BACHELOROPPGAVE:

BO19E-05 AUTOMATISERING AV KVALITETSKONTROLL AV ALGER

Anders Kommedal

Audun Møgster

Martin Græsdal

31. mai. 2019

Dokumentkontroll

Rapportens tittel: BO19E-05 Automatisering av kvalitetskontroll av alger	Dato/Versjon 31. mai. 2019/10
	Rapportnummer: B019E-05
Forfatter(e): **Anders Kommedal:** 181383@stud.hvl.no **Audun Møgster:** 151215@stud.hvl.no **Martin Græsdal:** 151227@stud.hvl.no	Studieretning: 16HEAU
	Antall sider m/vedlegg 72
Høgskolens veileder: Svein Haustveit	Gradering: Åpen
Eventuelle Merknader: Vi tillater at oppgaven kan publiseres.	

Oppdragsgiver: Sars internasjonale senter for marin molekylærbiologi	Oppdragsgivers referanse: Jørgen Høyer
Oppdragsgivers kontaktperson(er) (inklusiv kontaktinformasjon): Jørgen Høyer – 41636030, jorgen.hoyer@uib.no	

Revisjon	Dato	Status	Utført av
0.10	31.01.19	Første utkast	Anders Kommedal, Audun Møgster, Martin Græsdal
0.15	07.03.19	Lagt til løsningsalternativ 3, slik vi ser det nødvendig for å få løst oppgaven i henhold til kravspesifikasjoner.	Anders Kommedal
0.20	27.05.19	Ryddet, fikset kildene	Anders Kommedal, Audun Møgster, Martin Græsdal
0.25	30.05.19	Ferdigstilling av bachelorrapport	Anders Kommedal, Audun Møgster, Martin Græsdal
1.0	31.05.19	Bachelorrapport ferdig	Anders Kommedal, Audun Møgster, Martin Græsdal

Forord

Denne rapporten er skrevet som en avsluttende del av bachelorstudiet i Automatiseringsteknikk ved Høgskulen på Vestlandet, avdeling Bergen.

Prosjektet har bydd på flere utfordringer, som har krevd faglig kunnskap innen blant annet programmering, analog- og digitalteknikk og generell automatiseringsteknikk. Vi har benyttet ulike verktøy og programmer vi har lært å bruke gjennom vårt studie, men vi har også måtte sette oss inn i mye nytt. Gjennom prosjektet har vi opparbeidet oss erfaring i hvordan en planlegger og gjennomfører langvarige prosjekter. Det har vært en lærerik og spennende prosess, hvor vi har fått større erfaring i å samarbeide med en oppdragsgiver.

Vi vil takke blant annet Jørgen Høyer, avdelingsingeniør på Sars, for god og åpen kommunikasjon gjennom hele perioden. Det har vært et godt og konstruktivt samarbeid, hvor begge parter har fått komme med innspill for å forbedre sluttproduktet. Vi ønsker også å takke gruppeleder for gruppe S13 ved Sars, Marios Chatzigeorgiou, for at vi fikk romplass på hans kontor under hele semesteret. Forutenom kontorplassen, har Chatzigeorgiou, også vært til god hjelp med å forstå hva de faktisk forsker på. En annen ting vi er takknemlige for, er måten de ansatte på Sars har tatt oss imot. Det å ha vært en del av et slikt forskningsmiljø, har vært en svært positiv opplevelse.

Vi vil også rette en takk til veilederen vår Svein Haustveit og faglærer Adis Hodzic ved Høgskolen i Bergen, som begge har gitt gode tilbakemeldinger underveis i prosjektet. Haustveit har bistått med gode råd til hva som er lurt å ha med i en rapport samt hvilke formelle krav som gjelder, mens Hodzic har hjulpet oss med programmering.

Sammendrag

Ved Sars Internasjonale senter for molekylærbiologi studeres larvene til en type sekkedyr. Disse sekkedyrene mates av fire forskjellige typer alger. Med hjelp av programvaren vi har designet vil de nå få ut et estimat på hvor mye alger som brukes til å mate disse sekkedyrene. Ved å kunne måle lysforhold, temperatur, PH og turbiditet kan forskerne også få en bedre innsikt i levevilkårene til algene.

Vi har laget et fysisk system, som består av fire sensorer som måler hver av disse verdiene. Det har også blitt laget et 3D-printet skall som alle sensorene får plass i, samt deres sensorkort og ledere. Vi har tatt i bruk en mikrokontroller for bruk som klient og en Raspberry pi som server.

Serverprogrammet er designet for å kunne ta imot flere klienter, og å kunne vise deres tilhørende sensorverdier. Serverprogrammet lagrer også data, som kan søkes opp for historikk ved måling over tid. Klientprogrammet er laget slik at det leser de ulike sensorverdiene, og sender det med trådløs kommunikasjon til serveren.

Systemet er enkelt å replikere da delene er rimelige, og en enkelt kan printe ut nytt skall til nytt produkt. Skallet kan enkelt plukkes fra hverandre, dersom det er nødvendig med tilgang til et av sensorkortene eller sensorene.

Noen forenklinger har vært nødvendige for å komme oss i mål, men oppgaven er løst i henhold til kravspesifikasjonene som er satt opp med oppdragsgiver.

1 Innhold

Figurliste

2 Innledning

2.1 Organisering av rapporten

Ord som krever en utdyping eller forklaring er markert med *. Disse vil en finne forklaring på under Appendiks B . For kildehenvisning er det brukt IEEE*, som i teksten er markert med [X], hvor X tilsvarer nummeret på kilden.

2.2 Oppdragsgiver

Figur 1 - Logo Sars

Oppdragsgiveren for denne oppgaven er Jørgen Høyer, fra Sars internasjonale senter for marin molekylærbiologi. Sars er et forskningssenter stasjonert på Marineholmen, og er en del av Universitetet i Bergen. Senterets formål er å bruke sjødyr som modell for å studere evolusjonær utvikling, ved hjelp av molekylære teknikker [1] . Senteret ble etablert i 1997, etter å ha blitt finansiert av Norges forskningsråd (NFR). Siden 2003 har Sars vært i samarbeid med det Europeiske Molekylærbiologiske Laboratoriet (EMBL). Senteret er oppkalt etter de to Bergenske marinbiologene Michael Sars og sønnen Georg Ossian Sars [2].

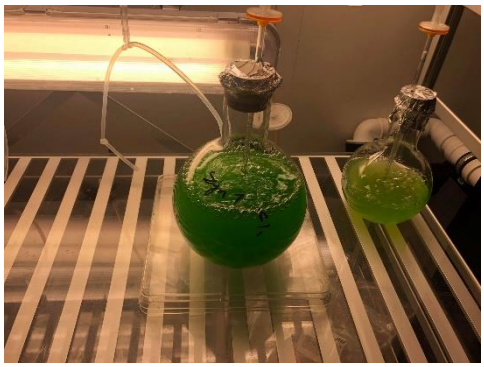
I dag er rundt 60 personer ansatt hos Sars, med over 20 forskjellige nasjonaliteter. Her jobbes det i åtte forskningsgrupper, som ledes av en gruppeleder som er ansatt på seksårskontrakt. Gruppene består av alt fra masterstudenter til folk med doktorgrad. I tillegg har gruppene folk som besitter både teknisk- og administrativ kompetanse. Det er også til enhver tid flere studenter og forskere på senteret som er på lån fra andre institusjoner [3]. Det er forskningsgruppe S13 som har fremstilt denne oppgaven for oss, med gruppeleder Marios Chatzigeorgiou.

3 Problemstilling

Ved Sars forskes det på larvene til et sekkedyr kalt Ciona. Dyrene blir hentet opp fra havet utenfor Sotra, og plassert i kar på laboratoriet. Her blir de tatt vare på frem til de produserer egg og sædceller, som forskerne bruker for å produsere larver. Mens dyrene ligge i karet, blir de matet med alger som kultiveres i separate beholdere. Det er disse algene vår oppgave omhandler.

Per dags dato er det ingen overvåking av algene. Forskerne beregner antall alger i beholderen etter erfaring og skjønn. Vår oppgave blir å overvåke algene, slik at forskerne har full kontroll på hvor mye alger som finnes i beholderne, og hvor mye mat de gir til dyrene. I tillegg til antall alger i beholderen, skal vi også måle temperatur, pH og lysstyrke. Det er totalt fire ulike typer alger som skal måles, disse kalles CC, Syn, Rhino og Iso.

Figur 2 - Sekkedyrene



Figur 3 - Alger i kolbe

3.1 Hovedidé for løsningsforslag

Oppdragsgiver ønsket et automatisert system for kvalitetskontroll av alger. Ønsket var å gå fra en relativt usikker løsning, hvor en baserer seg på farge og volum til å vurdere hvor mye alger som er i en kolbe, til en mer nøyaktig og presis automasjonsløsning. Algene dyrkes frem over tid, og må etter hvert tynnes ut med saltvann. Ved en mer presis måling, vil forskeren kunne vite mer om kulturen og dermed har lettere for å vite når en slik uttynning bør skje. Dette kan føre til bedre kontroll av matmengden til dyrene, og bedre vekstmulighet for algene. Ønsket var da en ferdig prototype, med både HW og SW. Det ble diskutert en mulig løsning med et ferdig kar med de ulike sensorene montert på, hvor en kan helle en del av kulturen oppi for å få en presis og kontinuerlig måling, samt en potensiell regulering. I tillegg ble det foreslått en form for grafisk monitorering, enten via nettside eller app. Det var også ønskelig å kunne se endring i algetetthet over tid.

4 Kravspesifikasjon

Kravspesifikasjonene er satt opp av oppdragsgiver, i samarbeid med oss. Vi har hatt flere møter underveis, hvor vi har fått utbedret og spesifisert kravspesifikasjonene nøyere. Kravspesifikasjonene deles inn i to deler; det fysiske systemet og programvaren.

4.1 Det fysiske systemet

- Raspberry pi skal brukes som server.
- Mikrokontroller av typen NodeMCU ESP8266 skal brukes som klient.
- Kolben/tanken som brukes skal være av begrenset størrelse.
- Kolben/tanken må være transparent.
- Kolben/tanken må ha mulighet for å autoklaveres*.
- Systemet skal ha en sensor for måling av lys, pH, temperatur og turbiditet.

Raspberry pien er ønskelig da de som skal bruke systemet har tidligere erfaring med nettopp denne. Sensorene og mikrokontrolleren er valgt ut på forhånd, basert på pris og utskiftbarhet. Det skal være både rimelig og enkelt å erstatte ødelagte komponenter, eller å duplisere systemet. Ved å ha kolben i en begrenset størrelse, vil det ikke være nødvendig å helle over en veldig stor mengde algekultur over i den for å utføre målinger. Lys er en veldig viktig faktor for algenes vekst og trivsel, derfor vil det nødvendig å ha en transparent kolbe. I tillegg er det fortsatt ønskelig å kunne se fargen på algene. Den må også kunne steriliseres i deres autoklaveringsmaskin.

4.2 Programvaren

- Programkoden skal være på engelsk.
- Brukergrensesnittet skal være på engelsk.
- Programmet skal være mottakelig for å legge til flere klienter til en server.
- Det skal være en grafisk fremstilling av alle sensordataene.
- Det skal gjøres kontinuerlige målinger.
- Dataene skal lagres.

Da det jobber folk med ulike nasjonaliteter på Sars, er det ønskelig med engelskspråklig programkode og brukergrensesnitt. Dersom de ønsker å duplisere systemet, skal også programkoden tillate dette. Det skal være enkelt å finne lese av sanntidsdata, men det skal også være mulig å søke opp historiske måledata for spesifikke tidspunkt.

5 Analyse av problemet

Den ene komponenten vi brukte var en Raspberry pi. Vi hadde ikke så mye erfaring med denne komponenten så det kunne bli en utfordring å bruke denne. I tillegg skulle Raspberry pi og mikrokontrolleren ESP8266 fungere som et server/klient oppsett. Forslaget fra arbeidsgiveren var å bruke en kommunikasjonsprotokoll kalt MQTT*, siden den er mye brukt og er enkel å sette opp. Alternativet til denne protokollen var å bruke TCP/IP* (Socket) sammen med C#*. For å kunne konfigurere og sette opp Raspberry pi med mikrokontrollere krevde det at man bruker terminalen i operativsystemet Rasbien. For å kunne sette sammen og få mikrokontrolleren med de forskjellige sensorene til å fungere, måtte vi først finne de nødvendige instruksene som skal legges inn i terminalen.

Programmeringsdelen kunne bli en utfordring med tanke på hvilket språk vi skulle bruke til å programmere grensesnittet (GUI*). Enten måtte vi bruke Python*, Visual studio og lage programmet helt selv, eller så kunne vi bruke et ferdigprodusert program. For å kunne bruke Python må en bruke et verktøy kalt Tkinter for å lage GUI. Utfordringen her var at vi ikke hadde noe erfaring med dette verktøyet. Visual Studio hadde vi erfaring med, i tillegg var programmeringsmetoden særdeles mer brukervennlig enn Tkinter sin. Hvis vi skulle lage GUI-en selv kunne det bli en større utfordring å sette opp programmet via MQTT protokollen. Dette kunne man slippe hvis man har et ferdigprodusert program som kommuniserer med mikrokrontollere via MQTT protokollen.

En annen mulig utfordring var det fysiske målesystemet med tanke på hvordan sensorene skulle plasseres. Sensorene har forskjellige størrelser så vi måtte tilpasse og justere systemet etter dem. Selve kolben/tanken kunne bestå av valgfritt materiale så lenge den var transparent. I tillegg til dette skulle kolben/tanken være steril, så dette kunne bli problematisk når man kanskje måtte lage hull for å feste sensorene.

5.1 Utforming av mulige løsninger

Løsningen vår på oppgaven var å måle PH, temperatur, lys og turbiditet i en kolbe/tank bestående av sjøvann med alger. Målingene skulle sendes til en server (Raspberry Pi) gjennom en nettverksprotokoll for så å kunne monitorer og lagres i en database.

Oppdragsgiveren vår bestilte inn de fleste komponentene vi trengte på forhånd, dermed var det bare for oss å sette oss inn i virkemåtene til disse. Hovedfokuset i oppgaven ble dermed å bygge en løsning som fikk de forskjellige komponentene til å kommunisere med hverandre. I tillegg måtte vi velge en løsning på hvordan vi ønsket å bygge programmet. Som nevnt tidligere hadde vi lite erfaring med Raspberry pi og hvordan man setter opp et trådløst system med denne, og andre mikrokontrollere. Siden det fantes gode opplæringsressurser på internett så ville vi ta nytte av disse. Det samme gjaldt for GUI løsningen.

Vi hadde allerede det meste av nødvendig programvare, men måtte anskaffe enkelte programmer for overføring av filer og lignende fra Windows til Raspberry pi. I forhold til arbeidsrutiner inngikk vi en avtale om å være på Sars to til tre ganger i uken, og ellers arbeide på høgskolen om nødvendig. Vi satte opp mandag, onsdag og torsdag som faste dager til å jobbe med oppgaven. For oss var det ønskelig med kontinuitet i arbeidet, dermed passet dette oss bra.

5.1.1 Løsningsalternativ 1

Å bruke Python og raspberry pi. Python er integrert i Raspberry pi. Hvis vi hadde brukt dette språket til å lage GUI-løsningen måtte vi først lære oss det. I tillegg bruker Python et verktøy som heter TKinter for utvikling av GUI, der man må skrive linjekode, noe som var mindre brukervennlig for oss per dags dato.

5.1.2 Løsningsalternativ 2

Raspberry pi er kompatibel med C#, et språk vi er kjent med. Da kunne vi lage GUIen der og deretter implementere programmet i Raspberry Pi. I tillegg til dette måtte en kunne bruke enten kommunikasjonsprotokollene Socket eller MQTT. Socket hadde vi erfaring med og valgte derfor denne protokollen. Dette gjaldt for alle løsningsalternativene. Selve GUIen, og eventuelle andre kommunikasjonsprogram kunne vi lage i Visual studio, og overføre til Raspberry pi via et program som heter WinSCP. Denne lot oss overføre .exe filene via nettverk.

5.1.3 Løsningsalternativ 3

Et alternativ var å bruke Raspberry pi med Windows 10 IoT-core*. Fordelen med dette var at samspillet mellom Windows og Raspberry pien fungerte bedre enn dersom Raspberryen brukte Raspbian. Blant annet åpnet det for å kunne bruke andre biblioteker enn det som fungerte ved å overføre en Windows forms .exe-fil direkte til Raspberryen. Dette var en fordel hvis vi skulle lage det grafiske brukergrensesnittet, for da kunne vi velge blant nyere bibliotek for grafer og monitorering. Vi kunne fortsatt programmere i Visual studio, men heller gå over til UWP* istedenfor Windows forms. Ønsket var også å benytte seg av sockets for kommunikasjon, og C# som programmeringsspråk.

5.1.4 Vurderinger i forhold til verktøy og HW/SW komponenter

Til tross for at de fleste nødvendige komponentene var bestilt inn på forhånd, var det flere valg som måtte tas. Blant annet måtte vi ta stilling til hvilket programmeringsspråk som skal brukes i forbindelse med oppgaven. Gruppens kompetanse var definitivt tyngst innen C# og C++, noe som gjorde dette til foretrukne språk for oss. Her hadde vi også erfaring innen GUI, og dersom vi holdt oss til tidsplanen burde det være mulig å lage et solid program til dette. Samtidig visste vi at Python ville være et godt program å bruke til Raspberry Pi, og at det fantes mye ferdig kode som ville være til hjelp dersom vi kom i tidsklemma. Det hadde også vert en god erfaring for gruppen med tanke på å lære mer om programmeringsspråket.

Raspberry Pi og ESP8266 ble ansett som krav i denne oppgaven, og ble derfor tatt i bruk. Fordelen med ESP8266 var at denne bygger på samme innganger/utganger som en arduino, som vi allerede hadde kjennskap til. En utfordring ville bli å få Raspberry Pien og mikrokontrolleren til å kommunisere trådløst, da arduino har sitt eget C-basert språk. Enkelte av sensorene kunne også ta tid å få opp å gå. Flere av sensorene måtte kalibreres, og vi måtte finne programkode som gjorde det mulig for oss å samle inn data.

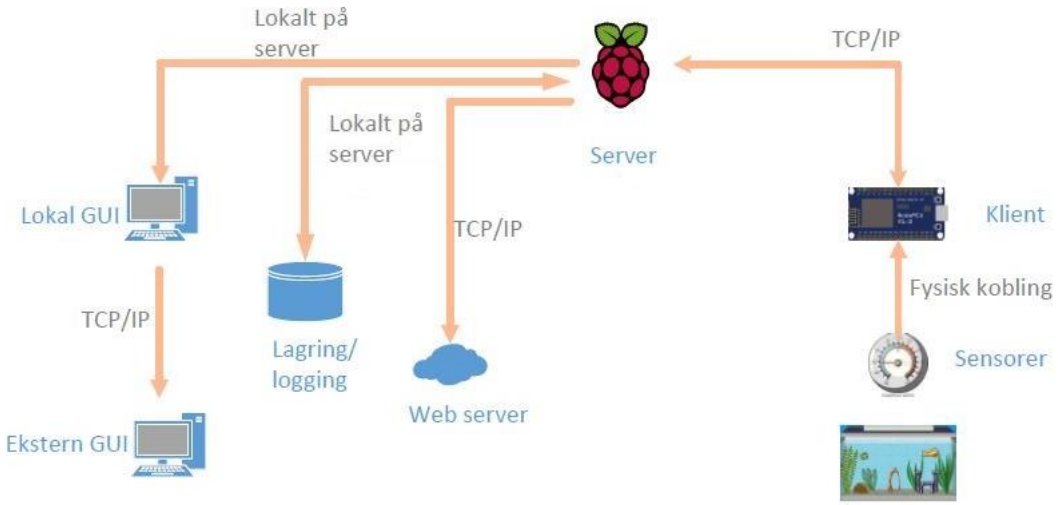
Det var også vanskelig å få begynt på utforming av selve kolben før vi hadde fått alt av utstyret. For det første måtte vi finne passende kolbe som måtte bestilles, for så å se på ulike løsninger for å montere sensorene til denne. Kolbene måtte spesialbestilles fra leverandører som oppdragsgiver brukte, siden kolben var nødt til å kunne autoklaveres. Vi så for oss en løsning med et lokk til en åpen kolbe, der alle sensorene hang ned fra bunnen av dette lokket. Lokket kunne lages ved hjelp av Sars' egen 3D-printer.

5.2 Konklusjon

Etter å ha overveid de ulike løsningene, bestemte vi oss først for å gå for løsningsalternativ 2. Dette fordi det dekket mye av vår kompetanse, og at vi så det som en gjennomførbar oppgave.

Som beskrevet i løsningsalternativ 2, er det mulig å bruke rene Windows forms programmer med Raspberry pi, bare med å overføre .exe-filen via WinSCP. Vi fant derimot tidlig ut at dette var en svak løsning for hvordan vi ønsket å utforme GUI-en. Da Raspberry pien er linuxbasert og bruker i utgangspunktet en mikroprosessor med ARM arkitektur. For oss betydde det at flere biblioteker, sett borti fra .NET, som er laget for Windows forms og x86 arkitekturen ikke fungerte når man overførte .exe-filen. Dermed ble løsningsalternativ 3 introdusert. Dette viste seg å være en god og fleksibel løsning, hvor vi benyttet oss av vår tidligere opparbeidet kunnskap, men samtidig hadde flere utfordringer vi også jobbet med. En av disse utfordringene var blant annet å programmere i UWP. Det var også fortsatt en del utfordringer med å sette opp og programmere klient-delen slik at den ble kompatibel med server og sensorer. For å holde en god struktur og oversikt på prosjektet vårt, delte vi det inn i forskjellige seksjoner;

- Design og bygging av lokk
- Målekrets med sensorer
- Klient og server
- Server, lagring/logging og eksterne monitorer som Web server og ekstern GUI



Figur 4 - Info-graf av produktet

6 Realisering av valgt løsning

6.1 Komponenter

De fleste komponentene var bestilt på forhånd av oppdragsgiver. Fellesbetegnelsen for komponentene er at de er rimelige, og er lett å erstatte. Dette skal gi fleksibilitet til både å duplisere produktet, og å erstatte defekte enkeltdeler.

6.1.1 Mikrokontroller

Vår mikrokontroller er en ESP8266 NodeMCU. Denne kan brukes med flere ulike IDEer, men siden vi har god erfaring med Arduino IDE, har vi valgt å bruke denne. Mikrokontrolleren kan driftes på ulike måter, men vi har valgt å drifte den ved å bruke 5V direkte inn på VIN. Dette har vi gjort for å spare plass, da den skal ligge inne i lokket. Kontrolleren kan gi ut to ulike spenninger, både 5V og 3,3V. Den har 9 digitale inn- eller utganger, og en analog inn- eller utgang. Da denne kun har en analog inn- eller utgang har vi lagt til en multiplekser i kretsen, da både PH- sensoren og turbiditetssensoren bruker analog inngang. Fordelene med denne mikrokontrolleren er blant annet at den har en innebygget WiFi-modul, og at den er svært rimelig. For oppsett/bruk se Appendiks E.1 .

Figur 5 - NodeMCU ESP8266 mikrokontroller. [37]

6.1.2 Turbiditetssensor (SEN0189)

Turbiditetssensoren har en output på 0-4,5V og måler mellom 0 og 3000 NTU*, noe som gir et mål på klarheten i vann [4]. Vanlig drikkevann bør normalt sett ligge på under 1 NTU, mens noen renseanlegg streber etter en NTU på 0,1. Et problem med denne sensoren er at den ikke er vanntett.

Måleprinsippet til turbiditetssensoren går ut på å sende et lys mellom to plater. Hvor mye lys som blir mottatt på andre siden blir da en indikator på hvor mange partikler som er i vannet. Nærmere forklaring finnes under Appendiks 0.

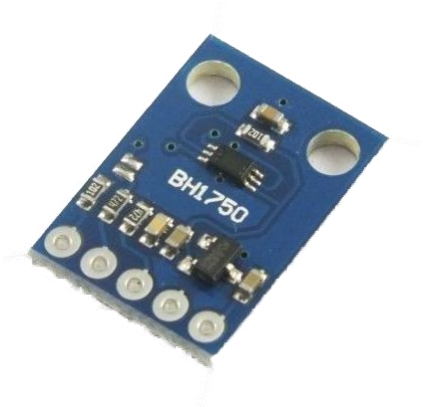
Figur 6 – Turbiditetssensor [38]

6.1.3 pH-sensor (SEN0161)

pH-sensoren har en output på 0-5V og måler en pH mellom 0-14 [5]. Væsker med pH <7 er syrlige, pH = 7 er nøytral og pH >7 er basisk.

Måleprinsippet til pH-sensoren går ut på å sette en liten spenning på et gelelag på tuppen av proben [6]. Dette gjør at det tiltrekkes hydrogenioner i væsken. Antall hydrogenioner som samles på innsiden av proben i forhold til utsiden blir da et mål på pH-verdien til væsken. Forklaring av kalibrering og bruk av sensoren kan finnes under Appendiks E.3 .

Figur 7 - pH-sensor [35]



Figur 8 – Lyssensor [36]

6.1.4 Lyssensor (BH1750)

Lyssensoren driftes på 3.3V, og kommuniserer ved hjelp av I^2C protokoll, som er en master/slaveprotokoll. Med andre ord er også denne sensoren digital. Sensoren måler lys mellom 0-65535 lux [7]. Selv om den kan måle lys i rundt 65klux, er ikke det en lysstyrke i nærheten av det som er aktuelt for oss. Vanlig lys i et rom ligger normalt sett på rundt 400lux. For bruk, og kodesnutt, se Appendiks 0.

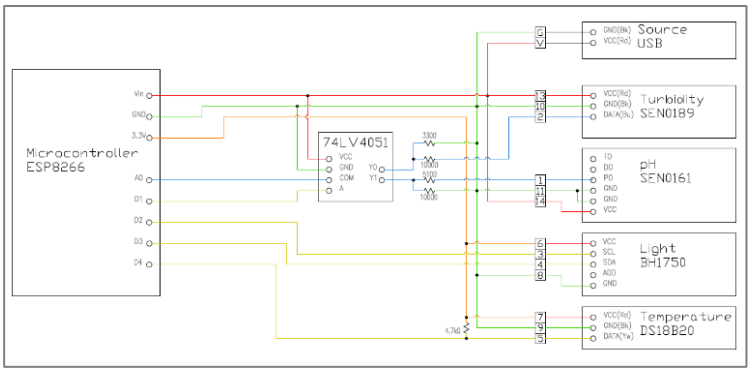
6.1.5 Temperatursensor (DS18B20)

Temperatursensoren kan drives på både 3V og 5V, og har et måleområde mellom -55 - 125°C [8]. Den bør riktignok ikke brukes over 100°C på grunn av PVC-kabelen*, men dette vil ikke være aktuelt for oss. Sensoren har en innebygd krets kalt DS18B20 som gjør om den målte temperaturen til et digitalt signal. Temperaturen blir sendt som et pulstog som representerer temperaturen digitalt. Dette gjør at det kreves en del kode for å lese verdiene. For bruk, se Appendiks E.5 .

Figur 9 - Temperatursensor [39]

6.2 Målekrets

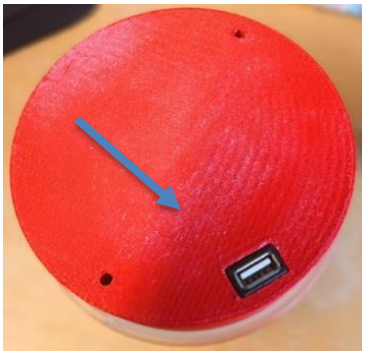
Alle komponentene i målekretsen er vanlige å bruke i sammen med mikrokontrollere. Dermed ble det å jobbe med å tegne kretstegningen veldig enkelt. Vi gikk gjennom setup-guidene (se 0), og slo sammen alle kretskoblingene til en kretstegning.



Figur 10 Kretstegning. For større versjon, se Appendiks F.1

6.2.1 Beskrivelse

Mikrokontrolleren har som sagt kun en analog inngang, derfor bruker vi en multiplekser av typen 74LV4051. Multiplekseren har en 8:1 inngang/utgang, men vi bruker kun to innganger og en utgang. For å velge hvilken sensor man henter data fra, manipuleres den digitale utgangen D1. Når den D1 er lav, mottas data fra turbiditetssensoren, når D1 er høy mottas data fra pH-sensoren. Den digitale porten veksles hvert tidels sekund i arduinokoden, slik at vi fortsatt får ganske kontinuerlige målinger. pH-sensoren gir ut et signal på 0-5V og turbiditetssensoren, gir ut 0-4.2V. Dette er over kapasiteten til den analogeporten til ESP8266, som kun kan motta signal mellom0-3.3V. Derfor bruker vi spenningsdelere for å justere ned disse verdien. Turbiditetssensoren deles med 330Ω over 1kΩ, og pH-sensoren med 510Ω over 1kΩ. Lyssensoren gjør bruk av to forskjellige digitale porter. En inngang brukes til å sende et pulstog med den målte verdien, og den andre brukes til å synkronisere sending og mottaking av datasignalet. Temperatursensoren bruker kun en digital port for å sende datasignalet. Denne sensoren trenger en pull-up-motstand på 4.7kΩ mellom strømtilførselen og datalinjen for å sikre at det er nok strøm i linjen til å kunne klare å overføre signalet [8].



Figur 11 - Strøminntak

Kretsen forsynes ved hjelp av en USB-port på oppsiden av lokket (se Figur 11), som igjen er koblet i en 5V – poweradapter som plugges i en stikkontakt. Inngangsspenningen drifter både mikrokontrolleren, multiplekseren, turbiditetssensoren og pH-sensoren, som alle er parallellkoblet. Lys- og temperatursensoren forsynes av 3.3V-utgangen på mikrokontrolleren.

6.2.2 Strømforbruksanalyse

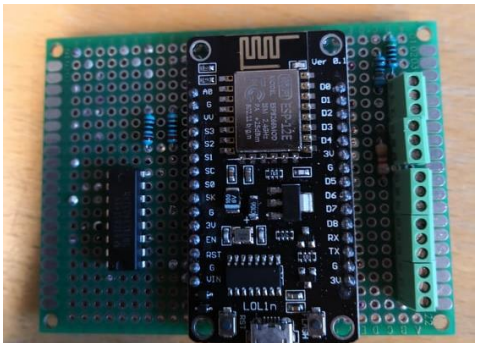
For å forsikre oss om at spenningskilden vår har kapasitet til å drifte hele kilden, uten en ekstra spenningskilde, samlet vi inn data over strømforbruket til de forskjellige komponentene.

Tag	Type	Kilde	Strømforbruk
ESP8266	Mikrokontroller	5V	275mA [9]
SEN0189	Turbiditet	5V	40mA [4]
SEN0161	pH	5V	10mA [10]
SN74LV4051A	Multiplekser	5V	1µA [11]
BH1750	Lys	3.3V	190µA [7]
DS18B20	Temperatur	3.3V	1.5mA [8]
Totalt strømforbruk:		5V	50.001mA
		3.3V	1.690mA
		Hele	51.690mA

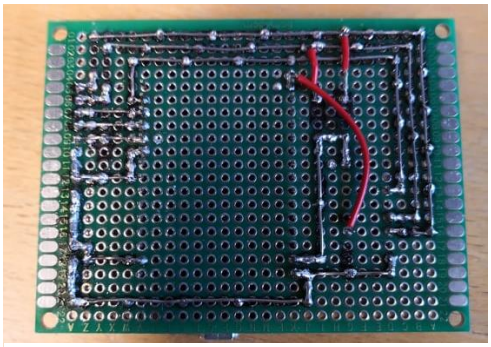
Siden alle komponenten som koblet på 5V er koblet i parallell, kan disse strømmene summeres sammen. 3.3V-utgangen på mikrokontrolleren har mulighet til å forsyne maksimalt 550mA [12], noe våre to komponenter ikke er i nærheten av å trekke. Det gjør at vår spenningskilde må kunne levere 51.690mA. Spenningskilden vi bruker leverer 500mA, og er mer enn god nok til å forsyne kretsen vår. Vi trenger derfor ingen ekstern spenningskilde.

6.2.3 Lodding av målekrets

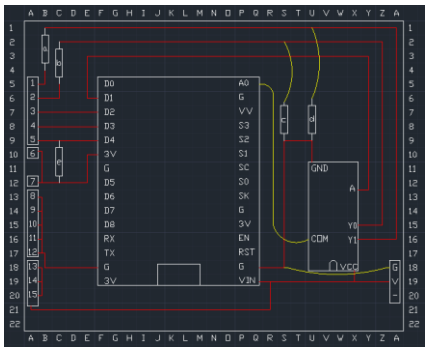
Siden Sars senteret har lodde utstyr, bestemte vi oss for å lodde kretsen sammen selv. Vi hadde lite erfaring med lodding så vi anså det som en god anledning for å tilegne oss større kunnskap innen lodding. Det ligger mye bra ute på nettet som gav oss et godt grunnlag til hvordan en lodder. Vi fant ut at det er flere metoder en kan bruke for å lodde. Det tre metodene vi ble kjent med var «larve» lodding, lodde isolerte ledninger og lodde uisolerte ledninger. Metoden vi valgte var å lodde uisolerte ledninger sammen med noen isolerte ledninger fordi det ville ta mindre tid i forhold til om vi hadde laget «larver». I tillegg så ser det ryddigere og mer oversiktlig ut. De røde isolerte ledningene som en ser på Figur 13 var nødvendig for å kunne krysse kretsen. Det eneste med å ha stort sett hele kretsen uisolert er at kretsen er mer sårbar mot kortslutning. Uansett så står kretsen på et fundament med komponentene opp, så det skal en del til for at det blir kortslutning. En utbedring som kunne blitt aktuell er å smøre på et isoleringsmateriell. Med tanke på hvordan vi gikk fram med rutingen av kretsen, fulgte vi tegningen på Figur 12. Tegningen er sett fra undersiden.



Figur 14 - Oversiden av kretskort



Figur 13 - Undersiden av kretskort



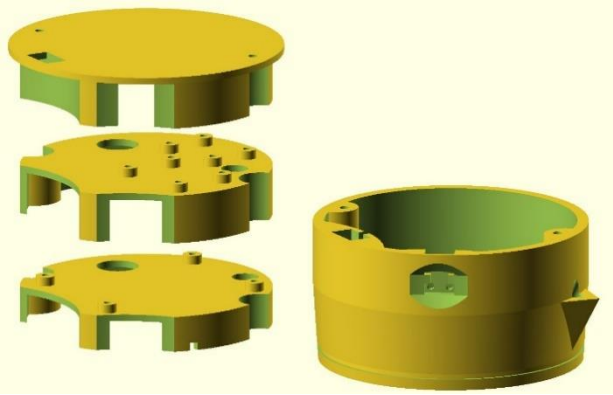
Figur 12 - Ruting av krets. For større bilde se Appendiks F.2

6.3 Design av målesystemet

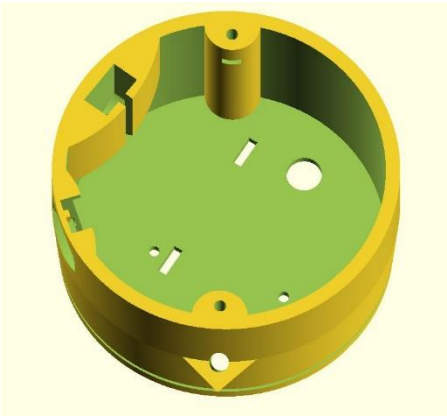
6.3.1 Idéen bak designet

Vi fikk noen problemer med å plassere hele kretsen nedi skallet, derfor har vi laget to etasjer. På det nederste laget plasserer vi mikrokontrolleren, sammen med multiplekseren og motstandene. På øverste laget har vi plassert forsterkerkretsene til turbiditetssensoren og pH-sensoren. pH-sensoren har en ganske lang sonde som det var litt problematisk å plassere. Store deler av sonden stikker ut under skallet, mens hodet går igjennom begge etasjene på innsiden. Temperatursensorene henger fra et hull i skallet. Mens lyssensoren er montert i en «fuglekasse» på siden skallet, på oppsiden av kolben. Turbiditetssensoren må også monteres i kontakt med algekulturen. Denne monteres ved hjelp av et feste som henger ned på undersiden av systemet. Vi valgte å lage dette festet som en løs del, slik at det er mulig å erstatte det hvis feste skulle gå i stykker. Da trenger man bare å skrive ut den delen, i stedet for å skrive ut hele ytterskallet på nytt.

Øverst på systemet har vi et lokk, som blir festet ved hjelp av maskinskruer av stål. På siden av skallet er det laget søyler med hull til å plassere mutterne i. Disse søylene går helt til bunnen av skallet, og holder dermed etasjene på plassen og sørger for at disse ikke kan rotere. En USB-hunnkontakt er også montert i ytterkanten av skallet, og stikker ut gjennom lokket. Det er her strømtilførselen blir tilkoblet. I tuten på skallet, som følger tuten til kolben, har vi et hull som fører til undersiden av systemet. Dette gir derfor muligheten til å montere et boblerør som brukes til å skape sirkulasjon i algekulturen.



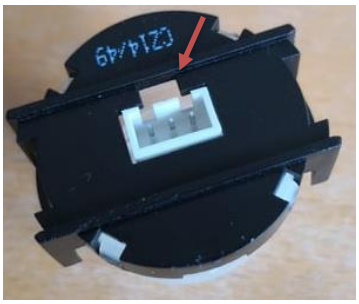
Figur 16 - Systemdesign, med skall og nivåer.



Figur 15 - Systemet sett ovenfra. Hull i bunn til utstyr.

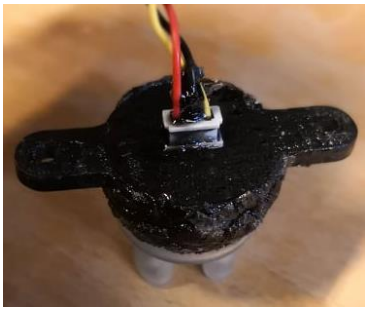
6.3.1.1 Problemer knyttet til turbiditetssensoren

Hovedproblemet med turbiditetssensoren, er at den har et stort hull ved koblingen til kablene. Dette første til utfordringer for hvordan denne sensoren kunne monteres. Dessuten hadde ikke sensoren noen festemuligheter. Vi ble derfor nødt til å designe et nytt lokk til sensoren.



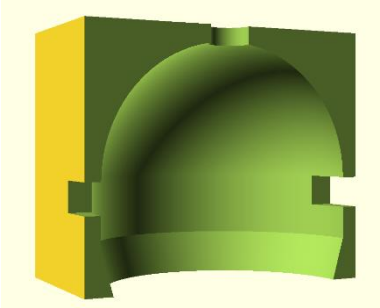
Figur 17 - Originallokket med hull i toppen.

Vår originale løsning var å støpe sensoren i en epoxy som skulle gjøre sensoren vanntett. Vi designet et nytt lokk som gjorde at vi kunne montere sensoren horisontalt i tanken. I tillegg designet vi og printet ut en forskaling som vi skulle bruke til å støpe sensoren og festet i. Fordelen med denne løsningen var at da kunne vi montere sensoren på bunn av tanken, og brukeren slapp å tenke på hvor mye væske han kunne ha i begeret.



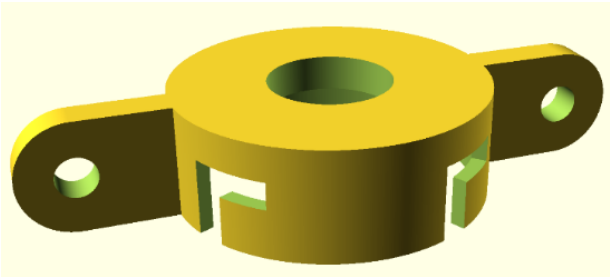
Figur 18 - Første lokk. Horisontalt feste.

Da vi skulle til med å støpe epoxyen, oppdaget vi raskt at middelet var veldig flytende. Dette førte til at vi fikk store problemer med å påføre epoxyen slik vi ønsket. Etter hvert trodde vi at vi hadde fylt opp hele formen, men etter at epoxyen hadde herdet i 3 døgn, viste det seg det var en veldig stor luftpute i støpeformen. Dermed ble ikke sensoren tett. Dessuten oppdaget vi at epoxyen var klissete, noe som kunne bli et problem med tanke på at systemet skal være sterilt. Både det at epoxyen var vanskelig å jobbe med, og problemene med å gjør det sterilt gjorde at vi valgte å gå for en annen løsning.

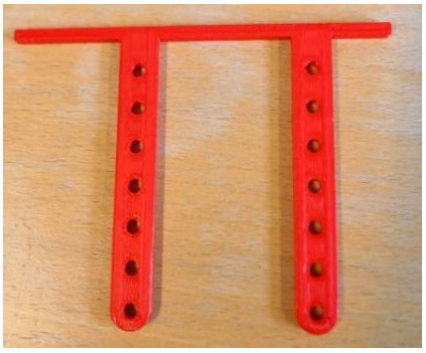


Figur 19 - En halvdel av forskalingen.

Løsningen vi gikk for, var å lage et nytt feste der sensoren henger vertikalt. Vi lagde et nytt feste til sensoren, med mange hull oppover slik at sensorens høyde kan reguleres. Denne gang ble ikke sensoren støpt inn, derfor blir ikke sensoren helt tett. Det blir derfor viktig at bruker er bevisst på hvor mye medium han har oppi begeret. Hvis man har for lite væske, får ikke sensoren kontakt med mediet, har man for mye, kan man risikere å fylle sensoren med væske og dermed ødelegge den. Vi brukte silikon til å tette igjen hull, men dette er ikke en god nok tetting til at vi kan garantere at sensoren tåler å senkes ned i mediet.



Figur 21 - Nytt lokkdesign. Vertikalt feste.



Figur 20 - Nytt festedesign. Det originale designet hadde bare hull nederst.

6.3.2 OpenScad

Programmet vi brukte for å designe systemet heter OpenScad. Det er en gratis programvare [13]. OpenScad er en tekstbasert CAD*, hvor man skriver et skript som deretter blir kompilert. Vi valgte OpenScad fordi det et er intuitivt program, som det er lett å sette seg inn samtidig som det gav oss frihet til å designe akkurat de delene vi ønsket.

```
275 module lowLayer(bubble = false)
276 {
277     difference(){
278         // Grunnform
279         difference()
280         {
281             cylinder(d=dInner, h = hLower);
282
283             translate([-dInner/2,0,0])
284                 cylinder(r=rAttach, h = hLower);
285
286             translate([dInner/2,0,0])
287                 cylinder(r=rAttach, h = hLower);
288         }
289         // Fjerning av innmat
290         difference()
291         {
292             cylinder(d = dInner - tWall*2,h = hLower - tWall );
293
294             translate([-dInner/2,0,0])
295                 cylinder(r=rAttach+tWall, h = hLower);
296
297             translate([dInner/2,0,0])
298                 cylinder(r=rAttach+tWall, h = hLower);
299         }
300
301         // Utskjøring for turbiditetssupport
302         rotate([0,0,turbSupOffsetDeg+180])
303             translate([-(dInner+20)/2, turbSupOffsetY-tTurbSupport/2, 0])
304                 cube([dInner+20, tTurbSupport, tTurbSupport]);
```

Figur 22 - OpenScad

Skriptet baserer seg på enkle geometriske figurer som kule, sylinder, kube. For å lage en figur, kaller man på funksjonen med samme navn som figuren, og legger inn parametere som radius, høyde bredde og diameter. For å plassere figurene der du ønsker dem, brukes funksjonen translate. Den tar inn en vektor som argument hvor du spesifiserer hvor langt du ønsker å flytte objektet i X-, Y- og Z-retning. Hvis du ønsker å rotere figuren din, bruker du rotate. Denne funksjonen kan brukes på to måter, enten ved å oppgi vinkelen du ønsker å rotere, og så gi vektoren du ønsker at figuren skal rotere rundt (rotate(a=180, [0,1,0])). Alternativt kan du bare oppgi vinklene som en matrise. Funksjonskallet rotate([90,180,0]) vil gi første en 90 graders rotasjonen rundt X-aksen, og deretter en rundt Z-aksen. Vi brukte den siste notasjonen mest, fordi det er sjeldent man roterer rundt andre vektorer enn de som ligger parallelt med aksene [14].

Kanskje den viktigste funksjonen er funksjonen difference. Denne figurer brukes til utskjæringer og gir oss muligheten til å lage komplekse figurer. Difference tar utgangspunkt i figuren som står først innenfor klammeparentesene og trekker fra alle de neste figurene innenfor parentesen. Det betyr at alle plasser hvor andre figurer overlapper med hovedfigurene, blir slettet. Kun hovedfiguren blir skrevet ut, noe som betyr at de andre figurene kun blir generert for å gjør innhugg i hovedfiguren og deretter slettet [15].

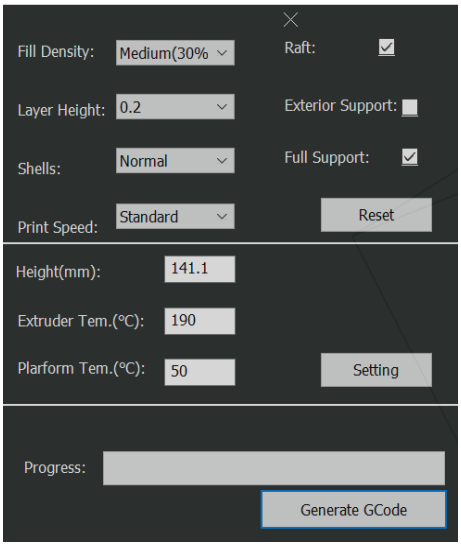
Vår opplevelse med OpenSCAD er at det er enkelt å sette seg inn. Læringskurven er veldig bratt, og etter hvert blir du flink til å så for deg hvordan scriptet jobber og dermed klarer du å gjenskape de visjonene du har for designet. Noe av utfordringene med programmet er at når skriptet begynner å bli ganske langt, er det lett å miste oversikten over koden. Da er det viktig at man har klart å legge opp en god struktur med fornuftige variabelnavn og kommentarer underveis. Ved å lage en separat fil med alle variablene ble det lettere å holde kontroll på variabelverdien. Vi fikk også problemer med å at kompilatoren begynte å ta mye tid hver gang vi skulle kompilere skriptet. Vi ble nødt til å justere ned oppløsningen på koden for å spare tid. Når skriptet var ferdig brukte kompilatoren opp mot en time på å bli ferdig.

6.3.3 Printing

Vi fikk muligheten til å benytte oss av printeren som Sars-gruppen disponerte. Printeren er en Weistek WT280X Speed. Den printer i plastikk av typen PLA, har et arbeidsområde på 150x150x140mm, og en printhastighet mellom 20 og 450 mm/s [16]. Vi opplevde at vår største del brukte rundt 12 timer. Denne modellen er relativt billig printer (tilsvarende modeller ligger ute til 9500kr, per 26.05.2016 [17]), men den gjorde nytten for de delene vi ønsket å konstruere. Så lenge skissene ikke hadde for små detaljer, gjorde printeren jobben sin. I starten hadde vi et mål om å lage til skruer og gjenger med printeren, men det viste seg at de smaleste flatene printeren kunne skape var 1.5 cm, da ble det uaktuelt.

Figur 23 - Weistek WT280X Speed [16].

Prosessen med å printe ut designet starter med å kompilere skriptet med høy oppløsning. Deretter lagres den kompilerte delen i en stl-fil. Denne filen lastes så opp i et program kalt DoraWare, dette er en software som følger med printeren. DoraWare generer en gscode av stl-filen. I tillegg legger man her inn innstillinger som tykkelse på veggen, og temperatur på filamentet og plattform. Vi brukte medium tykkelse på veggene, filamenttempertur på 190 grader og plattformtemperatur på 50 grader. I tillegg må det velges supporttype. Man bør alltid velge Raft, dette er en plattform som lages som en base for resten av printet. Exterior Support bygger opp støttevegger for utstikkende deler, slik at printeren ikke må printe i fri luft. Exterior Support printer bare disse veggene med rot i plattformen. Full Support vil lage støttevegger for alle utstikkende deler, selv om det innebærer å printe støtteveggene opp selve sluttproduktet. GCode-en og instillingene lagres i en wtk-fil, og den overføres via en minnepinne til printeren.



Figur 24 - Innstilliger i DoraWare.

6.4 Operativsystem

Operativsystemet som vi bruker på Raspberry Pi er som nevnt tidligere Windows 10 IOT Core. Dette er et operativsystem som Windows laget for å kunne sette opp IOT system med enheter som Raspberry Pi, dragonboard, arduino og andre enkeltbord datamaskiner. Operativsystemet fungerer både på enheter med og uten skjerm som kjører enten med ARM eller x86/64 struktur [18]. Det er ofte brukt i hobbyprosjekt, men er også i bruk i industrielle og kommersielle prosjekter. I vårt tilfelle så bruker vi Windows 10 IOT Core med skjerm, tastatur og datamus. En kan også bruke touch skjerm sammen med operativsystemet. Det er et brukervennlig operativsystem og gjør det veldig enkelt for brukerne som ønsker å bruke C# kode sammen med et prosjekt som krever sammenstilling av forskjellige enheter. I tillegg så ligger det mye forskjellige open source prosjekter ute på nettet så en kan bruke som utgangspunkt.

6.5 Programkode

Som tidligere nevnt valgte vi å skrive programmet i et språk vi er kjent med fra undervisning, nemlig C#. Det ble brukt Visual Studio* som IDE* for programkoden til serveren, mens klientene bruker Arduinos egen IDE. Programkoden kan deles inn i tre ulike segmenter; Hovedprogram for server, Arduinokoden og database.

6.5.1 Arduino

Klienten vi bruker i oppsettet er som tidligere nevnt en NodeMCU ESP8266. Den lar seg best bruke med en Arduino IDE. Alle sensorene er koblet via denne mikrokontrolleren, og sendes trådløst til serveren. Kode for å prosessere og håndtere data fra sensorene har vi funnet på nettet. Det er brukt fire typer biblioteker, et for å håndtere den trådløse kommunikasjonen, to for å håndtere dataen fra temperatursensoren, og to for å håndtere data fra lyssensoren. Det andre sensorene trenger ikke noe ekstra bibliotek siden det er enklere. Turbiditet og PH sensoren bruker den analoge inngangen på mikrokontrolleren med en multipleksing. Detaljer om det forskjellige sensorene og deres bruk av det tilhørende bibliotekene, finner en i setup guiden til hver av dem. Det samme gjelder for den trådløse kommunikasjonen.

6.5.2 Hovedprogram for server

Vi bruker UWP i C# for å programmere applikasjonen som vi bruker i Windows 10 IOT Core. Design av brukergrensesnittet blir gjort i xaml språket som er en del av UWP plattformen. Hovedprogrammet består av tre vinduer. Et hovedvindu som presentere alle målingene fra sensorene både som tallverdier og i individuelle dynamiske grafer. Selve klienten blir også presentert i dette vinduet. Så hvis en har flere klienter så har brukeren valget om hvilken klient en skal presentere sensor data fra. Hvis en trykker på en av grafene så vil en åpne et nytt vindu som viser grafen med et lengre tidsintervall. I det tredje vinduet har en mulighet til å hente fram lagret data fra valgt sensor. En har valget om å se verdiene både i dynamisk og statisk graf i dette vinduet.

I hovedvinduet har en mulighet til å koble seg til en klient. Dette blir gjort ved at brukeren trykker på tilkoblingsknappen. I bakgrunnen vil tråden for bruk av klassen sokkel bli kallet på og programmet vil søke over nettet etter en klient. Hvis en klient blir funnet så blir denne klienten koblet sammen med en annen tråd som opprettholder kommunikasjonen mellom programmet og klienten. Tråden som søker blir løsrevet fra denne jobben og er dermed klar for et nytt søk hvis en ønsker å koble til flere klienter. Kommunikasjonstråden vil kontinuerlig oppdatere dataen fra klienten. Hver klient er et objekt av klassen Client.

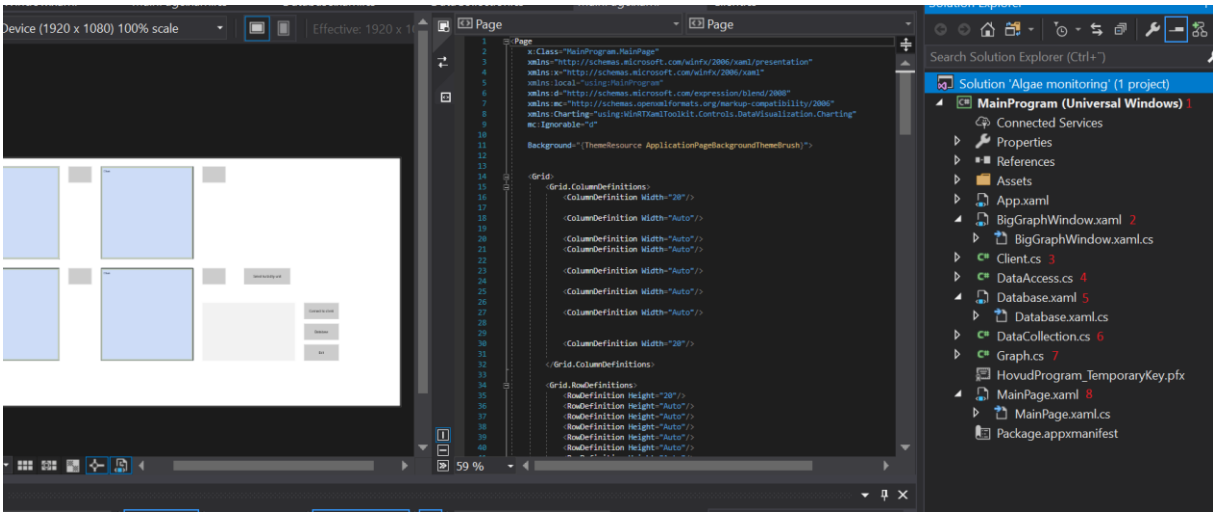
Algene som Sars bruker er delt inn i fire kulturer. Derfor har vi lagt inn valgmulighet i hovedprogrammet der en bestemmer hvilken kultur en skal måle for å gjøre de rette beregningene i turbiditet for hver av dem.

For å kunne bruke klientdataen i koden som tilhører de forskjellige vinduene, så laget vi en klasse som heter Datacollection for det. Her blir en liste av alle klienter lagt inn. En bruker også denne klassen til å ha kontroll over hvilken graf som blir trykket på i hovedvinduet. Dette er fordi vi bruker bare en graf i det vinduet som presenterer sensordataen med et lengre tidsintervall. Så da vet programmet ved hjelp av Datacollection hvilken sensor den skal vise data fra.

Siden vi har hatt en del problem med treghet i programmet på grunn av de dynamiske grafene, har vi valgt å lage et program til slik at en kan bruke det istedenfor. Det vil fortsatt være mulig å presentere dataen til sensorene i en statisk graf.

På Figur 25 blir c# programmet vist med brukergrensesnitt vinduet. Helt til venstre i figuren kan en se hele mappen med de forskjellige klassene og xmal filene til programmet merket med røde tall.

1. Dette er projektmappen.
2. xaml fil for presentering av data i stor graf. Kode bak ligg under xaml fil.
3. Klientklassen.
4. Klasse for datalagring med sql.
5. xaml fil for presentering av data som er lagret. I tillegg blir det presentert dynamisk og statisk graf for hver av sensorene.
6. Klassefil som blir brukt til å mellomlagre klient informasjon slik at en har tilgang til det i alle kode filene.
7. Klassefil for å kunne opprette en dynamisk graf eller statisk graf.
8. Xaml fil for presentering av sensor data samt dynamisk graf.



Figur 25 - c# kode med de forskjellige kode filene

6.5.3 Database

For lagring av data brukte vi SQLite*. Denne fungerer ved at den oppretter en .db-fil (database-fil) på enheten den blir brukt på. I vårt tilfelle oppretter den da en database fil på Raspberry pien, ved navn SQLiteSample2.db. Her oppretter den en tabell som inneholder de relevante dataene vi ønsker å lagre etter kategori. En henter ut og legger til data i tabellen ved hjelp av SQL* språket. Fordelen med dette er at SQLite da blir svært enkel å bruke, i tillegg til at det fungerer godt på de fleste plattformer.

Slik vi bruker database i denne oppgaven er å lagre dataene i databasen underveis som målingene gjøres. De kan aksesseres direkte i programmet. En kan enten hente ut alle dataene i hele tabellen i en liste, eller så kan en søke mellom to tidspunkt for å hente ut dataene i dette tidsrommet. En kan velge å se data fra alle sensorene samtidig, eller en kan hente ut dataene til spesifikke sensorer. Databasen vil ligge, med dataene lagret, på Raspberry pien helt frem til en velger å fjerne den.

```
1     /* This is the code behind the SQL data storage.*/
2
3   using System;
4   using System.Collections.Generic;
5   using System.Linq;
6   using System.Text;
7   using System.Threading.Tasks;
8   using Microsoft.Data.Sqlite;
9
10  namespace DataAccessLibrary
11  {
12
13      public static class DataAccess
14      {
15          private static readonly String DB_NAME = "SQLiteSample2.db";
16          private static bool tableExists = false;
17          public static void InitializeDatabase()...
38
39          public static void AddData(string temp, string PH, string light, string turbidity,  DateTime dateTime, int clientNr)...
70
71          public static void ClearDatabase()...
95
96          public static void DropTable()...
126
127         public static List<String> GetData(int clientNr, string flyOutChoice)...
230
231         public static List<String> GetSpecificData(string dateFrom, string secFrom, string dateTo, string secTo, string flyOutChoice, int clientNr)...
326     }
327
328
329 }
```

Figur 26 - Klasse for database

Klassen inneholder totalt seks metoder. For opprettelse av databasen brukes InitializeDatabase(). Den fungerer med at den lager/åpner en databasefil. Dersom det ikke eksisterer en tabell i databasefilen, vil den opprette denne med variablene som er definert i tableCommand. Tabellen inneholder også en Primary Key, dette er en unik verdi som identifiserer hver rekke i tabellen [19].

```
public static void InitializeDatabase()
{
    using (SqliteConnection db =
        new SqliteConnection("Filename=" + DB_NAME))
    {
        db.Open();

        String tableCommand = "CREATE TABLE IF NOT " +
            "EXISTS MyTable (Id INTEGER PRIMARY KEY, " +
            "temp NVARCHAR(2048) NULL, " +
            "PH NVARCHAR(2048) NULL, " +
            "light NVARCHAR(2048) NULL, " +
            "turbidity NVARCHAR(2048) NULL, " +
            "date NVARCHAR(2048) NULL, " +
            "clientNr INTEGER) ";

        SqliteCommand createTable = new SqliteCommand(tableCommand, db);
        tableExists = true;
        createTable.ExecuteReader();
    }
}
```

Figur 27 - Metode for initialisering av databasen

```
public static void AddData(string temp, string PH, string light, string turbidity,  DateTime dateTime, int clientNr)
{
    string dateTimeS = dateTime.Date.ToString().Substring(0, dateTime.Date.ToString().IndexOf(' ') + 1) + dateTime.TimeOfDay.ToString().Substring(0,dateTime.TimeOfDay.ToString().IndexOf('.'));
    using (SqliteConnection db =
        new SqliteConnection("Filename=" + DB_NAME))
    {
        db.Open();

        SqliteCommand insertCommand = new SqliteCommand
        {
            Connection = db,

            // Use parameterized query to prevent SQL injection attacks
            CommandText = "INSERT INTO MyTable VALUES (NULL, @temp, @PH, @light, @turbidity, @date, @clientNr);"
        };
        insertCommand.Parameters.AddWithValue("@temp", temp);
        insertCommand.Parameters.AddWithValue("@PH", PH);
        insertCommand.Parameters.AddWithValue("@light", light);
        insertCommand.Parameters.AddWithValue("@turbidity", turbidity);
        insertCommand.Parameters.AddWithValue("@date", dateTimeS);
        insertCommand.Parameters.AddWithValue("@clientNr", clientNr);


        insertCommand.ExecuteReader();

        db.Close();
    }

}
```

Figur 28 - Å legge til data i databasen

Når en skal legge til data i tabellen brukes metoden AddData(). Denne tar inn dataverdiene og legger det til i tabellen. Linjen i CommandText bruker SQL språket til å legge til verdier i MyTable som er tabellen i databasen vår. Det brukes parametriserte verdier for å unngå «SQL injection attacks». Dette er en sikkerhetsprosedyre som skal sørge for at ingen kan lure inn andre kommandoer i databasen, som kan endre verdiene eller i verste fall fjerne hele tabellen. Dette er i utgangspunktet ikke en stor trussel for oss av to grunner. For det første er ikke programmet laget for langtidslagring av data. Det vil normalt sett kun brukes til å lagre data når målingen utføres. Det vil ikke være nødvendig å lagre dataene over lengre tid, noe som gjør det mindre farlig om en slik kommando skulle forekomme. For det andre vil programmet i utgangspunktet bli brukt på en lokal server, og vil være utilgjengelig for folk utenfor det lokale nettverket.

```
public static void DropTable()
{
    if (tableExists)
    {
        using (SqliteConnection db =
            new SqliteConnection("Filename=" + DB_NAME))
        {
            db.Open();

            SqliteCommand insertCommand = new SqliteCommand
            {
                Connection = db,

                CommandText = "DROP TABLE MyTable;"
            };

            insertCommand.ExecuteNonQuery();

            db.Close();
        }
        tableExists = false;

        InitializeDatabase();
    }


}
```

Figur 29 - Å forkaste tabellen

ClearDatabase() er også en funksjon, som sletter innholdet i tabellen, men fjerner ikke tabellen. Denne brukes ikke i vårt program.

Dersom en skal starte en ny måling, og dermed ikke trenger de gamle måledataene, vil en kunne bruke DropTable(). Denne vil da bruke SQL kommandoen «DROP TABLE MyTable;». Dette gjør at tabellen fjernes helt fra databasen. Her skal ingen variabel være med, og dermed er ikke en parametrisering nødvendig.

```
if (flyOutChoice == "Temperature")
{
    SqliteCommand selectCommand = new SqliteCommand
        ("SELECT temp, date, clientNr FROM MyTable WHERE date BETWEEN @dateFrom AND @dateTo AND clientNr = @clientNr;", db);
    selectCommand.Parameters.AddWithValue("@dateFrom", dateFrom + ":" + secFrom);
    selectCommand.Parameters.AddWithValue("@dateTo", dateTo + ":" + secTo);
    selectCommand.Parameters.AddWithValue("@clientNr", clientNr);
    SqliteDataReader query = selectCommand.ExecuteReader();
    while (query.Read())
    {

        entries.Add($"Time: {query.GetString(1)} - Temperature: {query.GetString(0)}°C");
    }
}
```

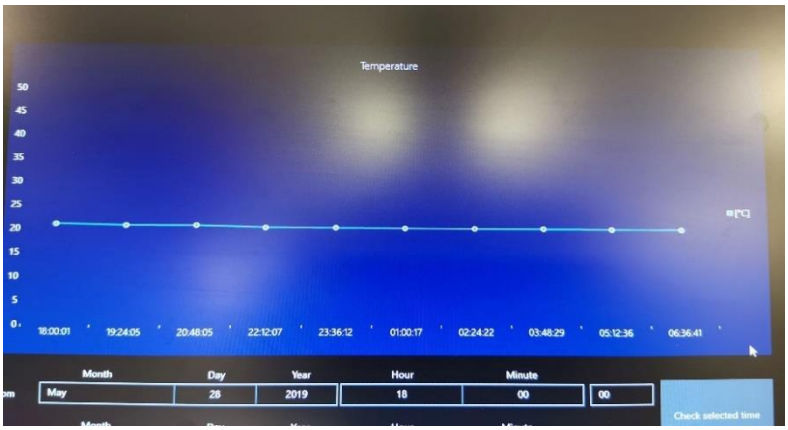
Figur 30 - Å hente ut spesifikke data fra tabell

Over vises et lite utdrag av koden under metoden GetSpesificData(). Denne bruker SQL funksjonen SELECT, som velger hvilke tabellverdier vi skal hente ut data for. Den spesifiserer så hvilke tidspunkt vi skal se mellom ved å bruke «WHERE date BETWEEN @dateFrom AND @dateTo AND clientNr = @clientNr;». Denne sjekker altså der dato er mellom de to valgte tidspunktene, og den sjekker kun for den utvalgte klienten. Det opprettes en string som legges inn i en liste ved navn entries, som returneres av funksjonen. Query.GetString(x) henter ut verdien i kolonne x som stringverdi. Resten av koden er lagt ved i vedlegg.

7 Testing

7.1 Første langtidstest

For å sjekke at programmet kan brukes over lengre tid, tok vi en langtidstest over natten. Denne testen gjorde vi på programversjonen hvor vi ikke har dynamiske grafer med. Testen var vellykket, og vi kunne se tilbake gjennom hele natten med resultater fra både lys, temperatur, endring i turbiditet og PH. Selv om en ikke kan se det veldig godt på bildet (Figur 31) var det omtrent en grad celsius i forskjell på dagtid og nattetid. Den varierte fra ca. 20.7°C - 19.7°C. Vi kunne se i databaselisten at nesten nøyaktig rundt 08.00 var temperaturen akkurat kommet seg opp til rundt 20°C. Det vil si at rommet produktet skal stå i å gjøre prøver, endres relativt lite i temperatur, noe som ble bekreftet av oppdragsgiver. Dette betyr at vi kan se vekk ifra temperaturendringer, med tanke på temperaturavhengigheten til turbiditetssensoren.



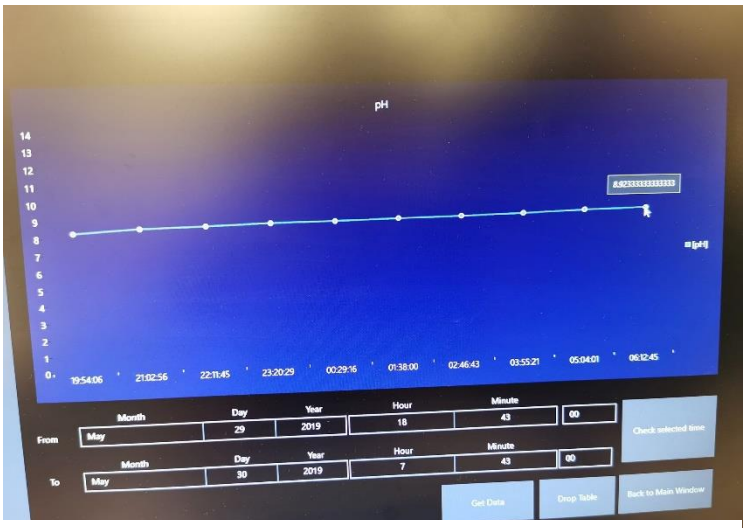
Figur 31 - Temperatur over natten

I tillegg fikk vi teste en rekke andre ting.

1) Vi fikk verifisert at databasen kunne lagre verdier over lang tid.
2) Vi slo av og på Raspberry pien, og fikk verifisert at databasen har verdiene lagret lokalt til enhver tid.
3) Vi fikk testet at nettverket som er satt opp i det aktuelle bruksrommet fungerte optimalt over lengre tid.
4) Vi fikk testet at sensorverdiene gav kontinuerlig målinger uten å miste kontakt.

7.2 Andre langtidstest

Den andre langtidstesten ble gjort med alger i kolben. Denne ble gjort under samme forutsetninger som kolbene som står på laben. Disse står med et konstant lys som ble målt til rundt 5580 lux fra lyssensoren vi bruker. I tillegg står de noe innelukket bak et forheng. Dette gjør at temperaturen ligger relativt stabilt, men ligger en del høyere enn romtemperaturen ellers (rundt 23°C). Algekulturen som ble målt ble foret med mye næring før måling, for å kunne se en utvikling i kulturen over natten. Dette førte til at vi kunne se en tydelig endring i blant annet PH-verdien. Helt fra starten lå PH-verdien på rundt 8. I slutten var denne PH-verdien kommet opp mot 9. Dette er et relativt stort utslag i forhold til PH-skalaen (Figur 60). Vi fikk ellers følgende ut av testen:



Figur 32 - PH endring over tid

1) Vi hadde ikke fått tettet til turbiditetssensoren på tidspunktet testen ble utført. Dette bød på et par problemer. Blant annet kunne vi ikke bruke luftrøret som er ment for å gi luft til algekulturen slik at den ikke legger seg i bunnen av kolben, da dette gav luftbobler som kunne sprute vann oppi sensoren. Den beste løsningen på problemet er å finne en måte å tette sensoren på. Et alternativ er å ha en autoklavert magnet i bunn av kolben, som blir spunnet av en plate på undersiden. Så lenge det ikke er et sterkt nok magnetfelt til å påvirke sensorene, ville dette vært en tilstrekkelig løsning.
2) Temperaturen bak forhenget der produktet ble testet var det som nevnt noe varmere enn resten av rommet. Det gjorde at når kolben ble plassert her, og væsken ble varmet opp, ble det dannet kondens. Dette er også med å true turbiditetssensoren dersom den ikke er tett.
3) Fant feil med visning av statisk graf på PH-sensoren, men fikk rettet denne feilen.
4) Luftbobler ble dannet i væsken, som kan være med å påvirke sensorverdiene til turbiditetssensoren. Dette kan løses ved å få sirkulasjon i væsken, enten ved hjelp av luft eller magnet.

Figur 33 - Kolbe med kondens

7.3 Test av turbiditet

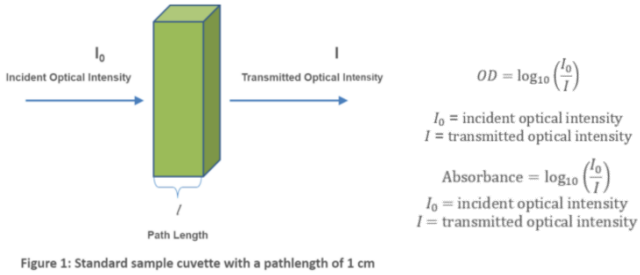
Turbiditetssensoren manglet et forhold mellom målt verdi og optical density som skulle brukes for å beregne celler/ml.

7.3.1 Spectrophotometer og turibiditet

For å teste turbiditeten satt vi opp en lineær sammenheng mellom resultatene vi fikk fra et spectrophotometer* som dem har ved instituttet. Dette apparatet gir ut en verdi kalt absorbance, som er en verdi på hvor mye av et lys som sendes gjennom som er blitt absorbert. For samme bølgelengde, ved bruk av samme bølgelengde vil optical absorbance og optical density være den samme [20]. Dermed vil den lineære sammenhengen være et mål på optical density, som er det vi er ute etter.

Figur 34 - Spectrophotometer

Kuvettene* vi brukte i målingene var på 1 cm og bølgelengden i spectrophotometeret var på 600nm. For å se en endring i tettheten av algene, tok vi fire ulike prøver. Da startet vi med 80 ml av algekulturen i første prøve. Andre prøve bestod av 40ml algekultur og 40ml sjøvann, tredje prøve av 20 ml algekultur og 60 ml sjøvann, mens fjerde prøve var med 10 ml algekultur og 70 ml sjøvann. Dermed fikk vi halvert densiteten for hver prøve. Samme prøvene ble testet på turbiditetssensoren for å sette opp den representative analoge verdien til hver målt densitet fra spectrophotometeret.



Figur 35 - Sammenheng OD og absorbance [20].

7.3.2 Linearisering av turbiditetssensor

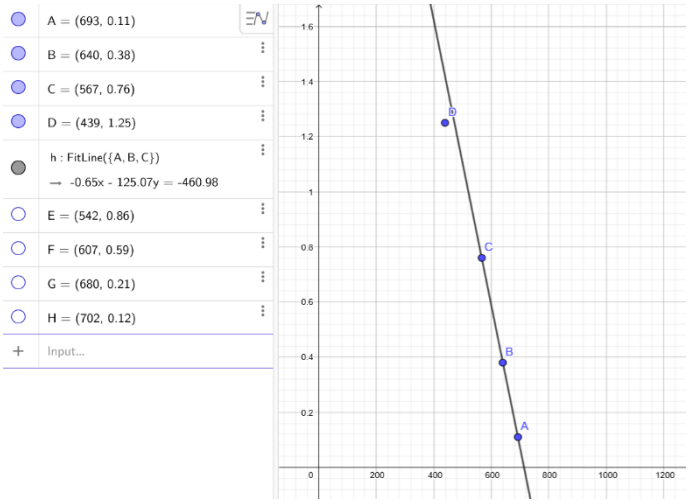
Vi tok flere målinger for å sette opp forholdet mellom turbiditetssensoren og spectrophotometeret. De første lineariseringene viste seg å være veldig upresise. De ulike grunnene til dette var:

1) De første målingene ble gjort i et annet rom enn det som skal brukes på laben. Dette førte til en stor temperaturforskjell, som har en innvirkning på hvor presis den lineære sammenhengen er mellom de to måleteknikkene.
2) På kretskortet inne i turbiditetssensoren ligger en justeringsskrue. Her kan en stille hvor høy spenningen på utsignalet skal være. Denne lå i starten for høyt, noe som førte til at flere av målingene var i metning og gav 1024 i analogverdi inn på ESP8266.

3) Strømforbruket til ESP8266en påvirker måleverdiene. Da vi først satte opp turbiditetssensoren for seg selv, ble det feil målinger i forhold til at målingene ble forskjøvet når alt var tilkoblet.

Etter en del testing fikk vi derimot ut gode resultater:

Væske	Abs	Analog Verdi
1	1.2531	439
2	0.7615	567
3	0.3804	640
4	0.1108	693

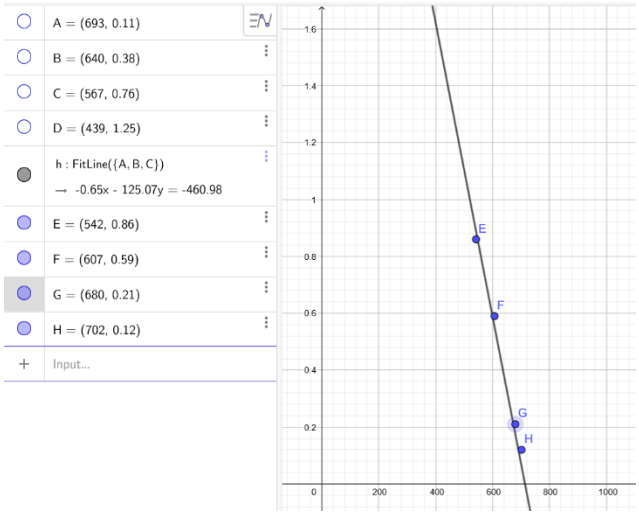


Figur 36 – Punkter og linearisering for CC alge.

Der Abs er absorbance verdien fra spectrophotometeret, mens analog verdi er verdien inn på ESP8266. Lineariseringen ble gjort med CC alger, men ble senere verifisert med de andre algene.

Blant annet ble punktene for Syn som følgende:

Væske	Abs	Analog Verdi
1	0.8567	542
2	0.5903	607
3	0.2120	680
4	0.1159	702



Figur 37 – Punktene fra Syn alge på samme linearisering

Vi ser at målingene for de to ulike algetypene følger lineariseringen svært presist.

Slik fikk vi ut følgende formel, hvor x er analog innverdi på sensoren:

$$Abs = OD = -\frac{0.65}{125.07} * x + \frac{460.98}{127.07} = -0.0052 * x + 3.6278$$

Denne kan igjen brukes i følgende formler for å finne celler/ml. Formlene er oppgitt av oppdragsgiver:

Iso [celler/ml]:

$Y = (2 * 10^7 * OD) - 231019$, Gjelder for OD_{max} = 0.31.

CC [celler/ml]:

$Y = (4 * 10^7 * OD) - 27903$, Gjelder for OD_{max}= 0.15.

Rhino [celler/ml]:

$Y = (7 * 10^6 * OD) - 18162$, Gjelder for OD_{max}= 0.42.

Syn [celler/ml]:

$Y = (7 * 10^6 * OD) - 18162$

Dersom en bruker formelen for å finne celler/ml:

Væske	Abs	Analog Verdi	Celler/ml
1	0.8567	542	5,9*10^5
2	0.5903	607	4,1*10^6
3	0.2120	680	1,5*10^6
4	0.1159	702	7,9*10^5

En kan dermed gi ut et estimat av antall celler i algekulturen til ethvert tidspunkt under en kontinuerlig måling.

Figur 38 - Prøver som er halvert i densitet

Figur 39 - Prøver som fylles i kuvettene

8 Diskusjon

8.1 Forslag til forbedring

Selv om sluttproduktet tilfredsstiller kravspesifikasjonene, er det flere ting som kan forbedres. Grunnet blant annet begrenset tid til oppgaven, er det blitt gjort noen forenklinger som har vært nødvendig for å komme i mål. I denne delen vil vi diskutere og foreslå endringer, og forbedringer til sluttproduktet.

8.1.1 Software

- Løsningen vi valgte for å presentere verdiene våre i graf var en lavnivås løsning. Grunnen til dette var fordi vi fant ikke et bedre verktøy. Det verktøyet vi ønsket å bruke var LiveChart, men det har ikke blitt utviklet slik at en kan bruke det i UWP [21]. Så i senere tid hvis det kommer en versjon av UWP som aksepterer bruk av LiveChart, vil dette være en mye bedre løsning. Det virker også ut som grafløsningen vår skaper en treghet i programmet som gjør at alt av innhentning til presentering av data blir tregere. I tilfelle dette blir et stort problem så har vi som tidligere nevnt, laget en versjon av programmet der vi ikke har med det dynamiske grafene. En vil fortsatt kunne se historikken av målingene i en statisk graf.
- Når vi har flere klienter koblet til programmet så blir det en kraftig treghet. Hvis vi gjør det samme i programmet som ikke inneholder det dynamiske grafene vil vi ikke ha samme treghet.
- Brukergrensesnittet vil ikke skalere seg etter hvilken som helst skjerm. Vi vet at programmet er skalert rett etter HP2311x (Monitor på Sars).
- Det er ikke blitt tatt hensyn til om programmet mister kontakten med nettverket. Data vil fortsatt bli lagret, men programmet vil krasje og en må starte det på nytt.
- Det er ikke blitt tatt hensyn til om en klient mister kontakten med programmet.
- En kan ikke koble fra en klient.
- Hvis en trykker connect to client flere ganger vil en få feilmelding og en må starte programmet på nytt.

8.1.2 Komponenter

- Dersom Raspberry piens CPU når 80°C vil den begynne å klokke ned [22], for ikke å skade enheten. Når hovedprogrammet kjøres, vet vi at CPUen blir veldig varm, og programmet virker å gå tregere. Dermed kan det antas at temperaturen når rundt 80°C. I Raspbien kan en måle CPUens temperatur, men foreløpig finnes det ingen god måte å gjøre dette på i Windows IoT. Dersom den når denne temperaturen, vil kjøleribber til CPUen være en god investering for å holde hastigheten på Rpien* oppe.

Figur 40 - Raspberry pi med kjøleribber [40].

- Raspberry pi 3 B har ikke innebygd RTC*. Det betyr at når strømmen brytes, vil ikke klokken oppdateres. Den vil først oppdatere klokken når den er tilkoblet et nettverk. Dette fører til at Rpien av og til må slås av og på for å få korrekt tid i programmet. En løsning for dette kan være å kjøpe en RTC-modul. En RTC-modul fungerer ved at den fortsetter å gi strøm til klokken, selv om hovedstrømmen til kortet brytes. Dette gjøres via et batteri.

Figur 41 - RTC-modul til Rpi 3 [41].

- Det var ønskelig fra oppdragsgiver at vi skulle benytte Rpi. Dersom dette ikke var en kravspesifikasjon, kunne det vært aktuelt å bytte til Dragonboard. Denne skal ha høyere ytelse enn Rpien, og kan også kjøres med Windows IoT [23].

- Turbiditetssensoren viser seg å være svært temperaturavhengig. I og med at romtemperaturen kan være forskjellig fra dag til dag, kunne det vært aktuelt med en turbiditetssensor som er mer konsistent i målingene, uavhengig av temperatur. Vi har også sett på løsning hvor vi bruker temperatursensoren for å kompensere, til å gjøre målingene konsistente.

8.1.3 Det fysiske produktet

- Det mangler en egen port for programoverføring til Arduinokortet. Når eksempelvis PH-sensoren må kalibreres, må dette gjøres i programkoden i Arduino, for så å overføres via USB til micro-USB inngangen på ESP8266. Per dags dato må en komme til kortet, som ligger i nederste «etasje», for så å koble seg til med kabel. Det ideelle hadde vært at denne lå klar i 3D-konstruksjonen.
- Kretskortet er loddet med uisolerte helkordeler. Det er en viss risiko for at det kan bli kortslutning på grunn av det. Så en kan løse det med å bruke isoleringsmateriell som en smører på kretsen, eller bytte ut med jumperwires.

9 Konklusjon

Dette prosjektet har gitt oss mange gode erfaringer vi vil ta med oss videre. Vi har blant annet fått kjennskap til hvordan det er å samarbeide med en oppdragsgiver. Det å kunne forholde seg til visse kravspesifikasjoner, men også å foreslå endringer og forbedringer til en tenkt løsning. Vi har fått jobbe i team, på en oppgave som krever god kommunikasjon både mellom gruppedeltakere og oppdragsgiver, men har også måttet sette oss inn i individuelle arbeidsoppgaver. Vi har fått brukt mye av kunnskapen vi har opparbeidet oss på høgskolen, men har også måttet tilegne oss mye ny kunnskap. Oppgaven har bydd på utfordringer innen programmering, mekanisk arbeid og rapportskriving. Grunnet mye ventetid på flere komponenter, og begrenset tid vi hadde tilgjengelig på Sars, har det vært essensielt fleksibilitet i forhold til arbeidstider.

Vi har også fått erfare hvordan det er å skulle planlegge å prøve å sette tidsfrister for å fullføre ulike arbeidsoppgaver. Uforutsette problemer og forsinkelser av deler har gitt utfordringer med å holde tidsfristen. Til tross for at vi så oss nødt til å ta en del veivalg underveis, og å utsette eller endre tidsfrister, erfarte vi at god planlegging var en av nøklene til suksess. Tidsbegrensningen har ført til at noen forenklinger har vært nødvendig, men vi har identifisert det meste av problemer, og har kommet med forslag til utbedring for oppdragsgiver.

En stor del av arbeidet har gått ut på feilsøking av komponenter og kretser. I tillegg har testing og verifisering av enkeltkomponenter og systemet som en helhet vært viktig for å kunne sikre en viss kvalitet på produktet.

Vi har bygget en komplett løsning som inneholder både software og hardware som tilfredsstiller kravspesifikasjonene. Produktet kan gi et rimelig estimat over antall partikler i algekulturen. Oppdragsgiver kan nå følge utviklingen på algene over tid, og kan følge endringene mens de skjer. De to programmene som er laget kan vise kontinuerlige målinger, og statisk graf av endringen over tid. Det er laget program til både klient og server, på en måte som tillater at flere klienter er tilkoblet serveren samtidig. I tillegg er produktet bygget på en måte som lar det lett dupliseres og erstattes dersom deler skulle blitt ødelagt. Det har også blitt laget detaljerte beskrivelser og guider på hvordan en kan sette opp og kalibrere de ulike komponentene som krever det.

Appendiks A Litteraturliste

[1] Sars, «Research,» [Internett]. Available: https://www.sars.no/research/. [Funnet 30 01 2019].

[2] Sars, «About sars,» [Internett]. Available: https://www.sars.no/about/. [Funnet 30 01 2019].

[3] Sars, «Personell,» [Internett]. Available: https://www.sars.no/personnel/. [Funnet 30 01 2019].

[4] DFRobot, «Turbidity Sensor,» 25. 05. 2017. [Internett]. Available: https://www.dfrobot.com/wiki/index.php/Turbidity_sensor_SKU:_SEN0189. [Funnet 21. 05. 2019].

[5] BotShop, «botshop.co.za,» [Internett]. Available: https://www.botshop.co.za/how-to-use-a-ph-probe-and-sensor/. [Funnet 24. 05. 2019].

[6] p. Professor, «youtube.com,» 25. 05. 2017. [Internett]. Available: https://www.youtube.com/watch?v=PBTn4gTEbkU. [Funnet 27. 05. 2019].

[7] R. Semiconductor, «Datasheet BH1750FVI,» 01. 04. 2010. [Internett]. Available: http://www.elechouse.com/elechouse/images/product/Digital%20light%20Sensor/bh1750fvi-e.pdf. [Funnet 18. 02. 2019].

[8] Dallas Semiconductors, «Datasheet DS18B20,» [Internett]. Available: https://cdn.sparkfun.com/datasheets/Sensors/Temp/DS18B20.pdf. [Funnet 18. 02. 2019].

[9] Einstronic, «Datasheet ESP8266,» 01. 07. 2017. [Internett]. Available: https://einstronic.com/wp-content/uploads/2017/06/NodeMCU-ESP8266-ESP-12E-Catalogue.pdf. [Funnet 18. 02. 2019].

[10] dcervantes, «Manual SEN0161,» Scidle.com, 10. 03. 2017. [Internett]. Available: https://scidle.com/how-to-use-a-ph-sensor-with-arduino/. [Funnet 18. 02. 2019].

[11] Texas Instruments, «Datasheet SN74LV4051A,» 01. 09. 2015. [Internett]. Available: http://www.ti.com/lit/ds/symlink/sn74lv4051a.pdf. [Funnet 18. 02. 2019].

[12] oxurane, «ESP8266.com,» 20. 09. 2017. [Internett]. Available: https://www.esp8266.com/viewtopic.php?f=13&t=16199. [Funnet 28. 05. 2019].

[13] M. Kintel, «About OpenScad,» [Internett]. Available: https://www.openscad.org/about.html. [Funnet 07. 05. 2019].

[14] OpenSCAD, «OpenSCAD User Manual/Transformations,» 12. 02. 2019. [Internett]. Available: https://en.wikibooks.org/wiki/OpenSCAD_User_Manual/Transformations. [Funnet 21. 05. 2019].

[15] OpenSCAD, «User_Manual/CSG_Modelling,» 06. 12. 2018. [Internett]. Available: https://en.wikibooks.org/wiki/OpenSCAD_User_Manual/CSG_Modelling. [Funnet 21. 05. 2019].

[16] Weistek, «weistek.net,» [Internett]. Available: http://www.weistek.net/3414/3514/6103/1742.html. [Funnet 26. 05. 2019].

[17] RS, «no.rs-online.com,» [Internett]. Available: https://no.rs-online.com/web/p/3d-printers/8625705/. [Funnet 26. 05. 2019].

[18] V. Boricha, «hub.packtpub.com,» 07. 05. 2018. [Internett]. Available: https://hub.packtpub.com/windows-10-iot-core-what-you-need-to-know/. [Funnet 24. 05. 2019].

[19] W3Schools, «SQL Database,» [Internett]. Available: https://www.w3schools.com/sql/sql_primarykey.asp. [Funnet 24. 05. 2019].

[20] NanoHybrids, «nanohybrids.net,» [Internett]. Available: https://nanohybrids.net/pages/differences-between-optical-density-absorbance-and-extinction-of-gold-nanoparticles. [Funnet 30. 05. 2019].

[21] msvaillant, «GitHub,» 24. 04. 2018. [Internett]. Available: https://github.com/Live-Charts/Live-Charts/issues/776. [Funnet 24. 05. 2019].

[22] Raspberry PI, «Documentation,» [Internett]. Available: https://www.raspberrypi.org/documentation/faqs/#pi-performance. [Funnet 29. 05. 2019].

[23] DeviceHive, «TowardsDataScience.com,» 26. 05. 2017. [Internett]. Available: https://towardsdatascience.com/rpi3-vs-dragonboard-f4dd877b7da9. [Funnet 21. 05. 2019].

[24] Microsoft, «docs.microsoft.com,» 07. 05. 2018. [Internett]. Available: docs.microsoft.com. [Funnet 21. 05. 2019].

[25] MQTT, «mqtt.org,» [Internett]. Available: http://mqtt.org/. [Funnet 21. 05. 2019].

[26] H. Gilbert, «web.archive.org,» 02. 02. 1995. [Internett]. Available: https://web.archive.org/web/20041204044203/http:/www.yale.edu/pclt/COMM/TCPIP.HTM. [Funnet 21. 05. 2019].

[27] W3Schools, «SQL Tutorial,» [Internett]. Available: https://www.w3schools.com/sql/sql_intro.asp. [Funnet 21. 05. 2019].

[28] SQLite, «sqlite.org,» 16. 04. 2019. [Internett]. Available: https://www.sqlite.org/index.html. [Funnet 21. 05. 2019].

[29] Microsoft, «visualstudio.microsoft.com,» [Internett]. Available: https://visualstudio.microsoft.com/. [Funnet 21. 05. 2019].

[30] R. Devine, «windowscentral.com,» 14. 03. 2019. [Internett]. Available: https://www.windowscentral.com/how-install-windows-10-iot-raspberry-pi-3. [Funnet 21. 05. 2019].

[31] Eland Cables, «elandcables.com,» [Internett]. Available: https://www.elandcables.com/electrical-cable-and-accessories/cables-by-type/pvc-cable. [Funnet 24. 05. 2019].

[32] claws, «GitHub,» 25. 03. 2013. [Internett]. Available: https://github.com/claws/BH1750. [Funnet 24. 05. 2019].

[33] RoboIndia, «roboindia.com/tutorials,» [Internett]. Available: https://roboindia.com/tutorials/ds18b20_temp_sensor_nodemcu. [Funnet 24. 05. 2019].

[34] P. P, «GitHub,» 08. 03. 2017. [Internett]. Available: https://tttapa.github.io/ESP8266/Chap01%20-%20ESP8266.html. [Funnet 24. 05. 2019].

[35] D. C. Caballero, «Scidle,» 10. 03. 2017. [Internett]. Available: https://scidle.com/how-to-use-a-ph-sensor-with-arduino/. [Funnet 24. 05. 2019].

[36] mfalkvidd, «MySensors.org,» 06. 03. 2019. [Internett]. Available: https://www.mysensors.org/build/light-bh1750. [Funnet 23. 05. 2019].

[37] Amazon, «Amazon.in,» 25. 08. 2015. [Internett]. Available: https://www.amazon.in/ESP8266-NodeMcu-WiFi-Development-Board/dp/B00UY8C3N0. [Funnet 23. 05. 2019].

[38] RobotShop, 12. 10. 2017. [Internett]. Available: https://www.robotshop.com/en/gravity-analog-turbidity-sensor.html#description. [Funnet 23. 05. 2019].

[39] AkosLukacs, «GitHub,» 14. 02. 2019. [Internett]. Available: https://github.com/espruino/EspruinoDocs/blob/master/devices/DS18B20.md. [Funnet 24. 05. 2019].

[40] AliExpress, «AliExpress.com,» 09. 12. 2018. [Internett]. Available: https://www.aliexpress.com/item/32336262043.html. [Funnet 22. 05. 2019].

[41] Dealextreme, «dx.com,» 09. 10. 2017. [Internett]. Available: https://www.dx.com/p/i2c-rtc-ds1307-high-precision-rtc-module-real-time-clock-module-for-raspberry-pi-red-2056687#.XOfJUOgzY2x. [Funnet 24. 05. 2019].

[42] JanSan Consulting, «jansanconsulting.com/ph-scale,» [Internett]. Available: https://www.jansanconsulting.com/ph-scale.html. [Funnet 24. 05. 2019].

[43] Wikipedia, «wikipedia.org/wiki/Spenningsdeler,» 05. 03. 2019. [Internett]. Available: https://no.wikipedia.org/wiki/Spenningsdeler. [Funnet 24. 05. 2019].

[44] Hach, «hach.com,» [Internett]. Available: https://www.hach.com/singlet-single-use-ph-buffer-kit-ph-7-00-10-01-pk-2x10/product?id=7640205064#. [Funnet 21. 05. 2019].

[45] R. Santos og S. Sara, «randomnerdtutorials.com,» 05. 08. 2016. [Internett]. Available: https://randomnerdtutorials.com/esp8266-ds18b20-temperature-sensor-web-server-with-arduino-ide/. [Funnet 14. 05. 2019].

[46] ElectronicWings, «electronicwings.com/nodemcu/,» [Internett]. Available: https://www.electronicwings.com/nodemcu/nodemcu-gpio-with-arduino-ide. [Funnet 14. 05. 2019].

Appendiks B Forkortelser og ordforklaringer

Forkortelse /Ord	Forklaring
UWP	Står for Universal Windows Platform. Gjør det mulig å bruke programmet på flere plattformer enn flere andre løsninger [24].
MQTT	En maskin-til-maskin (M2M)/ «Internet of Things» tilkoblingsprotokoll, som er designet for å være en lettvint måte å transportere datapakker/meldinger på [25].
TCP/IP	TCP/IP er en kommunikasjonsprotokoll. IP er ansvarlig for å sende pakker med data fra node til node. TCP skal verifisere at den korrekte dataen blir sendt fra klient til server. Dersom data mistes, skal TCP spør om ny sending helt til korrekt data er overført [26].
SQL	Står for Structured Query Language og er et standardspråk som brukes for å aksessere og manipulere databaser [27].
SQLite	SQLite er et C-basert bibliotek som implementerer en enkel, liten og rask SQL database. SQLite er den mest brukte databasen i verden [28].
Visual Studio	Integrert utviklingsmiljø (IDE) for Microsofts .NET plattform [29].
IDE	Integrated Development Environment
Autoklavering	En type trykkoker som brukes for å sterilisere laboratorieutstyr.
Windows IoT core	Windows IoT core er en versjon av windows som er optimalisert for mindre enheter med begrenset prosessorkraft. Den er blant annet optimalisert for Raspberry pi 2 og 3 [30].
Python	Python er et eget programmeringsspråk
Rpi	Forkortelse for Raspberry Pi
RTC	Står for Real Time Clock.
GUI	Graphical User Interface (Brukergrensesnitt)
NTU	Måleenhet for turbiditet. Står for Nephelometric Turbidity Unit.
PVC	Står for Polyvinyl Chloride og er en form for isolasjon [31].
Spectrophotometer	Et apparat som måler densitet i en væske ved å se på hvor mye lys som blir absorbert når det blir sendt gjennom væsken.
Kuvette	En liten glassbeholder laget for å kunne sende lys gjennom væsken som er i den.

Appendiks C Prosjektledelse og styring

C.1 Prosjektorganisasjon

Vi hadde ikke særskilte roller fra begynnelsen av, men disse ble naturlig dannet underveis. Dette skjedde blant annet fordi folk ble sittende med hvert sitt arbeid, hvor det var mye å sette seg inn i. Det ble dermed naturlig at dem som hadde opparbeidet seg mest kunnskap innen det som skulle jobbes med, ble sittende med det. Alle i prosjektet har samarbeidet på alle de ulike områdene, men ansvar er fordelt følgende:

- Ansvarlig for fysisk system: Martin Græsdal
- Ansvarlig for programkode: Audun Møgster
- Ansvarlig for rapport/dokumentasjon: Anders Kommedal

C.2 Fremdriftsplan

#	Aktivitet	Planl. Arbeid [timer]	Utført Arbeid [timer]	Start Dato	Slutt Dato	Fram drift	Ansvarlig	AG
1	Forstudie arbeid	150	36	21/1	31/1	100 %	Gruppen	
2	Forstudie innlevering	0		31/1		100 %	Gruppen	
3	Få sensorer og annet utstyr opp å gå	30	5	15/1	28/2	25 %	Gruppen	Vil avhenge av leveringstid på de ulike delene
4	Sette oss inn i ny software	80		1/2	31/3	0 %	Gruppen	
5	Bygging av tank med sensorer	80		15/2	31/3	0 %	Gruppen	
6	Programmering	700	12	15/1	15/5	5 %	Gruppen	
7	Oppbygging av GUI	300				0 %	Gruppen	
8	Bachelor oppgave arbeid	800	36	15/1	31/5	5 %	Gruppen	absolutt frist
9	Midtveis presentasjon	20		1/4	5/4	0 %	Gruppen	avtales med veileder
10	Bachelor oppgave seminar/ Siste UV dag	0		10/5		0 %	Gruppen	
11	Ordinær eksamen	0		13/5	11/6	0 %	Gruppen	
12	Konte eksamen	0		5/6	11/6	0 %	Gruppen	
13	Bachelor oppgave innlevering	0		31/5		0 %	Gruppen	absolutt frist
14	Bachelor oppgave presentasjon	0		5/6	12/6	0 %	Gruppen	avtales med veileder
15	EXPO / Avslutningsfest	0		13/6		0 %	Gruppen	alle må stille
		2160	89					

Figur 42 - Fremdriftsplan vol.1

#	Aktivitet	Planl. Arbeid [timer]	Utført Arbeid [timer]	Start Dato	Slutt Dato	Fram drift	Ansvarlig
1	Forstudie arbeid	150	59	21/1	31/1	100 %	Anders
2	Metode undervisning	25	25	7/1	11/6	100 %	Gruppen
3	Møtevirksomhet	15	25,5	7/1	25/5	100 %	Gruppen
3	Forstudie innlevering	0	0	31/1	31/1	100 %	Anders
4	Bestilling utstyr	3	6,5	31/1	29/02	100 %	Martin
4	Konfiguering av Raspberry PI	10	16	21/1	25/1	100 %	Audun
5	Oppkobling og testing av temperatur sensor	15	4	7/1	11/1	100 %	Gruppen
6	Oppkobling og testing PH sensor	15	16	15/1	28/2	100 %	Gruppen
7	Oppkobling og testing av lys sensor	7	4	18/2	31/3	100 %	Gruppen
8	Oppkobling og testing av turbiditet sensor	15	14	25/2	8/3	100 %	Gruppen
9	Sammensetting og testing av trådløskommunikasjon	20	24	15/2	31/3	100 %	Audun
10	Planlegging og Bygging av tank med sensorer	30	46	25/2	1/3	100 %	Martin
11	3D-design av lokk	60	62	25/2	7/4	100 %	Martin
12	Planlegging og design av målekrets	24	29	31/1	14/2	100 %	Martin
13	Lodding og konstruksjon av målekrets	15	40	8/4	26/4	100 %	Gruppen
14	3D-printing og tilpasning av deler	10	30	8/4	26/4	100 %	Martin
15	Planlegging av GUI-design/brukergrensesnitt	40	94	25/3		100 %	Audun
17	Design av GUI for lagrede dataverdier	40	25,5	12/4	17/5	100 %	Anders
18	Design av GUI for hovedvindu	30	36,5	12/4	17/5	100 %	Audun
19	Datalagring/ Datainnsamling	12	14	12/4	15/5	100 %	Anders
20	Sammensetting og testing av hele systemet	110	76	1/5	29/5	100 %	Gruppen
21	Bachelor oppgave arbeid	300	186,5	11/2	31/5	100 %	Anders
22	Midtveis presentasjon	20	41	1/4	5/4	100 %	Gruppen
23	Bachelor oppgave seminar/ Siste UV dag	4	4	13/5	11/6	0 %	Gruppen
24	Ordinær eksamen	0	0	5/6	11/6	0 %	Gruppen
25	Konte eksamen	0	0	31/5		0 %	Gruppen
26	Bachelor oppgave innlevering	0	0	5/6	12/6	0 %	Gruppen
27	Bachelor oppgave presentasjon	10	10	12/6		0 %	Gruppen
28	EXPO / Avslutningsfest	12	12	13/6		0 %	Gruppen
		992	888,5				

Timetallene for forstudie og bachelor oppgaver satt for hhv 1,2,3 og 4 studenter, antar ca minimum 25 timer arbeid pr uke.

Figur 43 - Revidert fremdriftsplan

Første fremdriftsplan var svært grov, med urealistisk timeantall. Det er blitt justert og korrigert på siste versjon.

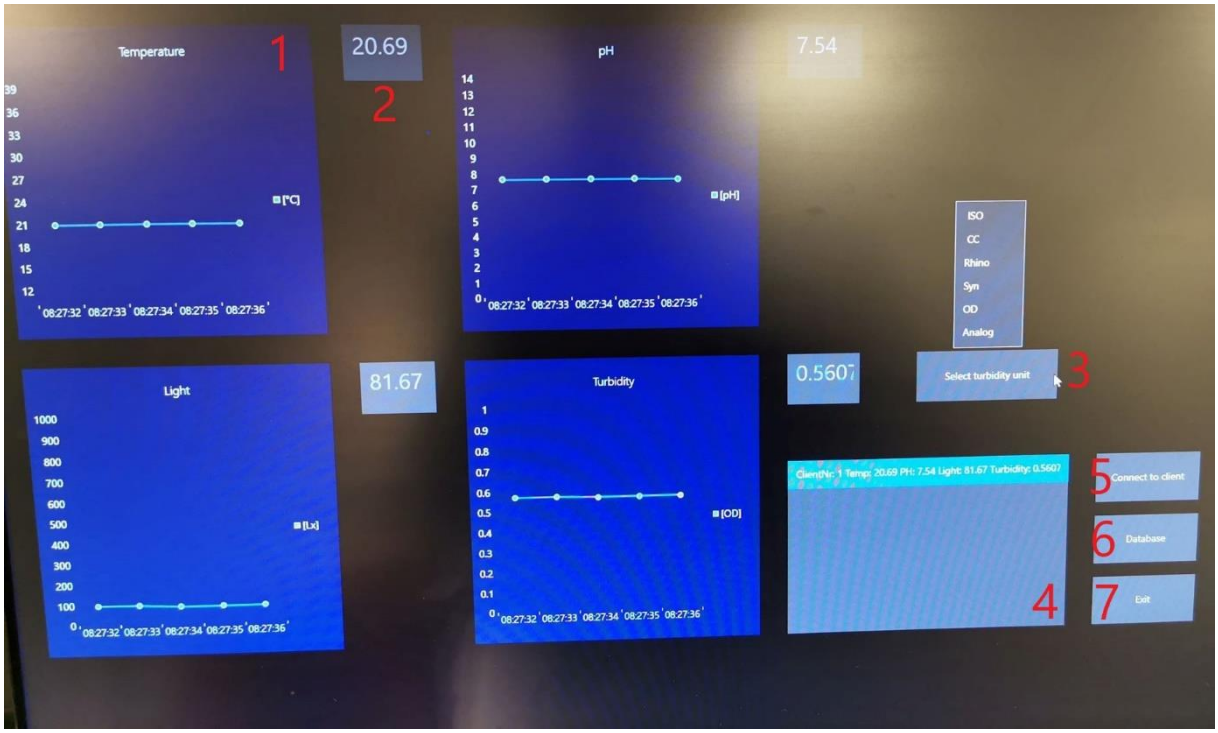
C.3 Risikoliste

Risiko	Sannsynlighet	Alvor	Konsekvens	Tiltak
Sykdom	Lav	Middels	Kan gå utover antall timer	Omprioritere timebruk mot slutten, eventuelt jobbe helger/kvelder.
Uenighet/konflikt med oppdragsgiver	Lav	Høy	Kan gå utover arbeidsforhold, og kan medføre redusert kvalitet i arbeidet	Prøve å inngå en løsning begge parter kan godta.
Uenighet/konflikt i gruppen	Lav	Høy	Kan gå utover kvalitet i arbeidet, og arbeidsmoral	Prøve å inngå en løsning mellom gruppedeltakere.
Deler blir levert for sent	middels	middels	Kan gå utover sluttproduktet.	Låne deler fra skolen og/eller oppdragsgiver. Bytte ut når deler eventuelt ankommer.
Deler blir ødelagt og må erstattes eller som ikke fungerer som forventet.	lav	middels	Kan gå utover sluttproduktet, kan medføre nye ventetider for nye deler.	Være nøye under testing/bygging av systemet slik at en ikke ødelegger komponentene. Låne nye deler dersom skaden skjer.
Beregnet for liten tid til arbeidet	middels	middels	Kan gå utover sluttprodukt/resultat.	Justere arbeidstimer underveis.

Appendiks D Brukerdokumentasjon

Her vil vi gi en generell forklaring på bruken av programmet.

D.1 Hovedprogram

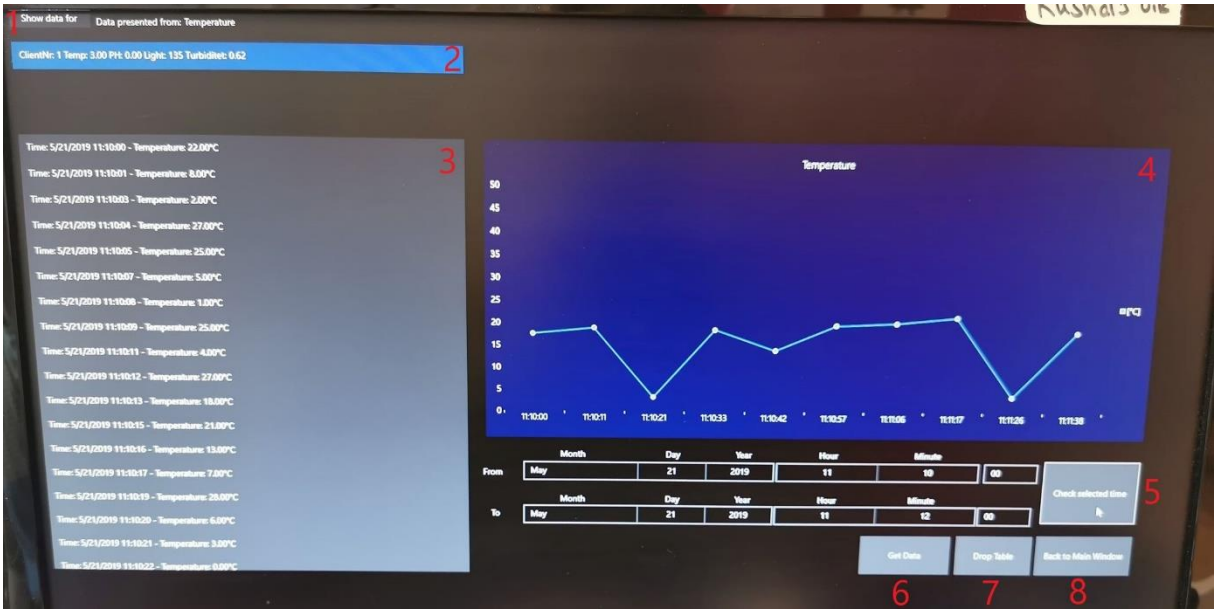


Figur 44 - Hovedvindu med dynamiske grafer.

Hovedprogrammet består av fire grafer som angir de kontinuerlig målte dataverdiene. De røde tallene er ikke en del av programmet, men angir de ulike delene som forklares under. Verdiene som viser på bildet er simulerte.

1) Dynamisk graf som viser verdiene av de målte dataverdiene. Disse kan også klikkes på, for å komme til en forstørret graf som viser kontinuerlig måling av den valgte dataverdien. Da åpnes et nytt vindu som er forklart under D.3 .
2) Den siste verdien målt. Det er også det siste punktet på grafen som vises.
3) «Roll-down» meny hvor en kan velge hva man ønsker å se på den fjerde grafen. Da kan en velge mellom de fire forskjellige algene (viser tetthet i celler/ml), eller Optical Density (OD, som er et tall for hvor mye lys som er absorbert), eller den analoge innverdien på arduinoen.
4) Liste over klientene som er koblet til. Dersom flere klienter kobler seg til, vil de legges til under den blå linjen. Viser også streng med data av alle de fire verdiene.
5) Tilkoblingsknappen for å koble til en klient. Dersom klienten søker etter serveren, vil denne knappen sørge for at serveren finner klienten, og kobler seg til. Klikk en gang, og vent til klienten er funnet. Husk at klient og server må være tilkoblet samme nettverk, og at klienten må bruke serverens IP-adresse.
6) Klikk på denne knappen for å ta deg til datalagringsvinduet. Åpner nytt vindu som står forklart under D.2 .
7) Klikk på denne for å avslutte programmet.

D.2 Datalagringsvindu



Figur 45 - Datalagringsvindu

Dette er vinduet som åpnes ved å trykke på «Database»-knappen i hovedprogrammet. På bildet brukes simulerte verdier.

1) «Roll-down menu». Dersom en klikker på denne, rulles det ned en meny hvor en kan velge mellom «All sensors», «pH», «Temperature», «Turbidity» og «Light». Alt etter hva en velger, vil da vise en dynamisk graf av den valgte enhetens dataverdier. Dersom en har klikket på «Get Data», vil også dataverdiene vise i listen til venstre for den aktuelle enheten. Dersom «All sensors» er valgt, vil turbiditeten vise på grafen.
2) Liste som viser de tilkoblede klientene.
3) Liste som viser dataene som er lagret i databasen. Vil vise data for den valgte enheten fra «Roll-down» menyen. Denne listen kan også skrulles i.
4) Dynamisk eller statisk graf. Viser enten dynamiske verdiene fra den kontinuerlige målingen, ellers kan den vise statiske verdier mellom valgte tidspunkt.
5) Ved å velge dato/klokkeslett mellom to valgte tidspunkt, for så å trykke «Check selected time», vil grafen bli statisk og den vil vise endringen i verdiene mellom de valgte tidspunktene. Dersom en velger ugyldige tidspunkt, vil den forrige grafen fortsatt vises.
6) Klikk for å hente inn alle dataverdiene som er lagret i databasen. Dersom «All sensors» er valgt, vil den vise en string med alle verdiene, ellers vil den vise verdier for den valgte sensoren.
7) Klikk for å «droppe» tabellen som er lagret i databasen. Dette kan typisk gjøres mellom to ulike målinger, av to ulike algekulturer. Dersom det ikke er interessant å ha begge algekulturene lagret i tabellen, kan den gamle tabellen slettes. Det vil da lages en helt ny tabell, og den vil fortsette å lagre verdier i denne.
8) Klikk for å komme tilbake til hovedvinduet.

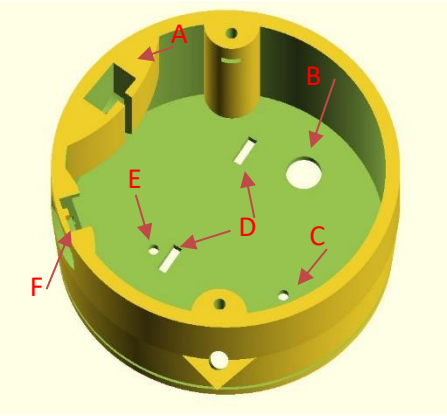
D.3 Forstørret grafvindu for valgt sensor

Figur 46 - Forstørret grafvindu

Dersom en klikker på en av grafene i hovedvinduet vil en lignende graf komme opp. Her er det en forstørret variant av grafen fra hovedvinduet. For å komme tilbake til hovedvinduet klikker en på «Back to main window»-knappen. På bildet brukes simulerte verdier.

D.4 Montering av målesystemet

Monteringen av målesystemet kan være litt kronglete når hele systemet skal passe ned i lokket. Derfor har vi laget en liten guide for å gjøre jobben litt lettere.

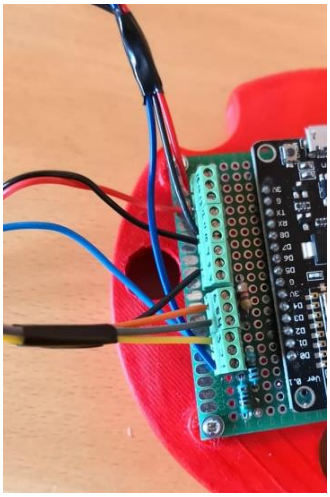


Figur 47 - Komponentplassering

1. Sett på plass USB-porten, pH-sonden, temperatursensoren og turbiditetsfestet. USB-porten skal på A, pH-sonden skal i B, temperatur i C og turbiditetsfestet i D. Tre turbiditetssensoren gjennom hullet E. Sørg for å ha nok kabel slik at turbiditetssensoren kan monteres på det nederste nivå på festet. Bruk silikon og tett godt rund hullene.

2. Monter kortene på de to nivåene. Hovedterminalene på mikrokontroller-kortet skal peke mot kabelgjennomføringen på førstenivået. Kontakten til pH-sonden skal stikke inn mot midten av nivået, og skruterminalene på turbidetskortet skal peke mot kablegjennomføringen.

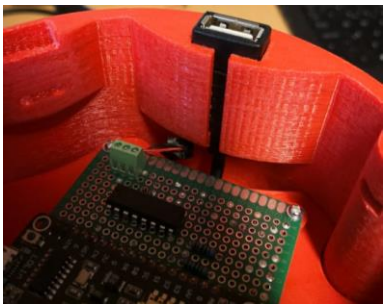
Figur 49 - Montering av kort



Figur 50 - Koble til internkoblinger

3. Koble til interne kabler for lyssensor, pH-sensoren og turbiditetssensoren i henholdt til koblingsskjemaet. Bruk relevante farger for å kunne skille lederne etter å ha tredd dem gjennom til 2 nivå. Lyssensor og pH-sensor skal ha Femal-connector på sensorsiden, for turbiditeten bruke kabel som fulgte med turbiditetskortet, med egen kontakttype på sensorssiden.

4. Tre kablene til turbiditetssensoren og temperatursensoren gjennom kabelgjennomføringen i nivå 1, og pH-kabelen gjennom pH-hullet. Før du fører nivået ned i lokket, kobles temperatursensoren til kortet iht. koblingsskjemaet.



Figur 51 - Spenningskilde

5. Koble til spenningskilde. Denne kabelen kan være litt vrien å få tak i, så her kan det være lurt å bruke en pinset for å fiske opp kabelen.

6. Tre alle kabler gjennom hullene i nivå 2 og plasser det i lokket.

7. Monter lyssensoren i «fuglekassen» (se Figur 47, plassering F). Koble alle sensorene etter koblingsskjema. Plugg koblingen til pH-sonden i kontakten, og lag en liten kveil.

Figur 52 - Ferdig montert krets

Figur 53 - Ferdigmontert system

8. Sett på lokket og skru i skruene. Monter turbiditetssensoren på rett nivå i forhold til nivået i kolben. Sørg for at sensoren ikke senkes ned i mediet, men pass også på at hele splitten i sensoren er dekket.

Appendiks E Drifts- og vedlikeholdsdokumentasjon

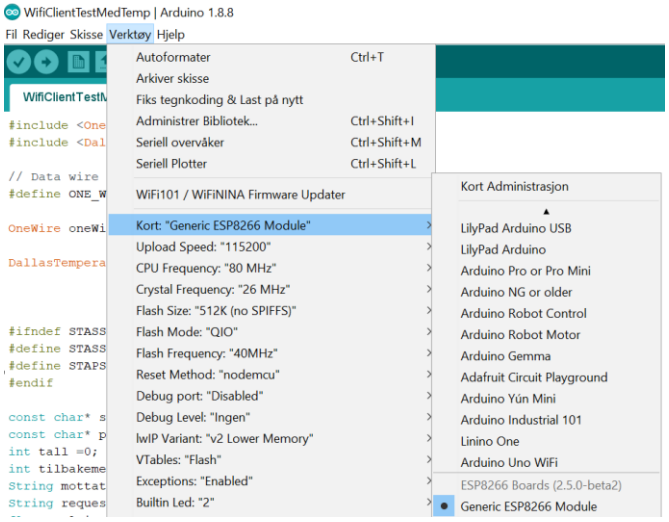
Det var et ønske fra oppdragsgiver at vi skulle lage setup guider til de ulike komponentene og programmet i seg selv. Appendiks F inneholder derfor disse. Noen av guidene er hentet fra allerede ferdige guider, og er blitt oversatt og utdypet til vårt bruk.

E.1 Mikrokontroller ESP8266

ESP8266 er kompatibel med Arduino IDE. For å kunne bruke ESP8266 i programmet må en gjøre noen innstillinger.

1. I Arduino IDE under verktøy fanen kan en velge hvilket kort en skal bruke. Her må en velge Generic ESP8266 Module. Hvis en ikke finner kortet i listen kan en installere det fra kort administrasjon. I kort administrasjonen søker en ESP8266, velger deretter den som heter esp8266 by esp Community.

2. For å laste opp programmet til kortet må en velge nodemcu i Reset Method (se Figur 54).

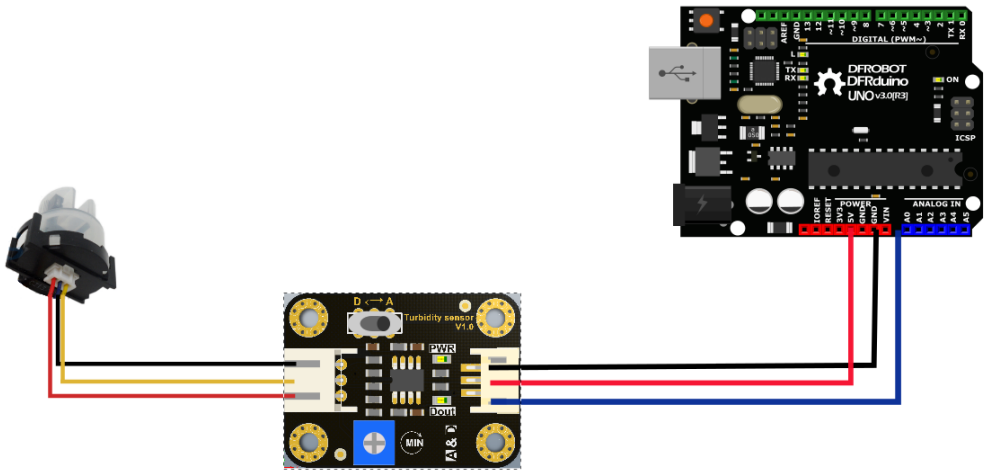


Figur 54 - ESP8266 setup

E.2 Turbiditetssensor

Denne guiden er laget med inspirasjon fra wiki.dfrobot.com [4].

Turbiditetssensoren skal kobles som følgende (Bildet illustrerer tilkobling med annet sensorkort, men gjelder de samme portene for ESP8266):



Figur 55 - Koblingsskjema

Tilkoblinger:

Mellom sensorkortet og mikrokontrolleren vil det være pluggtilkobling på sensorkortsiden. En må derimot være nøye med å koble riktig inn på mikrokontrolleren.

Rød leder ➔ Kobles til **VV** på ESP8266.

Sort leder ➔ Kobles til **GND** på ESP8266.

Blå leder ➔ Kobles til **A0** på ESP8266.

Mellom sensorkortet og turbiditetssensoren er det opprinnelig plugg på begge sider, altså ingen fare for å koble feil.

Turbiditetssensoren kan justeres på sensorkortet på følgende måter:

«A» ➔ Analog modus. Gir et analogt signal ut. Jo mer grums som er i vannet, jo lavere vil den analoge outputverdien bli.

«D» ➔ Digital modus. Gir kun høy eller lav, alt etter om den passerer en satt grenseverdi.

Potensiometer ➔ Endrer grenseverdien til den digitale modusen.

Eksempel analog modus:

```
void setup() {
  Serial.begin(9600); //Baud rate: 9600
}
void loop() {
  int sensorValue = analogRead(A0);// read the input on analog pin 0:
  float voltage = sensorValue * (5.0 / 1024.0); // Convert the analog reading (which goes from 0 - 1023) to a voltage
  Serial.println(voltage); // print out the value you read:
  delay(500);
}
```

Figur 56 - Analog modus turbiditet

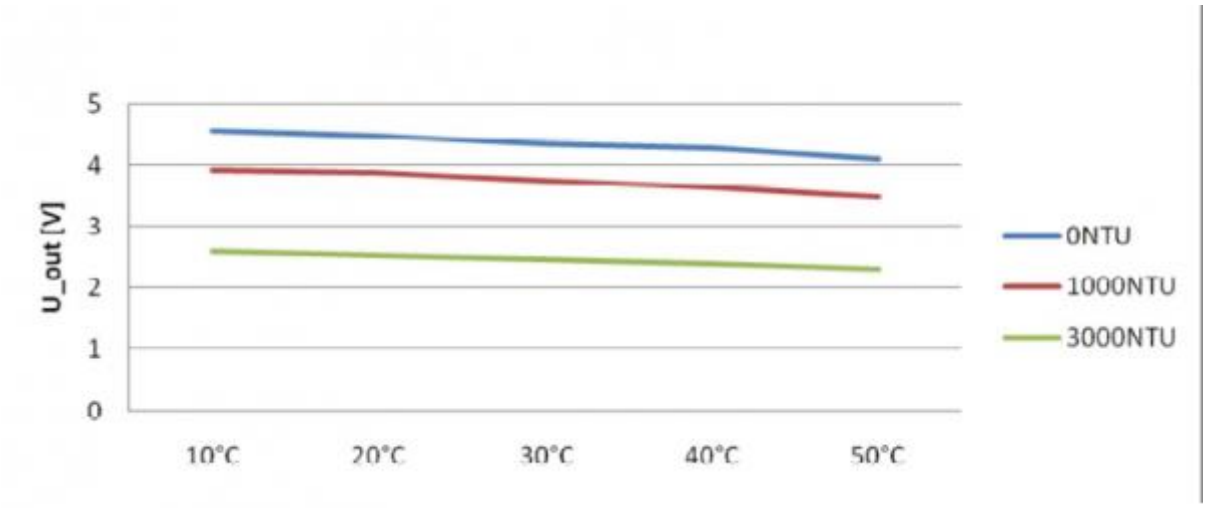
Kodesnutten over er et eksempel for å få turbiditetssensoren opp å gå i analog modus. Den gir ut en spenningsverdi mellom 0-5V. Dersom den skal brukes med ESP8266 må spenningsdeler brukes (se Appendiks E.3 for spenningsdeler), for å unngå for høy spenning på analog inngangen til mikrokontrolleren.

Eksempel på digital modus:

```
int ledPin = 13;                // Connect an LED on pin 13, or use the onboard one
int sensor_in = 2;                 // Connect turbidity sensor to Digital Pin 2

void setup(){
  pinMode(ledPin, OUTPUT);      // Set ledPin to output mode
  pinMode(sensor_in, INPUT);       //Set the turbidity sensor pin to input mode
}

void loop(){
  if(digitalRead(sensor_in)==LOW){       //read sensor signal
    digitalWrite(ledPin, HIGH);   // if sensor is LOW, then turn on
  }
  else{
    digitalWrite(ledPin, LOW);    // if sensor is HIGH, then turn off the led
  }
}
```

Figur 57 - Digital modus turbiditet

Kodesnutten er et eksempel hvor en bruker grenseverdien til å slå på et lys dersom verdien blir for høy. På samme måte kan det brukes til å for eksempel avbryte en prosess dersom væsken blir for grumsete.

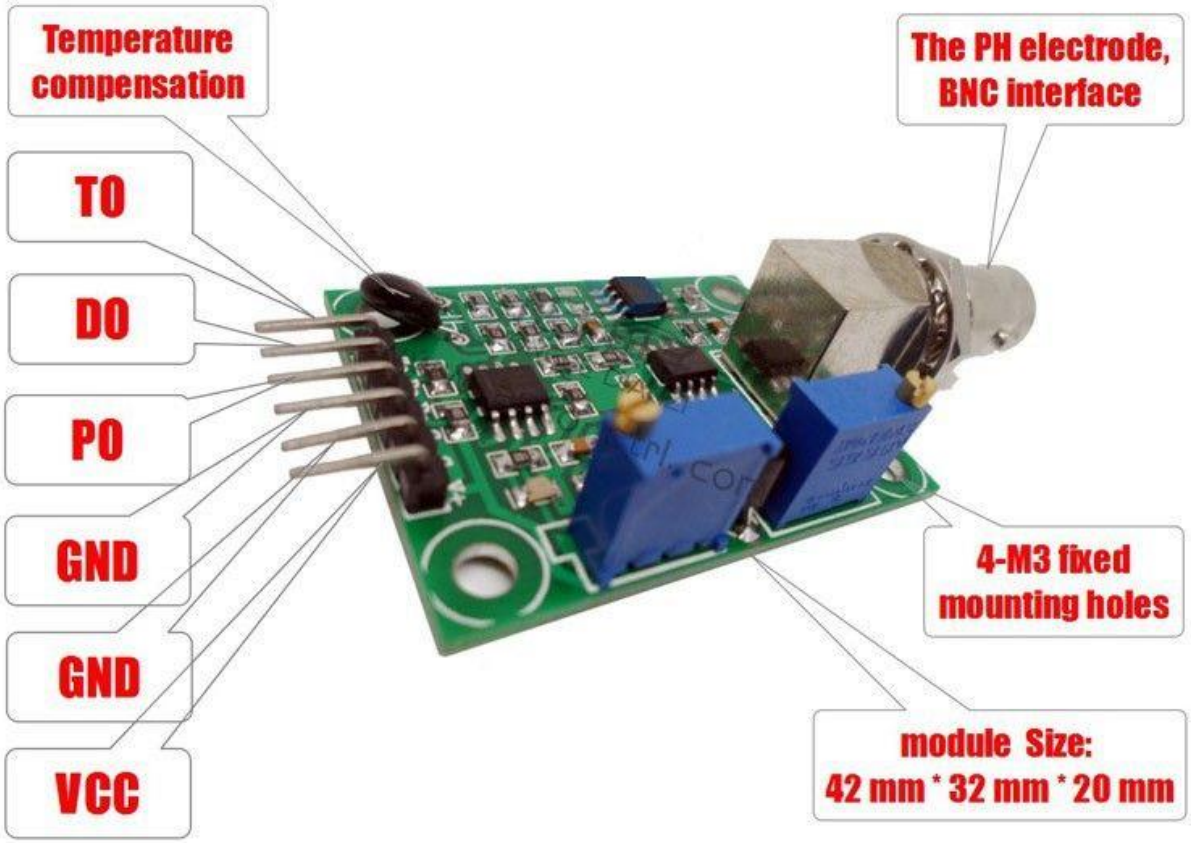


Figur 58 - Drifting pga. temperaturendring

Temperaturendringer vil påvirke turbiditetssensoren. Dersom en måler høye turbiditeter er endringen nokså lav. Men dersom en kun måler ørsmå endringer, vil disse kunne drifte ganske mye basert på temperaturendringene.

E.3 PH-Sensor

Her er en liten setup guide mellom PH-proben E201-C med kortet PH-4502C og NodeMCU ESP8266. Guiden er laget med inspirasjon fra botshop.co.za [5].



Figur 59 - PH-kortet [5].

PH-kortets (PH-4502C) tilkoblinger:

TO – Temperatur output (brukes ikke)

DO – 3.3V output (justeres med PH-limit potensiometer)

PO – PH analog output ➔ kobles til **A0** på ESP8266

G – Gnd for PH-proben ➔ kobles til **G** på ESP8266

G – Gnd for kortet ➔ kobles til **G** på ESP8266

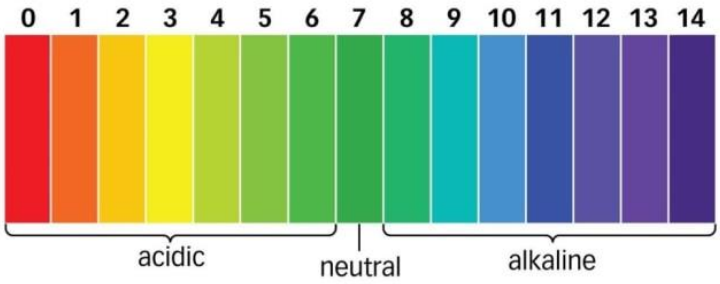
V+ – 5V DC ➔ kobles til **VV** på ESP8266

Kortets Potensiometre:

POT 1 – For å justere offset på analog utgang (Plassert nærmest BNC tilkoblingen).

POT 2 – For å justere grensen på 3.3V output.

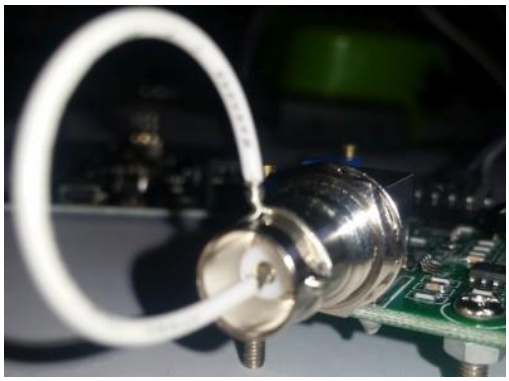
Justere offset potensiometeret (POT 1):



Figur 60 - PH-tabell [42].

PH-verdiene ligger mellom PH 0-14, hvor 0-6 er syrlige PH-verdier, mens 8-14 er basiske.

Ideelt sett ville det vært ønskelig at PH 0 er representert med 0V og at PH 14 representeres med 5V. Men kortets standardinnstillinger er ikke dette tilfellet. Dersom en bruker standardinnstillingene, vil nemlig PH 7 være representert som 0V. Dermed vil PH 0-6 være representert som negative spenninger, noe som ikke er mulig å lese av på ESP8266. Derfor må en justere på **potensiometer 1,** slik at PH 7 representeres som 2.5V ut fra **PO.** Vi fant ut at enkleste måte å gjøre dette på, var ved å koble til multimeter, å måle mellom **PO** og **G.**



Figur 61 - Kortsluttet BNC-tilkobling [5].

For å simulere PH 7, kan en kortslutte BNC tilkoblingen ved å ta en leder fra innsiden av BNC-tilkoblingen, til utsiden som vist på bilde. Når dette er gjort, vil multimeteret vise hvilken spenning som representerer PH 7. Juster **potensiometer 1** helt til multimeteret viser 2.5V.

Eventuelt kan følgende kodesnutt brukes:

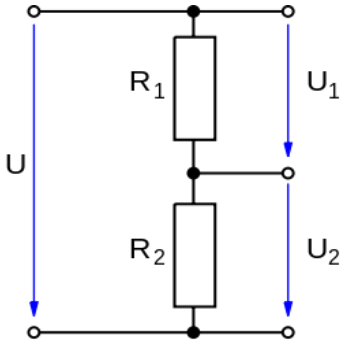
sketch_may24a §

```
void setup() {
 // initialize serial communication at 9600 bits per second:
 Serial.begin(9600);
}

// the loop routine runs over and over showing the voltage on A0
void loop() {
 // read the input on analog pin 0:
 int sensorValue = analogRead(A0);
 // Convert the analog reading (which goes from 0 - 1023) to a voltage (0 - 5V):
 float voltage = sensorValue * (5.0 / 1023.0);
 // print out the value you read:
 Serial.println(voltage);
 delay(300);}
```

Figur 62 - Kodesnutt for å sette potensiometerverdi

NB! Avhengig av hvilket kort som brukes sammen med PH-proben, vil det variere hvilken spenning den analoge inngangen **A0** maksimalt kan ta. Dersom kortet ikke tar opp til 5V, må en spenningsdeler brukes.

Formel for spenningsdeler: $U_{ut} = U_{inn} * \left(\frac{R2}{R1+R2}\right)$



Figur 63 - Spenningsdeler [43].

Hvor U_{ut} blir spenningen som går inn på analog inngang **A0** på kortet.

(U_2 på figuren representerer U_{ut} i formelen, U representerer U_{inn}).

Vi prefererte å bruke multimetermetoden. Dette fordi:

- ESP8266-kortet vi brukte skal maksimalt ha 3.3V inn på den analoge inngangen **A0**. Med kodesnutten over er det antatt at 5V er signalet som sendes inn på **A0**. Dette kan naturligvis endres til at det er ca. 3.3V som sendes inn, men vi opplevde det som noe ustabilt i forhold til multimetermåling. I tillegg vil dette variere i forhold til hvilket kort en bruker.
- Ved å bruke multimeter kan man enkelt måle både spenning over PH-sensoren i seg selv, og etter målebroen (spenningen inn på **A0**).

PH-grense potensiometer (POT2):

Potensiometer 2, plassert lengst fra BNC kontakt, brukes for å sette en viss grense for PH-målingen. PH-kortet vil da gi ut 3.3V på utgang **DO**, helt til den satte grensen overstiges. Da vil spenningen på **DO** gå til 0V, og et rødt LED vil lyse på kortet. Denne kan dermed fungere som en slags alarm dersom PH-sensoren når en viss verdi.

Koble til og å kalibrere PH-sensoren:

Når offseten er satt, skal det ikke være nødvendig å endre på denne igjen. Da er det på tide å sette opp/kalibrere PH-sensoren.

Viktig informasjon om PH-proben:

1. Probens avlesning vil endres over tid, og må normalt sett kalibreres på nytt i ny og ne.

2. For å kalibrere proben trengs minst en referanseløsning med kjent PH-verdi. Dersom en bruker én løsning, bør dens PH-verdi ligge så nært målingene en skal gjøre med proben som mulig. Om mulig, bruk to løsninger med forskjellige PH-verdier som ligger rundt målingene som skal gjøres. Vi brukte en løsning på PH 7, og en på PH 10.01.

Figur 64 - PH-løsninger brukt for kalibrering av PH-proben [44].

3. PH-proben bruker noe tid på å komme frem til riktig PH-verdi. En bør derfor under måling la proben ligge i løsningen i rundt 2-3 minutter til den stabiliserer seg. Den vil da variere med 3-4 verdier på siste desimal (Eks. 6.84-6.88).

4. PH-verdier vil variere med ulike temperaturer. Riktignok vil ikke små variasjoner rundt romtemperatur påvirke i noen særlig grad, da den har en komponent som kompenserer for temperaturen i rommet.

Fjern så kortslutningen fra BNC-tilkoblingen, og bruk følgende arduinokode:

sketch_may24a §

```
float calibration = 0.00; //change this value to calibrate
const int analogInPin = A0;
int sensorValue = 0;
unsigned long int avgValue;
float b;
int buf[10],temp;
void setup() {
 Serial.begin(9600);
}
void loop() {
 for(int i=0;i<10;i++)
 {
 buf[i]=analogRead(analogInPin);
 delay(30);
 }
 for(int i=0;i<9;i++) //GJØR 10 MÅLINGER
 {
 for(int j=i+1;j<10;j++) //SORTERER MÅLINGENE
 {
 if(buf[i]>buf[j])
 {
 temp=buf[i];
 buf[i]=buf[j];
 buf[j]=temp;
 }
 }
 }
 avgValue=0;
 for(int i=2;i<8;i++) // Tar ut de seks mellomste verdiene
 avgValue+=buf[i];
 float pHVol=(float)avgValue*5.0/1024/6; //Tar gjennomsnittet av de //seks.
 float phValue = -5.70 * pHVol + calibration;
 Serial.print("sensor = ");
 Serial.println(phValue);

 delay(500);
}
```

Figur 65 - Kodesnutt for kalibrering PH-sensor

Om kodesnutten:

Koden tar 10 målinger, sorterer fra lavest til høyest verdi, finner de seks mellomste målingene og tar et gjennomsnitt av disse for å finne den mest presise verdien.

I følgende linje `float pHVol=(float)avgValue*5.0/1024/6;` vil det være aktuelt å bytte ut 5.0 med den spenningsverdien U_{ut} du får ved en eventuell spenningsdeler.

I linjen `float phValue = -5.70 * pHVol + calibration;` vil den negative verdien -5.70 muligens måtte endres, alt etter hvor responsiv PH-sensoren er. Calibration er en variabel som i utgangspunktet er satt til null.

Fremgangsmåte for kalibrering:

Først og fremst er det essensielt at ikke de to PH-referanseverdiene blir kontaminert. Dette kan skje ved at en hyppig tar proben fra ene glasset, til det andre, med de to ulike PH-verdiene i. Skyll av proben mellom hver måling, og rist av så mye væske som mulig. Jeg la også til kodebiten:

```
Serial.print("Spenning = ");

Serial.println(pHVol);
```

For å skrive ut spenningen, under koden som printer PH-verdien.

1. Sett proben i første løsning (f.eks. PH 7.0). La den stå til den stabiliserer seg. Les av PH-verdi (verdi på PH-løsningen du brukte) og spenningen inn på kortet. Sett så proben, etter at den er skylt/renset i løsning nummer to (f.eks. 10.01). Les av PH-verdi og spenningen. For å finne riktig verdi for skaleringen (verdien som er -5.7 i koden) kan en bruke følgende formel:

$$\frac{PH2 - PH1}{V2 - V1}$$

   Sett så denne verdien inn istedenfor -5.7 i koden.

2. Mål så PH7 løsningen. Finn avviket fra den målte PH-verdien og PH7. Sett dette avviket inn i variabelen calibration. Last opp ny kode, og se at måleverdien nå ligger på PH7. Verifiser med å måle PH10.01-løsningen, og juster dersom det fortsatt er store avvik.

Kodesnuttene som er brukt, finnes i klartekst i kilden. De er også brukt i koden som ligger i vedlegg.

E.4 Lyssensor BH1750

Her følger en setup-guide for lyssensoren. Guiden er fritt oversatt fra github/claws [32].

Lyssensoren bruker to biblioteker. Wire.h for å sette opp I2C protokoll og BH1750.h for å få tilgang til sensordataen.

Oppkobling:

BH1750	ESP8266
VCC	3V
GND	G
SCL	D3
SDA	D2
ACC	(Ikke i bruk)

1. Last ned bibliotek for sensoren. (https://github.com/claws/BH1750)

Klikk Clone or Download -> Download ZIP

2. Last inn biblioteket på Arduino IDE

Sketch -> Include library -> Add .ZIP library

Velg .zip-filen lastet ned fra punkt 1

Restart Arduino IDE

3. Legg inn kodesnutt som er vist på Figur 66

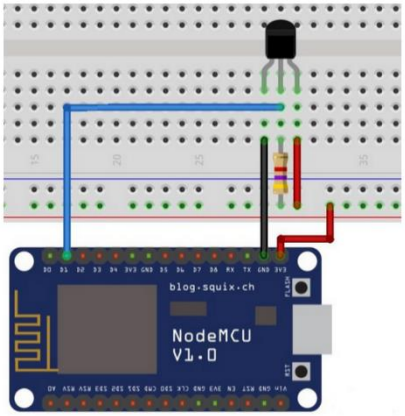
Figur 66 – Kodesnutt for lys sensor

E.5 Temperatur Sensor (DS18B20)

Her er en liten guide for å sette opp temperatur sensoren med esp8266. Guiden er laget med inspirasjon fra guiden fra RoboIndia [33].

Oppkobling

Bilde viser oppkoblingen mellom en DS18B20 temperatursensor og ESP8266 mikrokontroller. Sensoren er ikke vanntett, men det eksisterer en av samme type som er vanntett. Den har samme oppkobling. Motstanden er på 4.7 KΩ.

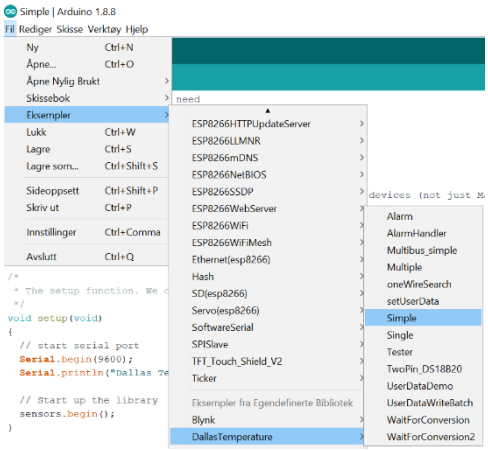


Figur 67 - Kobling for testkrets [45]

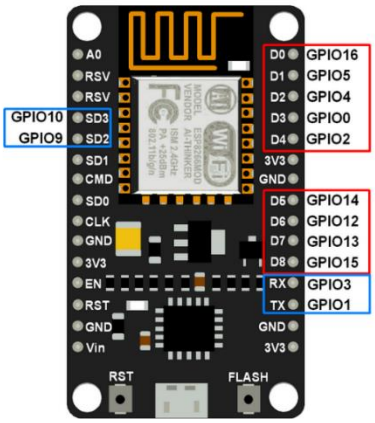
Programmeringsdel

For å kunne hente inn data, og vise det på den serielle overvåkeren, kan en bruke en av eksemplene i Arduino programmet. Se **Feil! Fant ikke referansekilden.**. **Feil! Fant ikke referansekilden.** viser hvilke in nganger en kan bruke.

Dette eksempelet bruker to biblioteker for å håndtere dataen fra temperatursensoren. DS18B20 er laget slik at en kan koble flere temperatursensorer over samme ledning. En bruker onewire.h biblioteket sammen med dallastempeture.h. Der onewire.h håndterer overføringen og dallastemperature.h håndterer hvilken sensor en skal hente data fra.



Figur 69- Åpning av eksempel for dallasTemperture



Figur 68 - Innganger tilgjengelig [46].

E.6 R-PI 3 B med windows 10 IOT Core

Denne guiden er laget med inspirasjon fra Richard Devine på nettsiden til windowscentral [30]:

Raspberry pi krever at en har et operativsystem. En kan installere Windows 10 IOT Core ved hjelp av et program som heter Windows 10 IOT Core dashboard. Last ned programmet fra linken her; https://docs.microsoft.com/en-us/windows/iot-core/connect-your-device/iotdashboard.

Obs! Per i dag så fungerer ikke dette operativsystemet på Rpi 3 B+. En kan bruke Rpi 3 B.

- En trenger et SD-kort som er class 10 og minst 8GB. Sett det inn i Pcen.
- Åpne Windows 10 IOT Core Dashboard. Klikk; «Set up new device».
- Gjør følgende; **Device type:** BroadComm [Raspberry Pi 2&3]
- **OS Build:** Windows 10 IOT Core (17763). Hvis det blir foreslått en høyere versjon en 17763, kan en velge det om ønskelig.
- **Device name:** Velg ønsket navn.
- New Administrator password: Velg ønsket passord.
- Confirm Administrator password: Bekreft det ønskede passordet.
- Trykk «download and install» nede til høyre.

Etter operativsystemet er lastet ned på SD-kortet kan en ta det i Raspberry pi og starte opp Windows 10 IOT Core.

E.7 Trådløs kommunikasjon

Arduino delen av denne guiden er laget med inspirasjon fra Pieter P på GitHub [34].

For å sette opp en C#-applikasjon fra pc eller RPI sammen med ESP8266, kan en bruke TCP/IP protokoll med Socket. Et eksempel på hvordan en kan overføre data fra ESP8266 til en enhet som bruker applikasjon laget med C# (UWP) er vist nedenfor.

Kode i C#

```
async void NewClientThread()
{

    Socket listeningSocket = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);


    IPEndPoint serverEP = new IPEndPoint(IPAddress.Parse(GetLocalIp()), 9050);

    try
    {
        listeningSocket.Bind(serverEP);

    }
    catch (Exception ex)
    {
        MessageDialog messageBox = new MessageDialog(ex.Message);
        await messageBox.ShowAsync();

    }

    listeningSocket.Listen(10);

    await Task.Run(() =>
    {

        Socket comSocket = listeningSocket.Accept(); // Blocking method

        listeningSocket.Dispose();


        ClientThread(comSocket);


    });


}
```

Figur 70 - Kode for ny klienttråd.

Dette er kode for lyttesokkelen. Det vil si at den lytter på nettverket etter en klient den kan koble seg opp mot. Hvis den finner en klient, så blir klienten en del av Threadpool.

```
public static string GetLocalIp(HostNameType hostNameType = HostNameType.Ipv4)
{
    var icp = NetworkInformation.GetInternetConnectionProfile();

    if (icp?.NetworkAdapter == null) return null;
    var hostname =
        NetworkInformation.GetHostNames()
            .FirstOrDefault(
                hn =>
                    hn.Type == hostNameType &&
                    hn.IPInformation?.NetworkAdapter != null &&
                    hn.IPInformation.NetworkAdapter.NetworkAdapterId == icp.NetworkAdapter.NetworkAdapterId);

    // the ip address
    return hostname?.CanonicalName;
}
```

Figur 71 - Automatisk innhenting av serverens IP-adresse.

GetLocalIP() blir brukt for å hente IP-adresse til pc-en slik at en slipper å gjøre det manuelt. Den returnerer da IP-adressen.

Dette er kode for metoden som blir brukt som en egen tråd for hver klient som blir koblet til. Det er her dataen til ESP8266 blir tatt imot. Den legger først til klienten i en liste av klienter. Den mottar så en streng fra kommunikasjonssocketen. Denne strengen sendes deretter videre til en metode som henter ut data for hver sensor. Dette skjer så lenge kommunikasjonssocketen er tilkoblet.

```
void ClientThread(object o)
{

    clientNr++;
    Client client = new Client(clientNr);
    clientList.Add(client);

    Socket comSocket = o as Socket;

    IPEndPoint clientInfo = (IPEndPoint)comSocket.RemoteEndPoint;
    IPEndPoint serverInfo = (IPEndPoint)comSocket.LocalEndPoint;

    int recv;

    byte[] data = new byte[1024];



    while (comSocket.Connected == true)
    {

        recv = comSocket.Receive(data);
        data = new byte[1024];
        recv = comSocket.Receive(data);

        textRecieved = Encoding.ASCII.GetString(data, 0, recv);
        StringManipulation(textRecieved, client);




    }


    clientList.Remove(client);

    comSocket.Dispose();

}
```

Figur 72 - Klienttråd

Kode i Arduino

Kode for å sende data fra ESP8266 er rammet inn med rødfarge og nummerert på figurene nedenfor. I arduinokoden må bruker sette opp ønsket nettverksnavn, nettverkspassord, ip-adresse til server (samme som den i C# koden) og port.

Biblioteket som blir brukt er ESP8266Wifi.h.

1. Implementering av biblioteket ESP8266Wifi.h.
2. Her skriver brukeren inn nettverksnavn og nettverkspassord til nettverket serveren er koblet opp mot. (STASSID «nettverksnavn», STAPSK «nettverkspassord».
3. Nettverksnavn og nettverkspassord blir lagt inn i variabler.
4. Bruker skriver inn adressen til server. Port nr. skal være 9050.
5. Deklarerer en klientklasse for å bruke senere i koden til trådløs kommunikasjon.
6. Kobler til nettverket og skriver ut status på den serielle overvåkeren.
7. Kobler til server.
8. Sender data til server med klassen nevnt i punkt 5.

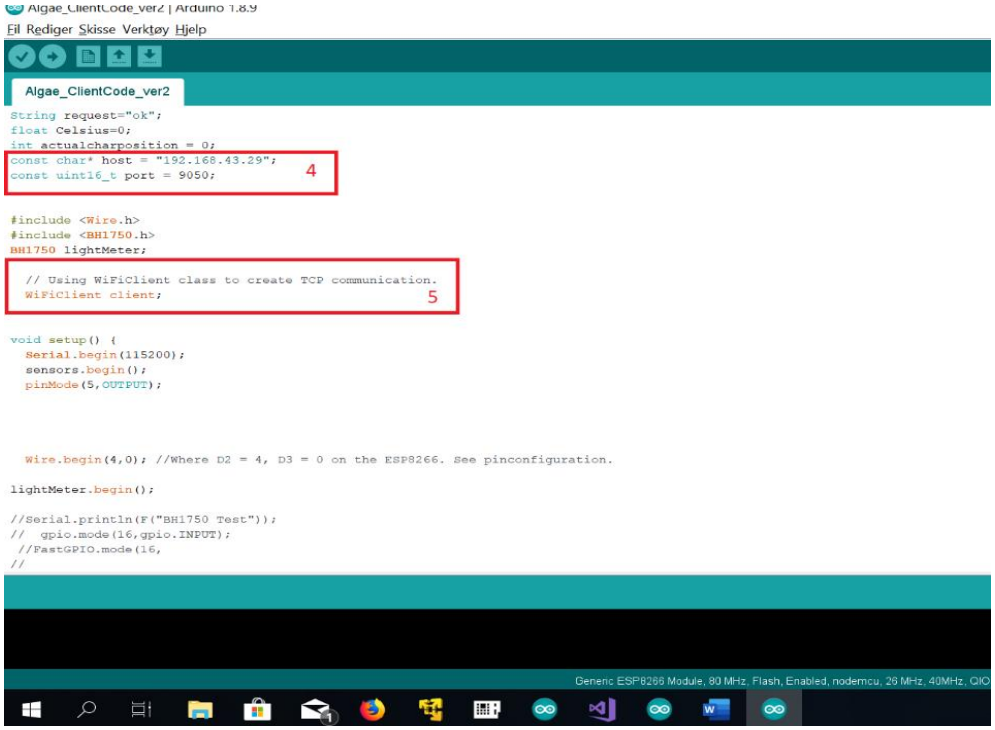
Algae_ClientCode_ver2 | Arduino 1.8.9

Fil Rediger Skisse Verktøy Hjelp

Algae_ClientCode_ver2 §

```
#include <ESP8266WiFi.h>                                   1
#include <OneWire.h>
#include <DallasTemperature.h>

// Data wire is plugged into pin D1 on the ESP8266 12-E - GPIO 5
#define ONE_WIRE_BUS 2

OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);

#ifndef STASSID                                            2
#define STASSID "AK"
#define STAPSK  "test1234"
#endif

//For PH-sensor ////////////////////////////////////////////////
float calibration = 7+15.77; //change this value to calibrate //
const int analogInPin = A0;
int sensorValue = 0;
unsigned long int avgValue;
float b;
int buf[10],temp;
float phValue, voltageTur, lux,ntu;
//For PH-sensor////////////////////////////////////////////////


String data = "";
const char* ssid     = STASSID;                            3
const char* password = STAPSK;
int tall =0;
```

Figur 73 - Arduino kodesnutt en



Figur 74 – Arduino Kodesnutt to

Fil Rediger Skisse Verktøy Hjelp

Algae_ClientCode_ver2 §

```
//FastGPIO.mode(16,
//
// Connecting to WiFi.
 WiFi.mode(WIFI_STA);                    6
 WiFi.begin(ssid, password);


Serial.println();
 Serial.println();
 Serial.print("Wait for WiFi... ");

 while (WiFi.status() != WL_CONNECTED) {
  Serial.print(".");
   delay(500);
 }

 Serial.println("");
 Serial.println("WiFi connected");
 Serial.println("IP address: ");
 Serial.println(WiFi.localIP());

 delay(500);

 Serial.print("connecting to ");
 Serial.print(host);
Serial.print(':');
Serial.println(port);
}
```

Figur 75 -Arduino kodesnutt tre

Fil Rediger Skisse Verktøy Hjelp

Algae_ClientCode_ver2 §

```
pH();
lightSensor();
turbidity();
data = "C" + (String)Celsius + "P" + (String)phValue + "L" + (String)lux + "T" + (String)ntu + "E";
Serial.println(data);
if(!client.connected())
{
  Serial.println("Try to connect...");                    7
  feedback=client.connect(host,port);


  if(feedback==1)
{

  Serial.println("connection succeded");
}

else
{
   Serial.println("connection failed");
}
}
```

```
 if(client.connected())                                    8
{
  client.print(data);
}
```

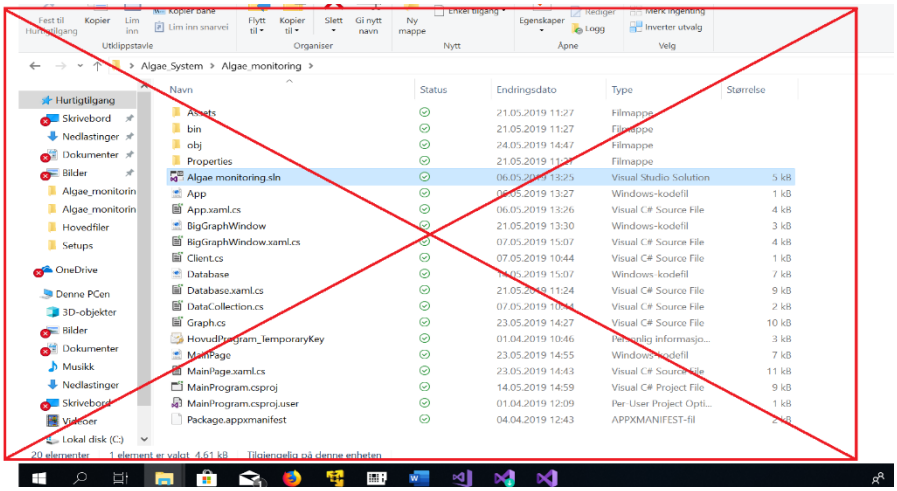
Figur 76 - Arduino kodesnutt fire

E.8 Prosedyre for å starte og bruke C# programmet

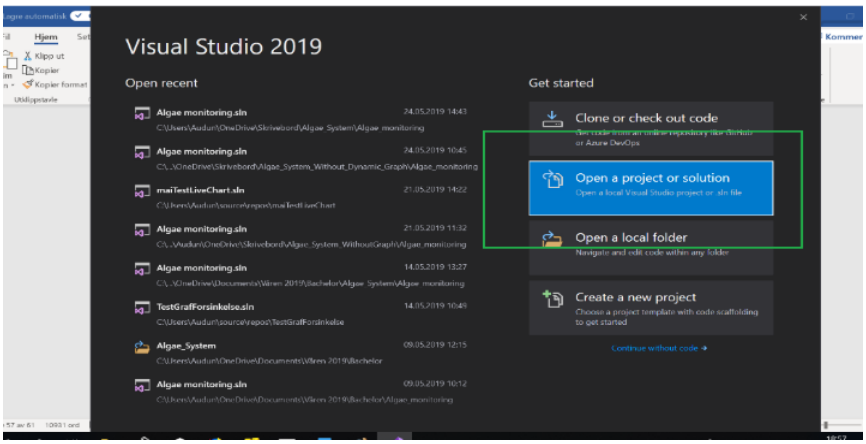
Dette er en prosedyre for å sikre at brukeren får åpnet c# programmet på rett måte. En kan enten bruke gratisversjonen av visual studio som heter visual studio community, eller bruke visual studio enterprise. Uansett så må versjonene våre av nyere dato, 2017 eller nyere.

En kan laste ned visual studio fra denne linken: https://visualstudio.microsoft.com/vs/.

Det er viktig at en åpner visual studio først og deretter henter inn programmet i Visual studio. Hvis en prøver å gå direkte inn i mappen til programmet og deretter åpner solution filen, kan en risikere mange feilmeldinger. Figur 77 går en direkte inn for så å åpne sln filen, dette er feil. **Feil! Fant ikke r eferansekilden.** åpner en først selve Visual studio applikasjonen, deretter åpne sln filen via; «open a project or solution». Dette er rett metode.

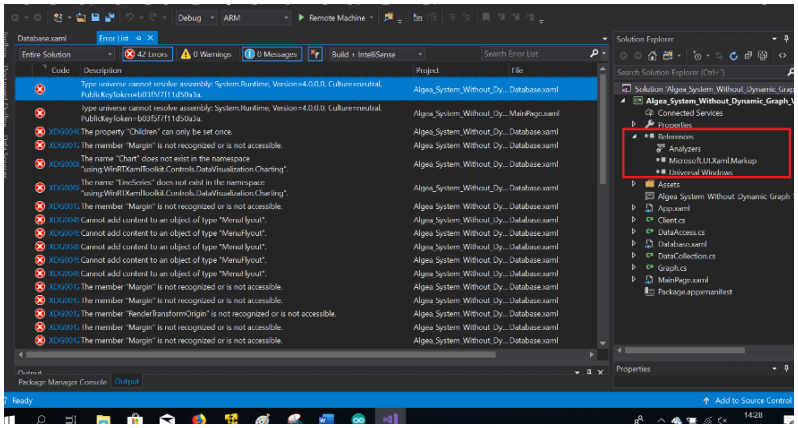


Figur 77 -program mappe

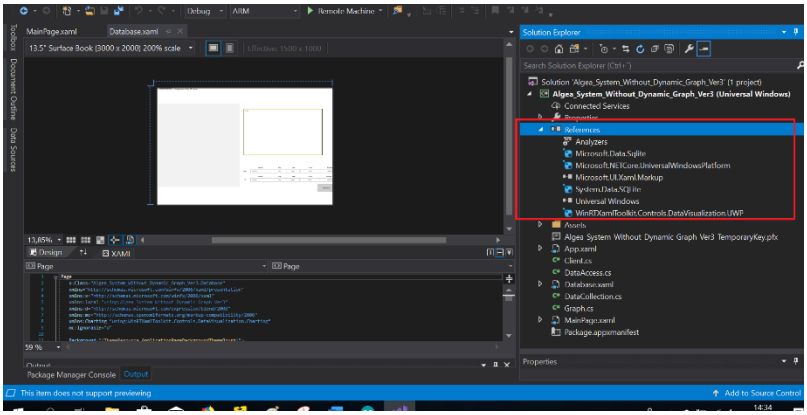


Figur 78- Visual studio app

Noen ganger klarer ikke Visual studio å åpne med referansene som er vist på Figur 79. Dette skjer typisk når en flytter på hele mappen og referansene klarer ikke å følge med. Da vil en få mange feilmeldinger. For å løse dette kan en høyre klikke på solution filen, trykke build solution, save all og deretter avslutte og starte programmet på nytt. Da skal de filene på Figur 80 komme opp.

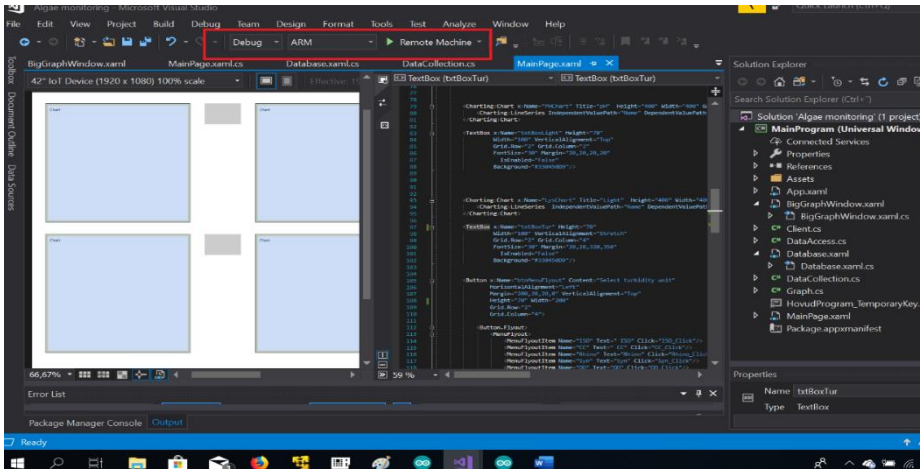


Figur 79 -Mangler referanser



Figur 80 - Referanser er implementert

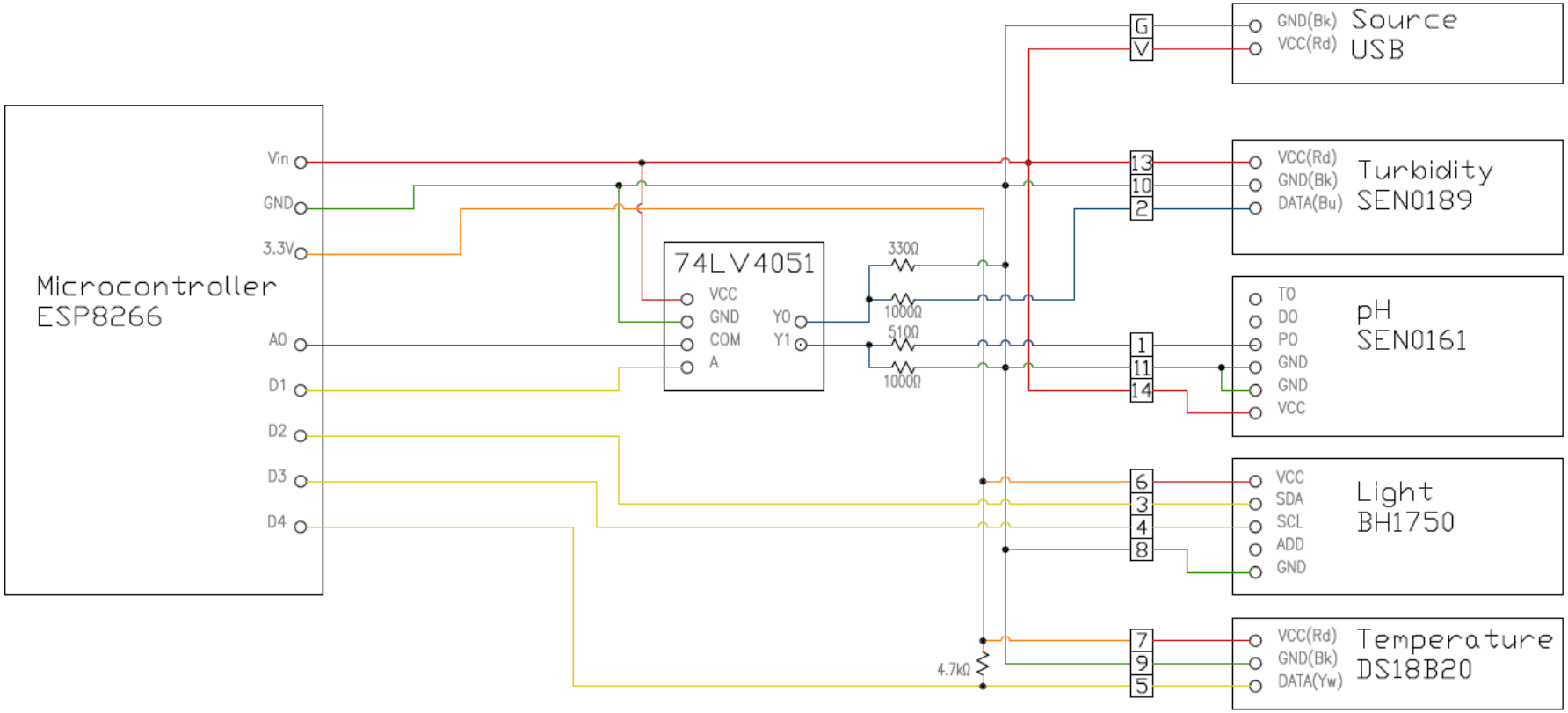
For å laste over applikasjonen fra pc til Windows 10 IOT Core trykker en på Remote Machine. Det er vist på Figur 81. Pc og enheten som kjører Windows 10 IOT Core må være på samme nett. Etter suksessfull overføring vil appen bli automatisk kjørt. Da kjører programmet i Debug mode, det vil si at en kjører programmet samtidig så pcen er forberedt på bugs. En kan velge release mode istedenfor debug for å løse pcen fra appen. Uansett så vil appen bli lagret på Windows 10 IOT Core.



Figur 81 – Overføring av app til Window 10 IOT Core

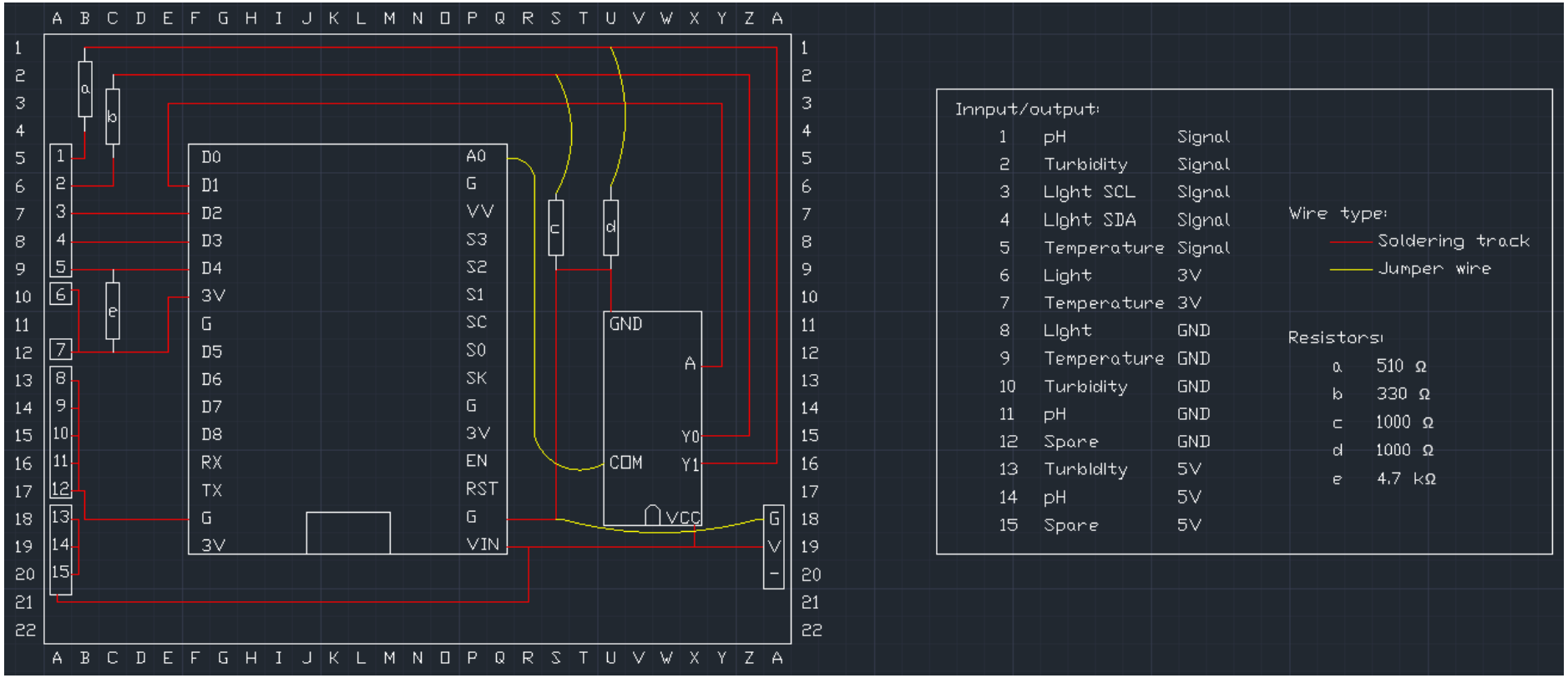
Appendiks F Kildekode, skjemadesign, Bill Of Materials mm

F.1 Kretstegning



Figur 82 - Kretstegning

F.2 Ruting for lodding kretskort



Figur 83 – Ruting for kretskort

F.3 Bill of Material

					Total (NOK)		Per unit (NOK)		
Product	Description	Bought From	Quantity	Unit	Price	Cost	Used	Cost	Link
Turbidity sensor		Ebay	14		367.50	5145.00	1	367.50	https://www.ebay.com/p/DFRobot-Gravity-Analog-Turbidity-Sensor-for-Arduino/1774788334?iid=12347861
Light sensor		Ebay	5	Packs of 2	20.47	102.35	1	10.24	https://www.ebay.com/itm/2pcs-BH1750-BH1750FVI-Digital-Light-intensity-Sensor-Module-For-Arduino-/19
Temperature sensor		Ebay	2	Packs of 10	111.08	222.16	1	11.11	https://www.ebay.com/itm/1-2-5-10-PCS-1M-Length-Waterproof-Digital-Thermal-Probe-or-Sensor-DS18B20
pH sensor		Ebay	6		127.20	763.20	1	127.20	https://www.ebay.com/itm/Aquarium-Hydroponic-PH-Electrode-Probe-Liquid-PH-Value-Detection-Sensor/3
Rpi 3 B	Including 32Gb Sd card, casing and PS	Komplett.no	1		728.00	728.00	1	728.00	https://www.komplett.no/product/877502#
Rekkeklemmer	3pin	Amazon	3	Packs of 30	93.31	279.93	6	18.66	https://www.amazon.com/Atoplee-30pcs-2-54mm-Terminal-Connector/dp/B00QWTHAEM?crid=3TQ50Z8E
Jumper Wires	Dupont 20cm 2.54MM	Amazon	4	Packs of 40	8.63	34.52	7	1.51	https://www.ebay.com/p/for-Arduino-Breadboard-40pcs-20cm-Dupont-Wire-Jumper-Cable-Male-to-Female
Multiplekser 8:1 gruppe 1		DigiKey	1	Packs of 10	60.72	60.72	1	6.07	https://www.digikey.no/product-detail/en/texas-instruments/SN74LV4051AN/296-3828-5-ND/374747
ESP8266 nodemcu 1		Ebay	10		27.30	273.00	1	27.30	https://www.ebay.com/itm/NodeMCU-ESP8266-ESP-12E-V1-0-Wifi-CP2102-IoT-Lua-267-NEW/23245193450
Perfboard		Kjell & Company	1	Pack of 10	99.90	99.90	1	9.99	https://www.kjell.com/no/produkter/elektro-og-verktoy/elektronikk/kretskort/testkort/luxorparts-eksperim
O ring cord		RS	1		141.56	141.56	0	0.00	https://no.rs-online.com/web/p/o-ring-cords/1591484/
USB	micro A male to micro B female	Elfa Distralec	4		62.70	250.80	1	62.70	https://www.elfadistrelec.no/no/usb-otg-kabel-usb-micro-plugg-usb-kontakt-200mm-svart-nedis-ccgp60515
Silikon		Clas Ohlson	1		59.90	59.90	10%	5.99	https://www.clasohlson.com/no/Pattex%20silikon%20til%20kj%C3%B8kken%20og%20bad,%20hvit/Pr31916
Epoxy Potting Compound	Didn't use	RS	1		126.36	126.36	0	0.00	https://no.rs-online.com/web/p/potting-compounds/0851044/
Potting Compound	Didn’t arrive	Ebay	1		95.34	95.34	0	0.00	https://www.ebay.com/itm/Potting-Compound-Twin-Pack-Robnor-PX804C-250g-Encapsulation-Compound/
Beaker		FisherSci	1	Pack of 3	294.43	294.43	1	98.14	https://www.fishersci.no/shop/products/nalgene-pmp-griffin-low-form-beakers/p-4521491
Filament	For 3D-printer	RS	1		386.19	386.19	75%	289.64	https://no.rs-online.com/web/p/products/832-0220
Resistance 4.7k ohm	Already on Lab	Elfa Distralec	0		0.31	0.00	1	0.31	https://www.elfadistrelec.no/no/motstand-kohm-25-rnd-components-rnd-155mor0w4j0472a50/p/3008854
Resistance 1k ohm	Already on Lab	Elfa Distralec	0		0.57	0.00	2	1.14	https://www.elfadistrelec.no/no/motstand-kohm-rnd-components-rnd-155mor0w2j0102a10/p/30088512?p
Resistance 510 ohm	Already on Lab	Elfa Distralec	0		1.08	0	1	1.08	https://www.elfadistrelec.no/no/motstand-510-ohm-ohm-rm0207sfcn5100t52/p/16071948?pos=4&origPos
Resistance 330 ohm	Already on Lab	Elfa Distralec	0		0.79	0	1	0.79	https://www.elfadistrelec.no/no/motstand-330-ohm-vishay-mbb02070c3300fct00/p/16059202?pos=8&orig
Total						9063.36		1765.51	

Appendiks G Vedlagte filer

Mappe	Fil/prosjektmappe	Beskrivelse
Prosjektdokumentering	Logg	Logg av hele prosjektperioden
	Timeliste	Timeliste over hele prosjektperioden
	BO19E-5_GANT	Fremdriftplan
	BillOfMaterial	Material liste
Programkode	Algae_System.zip	Åpne algae_monitoring.sln som ligg i mappen Algae_monitoring Dette er program med dynamisk graf.
	Algae_System_Without_Dynamic_Graph.zip	Åpne Algae_System_Without_Dynamic_Graph_Ver3 som ligg I mappen Algae_System_Without_Dynamic_Graph Dette er program uten dynamisk graf
	Algae_System_Arduino_Code	Arduino kode som blir brukt på ESP8266
	Arduino kode sensorer	Inni i denne mappen ligg arduino kode for hver sensor
Tegninger	Circuit.dwg	Kretstegning. Autocadfil.
	Circuit.JPG	Kretstegning. Bilde
	Circuit.pdf	Kretstegning. Pdf
	Routing_Soldering.dwg	Ruting for lodding av krets. Autocadfil.
	Routing_Soldering.JPG	Ruting for lodding av krets. Bildefil, svart bakgrunn
	Routing_Soldering.pdf	Ruting for lodding av krets. Pdf, hvit bakgrunn

3D-printing	SystemDesign.scad	Hovedfil for OpenScad kode. Kan kompilere alle deler herfra.
	SystemDesign-lib.scad	Fil med alle variabler.
	turbidityCap.scad	Kode for turbiditeslokket.
	Main.WTK	Ytterskallet på lokket. Fil til printing på sars
	BottomLayer.WTK	Nederste nivå. Fil til printing på sars
	TopLayer.WTK	Øverste nivå. Fil til printing på sars
	Lid.WTK	Lokk til hovedskallet. Fil til printing på sars
	TurbSupport.WTK	Festet til turbsensor. Fil til printing på sars
	TurbCap.WTK	Lokk til turbsensor. Fil til printing på sars
	Main.gcode	Ytterskallet på lokket. Generell printingkode
	BottomLayer.gcode	Nederste nivå. Generell printingkode
	TopLayer.gcode	Øverste nivå. Generell printingkode
	Lid.gcode	Lokk til hovedskallet. Generell printingkode
	TurbSupport.gcode	Festet til turbsensor. Generell printingkode
	TurbCap.gcode	Lokk til turbsensor. Generell printingkode
	Main.stl	Ytterskallet på lokket. Modell av del.
	BottomLayer.stl	Nederste nivå. Modell av del.
	TopLayer.stl	Øverste nivå. Modell av del.
	Lid.stl	Lokk til hovedskallet. Modell av del.
	TurbSupport.stl	Festet til turbsensor. Modell av del.
	TurbCap.stl	Lokk til turbsensor. Modell av del.