->TEMPt = 2000 + (dev->dT*dev->TEMPSENS/pow(2,23));  // Formel fra datablad.
dev->temp_f = dev->TEMPt/100.0;     // Faktisk temperatur i degC

// Regn ut temperaturkompensert trykk
dev->OFF = (dev->OFF_T1*pow(2,18)) + ((dev->TCO*dev->dT)/pow(2,5)); // Formel fra datablad
dev->SENS = (dev->SENS_T1*pow(2,17)) + (dev->TCS*dev->dT)/pow(2,7); // Formel fra datablad.

// [...]

dev->trykk_comp = ((dev->D1*dev->SENS/pow(2,21)) - dev->OFF)/pow(2,15); // Formel fra datablad.
dev->trykk_f = dev->trykk_comp/100.0; // Faktisk trykk i mbar
```

Kode 13: Utregning av trykk. Uten 2.ordens temperaturkompensasjon.

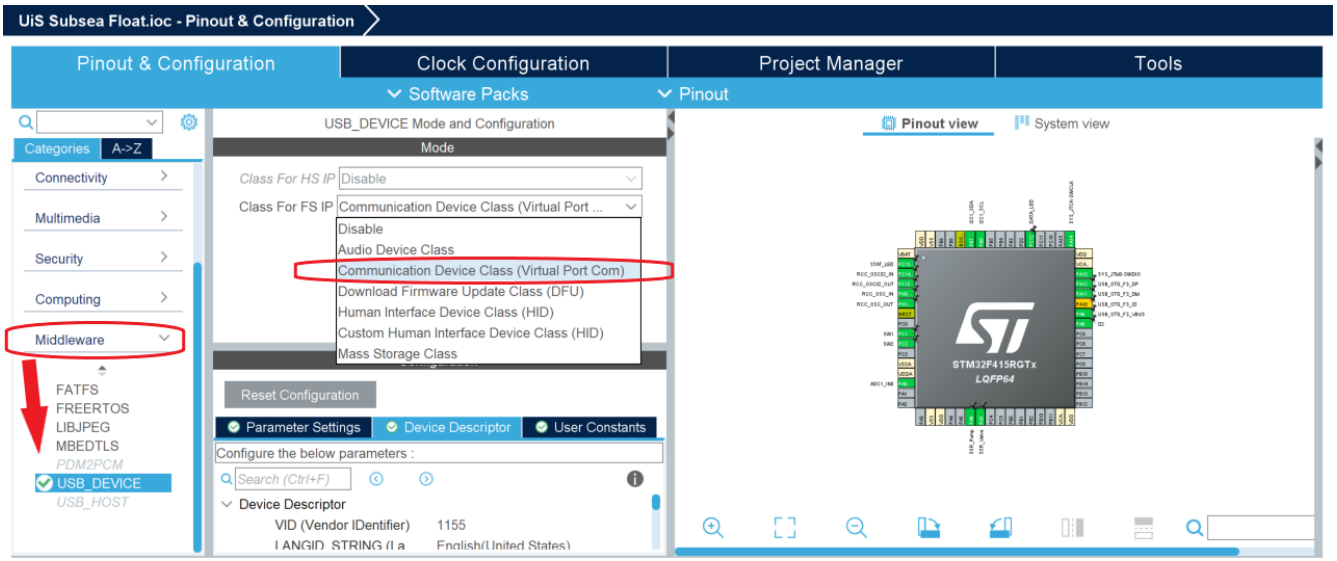
Her er formlene fra siste del av figur 92, direkte implementert i koden. Kommandoen `pow(x, y)`, tilsvarer den matematiske operasjonen, x^y. Variablene som regnes ut, er listet opp nedenfor med forklaring.

- `dT` - Differanse mellom faktisk- og referansetemperatur
- `TEMPt` - Faktisk temperatur
- `temp_f` - Faktisk temperatur i grader *Celcius*
- `OFF` - Offset ved faktisk temperatur
- `SENS` - Sensitivitet ved den faktiske temperaturen
- `trykk_comp` - Temperaturkompensert trykk
- `trykk_f` - Temperaturkompensert trykk i millibar

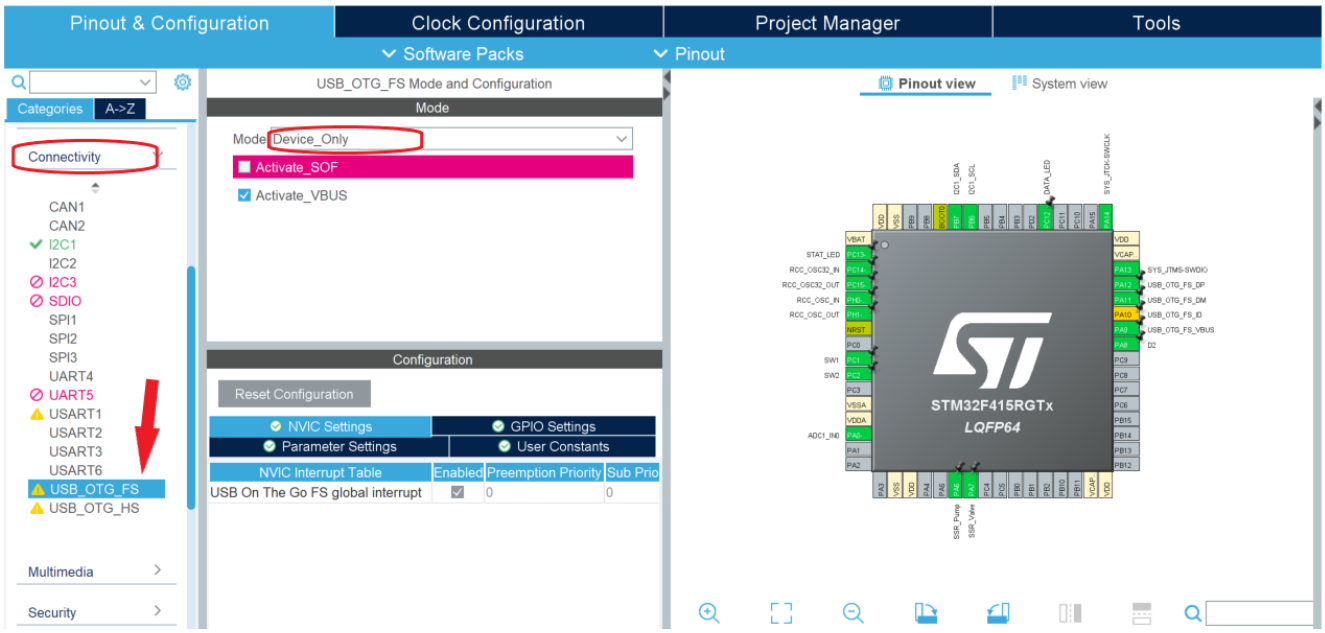
6.4.3 Seriell kommunikasjon via USB

Til testing og dokumentasjon av systemet, er det ønskelig å kunne hente ut data, for å lage plott av måledataene fra mikrokontrolleren, til verifisering. Plattformkortet *STM23F4 ARM MINI*, har en mikro USB-tilkobling, koblet til mikrokontrolleren. Her kan man konfigurere mikrokontrolleren til å sende og motta data via seriell kommunikasjon. Det trengs kun å sende data fra mikrokontrolleren, man vil derfor kun konfigurere dette. Dataen skal sendes fra mikrokontrolleren til en COM-port på PC-en, for så å bli behandlet i programmeringsspråket *Python*.

USB-modulen til mikrokontrolleren, er konfigurert ved hjelp av HAL. Konfigurasjon av pinner og funksjonalitet for USB-modulen, er vist i figur 93, hvor den virtuelle COM-porten konfigureres. I figur 94, settes modulen til "Device Only", ettersom det i vårt tilfelle er PC-en som er "host".



Figur 93: Konfigurering av USB-modulen til mikrokontrolleren.



Figur 94: Konfigurering av USB-modulen til mikrokontrolleren.

Som vist i filhierarkiet i kapittel 6.1.1, finner man funksjonene tilknyttet USB-kommunikasjon under mappen `USB_DEVICE` → `App`, og i filen `usbd_cdc_if.c`. Fra denne filen brukes den ferdigskrevne funksjonen `CDC_Transmit_FS` ("Communications Device Class"), vist i kode 14, til å sende data fra mikrokontrolleren til PC-en.

```
1  uint8_t CDC_Transmit_FS(uint8_t* Buf, uint16_t Len)
2  {
3    uint8_t result = USBD_OK;
4    /* USER CODE BEGIN 7 */
5    USBD_CDC_HandleTypeDef *hcdc = (USBD_CDC_HandleTypeDef*)hUsbDeviceFS.pClassData;
6    if (hcdc->TxState != 0){
7      return USBD_BUSY;
8    }
9    USBD_CDC_SetTxBuffer(&hUsbDeviceFS, Buf, Len);
10   result = USBD_CDC_TransmitPacket(&hUsbDeviceFS);
11   /* USER CODE END 7 */
12   return result;
13 }
```

Kode 14: Funksjon for sending av data fra mikrokontroller til PC.

Funksjonen tar inn en databuffer på åtte bit (`uint8_t Buf`), og en integer som forteller hvor lang databufferen er (`uint16_t Len`). I `variables.h`-filen, er databufferen deklarert med datatype `uint8_t`, og med en lengde på ni bytes, vist i kode 15.

```
1 uint8_t dataBuf[9];
```

Kode 15: Databufferens deklarering i `variables.h`-filen.

I `while`-løkka i `main.c`-filen, er det laget en samlet kode for lesing av både trykksensor og Hall effect-sensor, samt utregning av tidspunktet målingene ble tatt på. Koden for utregning av tid er kopiert fra Jørgen Hemnes Johannessen fra sensorgruppa (fra linje 11 til linje 25). Denne koden er vist i kodeutsnitt 16, hvor trykksensoren og Hall effect-sensoren blir lest i linje 5 og 8. Deretter samles all denne dataen, åtte bit av gangen, i databufferen `dataBuf`.

```
1  /* ----------------------------------------------------------------
2   * AVLESNING AV TRYKKSENSOR OG MAGNETSENSOR MED SENDING AV DATA OVER USB
3   ----------------------------------------------------------------------*/
4 // Les trykket og temperaturen, variablene blir lagt i structen.
5   TRYKKSENSOR_LesTrykk( &trykksensor );
6
7 // Les magnetsensoren
8   magnetsens = MAGNETSENSOR_LesVerdi(&hadc1);
9
10  // Setter opp variabel for millisekund
11  if(tid_start == 0){
12      logge_tid_ms = 0;
13      tid_start = HAL_GetTick();
14  }
15  else{
16      oppdatert_tid = HAL_GetTick();
17      logge_tid_ms = oppdatert_tid - tid_start;
18  }
19
20  // Setter opp variabel for sekund
21  if(logge_tid_ms >= 1000){
22      logge_tid_ms -= 1000;
23      logge_tid_sek += 1;
24      tid_start = HAL_GetTick();
25  }
```

Kode 16: Lesing og samling av data for sending.

Videre, fra linje 11 til 25, blir tiden målingene blir tatt logget. Dette gjøres ved bruk av HAL-funksjonen `HAL_GetTick()`, som returnerer en verdi i millisekunder, som tilsier hvor lenge koden har kjørt. Denne verdien brukes til å definere når første måling ble tatt i første `if`-setning (linje 11). Variabelen `tid_start`, settes dermed til tiden første måling ble tatt, og variabelen `logge_tid_ms`, blir satt til 0. Dette er tiden, i millisekunder, som skal sendes over til PC. Dersom det ikke er første måling som tas, inkrementeres tiden, i millisekunder, ved å regne ut differansen mellom første måling og siste måling i linje 17.

Variabelen `logge_tid_ms`, er en 16-bits variabel. Dersom man sender alle målingene i millisekunder til PC-en, kan tiden det tar før variabelen går i metning, regnes ut som vist i utregning 59.

$$2^{16} \Rightarrow \frac{65535\ ms}{1000} \Rightarrow 65.5\ sek \tag{59}$$

Det er ønskelig å kunne logge data i over 65.5 sekunder, derfor telles det kun til 1000 millisekunder, hvor variabelen `logge_tid_sek`, i andre `if`-setning, inkrementeres med ett sekund for hver gang `logge_tid_ms` har nådd 1000. Ettersom variabelen `logge_tid_sek`, er en 16-bits variabel og tilsvarer sekunder, kan den nye loggetiden regnes ut som vist i 60.

$$2^{16} \Rightarrow \frac{65535\ sek}{60} \Rightarrow \frac{1092.3\ min}{60} \Rightarrow 18.2\ timer \tag{60}$$

Fra linje 26 til 41 i kodeutsnitt 17 legges dataen, som man ønsker å sende over til PC, i databufferen. Som man så tidligere i kode 14, kan databufferen kun sende åtte bit av gangen. Det vil si at de 32-bits og 16-bits variabelene man har, må deles opp til henholdsvis tre og to bytes. 32-bits variabelen deles kun opp i tre bytes, fordi den kun bruker 24 av de 32 bitene.

```
26  // Legger til målt trykk i databufferen for sending via USB.
27  dataBuf[0] = (trykksensor.trykk_comp>>16) &255;
28  dataBuf[1] = (trykksensor.trykk_comp>>8) &255;
29  dataBuf[2] = trykksensor.trykk_comp &255;
30
31  // Legger tid i ms inn i databufferen via USB
32  dataBuf[3] = (logge_tid_ms>>8) &255;
33  dataBuf[4] = (logge_tid_ms>>0) &255;
34
35  // Legger tid i sekunder inn i databufferen via USB
36  dataBuf[5] = (logge_tid_sek>>8) &255;
37  dataBuf[6] = (logge_tid_sek>>0) &255;
38
39  // Legger til data målt av Hall effect-sensor i databufferen for sending via USB.
40  dataBuf[7] = (magnetsens >>8) &255;
41  dataBuf[8] = (magnetsens >>0) &255;
42
43  // Sender hele datapakken til PC.
44  CDC_Transmit_FS((uint8_t *) dataBuf, 9);
```

Kode 17: Lesing og samling av data for sending

For å illustrere hvordan dette gjøres, ser man på 32-bits variabelen `trykk_comp` i linje 27-29. Denne bruker kun 24 bit. De første åtte bitene av `trykk_comp`, legges på første plass i databufferen, ved å høyreskifte disse 16 plasser. Deretter legges de neste åtte bitene av `trykk_comp` på andre plass i databufferen, ved å høyreskifte disse åtte plasser. Til slutt legges de resterende åtte bitene på tredje plass i databufferen, ved å legge disse direkte inn. På samme måte høyreskiftes resten av dataen som skal sendes til PC. Med kommandoen i linje 44, sendes hele databufferen til COM-port på PC. Videre skal man se på hvordan dataen blir mottatt og håndtert i *Python* på PC-en.

Til lesing av seriell data fra COM-porten, brukes *Python*-biblioteket *serial*. Koden som leser av COM-porten er hentet fra video [42], modifisert på av Jørgen Hemnes Johannessen, og videre modifisert på av oss til vårt formål. Koden er vist i kode 18 og 19, og man spesifiserer i starten, i variabel `antallMaaling`, hvor mange målinger man ønsker å logge før programmet avlsuttes.

```python
import serial
import serial.tools.list_ports

# Brukervariabler som settes selv:
antallMaaling = 4000

ports = serial.tools.list_ports.comports()
portList = []

for onePort in ports:
    portList.append(str(onePort))
    print(str(onePort))

val = input("select Port: COM")

for x in range(0,len(portList)):
    if portList[x].startswith("COM" + str(val)):
        portVar = "COM" + str(val)
        print(portList[x])

emergency = []

trykk = []
tid_sek = []
tid_ms = []
magnet = []
tid = []

ser = serial.Serial(portVar,115200, timeout=0)
```

Kode 18: Lesing av serieporten i *Python*.

```
while len(trykk)<antallMaaling:
    if ser.in_waiting:
        datapakke = ser.readline().hex()
        print("datapakke før if = ", datapakke)

        if len(datapakke) < 14:
            databuf = datapakke
            emergency.append(databuf)
            datapatched = "".join(emergency)


            if len(datapatched) == 14:
                print('reparert data: ' + datapatched)
                data = datapatched
                emergency.clear()

        else:
            data = datapakke
            trykk.append(int(data[0:6], 16))
            tid.append(int(data[10:14], 16) + (int(data[6:10], 16)/1000 ))
            magnet.append(int(data[14:18], 16))


# KODE FOR Å LOGGE BÅDE TRYKK OG MAGNET
file = open("Matlab_Data.txt", "w+", encoding='utf-8')
string =  "trykk = "+ str(trykk) + "/100;\n"+ "tid = "+ str(tid)+ "; \n"
                + "magnet = "+ str(magnet)+ ";"
file.write(string)
file.close()
```

Kode 19: Lesing av serieporten i *Python* fortsettelse.

I serieporten, leses hver byte som to heksadesimale tall. Det vil si at en 16-bits variabel tilsvarer fire bytes. For å dekode dataen som blir mottatt, er det viktig å ha kontroll på rekkefølgen dataen er lagt inn i databufferen. Dette er vist nedenfor, i tabell 11, hvor dataen som blir mottatt, blir sammensatt i en variabel, `data`.

Variabel	Mikrokontroller ⇒ PC
Trykksensor	`dataBuf[0:2]` ⇒ `data[0:6]`
Tid i millisekund	`dataBuf[3:4]` ⇒ `data[6:10]`
Tid i sekund	`dataBuf[5:6]` ⇒ `data[10:14]`
Hall effect-sensor	`dataBuf[7:8]` ⇒ `data[14:18]`

Tabell 11: Oversikt over data sendt fra mikrokontroller og mottatt på PC.

Deretter sorteres disse ut med hensyn til tabellen over, og legges i hver sin liste etterhvert som dataen leses inn, vist i kode 19, linje 47-50. Når programmet har logget ønsket antall målinger, avsluttes `while`-løkken, og vektorene som er lagret blir skrevet til en tekstfil, `Matlab_Data.txt`, på *MATLAB*-format. Det vil si at, for å enkelt kunne kopiere alt fra tekstfilen, og lime det direkte inn i *MATLAB* for å lage plott, skrives vektorene med variabelnavn foran, sammen med resterende syntaks som *MATLAB* krever. Dette kan ses i kode 19, linje 54 til 58. Deretter kan dataen skrevet til `Matlab_Data.txt`, kopieres og limes inn øverst i koden i *MATLAB*, vist i kode 20.

```
1  subplot(2,1,1)
2  plot(tid, trykk);
3  title('Målt trykk [mbar]');
4  ylabel('Trykk [mbar]');
5  xlabel('Tid [s]');
6
7  subplot(2,1,2)
8  plot(tid, magnet);
9  title('Målt magnetfelt [mV]');
10 ylabel('Magnetfelt [mV]');
11 xlabel('Tid [s]');
```

Kode 20: Kode i *MATLAB* for plotting av sensordata mot tid.

Denne metoden blir brukt til å gjennomføre testrapporter som krever plotting av data for videre analyse. Et alternativ til å plotte i *MATLAB*, ville vært å lage en automatisk plotte-funksjon i *Python*-koden, ettersom *Python* har egne bibliotek med kraftige plottefunksjoner.

7 Resultat

Innhold

Dette kapittelet tar for seg de ulike testene som er gjort av systemet, og hvordan det fungerer i praksis. Selve testrapportene ligger i vedlegg B, og resultatene blir her tatt opp og diskutert.

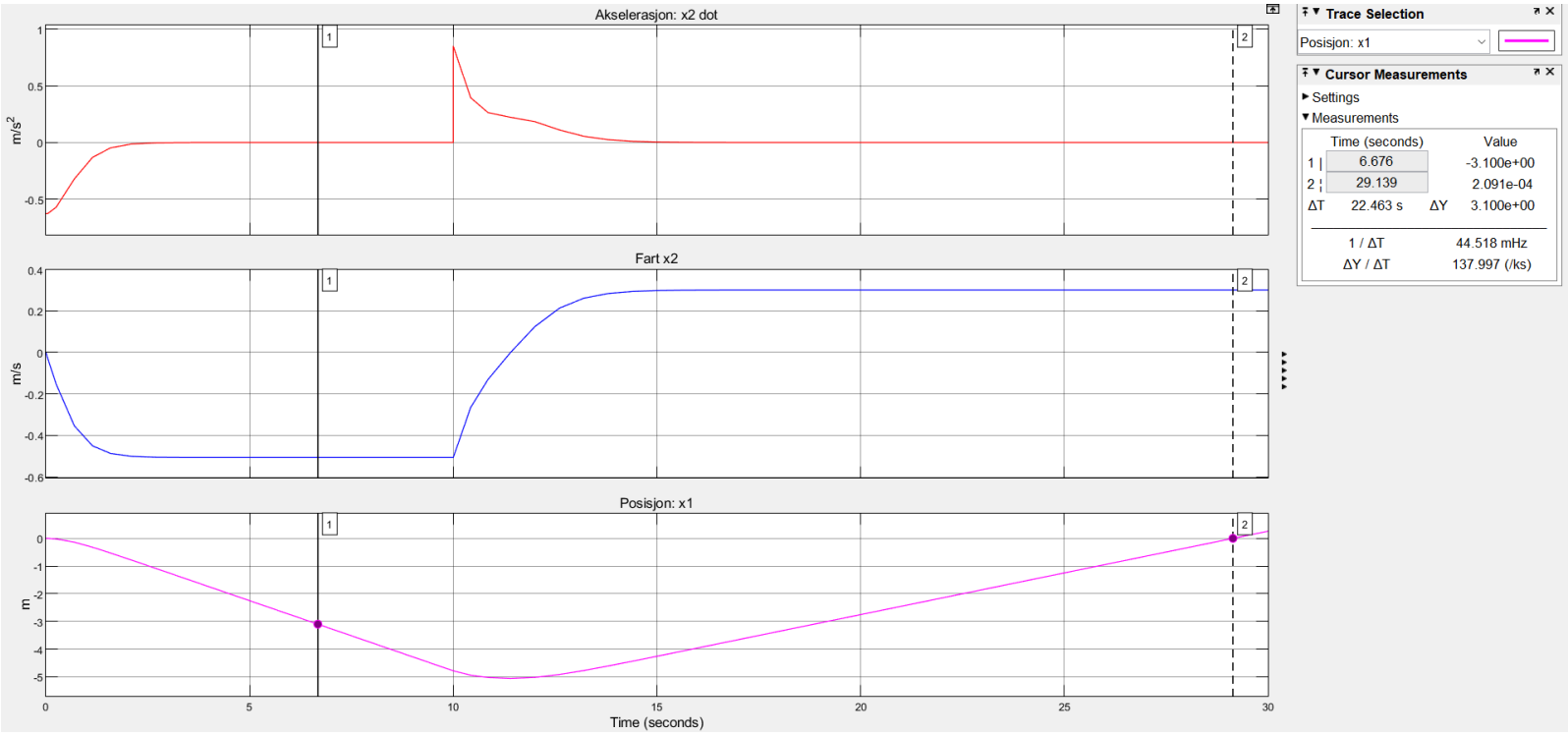
7.1 Vanntesting

Ved vanntesting av flyteren fungerte systemet som det skulle. Det startet opp etter man berørte med magneten, og utførte deretter to profiler. En video av testen er vist i [36]. i tabell 12, er tidene flyteren brukte på de ulike delene av profilene, listet opp. Dette er tilnærmede tider, i og med at det er vanskelig å ta tiden helt nøyaktig ut ifra videoen.

Hendelse	**Målt tid**
Venter etter at magnet er blitt fjernet	12 sekunder
Synker ned første gang	12 sekunder
Står på bunnen og måler trykk	11 sekunder
Flyter opp første gang	12 sekunder
Venter ved overflaten og måler trykk	7 sekunder
Synker ned andre gang	12 sekunder
Står på bunnen og måler trykk	10 sekunder
Flyter opp siste gang	11 sekunder

Tabell 12: Målte tider ved hver hendelse.

Totalt bruker flyteren altså 87 sekunder, eller 1 minutt og 27 sekunder. I kapittel 3.2, ble det utledet og forklart en dynamisk modell for systemet. Grafen fra modellen, som viser akslerasjonen, farten og posisjonen til flyteren, er gjengitt i figur 95.



Figur 95: Grafen fra den dynamiske modellen. Denne er nærmere forklart i kapittel 3.2.

Bassenget som ble brukt under vanntesting er 3.1 meter dypt. Derfor ser man på tiden modellen bruker ned til 3.1 meter, samt tiden den bruker fra 3.1 meter og opp til overflaten. Dette ser man blir 7 sekunder på å synke, og 10 sekunder på å stige opp. For å legge til tiden den bruker i overflaten og på bunnen, må man se på hvordan koden er lagt opp for dette.

I kapittel 6.2.1, i delen om konkurransemodus, er sekvensene flyteren skal gjennom, listet opp. Her kan man se at etter et magnetfelt er detektert, venter flyteren først på at det skal bli borte. For å sjekke om magnetfeltet er borte (og for å sjekke om det er der), brukes en variabel, `timsek`, som blir satt til 1 hvert 100 ms = 0.1 s. Deretter sjekkes det om magnetfeltet har vært unnværende 50 målinger på rad. Da kan man regne ut tiden det tar, fra magneten forsvinner, til Hall effect-sensoren har registrert og godtatt det, som i ligning 61.

$$0.1 \cdot 50 = 5\ s \tag{61}$$

Det neste steget flyteren skal gjennom, er å sjekke at den ligger i overflaten før den begynner på profilene. Til å gjøre dette, brukes trykksensoren. Denne bruker den samme variabelen (`timsek`) som Hall effect-

sensoren, men godtar ikke trykket som uendret før den har 100 like målinger. Dermed blir tiden den bruker på å registrere at trykket ikke endrer seg, som i ligning 62.

$$0.1 \cdot 100 = 10\ s \tag{62}$$

Altså vil det ta $5 + 10 = 15\ s$ fra magneten blir fjernet til flyteren begynner å synke. Deretter vil den samme koden, for å registrere at trykket ikke lenger endrer seg, bli brukt for å detektere at flyteren er på bunnen, i overflaten og så på bunnen igjen. Når disse tidene legges sammen med tidene for å synke og flyte fra den dynamiske modellen, blir den totale tiden på to profiler som vist i ligning 63.

$$15 + 7 + 10 + 10 + 10 + 7 + 10 + 10 = 79\ s = 1\ min + 19\ s \tag{63}$$

Sammenlignet med tidene fra videoen, bruker altså flyteren 8 sekunder lenger i virkeligheten enn i den dynamiske modellen. For enklere sammenligning, er alle tidene gjengitt i tabell 13.

Posisjon/bevegelse	**Topp**	**Ned**	**Bunn**	**Opp**	**Topp**	**Ned**	**Bunn**	**Opp**	**Sum**
Dynamisk modell	15	7	10	10	10	7	10	10	79 s
Virkelighet	12	12	11	12	7	12	10	11	87 s

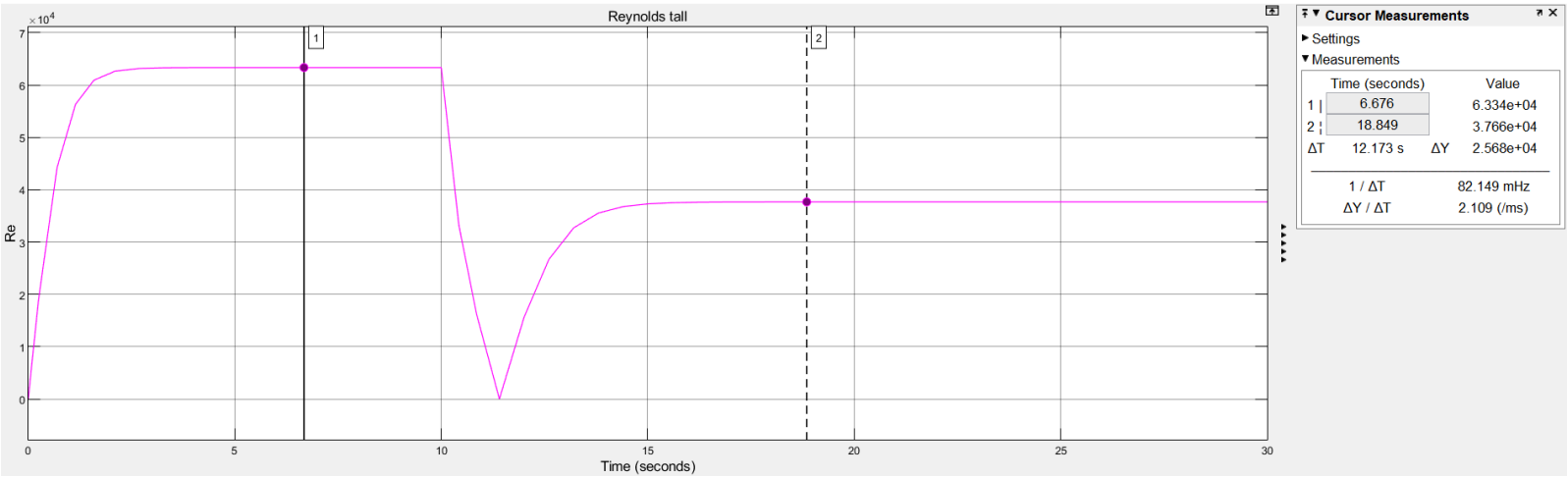
Tabell 13: Tidene i den dynamiske modellen og ved vanntest.

I den dynamiske modellen, er det tatt hensyn til dragkoeffisienten, som påvirker fremgangen til flyteren med en dragkraft. Det er derimot ikke tatt hensyn til Reynoldstallet. Reynoldstallet gir en beskrivelse av hvordan en strøm av væske eller gass beveger seg, og er gitt som i formel 64 [25], hvor variablene i formelen er listet opp nedenfor.

- r = 0.0625 m - radiusen til flyteren
- v - farten til flyteren
- $\nu = 10^{-6}\ m^2/s$ - den kinematiske viskositeten til vannet

$$Re = \frac{rv}{\nu} \tag{64}$$

I [10] står det at dersom Reynoldstallet er større enn en, vil ikke lenger Stokes lov gjelde. Det er Stokes lov som er brukt, for å finne dragkraften, i den dynamiske modellen. Derfor er formel 64 lagt inn i *Simulink*, for å finne Reynoldstallet. Resultatet er vist i figur 96.



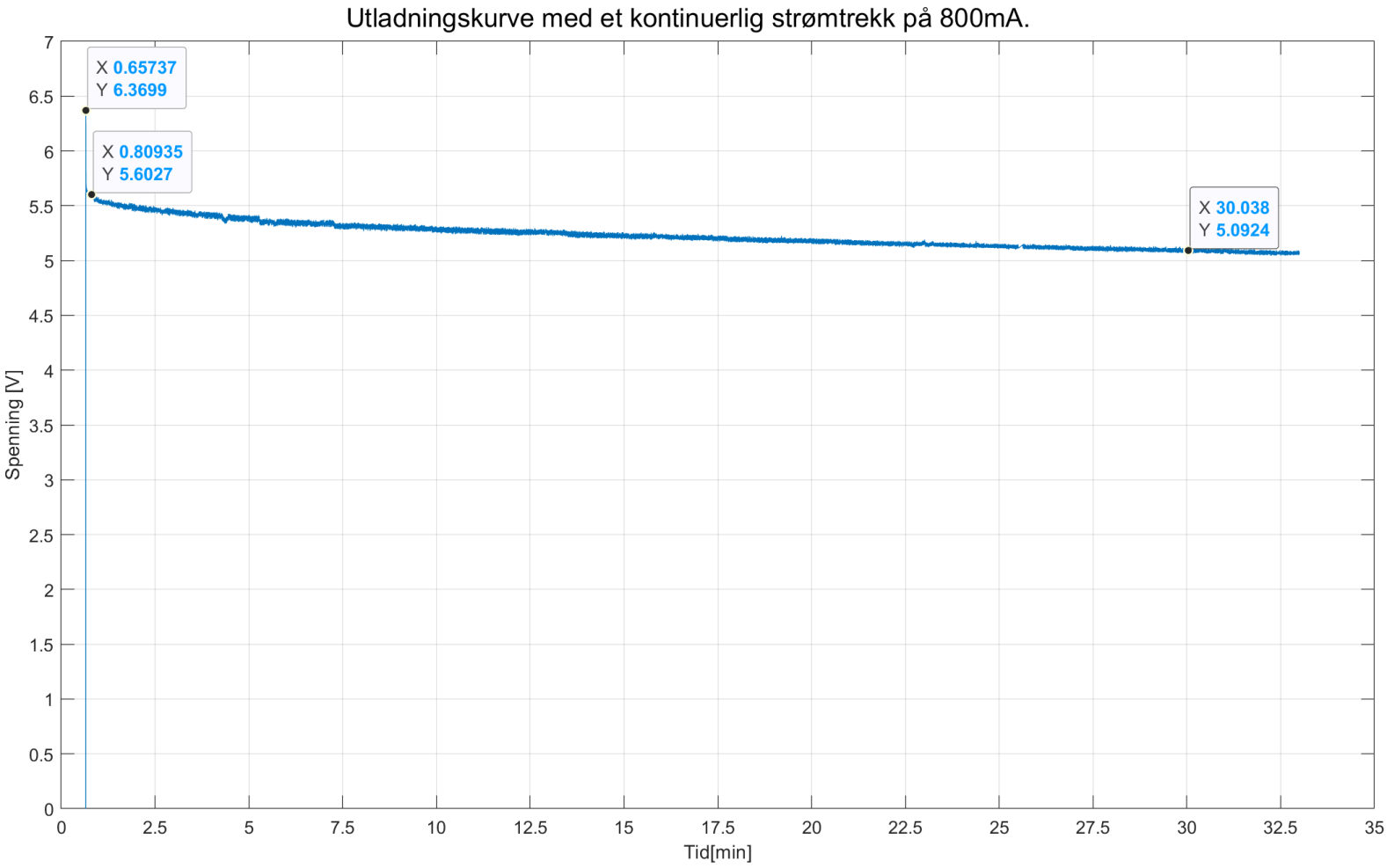
Figur 96: Reynoldstallet over tid i den dynamiske modellen.

Her kan man se at Reynoldstallet stiger veldig fort etter start og legger seg på $6.3 \cdot 10^4$. Etter ballongen har fylt seg opp, stabiliserer Reynoldstallet seg fort igjen, denne gangen på $3.8 \cdot 10^4$. I [10] står det at dersom $Re < 1$, betyr det at strømningene rundt flyteren er laminære. Laminær strømning vil si en strømning uten turbulens. Når $Re < 10^3$ derimot, vil det oppstå turbulente strømninger.

Dette kan være med på å forklare hvorfor flyteren bruker lengre tid i virkeligheten enn i den dynamiske modellen. Når det oppstår turbulente strømninger, vil dragkraften øke og farten blir dermed mindre. Det kan også, til en viss grad, ses i videoen. Spesielt når flyteren synker ned har den en del svingninger, som vil være med på å sakke ned farten.

7.2 Utladningstest

Det ble utført en utladningstest på en seriekobling av batteriene. Som nevnt i kapittel 4.4, består batteripakken av en parallellkobling av to seriekoblinger, der hver seriekobling har fire D-type alkaliske batterier. I testrapporten, ønsket man å teste hvor lenge batteriene kan levere med et konstant strømtrekk tilsvarende normalt strømtrekk i systemet. Testrapporten er vist i vedlegg B.1. Utladningstesten ble utført på laboratoriet, der pumpa ble kjørt konstant i en halvtime. Spenningen på batteriet ble logget i løpet av halvtimen, og grafen for utladningen er vist i figur 97.



Figur 97: Utladningskurve av batterispenningen iløpet av 30 minutter.

I figur 97, kan man observere at idet batteriet blir belastet, dropper batterispenningen med 0.77 V, som nok er et resultat av batteriets indre motstand. Videre observeres det at, med et konstant strømtrekk på 800 mA, holder batterispenningen seg relativt jevn gjennom alle de 30 minuttene, og dropper totalt 1.28 V.

For å slippe å bytte batteri for ofte, ble det satt et minimumskrav til et akseptabelt spenningsfall i løpet av 30 minutter. Dette minimumskravet er på $6\ V - 3.6\ V = 2.4\ V$, dersom man går ut ifra at minimumsspenningen på batteriet er 3.6 V.

Ettersom batteriet ble testet med et reelt strømtrekk over en lengre periode enn det som forventes i praksis, ble det konkludert med at batteriet holder seg godt innenfor kvalitetskravene som er ønskelige. I konkurransen, skal hver oppgave utføres på maksimalt 15 minutter. Her ble batteriene testet i 30

minutter, uten at spenningen falt for mye. I tillegg er batterikapasiteten dobbelt så høy i praksis i vårt system, ettersom vi har parallellkoblet to seriekoblinger, med fire D-type alkaliske batterier. Den reelle batterilevetiden skal nå ses nærmere på.

7.2.1 Reell batterilevetid ved vanntest

Under vanntesting av flyteren, startet man med nye batterier, for å kunne få et mål på reell batterilevetid. Vanntesten varte i omtrent en time, hvor man fikk kjørt 20 sekvenser i løpet av den timen. 20 sekvenser tilsvarer at flyteren utførte 40 profiler i vann. Spenningen på batteriet ble målt både før og etter test, hvor resultatet av dette er gitt i tabell 14.

	Spenning før vanntest [V]	**Spenning etter vanntest [V]**
Før pumpa er startet	6.30	5.73
Etter pumpa er startet	5.38	5.08

Tabell 14: Spenningen på batteriet før og etter vanntest.

Ut ifra tabellen, ser man at med batteriet tilkoblet, men uten at pumpa eller ventilen er aktiverte, har batterispenningen droppet med $6.30 - 5.73 = 0.57\ V$ etter å ha kjørt 20 sekvenser. Differansen på før og etter vanntest, mens pumpa kjørte, er $5.38 - 5.08 = 0.30\ V$. Dette gir oss en svært tilfredsstillende batterilevetid, og det kan konkluderes med at systemet er overdimensjonert, til fordel for god batterilevetid.

7.3 Kretskortet

På selve kretskortet, ble det utført tre ulike tester. Som nevnt i kapittel 5.8, ble kretskortet loddet for hånd, men før noe ble loddet på, ble det utført en kontinuitetstest. Testrapporten ligger i vedlegg B.2. I denne testen ble det funnet to feil på kretskortet. To av komponentene var ikke koblet til jord. Dette ble rettet opp i ved å lodde over fra loddelandet til en nærliggende via, eller fra pinnen til en annen komponent med jord. De to andre testene, testet kraftforsyningen og solid state-releene. Dette skal gjennomgås i de to neste delkapitlene.

7.3.1 Kraftforsyning

På kretskortet er det en egen del som styrer kraftforsyningen og regulerer den til 3.3 V og 12 V. Selve kretsen gjennomgås i kapittel 5.1, og testrapporten ligger i vedlegg B.3. Denne testen skulle verifisere at utgangsspenningen fra de to spenningsregulatorene, U3 og U4, var riktig, samt finne nedre spenningsgrense inn på regulatorene før utgangsspenningen ble påvirket. Konklusjonen fra testen blir gjentatt i neste avsnitt.

Forventet utgangsspenning fra U4 er 12.0 V, og fra U3 er det 3.3 V. U4 har en differanse på 0.12 V fra forventet verdi, og U3 har en differanse på -0.02 V fra forventet verdi. Dette er så små differanser at de

neglisjeres. Nedre grense på inngangsspenning til U4, ble funnet til å være 2.4 V. Dette stemmer godt overens med databladet [38]. Nedre grense for U3, ble funnet å være 3.3 V. Dette stemmer også godt med databladet [15]. Dette ser også riktig ut, ettersom U3 er en lineærregulator, og dermed ikke kan regulere en spenning ned til 3.3 V dersom inngangsspenningen er under 3.3 V. Denne testen ble utført uten en last på utgangen til U3, som tilsier at dette er tomgangsspenning. Dersom det hadde vært tilkoblet en last på utgangen, ville spenningen på utgangen hatt et høyere spenningsfall, på opp til 500 mV [15].

Det ble altså konkludert med at begge spenningsregulatorene fungerer slik de skal. U4 som skulle gi ut 12 V, ga ut 12.04 V. Denne spenningen forsyner pumpa og ventilen. I databladet til ventilen [7], står det på side 15, at inngangsspenningen kan variere med ±10%, altså ±1.2 V. Det vil si at en inngangsspenning på 12.04 V vil gå helt fint.

7.3.2 SSR

Solid state-releene som brukes for å styre pumpa og ventilen, blir beskrevet i kapittel 5.2.1. Der ble det funnet at de skal ha 1.65 V (typisk verdi) for å aktiveres. Denne spenningen ble derfor regulert med en motstand, og måtte derfor testes, sammen med selve releene (om de gir ut riktig spenning og til riktig tidspunkt). Denne testen er dokumentert i vedlegg B.4. Der ble det funnet at både spenningen inn på solid state-releene og spenningen ut, var som forventet. Altså 1.65 V inn på releene (når de var aktiverte), og 0 V eller 12 V ut, avhengig av om pumpa/ventilen skulle være av eller på.

7.4 Trykksensor

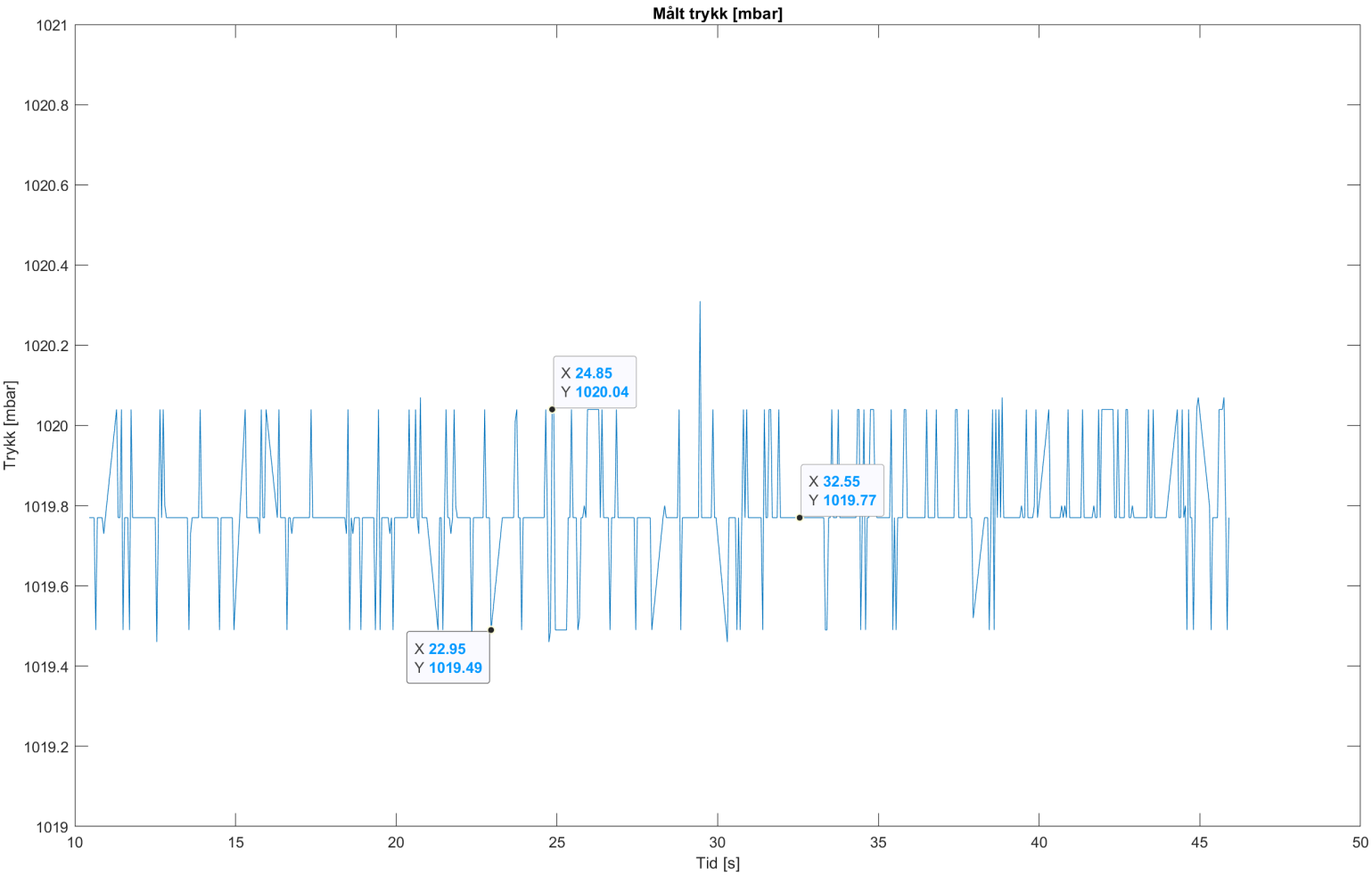
For trykksensoren, ble det utført to forskjellige tester. De er vist i vedlegg B.5 og B.7. Den første skulle teste trykksensorkretsen og I^2C-grensesnittet, samt selve trykksensoren. Dette innebærer kommunikasjonen mellom mikrokontrolleren og trykksensoren, i tillegg til sjekk av statisk målefeil på trykksensoren. I den andre testen, ble det brukt et trykkammer for å teste hele måleområdet til trykksensoren, både når trykket økte og når det minket. Dette var for å kunne redegjøre for tilfeldig målefeil over hele måleområdet. Dette ble utført med hjelp av Magnus Wersland og Kim Andre Nesse Vorland på kjemilaboratoriet.

7.4.1 Test av trykksensor og trykksensorkrets

Fra resultatene fra den første testen i vedlegg B.5, ble det funnet at råverdien fra sensoren var fornuftig, og samsvarte med hva atmosfæretrykket var denne dagen. Råverdien ble målt til å være 98343 mens trykksensoren lå i åpen atmosfære. Dette tilsvarer 0.983 bar. I følge [40] var gjennomsnittstrykket i atmosfæren for april, som er måneden denne testen ble utført, på 1.013 bar. Det vil si at trykksensoren har et avvik på omtrent 0.03 bar (30 mbar), avhengig av hva det faktiske atmosfæretrykket var denne dagen. Ut ifra dette, kan det konkluderes med at kommunikasjonsprotokollen og trykksensoren virker.

Videre er det ønskelig å kartlegge, mer nøyaktig, hva den faktiske målefeilen til trykksensoren er, både statisk, og gjennom hele måleområdet, for å kunne finne en eventuell hysterese. Trykksensoren ble dyppet under vann i et beger på 10 cm dybde, og 500 målinger ble logget. På denne måten kan man, mer nøyaktig,

se den statiske målefeilen. Resultatet av de 500 målingene er gjengitt i figur 98.



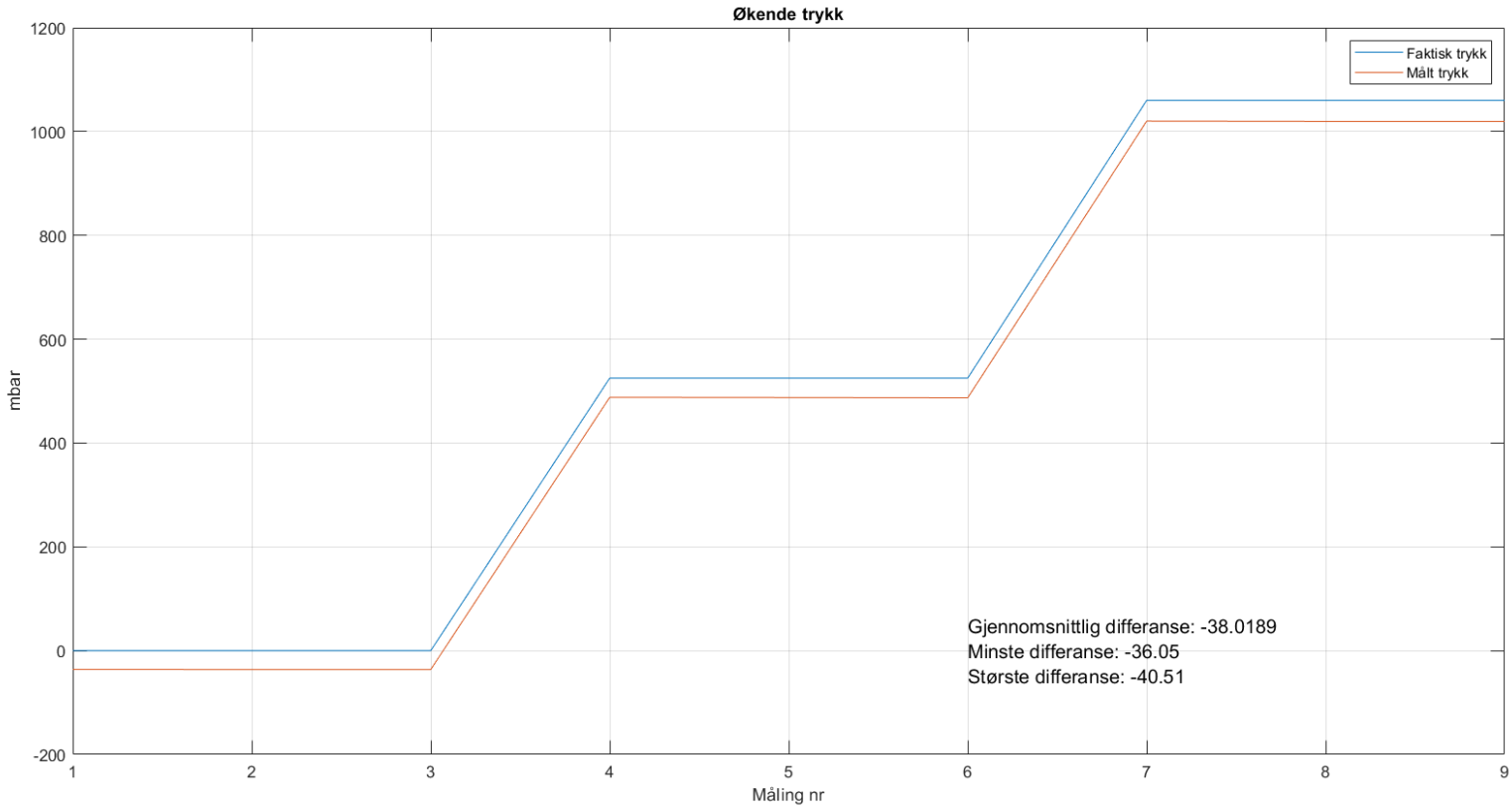
Figur 98: Plott av 500 målinger fra trykksensoren. Hentet fra vedlegg B.5.

Fra figuren, ser man at den gjennomsnittlige differansen, mellom topp- og bunn-verdiene, er $1020.04 - 1019.49 = 0.55\ mbar$. Det er beskrevet i databladet [6] at det kan forventes en nøyaktighet på $\pm 1.5\ mbar$. At målt verdi har en differanse på 0.55 mbar, er dermed godt innenfor og det kan konkluderes med at trykksensorens nøyaktighet er som forventet.

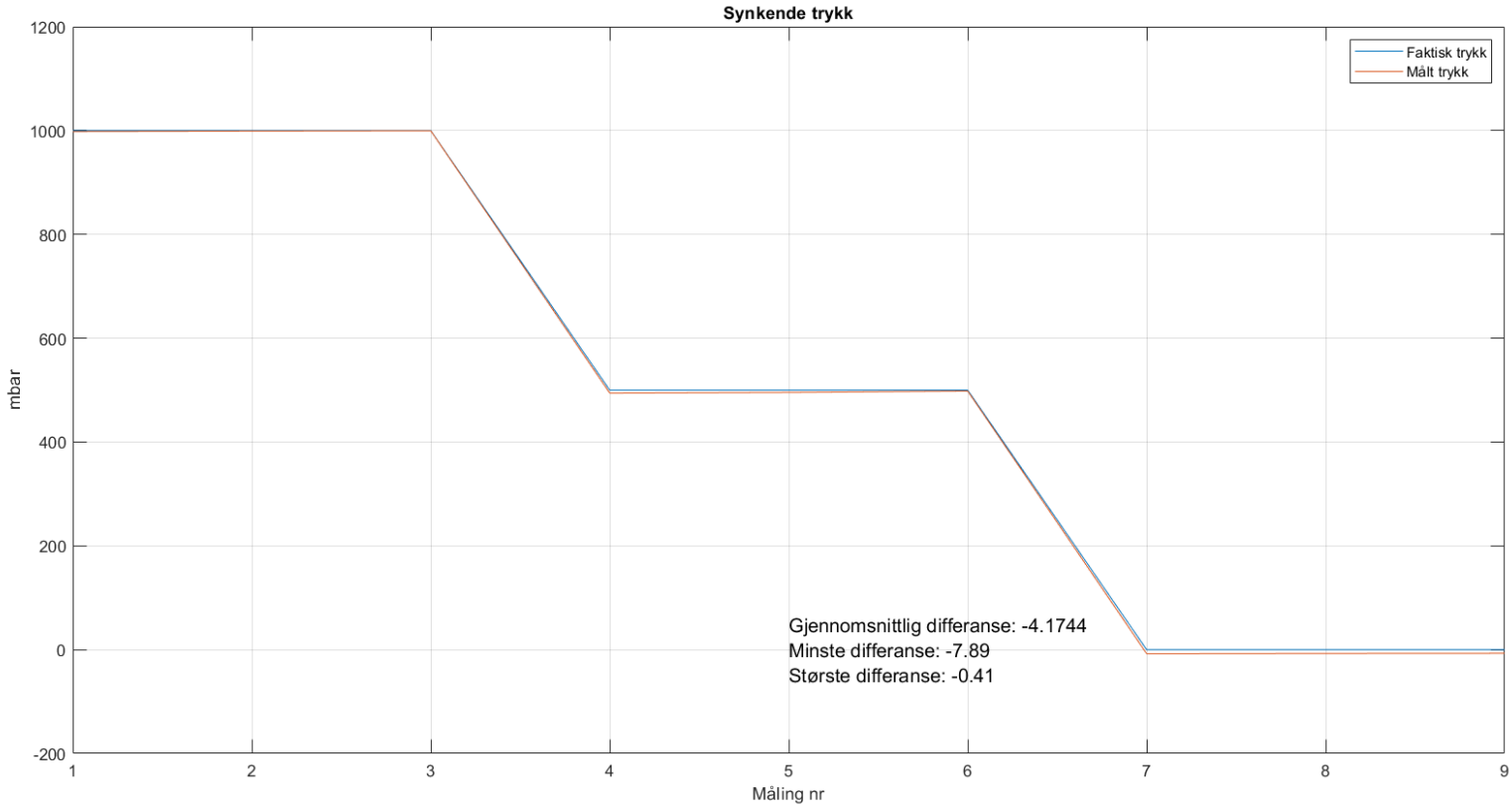
I konklusjonen i testrapporten, er den teoretiske verdien til trykket, 10 cm under vann, regnet ut til å være 1081.5 mbar. Den gjennomsnittlige målte verdien er 1019.8 mbar. Sammenligner man disse, får man en differanse på omtrent 62 mbar. Dette er den statiske målefeilen som blir observert ved et stasjonært trykk, og man skal videre se på hele måleområdet til trykksensoren.

7.4.2 Test av hele måleområdet til trykksensor

Resultatene fra den andre testen av trykksensoren i vedlegg B.7, der hele måleområdet til trykksensoren ble testet, viser blant annet to ulike figurer med plott av målt trykk og faktisk trykk over hele måleområdet, altså fra null til fem bar. Dette tilsvarer fra vannoverflaten, og ned til 50 meter. Ettersom flyteren kun skal gå ned til ti meters dybde, vises de samme grafene her, men kun fra null til en bar, for bedre visualitet. Grafene kan ses i figur 99 og 100.



Figur 99: Graf over målt trykk og faktisk trykk med stigende trykk.



Figur 100: Graf over målt trykk og faktisk trykk med synkende trykk.

Ved å studere det målte trykket sammen med det faktiske trykket, i stigende og synkende rekkefølge, kan det ses at det målte trykket følger det faktiske trykket, men med et avvik. Man observerer også at avviket er høyere ved økende trykk enn ved synkende. Det gjennomsnittlige avviket mellom faktisk og målt trykk, med økende trykk, er ≈ -38.02 mbar, mens med synkende trykk, er det gjennomsnittlige avviket ≈ -4.17 mbar.

Som nevnt i testrapporten i vedlegg B.7, har man regulert det påsatte trykket manuelt med en regulator, som skrus for hånd. Denne regulatoren har sannsynligvis en viss unøyaktighet, samt at laboratorieansatte nevnte at denne ikke var blitt kalibrert på et par år. Trykkmåleren, som viste hvilket trykk som var inne i trykkammeret, vippet ofte mellom to-tre verdier på det tredje desimalet. Usikkerheten ligger dermed i det tredje desimalet, altså opp til 10 mbar. Avviket som er målt, er større enn 10 mbar. Dermed er det en eller flere målefeil i systemet, som man ønsker å se nærmere på.

En målefeil kan være systematisk eller tilfeldig. Systematiske målefeil kan blant annet skyldes feil i trykkmåleren eller av oss, som observatører, som noterte verdier. Det har vært forsøkt, under testing, å ta hensyn til systematiske målefeil. Tilfeldige målefeil kan skyldes hysterese, produksjonsvariasjoner eller linearitetsfeil, og har en unøyaktighet som er oppgitt i databladet. Dette er vist i figur 101.

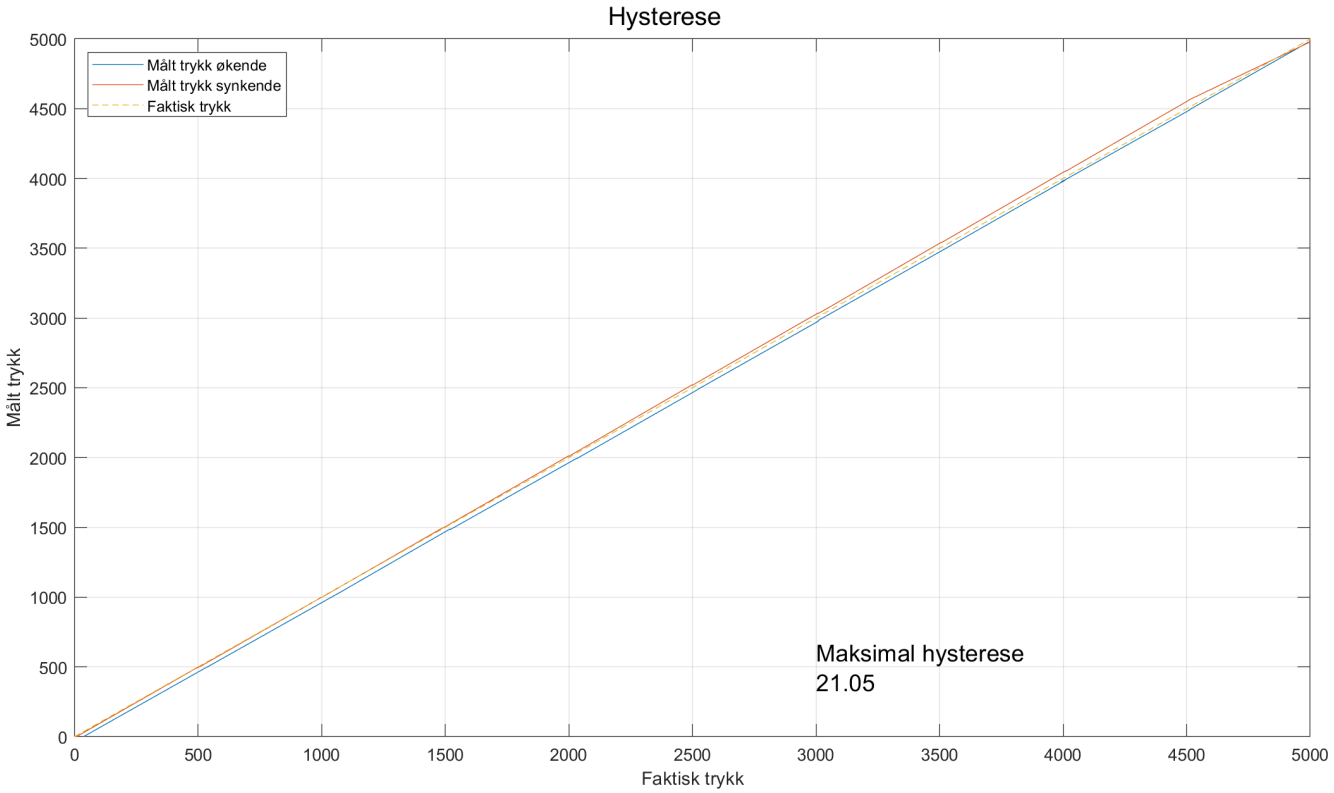
PRESSURE OUTPUT CHARACTERISTICS (V_{DD} = 3 V, T = 25°C UNLESS OTHERWISE NOTED)

Parameter	Conditions	Min.	Typ.	Max	Unit
Operating Pressure Range	P_{range} Full Accuracy	0		5	bar
Absolute Accuracy, autozero at one pressure point 300…1100 mbar (1)	at 20°C, 300..1100 mbar at 0..50°C, 300..1100 mbar at -40..85°C, 300..1100 mbar	-1.5 -4.0 -15.0		+1.5 +4.0 +15.0	mbar
Absolute Accuracy, autozero at one pressure point 0…5000 mbar (1)	at 20°C, 0..5000 mbar at 0..50°C, 0..5000 mbar at -40..85°C, 0..5000 mbar	-80 -100 -120		+80 +100 +120	mbar
Maximum error with supply voltage (3)	V_{DD} = 1.8 V … 3.6 V		+/-5		mbar
Long-term stability (2)			+/-1		mbar/yr
Resolution RMS	OSR 4096 2048 1024 512 256		0.036 0.054 0.081 0.126 0.195		mbar

Figur 101: Tabell hentet fra [6].

Dersom man ser på et trykk fra 0 til 5000 mbar, og en temperatur på 20°C, står det at målingene kan variere med ±80 mbar. Det vil si at alle målingene våre er godt innenfor det databladet sier de skal være. Det står også at mellom 300 og 1100 mbar, og ved 20°, har trykksensoren en nøyaktighet på ±1.5 mbar. I dette tilfellet er ikke målingene våre innenfor de variasjonene som er oppgitt.

Figur 102, viser en graf hvor det målte trykket, i både stigende og synkende rekkefølge, er plottet mot det faktiske trykket, sammen med det faktiske trykket.



Figur 102: Graf over målt trykk i stigende og synkende rekkefølge plottet sammen med faktisk trykk.

Man kan observere at det målte trykket, ved de samme verdiene, ikke er likt med økende og synkende trykk. Dette kan skyldes at membranen i trykksensoren får ytre påvirkninger av det påsatte trykket, som gjør at den endrer seg og ikke går tilbake til sin opprinnelige form, og som her skaper hysterese.

7.4.3 Trykksensoren i vårt system

I testrapporten blir det, i konklusjonen, bevist at dersom trykksensoren skal brukes til å regne ut dybde i vann, må denne kalibreres. For å ta resultatene man har sett på til nå, inn i vårt system, skal trykksensoren kun brukes til å måle endring i trykk. Det vil si at man ikke trenger å kalibrere trykksensoren til vårt formål.

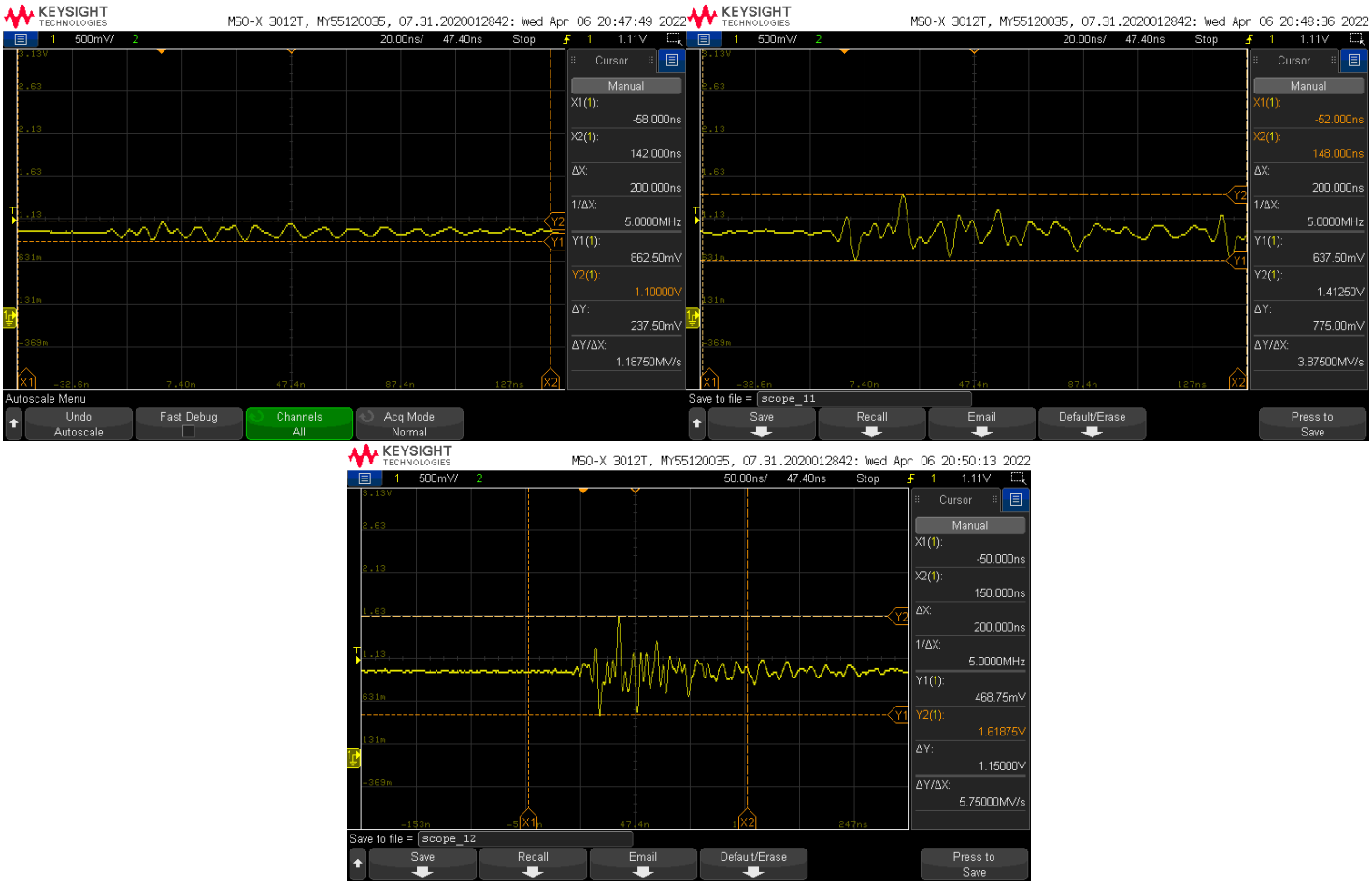
I hovedkoden hvor trykksensoren avleses, er det satt en variabel, `differanse_margin`. Denne er skrevet mer om i kapittel 6.1.2. Hensikten til denne variabelen er å avgjøre hvor stor differanse trykkmålingene kan ha, for at det skal telles som samme trykk. For å finne denne verdien, ser vi på den statiske målefeilen som ble funnet i testrapport B.5, og som er vist i figur 98. Den maksimale differansen ble målt til å være 0.55 mbar, men det var kun på 0.01 meters dybde. Det er ønskelig å vite hva målefeilen er helt ned til 10 meters dybde. Ettersom dette ikke er testet, går man utifra databladet [6] og figur 101. Her ser man

at mellom 300 og 1100 mbar, er nøyaktigheten på ±1.5 mbar ved 20°C. Vi ønsker å forsikre oss om at flyteren snur ved bunn og overflate, men det er ikke sikkert at temperaturen alltid vil ligge på 20°C. Derfor ser man på temperaturer mellom 0 og 50°C. Her er nøyaktigheten på ±4 mbar. Den digitale verdien til 4 mbar tilsvarer 400, som gir den totale differansen på 800.

7.5 Hall effect-sensor

I testen av Hall effect-sensoren i vedlegg B.6, ble selve sensoren testet. Her ble det analoge signalet fra sensoren avlest ved hjelp av et oscilloskop. Det var ønskelig å avdekke hvor mye støy pumpa utgjorde, i form av magnetisme, når den ble kjørt, for å sjekke om det er mulig å lese det magnetiske feltet fra en magnet mens pumpa kjører.

Fra resultatene i vedlegg B.6, fikk man de tre skopbildene som er vist i figur 103. Dette er det filtrerte analoge signalet fra Hall effect-sensoren mens pumpa kjører. Flere detaljer rundt RC-filteret signalet går gjennom her, er beskrevet i kapittel 5.3.1.



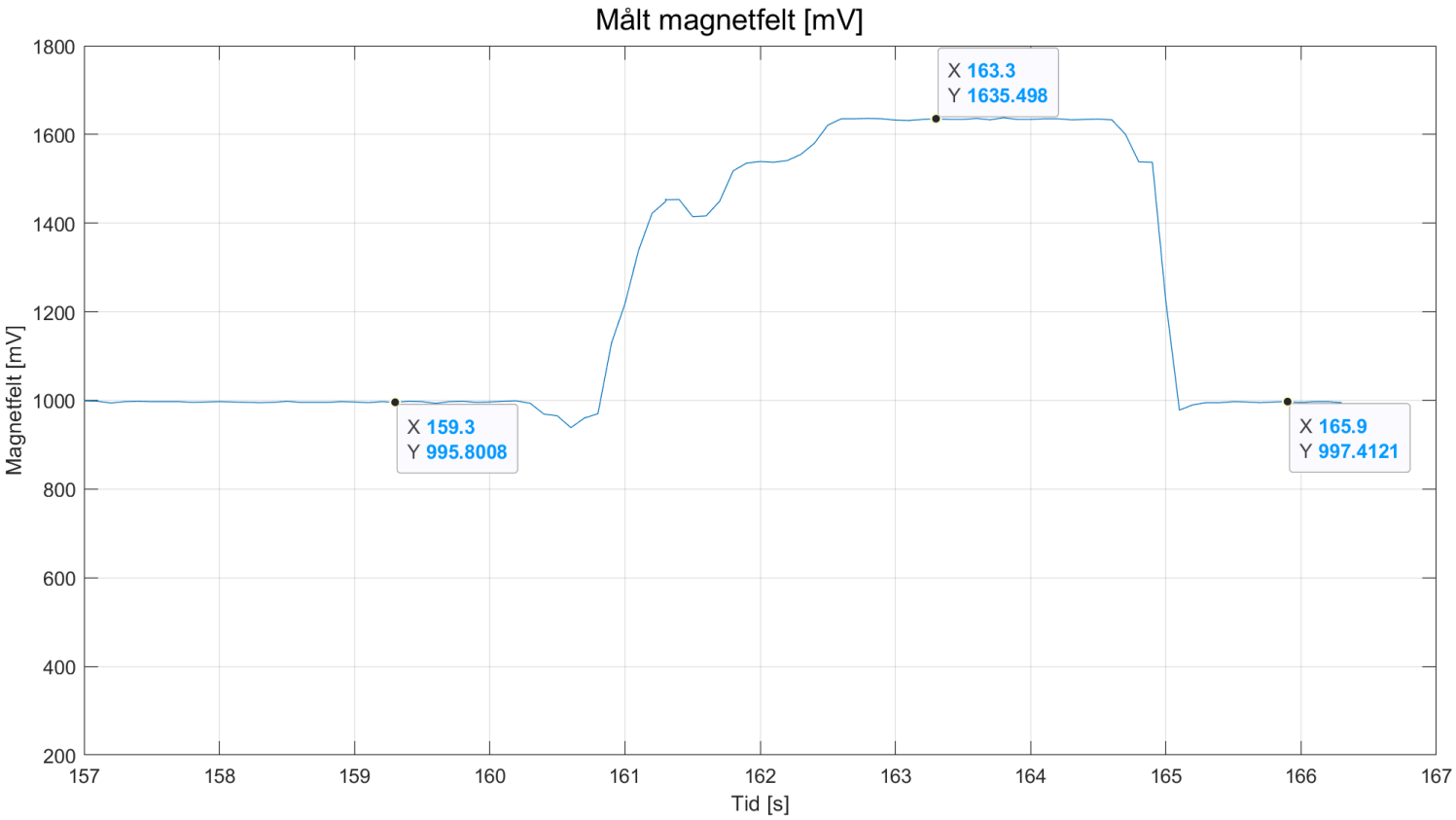
Figur 103: Skopbilde av filtrert signal mens pumpe.

I følge databladet [16], er minimal og maksimal utgangsverdi på Hall effect-sensoren, henholdvis 0.2 V til 1.8 V. Det filtrerte signalet fra skopbildene, i figur 103, er varierende og får i blant spikere, som det nederste bildet viser. Her er maksimalverdien målt til å være 1.6 V. Det støyende signalet fra pumpa når altså ikke hele området til sensoren, fra det som er observert.

7.5.1 Hall effect-sensoren i vårt system

I vårt system, skal ikke Hall effect-sensoren måle magnetfelt samtidig som pumpa kjører, dermed bør ikke pumpestøyen være et problem. Det ble likevel konkludert med at et digitalt filter burde implementeres. Dette gjøres ved å lese av sensoren hvert 100 ms, samt kreve at det skal være 50 like målinger etter hverandre, for at et avlest magnetfelt skal godkjennes.

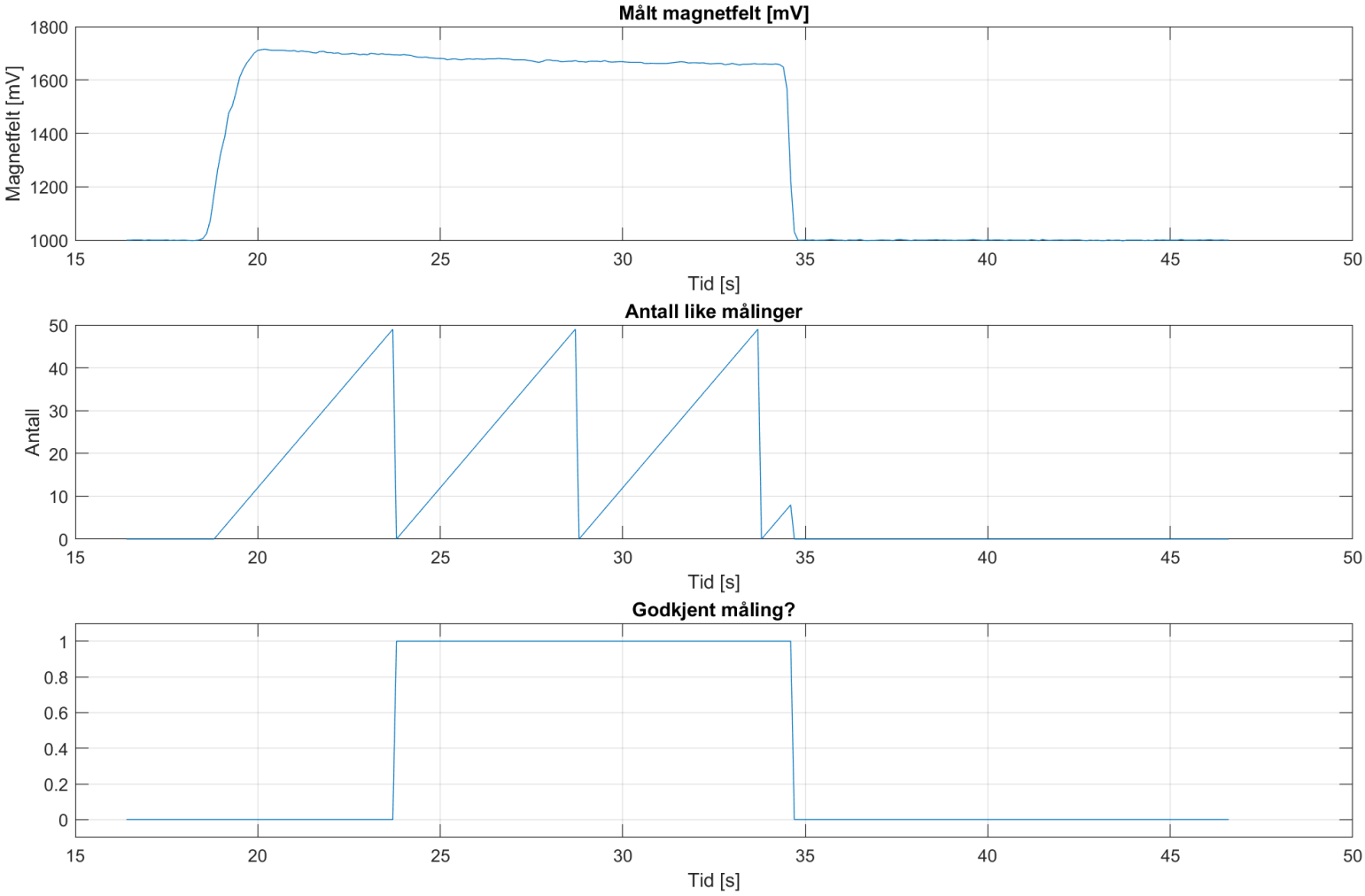
Ved å lese av Hall effect-sensoren hvert 100 ms, har man en avlesningsfrekvens på $\frac{1}{0.1\ s} = 10$ Hz. Dermed blir endringer som skjer raskere enn 10 Hz, filtrert bort. For å visualisere hva Hall effect-sensoren måler, når det kun blir avlest hvert 100 ms, ble målingene sendt over til PC, via seriell kommunikasjon, som er forklart i kapittel 6.4.3. Plottet er vist i figur 104.



Figur 104: Detektert magnetfelt mens pumpa kjører.

Her vises det målte signalet mens pumpa kjører. Etter tre til fire sekunder, ble magneten plassert inntil Hall effect-sensoren. Dette kan man se i figur 104, ved tid 160 - 161 s, når den målte verdien har en tydelig stigning etter et lite "dypp" i grafen. Magneten blir holdt inntil sensoren i omtrent 5 sekunder, til man ser at den målte verdien tydelig synker og går tilbake til 1000 mV (1 V).

Ut ifra denne grafen, kan det konkluderes med at dersom man, i koden, setter et krav på (for eksempel) 50 målinger, som må være over en bestemt verdi etter hverandre, for å få et godkjent magnetfelt, vil dette fungere til sin hensikt. Dette er vist i figur 105, men uten at pumpa kjøres.



Figur 105: Det digitale filteret som godkjenner et magnetfelt etter 50 like målinger.

Her viser den øverste grafen det målte magnetfeltet fra Hall effect-sensoren. Den midterste grafen, viser antall like målinger som telles opp etter hverandre, der variabelen `like_maalinger_magnet` blir resatt etter å ha talt 50 like målinger. En måling er lik den forrige dersom den målte verdien er større enn 1200 mV (positiv pol på magneten), eller mindre enn 725 mV (negativ pol på magneten). Den nederste grafen, viser et internt bit i koden, som blir satt lik 1 etter 50 like målinger etter hverandre, for deretter å bli satt til 0 igjen når magnetfeltet er borte.

I praksis, er dette digitale filteret implementert i sekvensen til konkurransemodus, som er forklart i kapittel 6.2.1. Figur 105, viser kun godkjenning av detektert magnetfelt, men ikke godkjenning av forsvunnet magnetfelt. Det vil si at i sekvensen, når flyteren skal vente på å bli sluppet av ROV-en, vil samme digitale filter gjelde, men for detektering av mistet magnetfelt.

I og med at grafen, i figur 104, viser målt magnetfelt mens pumpa kjører, ser det ut til at det digitale filteret i teorien også skal fungere selv om pumpa kjører. Dette er derimot ikke testet godt nok i praksis. Det er dermed ikke usannsynlig at pumpa fortsatt vil gi nok støy til at det kan påvirke hva som telles som godkjent magnetfelt eller ikke. Dette blir derimot ikke tatt mer hensyn til i vårt system, da man har en forigling i koden, som forhindrer programmet i å lete etter et magnetfelt mens pumpa kjører.

8 Diskusjon

Innhold

I dette kapittelet skal det gjennomgås en diskusjon av utførelsen av oppgaven. Dette inkluderer også feil, mangler og begrensninger ved systemet. Det skal diskuteres rundt økonomibruk og valg som ble gjort underveis, samt en gjennomgang av hva som skal bli gjort av videre arbeid, etter oppgaven er levert inn. Utover dette, er det også en god del forslag til forbedringer som kunne blitt gjort, dersom man hadde enda mer tid til å utvide oppgaven, eller skulle startet på nytt.

8.1 Diskusjon

I og med at flyteren som er utviklet skulle være uavhengig av ROV-en, var dette en bred oppgave som inneholdt mange ulike deler av teori. Vi har vært nødt til å sette oss inn i alle de ulike delene som inngår i et sammensatt system, som inkluderer både strømforsyning, sensorsystem, styresystem, kommunikasjon og riktig grensesnitt mellom alle de ulike delene. Til tross for at spekteret av teori som inngår i oppgaven er bredt, har læringskurven vært bratt, og det har vært spennende å arbeide med denne oppgaven gjennom semesteret.

Vi har analysert og satt oss inn i flere ulike teorier på forhånd, og under planlegging, for å dimensjonere og designe systemet, med hensyn til de begrensningene som er satt i konkurransemanualen. Det er lagt høyt fokus på å ha et så energieffektivt system som mulig, med tanke på at det skal være batteridrevet. Det er i tillegg flere begrensninger fra konkurransemanualen, i forhold til hvilke typer batteri som kunne velges, og hvor store de kunne være. Det er tatt i betrakning, under dimensjonering av batteri, at dette skal vare lenger enn selve konkurransen krever, for å spare både penger og miljø, ved å ikke måtte bytte batterier for ofte under testing av systemet.

For å få til dette, har det vært nødvendig å jobbe med de ulike delene parallelt med hverandre, ettersom den ene enden i systemet påvirker den andre enden av systemet. Dette har ført til at mye tid har gått på planlegging og analysering.

Det er gjort en analyse på hvordan oppdriftsmotoren skulle lages, der man tok hensyn til energieffektivitet og kostnad. For å spare miljø, tok man et bestemt valg om at oppdriftsmotoren skulle fungere på luft, i

atmosfæretrykk, som allerede finnes i flyteren, og ikke på for eksempel olje, annen gass, eller komprimert trykk.

Oppgaven flyteren har i konkurransen er ikke stor, da den "kun" skal blir plassert ut av ROV-en, for så å utføre to profiler. Det er likevel mye som inngår i et slikt system. Vi startet tidlig med planlegging av hvordan programvaren til systemtet skulle fungere, for å kunne bestemme hvilken maskinvare som trengtes til å utføre oppgaven. Oppgaven kan løses på utallige måter, både enklere og mer komplekse enn det vi har gjort. For å gi mer tyngde i oppgaven, har vi valgt at flyteren skal styres av sensorer, og ikke av, for eksempel, timere i programvaren.

For å bruke sensorer til styring av flyteren, måtte man ha en sensor til å detektere når flyteren har nådd bunn og overflate, for å kunne utføre to profiler. Flyteren skal også kunne plasseres ut av ROV-en, før den starter utføringen av profilene. Det vil si at man måtte finne en løsning for å detektere når flyteren ble sluppet av ROV-en. Her ble en Hall effect-sensor valgt, i samarbeid med resten av gruppene UiS Subsea, til å detektere magnetfelt fra ROV-en. Det vil i kapittel 8.4 beskrives hvordan dette kunne blitt løst på en mer optimal måte, dersom man hadde hatt mer tid.

Til å styre og kommunisere fra ende til ende, ble det designet et kretskort, som fungerte som et grensesnitt mellom alle delene i systemet. Her ble vi nødt til sette oss inn i teori om både spenningsregulatorer, ulike sensorgrensesnitt, mikrokontrollere og styringsmekanismer for pumpe og ventil.

8.2 Prosjektstyring

Dette delkapittelet tar for seg aktivitetsplanen som ble laget i begynnelsen av prosjektet, og tidsbruken på de ulike deloppgavene.

8.2.1 Aktivitetsplan

I figur 106, er aktivitetsplanen vist. Aktivitetene er delt opp i fem ulike hoveddeler, med en til fire underdeler. De oransje rutene markerer hvor det ble planlagt å jobbe med de ulike områdene, og kryssene viser hvor det ble jobbet med dem i realiteten. Det skal nå gås gjennom de ulike underdelene, og hva som gjorde at det ble avvik fra planlagt tidsbruk.

Aktivitetsplan

Aktivitet		Hoved-ansvarlig	Uke																	
			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Forberedelse/opplæring	Møter		X	X	X	X	X	X	X	X		X		X	X	X		X	X	X	X
	Planlegging/bestilling		X	X	X	X	X	X	X	X	X		X								
	Essay/egenstudie					X	X	X	X		X	X	X								
Dokumentasjon	Sluttrapport	MHO		X		X	X		X	X	X		X	X	X	X	X	X	X	X	X
	Teori/utregninger						X	X		X	X	X									
Maskinvareutvikling	Skjemategning	MHO				X			X	X	X										
	Utlegg										X	X									
	Produksjon/Montering									X		X			X	X		X	X	X	
	System vannklart																				
Programvareutvikling	Programmering	HLB								X	X		X	X	X	X	X	X			
Testing	Testing av maskinvare	HLB													X	X		X	X	X	X
	Testing av programvare													X	X	X	X	X		X	X

Figur 106: Aktivitetsplan med oppdeling av aktiviteter.

Forberedelse/opplæring:

Gjennom prosjektet har vi hatt møter med veileder, når det har vært behov for det. Det har også vært ukentlige møter med UiS Subsea, der alle gruppene stilte med oppdatering på hvordan de ligger ann med oppgavene sine, og eventuelt stilt spørsmål, dersom det var noe man trengte hjelp med. I tillegg har det vært noen tilleggsmøter med deler av UiS Subsea underveis, dersom noe har trengtes å diskuteres nøyere.

Det som endret mest på aktivitetsplanen i forhold til planlagt, var underdelen Planlegging/bestilling. Det tok lengre tid enn forutsett, å finne alt av komponenter til flyteren. Som tidligere nevnt, i 8.1, påvirker alle delene av systemet hverandre. Det ble derfor brukt mer tid på planleggingen, slik at vi kunne være sikre på at alt fungerte sammen.

Fem studiepoeng av bacheloren, går til å skrive et essay på 5000 ord. Under denne posten, går også

forelesningene som hører til. Selve essayet begynte vi ikke på for fullt før bestillingen på kretskortet var lagt inn.

Dokumentasjon:

Sluttrapporten har blitt jobbet jevnt med gjennom hele semesteret. Vi har prøvd å skrive ferdig kapitler så godt som mulig underveis, slik at det man jobber med ikke blir glemt, før man får skrevet om det. Det er denne underdelen det er brukt mest tid på, akkurat som forventet fra start.

Maskinvareutvikling:

Ettersom planleggingen tok så mye lengre tid enn forventet, ble både skjemategning, utlegg og produksjon/montering, naturlig nok, utsatt. Utlegget tok derimot kortere tid enn ventet, så kretskortene ble bestilt opp i uke 10. Loddingen ble ferdig i starten av uke 13, og resten av timene som er registrert på produksjon/montering etter det, går på å plassere komponentene på en praktisk måte inne i flyteren.

Målet var å ha systemet klart til vanntesting til uke 15 (påskeuken), som markert med rødt, i figur 106. Flyteren var klar til denne fristen. Da ROV-en var klar i uke 17, ble første vanntest gjennomført. Det som er oppført av produksjon/montering etter dette, er omplasseringer av komponentene på innsiden for å få plass til den nødvendige ballasten.

Programvareutvikling: Ettersom vi fordelte ansvarsområdene maskinvareutvikling og programvareutvikling mellom oss, ble programmeringen jobbet med parallelt med skjemategningen og utlegget. Det ble jobbet jevnt med programvaren utover og tok omtrent like lang tid som planlagt.

Testing:

Testing av maskinvare, ble gjort både underveis i loddingen, etter komponentene ble loddet på, og under vanntesting. Programvaren ble også testet underveis i programmeringen, og ved vanntesting. Alle tester ble dokumentert i testrapporter og gås nærmere inn på i kapittel 7.

8.3 Videre arbeid

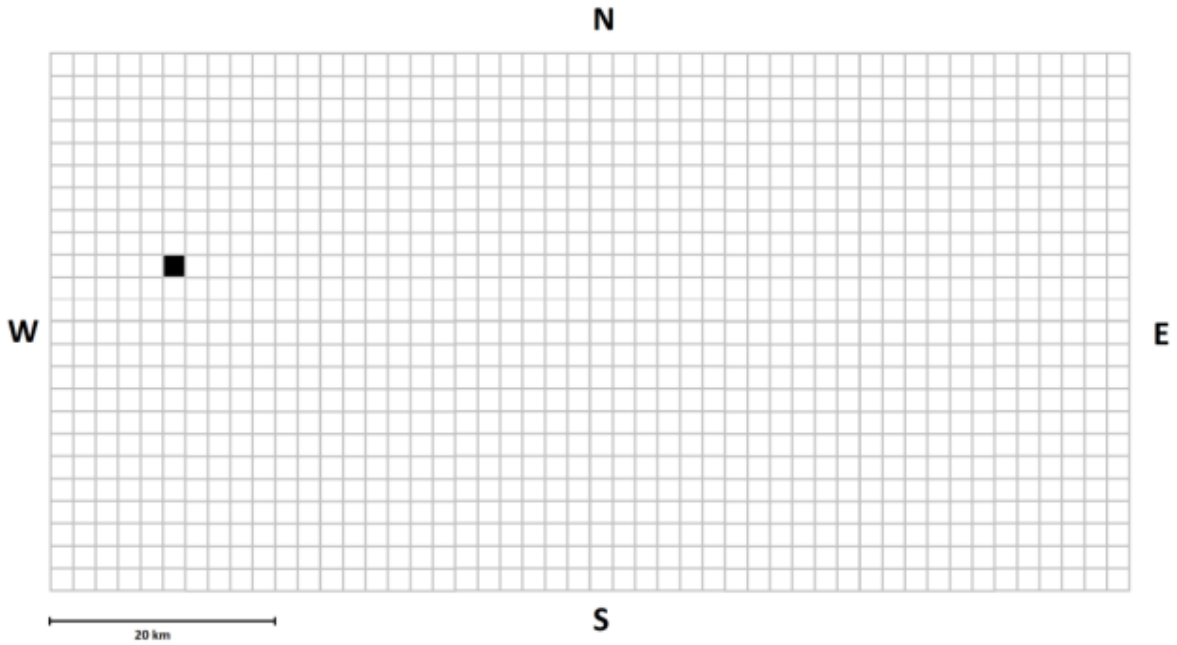
Ukene etter innlevert bacheloroppgave, vil det bli utført en del ekstra arbeid med flyteren, som man ikke har fått tid til under oppgaveskriving.

8.3.1 Beregning av teoretisk posisjon til flyter

I konkurransen, får man fem poeng for å kunne fastslå posisjonen hvor en flyter kommer til overflaten. Her er det ikke snakk om flyteren som er utviklet, men en imaginær flyter og dens teoretiske posisjon. I forkant av oppgaven får man utdelt følgende informasjon:

- Et rutenettkart, med markering der hvor flyteren var ved overflaten sist.
- Strømhastighet i vannet
- Strømretning i vannet
- Tid (i timer) til flyteren når overflaten

Ut fra denne dataen, skal man regne ut, og plotte på rutenettkartet, posisjonen hvor flyteren kommer til overflaten. Et eksempel på rutenettkartet man får tildelt er vist i figur 107.



Figur 107: Rutenettkartet man får tildelt på konkurransen. Hentet fra [20].

Man må altså kunne vise til nøyaktig rute, i rutenettkartet, hvor flyteren kommer til å dukke opp neste gang. Planen for gjennomføring av dette er å lage et *MATLAB*-skript, som tar inn dataen vi får utlevert, og som regner ut nøyaktig posisjon fra dette. Man oppnår dermed en effektiv løsning, som gir oss et raskt svar under konkurransen, ettersom man har begrenset med tid. Dette er altså en teoretisk oppgave, og skal ikke bevises fysisk.

8.3.2 Videre test med deteksjon av sluppet flyter

Til nå, er det funnet en løsning for å digitalt filtrere det målte magnetfeltet, og bruke dette som et av/på-signal. Magneten som er brukt under test frem til nå, er en kraftig magnet som kan løfte 32 kg. For at denne skulle fungere på flyteren i vann, måtte magneten plasseres nøyaktig der Hall effect-sensoren er inni flyteren, og helt inntil veggen av flyteren.

Det er bestilt inn en kraftigere magnet, som kan løfte opp til 63 kg. Det er ønskelig å ha et system, som ikke krever en så nøyaktig posisjonering av magneten, for å få riktig deteksjon. Dette fordi det er ROV-en som skal ha magneten montert på seg, for så å komme i kontakt med flyteren. Ettersom ROV-en er menneskelig styrt fra land, må vi sikre oss en løsning som tar høyde for unøyaktig posisjonering av magnet, og eventuell feiling.

Hovedproblemstillingen er hvordan ROV-en skal plukke opp flyteren. I videreutviklingsfasen, skal to metoder tas til vurdering og testing:

1. Den ene er å lage en manipulatorarm til ROV-en, som er stor nok til å gripe rundt flyteren. På denne måten kan den sterkeste magneten være montert på armen, slik at flyteren detekterer når den er sluppet. Denne metoden har følgende forutsetninger:
   - At ROV-en griper nøyaktig der Hall effect-sensoren detekterer magnetfelt.
   - At det digitale filteret er stabilt nok til at det ikke blir oppdaget falske startsignal, ved at for eksempel ROV-en "mister" kontakt med Hall effect-sensoren underveis, mens ROV-en holder flyteren.
2. Den andre metoden er å lage et håndtak på bunnen av flyteren, som ROV-en kan gripe tak i. Her kan ikke Hall effect-sensoren plasseres, da dette er på utsiden av flyteren. Denne metoden krever derfor at ROV-en manøvrerer seg til der Hall effect-sensoren er plassert, for å gi startsignal. Her må man passe på at Hall effect-sensoren ikke detekterer magnetfelt mens ROV-en fysisk holder flyteren.

8.4 Forbedringsforslag

Ved planlegging, under arbeid og i ettertid, er det oppdaget, og sett på, flere andre måter oppgaven kunne blitt løst på, samt en god del utbedringer og videreutvikling en kunne gjort på flyteren. Man skal i dette delkapittelet ta for seg de ulike forbedringsforslagene som kunne blitt gjort, dersom man hadde hatt mer tid til rådighet, eller skulle designet flyteren på ny. Dette vil også være tips og råd til fremtidige medlemmer i UiS Subsea, dersom en ny flyter skal utvikles.

8.4.1 Tilleggsfunksjoner

Det ble i kapittel 2, beskrevet at det var ønskelig at flyteren kunne lagre data, som hastighet og posisjon, og som kunne blitt hentet ut i etterkant av kjøring. Det ble også beskrevet at det var ønskelig at det ble implementert en regulering av systemet, slik at flyteren kan stoppe på en gitt dybde i vannet. Disse to funksjonene ville vi prioritert ved en videreutvikling av systemet.

Datalagring:

For å kunne logge data over tid lokalt i mikrokontrolleren, er det behov for et minne. Mikrokontrolleren *STM32F415*, som er den vi har, har et Flash-minne på 1 Mbyte. Det er ønskelig å undersøke hvordan en får tilgang til dette minnet, under kjøring, samt hvor mye plass man har i realiteten, med den eksisterende

programvaren. Alternativt, kunne man ha satt seg inn i andre måter å lagre data på, som for eksempel EEPROM-chip (Electrically Erasable Programmable Read-Only Memory) eller SD-kort (Secure Digital Memorycard).

Dataen vi er interessert i å logge, er målt trykk og tiden trykket ble målt på. Råverdien til det målte trykket, er en 24-bits verdi, men dersom man omgjør denne til millibar, under lesing, blir dette en 16-bits verdi (sett at man runder opp desimalene, og dermed får en neglisjerbar høyere unøyaktighet i målingen). Lagret tid er også en 16-bits variabel, som forklart i kapittel 6.4.3. Videre kan man regne ut hvor mange målinger, med tid og trykk (som totalt utgjør 4 byte), man kunne lagret lokalt i mikrokontrolleren før Flash-minnet er fullt, dersom man kan bruke denne.

$$\frac{1 \text{ Mbyte}}{4 \text{ byte}} = 250000 \text{ målinger} \tag{65}$$

Ettersom systemet leser ti målinger per sekund (en måling hvert 100 ms), kan man regne ut hvor lenge man kan logge verdier som i ligning 66.

$$\frac{250000 \text{ målinger}}{10 \text{ målinger/sek}} = 25000 \; sek \Rightarrow 416.6 \; min \Rightarrow 6.9 \; timer \tag{66}$$

Videre, måtte en ha satt seg inn i hvordan å skrive til- og lese fra Flash-minnet, eller eventuelt andre lagringsmetoder, slik at en kunne ha hentet dette ut, i etterkant av kjøring. Med uthentet data, kan man fra trykkmålingene regne ut posisjon og hastighet til flyteren. Flere detaljer rundt hastighet og posisjon, er beskrevet i kapittel 3.

Regulering av posisjon:

Systemet består allerede av et pådragsorgan (oppdriftsmotor), en prosess (flyteren i vannet) og en sensor som gir tilbakemelding om prosessen (trykksensor). Ved bruk av alle disse, kunne man laget et reguleringssystem som bruker trykksensoren til å måle posisjon i vannet, sammenligner dette med en referanseposisjon som er satt på forhånd, og regulerer pumpe og ventil, for å holde flyteren stabil på den bestemte posisjonen.

Regulering av pådragsorganet, i vårt eksisterende system, ville bestått av en av/på regulering av pumpe og ventil, for å opprettholde et nødvendig volum i ballongen. For å optimalisere dette, kunne pumpa og ventilen vært skiftet ut med en PWM-styrt pumpe, og en ventil som kan åpnes i ulike grader (typisk fra 0%-100%).

For at regulering av systemet skulle ha fungert optimalt med PID-regulering, måtte det ha blitt gjort en ny vurdering av hvor raske målinger en skal ta inn fra trykksensoren.

Trådløs kommunikasjon:

Dersom systemet hadde hatt både datalagring og et reguleringssystem, ville det i tillegg vært ønskelig å implementert et trådløst kommunikasjonssystem. Dette kan løses ved hjelp av en radiotranceiver, som bruker radiofrekvens til å sende og motta data. Radiosignaler kan ikke sendes eller mottas under vann, så dette måtte ha vært en antenne som ble montert på topplokket av flyteren. På denne måten, kunne man kommunisert med flyteren mens den var på overflaten, ved å sende både referanseverdier til en eventuell regulering, hente ut måledata, samt andre funksjonaliteter, som for eksempel styring eller tilbakemelding på flyterens tilstand.

8.4.2 Sikkerhetsfunksjoner

Sikkerhet er en viktig del i utvikling av ulike systemer/produkter. Flyteren har et stort forbedringspotensiale når det kommer til sikkerhet. Her skal de viktigste gjennomgås.

Lekkasjesensor:

Foreløpig, er det ikke en funksjon som sjekker for lekkasje i flyteren. Sensorgruppa i årets ROV-design, har flere lekkasjesensorer i elektronikkhuset deres. Dette ville vært hensiktsmessig å implementere i flyteren også, med eller uten trådløs kommunikasjon. En lekkasjesensor alene, ville derimot ikke gitt oss noen som helst indikasjon på at det kom vann inn i flyteren. Denne må dermed implementeres sammen med noe som kan gi tilbakemelding til oss på land. Ettersom flyteren ikke er tilkoblet land, og trådløs kommunikasjon kun fungerer over vann, ville et sterkt LED-lys på toppen, eller på utsiden av flyteren, gjort susen her som en indikator, dersom lekkasjesensoren ble utløst. Man kunne dermed lagt til i programmeringen, at profilkjøringen automatisk ble avbrutt, og at flyteren dermed flyter opp til overflaten, dersom sensoren ble utløst.

Sikkerhet mot støt:

Når flyteren synker ned til bunn, treffer den bunnen med et støt, ettersom det ikke er noe som senker farten på flyteren før den når bunnen. Dette kan utbedres ved hjelp av trykksensoren, dersom man omgjør trykk til dybde, og kalibrerer denne, slik at man får høy nøyaktighet på dybden i bassenget. Ved å vite hvor dypt bassenget flyteren skal i, kan det programmeres inn et sikkerhetssystem som blåser opp ballongen litt, for å senke farten til flyteren, før den treffer bunn.

8.4.3 Kretskortet *Plattformgrensesnitt*

Under lodding, testing og dokumentering av kretskortet *Plattformgrensesnitt* som er designet, ble det funnet en del forbedringspotensiale, som kunne vært utbedret dersom dette skulle bestilles på ny. Utbedringspotensialene er listet opp her:

- RC-filteret på Hall effect-sensorkretsen, filtrerer bort støy over 1 kHz, og har en dempning på -20 dB

per dekade. Dette er en veldig slak dempning, og det ville vært ønskelig å ha et skarpere RC-filter til analogkretsen, samt teste ut lavere knekkfrekvens, ettersom det ikke er ønskelig å få med oss så raske endringer i magnetfeltet, som man gjør nå.

- Til å omforme batterispenningen ned til 3.3 V, ble det valgt å bruke en lineærregulator, som man allerede hadde tilgang til på laboratoriet, og som vi hadde brukt tidligere. En lineærregulator er derimot mindre energieffektiv enn en svitsjet buck-regulator. Dersom vi hadde funnet en aktuell svitsjet buck-regulator, som var på lager i markedet, ville vi brukt den istedenfor.
- Når det gjelder utlegget, skulle vi brukt litt lenger tid på dette, for å optimalisere plasseringen av komponentene, samt rutingen av banene for å redusere eventuell støy, samt øke energieffektiviteten. Forslagene er listet opp nedenfor.
  - Kunne kortet ned lengden på banene til mikrokontrolleren, ved å planlegge komponentplassering samtidig som man konfigurerte pinnefunksjonene til mikrokontrolleren.
  - Ha lik lengde på I^2C-banene, for at dataen skal komme frem nøyaktig samtidig.
  - Nøyere planlegging av komponentplassering, ville resultert i mindre og mer kompakt kretskort.
  - Eget polygon på spenningslaget for batteriet, slik som man allerede har for 12 V og 3.3 V.
  - Lagt til mindre polygon i topp- og bunnlag for jord og spenning, der dette hører til.
- Estetiske forbedringer, som kunne blitt lagt til på kretskortet, er for eksempel LED-dioder. Her kunne man hatt to LED-dioder, koblet til utgangene på solid state-releene, for å indikere når pumpe og ventil er aktive. På samme måte kunne man lagt til LED-dioder på de tre ulike spenningsnivåene (batterispenning, 12 V og 3.3 V), som indikasjon på hva som er koblet til. Foreløpig har man kun en slik indikasjon koblet til 3.3 V.

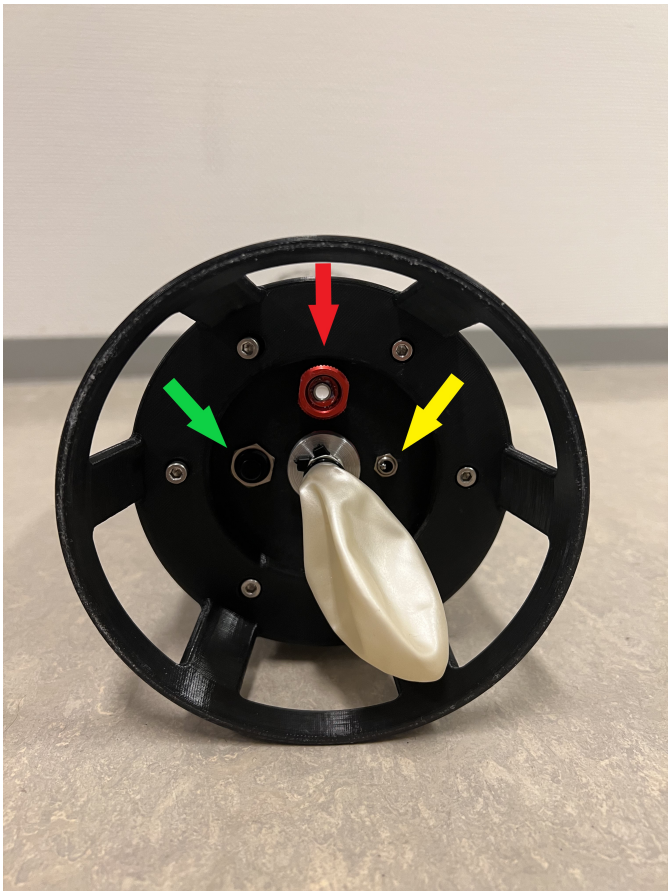
9 Konklusjon

I dette kapittelet, vil det gjennomgås hvordan oppgaven er løst samt de viktigste resultatene.

I behovsspesifikasjonen (kapittel 2.1), ble det listet opp flere punkter som flyteren måtte oppfylle. Alle disse punktene er blitt realisert, i tillegg til to av ønskene, som ikke var nødvendige, men ønskelige. Funksjonsspesifikasjonen (kapittel 2.2), gikk inn på funksjonene flyteren måtte ha for å møte behovene, med forslag til forskjellige løsninger. Her er også alle punktene realisert. Det er altså laget en fungerende flyter, som er klar for *MATE ROV Competition*.

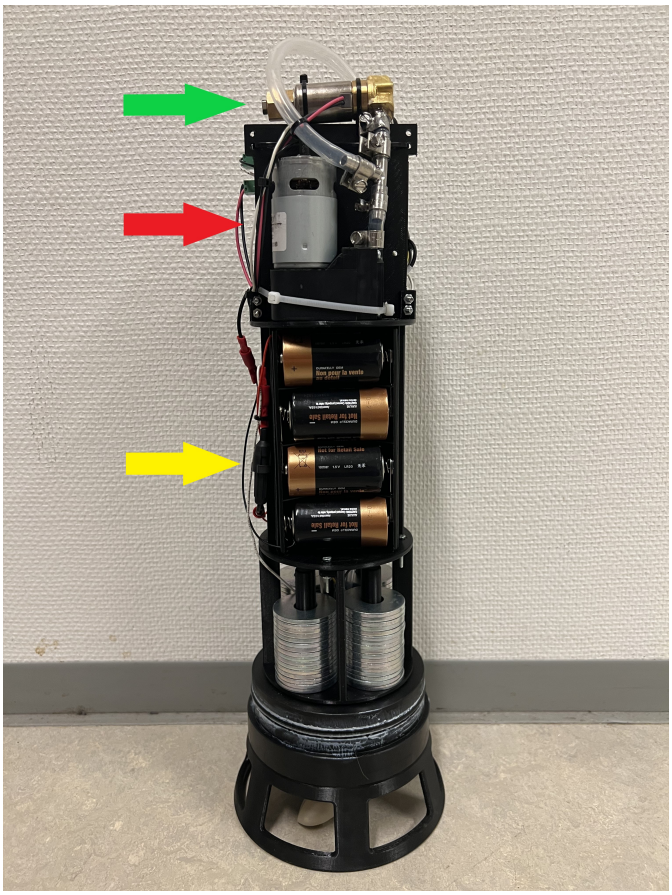
Hele flyteren er vist i figur 108. Den er laget av et rør og to endelokk, designet av maskingruppen, med ansvar for design og montering av ROV og flyter (kapittel 1.5.2). I topplokket, kan man se en rød ventilåpning, markert med rød pil. Den er der for å kunne slippe ut overtrykk manuelt. Figur 109, viser undersiden av flyteren. I midten er ballongen og rett over den, er trykksensoren festet, markert med rød pil. Til venstre for ballongen er bryteren, SW2, som brukes for å resette flyteren når den er i konkurransemodus. Denne er markert med grønn pil. Den gule pilen peker på vippebryteren, som er koblet til batteriets positive pol, for å kunne koble batteriet inn og ut.

Figur 108: Flyteren med sylinder og endelokk.

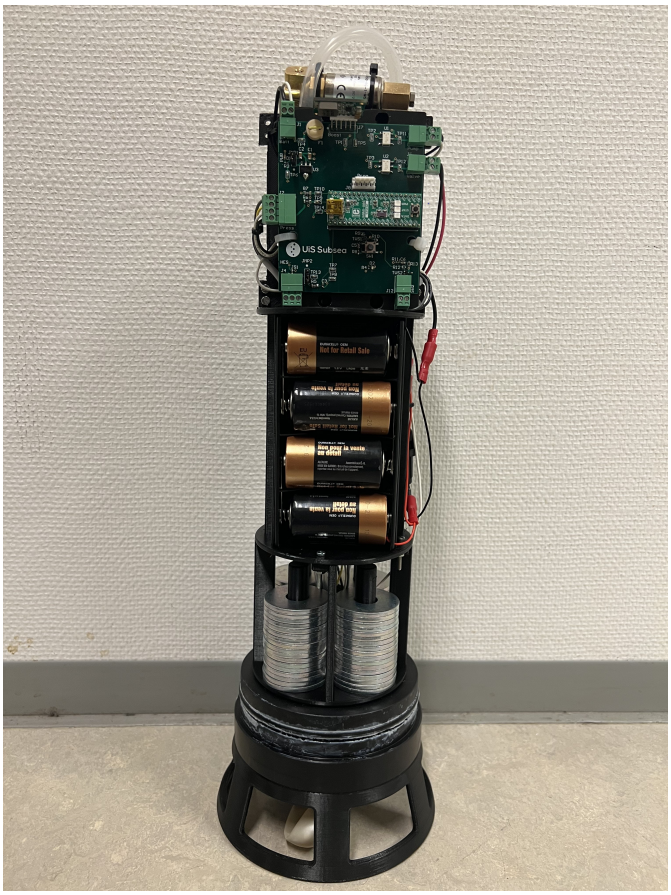


Figur 109: Flyteren fra undersiden.

I figur 110 og 111, er innsiden av flyteren vist. Nederst er det laget fire søyler, som brukes til å plassere ballasten på. Dette er også designet av Christine Nordal og Sandra Nygård fra maskingruppa. I midten, opp gjennom hele sylinderen, er det en vegg. Denne er for å feste de ulike komponentene på. På begge sidene av veggen, er det første som er festet, batteriene. Det er altså en batteriholder på hver side, med fire batterier i hver. I figur 110, kommer deretter pumpa, markert med en rød pil. Denne trekker inn luft fra flyteren, og pumper det ut i slangen over, som går videre ned til ballongen. Helt på toppen, markert med grønn pil, kan man se ventilen som slipper luften tilbake inn i flyteren igjen. 5 cm fra batteriets positive pol, er det montert en 5 A sikring på linja frem til kretskortet. Denne kan sees til venstre for batteriene, i figur 110, i en svart beholder, markert med gul pil. I figur 111, vises kretskortet som er plassert over batteriene.

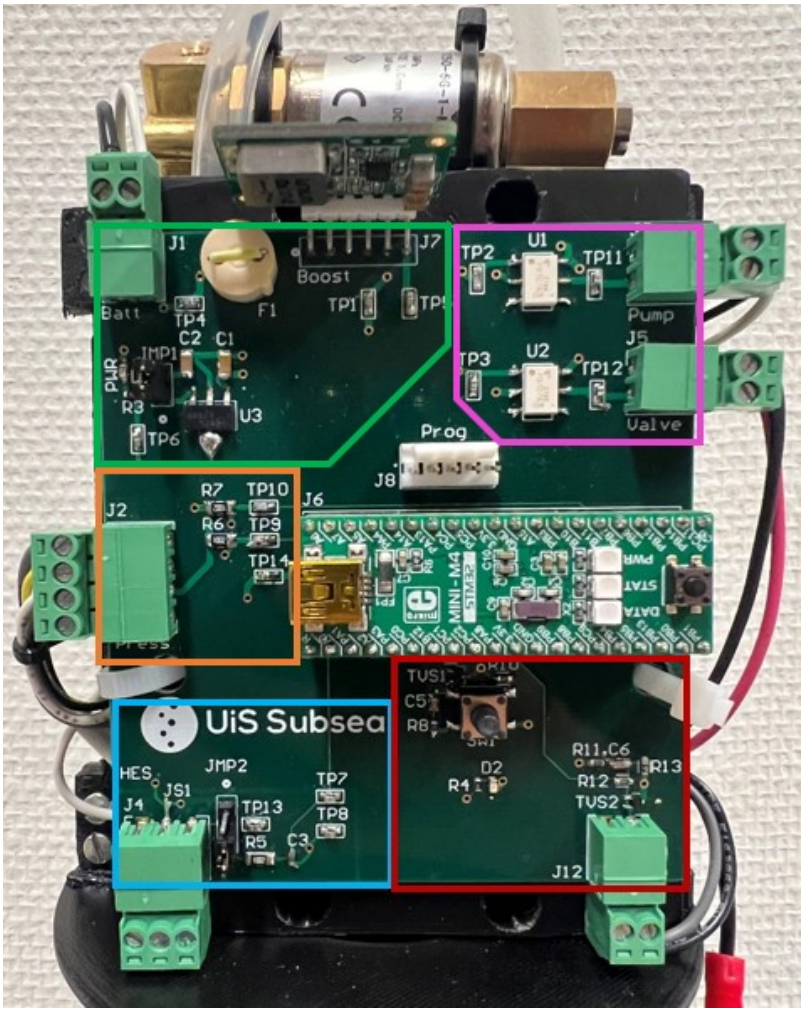


Figur 110: Siden med pumpa og ventilen.



Figur 111: Siden med kretskortet.

Kretskortet er vist nærmere i figur 112. Her kan man se alt koblet til, og at kretskortet er festet til midtveggen med strips. Over kretskortet, er også ventilen, som slipper lufta fra ballongen og inn i flyteren, vist nærmere.



Figur 112: Kretskortet i det ferdige systemet.

Kretskortet er her delt inn med samme fargekoder som i skjemategningen. Denne kan sees i vedlegg A, og gjennomgås nøye i kapittel 5. Den grønne boksen, viser kraftforsyningen med tilkoblet boost-regulator, lineærregulator og pluggen som fører til sikringen og batteriene.

Den rosa boksen, inneholder solid state-releene, og tilkoblingene fra dem til pumpa og ventilen. Motstandene som hører til, er plassert på undersiden av kortet, så de vises ikke på dette bildet. Under den rosa boksen, er tilkoblingen for programmering og plattformkortet *STM32F4 ARM MINI*.

Den oransje boksen, viser tilkoblingen til trykksensoren, med tilhørende komponenter. Som forklart i kapittel 5.4, ligger trykksensoren på et eget breakout board. Dette gjelder også Hall effect-sensoren, hvis

tilhørende krets ligger i den blå firkanten.

I den røde boksen, er knappen, SW1, samt tilkoblingen til SW2. SW2 er knappen på undersiden av flyteren, markert med grønn pil i figur 109. Her er det også lagt inn TVS-dioder, for å beskytte mot ESD. Hele kretskortet er, som tidligere nevnt, nærmere gjennomgått i kapittel 5.

Alt i alt, fungerer systemet tilfredsstillende, og til sin hensikt. Gruppen har lært mye gjennom å utvikle flyteren, og ved å være en del av et stort tverrfaglig lag. Styret og prosjektledelsen ved UiS Subsea, har også vært til god hjelp og gjort arbeidet med oppgaven mye enklere. En video av flyteren i aksjon kan sees på https://www.youtube.com/watch?v=KEAOKUOZ65Q.

Referanser

[1] National Aeronautics and Space Administration. What is drag? https://www.grc.nasa.gov/www/k-12/airplane/drag1.html. (Lastet ned 30.03.2022).

[2] Hanne Lovise Berger. Github lagringsplass. https://github.com/hannelb/Bachelor_UiS_Subsea_Float.

[3] BlueRobotics. Potted cable penetrator. https://bluerobotics.com/store/cables-connectors/penetrators/penetrator-vp/.

[4] Scott Campbell. Basics of the i2c communication protocol. https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/. (Lastet ned 10.04.2022).

[5] Advanced Circuits. PCB Trace Width Calculator. https://www.4pcb.com/trace-width-calculator.html.

[6] TE Connectivity. Ms5803-05ba, 2017. https://www.te.com/commerce/DocumentDelivery/DDEController?Action=showdoc&DocId=Data+Sheet%7FMS5803-05BA%7FB3%7Fpdf%7FEnglish%7FENG_DS_MS5803-05BA_B3.pdf%7FCAT-BLPS0011, (lastet ned 24.03.2022).

[7] SMC Corporation. Compact Direct Operated 2/3 Port Solenoid Valve For Water and Air. https://docs.rs-online.com/9a7d/0900766b815ddabf.pdf. (Lastet ned 08.02.2022).

[8] Duracell. Duracell MN1300. https://www.duracell.com/wp-content/uploads/2016/03/MN1300_US_CT1.pdf, (lastet ned 02.02.2022).

[9] Duracell. Duracell MN1400. https://www.duracell.com/wp-content/uploads/2016/03/MN1400_US_CT1.pdf, (lastet ned 02.02.2022).

[10] Horacio D Salomone et al. A non-traditional fluid problem: transition between theoretical models from Stokes' to turbulent flow, 2018. https://iopscience.iop.org/article/10.1088/1361-6404/aaa34b/pdf, (Lastet ned 09.05.2022).

[11] GO-BGC. Float technology. https://www.go-bgc.org/floats. Hentet: 01.04.2022.

[12] Finn Haugen. *Dynamiske Systemer*. Fagbokforlaget, 3. edition, 2012.

[13] Paul Horowitz Winfield Hill. *THE ART OF ELECTRONICS*. Cambridge University Press, 3. edition, 2019.

[14] Shawn Hymel. Getting started with stm32 - working with adc and dma. https://www.digikey.no/en/maker/projects/getting-started-with-stm32-working-with-adc-and-dma/f5009db3a3ed4370acaf545a3370c30c. (Besøkt 27.04.2022).

[15] Semiconductor Components Industries. MC33275, NCV33275, 2015. https://www.onsemi.com/pdf/datasheet/mc33275-d.pdf, (lastet ned 31.03.2022).

[16] Texas Instruments. DRV5053 Analog-Bipolar Hall Effect Sensor, 2015. https://www.ti.com/lit/ds/symlink/drv5053.pdf?HQS=dis-dk-null-digikeymode-dsf-pf-null-wwe&ts=1646053355324&ref_url=https%253A%252F%252Fwww.ti.com%252Fgeneral%252Fdocs%252Fsuppproductinfo.tsp%253FdistId%253D10%2526gotoUrl%253Dhttps%253A%252F%252Fwww.ti.com%252Flit%252Fgpn%252Fdrv5053.

[17] IPC. Generic Standard on Printed Board Design. http://www-eng.lbl.gov/~shuman/NEXT/CURRENT_DESIGN/TP/MATERIALS/IPC-2221A(L).pdf. (Lastet ned 25.02.2022).

[18] MATE. Hjemmeside mate center. https://www.marinetech.org/. Hentet: 30.01.2022.

[19] MATE. Hjemmeside mate rov competition. https://materovcompetition.org/. Hentet: 30.01.2022.

[20] MATE. Explorer class competition manual, 2022. https://files.materovcompetition.org/2022/2022_EXPLORER_Manual_21_JAN_2022.pdf.

[21] MATE. Explorer product demonstration prop building instructions. https://files.materovcompetition.org/2022/EXPLORER_prop_building_instructions_2022.pdf, 2022. Hentet: 01.04.2022.

[22] MATE. Mission_flythrough_2022_explorer. https://vimeo.com/679543161, 2022. Hentet: 01.04.2022.

[23] Mbari. Float profil. https://www.mbari.org/go-bgc-release/, 2020. Hentet: 31.01.2022.

[24] MikroElektronika. Mini-m4™ development board for stm32, 2012. https://www.elfadistrelec.no/Web/Downloads/_m/an/MIKROE-1367_eng_man.pdf, (lastet ned 18.03.2022).

[25] C.A. Lutken og A. Read. FRIKSJON OG VISKOSITET: bevegelsesmotstand i gasser og væsker, 13.04.2019. https://www.uio.no/studier/emner/matnat/fys/FYS2150/v19/kursmateriell/bevegelsesmotstand/bevegelsesmotstand.pdf, (Lastet ned 09.05.2022).

[26] Daniel Vasshus og Espen Christensen Myrset. Bacheloroppgave Universitetet i Stavanger. Utvikling av sensorsystem og elektronikkhus for fjernstyrt undervannsfartøy, 2021.

[27] Mikael Rodal Helgesen og Geir Arne Solland Kindingstad. Bacheloroppgave Universitetet i Stavanger. Utvikling av Mikro-ROV, 2021.

[28] Malin Harr Overland og Hanne Lovise Berger og Jørgen Hemnes Johannessen. Prosjekt - Høsten 2021, 2021.

[29] Alan S. Morris og Reza Langari. *Measurement and instrumentation Theory and Application*. Academic Press, 2. edition, 2016.

[30] Raytech. Raytech magic-gel gel potting compound 1000 g. https://docs.rs-online.com/532a/0900766b80e0bed4.pdf. (Lastet ned 12.04.2022).

[31] Schurter. Fuseholder. https://us.schurter.com/bundles/snceschurter/epim/_ProdPool_/newDS/en/typ_FDI.pdf. (Lastet ned 25.04.2022).

[32] STMicroelectronics. Discovery kit for STM32F303xx microcontrollers. https://docs.rs-online.com/5192/0900766b814876f9.pdf. (Lastet ned 17.04.2022).

[33] STMicroelectronics. Stm32f415xx stm32f417xx, 2020. https://www.st.com/en/microcontrollers-microprocessors/stm32f415rg.html, (lastet ned 23.03.2022).

[34] STMicroelectronics. Description of stm32f4 hal and low-layer drivers, 2021. https://www.st.com/resource/en/user_manual/dm00105879-description-of-stm32f4-hal-and-ll-drivers-stmicroelectronics.pdf, (lastet ned 17.03.2022).

[35] STMicroelectronics. Stm32 nucleo-64 boards, 2021. https://www.st.com/en/evaluation-tools/nucleo-f401re.html, (lastet ned 21.03.2022).

[36] UiS Subsea. Test of the float Frøya. https://www.youtube.com/watch?v=KEAOKUOZ65Q.

[37] Monolithic Power Systems. MP3429, 2017. https://www.monolithicpower.com/en/documentview/productdocument/index/version/2/document_type/Datasheet/lang/en/sku/MP3429GL-Z/document_id/1189, (lastet ned 17.03.2022).

[38] Monolithinc Power Systems. mEZD41502A-X 2.7V - 13V Input, 2A, Step-Up Power Supply, 2017. https://www.monolithicpower.com/en/documentview/productdocument/index/version/2/document_type/Datasheet/lang/en/sku/mEZD41502A-C/document_id/3592, (lastet ned 17.03.2022).

[39] Kristian Thorsen. EMI, 09-21.pdf. Tavlenotater fra ELE340, lastet ned 19.04.2022 fra Canvas.

[40] Timeanddate. https://www.timeanddate.no/vaer/norge/stavanger/siste-uke, (Besøkt 21.04.2022, Tidspunkt: 13:20 - 14:20).

[41] Toshiba. TLP3107A, 2019. https://www.digikey.no/no/products/detail/toshiba-semiconductor-and-storage/TLP3107A-TP-F/11568772?s=N4IgTCBcDaICoBkAKBmAjABgOwEEQF0BfIA, (lastet ned 02.04.2022).

[42] Von. Python tutorial - how to read data from arduino via serial port. https://www.youtube.com/watch?v=AHr94RtMj1A. (Besøkt 27.04.2022).

[43] Wikipedia. Drag coefficient. https://en.wikipedia.org/wiki/Drag_coefficient. (Lastet ned 30.03.2022).

[44] Wikipedia. Hall effect sensor. https://en.wikipedia.org/wiki/Hall_effect_sensor.

[45] Wikipedia. Sonar. https://no.wikipedia.org/wiki/Sonar.

[46] Wikipedia. Remotely operated underwater vehicle. https://en.wikipedia.org/wiki/Remotely_operated_underwater_vehicle#Educational_outreach, 2022. Hentet: 31.01.2022.

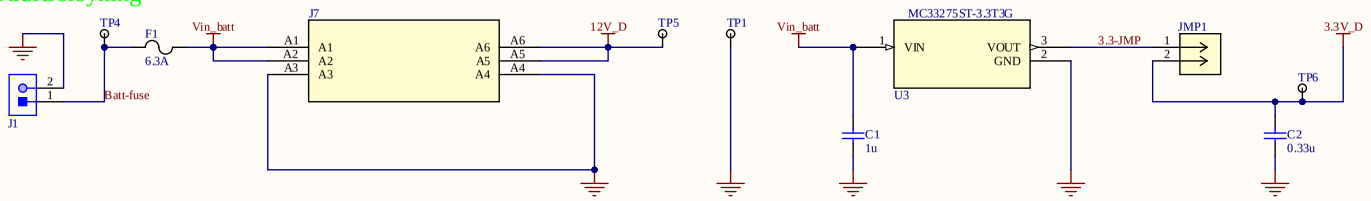
[47] Tim Wilmshurst. *An introduction to the design of small scale embedded systems*. Palgrave, 1. edition, 2001.

[48] Richard Wolfson. *Essential University Physics*. Pearson, 4. edition, 2020.

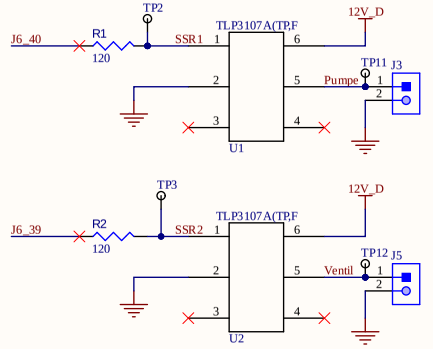
[49] Heng Wu. Power Electronics- Lecture 5 Steady-State Analysis of DC/DC Converters. https://signon.aau.dk/cas/login?service=https%3A%2F%2Fwww.moodle.aau.dk%2Flogin%2Findex.php. (Lastet ned 16.02.2022), Behøver innlogging.

A Skjemategning

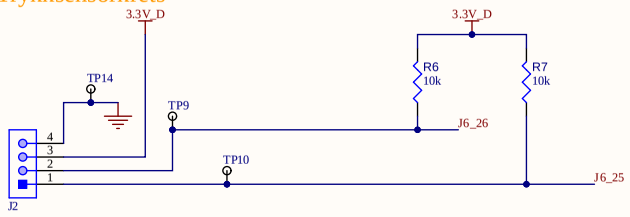
Kraftforsyning



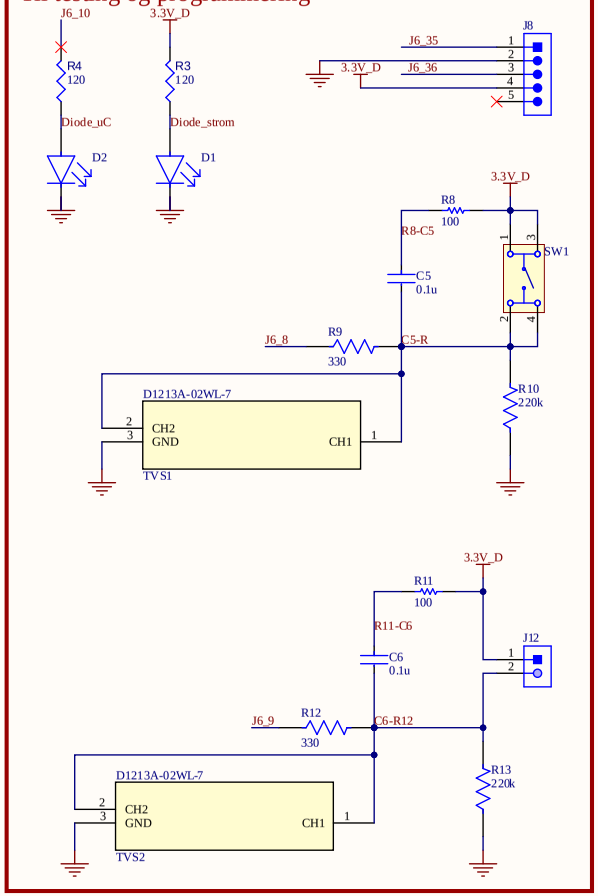
SSR til styring av pumpe og ventil



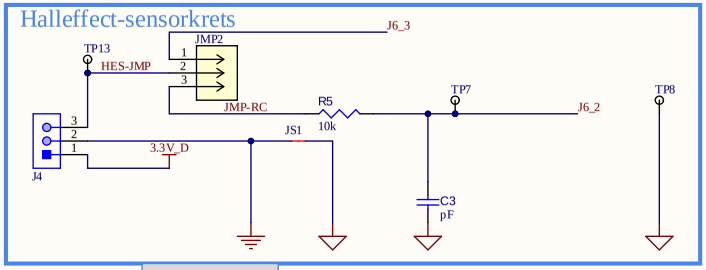
Trykksensorkrets



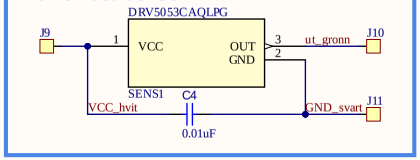
Til testing og programmering

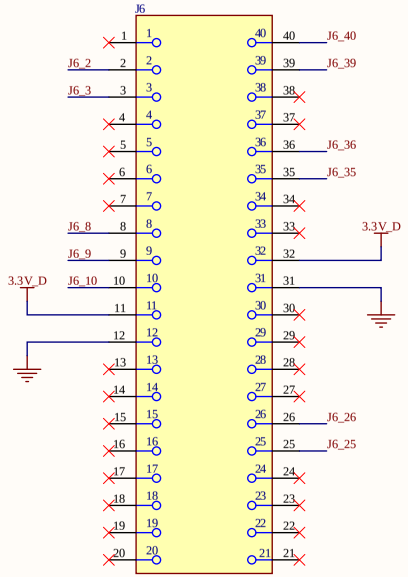


Halleffect-sensorkrets



Halleffect-sensor





Title: Plattformgrensesnitt		
Size: A3	Number: 1	Revision: 1
Date: 4.26.2022		Sheet of
File: C:\Users\..\Sheet1.SchDoc		Drawn By: Malin Harr Overland

B Testrapporter

B.1 Testrapport 1: Utladningstest

22D.1 - Intern

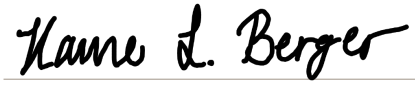
Prøverapport

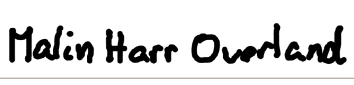
Utvikling av smart flyter

UiS Subsea, utladningstest av batteri

Forfatter(e)
Hanne Lovise Berger

1. april 2022





UTARBEIDET AV
Hanne Lovise Berger

GODKJENT AV
Malin Harr Overland

Postadresse:
www.uis.no

EMNEORD:
Batteri, batterikapasitet, leveringsevne

Prøverapport

Utvikling av smart flyter

UiS Subsea, utladningstest av batteri

RAPPORTNUMMER	VERSJON	DATO
22D.1	2.0	1. april 2022

FORFATTER(E)
Hanne Lovise Berger

OPPDRAGSGIVER(E)		OPPDRAGSGIVERS REFERANSE
Morten Tengesdal		Morten Tengesdal

PROSJEKT	GRADERING	ANTALL SIDER OG VEDLEGG
Utvikling av smart flyter	Intern	8

PRØVEOBJEKT		PRØVEOBJEKT MOTTATT
Batterioppsett		31. mars 2022

PRØVEPROGRAM	PRØVESTED	PRØVEDATO
N/A	UiS - E468	1. april 2022

SAMMENDRAG
Testrapporten innholder funksjonstesting av batterioppsettet.
Prøveresultatene gjelder kun de objekter som er prøvd.

Historikk

VERSJON	DATO	VERSJONSBESKRIVELSE
1.0	21-3-2022	Opprettet testdokument
2.0	01-4-2022	Utførte test og skrev testrapport

Innhold

1 Hva skal testes?

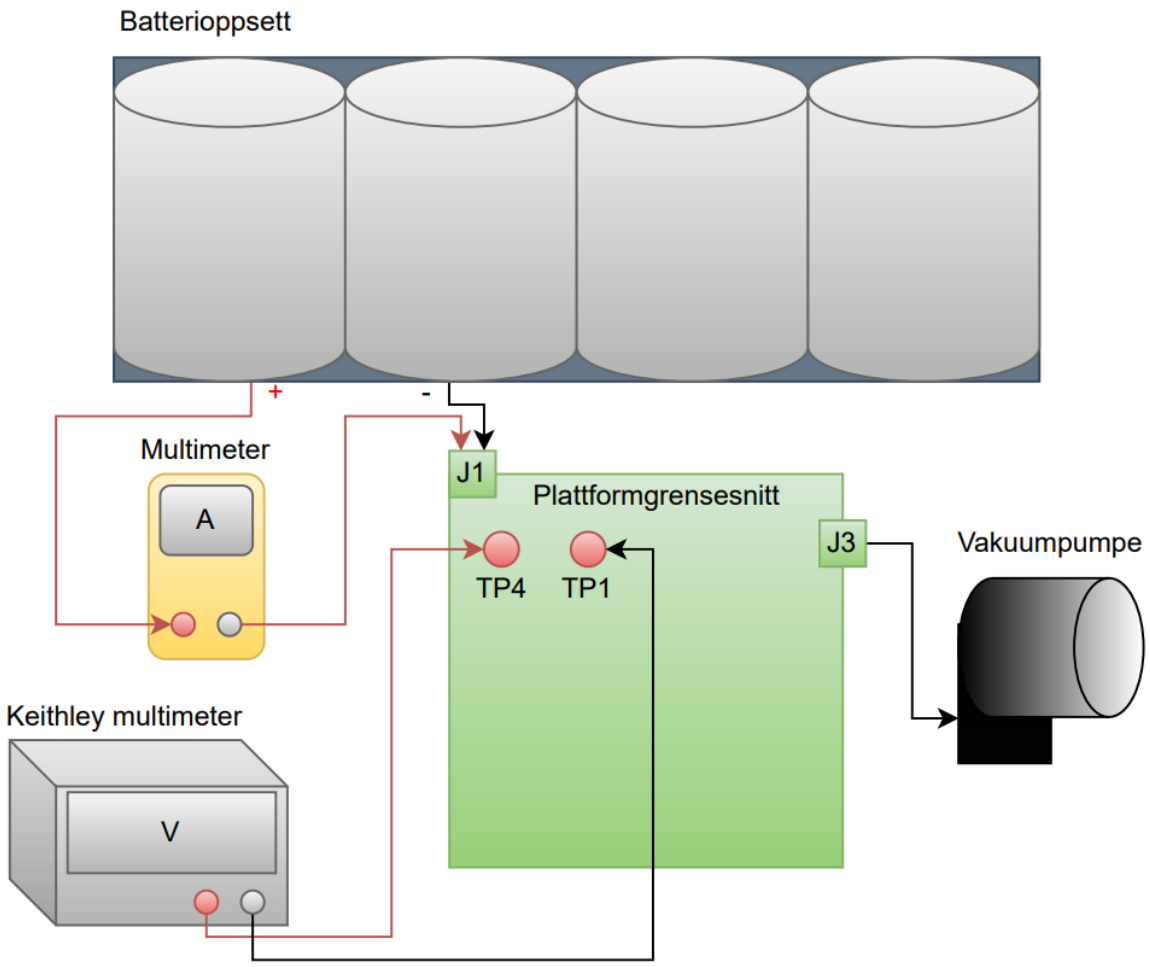
I denne testrapporten skal strømleveringsevnen og batterikapasiteten til batterioppsettet testes. Målet er å finne ut hvor lenge batterioppsettet kan levere med et konstant strømtrekk som tilsvarer et tilnærmet normalt strømtrekk i systemet. Det skal også verifiseres hvor fort og mye spenningen til batterioppsettet dropper fra 6 V ved start. Batterioppsettet som skal testes er en seriekobling av fire alkaliske type D batterier.

2 Utstyr og innstillinger

- *KEITHLEY DMM6500* digitalt multimeter (S/N: 4446162)
- Multimeter *EDM168A*
- 4x bananledninger med klype til digitalt multimeter og strømforsyning
- Vakuumpumpe
- Plattformgrensesnittet med montert mikrokontrollerkort
- 4x alkaliske type D batterier
- Diverse ledninger for tilkobling av batterioppsett til *Plattformgrensesnitt*.
- USB minnepinne

3 Oppkobling

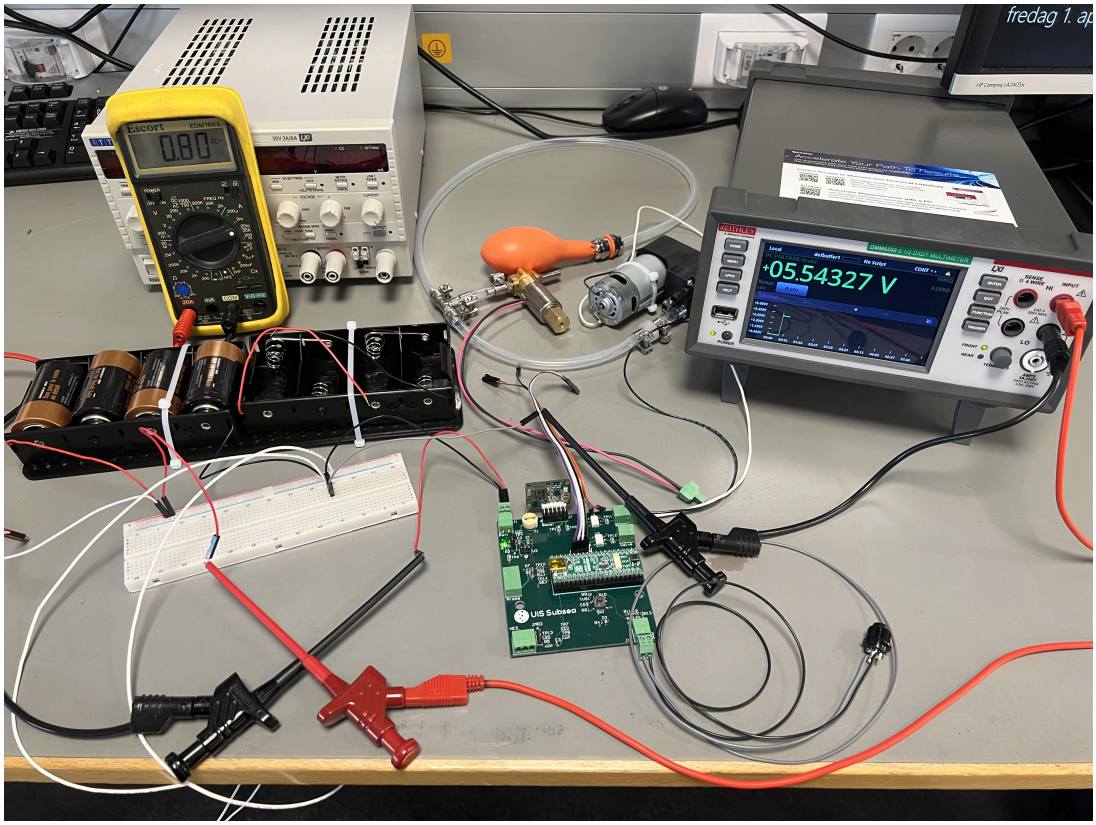
Utstyret kobles opp som vist i figur 1, hvor batterioppsettet med de fire alkaliske batteriene, koblet i serie, er vist øverst i figuren.



Figur 1: Oppkobling for batteriutladningstest.

4 Fremgangsmåte

Det gule multimeteret som er koblet i serie med den positive batteripolen, stilles inn på 20 A, og *Keithley*-multimeteret stilles inn til å måle spenning. Den positive proben kobles til TP4, og den negative til TP1, som er henholdsvis spenningen ut fra batteriet og jord. Batteripluggen kobles til J1 på *Plattformgrensesnitt*, og SW1 trykkes inn slik at pumpa starter. Lar pumpa kjøre i 30 minutter mens man observerer strømmen og spenningen som blir målt, som vist i figur 2.

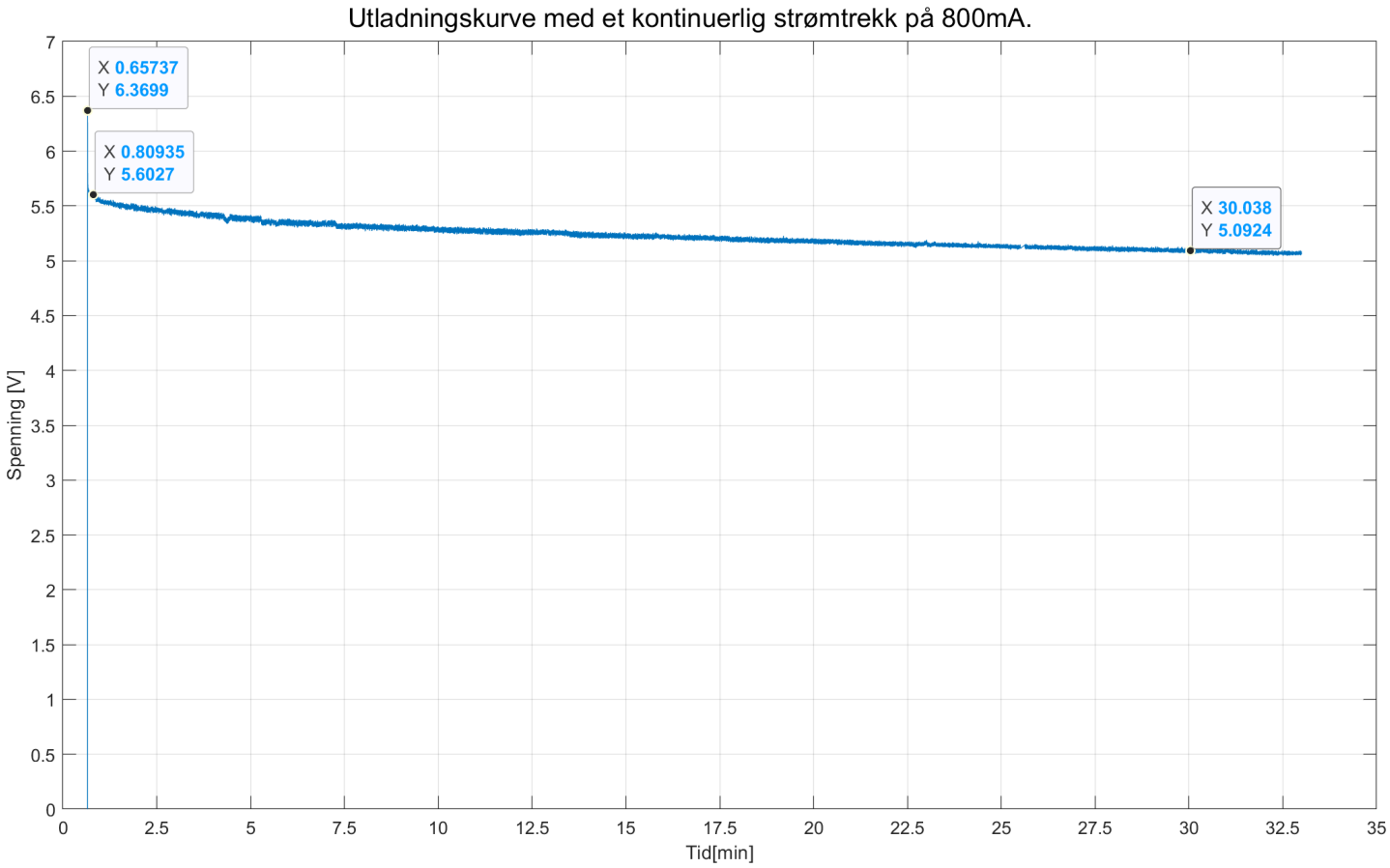


Figur 2: Oppsett av utladningstest på lab.

For å dokumentere målingene, blir det satt inn en minnepinne i multimeteret, som lagrer dataene i en Excel-fil. Disse blir så lagt inn i Matlab for å dokumenteres i en graf.

5 Resultat

Under observasjon av målingene som ble tatt mens pumpa kjørte, ble det målt et konstant strømtrekk på mellom 750 mA til 800 mA. Spenningen over batteriet startet på 6.37 V, men etter 30 minutter var den redusert til 5.11 V. Figur 3 viser utladningskurven i løpet av de 30 minuttene.



Figur 3: Utladningskurve av batterispenningen i løpet av 30 minutter.

6 Analyse og konklusjon

Som observert i figur 3, ser man at idet batteriet blir belastet, dropper batterispenningen med $6.37 - 5.60 = 0.77\ V$. Dette er på grunn av batteriets interne motstand. Man ser videre at med et konstant strømtrekk på 800 mA, holder spenningen seg relativt jevn gjennom alle de 30 minuttene, og dropper totalt $6.37\ V - 5.09\ V = 1.28\ V$. Minimumskravet vi har satt til et akseptabelt spenningsfall i løpet av 30 minutter er $6\ V - 3.6\ V = 2.4\ V$, dersom man går ut ifra at minimumsspenningen på batterioppsettet er 3.6 V. Batteriene er testet med det strømtrekket som kan forventes i praksis. Dette strømtrekket er derimot ikke forventet over lengre perioder som det er testet med her, men kun noen sekunder om gangen. Man kan derfor konkludere med at batteriet holder seg godt innenfor kvalitetskravene som er ønskelige. Denne testen ble i tillegg kjørt med en seriekobling av fire alkaliske batterier. For å doble levetiden på batterioppsettet, parallellkobles to slike seriekoblinger.

B.2 Testrapport 2: Kontinuitetstest

22D.2 - Intern

Prøverapport

Utvikling av smart flyter

UiS Subsea, test av jordingsforbindelser på kretskortet *Plattformgrensesnitt*.

Forfatter(e)
Hanne Lovise Berger og Malin Harr Overland

6. april 2022

Malin Harr Overland

UTARBEIDET AV
Malin Harr Overland

Hanne L. Berger

GODKJENT AV
Hanne Lovise Berger

Postadresse:
www.uis.no

EMNEORD:
Kontinuitet,
jordingsforbindelse, utlegg

Prøverapport

Utvikling av smart flyter

UiS Subsea, test av jordingsforbindelser på kretskortet *Plattformgrensesnitt*.

RAPPORTNUMMER	VERSJON	DATO
22D.2	2.0	6. april 2022

FORFATTER(E)
Hanne Lovise Berger og Malin Harr Overland

OPPDRAGSGIVER(E)	OPPDRAGSGIVERS REFERANSE
Morten Tengesdal	Morten Tengesdal

PROSJEKT	GRADERING	ANTALL SIDER OG VEDLEGG
Utvikling av smart flyter	Intern	6

PRØVEOBJEKT	PRØVEOBJEKT MOTTATT
Plattformgrensenitt	25. mars 2022

PRØVEPROGRAM	PRØVESTED	PRØVEDATO
N/A	UiS - KE E-468	6. april 2022

SAMMENDRAG
Testrapporten inneholder sjekk av jordingsforbindelsene på det designede kretskortet *Plattformgrensenitt*.

Prøveresultatene gjelder kun de objekter som er prøvd.

Historikk

VERSJON	DATO	VERSJONSBESKRIVELSE
1.0	01-4-2022	Opprettet testdokument.
2.0	06-04-2022	Utførte test og skrev testrapport.

Innhold

1 Hva skal testes?

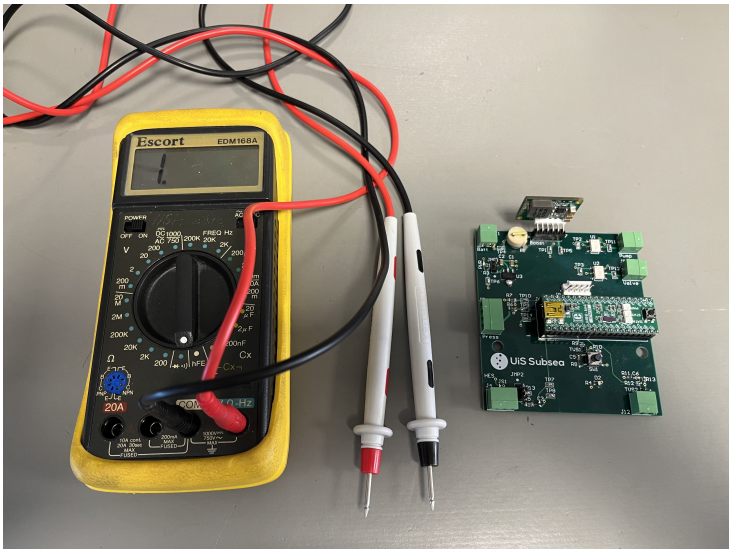
I denne testrapporten skal jordtilkoblingene på kretskortet, *Plattformgrensesnitt*, sjekkes. Det skal verifiseres at det, ved design og produsering av kretskortet, er jordforbindelse ved alle testpunkt, plugger og tilkoblinger som skal ha dette.

2 Utstyr og innstillinger

- Kretskortet *Plattformgrensesnitt*
- 2x bananplugger med prober
- Multimeter *EDM168A*

3 Oppkobling og fremgangsmåte

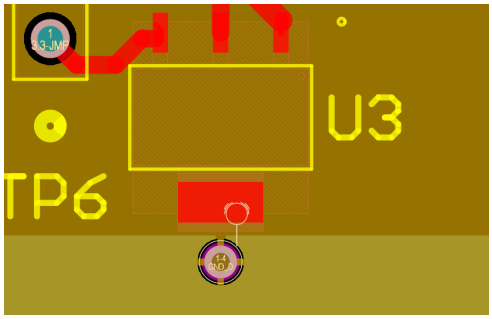
Multimeteret stilles inn på kontinuitetstestmodus, med probene tilkoblet. Probene settes mot ulike punkter på kretskortet som skal være koblet til jord (GND). Multimeteret gir dermed ut en lyd dersom det er kobling mellom punktene. Utstyret for testen er vist i figur 1.



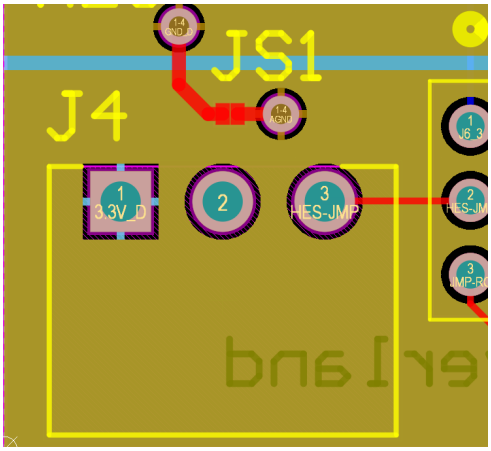
Figur 1: Utstyret for kontinuitetstesten.

4 Resultat

I testen kom det fram at loddelandet til pinne 4 på U3, og jordpinnen på J4 ikke var koblet til jord. Dette kan verifiseres i *Altium Designer*, vist i figur 2, hvor det er gjort klar en via som er koblet til jord. Her har ikke fotavtrykket hatt nettnavn, og ble dermed ikke koblet til. I figur 3 ser man på den midterste pinnen til J4 at denne heller ikke har nettnavn, og er dermed ikke blitt koblet til jord når det er brukt polygon.



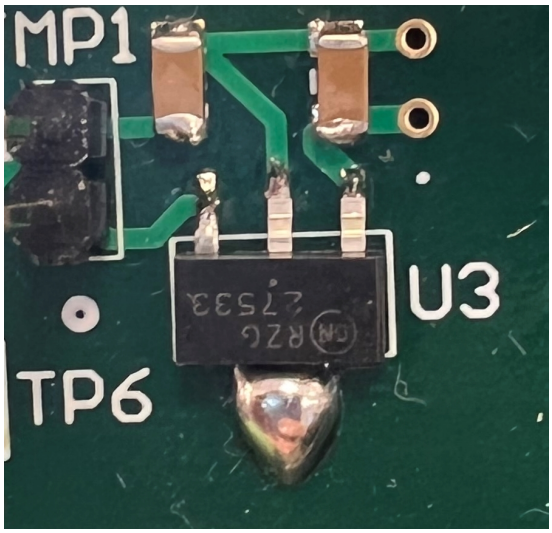
Figur 2: U3 fra utlegget i *Altium Designer*.



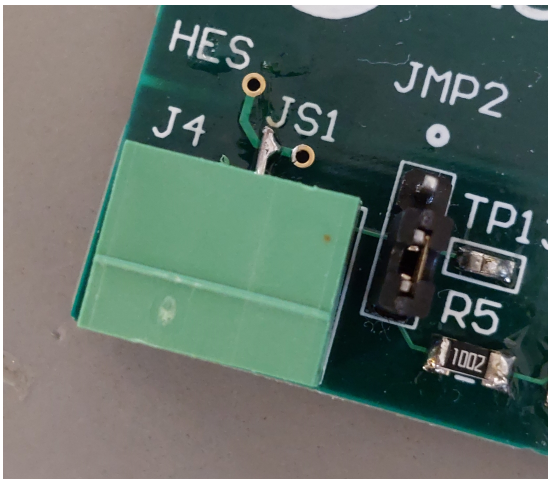
Figur 3: J4 fra utlegget i *Altium Designer*.

5 Analyse og konklusjon

For å koble U3 til jord, ble det brukt loddetinn som ble lagt over fra loddelandet og til via-en, som ligger rett under U3. For J4, ble det loddet på en liten leder fra pinnen til JS1. De to løsningene er vist i figur 4 og 5.



Figur 4: Fikset kobling mellom U3 og jord.



Figur 5: Fikset kobling mellom J4 og jord.

Dermed har alle komponenter som skal det, tilgang til jord, og jordplanet er som det skal være.

B.3 Testrapport 3: Kraftforsyning

22D.3 - Intern

Prøverapport

Utvikling av smart flyter

UiS Subsea, test av kraftforsyning

Forfatter(e)
Malin Harr Overland

28. mars 2022

Malin Harr Overland

UTARBEIDET AV
Malin Harr Overland

Hanne L. Berger

GODKJENT AV
Hanne Lovise Berger

Postadresse:
www.uis.no

EMNEORD:
Kraftforsyning,
boost-regulator,
lineærregulator

Prøverapport

Utvikling av smart flyter

UiS Subsea, test av kraftforsyning

RAPPORTNUMMER	VERSJON	DATO
22D.3	2.0	28. mars 2022

FORFATTER(E)
Malin Harr Overland

OPPDRAGSGIVER(E)	OPPDRAGSGIVERS REFERANSE
Morten Tengesdal	Morten Tengesdal

PROSJEKT	GRADERING	ANTALL SIDER OG VEDLEGG
Utvikling av smart flyter	Intern	7

PRØVEOBJEKT	PRØVEOBJEKT MOTTATT
Kraftforsyningskrets	28. mars 2022

PRØVEPROGRAM	PRØVESTED	PRØVEDATO
N/A	UiS - KE E-468	28. mars 2022

SAMMENDRAG
Testrapporten innholder funksjonstesting av kraftforsyningsdelen av kretskortet.

Prøveresultatene gjelder kun de objekter som er prøvd.

Historikk

VERSJON	DATO	VERSJONSBESKRIVELSE
1.0	28-3-2022	Opprettet testdokument.
2.0	28-3-2022	Utførte test og skrev testrapport.

Innhold

1 Hva skal testes?

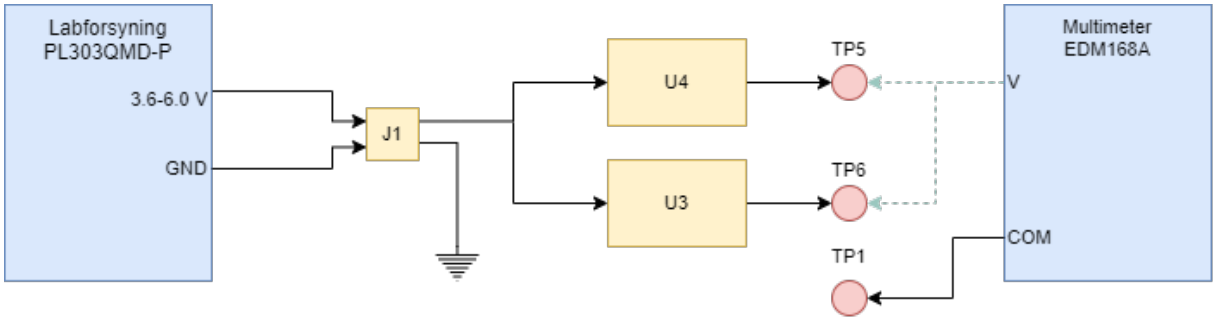
Denne testen skal verifisere at utgangsspenningen på boost- og lineærregulatoren (U4 og U3) er henholdsvis 12 V og 3.3 V. Inngangsspenningen skal varieres mellom 6 V og 3.6 V, da dette er maksimum- og minimumsspenningen som tilføres fra batteriene i det reelle systemet. Det skal også måles hva som er minimum inngangsspenning før utgangsspenningen blir påvirket.

2 Utstyr og innstillinger

- Labforsyning *PL303QMD-P*
- Multimeter *EDM168A*
- 4x bananplugger med prober
- Kretskortet *Plattformgrensesnitt* med regulatorkretsene til U3 og U4 loddet på.

3 Oppkobling

Utstyret skal kobles opp som vist i figur 1, hvor den blå stiplede linjen viser de to testpunktene multimeteret skal kobles til for å måle utgangsspenningen fra U4 og U3.



Figur 1: Koblingsskjema for test av U4 og U3.

4 Fremgangsmåte

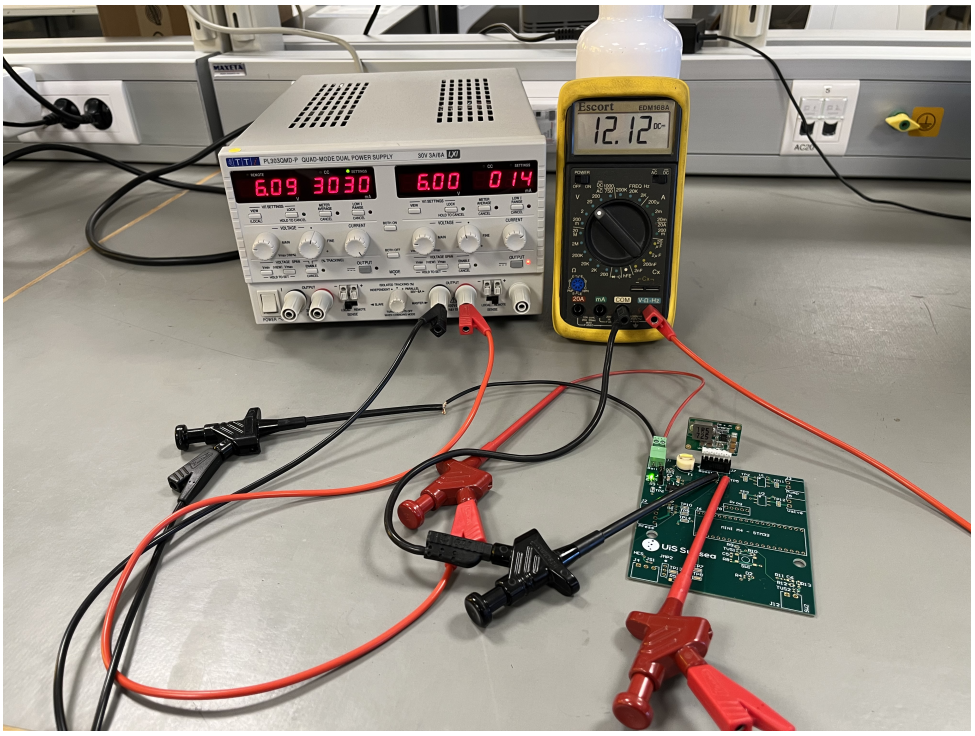
Labforsyningen stilles først inn på 6 V, og et maksimalt strømtrekk på 1000 mA. Forsyningen kobles inn på J1, og probene fra multimeteret settes på TP5 og TP1, hvor TP5 måler utgangsspenningen fra U4, og TP1 er jord. Etter å ha målt denne utgangsspenningen, skrus labforsyningen ned til først 4.5 V og deretter 3.5 V. Utgangsspenningen leses av på hvert steg, før den skrus ned helt til utgangsspenningen blir påvirket. Til slutt flyttes proben fra TP5 til TP6, for å måle utgangsspenningen fra U3. De samme målingene blir gjort her med samme innstillinger på labforsyningen.

5 Resultat

De avleste verdiene fra testen er vist i tabell 1. Ser her at utgangsverdien fra begge spenningsregulatorene holder seg stabil på alle tre inngangsspenningene. Det interne strømtrekket ble også avlest for alle inngangsspenningene og det holdt seg på 14 mA. Oppsettet for testen er vist i figur 2.

Resultater				
Regulator	6 V	4.5 V	3.6 V	Minimumsspenning
U4	12.12 V	12.12 V	12.11 V	2.4 V
U3	3.28 V	3.28 V	3.28 V	3.3 V

Tabell 1: Avleste verdier for testing av U4 og U3.



Figur 2: Oppsettet for testing av U4 og U3.

6 Analyse og konklusjon

Forventet utgangsspenning fra U4 er 12.0 V og fra U3 er det 3.3 V. U4 har en differanse på 0.12 V fra forventet verdi og U3 har en differanse på -0.02 V fra forventet verdi. Dette er så små differanser at de neglisjeres. Nedre grense på inngangsspenning til U4 ble funnet til å være 2.4 V. Dette stemmer godt overens med databladet [1]. Nedre grense for U3 ble funnet å være 3.3 V, og dette stemmer også godt med databladet [2]. Dette ser også riktig ut ettersom U3 er en lineærregulator, og dermed ikke kan regulere en spenning ned til 3.3 V dersom inngangsspenningen er under 3.3 V. Denne testen ble utført uten noe last på utgangen til U3, som tilsier at dette er tomgangsspenning. Dersom det hadde vært tilkoblet en last på utgangen, ville spenningen på utgangen hatt et høyere spenningsfall på opp til 500 mV [2].

Referanser

[1] M. P. Systems, «mEZD41502A-X 2.7V - 13V Input, 2A, Step-Up Power Supply», https://www.monolithicpower.com/en/documentview/productdocument/index/version/2/document_type/Datasheet/lang/en/sku/mEZD41502A-C/document_id/3592, (lastet ned 17.03.2022), 2017.

[2] O. Semiconductor, «MC33275, NCV33275», https://docs.rs-online.com/8736/0900766b81574569.pdf, (lastet ned 28.03.2022), 2015.

B.4 Testrapport 4: Solid State-rele

22D.4 - Intern

Prøverapport

Utvikling av smart flyter

UiS Subsea, test av solid state-rele

Forfatter(e)
Hanne Lovise Berger og Malin Harr Overland

29. mars 2022

Malin Harr Overland

Hanne L. Berger

UTARBEIDET AV
Malin Harr Overland

GODKJENT AV
Hanne Lovise Berger

Postadresse:
www.uis.no

EMNEORD:
Solid state-rele, signal til pumpe og ventil, GPIO

Prøverapport

Utvikling av smart flyter

UiS Subsea, test av solid state-rele

RAPPORTNUMMER	VERSJON	DATO
22D.4	2.0	29. mars 2022

FORFATTER(E)
Hanne Lovise Berger og Malin Harr Overland

OPPDRAGSGIVER(E)	OPPDRAGSGIVERS REFERANSE
Morten Tengesdal	Morten Tengesdal

PROSJEKT	GRADERING	ANTALL SIDER OG VEDLEGG
Utvikling av smart flyter	Intern	7

PRØVEOBJEKT	PRØVEOBJEKT MOTTATT
Solid state rele	28. mars 2022

PRØVEPROGRAM	PRØVESTED	PRØVEDATO
N/A	UiS - KE E-468	29. mars 2022

SAMMENDRAG
Testrapporten innholder funksjonstesting av solid state releene som sender aktiveringssignal til pumpe og ventil.

Prøveresultatene gjelder kun de objekter som er prøvd.

Historikk

VERSJON	DATO	VERSJONSBESKRIVELSE
1.0	29-3-2022	Opprettet testdokument.
2.0	29-3-2022	Utførte test og skrev testrapport.

Innhold

1 Hva skal testes?

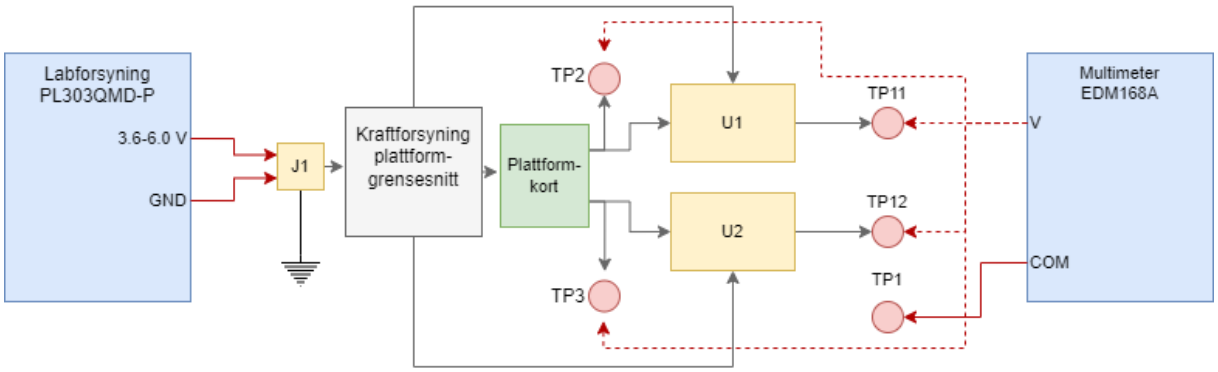
I denne testrapporten skal solid state-releene (U1 og U2), på kretskortet *Plattformgrensesnitt*, testes. Ved programaktivert GPIO-utgang på mikrokontrolleren, skal det verifiseres at det, på inngangen til U1 og U2, er en spenning på 1.65 V. Det skal også verifiseres at transistorene inni solid state-releene blir aktiverte, ved at en måler 12 V på utgangen til U1 og U2.

2 Utstyr og innstillinger

- Labforsyning *PL303QMD-P*
- Kretskortet *Plattformgrensesnitt* med plattformkortet montert på
- *STM32 Nucleo-64* programmeringskort
- Multimeter *EDM168A*
- 4x bananplugger med prober

3 Oppkobling

Utstyret skal kobles opp som vist i figur 1, hvor de røde heltrukne linjene alltid skal være tilkoblet, mens de stiplede røde linjene er testpunkt som skal kobles til en og en med samme måleprobe.



Figur 1: Koblingsskjema for test av U1 og U2.

4 Programvareverktøy

- *STM32CubeIDE 1.9.0*

5 Programfiler

Koden fra while-løkka i *main.c*-fila, som ble brukt til å aktivere U1 og U2, er vist i kodeutsnitt 1.

```
  /* ---------------------------------------------------
   * TESTKODE FOR PUMPE OG VENTIL:
  ---------------------------------------------------------*/

  // Trykk SW1 for aa kjoere pumpe.
  if(gyldig_trykk_SW1 & (Pump_Status == 0)) {
      Pump_On();
      gyldig_trykk_SW1 = 0;
  }
  if(gyldig_trykk_SW1 & (Pump_Status == 1)){
      Pump_Off();
      gyldig_trykk_SW1 = 0;
  }

  // Trykk SW2 for aa aapne ventil.
  if(gyldig_trykk_SW2 & (Valve_Status == 0)) {
      Valve_Open();
      gyldig_trykk_SW2 = 0;
  }
  if(gyldig_trykk_SW2 & (Valve_Status == 1)){
      Valve_Close();
      gyldig_trykk_SW2 = 0;
  }

```

Kode 1: Testkode for solid state-releene.

Programfilene til hele koden er lagt ved bacheloren. Denne koden er kommentert ut, og kun brukt til testing. Ved trykk av SW1, blir utgangen til U1 aktivert, og ved trykk av SW2, blir utgangen til U2 aktivert. SW1 og SW2 blir lest hvert 10. ms i SysTick_Handleren, som vist i kode 2.

```
TickTeller ++;
if(TickTeller >= 10) {
  Sjekk_SW1();
  Sjekk_SW2();
  TickTeller = 0;
}
```

Kode 2: Sjekk av bryterne i SysTick_Handleren

6 Fremgangsmåte

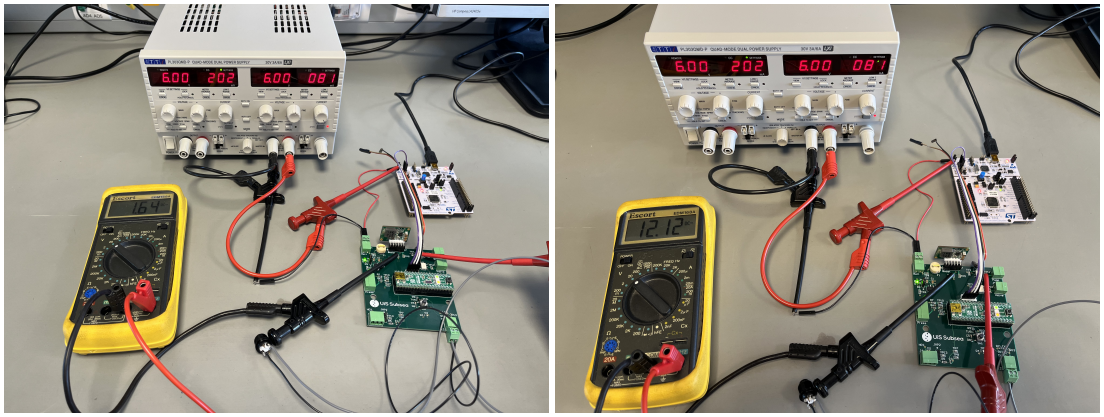
Labforsyningen stilles inn på 6 V, og med et maksimalt strømtrekk på 1000 mA. Forsyningen kobles inn på J1, og probene fra multimeteret settes på TP1 og TP2, hvor TP1 er jord og TP2 er spenningen inn på U1. Deretter flyttes proben fra TP2 til TP11 for å måle spenningen ut fra U1. Det er denne spenningen som forsyner pumpa. Trykker deretter på SW1, og måler begge de samme testpunktene igjen når GPIO-utgangen fra mikrokontrolleren er aktivert. Til slutt utføres de samme målingene, men med TP3 istedenfor TP2, og TP12 istedenfor TP11. Her brukes SW2 istedenfor SW1 for å aktivere GPIO-utgangen til U2. Dette er for å sjekke spenningen inn til og ut fra U2, som styrer ventilen.

7 Resultat

Resultatene for testen er oppsummert i tabell 1. Kan se at før knappetrykk er spenningen, både før og etter U1 og U2, null. Etter knappetrykk er spenningen 1.6 V og 12.1 V, på henholdsvis inngangen og utgangen til U1 og U2. Disse verdiene er som forventet. Oppsettet for testen er vist i figur 2.

Resultater			
		Før knappetrykk	Etter knappetrykk
U1	TP2	0 V	1.64 V
	TP11	0 V	12.12 V
U2	TP3	0 V	1.65 V
	TP12	0 V	12.12 V

Tabell 1: Avleste verdier for testing av U1 og U2.



Figur 2: Oppsettet for testing av U1 og U2.

8 Analyse og konklusjon

Ved knappetrykk, måles det en spenning på 1.65 V på GPIO-utgangene til mikrokontrolleren, som går til solid state-releene. Det vil si at motstandene (R1 og R2) som er satt på kretskortet *Plattformgrensesnitt*, for å begrense strømmen og spenningen til solid state-releet, er dimensjonert riktig. Dette utledes i kapittel 5.2 i sluttrapporten. Ettersom det måles 12 V på utgangen av solid state-releene når GPIO-utgangen er aktivert, vil det si at transistoren slås på, og det kan konkluderes med at denne delen av kretskortet virker som det skal.

B.5 Testrapport 5: Trykksensor

22D.5 - Intern

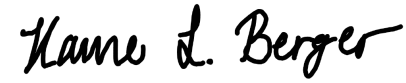
Prøverapport

Utvikling av smart flyter

UiS Subsea, test av trykksensorkrets og I^2C-protokoll.

Forfatter(e)
Hanne Lovise Berger og Malin Harr Overland

20. april 2022



UTARBEIDET AV
Hanne Lovise Berger

GODKJENT AV
Malin Harr Overland

Postadresse:
www.uis.no

EMNEORD:
Trykksensor, I2C, kommunikasjon

Prøverapport

Utvikling av smart flyter

UiS Subsea, test av trykksensorkrets og I^2C-protokoll.

RAPPORTNUMMER	VERSJON	DATO
22D.5	2.0	20. april 2022

FORFATTER(E)
Hanne Lovise Berger og Malin Harr Overland

OPPDRAGSGIVER(E)	OPPDRAGSGIVERS REFERANSE
Morten Tengesdal	Morten Tengesdal

PROSJEKT	GRADERING	ANTALL SIDER OG VEDLEGG
Utvikling av smart flyter	Intern	13

PRØVEOBJEKT	PRØVEOBJEKT MOTTATT
Trykksensor	5. april 2022

PRØVEPROGRAM	PRØVESTED	PRØVEDATO
N/A	UiS - KE E-468	20. april 2022

SAMMENDRAG
Testrapporten innholder funksjonstesting av trykksensorkrets, kommunikasjon med trykksensor via I^2C-protokoll og test av statisk målefeil.

Prøveresultatene gjelder kun de objekter som er prøvd.

Historikk

VERSJON	DATO	VERSJONSBESKRIVELSE
1.0	05-4-2022	Opprettet testdokument.
2.0	20-4-2022	Utførte test og skrev testrapport.

Innhold

1 Hva skal testes?

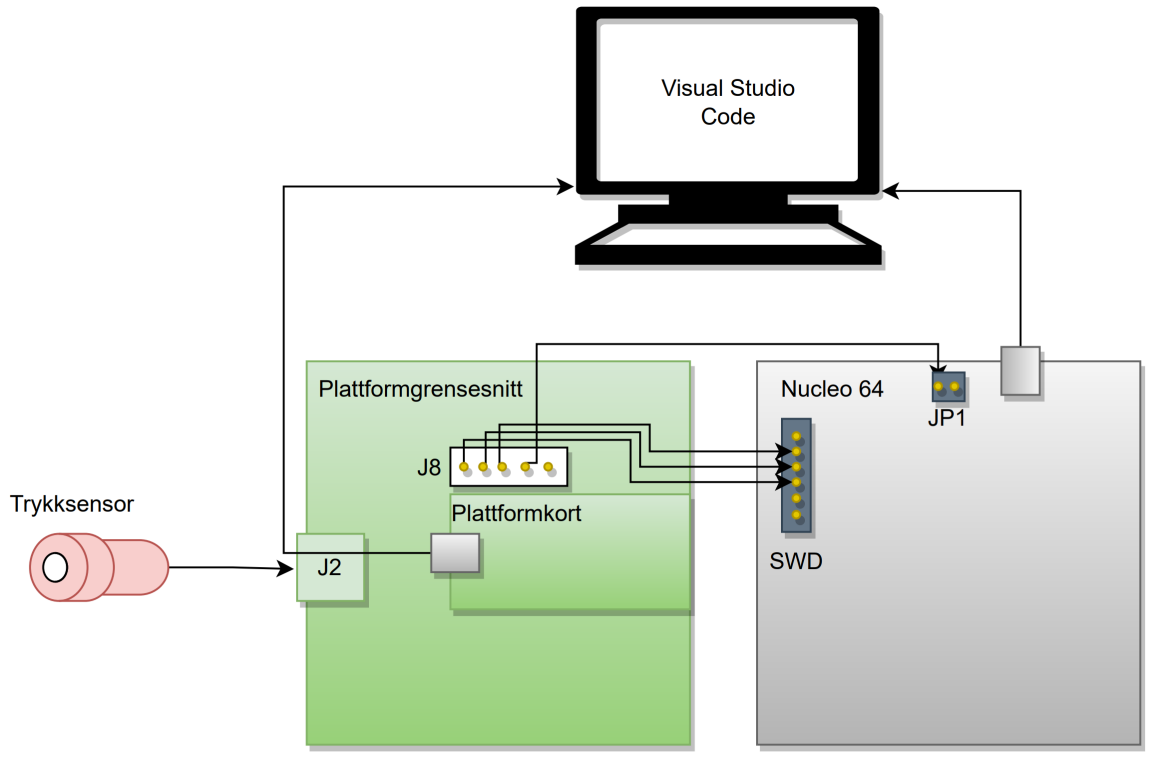
I denne testrapporten skal trykksensorkretsen på *Plattformgrensesnitt*, samt trykksensoren testes. Dette innebærer kommunikasjonsprotokollen, I^2C, mellom mikrokontrolleren og trykksensoren, samt den statiske målefeilen trykksensoren har. Fra databladet side 1 [1], skal trykksensoren ha en målefeil på ± 1.5 *mbar*.

2 Utstyr og innstillinger

- Kretskortet *Plattformgrensesnitt* med plattformkortet montert på
- Ferdig loddet og støpt trykksensor *MS5803-05BA*
- *STM32 Nucleo-64* programmeringskort
- 10 cm dypt beger med vann
- 2x mikroUSB til USB kabler

3 Oppkobling

Utstyret skal kobles opp som vist i figur 1.



Figur 1: Oppkobling for test av trykksensor og I^2C-grensesnitt.

4 Programvareverktøy

- *STM32CubeIDE 1.9.0*
- *Visual Studio Code 1.67.0* og *Python 3.8.10*
- *MATLAB R2019a*

5 Programfiler

Programkoden som blir brukt i *STM32CubeIDE* for å lese av trykksensoren, i tillegg til å sende trykk og tid til PC, er vist i kodeutsnitt 1. Denne er forklart i kapittel 6.4.3. Koden for utregning av tid, fra linje 5 til 19, er utarbeidet av Jørgen Hemnes Johannessen fra sensorgruppa. Her vises kun koden fra `while`-løkka i `main.c`-fila. Resten av koden for lesing av trykksensor, I^2C-kommunikasjon og seriell kommunikasjon, ligger vedlagt, og er forklart i kapittel 6.4.2.

```
// Les trykket og temperaturen, variablene blir lagt i structen.
TRYKKSENSOR_LesTrykk( &trykksensor );

// Setter opp variabel for millisekund
if(tid_start == 0){
  logge_tid_ms = 0;
  tid_start = HAL_GetTick();
}
else{
  oppdatert_tid = HAL_GetTick();
  logge_tid_ms = oppdatert_tid - tid_start;
}

// Setter opp variabel for sekund
if(logge_tid_ms >= 1000){
  logge_tid_ms -= 1000;
  logge_tid_sek += 1;
  tid_start = HAL_GetTick();
}

// Legger til målt trykk i databufferet for sending via USB.
dataBuf[0] = (trykksensor.trykk_comp>>16) &255;
dataBuf[1] = (trykksensor.trykk_comp>>8) &255;
dataBuf[2] = trykksensor.trykk_comp &255;

// Legger tid i ms inn i databufferet via USB
dataBuf[3] = (logge_tid_ms>>8) &255;
dataBuf[4] = (logge_tid_ms>>0) &255;

// Legger tid i sekunder inn i databufferet via USB
dataBuf[5] = (logge_tid_sek>>0) &255;
dataBuf[6] = (logge_tid_sek>>0) &255;

// Sender hele datapakken til PC.
CDC_Transmit_FS((uint8_t *) dataBuf, strlen(dataBuf));
```

Kode 1: Lesing og sending av trykk og tid.

Koden som er brukt i *Visual Studio Code*, for mottak og behandling av dataen som blir sendt fra mikrokontroller til PC, er vedlagt bacheloren. Koden er i hovedsak hentet fra [2], og ble først modifisert av Jørgen Hemnes Johannessen fra sensorgruppa. Deretter ble den modifisert til å fungere til vår hensikt. Denne koden er også vist og forklart i kapittel 6.4.3 om seriell kommunikasjon via USB. Den eneste forskjellen er at en her ikke tar i mot data fra Hall effect-sensoren. Variabelen `antallMaaling`, øverst i *Python*-fila, settes til 500. Dette bestemmer antall målinger som skal tas før programmet slutter å kjøre.

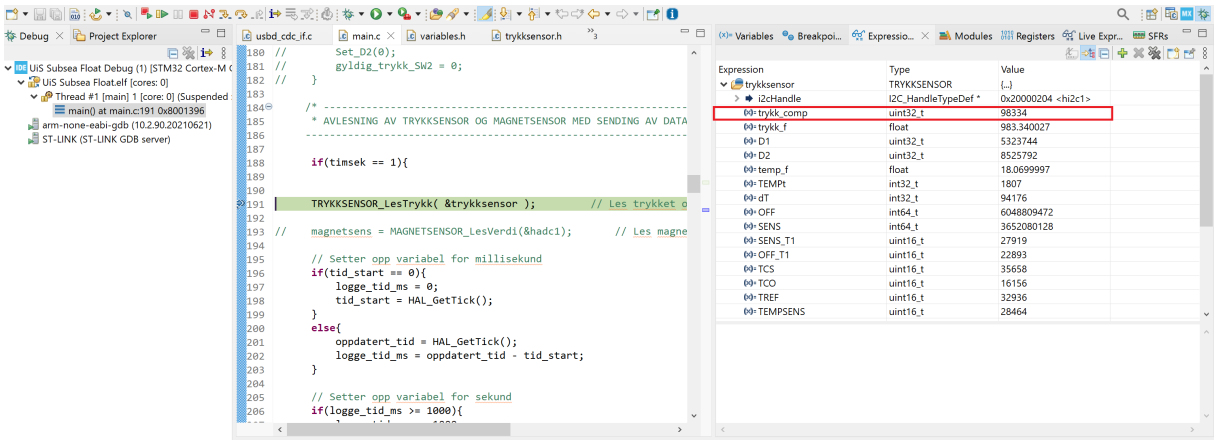
I *MATLAB* brukes koden vist i kodeutsnitt 2 til å plotte måleverdier mot tid. Hvordan denne brukes, forklares i neste kapittel.

```
plot(tid, trykk);
title('Målt trykk [mbar]');
ylabel('Trykk [mbar]');
xlabel('Tid [s]');
ylim([1019,1021]);
```

Kode 2: Kode for plotting av måleverdier med tid.

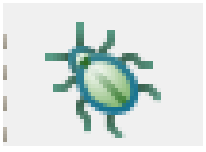
6 Fremgangsmåte

Etter å ha koblet opp utstyret, som vist i figur 1, starter man med trykksensoren liggende i åpen atmosfære. Før man går i gang med den serielle kommunikasjonen, debugger man koden i *STM32CubeIDe*, ved å sette et breakpoint ved linje 2 i kode 1. Dette er også vist i figur 2, linje 191.

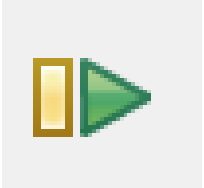


Figur 2: Debug-vinduet i *STM32CubeIDE*.

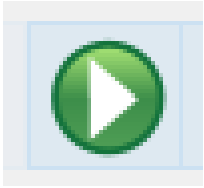
Trykker deretter på **Debug UiS Subsea Float**-knappen, vist i figur 3. Debug brukes for å kunne lese av variabelen `trykk_comp`, før den sendes til PC. Til høyre åpnes *Expressions*-fanen. Skriver inn `trykksensor` i det ene feltet, og får opp mappen med variabler, som vist til høyre i figur 2. Her ser man variabelen `trykk_comp`, som er verdien for trykk, hentet fra trykksensoren, markert i rødt. Deretter stepper man gjennom koden med **Step**-knappen, vist i figur 4, til det kommer frem verdier i mappen med variabler. Trykker deretter på **Run UiS Subsea Float**-knappen, vist i figur 5, som finnes ved siden av **Debug**-knappen.



Figur 3: **Debug**-knapp

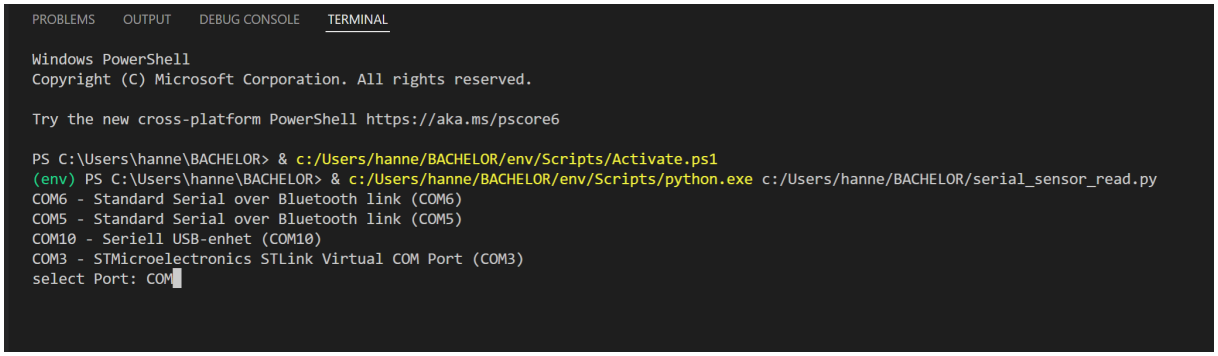


Figur 4: **Step**-knapp



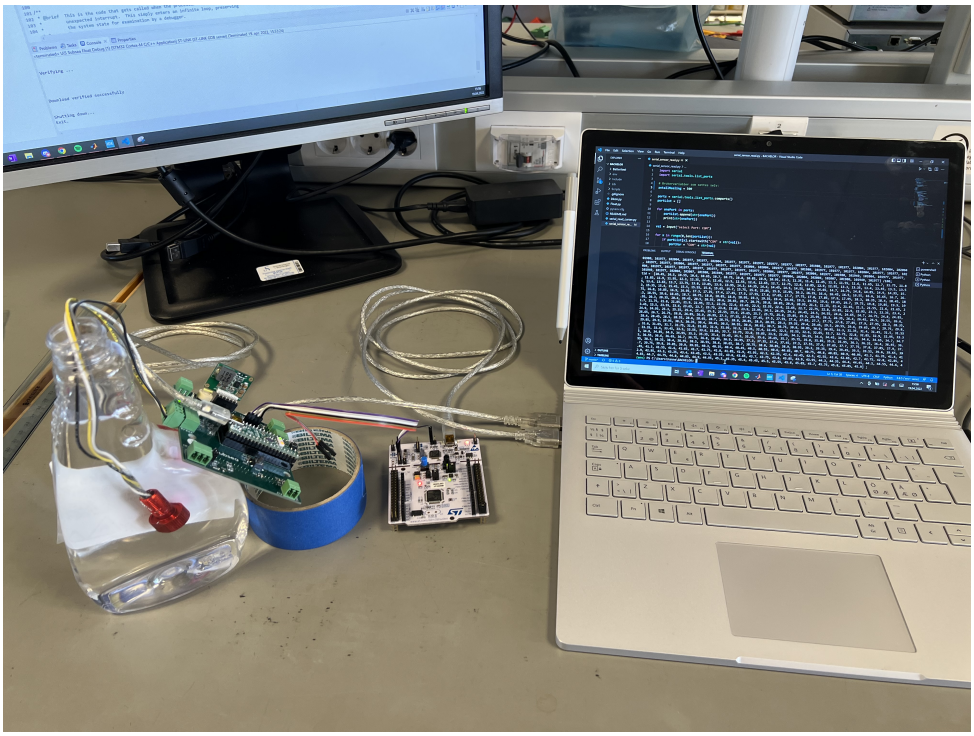
Figur 5: **Run**-knapp

Videre ønsker man å verifisere at verdien fra trykksensoren er lik det man observerte under debug av koden. Setter variabelen `antallMaalinger` til å være ti, ettersom man nå i første omgang kun ønsker å sjekke at den sendte verdien kommer frem og er lik. Den serielle kommunikasjonen settes deretter i gang via PC ved å trykke på **Run**-knappen i *Visual Studio Code*. Det blir deretter printet ut en liste med aktiverte COM-porter i terminalen i *Visual Studio Code*, samt et spørsmål om hvilken COM-port en ønsker å lese fra, vist i figur 6. Her skrives nummeret til den COM-porten som skal brukes, som i vårt tilfelle er COM10. Når man trykker `Enter` på tastaturet, logges ti måleverdier, sammen med tiden måleverdiene blir tatt på. Noterer verdiene som blir skrevet ut i terminalen i en `trykk`-vektor til senere sammenligning.



Figur 6: Valg av COM-port.

Plasserer deretter trykksensoren i vann på 10 cm dybde i begeret. Nå skal sensorens målefeil testes, derfor settes variabelen `antallMaalinger` i *Visual Studio Code* til å være 500. På denne måten får man nok data til å beregne gjennomsnittlig målefeil. Kjører deretter den serielle kommunikasjonen på ny, på samme måte som tidligere, og venter til programmet har kjørt ferdig. Oppsettet på laboratoriet er vist i figur 7.



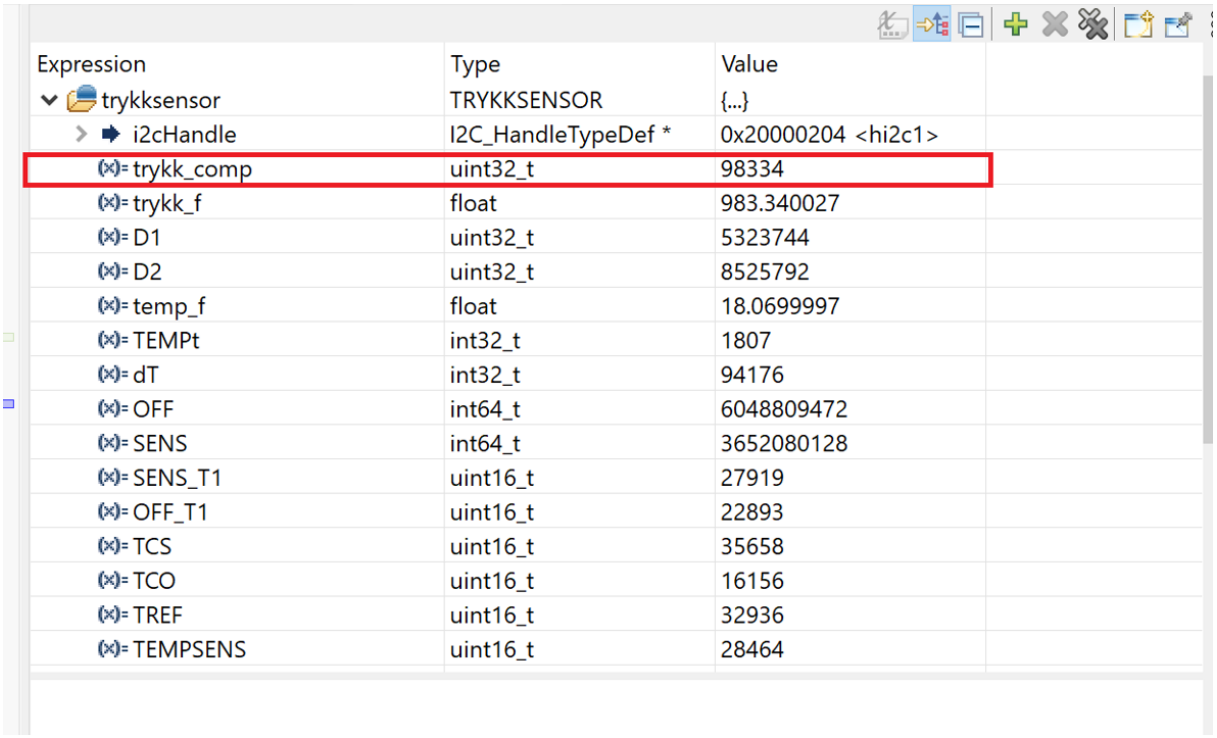
Figur 7: Laboratorieoppsettet for test at trykksensoren.

Når programmet har kjørt ferdig, skrives de ut i terminalen på *MATLAB*-format. Det som skrives ut i terminalen, kopieres og limes inn øverst i *MATLAB*-skriptet som ble vist i kode 2. Kjører til slutt *MATLAB*-skriptet, og lagrer plottet.

7 Resultat

Figur 8 viser resultatet fra debug-gjennomgangen av trykksensoren i *STM32CubeIDE*. Her ser man at variabelen `trykk_comp`, som er det målte trykket fra trykksensoren, har en digital verdi på 98334 i titallssystemet. Dette tilsvarer 983.34 mbar, som regnet ut i 1.

$$\frac{98334}{100} = 983.34\ mbar \tag{1}$$



Expression	Type	Value
trykksensor	TRYKKSENSOR	{...}
i2cHandle	I2C_HandleTypeDef *	0x20000204 <hi2c1>
trykk_comp	uint32_t	98334
trykk_f	float	983.340027
D1	uint32_t	5323744
D2	uint32_t	8525792
temp_f	float	18.0699997
TEMPt	int32_t	1807
dT	int32_t	94176
OFF	int64_t	6048809472
SENS	int64_t	3652080128
SENS_T1	uint16_t	27919
OFF_T1	uint16_t	22893
TCS	uint16_t	35658
TCO	uint16_t	16156
TREF	uint16_t	32936
TEMPSENS	uint16_t	28464

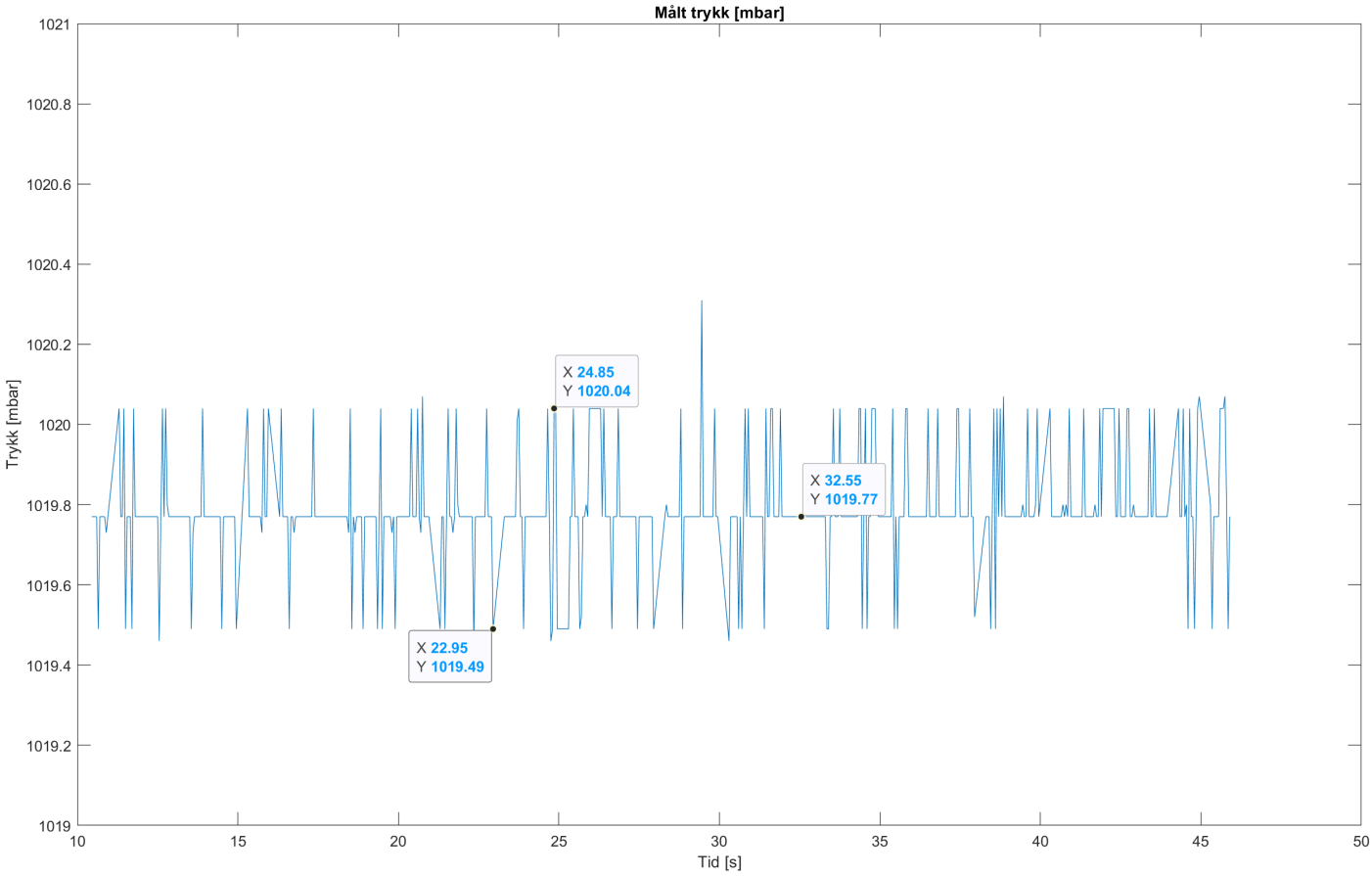
Figur 8: Verdi fra trykksensor ved debugging.

Figur 9 viser verdien på den samme variabelen fra figur 2, etter verdien er sendt over serieporten til PC. Vektoren `trykk` viser ti målinger etter hverandre, der verdiene varierer mellom 98316 og 98343, som tilsvarer mellom 983.16 *mbar* og 983.43 *mbar*. Her observerer man en tilfeldig feil på $983.43 - 983.16 = 0.27$ mbar.

```
trykk =  [98343, 98343, 98316, 98316, 98316, 98316, 98343, 98319, 98316, 98316] /100;
tid =  [5.6, 5.65, 5.7, 5.75, 5.8, 5.85, 5.9, 5.95, 6.3, 6.35] ;
```

Figur 9: Utskrift i terminalen i *Visual Studio Code*.

Figur 10 viser *MATLAB*-plottet av de 500 målingene, fra trykksensoren 10 cm under vann, mot tid.



Figur 10: Plott fra *MATLAB*-skriptet av målt trykk.

Her er verdiene fra variabelen `trykk_comp`, fra *STM32CubeIDE*, delt på 100, for å gjøres om til millibar. Man observerer at verdien, målt av trykksensoren, varierer mellom 1019.49 mbar og 1020.04 mbar, med en midtverdi på 1019.77 mbar.

8 Analyse og konklusjon

Fra resultatet av testene, ser man først og fremst at råverdien målt av trykksensoren, som legges i variabelen `trykk_comp`, er 98334 i *STM32CubeIDE* som tilsvarer 983.34 mbar. Man ser videre at de ti mottatte verdiene, i *Visual Studio Code*, varierer mellom 983.16 mbar 983.43 mbar. Ettersom verdien målt i *STM32CubeIDE* er innenfor de verdiene som ble sendt over seriell kommunikasjon, kan det konkluderes med at den serielle kommunikasjonen virker.

Verdien kan også dobbelsjekkes ved å se i databladet side 8 [1]. Der står det at råverdien til variabelen `trykk_comp`, som i databladet blir kalt P, kan variere mellom 0 og 600000. Råverdien til det målte trykket regnes om til bar, som vist i utregning 2.

$$\frac{98343}{100} = 983.43\ mbar = 0.98\ bar \tag{2}$$

Ettersom trykksensoren i denne testen lå plassert på bordet i åpen atmosfære, kan det konkluderes med at et målt trykk på omtrent 0.98 bar er fornuftig. Det gjennomsnittlige atmosfæretrykket for april lå på 1.013 bar [3]. Differansen mellom målt og reelt trykk, var dermed omtrent $1.01 - 0.98 = 0.03$ bar, som tilsvarer 30 mbar. Denne målefeilen kan skyldes både tilfeldig feil i trykksensoren, men også unøyaktig reelt trykk på akkurat denne dagen, ettersom dette er gjennomsnittlig trykk for en hel måned.

Ved måling av trykket 10 cm under vann, fikk man en graf med 500 målinger, vist i figur 10. Den gjennomsnittlige differansen mellom topp- og bunn-verdiene er $1020.04 - 1019.49 = 0.55\ mbar$. Det er beskrevet i databladet at det kan forventes en nøyaktighet på $\pm 1.5\ mbar$. At målt verdi har en differanse på 0.55 mbar er dermed godt innenfor, og det kan konkluderes med at trykksensorens nøyaktighet er som forventet.

Det målte trykket når trykksensoren er plassert 10 cm under vann, kan teoretisk regnes ut å være som vist i likning 3. Likningen er hentet fra kapittel 4.2.2. Her har man distansen, d, som er 0.01 m, tettheten i vann, ρ, som er 1000 kg/m^3, og jordas gravitasjonskraft, g som er $9.81 m/s^2$. Overflatetrykket p_0, tar man fra det som ble målt til å være atmosfæretrykket, altså 983.43 *mbar*.

$$\begin{aligned} p &= d\rho g + p_0 \\ &= 0.01 \cdot 1000 \cdot 9.81 + 983.43\ mbar \\ &= 1081.53\ mbar \end{aligned} \tag{3}$$

Sammenligner man teoretisk verdi med gjennomsnittlig målt verdi, som er 1019.77 *mbar*, får man en differanse på $1081.53 - 1019.77 = 61.76\ mbar$ mellom teoretisk og reell verdi. Dette er den statiske målefeilen som blir observert, og tilsvarer en målefeil på omtrent 60 cm i dybde. Her har man ikke sett på hele måleområdet til trykksensoren. Det gjøres i testrapport 7 i vedlegg B.7, og blir diskutert videre i kapittel 7.4. Til nå kan det konkluderes med at trykksensoren virker, samt koden som er skrevet for denne.

Referanser

[1] M. Specialties, «MS5803-05BA Miniature Altimeter and Diving Module», https://docs.rs-online.com/5840/0900766b8142cddd.pdf, (lastet ned 05.04.2022), 2013.

[2] Von, «Python Tutorial - How to Read Data from Arduino via Serial Port», https://www.youtube.com/watch?v=AHr94RtMj1A, (Besøkt 27.04.2022).

[3] Timeanddate, https://www.timeanddate.no/vaer/norge/stavanger/siste-uke, (Besøkt 21.04.2022, Tidspunkt: 13:20 - 14:20).

B.6 Testrapport 6: Hall effect-sensor

22D.6 - Intern

Prøverapport

Utvikling av smart flyter

UiS Subsea, test av Hall effect-sensor og analogkretsen på kretskortet *Plattform-grensesnitt*.

Forfatter(e)
Hanne Lovise Berger og Malin Harr Overland

11. april 2022

Malin Harr Overland

UTARBEIDET AV
Malin Harr Overland

Hanne L. Berger

GODKJENT AV
Hanne Lovise Berger

Postadresse:
www.uis.no

EMNEORD:
Hall effect-sensor,
analogsignal, magnetfelt

Prøverapport

Utvikling av smart flyter

UiS Subsea, test av Hall effect-sensor og analogkretsen på kretskortet *Platt-formgrensesnitt*.

RAPPORTNUMMER	VERSJON	DATO
22D.6	2.0	11. april 2022

FORFATTER(E)
Hanne Lovise Berger og Malin Harr Overland

OPPDRAGSGIVER(E)		OPPDRAGSGIVERS REFERANSE
Morten Tengesdal		Morten Tengesdal

PROSJEKT	GRADERING	ANTALL SIDER OG VEDLEGG
Utvikling av smart flyter	Intern	12

PRØVEOBJEKT		PRØVEOBJEKT MOTTATT
Hall effect-sensor		5. april 2022

PRØVEPROGRAM	PRØVESTED	PRØVEDATO
N/A	UiS - KE E-468	11. april 2022

SAMMENDRAG
Testrapporten innholder funksjonstesting av Hall effect-sensoren og tilhørende analogkrets på kretskortet *Plattformgrensenitt*.

Prøveresultatene gjelder kun de objekter som er prøvd.

Historikk

VERSJON	DATO	VERSJONSBESKRIVELSE
1.0	05-4-2022	Opprettet testdokument.
2.0	11-04-2022	Utførte test og skrev testrapport.

Innhold

1 Hva skal testes?

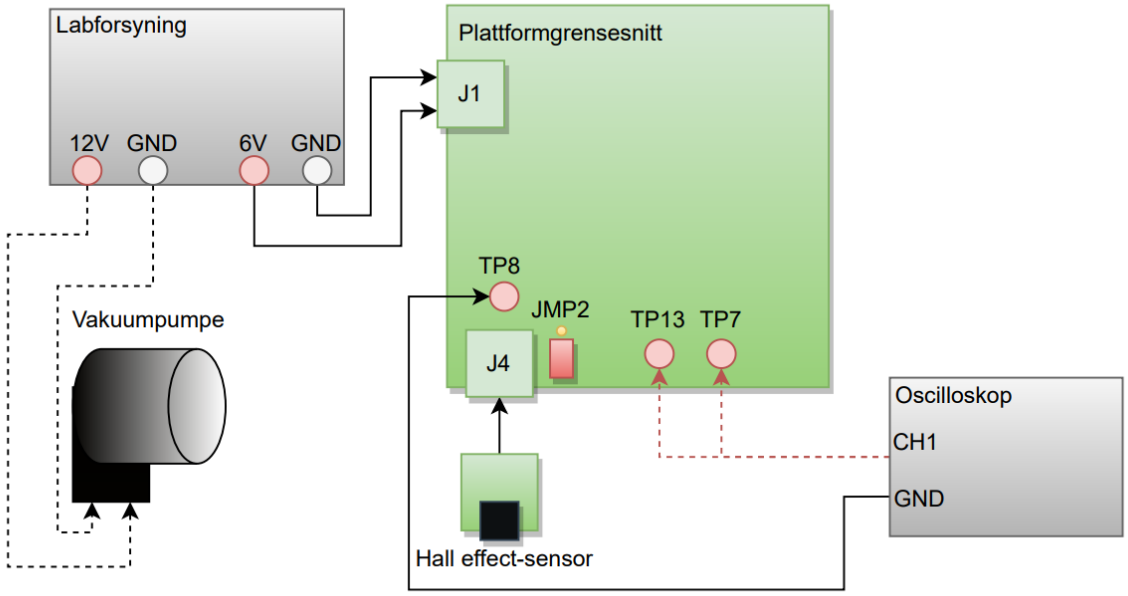
I denne testrapporten skal Hall effect-sensoren testes, samt tilhørende analogkrets på kretskortet *Plattformgrensesnitt*. Testrapporten skal verifisere funksjonaliteten til Hall effect-sensoren, og avdekke hvor mye støy pumpa utgjør, i form av magnetisme, når den kjøres. Det skal også undersøkes om det er mulig å filtrere bort eventuell støy fra pumpa, eller om systemet kan fungere til tross for støyen.

2 Utstyr og innstillinger

- Kretskortet *Plattformgrensesnitt* med plattformkortet montert på.
- Hall effect-sensor *DRV5053CAQLPG*
- Labforsyning *PL303QMD-P*
- 2x bananplugger med prober
- *STM32 Nucleo-64* programmeringskort
- Oscilloskop *Keysight InfiniiVision DSOX3054T*
- 2x Oscilloskopprober
- USB minnepinne
- Vakuumpumpe
- Magnet

3 Oppkobling

Utstyret skal kobles opp som vist i figur 1. Her er de røde stiplede linjene ulike målepunkt oscilloskopprobene skal kobles til, en av gangen. De svarte stiplede linjene er oppkoblingen til pumpa, denne skal kobles opp etterhvert i testen.



Figur 1: Oppkobling av testoppsett.

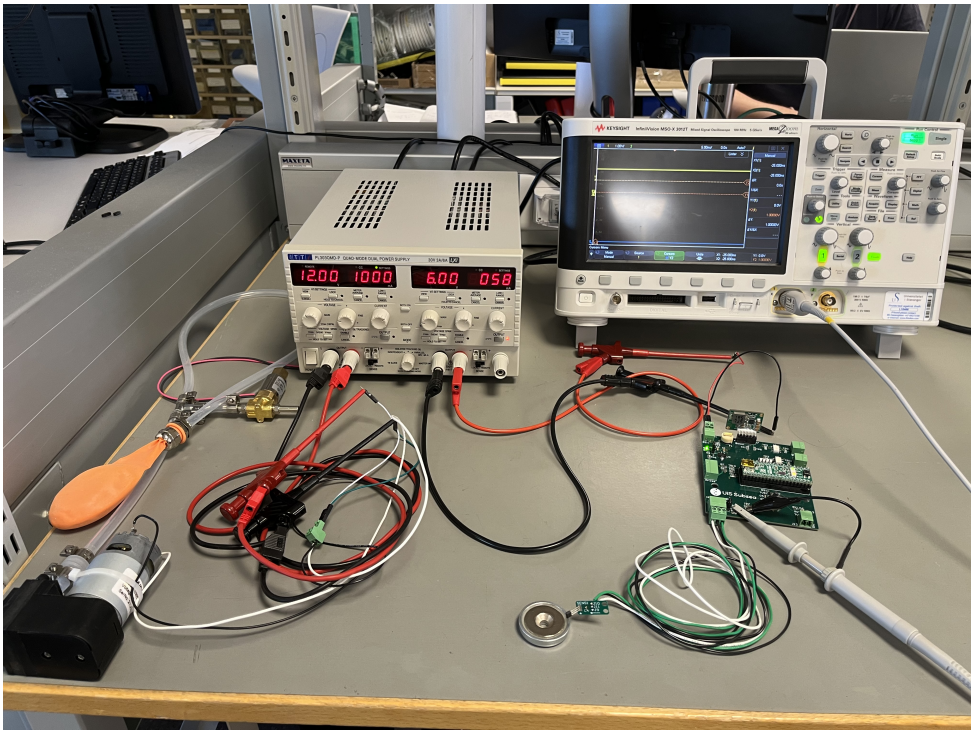
Labforsyningen har to forsyningsutganger. Midtstillingsbryteren vris helt til venstre, til *INDEPENDENT*, slik at det kan gis ut forskjellige spenninger på de to utgangene. Den første stilles inn på 12 V, med maksimalt strømtrekk på 1000 mA. Den andre stilles inn på 6 V, med maksimalt strømtrekk på 1000 mA.

4 Fremgangsmåte

Etter utstyret er koblet opp, plasseres Hall effect-sensoren 40 cm fra pumpa. Dette er den maksimale avstanden den vil kunne ha til pumpa. Labforsyningen, som er stilt inn på 6 V, skrus på. Med Hall effect-sensoren koblet til kretskortet, kobles måleproben fra oscilloskopet på TP13, for å måle det ufiltrerte signalet. Jord kobles til TP8, som er analog jord. Lagrer deretter signal fra oscilloskop, og skrur på labforsyning, stilt inn på 12 V, slik at pumpa kjører. Stopper signalet i oscilloskopet, og lagrer dette i tillegg. Dette er det ufiltrerte støysignalet.

Deretter skal det filtrerte støysignalet måles. Labforsyningen til pumpa skrus av, og oscilloskopproben flyttes fra TP13 til TP7. Skrur på labforsyningen til pumpa igjen, og tar flere bilder av signalet på oscilloskopet mens pumpa kjører.

Til slutt skal signalet fra Hall effect-sensoren når magneten er i nærheten måles, både ufiltrert, filtrert og mens pumpa kjører. Pumpa kobles fra, og oscilloskopproben flyttes tilbake til TP13. Magneten flyttes gradvis nærmere Hall effect-sensoren, og signalet ut blir observert på skopet. Kobler deretter skopproben til TP7, og magneten flyttes igjen gradvis nærmere Hall effect-sensoren. Laboratorieoppsettet er vist i figur 2.

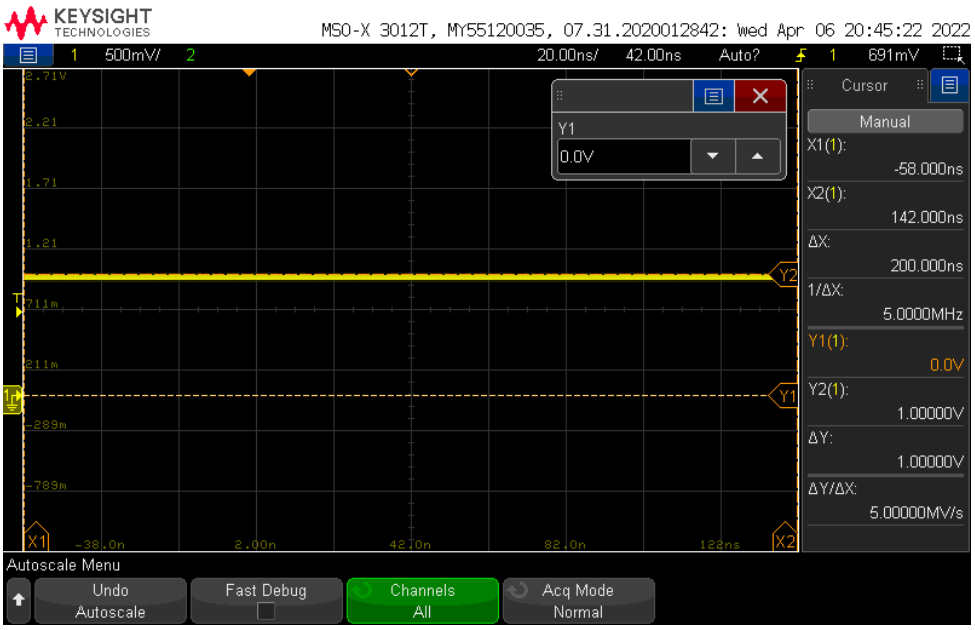


Figur 2: laboratorieoppsett for test av Hall effect-sensor.

Mens det filtrerte signalet måles, skrus pumpa på, og magneten flyttes gradvis nærmere igjen. Lagrer bilde fra oscilloskopet, som er det målte signalet med både pumpa på og magneten i nærheten.

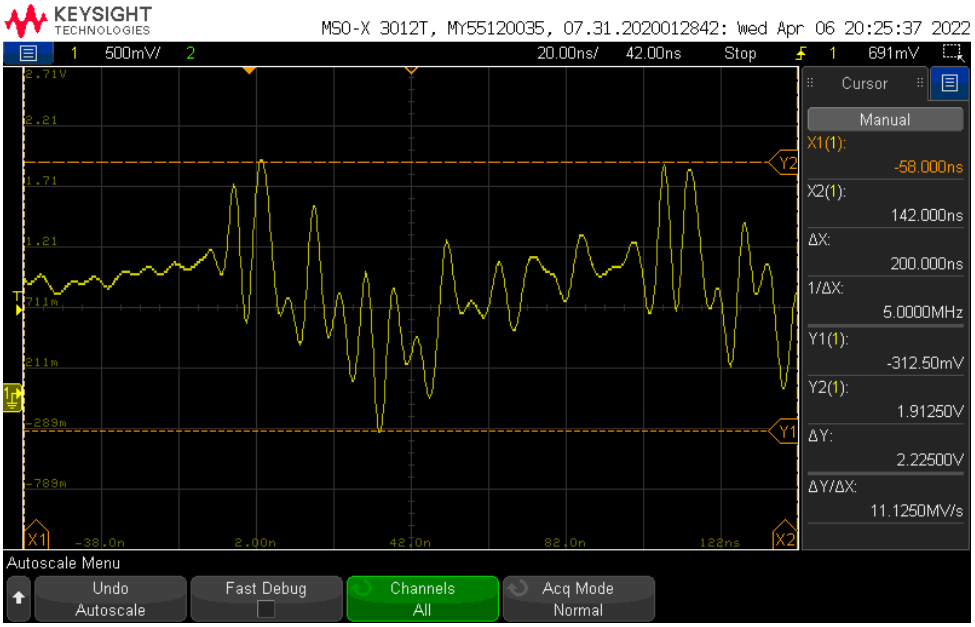
5 Resultat

Med Hall effect-sensoren spenningssatt, uten et nærværende magnetfelt, gir Hall effect-sensoren ut et analogt signal på 1 V, som vist i figur 3.



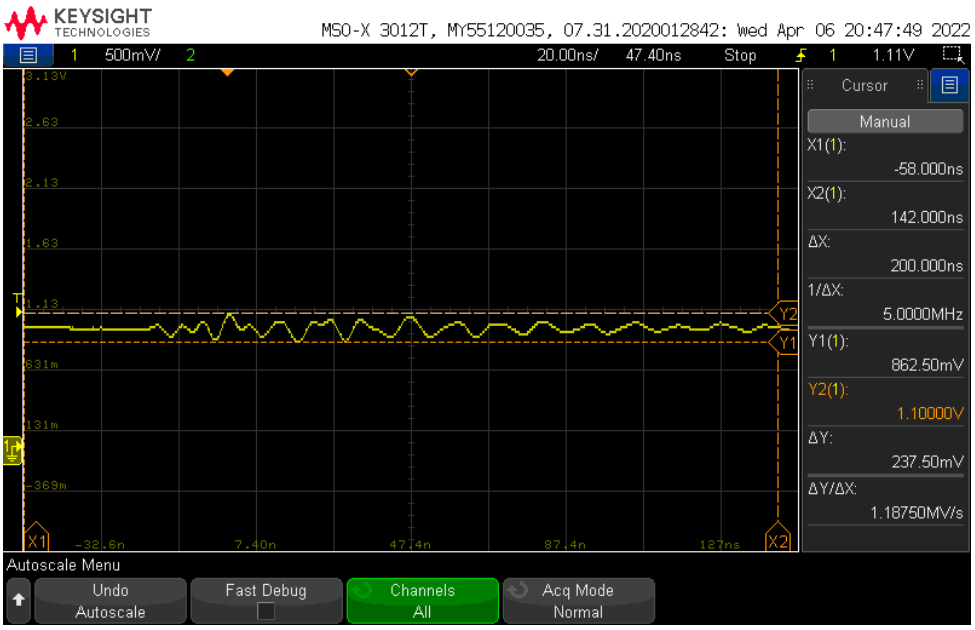
Figur 3: Skopbilde uten magnetisk felt.

Mens pumpa kjører, og er plassert 40 cm unna Hall effect-sensoren, måles det mye støy på det ufiltrerte signalet fra Hall effect-sensoren. Sensoren gir ut signaler mellom -0.3125 V og 1.9125 V, vist i figur 4.

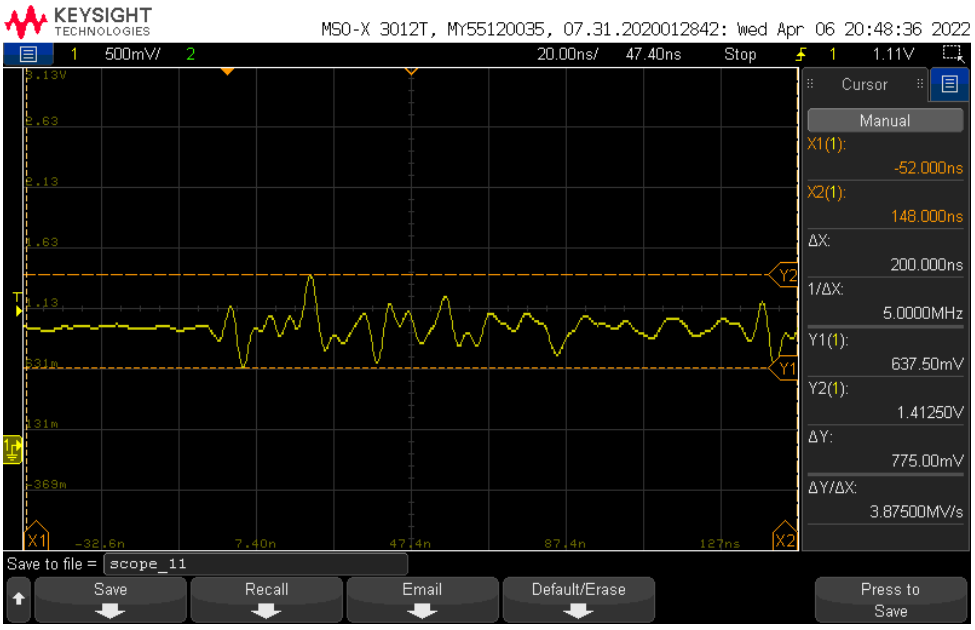


Figur 4: Skopbilde av ufiltrert signal.

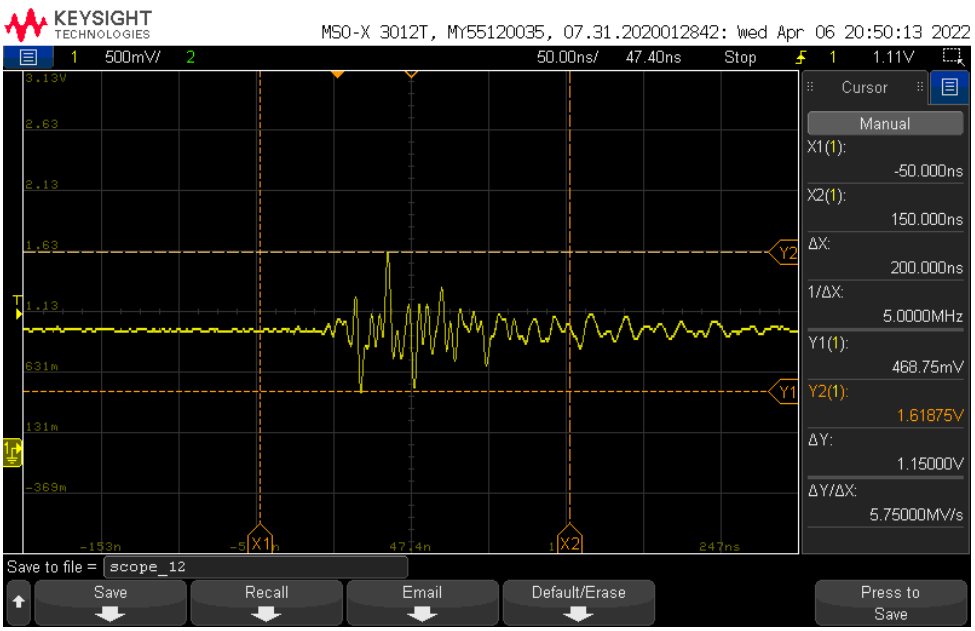
Det filtrerte signalet fra Hall effect-sensoren, når pumpa kjører, er vist i flere skopbilder i figur 5, figur 6 og figur 7. Her er det flere bilder fordi vi stoppet skopet på flere tidspunkt for å undersøke om signalet varierte mye over tid. RC-filteret som brukes, forklares nærmere i kapittel 5.3.1.



Figur 5: Skopbilde av det filtrerte signalet.



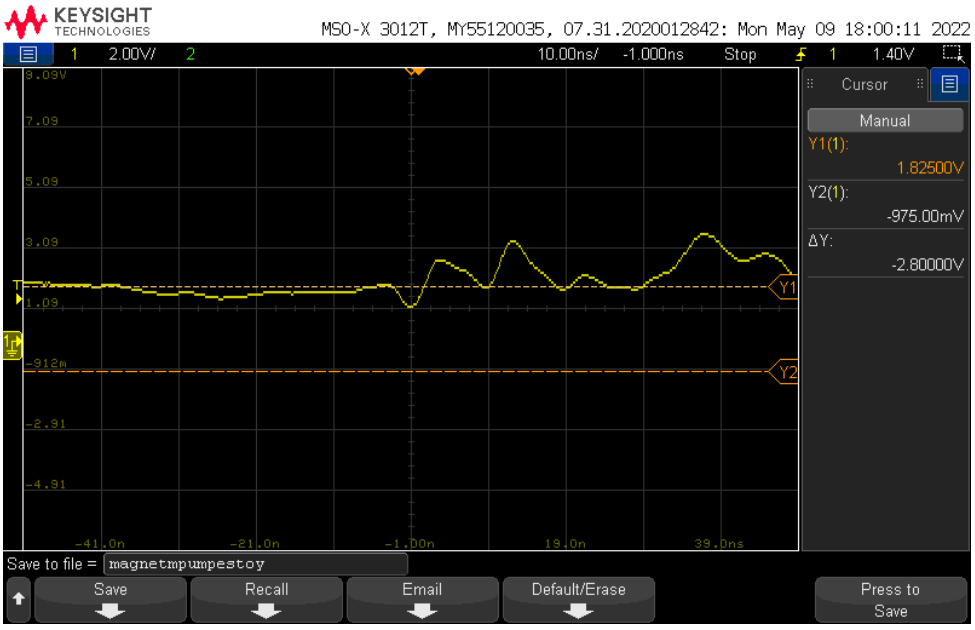
Figur 6: Skopbilde av det filtrerte signalet.



Figur 7: Skopbilde av det filtrerte signalet.

Kan se i de tre skopbildene at støyen varierer litt, med minste differanse fra topp til bunn på 238 mV, og største differanse på 1150 mV. Ved test av magneten i nærheten av sensoren, endret utgangsspenningen seg fra 1 V idet magneten kom på 4 cm = 40 mm avstand og nærmere. Sensoren ga ut to endeverdier ut ifra hvilken vei magneten ble holdt. De var på 1.85 V og -31 mV.

Et skopbilde fra da pumpa var på, og magneten var nær Hall effect-sensoren, er vist i figur 8.



Figur 8: Støy fra pumpe mens magneten plukkes opp av Hall effect-sensoren.

6 Analyse og konklusjon

Uten et nærværende magnetisk felt, gir Hall effect-sensoren ut et signal på 1 V. Dette stemmer godt overens med databladet [1], som beskriver at sensoren skal gi ut 1 V dersom 0 mT er oppdaget.

Når pumpa startes, observerer man, i figur 4, at signalet er støyende til tross for at sensoren er plassert 40 cm unna pumpa. Dette tilsvarer høyden til flyteren, som betyr at sensoren ikke kommer til å være plassert lenger unna enn dette. Det ufiltrerte signalet har en maksimalverdi på 1.91 V og en minimumsverdi på -312.5 mV. I databladet står det at maksimal- og minimumsverdi er 1.8 V og 0.2 V. Det støyende ufiltrerte signalet når altså maksimalverdiene til sensoren.

Ved måling av signalet som blir filtrert, gjennom RC-filteret på kretskortet *Plattformgrensesnitt*, observerer man en varierende grad av støy på signalet. Den største verdien som blir målt er på 1.6 V, i figur 7. Denne når ikke toppverdien for sensoren, og en slik spiker har man observert kommer sjeldent. Man kan se i de to andre skopbildene, figur 5 og figur 6, at de andre spikerne ikke har like høye verdier.

Veggen til flyteren er 7.4 mm tykk, og Hall effect-sensorens utgangssignal endret seg fra 1 V da magneten kom innenfor 40 mm. Man observerer videre at mens pumpa kjørte, i figur 8, er det fortsatt støy på signalet, fra pumpa. Støyen ser ut til å variere rundt en DC-komponent, som her er magneten i nærheten av Hall effect-sensoren.

Signalet til Hall effect-sensoren har som hensikt å fungere som en av/på bryter. Det vil si at nøyaktighet av signalet ikke er kritisk for at systemet skal fungere. Det er likevel ønskelig at signalet skal opptre stabilt nok til at det ikke gir falske av/på signaler. Ettersom støyen fra pumpa består av flere spikere (og ikke en jevn verdi), kan det konkluderes med at det, ved hjelp av programvare, kan settes krav om at mikrokontrolleren ikke godtar et magnetfelt dersom målingen ikke holder seg over en bestemt verdi i en bestemt tidsperiode. Hall effect-sensoren skal, i praksis, kun detektere et magnetfelt mens flyteren blir holdt av ROV-en, og pumpa skal bli skrudd på etter dette. Det vil si at pumpa og Hall effect-sensoren ikke skal være på samtidig. Det kan derfor konkluderes med at selv om støyen hadde vært vanskelig å filtrere ut helt, vil det spille liten rolle i det faktiske systemet. Det digitale filteret blir diskutert i kapittel 7.5.

Referanser

[1] T. Instruments, «DRV5053 Analog-Bipolar Hall Effect Sensor», https://www.ti.com/lit/ds/symlink/drv5053.pdf?HQS=dis-dk-null-digikeymode-dsf-pf-null-wwe&ts=1649172154027&ref_url=https%253A%252F%252Fwww.ti.com%252Fgeneral%252Fdocs%252Fsuppproductinfo.tsp%253FdistId%253D10%2526gotoUrl%253Dhttps%253A%252F%252Fwww.ti.com%252Flit%252Fgpn%252Fdrv5053, (lastet ned 05.04.2022), 2015.

B.7 Testrapport 7: Trykkammer

22D.7 - Intern

Prøverapport

Utvikling av smart flyter og sensor-system

UiS Subsea, test av trykksensor i trykkammer.

Forfatter(e)
Hanne Lovise Berger, Malin Harr Overland og Jørgen Hemnes Johannessen

23. april 2022

UTARBEIDET AV
Hanne Lovise Berger og Malin Harr Overland

GODKJENT AV
Jørgen Hemnes Johannessen

Postadresse:
www.uis.no

EMNEORD:
Trykksensor, trykkammer, måleområde

Prøverapport

Utvikling av smart flyter og sensorsystem

UiS Subsea, test av trykksensor i trykkammer.

RAPPORTNUMMER	VERSJON	DATO
22D.7	3.0	23. april 2022

FORFATTER(E)
Hanne Lovise Berger, Malin Harr Overland og Jørgen Hemnes Johannessen

OPPDRAGSGIVER(E)		OPPDRAGSGIVERS REFERANSE
Morten Tengesdal		Morten Tengesdal
PROSJEKT	**GRADERING**	**ANTALL SIDER OG VEDLEGG**
Utvikling av smart flyter og sensorsystem	Intern	18
PRØVEOBJEKT		**PRØVEOBJEKT MOTTATT**
Trykksensor		4. april 2022
PRØVEPROGRAM	**PRØVESTED**	**PRØVEDATO**
N/A	UiS - KE E-154	21. april 2022

SAMMENDRAG
Testrapporten innholder test av hele måleområdet til trykksensoren, utført i trykkammer på laboratorium, med god hjelp fra Magnus Wersland og Kim Andre Nesse Vorland som har hjulpet til med oppkobling og tilgang på utstyr og laboratorium.

Prøveresultatene gjelder kun de objekter som er prøvd.

Historikk

VERSJON	DATO	VERSJONSBESKRIVELSE
1.0	21-4-2022	Opprettet testdokument.
2.0	21-4-2022	Utførte test og skrev testrapport.
3.0	23-4-2022	Revidert beskrivelse av analyse og konklusjon.

Innhold

1 Hva skal testes?

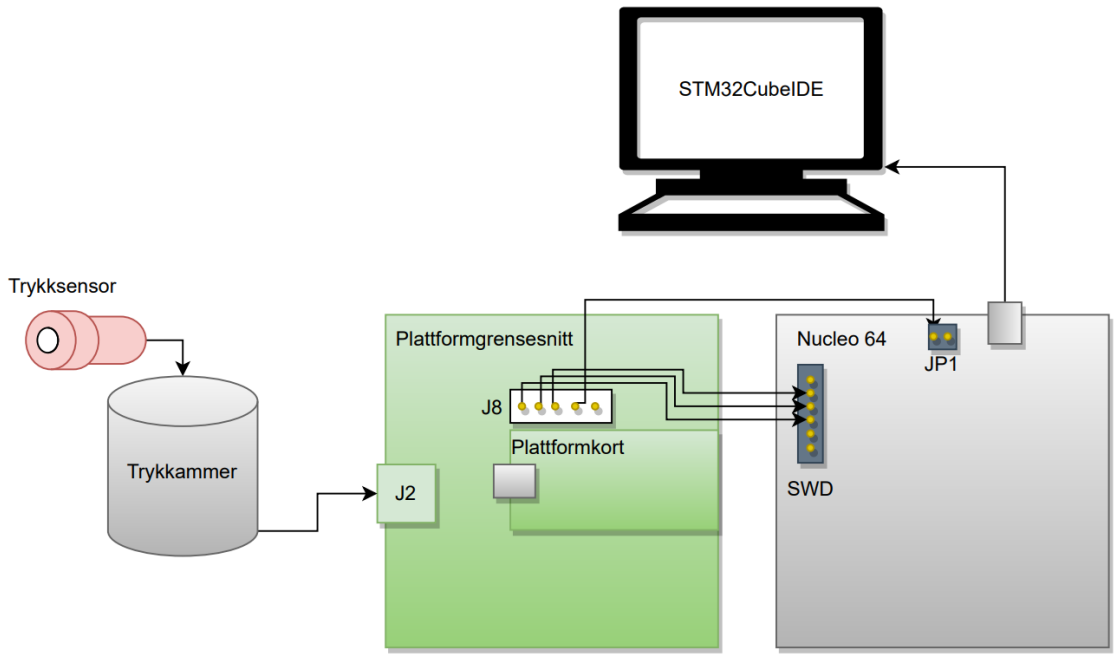
I denne testrapporten skal hele måleområdet til trykksensoren testes, det vil si fra 0 til 5 bar. Det målte trykket skal sammenlignes med det faktiske trykket for å redegjøre for avvik, og om en eventuell kalibrering må til. Måleområdet testes ved å øke trykket med 0.5 bar om gangen, der man tar tre målinger for hvert steg. Deretter utføres samme måling, men med å redusere trykket fra 5 til 0 bar.

2 Utstyr og innstillinger

- Kretskortet *Plattformgrensesnitt* designet til flyteren
- *STM32 Nucleo-64* programmeringskort
- MikroUSB- til USB-kabel
- Ferdig loddet og støpt trykksensor *MS580305BA01*
- Trykkammer *Triaxial cell*
- Trykkmåler *Digital Test Gauge XP*^{2i}

3 Oppkobling

Trykksensoren skal kobles opp som vist i figur 1, via trykkammeret og inn til PC.



Figur 1: Oppkobling av trykksensor via trykkammeret og til PC.

4 Programvareverktøy

- *STM32CubeIDE 1.9.0*
- *MATLAB R2019a*

5 Programfiler

Programkoden i *STM32CubeIDE* som blir brukt til å lese av trykksensoren under test er vist i kodeutsnitt 1 og 2, og er utarbeidet i samarbeid mellom begge bachelorgruppene, denne koden er nøyere beskrevet i kapittel 6.4.2.

```c
float TRYKKSENSOR_LesTrykk(TRYKKSENSOR *dev){

    trykk_status stat;

    uint8_t D2_buf[3];            // Digital temperatur value buffer
    uint8_t D1_buf[3];            // Digital pressure value buffer
    uint8_t trykkbuf[1];
    uint8_t tempbuf[1];
    float trykk_f;

    trykkbuf[0] = TRYKKSENSOR_D1_256;
    tempbuf[0] = TRYKKSENSOR_D2_256;

    float T2;
    float OFF2;
    float SENS2;

    dev->D2                 = 0; // Digital Temperature value
    dev->D1              = 0; // Digital Pressure value

    // Les Digital temperature value og legg det i D2buf

    stat = TRYKKSENSOR_SkrivRegister(dev, (uint8_t *)tempbuf);

    if(stat != TRYKK_SUKSESS){
        trykkfeil += 1;
    }
    HAL_Delay(10);

    stat = TRYKKSENSOR_LesRegistrene( dev, TRYKKSENSOR_ADC_READ, (uint8_t *)D2_buf, 3 );

    if(stat != TRYKK_SUKSESS){
        trykkfeil += 1;
    }

    // Les Digital pressure value og legg det i D1buf
    stat = TRYKKSENSOR_SkrivRegister(dev, (uint8_t *)trykkbuf);

    if(stat != TRYKK_SUKSESS){
        trykkfeil += 1;
    }
    HAL_Delay(10);

    stat = TRYKKSENSOR_LesRegistrene( dev, TRYKKSENSOR_ADC_READ, (uint8_t *)D1_buf, 3 );

    if(stat != TRYKK_SUKSESS){
        trykkfeil += 1;
    }
```

Kode 1: Kode for å lese av trykksensor.

```
    // Samle leste verdier til 32-bits variabler.
    dev->D2 = ((D2_buf[0]<<16)|(D2_buf[1]<<8)|D2_buf[2]);
    dev->D1 = (D1_buf[0]<<16|D1_buf[1]<<8|D1_buf[2]);

    // Regn ut temperaturen
    dev->dT = dev->D2 - (dev->TREF *pow(2,8));                  // Formel fra datablad.
    dev->TEMPt = 2000 + (dev->dT*dev->TEMPSENS/pow(2,23));      // Formel fra datablad.
    dev->temp_f = dev->TEMPt/100.0;                              //

    // Regn ut temperaturkompensert trykk
    dev->OFF = (dev->OFF_T1*pow(2,18)) + ((dev->TCO*dev->dT)/pow(2,5));// Formel fra datablad
    dev->SENS = (dev->SENS_T1*pow(2,17)) + (dev->TCS*dev->dT)/pow(2,7);// Formel fra datablad.

    // SECOND ORDER TEMPERATURE COMPENSATION FLOWCHART
    if(dev->TEMPt<2000){
        T2 = (float)(3*pow(dev->dT,2) / pow(2,33));
        OFF2 = (float)(3* pow((dev->TEMPt-2000),2) / pow(2,3));
        SENS2 = (float)(7*pow((dev->TEMPt-2000),2) / pow(2,3));

        if(dev->TEMPt<(-1500)){
            SENS2 = (float)(SENS2 + 3*pow((dev->TEMPt+1500),2));
        }

    }
    else {
        T2 = 0;
        OFF2 = 0;
        SENS2 = 0;
    }

    dev->TEMPt = dev->TEMPt - T2;
    dev->OFF = dev->OFF-OFF2;
    dev->SENS = dev->SENS-SENS2;

    // Formel fra datablad.
    dev->trykk_comp = (int32_t)(((dev->D1*dev->SENS/pow(2,21)) - dev->OFF)/pow(2,15));
    trykk_f = (float)(dev->trykk_comp/100);

    if(trykkfeil > 0) return TRYKK_FEIL;

    return trykk_f;
}
```

Kode 2: Fortsettelse på kodeutsnitt 1.

Til å regne om det målte trykket og det faktiske trykket til samme enhet i *MATLAB*, brukes koden vist i kodeutsnitt 3.

```
% Oppover
faktisk_trykk = 10^3*[0.0 0.0 0.0 0.525 0.525 0.525 1.06 1.06 1.06 1.52 1.52 1.52 ...
2.03 2.03 2.03 2.52 2.52 2.52 3.01 3.01 3.01 3.51 3.51 3.51 4.01 4.00 3.99 4.51 ...
4.52 4.53 4.99 5.01 ];
maalt_trykk = 1/100*[98695, 98664, 98664, 151101, 151056, 151002, 204303, 204249, ...
204249, 251168, 250725, 250652, 301572, 301524, 301334, 350849, 351080, 350984, ...
400071, 400590, 400578, 450646, 450772, 450626, 501234, 500718, 499523, 551162,...
552527, 553469 599745, 601002]-1023;
antall = 1:length(maalt_trykk);

differanse = maalt_trykk - faktisk_trykk;
gjennomsnitt = mean(differanse);
minst = min(differanse);
storst = max(differanse);

% Nedover
faktisk_trykk_ned = 10^3*[4.51 4.51 4.51 4.01 4.01 4.01 3.51 3.51 3.51 3.01 3.01 ...
3.01 2.5 2.5 2.5 2.0 2.0 2.0 1.49 1.49 1.49 1.0 1.0 1.0 0.5 0.5 0.5 0.0 0.0 0.0];
maalt_trykk_ned = 1/100*[558651, 558567, 558470, 507383, 507725, 507723, 456545, ...
456852, 456818, 405542, 406009, 406022, 354091, 354475, 354558, 303169, 303561,...
303605, 251715, 251915, 251979, 202116, 202198, 202259, 151722, 151853, 152120,...
101511, 101565, 101599]-1023;
antall2 = 1:length(maalt_trykk_ned);

differanse2 = maalt_trykk_ned - faktisk_trykk_ned;
gjennomsnitt2 = mean(differanse2);
minst2 = min(differanse2);
storst2 = max(differanse2);
u2 = storst - minst;
```

Kode 3: MATLAB-koden for å regne ut målt trykk og faktisk trykk i samme enhet.

Videre brukes koden vist i kode 4, til å lage plott av resultatverdiene.

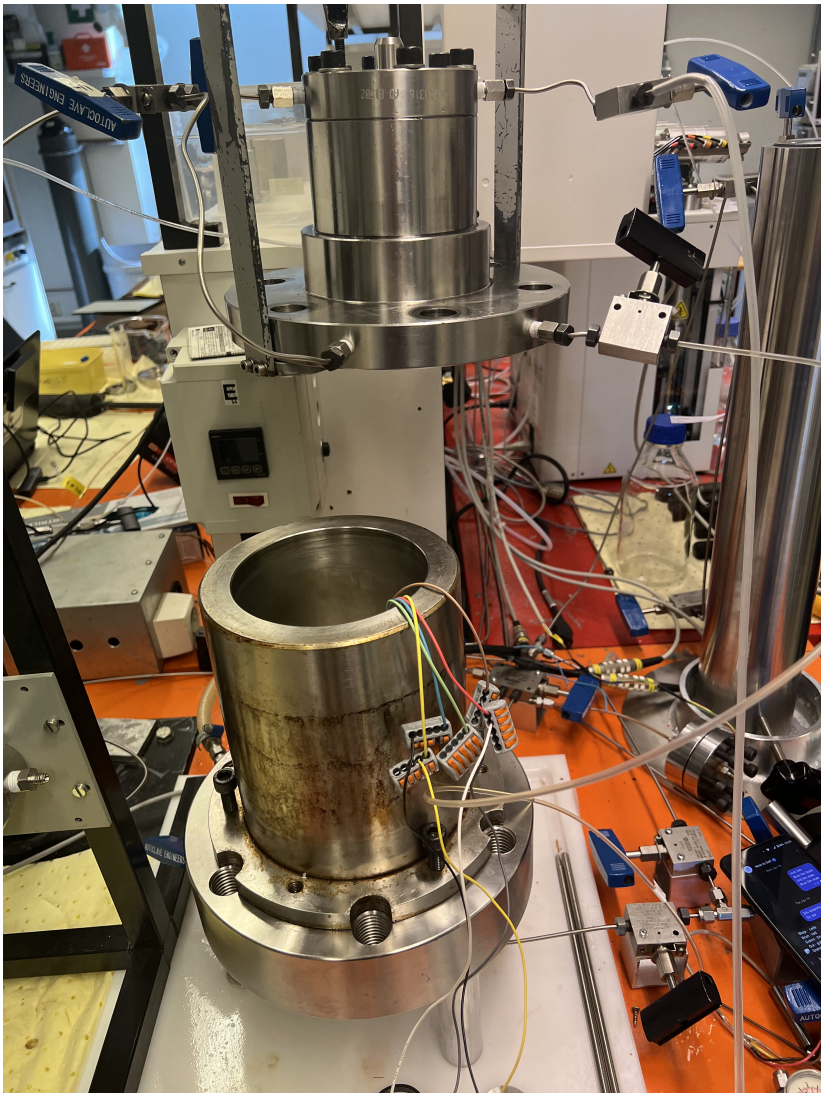
```
% Figurer
figure(1)
plot(antall, faktisk_trykk);
hold on
plot(antall, maalt_trykk)
hold off
legend('Faktisk trykk', 'Målt trykk')
grid on
xlabel('Måling nr')
ylabel('mbar')
title('Økende trykk')
tekst = {"Gjennomsnittlig differanse: " + gjennomsnitt, "Minste differanse: " ...
+ storst, "Største differanse: " + minst};
text(25,0,tekst, 'FontSize', 12)

figure(2)
plot(antall2, faktisk_trykk_ned);
hold on
plot(antall2, maalt_trykk_ned)
hold off
legend('Faktisk trykk', 'Målt trykk')
grid on
xlabel('Måling nr')
ylabel('mbar')
title('Synkende trykk')
tekst2 = {"Gjennomsnittlig differanse: " + gjennomsnitt2, "Minste differanse: "...
+ minst2, "Største differanse: " + storst2};
text(5,0,tekst2, 'FontSize', 12)
```

Kode 4: MATLAB-koden for å lage plott av trykkmålingene.

6 Fremgangsmåte

Kobler, med hjelp fra laboratorieansatte, trykksensoren til *Wago*-klemmene som allerede er montert i trykkammeret, vist i figur 2. Kobler deretter lederne, som går videre ut av trykkammeret, til kretskortet *Plattformgrensesnitt*.

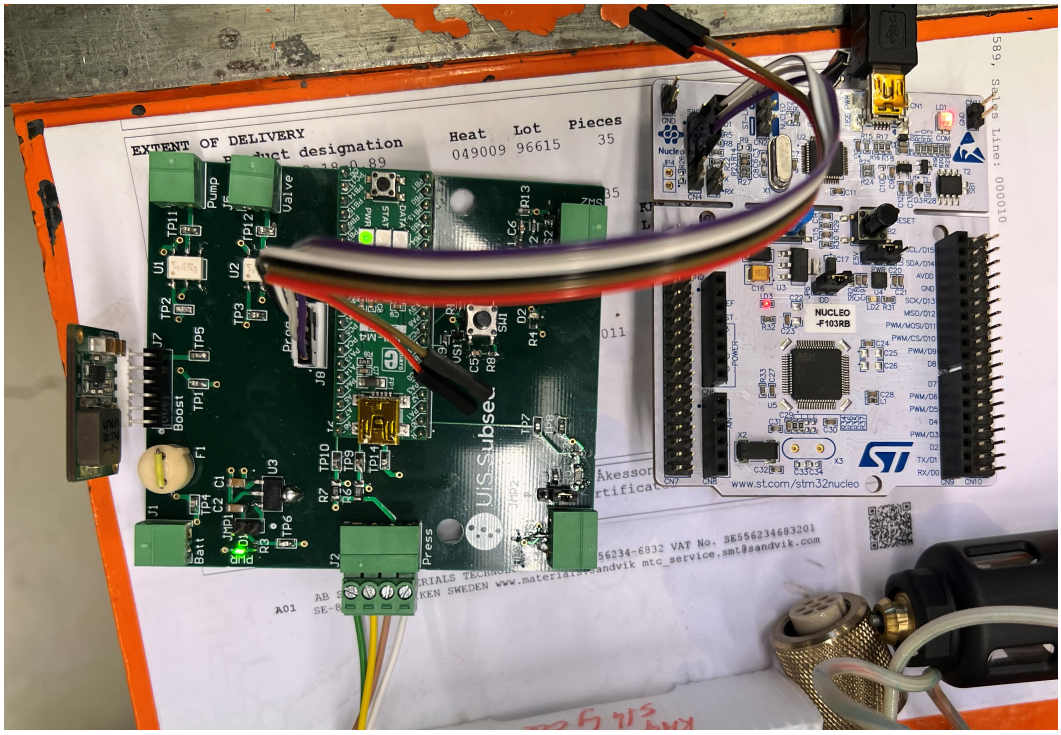


Figur 2: Åpent trykkammer med *Wago*-klemmer som trykksensoren skal kobles til.

Legger trykksensoren inni trykkammeret, hvorav dette blir lukket av Kim Andre Nesse Vorland. Trykkmåleren skal vise 0 bar ved start, som vist i figur 3.

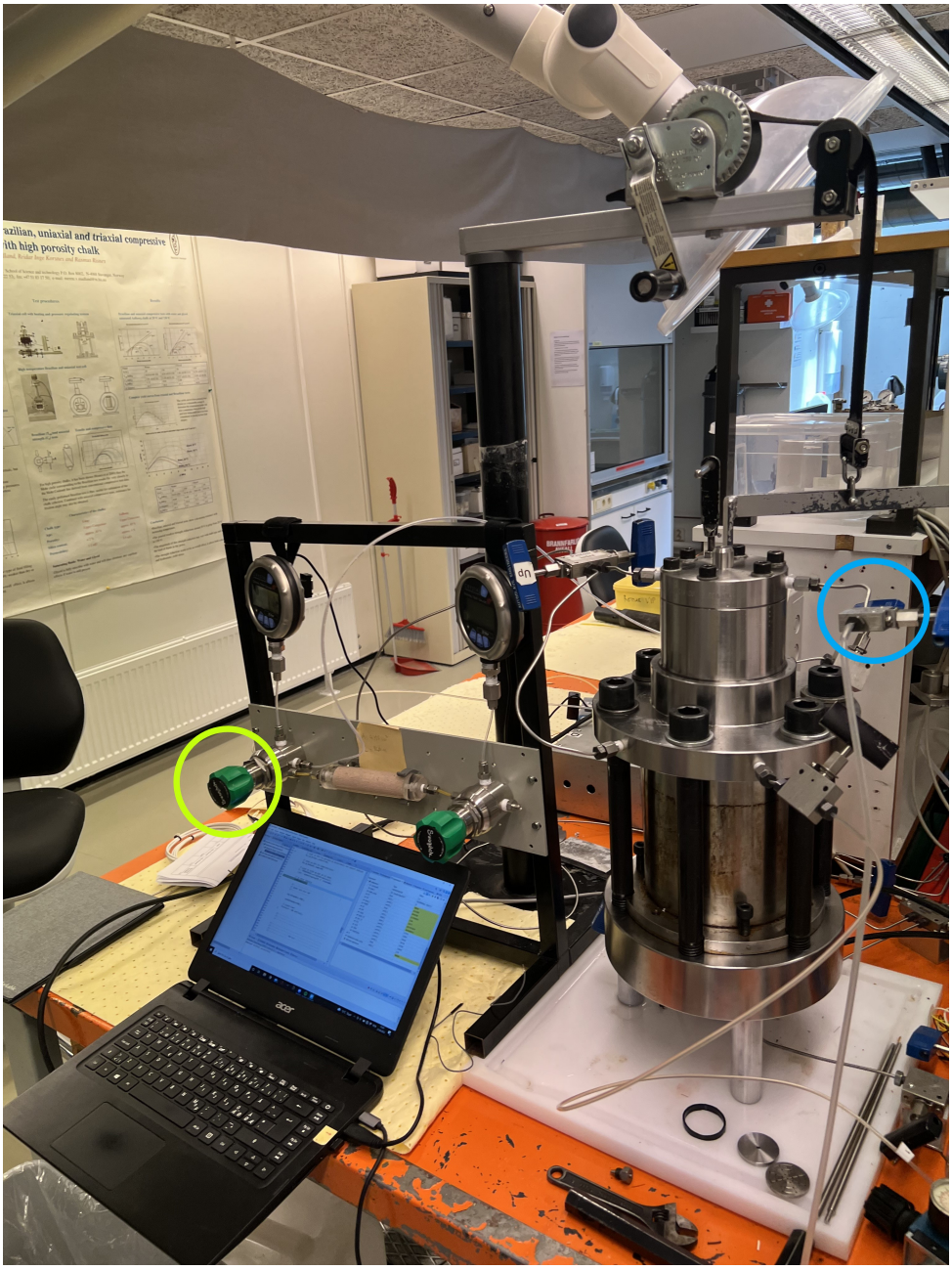
Figur 3: Trykkmåler viser 0.002 bar.

Med PC koblet til programmeringskortet, som videre er koblet til *Plattformgrensesnitt*, vist i figur 4, åpnes *STM32CubeIDE*. Kjører deretter programmet i debug-mode, med kun ett breakpoint på koden, der trykksensoren blir avlest.



Figur 4: Programmeringskort koblet til PC og *Plattformgrensesnitt*.

Stepper gjennom koden tre ganger, og noterer verdien til variabelen `trykk_comp` alle tre gangene. Justerer deretter sakte på regulatoren under trykkmåleren, som er markert med grønn ring i figur 5, for å øke trykket inne i trykkammeret. Regulatoren har en dødtid, så denne må justeres sakte. Stopper første gang når trykkmåleren viser 0.5 bar, og noterer deretter tre verdier, på samme måte som ved 0 bar.



Figur 5: Grønt håndtak til venstre justerer trykket inne i trykkammeret.

Deretter økes trykket i trykkammeret med 0.5 bar per justering, helt til trykkammeret har nådd 5.0 bar. Variabelen `trykk_comp` noteres tre ganger som før. Når trykkmåleren viser 5.0 bar, og verdiene er notert, gjennomføres samme metode, men fra 5.0 bar, og ned til 0 bar. For å redusere trykket i trykkammeret, åpnes en ventil på trykkammeret som slipper ut gass. Ventilen er markert med blå ring i figur 5.

Til slutt brukes *MATLAB* til å lage plott av målt trykk mot påsatt trykk. Dette er vist i resultatkapittelet.

7 Resultat

Verdiene som ble notert ved hvert trykk, er vist i tabell 1, der man regulerte opp trykket fra 0 til 5 bar, og i tabell 2, der trykket ble sluppet ut igjen fra 4.5 til 0 bar. I kolonnen med påsatt trykk, er det notert flere verdier dersom det, under test, ble observert at trykkmåleren viste ulike verdier mens målt trykk ble notert. Dette ble gjort for å kunne sammenligne faktisk trykk med målt trykk så nøyaktig som mulig.

Påsatt trykk [Bar]	Noterte verdier `trykk_comp`	Notert temperatur [DegC]
0.00	98695, 98664, 98664	-
0.53	151105, 151056, 151002	22
1.06	204303, 204249, 204249	22
1.52	251168, 250725, 250652	22
2.03	301572, 301524, 301334	22.1
2.52	350849, 351080, 350984	22.2
3.01	400071, 400590, 400578	22.2
3.51	450646, 450772, 450626	22.3
4.01, 4.00, 3.99	501234, 500718, 499523	22.4
4.51, 4.52, 4.53	551162, 552527, 553469	22.3
4.99, 5.01, 5.02	599745, 601002, 609342	22.1

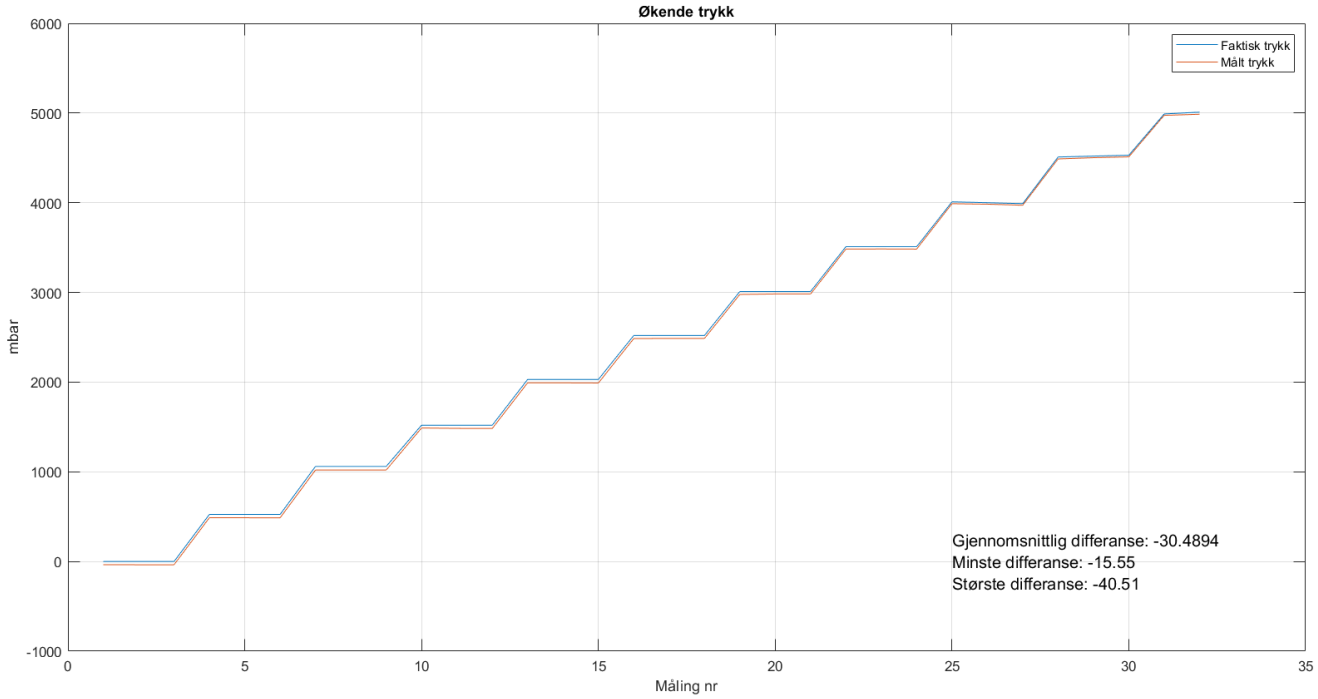
Tabell 1: Målt trykk ved hvert steg på 500 mbar, fra 0 bar til 5 bar.

Påsatt trykk [Bar]	Noterte verdier `trykk_comp`	Notert temperatur [DegC]
4.51	558651, 558567, 558470	21
4.01	507383, 507725, 507723	21
3.51	456545, 456852, 456818	21
3.01	405542, 406009, 406022	20
2.50	354091, 354475, 354558	20
2.00	303169, 303561, 303605	20
1.49	251715, 251915, 251979	20
1.00	202116, 202198, 202259	20
0.50	151722, 151853, 152120	20
0.00	101511, 101565, 101599	20

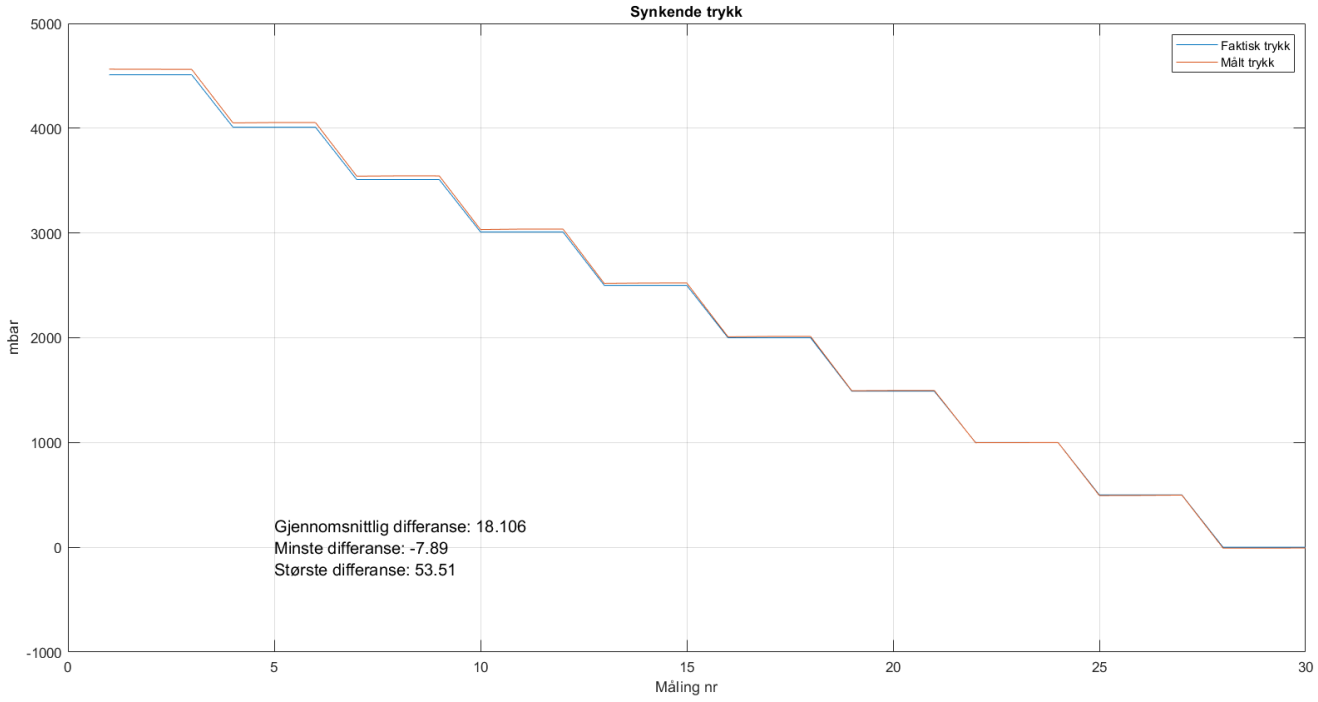
Tabell 2: Målt trykk ved hvert steg på 500 mbar, fra 5 bar til 0 bar.

Ettersom trykksensoren måler absolutt trykk, måler den atmosfæretrykket inne i trykkammeret, når trykkmåleren og regulatoren er satt til 0 bar. Derfor må atmosfæretrykket trekkes fra de målte verdiene for å sammenligne målt trykk med faktisk trykk. Bruker nettsiden [1] til å finne dagens trykk. Nettsiden ble besøkt 21.04.2022, og man ser på trykket for tidspunktet 13:20-14:20, som er på 1023 bar.

Plottet laget i *MATLAB* er vist i figur 6 med verdiene fra tabell 1, og i figur 7 med verdiene fra tabell 2. Avviket mellom målt trykk og faktisk trykk, varierer mellom 7 og 53 mbar. Gjennomsnittlig avvik ligger på -30.489 mbar ved økende trykk, og 18.106 mbar ved minkende trykk.



Figur 6: Graf over det målte trykket når trykket stiger.



Figur 7: Graf over det målte trykket når trykket synker.

8 Analyse og konklusjon

Som man observerer i plottene, laget i *MATLAB*, er målt trykk og faktisk trykk veldig like gjennom hele måleområdet. Ved regulering av trykket med trykkmåleren, som leste av faktisk trykk, er det ikke usannsynlig at denne trykkmåleren har en unøyaktighet i målingene. De laboratorieansatte nevnte at den ikke hadde vært til kalibrering på et par år. Trykkmåleren vippet ofte mellom to-tre verdier på det tredje desimaltallet. Det vil si at usikkerheten ligger i det tredje desimalet, som blir på opp til 10 mbar. Avviket som er målt er større enn dette.

Gjennomsnittlig avvik når trykket økes, ligger på -30.489 mbar, men når trykket reduseres har man et gjennomsnittlig avvik på 18.106 mbar. Fra databladet til trykksensoren, side fire [2], finnes tabellen som er gjengitt i figur 8.

PRESSURE OUTPUT CHARACTERISTICS (V_{DD} = 3 V, T = 25°C UNLESS OTHERWISE NOTED)

Parameter	Conditions	Min.	Typ.	Max	Unit
Operating Pressure Range	P_{range} Full Accuracy	0		5	bar
Absolute Accuracy, autozero at one pressure point 300…1100 mbar (1)	at 20°C, 300..1100 mbar at 0..50°C, 300..1100 mbar at -40..85°C, 300..1100 mbar	-1.5 -4.0 -15.0		+1.5 +4.0 +15.0	mbar
Absolute Accuracy, autozero at one pressure point 0…5000 mbar (1)	at 20°C, 0..5000 mbar at 0..50°C, 0..5000 mbar at -40..85°C, 0..5000 mbar	-80 -100 -120		+80 +100 +120	mbar
Maximum error with supply voltage (3)	V_{DD} = 1.8 V … 3.6 V		+/-5		mbar
Long-term stability (2)			+/-1		mbar/yr
Resolution RMS	OSR 4096 2048 1024 512 256		0.036 0.054 0.081 0.126 0.195		mbar

Figur 8: Tabell hentet fra [2].

Her kan man se at ved trykk fra 0 til 5000 mbar, og med en temperatur på 20°, har trykksensoren en nøyaktighet på ± 80 mbar. Det kan dermed konkluderes at gjennomsnittlige avvik på -30.489 mbar og 18.106 mbar, er innenfor maksimal feilmargin. Nøyaktigheten til sensoren sier noe om produksjonsvariasjoner, og hvor stor den tilfeldige feilen kan være. Dette vil bli nøyere diskutert i resultatkapittelet i hver vår sluttrapport.

Dersom trykksensoren skal brukes til å måle dybde i vann, er det aktuelt å regne på hvor mye dette avviket tilsvarer i centimeter. I formel 1 regnes det ut hvor mye 80 mbar tilsvarer i meter. Dette er den maksimale målefeilen som er gitt i databladet. I likning 2 regnes det ut hvor mye det gjennomsnittlige avviket, på omtrent 30 mbar, tilsvarer i meter. Her er ΔP avviket i Pascal, ρ er tettheten i vannet, og g er jordas gravitasjonskraft.

$$\begin{aligned} d &= \frac{P - P0}{\rho g} \\ &= \frac{\Delta P}{\rho g} \\ &= \frac{0.08 \cdot 100000\ Pa}{1000 \cdot 9.81} \\ &= 0.815\ m \Rightarrow 80.5\ cm \end{aligned} \tag{1}$$

$$\begin{aligned} d &= \frac{P - P0}{\rho g} \\ &= \frac{\Delta P}{\rho g} \\ &= \frac{0.03 \cdot 100000\ Pa}{1000 \cdot 9.81} \\ &= 0.306\ m \Rightarrow 30.6\ cm \end{aligned} \tag{2}$$

Trykksensoren har altså en maksimal unøyaktighet på ± 80.5 cm, ved trykk varierende over hele måleområdet (0 - 5000 mbar), i følge databladet. Det er derimot målt en gjennomsnittlig unøyaktighet på ± 30.6 cm ved trykk varierende over hele måleområdet. En videre vurdering av resultatet blir tatt i hver vår sluttrapport for å kunne relatere til faktisk bruk av trykksensoren.

Referanser

[1] Timeanddate, https://www.timeanddate.no/vaer/norge/stavanger/siste-uke, (Besøkt 21.04.2022, Tidspunkt: 13:20 - 14:20).

[2] T. Connectivity, «MS5803-05BA», https://www.te.com/commerce/DocumentDelivery/DDEController?Action=srchrtrv&DocNm=MS5803-05BA&DocType=Data+Sheet&DocLang=English, (Lastet ned 23.04.2022), 2017.

C Komponentlister

C.1 Komponentliste kretskort

Item No	Qty	Manufacturer prod. nr.	Beskrivelse/verdi	Part Reference	DIST. No. 2022	Footprint
1	1	MEZD41502A-B	Boostregulator	U4	Digikey: 1589-1493-ND	
2	2	TLP3107A	SSR - solid state rele	U1, U2	Digikey: 264-TLP3107A(TPFCT-ND	6-SOP
3	1	DRV5053CAQLPG	Hall-effect sensor	SENS1	Digikey: 296-38526-ND	TO-92
4	1	MS580305BA01-00	Trykksensor	SENS2	RS Components: 893-7219	
5	1	MC3327	Buckregulator	U3	RC: MC33275ST-3.3T3G	SOT-223
6	2		LED Diode	D1, D2		0603
7	4	691322110002	2 pin kontakt han	J1, J3, J5, J12	Digikey: 732-2769-ND	
8	4	691361100002	2 pin kontakt hunn		Digikey: 732-2751-ND	
9	1	691322110003	3 pin kontakt han	J4	Digikey: 732-2770-ND	
10	1	691361100003	3 pin kontakt hunn		Digikey: 732-2752-ND	
11	1	691322110004	4 pin kontakt han	J2	Digikey: 732-2771-ND	
12	1	691361100004	4 pin kontakt hunn		Digikey: 732-2753-ND	
13		RCUCTE	Testpunkt	TP1 - TP14	Digikey: RCUCTE	0603
14	1		2 pin header	JMP1		
15	1	PREC003SAAN-RC	3 pin header	JMP2	Digikey: S1012EC-03-ND	
16	1	640456-5	5 pin kontakt han	J8	Digikey: A19471-ND	
17	1	1375820-4	4 pin kontakt hunn		Digikey: A99615-ND	
18	1	535541-4	6 pin kontakt, fra boostregulator	J7	Digikey: 535541-4	
19	2	PH1-20-UA	20 pin connector mikrokontroller	J6	Digikey: 2057-PH1-20-UA-ND	
20	4		120 ohm	R1, R2, R3, R4		0603
21	3	AF1206FR-0710KL	10 kohm	R5, R6, R7	Digikey: 13-AF1206FR-0710KLCT-ND	1206
22	1	C0603X102K5RAC3316	0.01 uF	C4	Digikey: 399-C0603X102K5RAC3316CT-ND	0603
23	1	C0603X752J5JAC7867	7500 pF	C3	Digikey: 399-13414-1-ND	0603
24	1	C1206C334K1RECAUTO	0.33 uF	C2	Digikey: 399-C1206C334K1RECAUTOCT-ND	1206
25	1	C1206C105K5RAC7800	1 uF	C1	Digikey: 399-C1206C105K5RAC7800CT-ND	1206
26	1	56000001009	Fuseholder	F1	Digikey: WK6235-ND	
27	1	MRT 6.3AMMO	6.3A sikring	F1	Digikey: 507-1119-1-ND	
28	1	430182070816	Trykknapp	SW1	Digikey: 732-7006-1-ND	
29	2	CGA3E3X8R1H104K080AE	0.1 uF	C5, C6	Digikey: 445-173697-1-ND	0603
30	2	AC0603JR-07100RL	100 ohm	R8, R11	Digikey: YAG3636CT-ND	0603
31	2	AC0603JR-07330RL	330 ohm	R9, R12	Digikey: YAG3664CT-ND	0603
32	2	AC0603FR-07220KL	220 kohm	R10, R13	Digikey: YAG3579CT-ND	0603
33	2	D1213A-02WL-7	Overspenningsvern	TVS1, TVS2	Digikey: D1213A-02WL-7DICT-ND	SOT-323
34	1	8601.2001.08	in-line fuseholder		Digikey: 486-3836-ND	
35	1	5SF 5-R	Glassikring 5A		Digikey: 507-1232-ND	
36	1	T101LCFZBE	Batteribryter	S1	Digikey: CKN11362-ND	
37	1	NP8S2R202GE	Trykkknapp til GPIO in	SW2	Digikiey: CKN10335-ND	
38	1		Hylse til trykksensor		Blue Robotics: PENETRATOR-VP	

C.2 Komponentliste

Item No	Qty	Manufacturer prod. nr.	Beskrivelse/verdi	Part Reference	DIST. No. 2022
1	8	BATT ALKALINE D	D-type alkalisk batteri	BT1	D-MN1300
2	2	BH4DW	Batteriholder D type 4 celler		Digikey: BH4DW-ND
3	1	MIKROE-1367	Mikrokontrollerkort		RS: 820-9870
4	1	Vakuumpumpe	Pumpe		SM-00572
5	1	VDW250-6G-1-M5-Q	Ventil		RS: 838-8660

Figur 113: Komponentliste.

D Statusrapporter

Statusrapport januar - UiS Subsea Flyter

Frem til nå har det gått mye tid på planlegging og møter for å kunne ta diverse valg. Vi har nå bestemt oss for hvilken mikrokontroller og trykksensor vi ønsker å bruke. Etter møte med resten av Subsea, har vi kommet frem til at flyteren skal bli sluppet ut av ROV-en, og dermed trengs det en deteksjon av at flyteren er sluppet. Vi ble enige om å satse på muligheten for at ROV-en drar flyteren ved hjelp av elektromagnetisme, og flyteren skal dermed ha en magnetstripe montert på, samt en “hall effect-sensor”. Denne sensoren og tilhørende sensorkrets skal inngå i kretskortdesignet vårt.

Det har i tillegg kommet nye begrensninger med tanke på batteriforsyningen i konkurransen. Maksimal spenning er på 12V, og 6A. Vi har deretter funnet forslag til både pumpe, solenoidventil, og diverse komponenter som vi kunne brukt.

Vi har også kommet greit i gang med skrivingen av oppgaven. Vi er ferdig med systemdesignkapittelet med unntak av systemdesign-delen. Vi har fått en bedre oversikt over strukturen i rapporten og tenker å begynne å skrive om valg av sensor denne uken. Batterivurdering/analyse av strømtrekk skal også begynnes på denne eller neste uke.

Planen fremover er å designe et kretskort som inneholder både transistorkrets for styring av pumpe og solenoidventil, boost-regulatorer for å holde stabil spenning etterhvert som spenningen over batteriene synker ved bruk, samt den analoge sensorkretsen for å detektere det magnetiske feltet.

Vi har også som mål å finne aktuell pumpe og ventil i løpet av uken. Har begynt å sette oss inn i hvordan man kan lage en dynamisk modell for flyteren.

Vi ligger litt bak det som var planlagt i aktivitetsplanen, da vi hadde beregnet mindre tid til planleggingsdelen enn det vi har brukt til nå. Usikkerheter rundt funksjonaliteten til flyteren har nå løsnet seg opp etter flere møter med ulike grupper i UiS Subsea.

Hanne Lovise Berger og Malin Harr Overland

Statusrapport februar - UiS Subsea Flyter

Vi har nå bestemt oss for alle delene vi skal bruke i systemet og alt er bestilt inn. Både ventil, pumpe, mikrokontroller og trykksensor har kommet. Vi har loddet og begynt støping av trykksensoren i samarbeid med sensor- og reguleringsgruppa.

PCB-design er godt igang. Skisse av skjemategning er så og si ferdig (mangler kun eventuelle overspenningsvern) og også i altium er det kun småplukk igjen på skjemategningen. Legger ved foreløpig skjemategning i mailen. Målet for denne uken er å gjøre ferdig utlegget og bestille opp PCB. Alt av komponenter er bestilt.

Når PCB er bestilt er planen å skrive essay, parallelt med skriving i rapporten. I februar har vi skrevet en del på kapittelet om maskinvare, men dette må finskrives. Vi har også skrevet på kapittelet om fysiske egenskaper, som omhandler den dynamiske modellen.

Hanne Lovise Berger og Malin Harr Overland

Statusrapport mars - UiS Subsea Flyter

I mars har vi fått bestilt opp og loddet ferdig kretskortet. Vi har testet kraftforsyningen og solid state-releene og skrevet testrapport på begge disse. Har lagt ved disse som vedlegg i mailen. Vi koblet også opp med pumpe, ventil og ballong og så at oppsettet fungerte, utenom litt lekkasje i slangene som maskingruppa skal fikse opp i.

Vi har også skrevet en del på rapporten. Kapittel to og tre er ferdigskrevet. I kapittel fire mangler kun slutten av batterivurderingen. Vi har utført en batteriutladningstest, men ikke rukket å skrevet det ferdig enda, så det er det som mangler i kapittel fire. Rapporten er også lagt ved som vedlegg i mailen.

Vi har også gjennomført og fått godkjent Fluid Power-quizen som var påkrevd fra MateROV-konkurransen.

Hanne Lovise Berger og Malin Harr Overland

Statusrapport april - UiS Subsea Flyter

I april har resten av kretskortet og komponentene blitt testet. Dette innebærer en utladningstest av batteriene, trykksensoren og hall effect-sensoren med tilhørende kretser, og trykksensoren er i tillegg testet i et trykkammer. Testrapportene til disse testene er også skrevet ferdig.

Hele systemet er også satt sammen og testet i bassenget. Elektronikken fungerte som den skulle og flyteren klarte å ta to profiler. Siden da har vi justert ballasten sammen med maskinstudentene, samt montert flyteren på ny med forbedret feste.

Rapporten begynner også å nærme seg slutten. Kapittel 5 og 6 mangler bare rettskriving og vi har begynt på både resultat, diskusjon og konklusjon samt sammendrag og forord kapitlene.

Framover blir det derfor mye skriving. Etter eksamen er overstått gjenstår mer vanntesting, der man sammenligner med dynamisk modell, samt tester hall effect-sensoren i det sammensatte systemet. Til slutt skal rapporten ferdigstilles og rettes.

Hanne Lovise Berger og Malin Harr Overland