BACHELOROPPGAVE:

BO21E-15 AUTOMATISERT MIKROFLUIDITETS-PERFUSJONSSYSTEM

Robert Løland
Trygve Sognnæs
Andres David Rodriguez Delgado

01. juni. 2021

Dokumentkontroll

Rapportens tittel: BO21E-15 AUTOMATISERT MIKROFLUIDITETS-PERFUSJONSSYSTEM	*Dato* 01. juni. 2021
	Rapportnummer: B021E-15
Forfatter(e): Robert Løland Trygve Sognnæs Andres David Rodriguez Delgado	*Studieretning:* HEAU18
	Antall sider m/vedlegg 49
Høgskolens veileder: Eivind Vågslid Skjæveland	*Gradering:* Åpen
Eventuelle Merknader: Vi tillater at oppgaven kan publiseres.	

Oppdragsgiver: Sars-senteret	*Oppdragsgivers referanse:* Jørgen Høyer
Oppdragsgivers kontaktperson(er) (inklusiv kontaktinformasjon): *Jørgen Høyer, Telefon: 41636030, Email: jorgen.hoyer@uib.no*	

Forord

Denne rapporten dokumenterer vårt bachelorprosjekt på Automasjonsstudiet på Høgskolen på Vestlandet.

Dette prosjektet har vært lærerikt for oss og har gitt oss gode erfaringer i nye og kjente fagfelt. Vi har blitt testet i programmering, design av krets, 3D-printing, generell problemløsning og samarbeid. Gjennom semesteret har vi også opparbeidet oss mye erfaring med hvordan man planlegger og gjennomfører et langvarig prosjekt som dette. Vi har også fått erfare hvordan det er å samarbeide med en arbeidsgiver for å oppnå et best mulig resultat for begge parter.

Vi vil gjerne takke Jørgen Høyer for hjelpen og samarbeidet. Kommunikasjonen har vært meget god og det har vært svært enkelt for oss å ta kontakt med han, stille spørsmål og diskutere prosjektet gjennom hele semesteret. Jørgen har vært meget forståelsesfull og hjelpsom, som har vært til veldig god nytte og vi er takknemlige for å ha han som veileder. Marios Chatzigeorgiou har også vært til god hjelp med bestilling av deler og kommunikasjon når Jørgen var utilgjengelig, og for det vil vi takke han.

Vår veileder på Høgskolen fortjener også en takk, Eivind Vågslid Skjæveland. Han har gitt oss god oppfølging og hjulpet oss med planlegging og strukturering av oppgaven. I tillegg vil vi rette en takk til faglærer Adis Hodzic for hjelp med diverse programmeringsutfordringer og for raske og gode svar på vår e-postkommunikasjon.

Sammendrag

Sars Internasjonale Senter for Marin Molekylærbiologi studerer biologiske prosesser i marine organismer med fokus på evolusjon og utvikling. Forskningsgruppen S13 studerer modellorganismen Ciona Intestinalis (grønnsekkdyr) i larvestadiet. Larvene legges i en petriskål fylt med vann under et mikroskop. Ved å lyse fluoriserende lys på larvene og tilsette ulike kjemikalier i vannet kan responsen fra visse genmodifiserte celler observeres ved at de lyser opp. Kjemikaliene blir tilsatt i små mengder med stor nøyaktighet gjennom et tynt rør rettet mot larvene. De ulike kjemikaliene blir vekslet mellom ved å åpne og lukke ulike solenoid-ventiler. Før dette prosjektet ble dette gjort ved å manuelt betjene brytere som var provisorisk montert til oppsettet med mikroskopet. Dette førte til vibrasjoner hver gang man skulle veksle mellom de ulike kjemikaliene, som gjorde at bildet fra mikroskopet flyttet på seg og ble uskarpt.

Vårt prosjekt har vært å lage et system som gjør det mulig å styre solenoid-ventilene fra en pc som står ved mikroskopet. Ved hjelp av et brukergrensesnitt på PC-en skal systemet kunne styre ventilene manuelt, samt lage, kjøre og logge sekvenser. Dette ville fjerne problemet med vibrasjoner som forplanter seg til mikroskopet, da man ikke trenger å berøre oppsettet for å styre ventilene.

Systemet består av et dataprogram skrevet i C# med tilhørende brukergrensesnitt, en innkapsling som inneholder en mikrokontroller, relékort og annen nødvendig hardware. I tillegg er innkapslingen utstyrt med banankoblinger som gjør det hurtig å koble systemet opp og ned etter behov. Dataprogrammet snakker med mikrokontrolleren ved hjelp av seriell kommunikasjon gjennom en USB-kobling. Kontrolleren får kommandoer om hvilke ventiler som skal åpnes og lukkes. Den styrer et relékort som igjen styrer ventilene. Signalene til ventilene går gjennom banankoblingene.

Alle kravene i oppgaven ble oppfylt, unntatt ett som var å fjerne bobler i slangene. Dette var ikke enkelt å gjøre automatisk og vi ble enige med den eksterne veilederen vår om at dette kravet kunne sløyfes. I tillegg fikk vi noen ekstra krav som vi også klarte å oppfylle, se avsnitt 2 om kravspesifikasjon.

Innhold

Figurliste

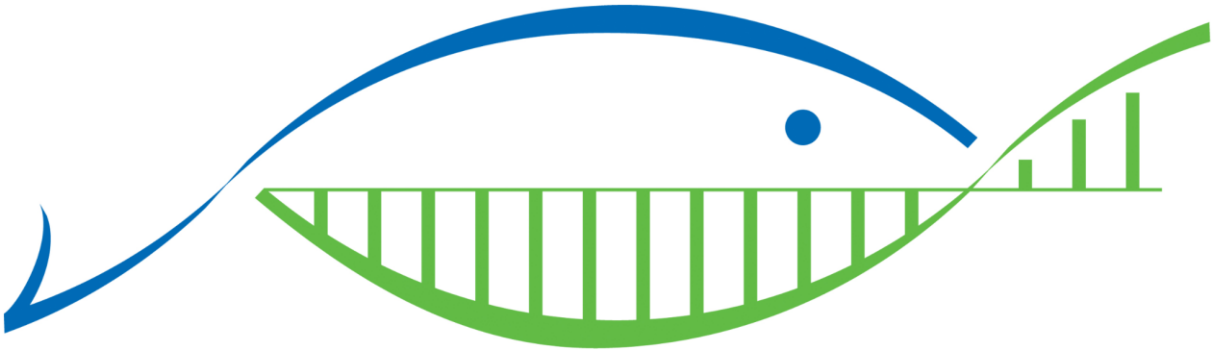
Tabelliste

2 Innledning

2.1 Organisering av rapporten

Vi bruker IEEE* sin standard til kildehenvisning. Vi har markert dette med firkantparenteser, slik [x], som viser hvilke kilder som er brukt i den tilhørende setningen. Ord, begrep og forkortelser som vi mener trenger en forklaring vil være forklart i appendiks A og vil være markert med en stjerne, slik *.

2.2 Oppdragsgiver



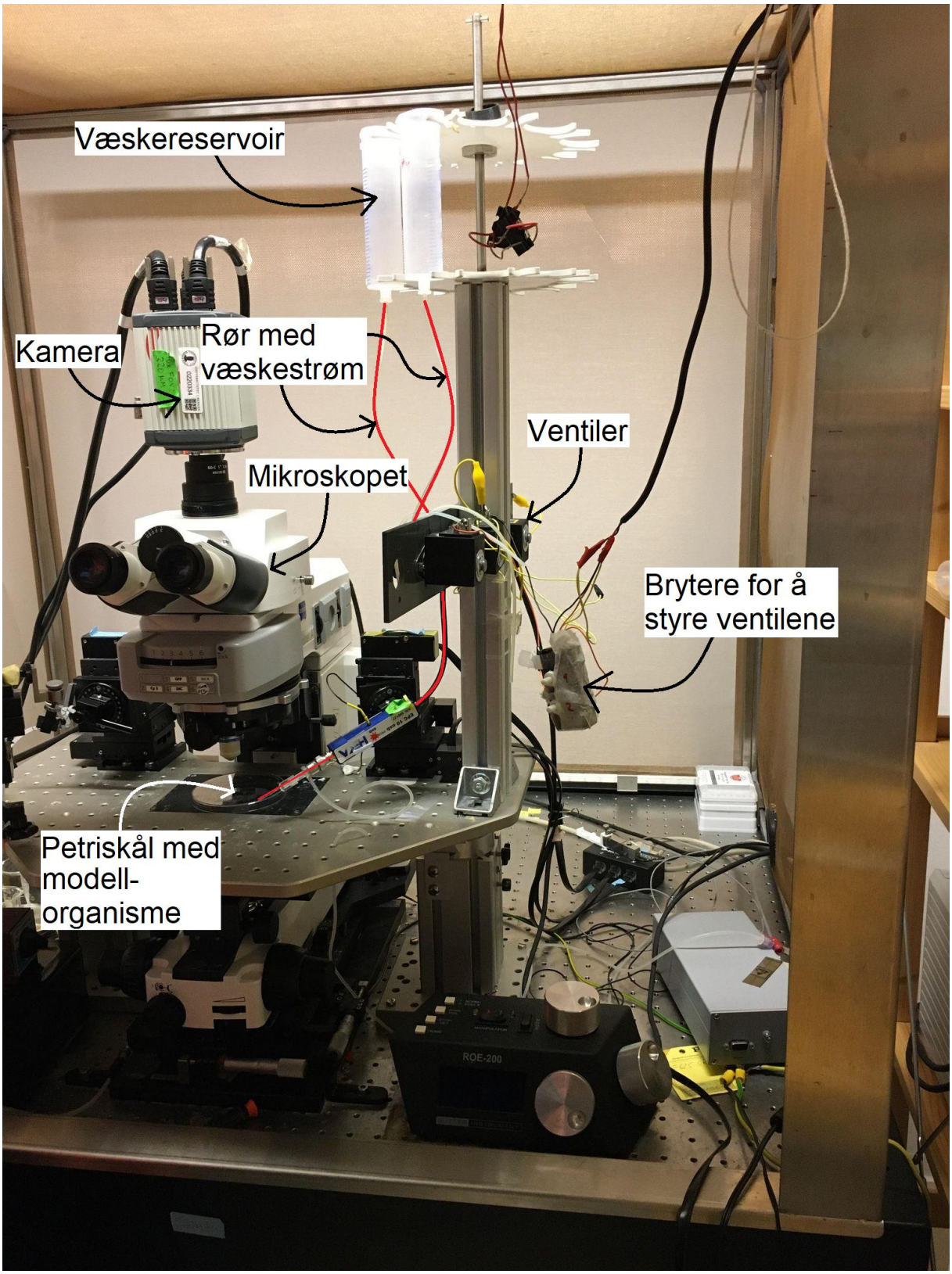
Figur 1: Sars-logo

Sars internasjonale senter for marin molekylærbiologi er en forskningsinstitusjon som studerer utvikling og evolusjon i marine organismer ved biologiske prosesser. Senteret er lokalisert på Høgteknologisenteret i Bergen, fordelt i åtte forskningsgrupper som forsker på biologiske endringer i forskjellige forskningsdyr, som Ciona Intestinalis (grønnsekkdyr). Det ble etablert i 1997, og har siden 2003 samarbeidet med European Molecular Biology Laboratory (EMBL) [1].

2.3 Problemstilling

Oppgaven vår er å utvikle et presist perfusjonssystem som forskningsgruppe S13 ved Sarssenteret skal benytte i sin forskning. Et av forskningsprosjektene deres er å undersøke hvordan celler i sanseapparatet i modellorganismen Ciona Intestinalis i larvestadiet reagerer på ulike kjemikalier. Dyret studeres ved å lyse fluoriserende lys på det, tilsette kjemikalier i vannet det ligger i, og observere hvilke celler som lyser opp ved hjelp av et mikroskop. Slik doseringen av kjemikaliene gjøres i dag, er ved å manuelt trykke på knapper som er fastmontert på et stativ i nærheten av petriskålen som forskningsdyret ligger i. Når disse knappene betjenes fører det til vibrasjoner i stativet som gjør at bildet fra mikroskopet beveger seg og blir uklart. I tillegg er det et problem med luftbobler som kan henge igjen i kanalene (slangene kjemikaliene tilføres i). Luftboblene kan løsne midt i et eksperiment og strømme ut med kjemikalie og feste seg til mikroskoplinsen slik at bildet blir uklart. Vår oppgave er å utvikle et automatisert system som kan dosere kjemikalier i sekvens* ved å betjene et

brukergrensesnitt på en datamaskin. Dette gjør det lettere å gjennomføre eksperiment og fjerner problemet med vibrasjon.



Figur 2: Mikroskop med original løsning

2.3.1 Utvidelsesmål

Det var ønskelig å kunne styre ventilene med brytere uten bruk av PC. Dette vil fungere parallelt med PC-programmet. Denne funksjonaliteten gjør det mulig å åpne og lukke ventilene uavhengig av software. Dette vil være fordelaktig til enkel styring, feilsøking og vil gi redundans i systemet i tilfelle dataprogrammet eller lignende svikter.

Sars ville også at systemet skal være kompatibelt med et konfokalmikroskop*. Dette mikroskopet vil sende sine egne signaler (H/L)* som systemet skal kunne håndtere og styre ventilene ut i fra. Systemet må derfor utvides til å ha innganger og koblinger som er kompatible med dette mikroskopet også.

2.4 Hovedidé for løsningsforslag

Poenget med systemet skulle være å gjøre det lettere å styre tilstrømningen av væsker uten å måtte betjene en knapp som er montert ved mikroskopet. Dette førte til vibrasjoner og uklarhet i bildet til kameraet som filmer gjennom mikroskopet. Av denne grunn var det ønskelig at en skulle kunne styre alt fra en ekstern PC. Oppdragsgiver ville ha et system styrt ved hjelp av en mikrokontroller som skal kommunisere med en PC. På PC-en skal det være et brukervennlig program som kan benyttes til å styre åpningen og lukkingen av ventilene. Dette programmet skal støtte funksjonaliteter som loggføring og muligheten til å kjøre flere ventiler i forhåndsbestemte sekvenser.

3 Kravspesifikasjon

Kravene til dette prosjektet er hovedsakelig de samme som originalt ble gitt i oppgaven, men med noen mindre endringer.

3.1 Liste over kravspesifikasjon

- Fire kanaler for tilstrømming.
- Brukervennlig grafisk brukergrensesnitt.
- Automatisert oppsett for bytte av kanaler.
    - Parametervalg som kanalnummer og varighet skal kunne legges til ved et brukergrensesnitt.
- Logging av parametere fra hvert eksperiment.
- Bytte av kanaler må være så raskt og presist som mulig.
- Den fysiske utformingen må være tilpasset arbeidsplassen der systemet skal brukes.
- Det er ønskelig at systemet styres av en mikrokontroller som Arduino eller Raspberry Pi.

Originalt var det tenkt at systemet skulle kunne styre åtte kanaler, men etter et møte med veileder kom vi frem til at kravet kunne reduseres til fire. I tillegg var det et krav at systemet kunne fjerne bobler som kan forekomme i kanalene. Ettersom dette problemet viste seg å være mer krevende enn forventet, har vi med tillatelse etter diskusjon med veileder besluttet å droppe det.

3.2 Liste over tilleggskrav

- Indikasjonsbar i GUI som viser hvor langt sekvensen har kommet
- Ekstern styring gjennom 5V signal ved bruk av BNC-koblinger
- Manuell styring med fysiske brytere

Tidlig i prosjektet da vi hadde utarbeidet første utkast av brukergrensesnittet fikk vi et tilleggskrav om en indikasjonsbar lignende slike man finner i videoredigeringsprogram, for å indikere hvor i sekvensen programmet er.

Figur 3: Indikasjonsbar

Vi fikk laget en fysisk prototype med delvis funksjonalitet med ganske mye tid igjen av prosjektet, og derfor fikk vi et tilleggskrav som var mulighet for ekstern styring gjennom 5V-signal. Signalet skal komme fra konfokalmikroskopet som kan time styringen av diverse tilleggsutstyr, blant annet når ventilene skal åpnes og lukkes. Mikroskopet sender 5V-styresignal gjennom BNC-koblinger.

4 Analyse av problemet

Systemet skal kunne styre fire kanaler på en gang. Det vil si at mikrokontrolleren til enhver tid skal kunne sende 4 distinkte signaler. Disse signalene skal brukes til å styre hver sin ventil. Ventilene opererer på 24V og de fleste mikrokontrollere på 5V eller 3.3V. På grunn av dette vil det være nødvendig å benytte et relé som kontrolleren kan styre med sitt lavspenningssignal. Det blir da nødvendig å bruke 4 slike reléer for å kunne håndtere kravet om fire kanaler.

Brukergrensesnittet skal på en mest mulig intuitiv måte la brukeren konstruere og kjøre sekvenser, styre ventilene manuelt, koble til mikrokontrolleren, lagre og lese logger. Loggene skal kunne søkes opp etter dato og tid. De skal inneholde hvilke kanaler som var åpne og hvor lenge, hvordan sekvensen gikk (fullført, stoppet, kom. feil, etc.) og eventuelle kommentarer.

4.1 Utforming av mulige løsninger

Oppgaven var gitt veldig åpen og det gjorde at vi kunne løse oppgaven på mange forskjellige måter med forskjellig utstyr, programvare og fremgangsmåte. Vi måtte velge mikrokontroller, programmeringsspråk, program for grafisk brukergrensesnitt, bruk av pumpe eller ventil og generelt design av systemet. De fleste valgene for oppgaven ble tatt ganske tidlig i prosessen, og etter et møte med veileder på Sars-senteret hadde vi en god ide om hvilke program og programmeringsspråk vi skulle velge.

4.1.1 Løsningsalternativ 1

Bruk av ventiler for å styre strømningen av væske i slangene. Når kanalene blir åpnet vil tyngdekraften være nok til at væsken vil strømme. Dette er samme mekaniske som originalt ble brukt. Disse ventilene styres av mikrokontrolleren som mottar meldinger fra PC-en. Programmet på PC-en programmeres i Windows forms* (C#*) med Visual Studio*.

4.1.2 Løsningsalternativ 2

Et alternativ var å bruke pumper i stedet for ventiler. Man kan bruke pumper for kontrollere væskestrømning i slangene. Håpet var at pumpene ville holde tett når de ikke var på, slik at det ikke ville være behov for ventiler. I tillegg var tanken at pumpene kunne fjerne eventuelle bobler som oppstår i slangene. Med dette valget vil det ikke bli flere komponenter å styre, slik at det ikke blir økt kompleksitet, samt at vi potensielt vil oppfylle kravet om fjerning av luftbobler. Mulige problem kan være ujevn væskegjennomstrømning og forplantning av vibrasjon.

4.1.3 Løsningsalternativ 3

Bruk av både ventiler og pumper parallelt for å styre kanalene og fjerne luftbobler. Pumpene slås på for å fjerne eventuelle bobler i slangene. Når dette er gjort kan pumpene slås av, og så kan ventilene

benyttes for å åpne og lukke kanalene. Denne løsningen kan potensielt løse problemet med boblene, og samtidig sikre jevn væskegjennomstrømning.

4.1.4 Vurderinger i forhold til verktøy og HW/SW komponenter

Oppgaven spesifiserte at systemet burde styres av en mikrokontroller av typen Arduino eller Raspberry Pi. Vi har tidligere erfaring med Arduino gjennom faget ELE102, programmering og mikrokontrollere, og egne hobbyprosjekter. I tillegg vil de eneste oppgavene til mikrokontrolleren være å styre fire digitale utganger og kommunisere med en PC. Derfor konkluderte vi med å bruke en Arduino. En Raspberry Pi har mer regnekraft og mulighet for å koble til skjerm, mus og tastatur, men dette er ikke nødvendig for vår oppgave. Når det kommer til den spesifikke modellen av mikrokontrolleren, valgte vi Arduino Micro da den har en liten formfaktor og rikelig med I/O-pinner*. Uten ekstra moduler, er den eneste måten kontrolleren kan snakke med PC-en på gjennom seriell kommunikasjon via en USB-kabel. Dette har vi erfaring med fra ELE124, videregående programmering, og er en enkel og tilgjengelig kommunikasjonsmetode. Alternativt kunne vi benyttet Bluetooth eller Wi-Fi til kommunikasjon mellom Arduinoen og PC-en. Det er ikke alle stasjonære datamaskiner som har Bluetooth og det er greit å slippe å være avhengig av Wi-Fi når ikke andre deler av systemet krever det.

For å styre ventilene valgte vi vanlige elektromekaniske relé. Alternativer kunne vært solid-state-relé eller en type transistor, for eksempel MOSFET*. Den ekstra hurtigheten og muligheten for PWM* disse alternativene kunne tilby var ikke nødvendige for denne oppgaven, og vanlige relé er veldig tilgjengelige fra fysiske butikker og norske nettbutikker med rask leveringstid.

Innkapslingen til det fysiske systemet har vi 3D-printet med en 3D-printer vi fikk bruke på Sars-senteret. Dette valget ga oss muligheten til å lage en innkapsling som var fullstendig designet etter vårt behov. Ulempen var at det tok mye tid å finne ut av selv bare det grunnleggende i designprogrammet, men til gjengjeld sitter vi igjen med litt erfaring med noe vi ikke kunne noe om. Et alternativ kunne vært å finne en generell plastboks som vi kunne laget nødvendige hull i for å montere HW, men dette hadde mest sannsynlig ikke gitt en like egnet innkapsling som den vi designet selv.

Dataprogrammet som skulle styre hele systemet valgte vi å skrive i C# ved hjelp av Visual Studio 2019. Dette språket og utviklingsmiljøet har vi erfaring med fra ELE102 og ELE124. Visual Studio har veldig gode brukervennlige verktøy for å lage GUI*. Et alternativ kunne vært å skrive det i Python*, og ved hjelp av ekstra bibliotek kunne vi laget GUI. Python er mye brukt på Sars-senteret, men vi har ingen erfaring med det og kunne blitt tidkrevende å sette seg inn i, spesielt med tanke på utviklingen av GUI. Vi valgte heller C# som sparte oss for mye tid som vi kunne bruke på å gjøre programmet mer robust, som har vært en viktig prioritering.

4.2 Konklusjon

Det ble bestemt å gå for løsning nr. 1. Ventilene ble brukt i det gamle systemet og vi vet derfor at de fungerer godt til å åpne og lukke de forskjellige kanalene med. Denne løsningen var det enkleste og det mest realistiske løsningsalternativet. Denne løsningen krever at man manuelt trykker litt væske ut av slangene for å få væskegjennomstrømningen til å gå, da slangene er for tynne til at væsken renner gjennom når de er tomme. Dette blir gjort ved å trykke et stempel ned på væskereservoaret, og er slik de har gjort det tidligere. Med denne metoden får man fjernet en del luftbobler, men ikke alle.

Løsning nr. 2 var tenkt for å kunne fjerne luftboblene fra slangen og slippe det manuelle arbeidet som det krever. Etter å ha bestilt en pumpe til testing, og testen den på labben på Høgskulen, fant vi fort ut at denne løsningen ikke kom til å fungere. Grunnen til dette var at pumpene lager store vibrasjoner når de er på, som forplanter seg i mikroskopet og skaper vibrasjoner i bildet. I tillegg var væskestrømningen fra pumpen meget ujevn, noe som ikke er optimalt.

Løsning nr. 3 involverte bruk av både ventiler og pumper for å kunne hindre luftboblene i å oppstå, samtidig som man unngår vibrasjoner i systemet under selve eksperimentene. Denne løsningen ville krevd mye tid og testing på Sars. På grunn av coronatiltak var gruppen begrenset i hvor mye vi kunne være der. I tillegg er rommet mikroskopet står i veldig lite, slik at det er maks tillatt med to personer der om gangen og det ble ofte brukt av de ansatte på Sars-senteret.

Løsningsalternativ 1 var det som passet best etter vurdering av fordelene og ulempene.

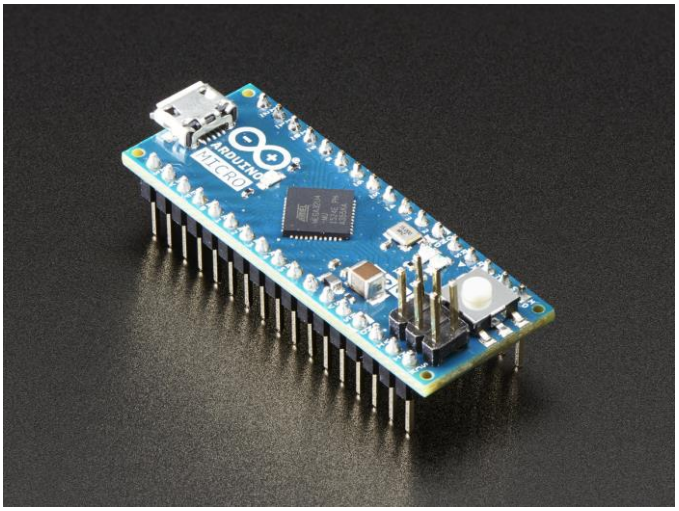
5 Realisering av valgt løsning

5.1 Komponenter

Komponentene ble bestilt av veileder på Sars Jørgen og Marios, og ble bestilt etter hvert som vi fant ut hvilke komponenter vi ville trenge. Vi kunne nok ha planlagt innkjøpene litt bedre for å unngå å bestille flere ganger fra samme leverandør. Komponentene er kjøpt med tanke på at de skal være rimelige og lette å erstatte/bytte. Vi har også prøvd å bruke komponenter som er lette å få tak i og som gir en god brukeropplevelse. Vi har prøvd å holde oss til færrest mulig forhandlere og ideelt noen som har lokaler i Bergen. Derfor er mange av våre komponenter bestilt fra Kjell & Company.

5.1.1 Mikrokontroller

Vi har brukt en Arduino Micro og vi valgte å programmere i Arduino IDE*. Kontrolleren opererer på 5V-logikk og har 20 tilgjengelige inn/utganger og er valgt mest på grunn av sin størrelse. Arduinoen vil være ansvarlig for å ta imot signaler fra PC-en, åpne og lukke ventilene samt lese signaler fra konfokalmikroskopet, og styre ventilene ut ifra det. På grunn av dette er det eneste vi ser etter i mikrokontrolleren at den er lett for oss å programmere, liten og relativt rask. Arduino Micro passer derfor perfekt.



Figur 4: Arduino Micro

5.1.2 Relémodul

Vi trengte fire relé for å kunne sende 24V til ventilene ved hjelp av Arduinoen. Vi valgte en modul med fire relé fra Kjell & Company som har en forretning her i Bergen. Denne modulen er enkel å styre med en Arduino ettersom den har innganger med samme dimensjoner som Arduinoen sine, kan enkelt få jording og 5V fra Arduinoen og opererer på 5V-logikk. Det å ha alle reléene på én og samme modul gjør den både billigere og mindre, noe som er en fordel med tanke på innkapslingen.

Figur 5: Relémodul

5.1.3 Brytere

Til den manuelle styringen, som skal fungere uten bruk av Arduinoen, ville vi og oppdragsgiver ha noen brytere som er lett tilgjengelige, enkle å betjene. I tillegg skal det være lett å se hvilken ventil som er på og hvordan den kan bli slått av manuelt. Vi har derfor valgt noen miniatyrvippebrytere fra Kjell & Company. De er små, lette å koble på, kan festes i lokket på innkapslingen og har tydelige av- og på-tilstander. I tillegg har de lav kostnad og god tilgjengelighet.

Figur 6: Miniatyrbryter

5.1.4 Sikringer

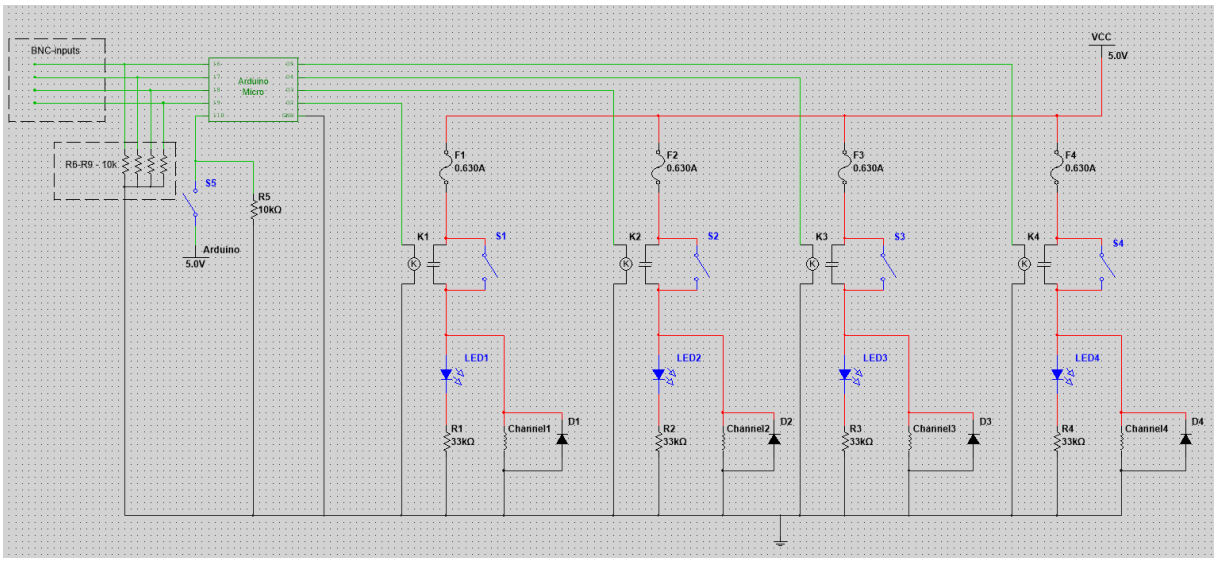
Vi ville sikre hele systemet fra feil og unngå ødelagte komponenter og brann ved hjelp av sikringer. Vi valgte glassrørsikringer på størrelsen 630mA, dette både fordi dette er en fornuftig størrelse for vår applikasjon, men også fordi de har sikringer på denne størrelsen på Sars. Vi har også brukt noen ledningsmonterte sikringsholdere og med disse vil det være enkelt for brukeren å kunne bytte sikringer ved å ta ledningen ut av innkapslingen uten å måtte demontere eller fjerne noen andre komponenter.

Figur 7: Glassrørsikring

5.2 Kretsen

Kretsen er tegnet i MultiSim* og beskriver hvordan Arduinoen og vippebryterne skal kunne styre ventilene i parallell ved hjelp av relémodulen, samt hvordan styringen ved hjelp av BNC-signaler fra det andre mikroskopet vil fungere.

5.2.1 Beskrivelse



Figur 8: Koblingsskjema

Kretsen er delt i fire identiske deler som ligger parallelt med hverandre loddrett på tegningen. Hver av disse representerer styringen av en ventil og inngår hver sin sikring F, øverst, relemodul K, vippebryter S, indikatorlys med tilhørende motstand, LED og R, og en ventil med navnet Channel. Arduinoen er også koblet til alle relémodulene. Både Arduinoen og vippebryteren vil kunne åpne for 24V til ventilene og indikatorlysene. Det vil si at en vil kunne åpne ventiler manuelt parallelt med Arduinoen, men en vil ikke kunne lukke en kanal manuelt dersom Arduinoen har åpnet den. Dette er manglende funksjonalitet, men siden bryterne ikke er ment til å brukes samtidig som Arduinoen, men heller mens Arduinoen ikke er operativ, vil vi heller holde kretsen enkel.

Arduinoen får også signal fra BNC-kablene som vil komme fra det andre mikroskopet. Disse signalene blir bare tatt i betraktning dersom vippebryter S5 er lukket, da vil Arduinoen høre eksklusivt på disse signalene og ignorere seriell kommunikasjon med PC-en. Alle BNC-inngangene har også hver sin 10k Ohm pull-down motstand. Denne motstanden forsikrer at inngangen ligger på 0V når BNC-portene ikke er koblet til. Uten en slik motstand ville inngangene vært floating*. Det vil si at de vil kunne hoppe mellom høyt og lavt ved tilfeldige tidspunkt ved å plukke opp elektromagnetiske signal i luften. Vi trenger derfor å koble inngangen til jord med noen motstander som stopper signalet fra BNC-kablene fra å gå til jord.

Kretsen blir forsynt av en 60W 24V strømforsyning som blir koblet til innkapslingen ved hjelp av en DC-kontakt. Dersom noen komponenter kortslutter eller trekker for mye strøm vil sikringene ryke og redde komponentene og strømforsyningen.

5.2.2 Strømforbruksanalyse

Ettersom vi skal bruke sikringen trenger vi å vite cirka hvor mye strøm kretsen vår kommer til å trekke slik at vi kan velge riktig størrelse på sikringene. Ettersom ventilene trekke 375mA hver og hver ventil vil ha sin egen sikring, vil vi trenge noen sikringer som kan lede mer enn dette, men heller ikke være for store slik at de ikke ryker når de skal. Vi valgte derfor 640mA. Dette er også den størrelsen som Sars har brukt i ventilene tidligere. Dette tydet på at valget var fornuftig. I tillegg vil de da ha ekstra sikringer på Sars hvis noen må byttes.

Vi måtte også sjekke om strømforsyningen de hadde på Sars hadde høy nok effekt til å kunne levere strøm til flere ventiler samtidig, noe som ikke ble gjort med den tidligere løsningen. Strømforsyningen må kunne levere minst 375mA og 24V til fire ventiler samtidig. $0.375A \cdot 24V \cdot 4 = 36W$. Strømforsyningen de hadde på Sars var bare på 24W, og vi måtte derfor gå til innkjøp av en større. Vi valgte en med lang kabel, for lettere bruk, på 60W.

5.2.3 Lodding av kretsen

En av oss har tidligere erfaring med lodding og siden Sars har loddeutstyr på senteret bestemte vi oss for å lodde kretsen der. Loddeprosessen gikk ganske bra og uten store problemer. Annet enn noen komponenter som ble loddet på feil sted og dermed måtte avloddes og noen loddinger med en generøs mengde tinn, er vi fornøyde med arbeidet. Vi som ikke hadde erfaring med dette fikk lære mye og fikk verdifull erfaring om lodding som er en god ferdighet å ha. Siden vi har planlagt å ha kretsen hevet fra bunnen av boksen kunne vi lodde på både undersiden og oversiden av hullbrettet ute å være bekymret for plass. Vi loddet all jordingen på en plass for enkelhets skyld, og det samme gjorde vi med 24V. Bananpluggene vi skulle bruke til å lage banankabler med viste seg å være svært dårlige, festemekanismen var løs og vi måtte lodde endene av kablene med en god del tinn for at endene skulle kunne henge godt fast.

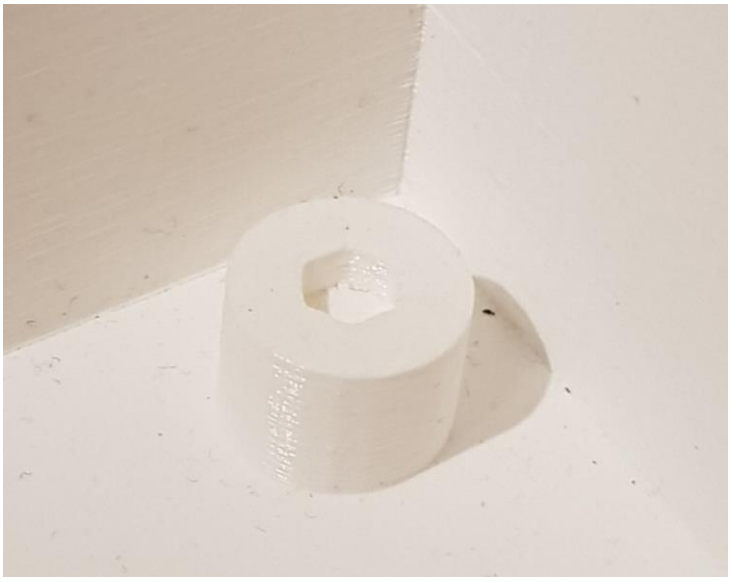
5.3 Design av innkapsling

5.3.1 Hovedide til designet

Oppdragsgiver ville ha en innkapsling til kretsen som var kompakt, lett å jobbe med, lett å flytte på og enkel å koble ledninger til. I tillegg skulle den fungere som en manuell kontroller med vippebryterne i lokket. Vi bestemte oss fort for å gjøre dette ved hjelp av 3D-printing ettersom Sars har en 3D-printer på senteret som de bruker til å printe diverse deler og bokser. I tillegg er 3D-printing en enkel og billig måte å kunne designe noe som passer perfekt til vårt formål og som kan lages flere prototyper av på kort varsel og ved lav kostnad.

Vi hadde ingen erfaring med 3D-printing fra tidligere, men vi hadde veldig lyst til å lære om det og så frem til utfordringen. Første steg var å planlegge hvordan boksen skulle se ut og hvilke hull og funksjonaliteter den skulle ha. Vi visste akkurat hva som skulle inn og ut av boksen ved hjelp av kretstegningen vår. Vi trengte en inngang til vår 24V strømforsyning, en USB-inngang til å koble Arduinoen til PC-en, fire bananplugginnganger til å styre de fire ventilene og en bananplugginngang for å jorde alle ventilene. I tillegg måtte lokket ha åtte hull, fire til vippebryterne og fire til indikatorlysene. Etter å ha målt hullbrettet* og bestemt oss for hvor mye plass vi trodde alle komponentene ville trenge inni boksen, begynte vi å tegne i 3D-programmet FreeCad*.

Vi hadde bestemt av vi ville ha 1 cm rom på alle sider av hullbrettet, samt 1 cm under, for å gi litt bedre plass til kabler og gjøre det betydelig lettere å montere og demontere komponenter inne i boksen. Vår ide for å heve kretsen var å ha fire sylindere under hvert hjørne av kortet, hvor hver sylinder har en mutter som er limt fast 5mm ned i sylinderen, se figur 9. Disse vil da passe slik at en bolt som går gjennom hullene i kortet vil kunne festes i gjengene i mutterne i sylindrene og på den måten vil kortet være festet.

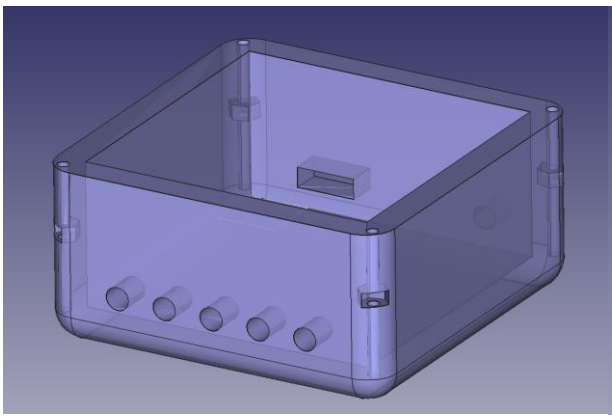


Figur 9: Sylinder-opphøyning

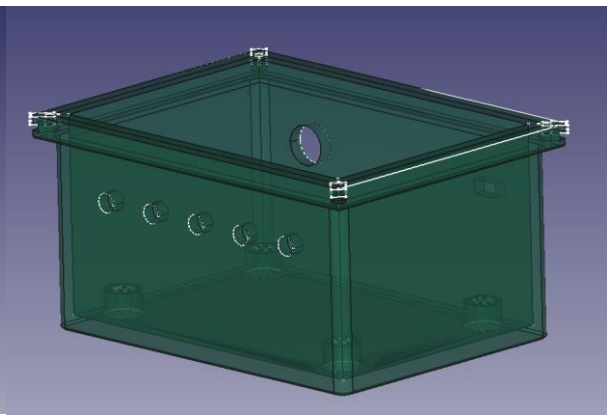
Vi ville også ha en lignende festemekanisme for lokket. Det var ønskelig at lokket var godt festet ettersom det skulle være vippebrytere i lokket som skal betjenes. Dessuten ville vi at det skulle være mulig å løfte hele boksen fra bare lokket uten problemer. Vår løsning var å designe en «leppe» på boksen der vi kunne sette inn noen muttere i hjørnene slik at lokket kunne festes ved fire bolter som skrus fast i boksen ved hjelp av mutternes gjenger, se figur 11.

5.3.2 FreeCad

Bare en av oss hadde brukt FreeCad før, men han hadde ikke mye erfaring med det. Vi hadde noen utfordringer i starten, ettersom programmet ikke var særlig brukervennlig eller intuitivt. Etter litt undersøking fant vi funksjonalitetene som vi trengte for å konstruere boksen vår [8]. Vi startet med å lage en helt enkel boks før vi tok noen målinger, bare for å teste ut programmet og se at vi kunne bruke det. Se figur 10. Etter dette ville vi lage en boks med riktige dimensjoner og en leppe for festing av lokket. Vi valgte en høyde på 8cm og en veggtykkelse og leppehøyde på 5mm, se figur 11.



Figur 10: Innkapsling, første utkast

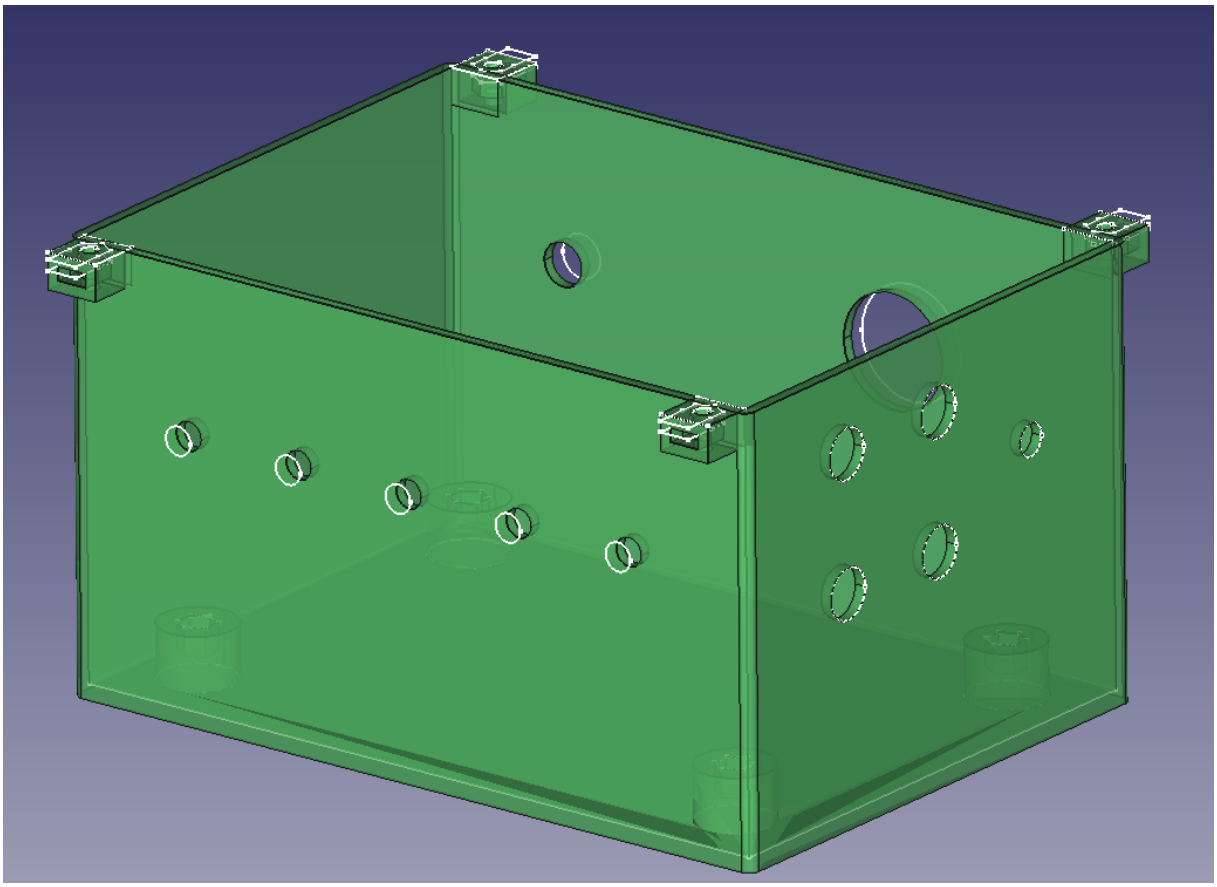


Figur 11: Innkapsling, med leppe

5.3.3 Revisjoner

Etter å ha reist ned på Sars for å prøve å 3D-printe versjon 1.0 av boksen støtte vi på et problem. Vårt design var for stort for 3D-printeren å printe. Printeren har et arbeidsområde på 15x15cm, boksen vår var 16x12cm. Vi hadde to valg, vi kunne enten printe boksen i to deler og feste de sammen etterpå, eller redesigne boksen. Vi kom fort fram til at det var noen forbedringer som kunne gjøres og dermed spare nok plass til å kunne printe hele boksen på en gang. Ekstern veileder, som har mer erfaring med 3D-printing enn oss, informerte oss om at 2mm var nok en mer rimelig veggtykkelse. I tillegg ble vi informert om at leppen brukte unødvendig mye plass og kom også til å ta svært lang tid å printe.

Etter dette satt vi oss ned for å endre på designet. Veggene ble redusert til 2mm og leppen ble fjernet til fordel for noen mindre matrealintensive «ører» som bare henger på langsiden av boksen ved de fire hjørnene. Ørene er fortsatt mer enn sterke nok men minker plastmengden og lengden på boksen. Designet var nå 14.4x10.4cm og printingen av den første prototypen kunne starte.



Figur 12: Innkapsling, med ører for mutterfeste, endelig design

5.4 3D-Printing

Vi sendte vår første prototype for printing til oppdragsgiver 8. Mars og etter noen dager var den klar. Oppdragsgiver valgte å printe med 10 prosent fyllmengde. Dette vil si at bare de ytterste lagene av boksen er solide, mens volumet inne i veggen og alt annet kun er fylt med en tynn struktur som tar opp bare 10 prosent av volumet de befinner seg i. På denne måten vil vi spare både tid på å printe og plastikk. Oppdragsgiver mente og at det ville nok også være nok så sterkt selv med lav fyllprosent.

Vår første printing gikk ikke helt som planlagt. Vi støtte på et par problemer som måtte undersøkes og fikses til neste printing.

5.4.1 Forskyvning

Det største problemet var at hele boksen har forskjøvet seg ca. 70 prosent inn i printeprossessen. Oppdragsgiver visste ikke hva som hadde skjedd ettersom printeprosessen ikke hadde kommet så langt da han reiste fra jobb. Etter å ha sett på tidspunktet printen var ferdig, konkluderte vi med at hendelsen sannsynligvis tok sted på kvelden 9. Mars. Vi tror at boksen eller printeplattformen har blitt flyttet, sannsynligvis da noen med uhell har kommet borti maskinen. Det var derfor ikke så mye vi kunne gjøre

for å forhindre dette til neste gang, men vi konkluderte med at det nok var usannsynlig at det ville skje igjen.

Figur 13: Forskyvning i print av tidligere design

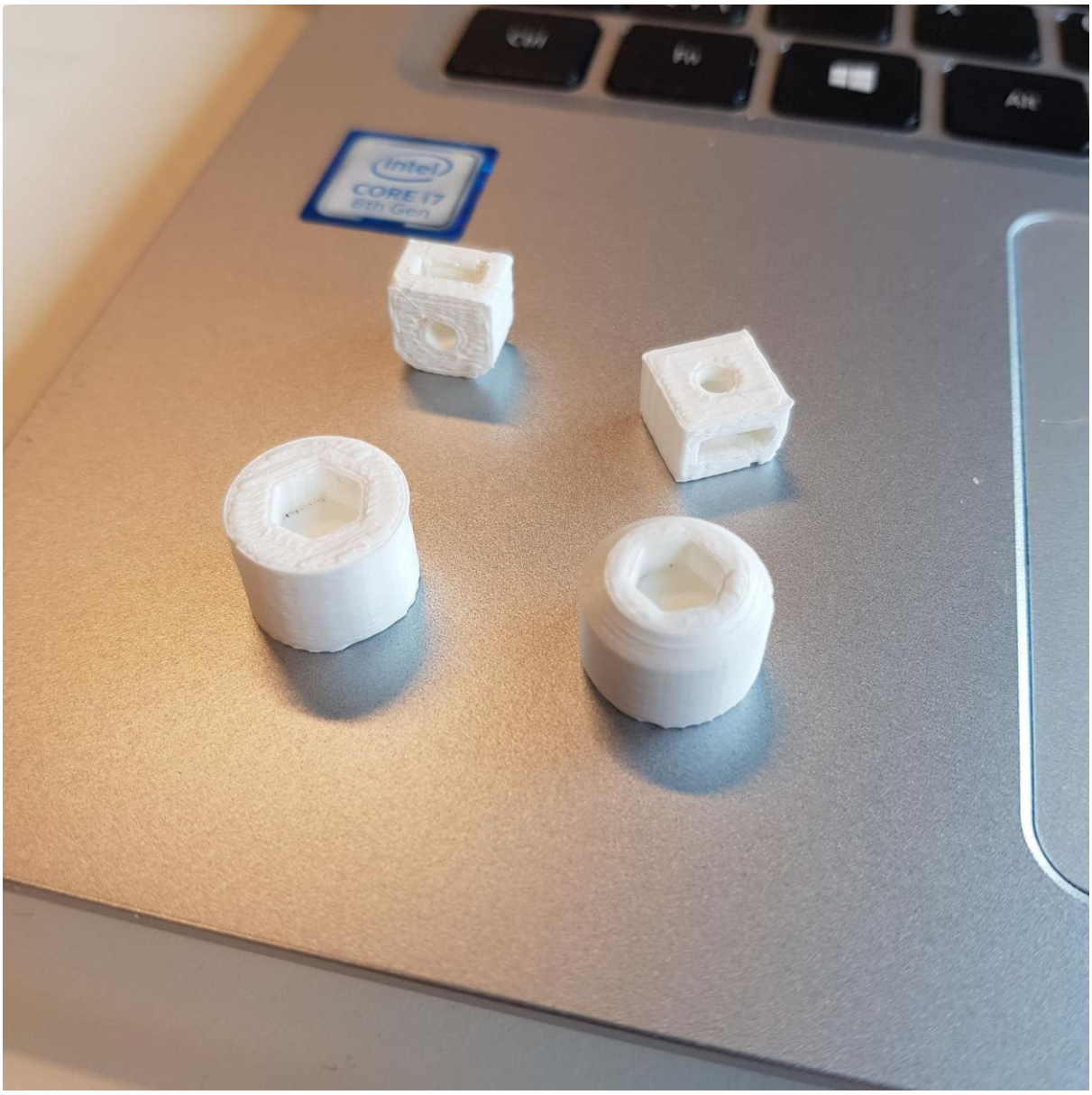
5.4.2 Feil i designet

Det viste seg også at vi hadde gjort noen feil i modellen. Noen av hullene var ikke riktig størrelse, men av to forskjellige grunner. Inngangene til bananpluggene og strømforsyningen var for store, dette var på grunn av en enkel radius/diameter blanding fra vår side og vi endret modellen i FreeCad straks. Hullene til mutterne, både i hevingssylindrene og i «ørene», var også for små. Grunnen til dette viste seg å være at plastikken ekspandere litt etter printing, og slike hull som passer på 0.1mm skala, blir mindre enn på modellen. Ikke mye, bare noen tiendels millimeter, men nok til at mutterne ikke passet. Etter denne oppdagelsen bestemte vi oss for å printe noen sylindre og «ører» med mutterhull, separat fra boksen, med forskjellige høyde og bredde på hullene. På denne måten kunne vi teste hvilke hull som ville passe til neste boksdesign.

5.4.3 Printing av mutterhull

Vi lagde fire modeller i FreeCad, to sylindre og to «ører», med varierende diameter på hullet. Versjon 1.0 av boksen, som ble forskjøvet i printeprosessen, hadde mutterhull på 5.5mm. Derfor bestemte vi

oss for å prøve ut 6.0mm og 6.5mm hull denne gangen. Etter printing og testing fant vi ut at 6.0mm passet bra og vi ville derfor bruke det i det nye designet av boksen.



Figur 14: Testprint av ører og sylinderopphøyninger

5.4.4 Første brukte prototype

Etter å ha fikset på problemene og feilene til det første designet i FreeCad, sendte vi det nye designet til veileder på Sars til printing og dagen etter hadde vi den nye versjonen klar. Denne var også printet med 10 prosent fyllmengde for at det skulle gå raskere ettersom vi forventet at dette ikke kom til å bli den siste boksen vi printet. Vi må ta hensyn til ønsket mengde fyllprosent etter at vi har testet boksen. Denne gangen ble boksen printet uten forskyvning, og etter å ha skrapt bort støttematerialet som ble printet inni mutterhullene i "ørene" var boksen klar til å bli montert på. Alle hullene passet slik de skulle og vi begynte da å koble og lodde kretsen som boksen skal huse.

5.4.5 Siste versjon

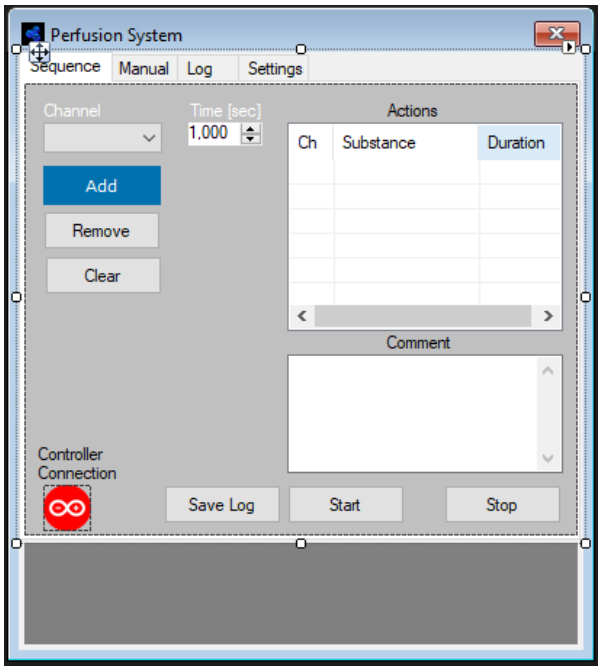
Etter at det ble ønskelig at systemet også skulle være kompatibelt med et annet mikroskop som kunne bli tilsendt 5V signaler via en PC i rommet, måtte vi også endre på designet til boksen slik at den var kompatibel med innganger til noen BNC-kabler som kom fra det andre mikroskopet. Vi kom også frem til at det ville være praktisk med en vippebryter som lar en veksle mellom at Arduinoen blir styrt av vår programvare, og signaler fra disse BNC-kablene. Vi laget derfor 5 nye hull i boksen og endret USB-inngangen, ettersom kabelen vi hadde kjøpt virket til å være dårlig kontakt i. Vi sendte det nyeste designet til Sars og ble enig med oppdragsgiver om at vi skulle bruke 50 prosent fyllmengde denne gangen. Grunnen til dette var at de lengste veggene på boksen bøyde seg når en prøvde å dytte eller dra kabler inn eller ut av boksen og vi ville derfor prøve å forsterke veggene. Hvis det viste seg at dette ikke var nok til å forsterke veggene til et ønskelig nivå, ville vi legge til to flere ører på midten av de lengste veggene for å unngå slik bøying.

5.5 Utvikling av programvare

Utviklingen av programvaren var den mest tidkrevende oppgaven med god margin. I det ferdige programmet har vi benyttet kunnskap fra store deler av pensum fra alle programmeringsfagene vi har hatt. Dette førte til at en god del oppslag og oppfriskning ble nødvendig, noe som tar betydelig tid. I tillegg har vi brukt mye tid på å utvikle et intuitivt og estetisk brukergrensesnitt. Vi gikk gjennom flere iterasjoner før vi ble fornøyde med resultatet. Dette har aldri vært stort fokus på i fagene vi har hatt tidligere, så dette har vært en læringsprosess for oss. Ettersom dette programmet skal brukes over lengre tid, ulikt mindre program vi har utviklet i forbindelse med utdanningen for læringens del, så hadde vi stort fokus på å gjøre programmet så robust som mulig og fjerne alle mulighetene for programkrasj som vi oppdaget. Dette var ved noen tilfeller en stor utfordring da vi måtte prøve å ta hensyn til alle mulige “hva om” tilfeller, spesielt med tanke på at det er ting i virkeligheten som blir styrt av programmet og brukeren kan plutselig velge å lukke det midt i en sekvens. Programmet kjører til tider flere tråder, blant annet under kjøring av sekvenser, dette ga interessante problemstillinger som måtte tas hensyn til, slik som for eksempel trådsynkronisering.

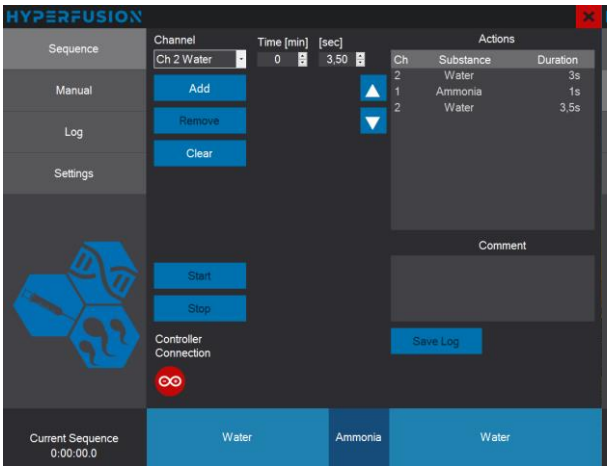
5.5.1 Brukergrensesnitt

Brukergrensesnittet la vi mye tid i da vi så på dette som en sentral del av oppgaven. Målet var at personer som ikke har erfaring med programmet raskt skal kunne sette seg inn i det, gitt at de vet hva formålet til programmet er.

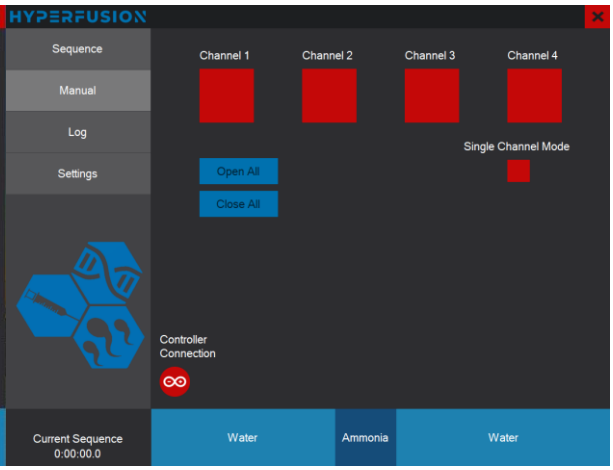


Figur 15: Tidlig design av GUI

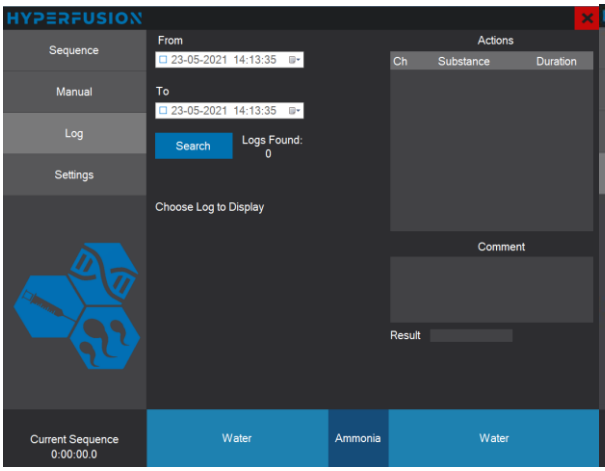
Den første iterasjonen av brukergrensesnittet designet vi på et whiteboard på et grupperom på skolen. Der fikk vi diskutert hva vi tenkte ville fungere bra og dårlig. Vi ble enige om hvilke grafiske element vi skulle bruke og en struktur som var fordelt over flere faner. Etter denne planleggingen i startfasen fikk vi bygget opp GUI-en i Visual Studio ganske raskt, og den kunne fungere som en testplattform for de ulike logiske funksjonalitetene, slik som at de riktige tingene skjer når en knapp blir trykket på. Denne første versjonen var ikke særlig fin å se på og utseendet endte opp med å undergå store endringer, som etter vår mening var forbedringer. Til tross for dette ble oppsettet med de ulike grafiske elementene nokså uforandret gjennom prosjektet, da vi synes at det fungerte bra slik vi originalt designet det. Noe vi endret var måten man skiftet mellom de ulike fanene. Først brukte vi en fane-velger, men denne hadde noen programmeringsmessige begrensninger, i tillegg til at den ikke så særlig bra ut, som gjorde at vi byttet den ut med knappsystemet helt til venstre i GUI-en. Det er dette som brukes på den endelige versjonen. Vi designet også vårt eget grafiske element som er den progresjonsbaren nederst i GUI-en. En slik type grafisk indikator på hvordan sekvensen ligger an, var et ønske fra vår eksterne veileder. Denne måtte vi programmere logikken til helt fra grunnen av.



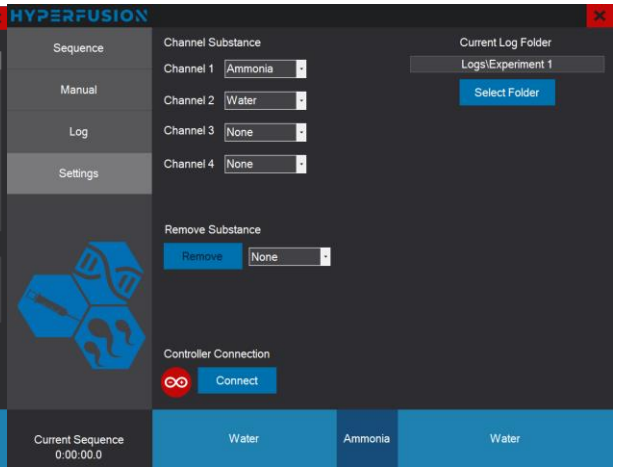
Figur 16: Ferdig GUI, "Sequence"



Figur 17: Ferdig GUI, "Manual"



Figur 18: Ferdig GUI, "Log"



Figur 19: Ferdig GUI, "Settings"

For å lære mer om hvordan vi kunne gjøre brukergrensesnittet mer behagelig å se på, studerte vi ulike eksisterende programmer som mange av oss bruker i hverdagen. Vi forsøkte å legge merke til hvordan ting var organisert og fargevalgene som var gjort. Ved hjelp av digitale bildeverktøy kunne vi undersøke fargene nærmere ved å finne RGB-verdien* til pikslene. Da oppdaget vi en trend som gikk igjen flere plasser, nemlig å bruke flere fargetoner av samme fargen. For å gjøre en farge mørkere senker man R-, G- og B-verdien til pikslene med samme prosentandel. Så dersom en farge har RGB-kode 100, 150, 200 vil en 10% mørkere versjon ha kode 90, 135, 180. Ved å benytte ulike fargetoner kan man lettere oppnå noe som er behagelig å se på. Fargetemaet vi valgte var ulike mørke gråfarger for bakgrunner og en blåfarge til knappene. Grunnen til å velge mørke gråfarger er siden rommet programmet skal være i bruk i skal være mørkt under eksperimentene, da blir det mindre lys fra dataskjermen. Dette er også mer behagelig å se på i mørket.

5.5.2 Programkode

Kort tid etter vi hadde designet den første versjonen av brukergrensesnittet begynte vi å utvikle programkoden med det som utgangspunkt, da vi følte oss ganske sikre på at valget av grafiske element ville forbli nokså uendret. Arbeidsmetoden vår var å legge til en begrenset mengde funksjonalitet, for å så teste det og fikse de fleste feilene. På denne måten kunne vi begrense omfanget av feilsøkingen for å gjøre utviklingsprosessen mer effektiv, men det var allikevel tidkrevende når man summerer all feilsøkingstiden. Dessuten ble det stadig mer komplisert å feilsøke etter hvert som programmet ble mer og mer ferdig, og dermed flere metoder og klasser involvert i logikken.

Et av kravene som ble gitt med oppgaven var at bytte av kanaler skulle være så raskt og presist som mulig. Dermed måtte vi finne gode løsninger for å time kommandoene som ble sendt til mikrokontrolleren og at GUI-en oppdaterte seg i samsvar med dette. Vi gikk gjennom flere løsninger som hadde hver sine problemer, før vi ankom en god middelvei mellom timing og ressursbruk. I den første løsningen brukte vi en Forms-timer som kan kjøre på en bakgrunnstråd og bruker tilnærmet ingen ressurser. Denne typen timer kaller hendelser* hver gang den innstilte tiden har gått. Problemet var at vi trengte å time mange små intervaller for å kunne flytte progresjonsbaren litt og litt, og det var noe Forms-timeren ikke egnet seg til da den er veldig unøyaktig på korte intervaller og har dårlig repeterbarhet.

Den neste løsningen vår var å bruke while-løkker til å sjekke et StopWatch-objekt*, noe som viste seg å være en dårlig løsning. Ved bruk av while-løkker på denne måten, vil prosessoren kjøre gjennom løkken så fort den klarer helt til intervallet er ferdig. StopWatch-objektet gir veldig nøyaktig tid, men man må selv velge når og hvor ofte man vil lese av tiden. Med denne while-løkke-løsningen ble StopWatch-objektet sjekket mange ganger per millisekund, noe som er helt meningsløst for dette prosjektet, i tillegg til at tråden denne koden kjørte på ble maksimalt belastet. Så dersom datamaskinen programmet kjører på har fire fysiske tråder, og to tråder trengte å time noe på denne måten, ville 50% av prosessoren være i konstant bruk til å time noe unødvendig nøyaktig.

Timing-løsningen vi endte opp med å bruke var en kombinasjon av de to andre. En forms timer brukes for å sjekke et StopWatch-objekt i fornuftige intervaller, noe som kan føre til litt lengre eller kortere tider enn den ønskede tiden, men dette er snakk om få millisekund. På denne måten får vi rimelig nøyaktig timing og ressursbruken er tilnærmet null. Dersom et steg i en sekvens varer i for eksempel tre sekund, kan vi med denne løsningen time de tre sekundene delt opp i mange små intervaller for å gradvis øke fremdriftsbaren i GUI-en, og sende en kommando til Arduinoen når ganske så nøyaktig tre sekund har gått.

```
void SequenceTickTimer_Tick(object sender, EventArgs e)
{
    lock (stopSync) bufferStop = stopCommand;

    if (stopwatch.ElapsedMilliseconds >= currentAction.Duration - (TICK_TIMER_INTERVAL / 2) || bufferStop)
    {
        waitForActionEwh.Set();
    }
}
```

Figur 20: Valgt timing-løsning

Under utviklingen av programmet har vi prøvd å tenke på hva som skjer om brukeren gjør noe som faller utenfor "normal bruk". For eksempel hva som skal skje dersom brukeren lukker programmet ved å krysse det ut under kjøring av en sekvens. Ved slike tilfeller har vi forsøkt å gjøre det sånn at programmet alltid vil prøve å lukke alle kanalene før det avsluttes. Denne problemstillingen førte til at vi måtte friske opp våre kunnskaper om trådsynkronisering. En utfordring vi møtte på var at programmet frøs når man prøvde å lukke det under kjøring av en sekvens. Dette skjedde ettersom GUI-tråden ventet på at sekvens-tråden skulle avslutte, men sekvens-tråden ventet på at GUI-tråden skulle oppdatere brukergrensesnittet. Dette er en såkalt deadlock. Løsningen vi kom frem til var å la sekvens-tråden sjekke om GUI-tråden holder på å avslutte før den prøver å be den om å oppdatere brukergrensesnittet.

5.5.3 Kommunikasjonsprotokoll

Kommunikasjonen mellom C#-programmet på datamaskinen og Arduinoen foregår over seriell kommunikasjon via en USB-kabel. Kommandoene blir sendt som datatypen streng. Arduinoen gjør ikke mye annen logikk enn å motta meldingene, dekode de og endre tilstanden til de digitale utgangene dersom den fikk en gyldig melding. Formatet til kommandomeldingene er slik: "$A/B#%", hvor "A" erstattes med kanalnummeret og "B" erstattes med 0 eller 1 ettersom kanalen skal åpnes eller lukkes. "$" indikerer starten av meldingen, "#" indikerer slutten og "%" er symbolet Arduinoen skal lese til i seriell-bufferen*. Dersom den ikke finner %-symbolet timer den ut lesingen etter 20 millisekunder. Når Arduinoen har mottatt og eventuelt isolert en komplett melding, verifiserer den at alle symbolene er det den forventer, og at tallene i meldingen er innenfor de intervallene som gir mening, [1, 4] for kanalvalg og [0, 1] for åpne/lukke-kommandoen.

For at kommunikasjonen skulle bli mer robust, har vi implementert et system i C#-programmet som krever at Arduinoen svarer "Valid" når den mottar en gyldig melding. Når det ordet blir mottatt av C#-programmet, blir kommandoen regnet som gyldig og utført. Dersom "Valid" ikke blir mottatt, forsøker dataprogrammet å sende meldingen på nytt og venter på svar igjen. Dette gjøres opptil tre ganger før det vil dukke opp en feilmelding som sier at det er noe galt med kommunikasjonen til mikrokontrolleren. Denne løsningen blir benyttet når dataprogrammet sender hvilken som helst

kommando til Arduinoen. Kjøres det for eksempel en sekvens vil den ikke gå videre til neste steg før Arduinoen svarer med "Valid". Disse tre kommunikasjonsforsøkene blir utført i bakgrunnen uten at brukeren merker noe, med mindre det går tre forsøk uten suksess, da dukker en feilmelding opp.

6 Testing

Det ble utført en rekke små tester i løpet av prosjektet for å forsikre oss om at alt virket som det skulle før vi begynte på noe nytt.

6.1 Testing av programvare

Når en utvikler programvare vil en alltid kjøre tester etter hvert som nye funksjonaliteter blir implementert for å sjekke om det fungerer og om det er noen sideeffekter som må fikses. Vår første test var av kommunikasjonen mellom PC-en og Arduinoen. Arduinoprogrammet inneholdt en enkel dekrypteringsalgoritme som kunne ta imot seriell data fra PC-en via USB-porten og utføre kommandoer ut fra det. Arduinoen var kun koblet til noen LED-lys, men hvis vi kan styre LED-lys, kan vi også styre ventilene. Testen var en suksess og vi testet senere seriell kommunikasjonene begge veier som også fungerte.

6.2 Testing av pumper

I løpet av utviklingen av programvaren fikk vi bestilt en av pumpene som vi kanskje skulle bruke til fjerning av eventuelle bobler i slangene. Før vi bestemte oss for om vi skulle bruke pumper i systemet, ville vi teste to ting. Vi tok med oss pumpen på labben på høgskolen, hvor vi ville pumpe noe vann fra en kopp til en annen. Det vi ville teste var blant annet om slangen gjennom pumpen var tett dersom pumpen ikke var i gang. Dersom den var tett ville vi kunne la være å benytte ventiler, siden pumpen vil fungere som en ventil. Dette stemte, pumpen var tett. Den andre tingen var at vi ville teste hvor mye pumpen vibrerte under bruk. Dette viste seg å være en del, i tillegg var strømningen av vann nokså ujevn, som ikke er ideelt for vårt formål. Slik vibrasjon vil gjøre det slik at det vil bli nesten umulig å kunne se noe i mikroskopet ettersom vibrasjonene fra pumpene vil propagere gjennom stativet de henger i og mikroskopet vil forsterke dem. Bruken av pumper under eksperimenter ble derfor uaktuelt. Dersom pumpene skulle blitt brukt til å fjerne bobler måtte de har vært koblet i parallell med ventilene og bare bli brukt på forhånd, før noen eksperimenter hadde startet. Med denne informasjonen bestemte vi oss for å ikke jobbe med dette kravet, fokusere på resten av systemet og heller komme tilbake til det dersom det var tid, eller heller implementere en annen funksjon som var lettere å realisere.

6.3 Testing av 3D-printing

Som nevnt i del 5.4 gjorde vi litt testing for å finne ut av hvordan vi ville printe husingen til systemet. Den viktigste delen var å printe små deler av boksen for å enklest mulig finne ut av hvilke dimensjoner som ville passe best når de ble printet. Dette gjorde vi etter at vi så på den første prototypen og la merke til at dimensjonene på den fysiske printen ikke var helt de samme som i modellen i FreeCad. Hullene var litt mindre, sannsynligvis grunnet utviding i plasten etter at den blir nedkjølt. Denne testen sparte oss for mye bry og var viktig for å få en innkapsling som passet slik vi trengte.

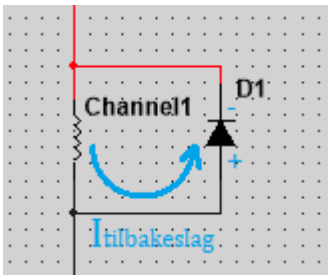
6.4 Testing av manuell styring på Sars

Når kretsen var ferdig loddet, boksen printet og systemet klart for manuell bruk, ble systemet gitt til oppdragsgiver for testing. Vippebryterne på boksen kunne brukes til å styre ventilene. Styring gjennom PC-en via Arduinoen var ikke enda i drift. Oppdragsgiver brukte systemet i cirka en uke og rapporterte tilbake om hvordan systemet fungerte. Bryterne hadde fungert som de skulle og styringen var rask og responsiv. Selv om systemet egentlig bare var noen brytere koblet til en 24V strømforsyning var det greit å få testet dette. Spesielt å få stresstestet komponentene. Det viste seg nemlig at en av bryterne sluttet å fungere mot slutten av testperioden og et av LED-lysene ikke lenger var responsive. Etter å ha undersøkt problemet oppdaget vi at en av ledningene hadde løsnet fra skrukoblingen i releet. Vi bestemte oss for å lodde sammen de ledningene som skal være i samme skrukobling slik at ingen av ledningen vil kunne ligge løst og kunne falle ut.

Først trodde vi at et lignende problem kunne være årsaken til at LED-lyset sluttet å fungere, men vi fant ingen løse ledninger. Etter å ha målt med multimeter fant vi heller ingen feil i kretsen og LED-lysene fikk riktig mengde spenning. Vi koblet da en ny LED i parallell for å se om den ville lyse. Det gjorde den ikke og vi bestemte oss for å ta en diodetest med multimeteret. Den nye viste det forventede spenningsfallet på 0.7V, men den gamle dioden viste bortimot 0V, som forklarer hvorfor ingen av dem lyste når de sto i parallell. Dioden var altså ødelagt og vi byttet den med den nye og alt fungerte igjen.

Noen dager senere var vi på Sars for å prøve å implementere resten av systemet slik at det også kunne testes. Under testingen oppdaget vi at når vi styrte ventilene ved å bruke vippebryterene, blinket alle diodene i et øyeblikk hver gang en bryter ble betjent. Etter en stund sluttet også en av LED-ene å fungere igjen, og vi visste da at det var et problem med kretsen som gjorde at diodene ble ødelagt, og det var nok også grunnen til at alle diodene blinket. Trygve identifiserte problemet, det var tilbakespenning fra spolene i ventilene. Når ventilen blir slått på, går strømmen gjennom en spole som skaper et magnetisk felt som åpner ventilen. Endring i magnetiske felt skaper strøm og når ventiler skrues av vil spolen miste sitt magnetiske felt og det blir indusert en spenning i motsatt retning enn

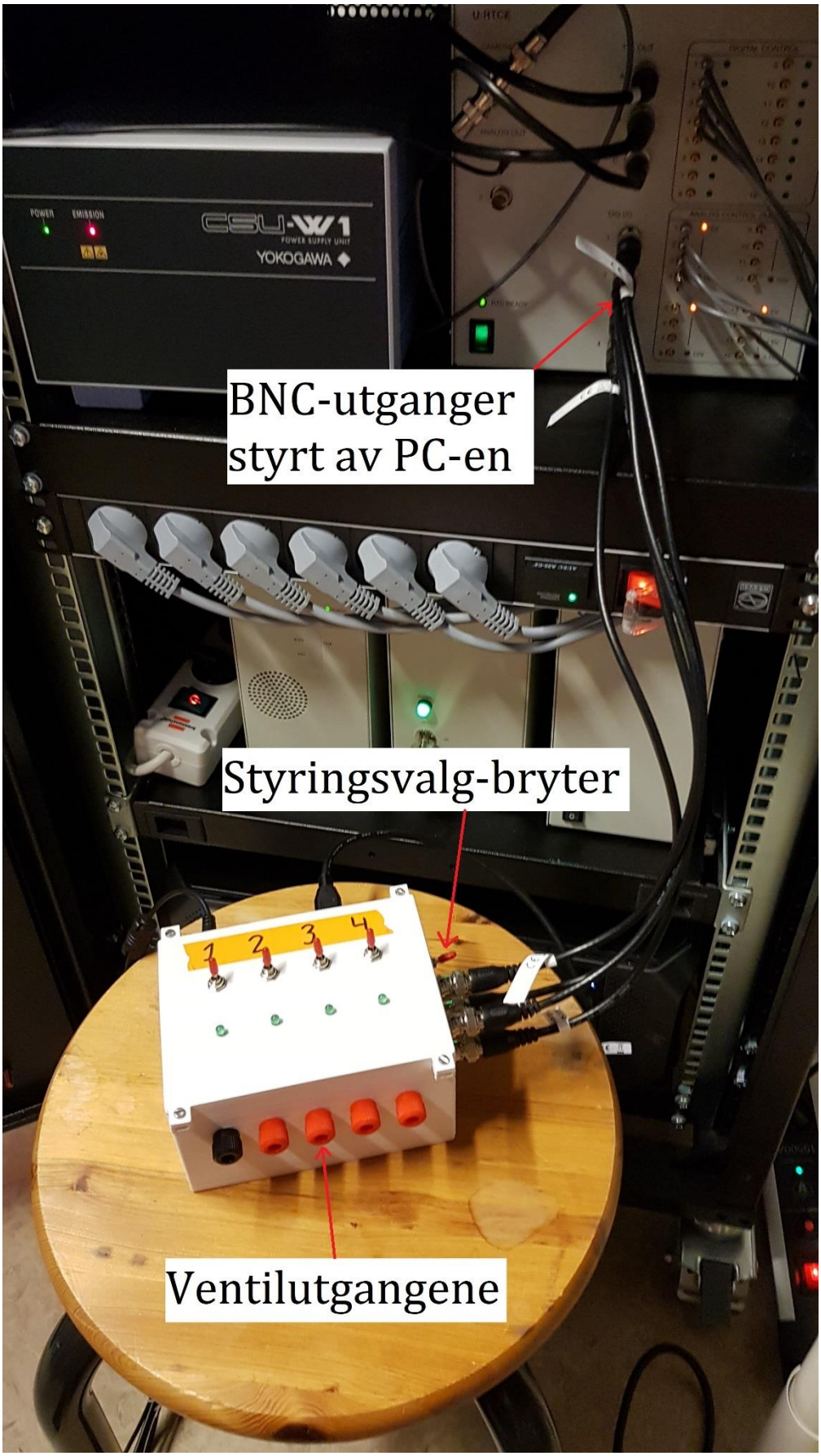
den som ble brukt til å skape magnetfeltet. Denne kortvarige spenningen gikk altså gjennom disse LED-lysene, og siden denne spenningen sannsynligvis er svært høy, tålte ikke LED-ene dette og ble etterhvert ødelagte. Vi måtte derfor installere noen tilbakeslagsdioder [6] for å stoppe denne spenningen fra å nå LED-lysene. Ved å ha en diode i parallell med ventilspolene i sperreretning vil tilbakeslagsspenningen bli kortsluttet da den kommer i motsatt retning. Strømmen vil da gå i ring slik som vist på figur 21 og energien går med til å varme opp dioden. De hadde noen dioder av riktig størrelse på Sars og vi kunne enkelt montere dem på ventilene. Etter å ha gjort det og loddet på nye LED-lys, virket alt som det skulle og vi kunne ikke lenger se noen blinking i de tre andre lysene, som betyr at tilbakeslagsspenningene hadde blitt ladet ut gjennom tilbakeslagsdiodene.



Figur 21: Tilbakeslagsspenning

6.5 Testing av BNC-styring på Sars

Etter å ha montert og loddet BNC-inngangene til systemet, utførte vi en test med mikroskopet som sender 5V-signalene. Ved dette mikroskopet står en PC som har et program hvor man kan programmere styringen av diverse utstyr som blir brukt under eksperimentene. For å utføre testen vår brukte vi dette programmet til å lage en sekvens som aktiverte de fire BNC-utgangene en etter en. Disse utgangene er koblet til systemet vårt, og når styringsvalg-bryteren på siden av innkapslingen står i BNC-kontrollmodus vil Arduinoen registrere disse. Da blir 5V-signalene sendt videre til reléene som aktiverer ventilene. Før vi testet med mikroskopet lasket vi 5V til BNC-inngangene for å verifisere at logikken virket. Testen med mikroskopet gikk helt etter planen og alt virket som det skulle.



Figur 22: Testing av BNC-styring

6.6 Testing av ferdig System på Sars

Det ble gjennomført en ukes test på Sars av oppdragsgiver hvor systemet ble brukt til eksperimentering. Denne gangen ble både hardware og software tatt i bruk og vi fikk tilbakemelding på hvordan ting hadde gått uken etter. Oppdragsgiver fortalte at det ikke hadde vært noen problemer og at alle funksjonene fungerte som forventet. Denne testen ble gjort i uke 18. Oppdragsgiver hadde noen forslag til nye funksjonaliteter i programmet, nemlig en timer på skjermen slik at en kan se hvor lenge et eksperiment har holdt på. Muligheten til å omorganisere rekkefølgen på action-er i en sekvens, det vil si å kunne flytte på hvilken rekkefølge ventilene skal åpne seg i når en konstruerer en sekvens. Siste ønsket var å ha muligheten til å lagre en spesifikk sekvens til senere bruk, slik at en kan ha ofte brukte sekvenser allerede lagret i programmet. Ettersom vi begynte å nærme oss slutten av prosjektet bestemte vi oss for å fokusere på å skrive rapporten og eventuelt legge til disse funksjonen senere dersom det ble tid. Klokken som viser tiden som en sekvens har gått og omorganisering av action-rekkefølgen fikk vi lagt inn, ettersom det var relativt enkelt å legge til. Lagring av sekvenser krever lengre tid å få til, og vi bestemte oss derfor for å prioritere rapporten.

7 Diskusjon

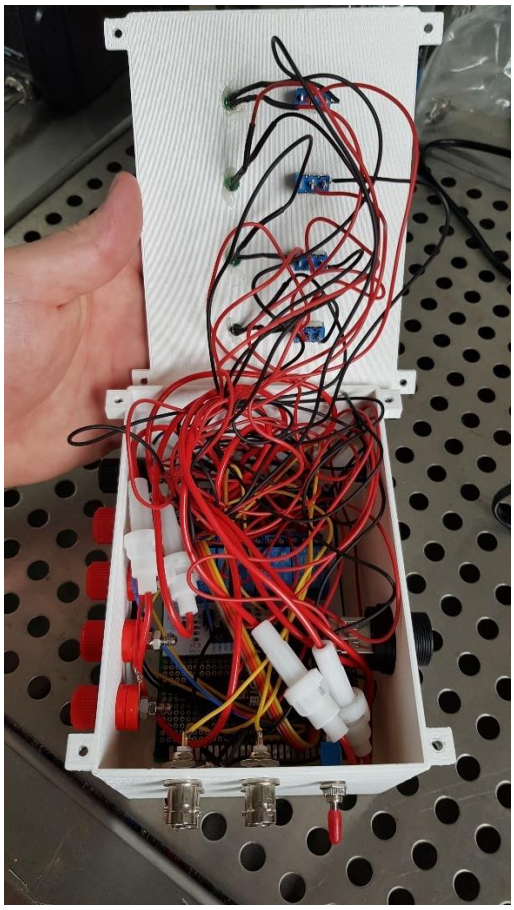
7.1 Forslag til forbedringer

7.1.1 Fjerning av bobler

Vår største mangel i systemet i forhold til de originale kravene er at systemet ikke har mulighet til å fjerne bobler i slangen. Vi vurderte en løsning som involverte pumper som kan pumpe væsken som inneholder boblene ut av slangen før en begynner med eksperimentet. Vi bestilte en slik pumpe for å teste den og etter den testen fant vi ut at pumper ikke kom til å være et alternativ for ventilene, grunnet vibrasjonen. Vi startet derfor å jobbe med andre funksjonaliteter og deler av oppgaven, og kravet om fjerning av bobler ble droppet til fordel for andre krav, som muligheten til å bruke systemet på et konfokalmikroskop ved hjelp av BNC-kabler.

7.1.2 Kabelhåndtering

Vi tenkte i utgangspunktet at det var lurt å bruke litt lange ledninger i boksen slik at det var mulig å ta av lokket og ta ut koblingsbrettet uten å måtte koble fra noen ledninger. Etter hvert som flere ledninger ble lagt til viste det seg at det ble svært dårlig plass i boksen til alle ledningene, og det ble ikke bedre av at vi ikke hadde samlet ledningene på noen fornuftig måte. Mange av ledningene går på kryss og tvers av hverandre inne i boksen og det gjør det vanskelig å se hvor en spesifikk ledning går. Forbedringen som kunne blitt gjort er da å samle ledninger som holder sammen med enten strips eller krympestrømpe. I tillegg hadde det vært optimalt om det var mulig å ta av lokket uten å benytte en lang ledning som tar mye plass og skaper rot. Dette kunne blitt oppnådd ved å lett kunne koble ledningene fra lokket uten å måtte skru eller avlodde noe. Dersom vi hadde festet en skjøt på de ledningene som går i lokket, ville det være mulig å separere lokket helt uten problemer. Her kunne vi brukt banankabelkontakter.



Figur 23: Innsiden av innkapslingen

7.1.3 Lagring av sekvens

Det var ønskelig å kunne lagre predefinert sekvens for senere bruk. Denne funksjonaliteten fikk vi ikke tid til å legge til i programmet. Det finnes allerede en funksjon som ligner. Når programmet lukkes, blir den forrige sekvensen lagret og den vil fortsatt være der når du starter programmet igjen. Samme med listen over hvilke væsker som er i hver av kanalen. Denne funksjonen er fortsatt en potensiell forbedring.

8 Konklusjon

I starten av prosjektet fokuserte vi hovedsakelig på programmeringsdelen for å gjøre programmet så bra som mulig. Vi mener at programmets kvalitet reflekterer arbeidet vi la i det tidlig i prosessen. Etter en endring i stilen til programmet tror vi at grensesnittet har blitt både brukervennlig og har et estetisk utseende. Etter at programmet begynte å bli ferdig, startet vi å planlegge den fysiske kretsen. Bestilling av deler var noe vi var noe vi prøvde å gjøre så fort som mulig slik at vi ikke måtte vente på dem for å jobbe. Vi prøvde å planlegge slik at vi fikk bestilt mest mulig på en gang, men det vist seg å ikke være så lett og vi endte opp med å bestille opptil flere ganger. Noen ombestillinger inkluderer, en USB-kabel som ikke fungerte, en dårlig løsning på en USB-inngang i boksen og mangler på muttere. Vi tror at prosjektet kunne ha fokusert mer på den fysiske delen av oppgaven, slik at den ble mer polert og god. Systemet fungerer som det skal, men det er muligheter for forbedringer som kunne blitt unngått med bedre planlegging, se 7.1.2.

Selv med noen mangler, er vi i gruppen fornøyde med resultatet på prosjektet og vi håper også at Sars er fornøyde med systemet. Vårt håp er at vårt system skal bli brukt i deres forskning og at vi har kunnet hjelpe dem med å løse et problem de hadde. Vi er også forberedt på å hjelpe Sars med å løse eventuelle problemer som kan oppstå med systemet selv etter at dette prosjektet er over.

Bibliografi

[1] «Sars senter for marin molekylærbiologi,» Universitet i Bergen, Januar 2021. [Internett]. Available: https://www.uib.no/sarssenteret. [Funnet 29 Januar 2021].

[2] Microsoft, «Microsoft documentation,» [Internett]. Available: https://docs.microsoft.com/en-us/dotnet/api/system?view=net-5.0. [Funnet Februar 2021].

[3] RS, «rs-online,» [Internett]. Available: https://no.rs-online.com/web/. [Funnet Mars 2021].

[4] Stackoverflow, «Stackoverflow,» [Internett]. Available: https://stackoverflow.com/. [Funnet Februar 2021].

[5] Høgskulen på Vestlandet, «HVL Open,» [Internett]. Available: https://hvlopen.brage.unit.no/hvlopen-xmlui/. [Funnet April 2021].

[6] Wikipedia, «Diode,» [Internett]. Available: https://en.wikipedia.org/wiki/Diode. [Funnet Mai 2021].

[7] Arduino, «Language Reference,» [Internett]. Available: https://www.arduino.cc/reference/en/. [Funnet Februar 2021].

[8] The FreeCAD Team, «Freecad,» [Internett]. Available: https://www.freecadweb.org/. [Funnet Mars 2021].

[9] DrVax, «youtube,» 18 August 2020. [Internett]. Available: https://www.youtube.com/watch?v=l4WzCjmuJg0&list=PLxa9m2nC6N924jFUOYRECQUMm9xl4_jUI&ab_channel=DrVaxDrVax. [Funnet Mars 2021].

[10] Kjell & Company, «kjell,» [Internett]. Available: https://www.kjell.com/no. [Funnet Mars 2021].

1. Forkortelser og ordforklaringer

Arduino	Populær og billig mikrokontroller
Arduino IDE	Program for å programmere mikrokontrolleren
C#	Programmeringsspråk utviklet av Microsoft
ELE102	Emnekode; programmering og mikrokontroller
ELE124	Emnekode; videregående programmering
Floating	Inngang som ikke er tilkoblet noe
FreeCad	Gratis 3D-tegningsprogram
GUI	Grafisk brukergrensesnitt
Hendelser	En hendelse i programmet som utløser en eller flere metodekall
Hullbrett	Kort med hull og kobberspor, ment til å lodde kretser på
HW	Hardware
H/L	High/Low signal
IDE	Integrated Development Environment
IEEE	Institute of Electrical and Electronics Engineers
I/0	Input/Output port
Konfokalmikroskop	Mikroskop med høy optisk oppløsning
MOSFET	En type transistor
MultiSim	Program for å tegne og simulere elektriske kretser
PWM	Pulsbreddemodulasjon
Python	Programmeringsspråk med åpen kildekode
RGB	Rød, grønn, blå

Sekvens	Åpning og lukking av kanaler i en bestemt rekkefølge
Seriell-buffer	Datalager som holder info mottatt av en serieport
StopWatch-objekt	Et objekt i C# som holder tiden svært nøyaktig
SW	Software
Visual Studio	IDE brukt for å programmere prosjektet i C#
Windows Forms	Verktøy for å utvikle GUI

2. Prosjektledelse og styring

Prosjektstarten var i januar 2021 etter første møte med oppdragsgiveren. Da ble det bestemt og planlagt en del av gjennomføring av prosjektet, slik som arbeidsted, oppmøte, og arbeidsfordeling.

Arbeidsted

På grunn av Covid-19-tiltak, ble det meste av prosjektet gjort på HVL sin campus. Arbeidet relatert med SW og HW ble gjennomført på skolen, der gruppen møtte opp og jobbet i vanlige gruppe- og klasserom.

En del testing og montering ble gjort på Sars-senteret, og en del som rapportskriving ble gjort hjemmefra.

Arbeidsfordeling

I utgangspunktet ble det bestemt at hver i gruppen skulle være ansvarlig for en subdel av programmet. Senere ble det bestemt å samarbeide og gjennomføre de fleste av oppgavene sammen.

Logging

Alt arbeidet på prosjektet ble registrert i en logg og timeliste. Det var brukt en felles Excel-fil der alle kunne komme inn og registrere arbeidet gjennomført.

Regnskap

Tabellen nedfor viser prosjektets budsjett og kostnader.

Beskrivelse	Antall	Pris (nok)	Totalt
Arduino Micro	1	228	228
Sikring	4	10	40
Sikringsholder	10	19.8	198
Miniatyrbryter	10	11.9	119
Strømforsyning	1	288	288
Svarte banankobliger hunn	10	9.9	99
Røde banankoblinger hunn	10	9.9	99
Svarte banankobliger hann	10	7.9	79
Røde banankoblinger hann	10	7.9	79
Forlengelseskabel USB	1	110	110
Mikro USB-kabel	1	80	80
Svarte bananhylse	1	25	25
Røde bananhylse	10	8	80
DC power-conector	1	45	45
Krympestrømper (pack)	1	140	140
Tilkoblingskabel USB	1	100	100
USB-conector	1	165	165
Bolter (pack)	1	22	22
Mutrer (pack)	1	22	22
Peristaltisk Pumpe	2	213	426
BNC-kabel	4	70	280
BNC-kabinetthunn	5	40	200
Tilkoblingskabel USB (male to male)	1	100	100
Relemodul	1	200	200
			3224

Tabell 1: Budsjett

Det ble avtalt at Sars skulle dekke alle prosjektets kostnader. I utgangspunktet hadde vi en høy margin, men vi prøvde å finne rimelige priser for komponentene slik at prosjektet skulle holde lave kostnader.

Alle kostnader er innkjøp av komponenter. Ikke alle komponentene som ble kjøpt ble brukt, grunnen for at det ble innkjøp av mange (10x) er kvantumsrabatt.

Fremdriftsplan

Noen av oppgavene var iterative og var derfor vanskelig å fastslå en endelig sluttdato på. Etter hvert som systemet ble mer ferdig, utførte vi nye tester hvor vi fikk tilbakemeldinger om forbedringer, endringer og ting som kunne legges til.

Se Gantt-diagram-vedlegg.

Risikoliste

Risiko	Sannsynlighet	Alvor	Konsekvens	Tiltak
Korona	Middels	Høy	Fare for liv og helse. Kan hindre prosjektets fremdrift.	Munnbind, avstand, antibac, digitale møter, arbeid over nett
Lodding, varme	Stor	Middels	Forbrenning	Stativ for å holde komponenter og stødig loddebolтholder
Lodding, øyeskade	Lav	Høy	Øyeskade, nedsatt syn	Vernebriller

Tabell 2: Risikoliste

3. Brukerdokumentasjon

Hyperfusion brukerveiledning

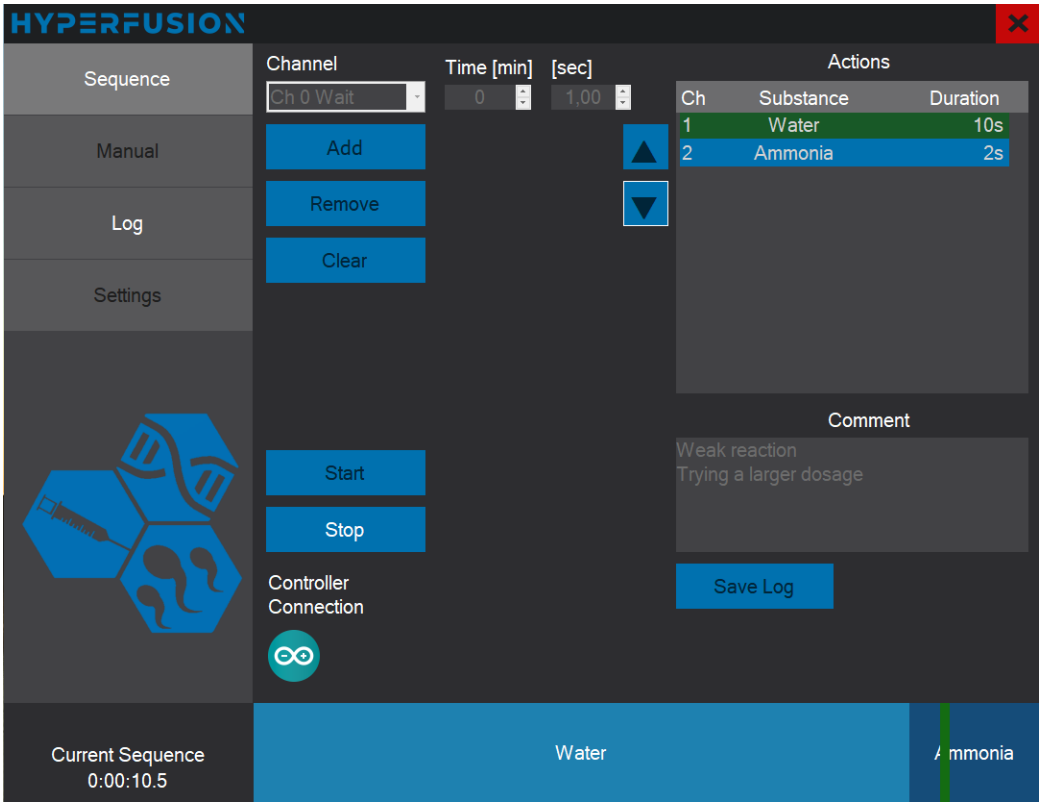
I alle fanene vil knappene ha hvit skrift dersom de er aktive og kan trykkes på, svart skrift indikerer at de er deaktiverte og ikke vil reagere.

Unntatt "Log"-fanen har alle fanene en "Controller Connection"-indikator nede til venstre som indikerer om kontrolleren er koblet til og at kommunikasjonen virker.

Figur 24: Tilkoblet/frakoblet

Exe-filen må ikke flyttes ut av hovedmappen til programmet hvor ulike nødvendige filer og mapper ligger. Man kan lage en snarvei til exe-filen som kan festes på skrivebordet eller der man måtte ønske.

Sequence



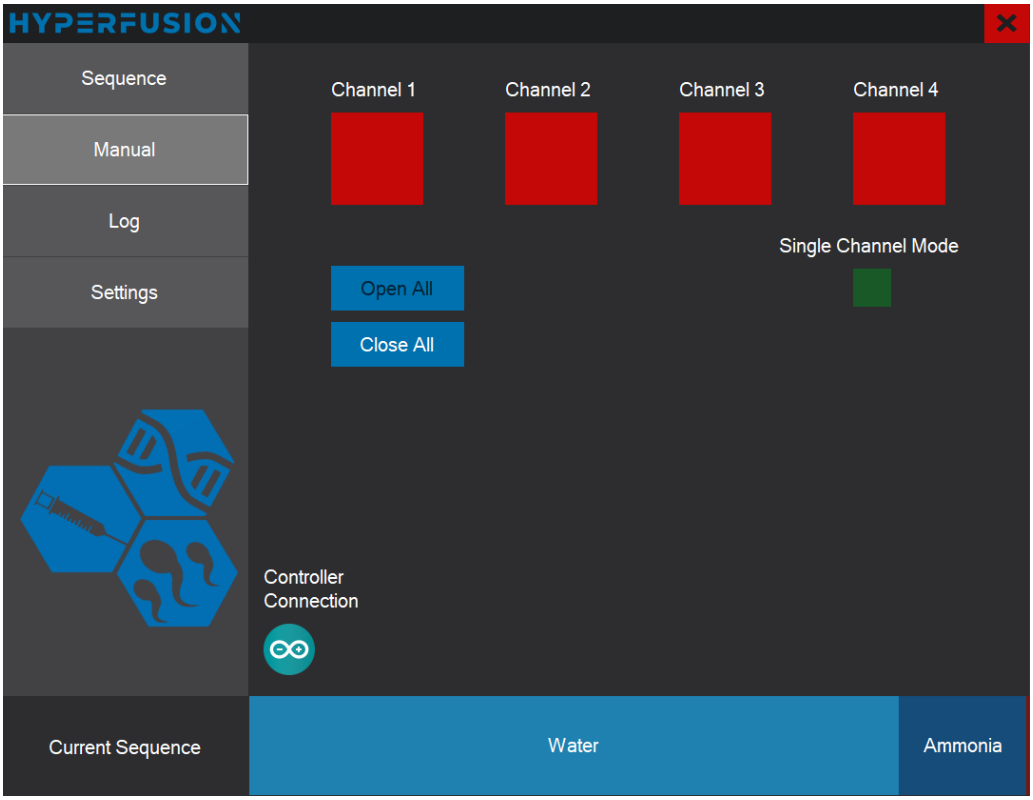
Figur 25: Sekvens under kjøring

Her kan man lage og kjøre sekvenser. For at man skal få opp alternativer i Channel-comboboxen må man først skrive inn navn til kanalene som er i bruk, dette kan man gjøre i "Settings"-fanen. Når man har valgt en kanal og stilt inn en varighet kan man trykke på "Add" for at det steget skal dukke opp i tabellen over "Actions". Dersom man ønsker å fjerne noe fra "Actions"-tabellen kan man klikke på det elementet og deretter trykke på "Remove". "Clear"-knappen fjerner alle elementene i hele tabellen.

De to opp og ned pilene til venstre for "Actions"-tabellen kan brukes til å flytte et valgt steg i sekvensen. "Start"-knappen starter sekvensen, og den kan da ikke endres før den er ferdig eller blir stoppet. Kontrolleren må være koblet til for å kunne starte sekvensen. Når en sekvens har blitt kjørt får man muligheten til å trykke på "Save Log", og det blir da lagret en loggfil som man kan hente frem og lese på "Log"-fanen.

Under kjøring av en sekvens vil fremdriften være indikert av farge i "Actions"-tabellen hvor blå indikerer "nåværende", grønn indikerer "fullført" og rød indikerer "stoppet" eller "feil". I tillegg vil fremdriften være indikert av en fremdriftsbar på bunnen av vinduet som viser de ulike stegene i sekvensen. Denne baren kan man se uansett hvilken fane man er i.

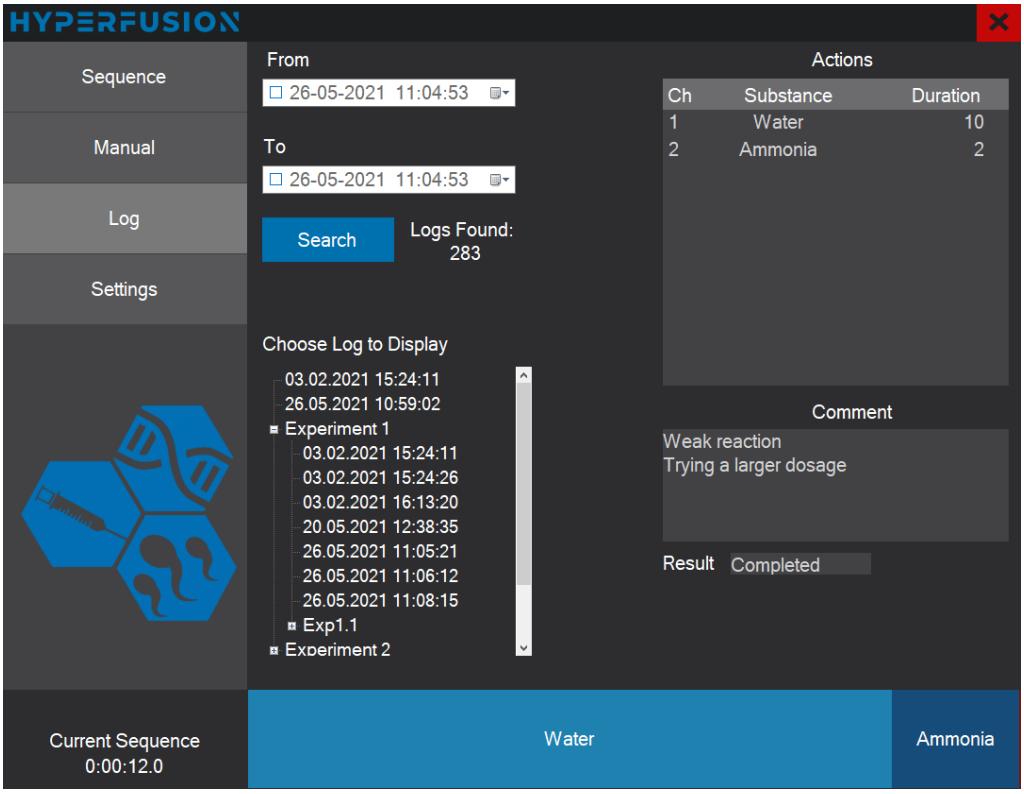
Manual



Figur 26: Manual-tab

I denne fanen kan man åpne og lukke kanalene manuelt ved å trykke på de ulike "Channel"-boksene. Rød boks indikerer lukket kanal, grønn boks indikerer åpen kanal. "Open All"-knappen åpner alle kanalene, og "Close All"-knappen lukker alle kanalene. "Single Channel Mode" alternativet gjør at kun en kanal kan være åpen om gangen. Når dette alternativet blir aktivert lukkes alle kanaler.

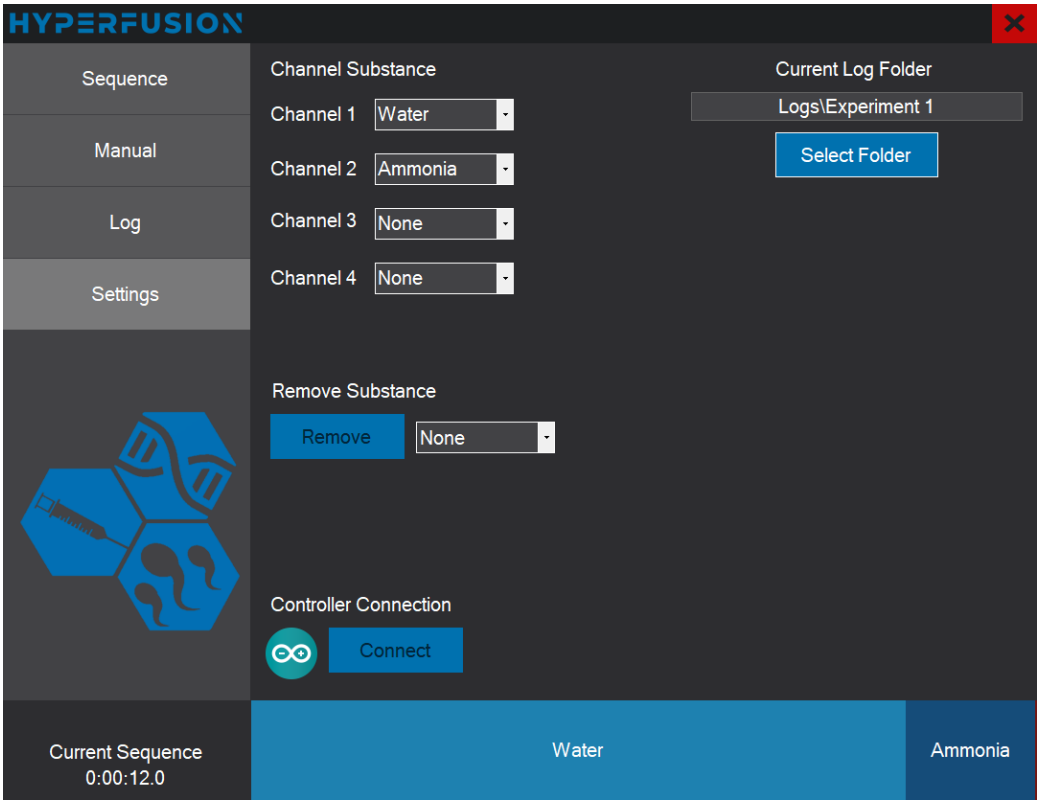
Log



Figur 27: Visning av loggfil

I denne fanen kan man søke frem og vise logger. Alle loggene er navngitt med dato. For å søke mellom to datoer må man krysse av på både "From" og "To" datoene og velge tidsrommet man vil søke i. Alternativt kan man søke i alle logger som er lagret etter en dato ved å kun krysse av "From". På samme måte kan man søke i alle logger som er lagret før en dato ved å kun krysse av "To". Dersom man vil søke i alle loggene krysser man ikke av noen av datoene. "Search"-knappen vil oppdatere "Choose Log to Display"-listen med de nye søkekriteriene. I den listen kan man se mappestrukturen loggene er lagret i. For å vise en logg trykker man på en logg i listen og da vil sekvensen, kommentaren og resultatet bli vist.

Settings



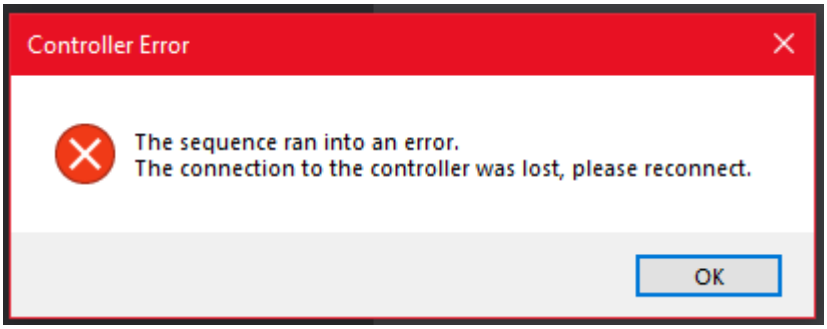
Figur 28: Settings-tab

I denne fanen kan man navngi kanalene ved hjelp av de ulike "Channel x"-comboboxene. Tidligere navn vil bli husket og man kan velge disse som et "drop-down" alternativ uten å skrive dem inn på nytt. Kanaler som ikke har navn, dvs. "None", vil ikke bli vist i "Channel"-comboboxen på "Sequence"-fanen. For å fjerne et navn kan man velge det i "Remove Substance"-comboboxen og trykke "Remove".

Oppe til høyre kan man velge hvor loggfilene skal lagres ved å trykke "Select Folder". Programmet krever at de lagres i "Logs"-mappen. Den valgte filstien vises i tekstboksen øverst til høyre i fanen.

Programmet forsøker å koble til kontrolleren under oppstart, men dersom det ikke skulle gå kan man forsøke å koble til ved å trykke på "Connect"-knappen nederst til venstre i fanen.

Feilmeldinger



Figur 29: Eksempel på feilmelding

Sequence-feilmeldinger:

The sequence ran into an error.

The controller was unable to execute the given command, please restart it. (1)

The sequence ran into an error.

The connection to the controller was lost, please reconnect. (2)

The sequence ran into an error.

The controller is connected but unresponsive, please check that the control mode switch is in the correct position. If so, try restarting the controller. (3)

Manual-feilmelding:

Failed to open/close the channel (4)

Forklaring til feilmeldinger:

(1) Feil i seriell kommunikasjon eller i programmet resulterte i at Arduino har fått en ugyldig melding. Det er anbefalt å restarte Arduinoen eller laste ned Arduino programmet igjen. Hvis ikke det hjelper burde en kontakte oss for feilsøking.

(2) Mulig avkobling av Arduinoen og PC-en. Sjekk USB-kabelen.

(3) Arduinoen er tilkoblet men svarer ikke. Dette skjer dersom kontrollmodus-bryteren på siden av innkapslingen er i BNC-styringsmodus. Alternativt kan det være på grunn av at programmet i Arduinoen har hengt seg opp eller er krasjet. Restart Arduinoen.

(4) Kommunikajsonsproblem. Sjekk USB-kabel og restart Arduino.

Drifts- og vedlikeholdsdokumentasjon

For å bruke systemet kobler man til 24V-strømforsyningen og så mange utganger man trenger til ventilene. En viktig ting å huske er å alltid koble til felles 0V, som er den svarte banankoblingen. Når dette er gjort kan bryterne benyttes til å styre ventilene. For PC-styring må man koble til USB-kabelen og passe på at kontrollmodus-bryteren på siden av innkapslingen står i PC-innstillingen. Da kan Hyperfusion-programmet brukes til å styre ventilene. Stiller man kontrollmodus-bryteren i BNC-innstillingen vil ikke styring fra PC-en virke, men man kan da styre ventilene med 5V-signaler på BNC-inngangene.