Høgskulen på Vestlandet

BACHELOROPPGAVE

Automatisering for lukket akvakulturanlegg

Automation for closed aquaculture system

Jan Einar Ådland Gulbrandsen
Ole Drange

Automatiseringsteknikk
Høgskulen på Vestlandet
Svein Haustveit
31.05.2019

Jeg bekrefter at arbeidet er selvstendig utarbeidet, og at referanser/kildehenvisninger til alle kilder som er brukt i arbeidet er oppgitt, *jf. Forskrift om studium og eksamen ved Høgskulen på Vestlandet, § 10.*

1 Dokumentkontroll

Rapportens tittel: BO19E-06 Automation for closed aquaculture system	*Dato/Versjon* 31. mai 2019
	Rapportnummer: B019E-06
Forfatter(e): Jan Einar Ådland Gulbrandsen Ole Drange	*Studieretning:* 16HEAU
	Antall sider m/vedlegg 4242
Høgskolens veileder: Svein Haustveit	*Gradering:* Åpen
Eventuelle Merknader: Vi tillater at oppgaven kan publiseres.	

Oppdragsgiver: Sars International Centre for Marine Molecular Biology	*Oppdragsgivers referanse:*
Oppdragsgivers kontaktperson(er) (inklusiv kontaktinformasjon): Jørgen Høyer, Telefon: 41636030, Email-adresse: jorgen.hoyer@uib.no	

Revisjon	Dato	Status	Utført av
0.5	31. Jan 2019	Laget forarbeidet	Jan og Ole
0.6	25. April 2019	La inn overskrifter	Jan og Ole
0.8	20. Mai 2019	Fylte inn under overskrifter	Jan og Ole
0.9	30. Mai 2019	Fylte ut alt som manglet	Jan og Ole
1.0	31. Mai 2019	Fikset utseende og referanser	Jan og Ole

2 Forord

Bacheloroppgaven har gitt oss muligheten til å prøve oss i nærmest alt vi har lært gjennom utdannelsen og i mange nye utfordringer. Vi setter stor pris på den tilliten Sarssenteret og høyskolen har gitt oss med denne store og utfordrende oppgaven.

En stor takk skal gå til Sarssenteret og Marios Chatzigeorgiou som har stilt sterkt med ressurser: økonomisk, intellektuelt og sosialt.
En spesiell takk til oppdragsgiver Jørgen Høyer ved Sarssenteret som har fulgt oss opp grundig gjennom hele oppgaven og alltid sørget for at alt skal være lagt til rette for at oppgaven skal kunne gjennomføres på en god måte. Han har bestilt det vi måtte ha og delt det han har av kunnskaper innenfor det vi måtte trenge hjelp til.
En takk til Høyskolen På Vestlandet og internveileder Svein Haustveit som har godtatt denne flotte oppgaven fra Sarssenteret.

Vi er også takknemlig for venner, bekjente, familie, Github og Stackoverflow for god støtte og motivasjon.

3 Sammendrag

Arbeidet vårt har gått ut på å planlegge, lage og teste et overvåkningssystem for et lukket vannsystem hvor Sarssenteret skal oppfostre dyret Ciona Intestinalis gjennom generasjoner. Dette er et dyr som ligner på en gjennomsiktig sjøpølse og brukes til evolusjonsforskning på gener og DNA. Vi har laget et system som måler vannkvalitet, lager sirkulasjon i vannet og mater dyrene. Vannkvalitet består av måling av temperatur, PH, turbiditet, vannstrøm og saltinnhold. Alle verdiene fra vannkvalitet pluss mating og sirkulasjon blir lagret i database slik at de kan optimalisere tankene for dyrene og bruke dem til forskning.

Gjennom oppgaven har vi lært mye om trådløs dataoverføring, programmering av mikrokontrollere, koding i terminal på Raspberry Pi, bruk av Node Red som brukergrensesnitt og lært litt om biologi, DNA og evolusjonslære.

Vi fikk ikke tid til å lage systemet ferdig, men det virker for en tank og man trenger bare å kopiere kretskortet som vi har laget for å fullføre systemet for alle 16 tankene.

4 Innholdsfortegnelse

5 Innledning

5.1 Oppdragsgiver

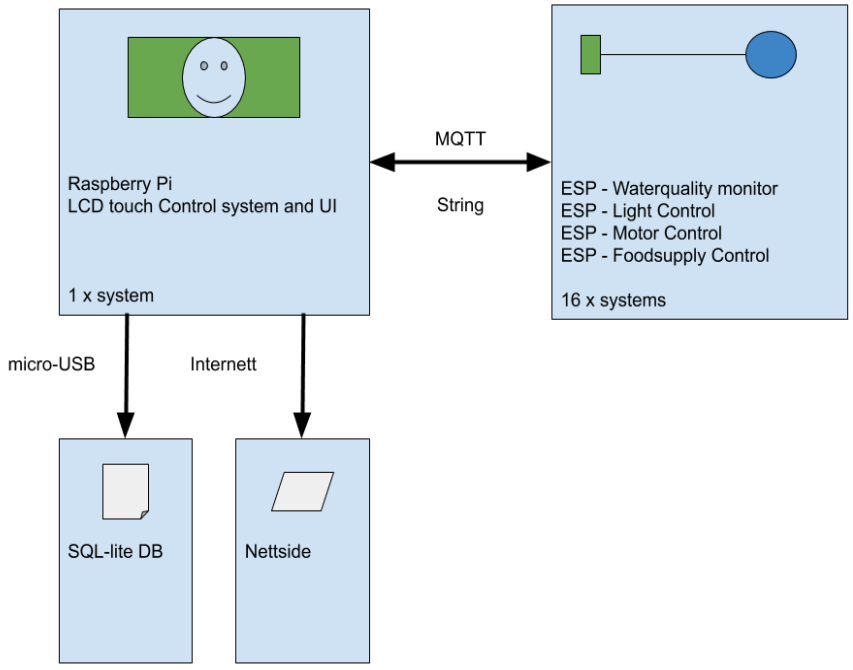
Sars internasjonale senter for marin molekylær biologi studerer biologiske prosesser i marine organismer med fokus på evolusjon. Senteret er for øyeblikket organisert i åtte forskningsgrupper og er lokalisert i moderne fasiliteter på Høgteknologi Senteret i Bergen. Forskningsgruppe S13 ved Sarssenteret ønsker å utvikle automasjon og monitorering for et lukket akvakultursystem der forskningsdyrene Ciona Intestinalis skal introduseres med mål om å fremdyrke nye transgenetiske linjer.

5.2 Problemstilling

Sars ønsker et system som gjør det mulig og lett å avle fram forskningsdyrene Ciona Intestinalis. For å vite hvilke klima disse dyrene avles best i trenger de et overvåkningssystem som måler og logger klimaet de blir avlet i. De vil også kunne justere natt og dag-modus, justere hvor mye bevegelse det er i vannet og justere mengden mat som blir gitt per dag.

5.3 Hovedidé for løsningsforslag

Oppdragsgiver ser for seg en løsning hvor vi bruker mikrokontrollere med innebygget Wi-Fi til å monitorere vannkvaliteten og styre lys, blandingsmotor og mat-tilførsel. Mikrokontrollerene sender informasjonen til en Raspberry Pi som skal vise verdiene på en touch-skjerm. Verdiene skal også lagres og skal kunne ses eksternt. Motorene, lysene og mengde mat per dag skal kunne justeres ved hjelp av touch-skjermen.



Figur 1: Systemoversiktsfigur

6 Kravspesifikasjon

Kravene er drøftet med oppdragsgiver og godkjent.

Krav:

1. Systemet består av 16 tanker.
2. Systemet skal måle temperatur, PH-verdi, vannflyt inn til tank, turbiditet og saltnivå i hver tank.
3. Systemet skal styre en blandingsmotor for hver tank.
4. Systemet skal styre tilførselen av mat til hver tank.
5. Systemet skal styre lys i natt og dag modus.
6. Systemet skal presentere de målte verdiene minst 6 ganger i minuttet på skjerm på anlegget, samt på ekstern pc.
7. Blandingsmotorene skal kunne slås av og på individuelt on-site.
8. Systemet skal ha dokumentasjon for HW og SW, samt brukermanual.
9. Systemet skal gi lydsignal ved stans i vannflyt.

7 Analyse av problemet

Krav 2: Systemet skal måle temperatur, PH-verdi, vannflyt inn til tank, turbiditet og saltnivå i hver tank.
Vi bruker mikrokontroller til å overvåke sensorene for hver verdi som står i kravet. Oppdragsgiver har valgt ut sensorene.

Krav 3: Systemet skal styre en blandingsmotor for hver tank.
Blandingsmotorene er allerede montert for hver tank. Vi vil styre de med mikrokontrollerne gjennom solidstate releer. De suppleres med 24VAC gjennom en 230VAC-24VAC transformator.

Krav 4: Systemet skal styre tilførselen av mat til hver tank.
Matventilene er allerede montert for hver tank. Vi vil styre de med mikrokontrollere gjennom solidstate releer. De suppleres med 24VDC fra en 24VDC strømforsyning.

Krav 5: Systemet skal styre lys i natt og dag modus.
Lysene er ikke bestilt, så vi venter på mer informasjon fra oppdragsgiver om hvilke lys han trenger. Når vi vet dette kan vi planlegge hovedstrømmen til lysene, men styrestrømmen vil uansett gå via mikrokontroller.

Krav 6: Systemet skal presentere de målte verdiene minst 6 ganger i minuttet på skjerm på anlegget, samt på ekstern pc.
Vi skal ved hjelp av kode i mikrokontrollere, samt Wi-Fi protokoll, sørge for at dette kravet blir oppfylt.

Krav 7: Blandingsmotorene skal kunne slås av og på individuelt on-site.
Vi skal bruke en touchskjerm som hører til Raspberry Pi for å kunne slå på av og på blandingsmotorene. Touchskjermen vil inneholde forskjellige faner hvor hver av tankene hvor det vil være mulig å slå av og på blandingsmotorene.

Krav 8: Systemet skal ha dokumentasjon for HW og SW, samt brukermanual.
Vi skal i samarbeid med oppdragsgiver utforme brukermanual på Engelsk. Systemet dokumenteres i bacheloroppgave rapporten.

Krav 9: Systemet skal gi lydsignal ved stans i vannflyt.
Raspberry Pien skal være koblet til en buzzer som gir lydsignal ved stans i vannflyt på en eller flere av tankene. Hvilken tank det gjelder vil vises på touchskjermen.

7.1 Utforming av mulige løsninger

Løsningsalternativene går ut på oppsettet av mikrokontrollere.
Hovedløsningen er uansett på linje med det oppdragsgiver har satt som løsning.

7.1.1 Løsningsalternativ 1

Figur 2: NodeMCU

Oppgaven kan løses med å se på hver tank som et eget system. Der vi får alle sensor verdiene og styringen til en tank inn på en mikrokontroller. Da vil alle tankene trenge sin egen mikrokontroller men systemet vil være lett å gjøre endringer på. Da vil det også være lett å gjenbruke både kode og koblingsskjema for alle tankene. For å gjøre systemet billig bruker vi mikrokontrolleren ESP8266. Denne Kontrolleren har 8 digitale IO pinns og en analog IO pin. Denne skal da styre fire sensorer, en motor og en ventil. To av sensorene er analoge og da trenger vi en to MUX for å få to analoge verdier inn på en inngang. To MUX bruker også en digital IO. To digitale IO går til de siste sensorene og to digitale IO går til motor og ventil. Dette gjør at en ESP8266 kan styre alt en tank trenger, men ikke nokk til to tanker.

7.1.2 Løsningsalternativ 2

En løsning er å dele opp i overordnede systemer slik at hver mikrokontroller fokuserer på en type måling eller en type styring. Hver kontroller har en analog inngang og 8 digitale inn/utganger. Samtidig vil noen av de digitale utgangene gi høyt signal ved restart og booting som gjør at vi ikke kan bruke disse til styring.

Vi vil da ha en kontroller for temperaturbuss og saltmåling og lysstyring. Som da bruker en analog inngang, en digital inngang og en digital utgang.
En kontroller med multiplex for PH verdi. PH sensoren gir et analogt signal, som vi multiplexer gjennom en 16 kanals MUX til den analoge inngangen. Det brukes da 4 digitale utganger for å velge kanal på MUXen.
En kontroller med multiplex for turbiditet. Turbiditetssensoren gir også et analogt signal, som vi multiplexer gjennom en 16 kanals MUX til den analoge inngangen. Det brukes da 4 digitale utganger for å velge kanal på MUXen.
To kontrollere for vannflyt måling. Vannflyten måles ved antall pulser per sekund og bruker en digital inngang. Vi trenger da to kontrollere for å overvåke 16 vannflyt målere.
Tre kontrollere for motorstyring.
Tre kontrollere for mat tilførsel.

7.1.3 Vurderinger i forhold til verktøy og HW/SW komponenter

Hardware som blir brukt i oppgaveløsningen kommer an på hvilken løsning vi går for. Men vi må bruke en form for mikrokontroller, en data/Raspberry Pi, en skjerm å vise data på og noe for å lagre dataen vi måler. Det vi har fått anbefalt av oppdragsgiver er en ESP8266 som nevnt i løsningsalternativene og en Raspberry Pi 3b+. Dette har vi valgt å bruke i begge løsningsalternativene, fordi vi er kjent med programvaren for ESP8266 og det er den billigste type mikrokontroller som vil løse oppgaven. Raspberry Pi er vi ikke så kjent med, så da har vi tenkt til å legge av en del tid til å lære å bruke den. Oppdragsgiver har stått for innkjøp av sensorer, og disse er vi ikke kjent med fra før av, så disse må vi lære å bruke. Skjerm og lagring har vi vært litt borti fra før, men ikke nok til at vi kan det godt nok til å fikse det uten mer opplæring. På software fronten er mesteparten av arbeidet og finne og sette sammen tidligere laget kode. Her er vi begge ganske kjent med programmene vi skal bruke så det vil ikke bli lagt spesielt mye tid i å lære oss opp i disse.

7.2 Konklusjon for valg av løsning

Vi har valgt å gå for løsningsalternativ 1 fordi det er lettest å skalere og gjøre endringer på i ettertid. Når hver tank er et eget system så vil det bli problemfritt å legge til eller trekke fra tanker i ettertid fordi tankene er sitt eget delsystem. Hver tank har bare forbindelse med Raspberry Pi og den vil da kunne styre og vise alt fra en tank ved å “snakke” med en mikrokontroller. Dette er ikke tilfellet i løsningsalternativ 2. Det vil også være lettere å endre antall tanker sett fra software siden. Her vil alle mikrokontrollerne ha lik kode, og det vil bare være å gjenbruke den om det skal legges til flere tanker i ettertid. Total systemet vil også bli mer oversiktlig fordi du lettere kan dele det opp i subsystemer.

8 Realisering av valgt løsning

8.1 Hardware

8.1.1 Sensorer

8.1.1.1 Vannflyt

YF-S201 er en digital hall-effekt vannflyt sensor, som sender en puls per omdreining på skovlhjulet.
Figur 3: Vannflytmåler, YF-S201

Den har et måleområde på 1 til 30 L/min, temperaturområde på -25°C til 80°C og nøyaktighet på ±10 %. Denne sensoren var kjøpt inn av ekstern-veileder og dekker behovene for å oppfylle kravene til prosjektet. Den lave nøyaktigheten spiller liten rolle da det viktigste med vannflyt-målingen er å se om det er vannflyt eller ikke.

8.1.1.2 Temperatur

Figur 4: Temperaturmåler, DSB18B20

DSB18B20 er en digital temperatursensor som sender en 9 - 12 bit Celsius temperaturmåling. Den har et måleområde fra -55°C til 125°C hvor vi kommer til å måle fra ca. 5°C til 20°C. Den har et innebygget serienummer som kan brukes for å sette opp “onewire” bus system fra sensor til sensor. Vi har valgt å ikke bruke denne funksjonen da vi uansett må ha en mikrokontroller per tank. Vi bruker et ferdiglaget bibliotek fra Dallas Sensor Technologies for å hente ut temperaturverdien. Grunnen til at vi bruker denne er at ekstern-veileder allerede hadde bestilt denne før vi begynte.

8.1.1.3 PH

Figur 5: PH probe, E-201 BNC

PH proben har en BNC-kontakt som kobles til et målebrett. Proben består av en sølv/sølvklorid elektrode og en glasselektrode som er sensitiv for hydrogen ioner. Sensorkortet har kobling for spenningstilførsel, analog utgang og temperaturkorreksjon.
Den første proben vi fikk ville ikke måle nøyaktig. Vi fikk en ny probe som het EC-201 som vi klarte å kalibrere innenfor kravet på 7-10 PH verdi.

8.1.1.4 Turbiditet

Figur 6: Turbiditetssensor, RB-Dfr-646

RB-Dfr-646 fra Robotshop er en analog turbiditetssensor. Den måler hvor lett lys går gjennom vannet, samt partikler i vannet. Toppen av sensoren er ikke vanntett, og må tettes med silikon.

8.1.1.5 Salt/Konduktivitet

Figur 7: Saltprobe, ASC probe k1.0

Atlas Scientific Conductivity Probe K 1.0 med **EZO Conductivity Circuit**.
Proben måler konduktivitet i vannet med et måleområde på 5 - 200 000 µS/cm og en nøyaktighet på ±2 %. Den tåler temperaturer fra 1-110 °C og en levetid på ca 10 år. Måleprinsippet går ut på å sette en AC-spenning på to elektroder som da vil skape en elektronflyt mellom probene av de frie elektronene i vannet.

Figur 8: Saltkort, EZO Conductivity circuit

Kretsen har kontakter for spenning (3.3 - 5V), BNC-kontakt og seriell (RX, TX). Den støtter UART og I2C data protokoller, kan lese av konduktivitet, TDS, saltholdighet, og spesifikk tyngdekraft. Den gjør en lesing per sekund og har et og to punkts kalibrering.

8.1.2 Pådragsorganer

8.1.2.1 Motor

Figur 9: Blandingsmotor

Systemet som var laget før vi fikk oppgaven hadde allerede en motor per tank. Motorene var koblet opp mot en PLS og er 24VAC motorer. Dette er litt upraktisk da ingen andre komponenter vi har eller planlagte å skaffe trenger 24VAC. Vi vurderte om vi skulle bytte motorer eller prøve å styre dem fra mikrokontrolleren gjennom PLS-en som kan gi 24VAC, men kom fram til at det var minst tidkrevende og lettest for oss å beholde motorene og heller styre de med releer og 24v transformator.

8.1.2.2 Ventil

Figur 10: Matingsventil

Ventiler var også ferdig oppkoblet og klar til bruk i systemet vi startet med. Men de var 24VDC styrt. De kan åpnes manuelt og ved strøm, noe som er veldig praktisk for oppdragsgiver. Dermed gikk vi for å beholde ventilene og kjøpe inn en strømforsyning og styre ventilene gjennom releer.

8.1.2.3 Lys

Figur 11: LED strip 2700K

Når vi fikk oppgaven var planen at oppdragsgiver ville finne lys til systemet fordi dyrene som skal overvåkes trengte lys med spesifikt fargetemperatur. Hvilken fargetemperatur oppdragsgiver ønsket var uvisst på denne tiden og derfor ønsket de å finne lys selv. Når systemet hadde en prototype og det meste var testet ønsket oppdragsgiver at vi fant lys med fargetemperatur 2700k. Dette var ganske vanskelig da de fleste lys vi fant var laget for 12VDC som vi ikke har på systemet. Vi ønsket å bruke led-stripe fordi det er lettest å montere. Vi bestilte 5M LED Strip 2700K fra ledonline.no fordi den hadde ønsket fargetemperatur og lav lumen.

8.1.2.4 LCD-skjerm

Figur 12: RPI LCD Touchscreen

For å vise sensor verdi “on-site” har vi flere muligheter. Vi kunne brukt lys for å indikere ok eller kritisk tilstand. Vi kunne hatt skjermer til hver tank, men det beste alternativet synes vi var å ha en skjerm som har all informasjon og mulighet for å styre pådragene. For å få dette til måtte vi lage et oversiktlig brukergrensesnitt. Skjermen vi bruker er en 3.5 tommer touchskjerm som kobles ved å sette den direkte oppå RPI. På skjermen er det en fane for hver tank hvor du kan se sensor verdi samt styre mating og sirkulasjon for den spesifikke tanken. Skjermen er veldig liten for all informasjon som skal vises og styres, og vi må derfor lage et kompakt og oversiktlig brukergrensesnitt for at det skal kunne brukes.

8.1.3 3 Kontrollere

8.1.3.1 ESP8266

Figur 13: NodeMCU

ESP8266 er en mikrokontroller med innebygget Wi-Fi modul. Den har en analog inngang og 8 digitale inn og utganger. Den suppleres med 5V enten med micro-USB eller til to av pinnene. Den har en innebygget spenningsdeler slik at du kan ta ut 3.3V på to av pinnene. Den har også en rx, tx seriell tilkobling. NodeMCU programmeres med Arduino IDE i en form for C+ språk.

8.1.3.2 Arduino Uno

Figur 14: Arduiono Uno

Arduino Uno er en mikrokontroller fra Arduino med Wi-Fi modul. Fordelen med Arduino sine kort er at de har god programvarestøtte og er godt dokumenterte. Bakdelen er at de er ganske mye dyrere.

8.1.3.3 Raspberry Pi 3b+

Figur 15: Raspberry Pi 3b+

Raspberry Pi er en liten datamaskin som kan gjøre stort sett alt som en vanlig datamaskin kan bare mye tregere. Vi bruker den som en hovedhjerne i systemet vårt ved at all data går igjennom den før det går videre. Som for eksempel sensordata går først til Raspberry Pi for å så vises på skjerm og lagres på database. Vi trenger den for å gjøre systemet mye lettere å lage og mye mer oversiktlig. Den er også MQTT broker-en våres som gjør det lett å sende informasjon over Wi-Fi. Skjermen kobles også direkte på Raspberry Pi-en noe som sparer oss for kabler og montering. Den bruker https://bit.ly/2lyGi0Xoperativsystemet som heter Raspbian og kjører Node-Red noe vi kommer mer tilbake på lengre nede i dokumentet

8.1.4 Strømforsyning

Fordi det var et påbegynt prosjekt når vi fikk oppgaven bruker vi tre forskjellige Spenningskilder.

8.1.4.1 5VDC

Figur 16: Strømforsyning 5VDC

5VDC trenger vi til å forsyne mikrokontrollerne, sensorene og IC-en. Her bestilte vi en 300w fordi det er nokk til å forsyne 16 mikrokontrollere og alle sensorene. Den er også valgt på grunn av pris i forhold til watt.

8.1.4.2 24VDC

Figur 17: Strømforsyning 24VDC

Alle ventilene trenger 24VDC for å styres. Ventilene er på 8w hver må kunne brukes samtidig. Dermed trenger vi 9 Watt *16 stykk = 144 Watt. Den vi valgte er på 360 Watt fordi det var vanskelig å finne noe billigere enn den. Vi tenkte også at lys til kretsen kan gå på 24VDC og dermed kan det være greit å ha litt mer enn akkurat nok til ventilene.

8.1.4.3 24VAC

Figur 18: Transformator 230VAC til 24VAC

Motorene trenger 24VAC tilførsel noe som var overraskende vanskelig å finne. Transformatoren vi valgte var den billigste vi fant som kunne kjøre alle 16 motorene samtidig. 8 Watt * 16 stykk = 128 Watt. Transformatoren har to utgangsspoler som hver tåler 150 Watt.

8.1.5 Innkapsling

Figur 19: Innkapslingsboks Lemotech

«Lemotech ABS Plastic Junction box» er vann og støvsikker med en IP-grad på 65. De hadde bokser i riktig størrelse for mikrokontroller kortet, solidstate releene og strømforsyningene. Da får vi samme innkapsling til alt utstyret.

8.2 Software

8.2.1 Arduino IDE

Arduino IDE var det første programmet vi jobbet med på starten av bacheloroppgaven. Vi ble kjent med Arduino IDE fra skolen og brukte dette til å teste mikrokontrolleren og teste sensorene etterhvert som vi fikk dem. Programmet er enkelt å sette opp til forskjellige mikrokontrollere og har mange "open-source" biblioteker. Det er hverken «autofullfør» eller korrekturleser i Arduino IDE som gjør at det går en del ekstra tid på å prøve å kompilere programmer som har små enkle skrive feil eller mangler semikolon. Mot slutten av oppgaven gikk vi over til Visual Studio Code for å enklere kunne redigere koden.

8.2.1.1 Biblioteker

DallasTemperature brukes for å lese av og omgjøre bit-verdien til tempereratursensoren om til desimal.

OneWire brukes for å definere temperatursensorer og bruke deres unike ID for å kunne utnytte databussystemet som er innebygget i disse temperatursensorer. Vi har ikke tatt i bruk dette, men trenger biblioteket for å definere at vi har en sensor av typen OneWire.

ESP8266WI-FI brukes for å sette opp mikrokontrolleren til å koble seg opp mot Wi-Fi nettverk.

PubSubClient brukes for å publisere og abonnere på emner i MQTT nettverket gjennom Mosquitto serveren.

SoftwareSerial brukes for å sette opp seriell kommunikasjon på RX, TX portene. Uten dette biblioteket vil vi ikke kunne definere RX, TX som seriell port og all seriell kommunikasjon ville gått på mikro-USB.

8.2.2 Visual Studio

Når vi fikk oppgaven hadde vi bare forkunnskaper innenfor brukergrensesnitt i C#. Dermed ble dette et naturlig programmeringsspråk å velge. Når vi programmerte på skolen har vi brukt Visual studio for å programmere C# og tenkte at dette kom til å gå fint. Men når vi begynte å overføre programmene til RPIen fungerte noe mens annet ville ikke virke i det hele tatt. Vi fant ut at Raspberry Pi med Raspbian operativsystem støtter bare noen av funksjonene vi ønsket å bruke. Dermed vurderte vi å bytte til en annen metode/ program for å lage brukergrensesnittet. Vi vurderte også om vi skulle kjøre Windows operativsystem på RPIen, men fordi vi hadde funnet ut mye som var bra med Raspbian operativsystemet valgte vi å ikke bytte det. Vi fant Node-Red, noe som ingen av oss hadde noen forkunnskaper i, men det så ut som det var lett å lære. Etter litt prøving med Node-Red så vi at det er mye bedre program for oppgaven og det er forhåndsinstallert og laget for RPIen. Vi valgte da å droppe C# og Visual studio helt og fokusere på å lære oss og lage så bra program som mulig med Node-Red.

8.2.3 Visual Studio Code

For å programmere mikrokontrollerne brukte vi Arduino IDE i starten fordi det var programmet vi kjente til og viste hvordan virket. Det er også programmet som er laget for å programmere mikrokontrollere noe som gjør at det er veldig lett å bruke og legge til biblioteker i. Problemet med Arduino IDE er at det ikke har "autofullfør" eller noe god fargekoding. Disse fordelene er vi vant til fra tidligere programmerings program, og det gjør det mye lettere å skrive rask og oversiktlig kode. Vi fant ut at vi kunne bruke Visual Studio Code (som er et annet program enn Visual studio) som har "autofullfør" og god fargekoding til å skrive "Arduino" kode. Vi byttet over til dette men for å kalibrere salt sensoren måtte vi bruke Arduino IDE. Dette gjør at vi har brukt begge programmene til å skrive og endre på den samme koden, noe som er ganske upraktisk. Stort sett har vi kun brukt Visual Studio Code når vi feilsøkt og for å ordne opp i utseende av koden.

8.2.4 Raspbian

Vi valgte å holde oss til Raspbian operativsystemet da dette dekker systemets behov og vi fant god software som Node Red til UI og Mosquitto til trådløs dataoverføring.

8.2.4.1 Node Red

Node-Red fant vi etter vi fikk problemer med C#. Det er et blokkbasert programmerings program som bruker Javascript. Ingen av oss har programmert i Javascript før, men siden Node-Red er brukt så mye til oppgaver innenfor IOT var det masse gode guider til den type oppgaver vi hadde. Vi valgte å lage en fane for hver tank pluss en fane for hovedskjermen. Fra hovedskjermen kan du se salt verdien, styre lysene. Du kan også ta på eller av alle motorene og ventilene slik at du slipper å gå innom alle fanene. Hovedskjermen viser også de største og minste sensor verdiene den siste dagen. På tank fanene kan du styre hvor lenge en matings periode er og hvor lang tid det skal gå mellom matings periodene. Du kan se sensorverdiene fra den tanken og styre motoren av og på. Alle verdier som blir sendt til RPIen går via Node-Red og seks ganger hvert minutt blir alle verdiene lagret i databasen. Siden Node-Red hadde en god "Add-On" for SQLite database valgte vi å bruke den. I databasen lagrer vi alle verdier til systemet og hver tank i hver sin tabell slik at det blir oversiktlig og lett å finne de verdiene du trenger.

8.2.4.2 Mosquitto

Mosquitto er en open-source (EPL/EDL lisensiert) server som implementerer MQTT protokollen. Det tar opp lite ressurser og passer godt for vårt prosjekt med små mikrokontroller og single-board computers som Raspberry Pi. MQTT protokollen bruker en publiser/abonner metode for å dele strenger med tekst mellom klientene i nettverket. Det opprettes emner på serveren etterhvert som det publiseres i emnene. Alle som abonnerer på et emne vil motta strengene som publiseres i emnet. Vi tenker å bruke det for vi har en haug med data som skal sendes fra mange enheter til en enhet. Raspberry Pi-en vil abonnere på et emne og alle mikrokontrollerne vil publisere til dette emnet med sin unike ID i starten av strengen. Raspberry Pien blir også publisere til et emne hvor den unike IDen i starten av strengen vil bestemme hvilke mikrokontroller som skal reagere på strengen. Serveren blir installert på Raspberry Pien og starter opp automatisk når Raspberry Pien går på. Serverens IP adresse er den samme som Raspberry Pien sin og er derfor satt statisk for å gjøre programmet til mikrokontrolleren enklere.

8.2.4.3 LCD

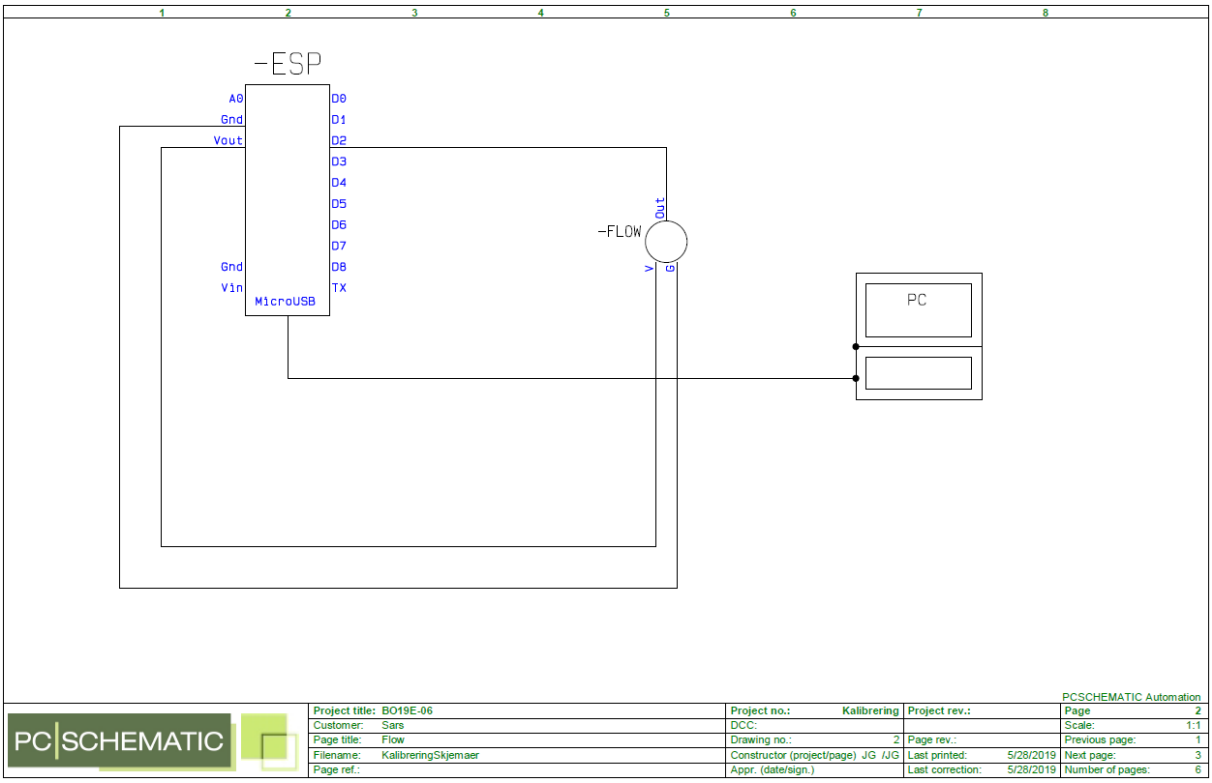
LCD displayet vi bruker var bestilt før vi fikk oppgaven, men vi hadde mulighet til å kjøpe et annet om vi ønsket det. Software delen av displayet vi bruker er veldig lett. Det er 5 kommandoer som ligger i en guide på gitthub.com som vi skrev inn i konsoll vinduet i Raspbian. Etter en restart så virker det. Eneste negative med dette var at vi ikke fikk bruke HDMI-porten på RPIen lenger. Dermed er dette det siste vi gjorde slik at vi kunne programmere Node-Red og RPIen fra en stor skjerm.

9 Testing

9.1 Kalibrering og Testing

9.1.1 Vannflyt

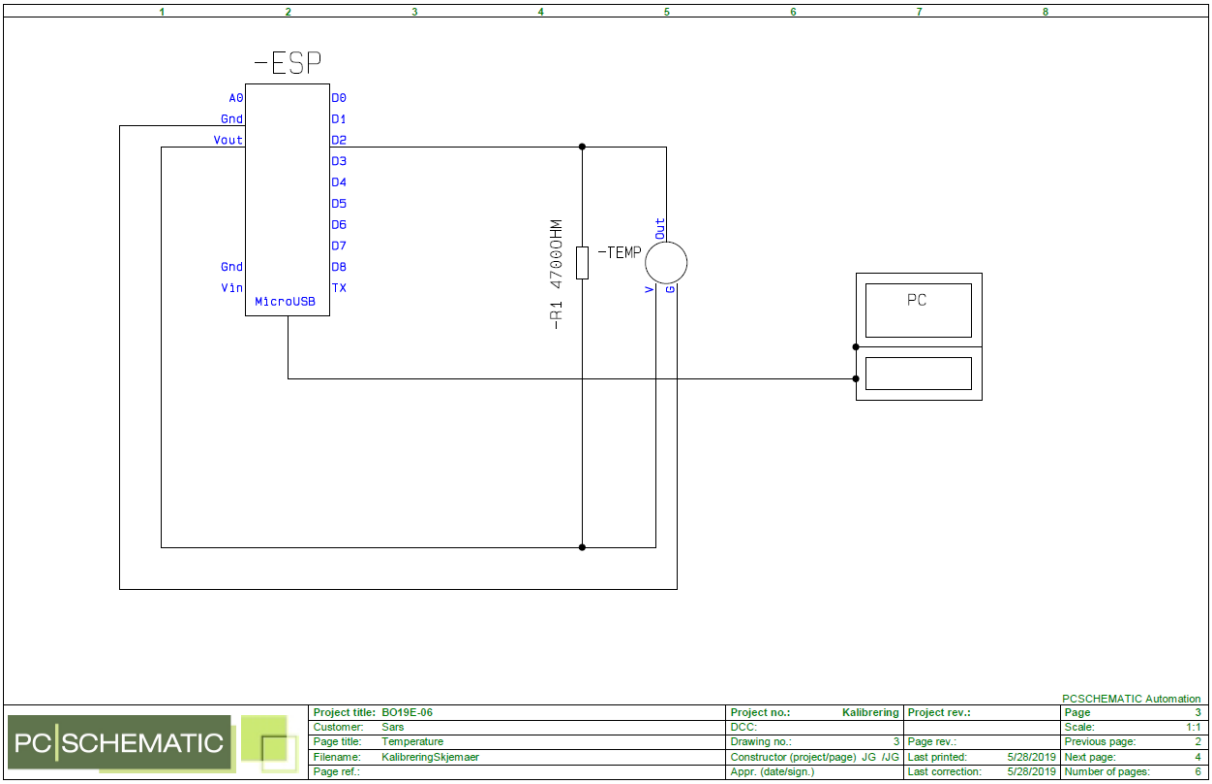
Vannflytmåleren kobles opp som anvist på *Figur 20*. Vannflytmåleren gir en puls per omdreining på skovlhjulet inni. Den kommer dokumentert med en kalibreringsfaktor på 4.5 for å omgjøre fra pulser til L/min. Hvis du vil kontrollere måleren kan du sette opp programmet for måling av vannflyt og helle en liter vann gjennom måleren mens du tar tiden og kontrollere at gjennomsnittet av målingene tilsvarer det fysiske resultatet.



Figur 20: Vannflytkalibrering koblingsskjema

9.1.2 Temperatur

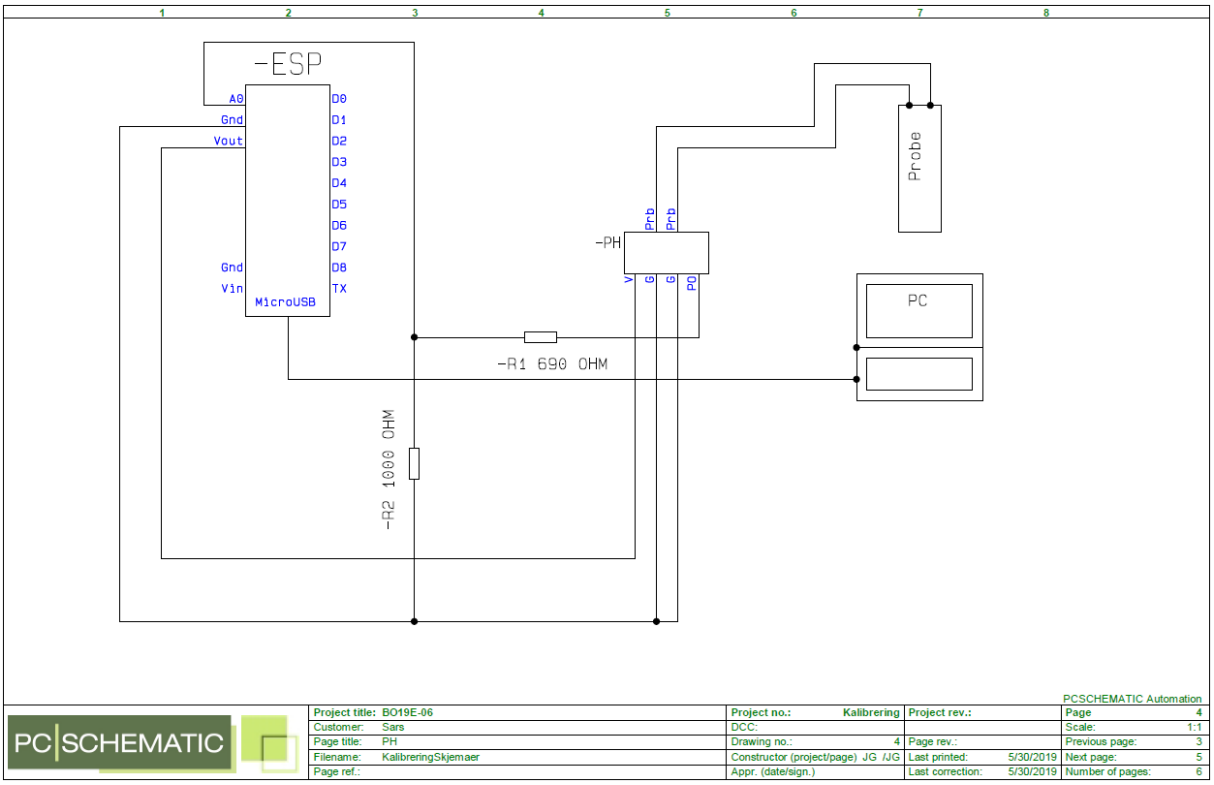
Temperatursensoren kommer ferdig kalibrert med en digital output. For å teste at sensoren gir korrekte verdier kan du måle i isvann og kokende vann for å se at du får 0 og 100 grader Celsius. Koble opp sensoren som vist på *Figur 21*. Sett opp programmet for lesing av temperatur og test. Om verdiene avviker kan du justere ved å trekke fra eller legge til avviket.



Figur 21: Temperaturkalibrering koblingsskjema

9.1.3 PH

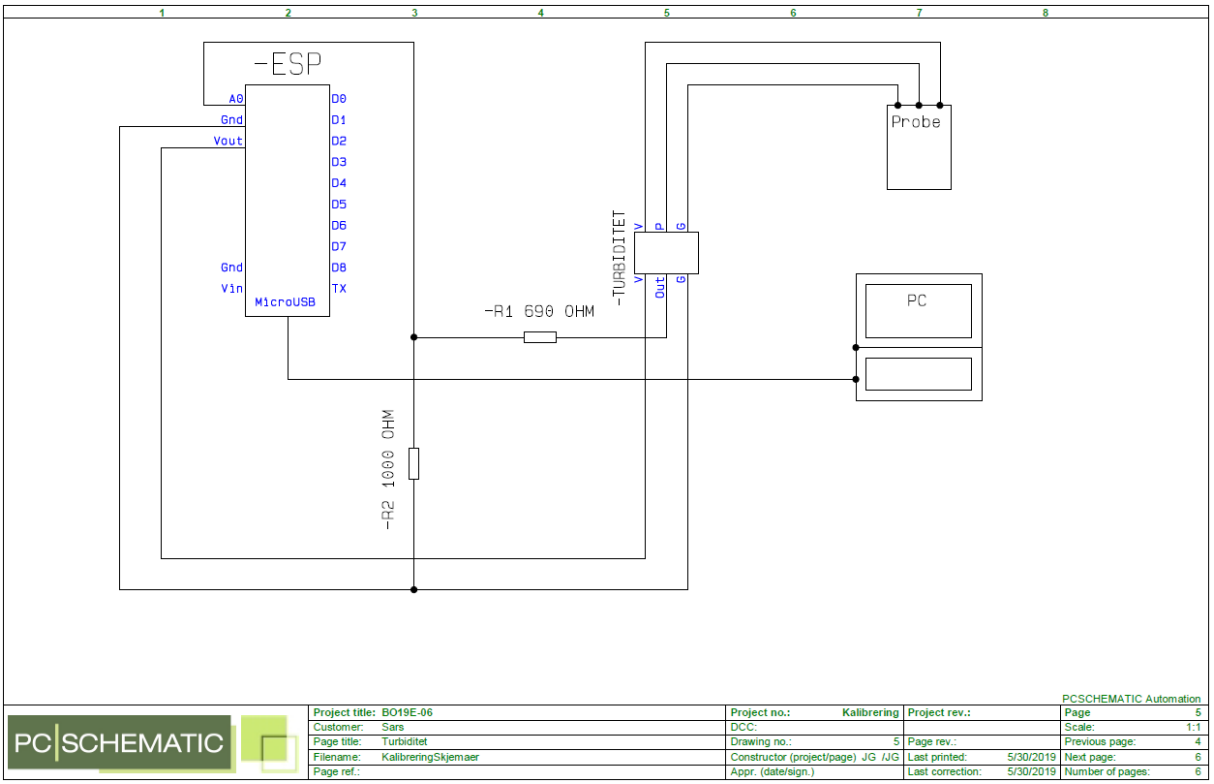
PH sensoren med kort og probe kobles opp slik som på *Figur 22*. Sett opp Seriell overvåkning på analog inngang. Sett proben i to ulike PH løsninger og noter måleverdien. PH sensoren gir lavere output ved høyere PH løsning. Formelen vi kom fram til for lineariteten til PH kortet er: (1024 - analogRead)*0.032). Deretter flytter vi nullpunktet ved å trekke fra eller legge til verdi til denne formelen for å få god linearitet innenfor vårt måleområde på 7-10 PH.



Figur 22: PHkalibrering koblingsskjema

9.1.4 Turbiditet

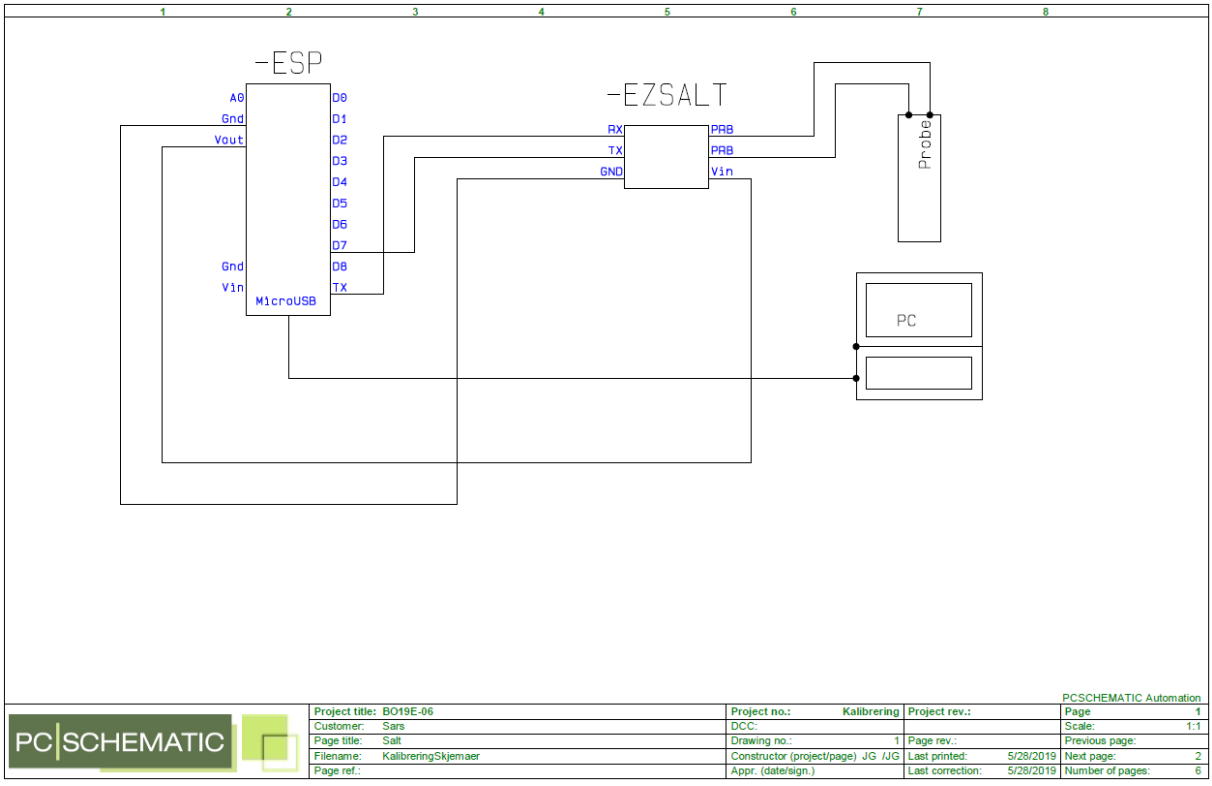
Turbiditetsensoren kobles opp som på *Figur 23* for å kalibrere. Sett opp seriellmonitor av A0 inngangen og konverter analoginputen til en spenning ved formelen: «analogRead* (3.3 / 1024)». Legg proben i kaldt springvann, pass på at toppen ikke kommer under vann da den ikke er vanntett. Les av spenningen og juster på justeringsskruen til du leser av ca. 2.8V. For å konvertere til NTU kan du bruke formelen oppgitt fra produsenten: «NTU = -1120.42 * x^2 + 5742.3 * x - 4352.9».



Figur 23: Turbiditetskalibrering koblingsskjema

9.1.5 Salt

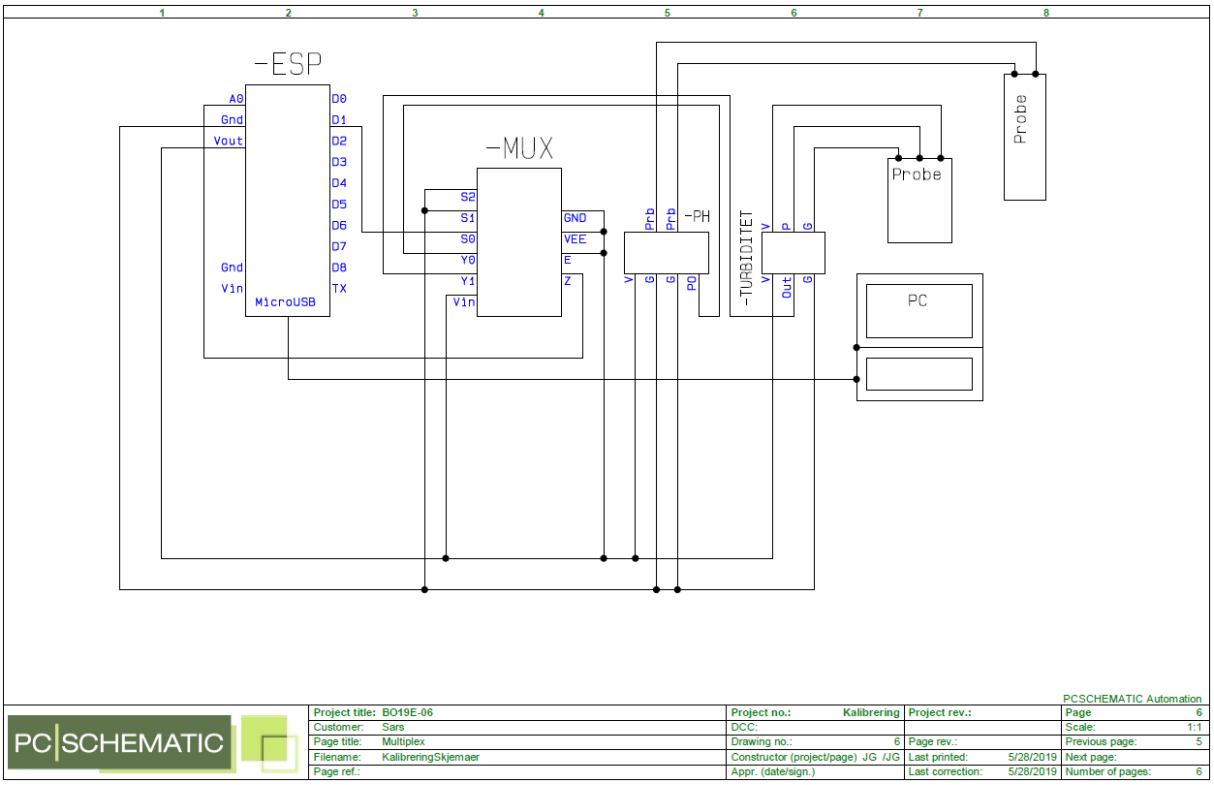
Saltsensoren kalibreres enkelt ved hjelp av UART seriell kommunikasjon. Det ligger ved et program for å kalibrere den for Arduino Uno i dokumentasjonen til EZkortet.
For å kalibrere den, kobler du den opp som på *Figur 24*, åpner programmet som følger med i Arduino IDE. Velg riktig port og overfør programmet til ESP8266. Åpne seriell monitor på samme baud-rate som i programmet og observer målingen. Når målingen er stabil med proben liggende tørt på pulten skriver du i seriell monitor tekst feltet "cal,dry". Da skal sensoren svare *OK og verdiene som leses av skal være 0.00. Deretter finner du frem to forskjellige saltløsninger. Vi hadde en på 1430 us/cm og en på 12800 us/cm. Legg proben i den laveste løsningen, pass på at all luft er ute av hullet i proben. Se at du får en stabil måleverdi og skriv i seriell monitor tekst feltet "cal,low,1430". Da skal du lese *OK, men ingen endring i verdien. Deretter skyller du saltproben og setter den i den høye saltløsningen. La verdien stabilisere seg og skriv "cal,high,12800". Da skal du les *OK og verdien skal endre seg til 12800. Skyll proben og sett den i den lave løsningen for å kontrollere at kalibreringen var god.



Figur 24: Saltkalibrering koblingsskjema

9.1.6 Multiplex

Multiplexen koblet vi opp som på Figur 25 med turbiditet og PH sensor. Laget et program som slår av og på utgang D1 og leser av A0 på Seriell monitor. Vi fikk ikke riktig verdier når vi prøvde den første gangen, og skjønte tilslutt at vi hadde bestilt «digital multiplexer». Denne kunne da ikke sende gjennom de analoge verdiene vi sendte fra PH og Turbiditetssensorene. Vi bestilte nye analog multiplexer som fungerte slik som de skulle.



Figur 25: Multiplextesting koblingsskjema

9.2 Pådrag

9.2.1 Skjerm

Testing av LCD skjermen gjorde vi allerede 4 uker etter vi fikk oppgaven. Den virket som den skulle bortsett fra vi ikke fikk bruke HDMI porten på RPIen. Vi så at skjermen ikke kom til å bli et problem, men vi ønsket å bruke HDMI-porten under programmeringen og testingen. Derfor reinstallere vi hele Raspbian på RPIen og ventet til siste uke med å installere skjermen igjen.

9.2.2 Motor og Ventil

Med en gang vi fikk strømforsyning og rele til motor og ventil testet vi dem. Vi koblet begge opp og lå styringen inn i Arduino koden og Node-Red programmet. Dette virket på første forsøk og alt gikk som forventet.

9.2.3 Lys

Vi har ikke fått testet lyset enda, da det ble bestilt for seint og har ikke kommet.

9.3 Mosquitto

Vi satte opp to terminaler i RPIen hvor vi i den første skrev “mosquitto_sub -d -t TEST” og i den andre skrev “mosquitto_pub -d -t TEST -m Tester”. Vi så da at det første terminalen som abonnerte på emnet “TEST” fikk tilsendt teksten “Tester” som den andre terminalen publiserte.
Den neste testen var å abonnere og publisere til emne med mikrokontrolleren. Vi fant et testprogram som fungerte som det skulle.

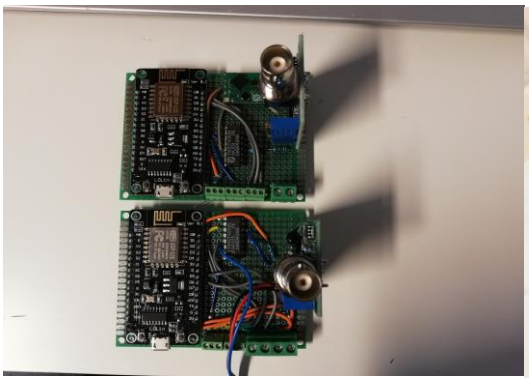
9.4 Node Red

Node-Red testet vi kontinuerlig fra vi sluttet å bruke C# til siste uke når hele systemet ble satt sammen. Mens vi jobbet med sensorer, Mosquitto og Arduino programmet hadde vi Node-Red oppe på skjermen og sjekket at vi fikk de verdiene vi ønsket samtidig som vi endret på utseende av brukergrensesnittet. Etterhvert som vi fikk flere sensorer og større system til å virke sammen i Arduino programmet, fikset vi Node-Red til å passe til det utvidete systemet. Og slik har vi kontinuerlig oppdatert og lært oss nye metoder i Node-Red.

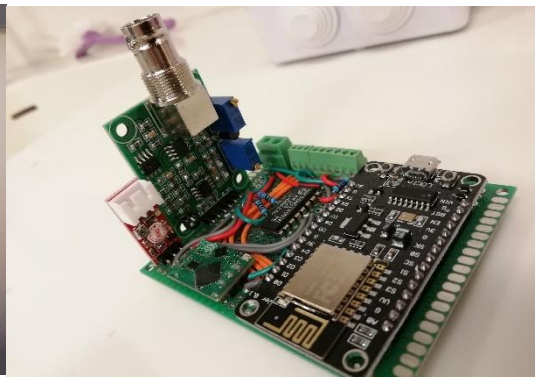
9.5 NodeMCU

Når vi testet sensorene brukte vi NodeMCU. Vi fikk lite problemer med kortet når vi testet de digitale sensorene, men når vi testet de analoge fikk vi rare verdier. Det kom av at alle sensorer og IO pinsene er laget for 5V bortsett fra A0. Dette var vi ikke klar over og fikk da problemer med “metning” når vi skulle teste de analoge sensorene. Da lagde vi en spenningsdeler som skalerte signalet fra 0-5V til 0-3.3V som passet med A0 inngangen. Seinere i oppgaven når vi lagde kretskort fikk vi flere problemer med NodeMCUen. NodeMCUen har en spesiell oppstartsprosedyre som vi ikke var klar over. Den sender signaler på noen av IO pinsene mens krever at noen får 5v og andre får 0v. Dette ble et problem da vi ikke lagde kretskortet i forhold til dette. Vi hadde koblet motor og ventil releene til to utganger som NodeMCUen sendte 5V til ved oppstart. Dette går ikke da det kan være farlig og er dårlig for hele systemet. Derfor måtte vi endre hvilke utganger som styrte releene. PIN 15 kunne ikke få 5V ved oppstart fordi da går NodeMCUen i SD-kort booting, noe vi ikke ønsker. Før vi var klar over dette hadde vi temperatursensoren koblet til PIN 15 som gir over 3V ved oppstarten. Dette gjør at kortet ikke virket i det hele tatt og vi måtte endre på kretskortet.
Når vi var klar for å teste hele systemet fikk NodeMCUen en software oppdatering som gjorde at vi ikke fikk bruke programmet vårt lengre. Dette fant vi ut var fordi den nye versjonen av NODEMCU programvaren klarte ikke å håndtere “Attach-interrupt” som er svært viktig for vannflyt sensoren vårt. Da måtte vi etter mye testing rulle tilbake NODEMCU programvaren til en eldre versjon slik at vi fikk vannflyt målingen vårt til å virke.

9.6 Kretskort



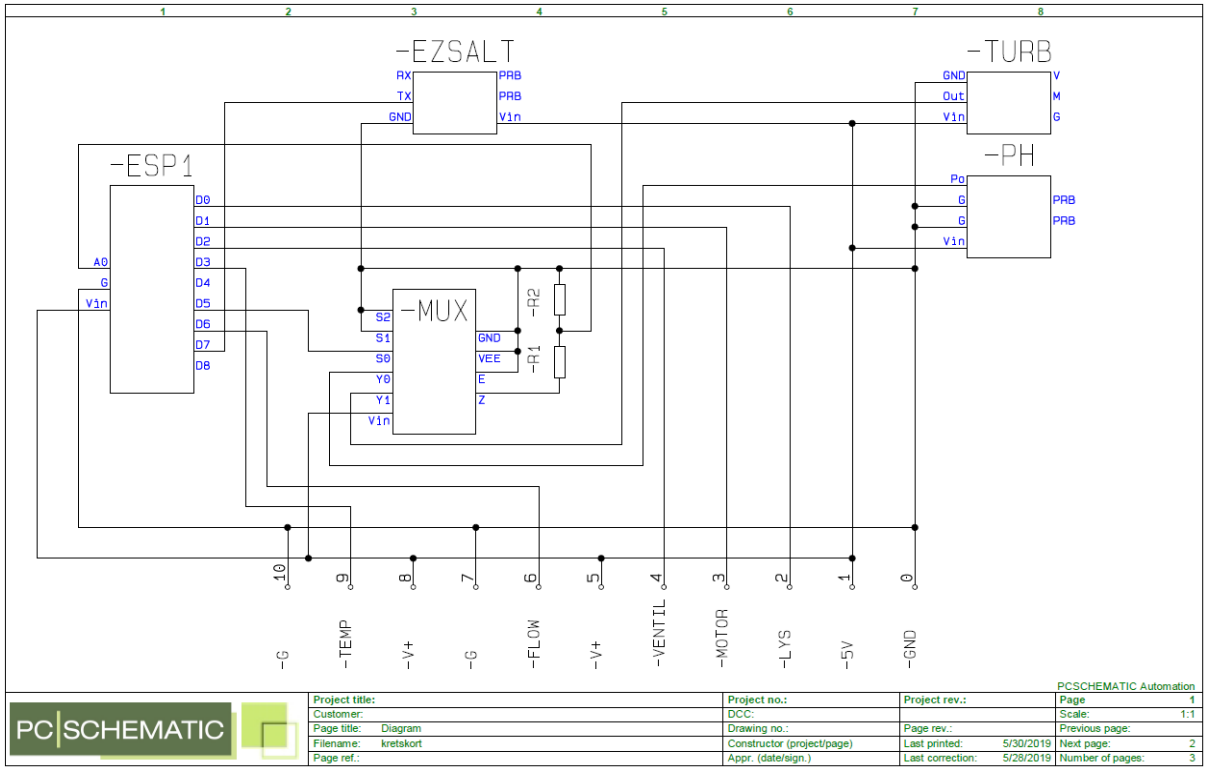
Figur 26: Kretskort Første Prototype



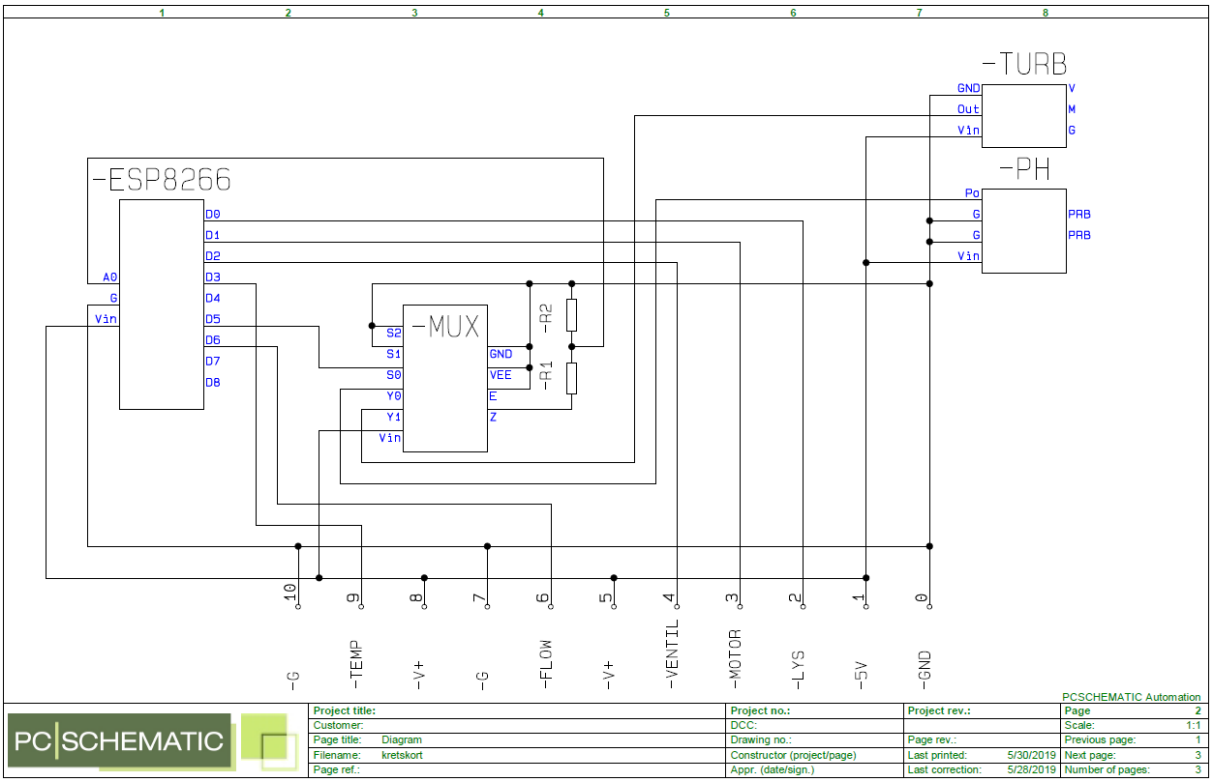
Figur 27: Kretskort final design

Kretskortene er laget på PCB prototypebrett med ferdigdrillede pinnhull. Det er tre forskjellige hoved design på det ferdige kortet. Alle tre er i prinsippet like, men ett er med saltsensor og lysutgang, ett med lysutgang og det siste uten disse to. Oppsettet på alle komponentene er likt og alt kobles likt på alle tre med unntak av de komponentene som ikke er tilstede. Komponentene er loddet på plass og vi bruker ledninger som loddes i hullet ved siden av der de skal. De loddes deretter til korrekt pinne på undersiden av kortet.

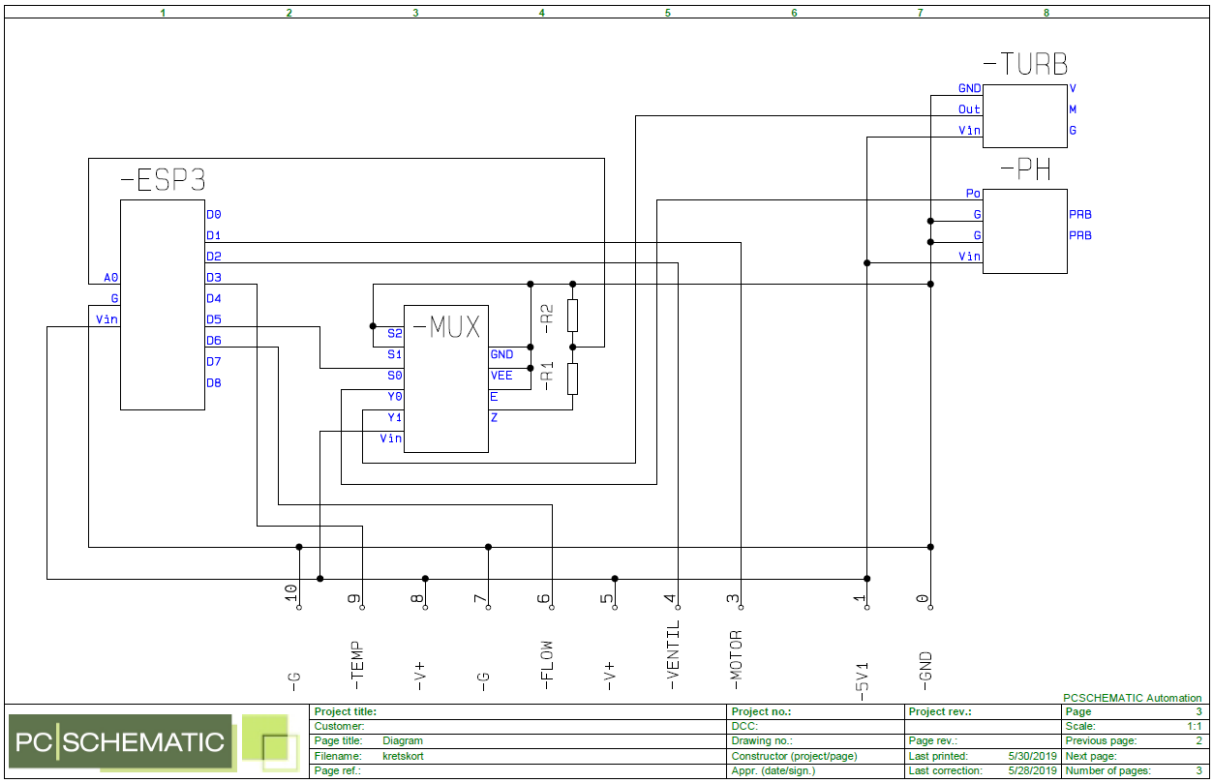
Metoden for å utvikle kretskortet har vært å prøve å plassere komponenter rundt på kortet og se hvor de passer best. Deretter legge opp alle ledningene for å se at det blir plass til de der de skal gå. Til slutt ble de loddet på plass og kortet ble testet. Evt. feil ble utbedret og den ferdigstilte modellen på kretskort nr.1 ser du på *Figur 27*. *Figur 26* er i planleggingsfasen hvor du ser en fungerende prototype av kort uten saltsensor og planleggingen av kortet med saltsensor. Etter at saltsensor kortet ble ferdigstilt ble denne brukt som mal for de andre kortene.



Figur 28: KrekskortSL koblingsskjema



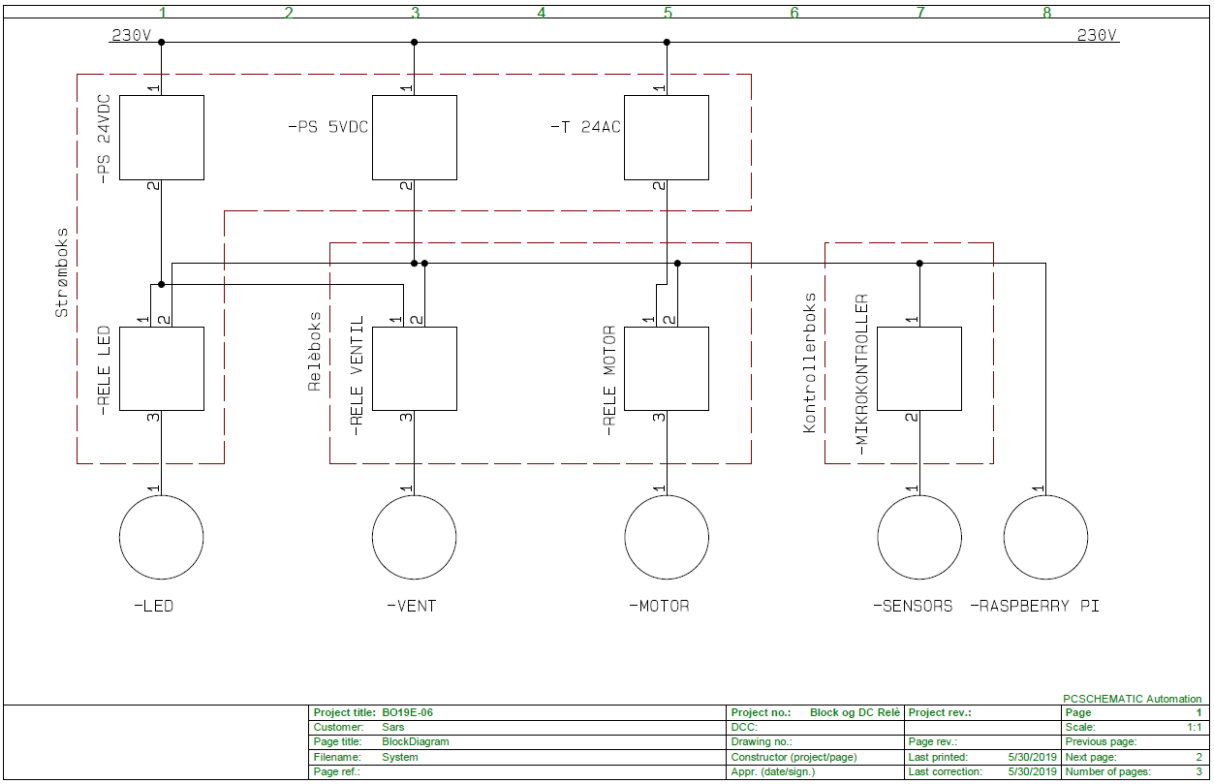
Figur 29: KrekskortL koblingsskjema



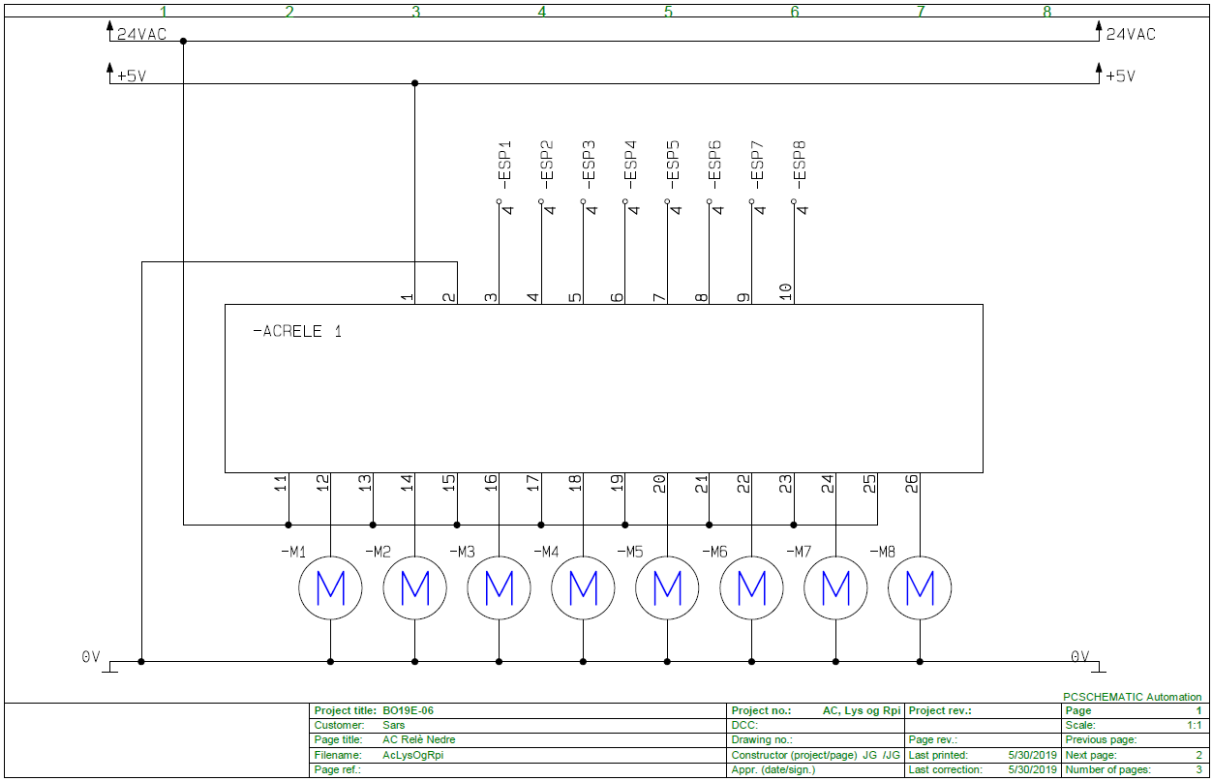
Figur 30: Krekskort koblingsskjema

9.7 Komplett systemet

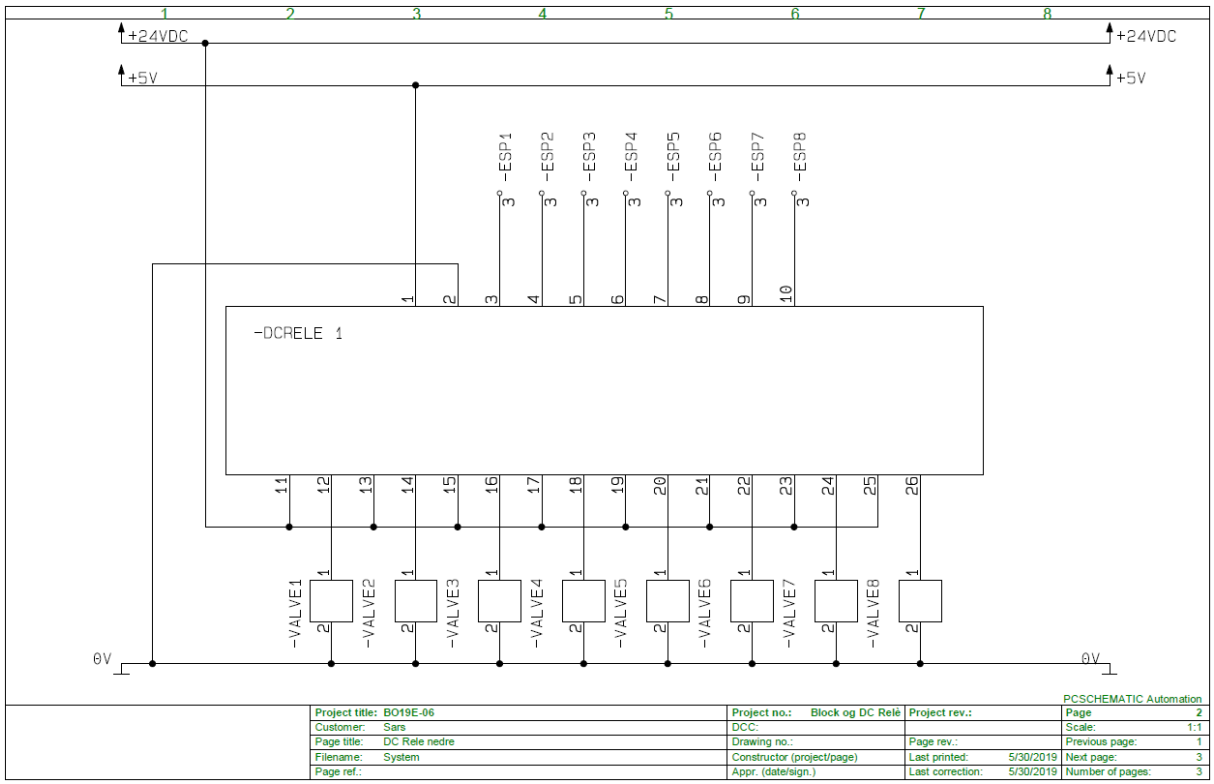
For å teste det komplette systemet har vi montert komponentene i sine respektive bokser og montert boksene der vi ønsket å ha dem. Vi har ikke rukket å lage mer enn en mikrokontroller modul. Vi trakk ledninger og koblet opp etter koblingsskjema. Når alt var koblet opp, satt vi opp serveren og koblet deretter strøm på anlegget. Når det var strømsatt så vi på UI at mikrokontrolleren hadde koblet seg opp og sendte verdier. Vi prøvde å slå på og av motor og ventil, men det fungerte ikke. Vi kontrollerte styrestrømmen og så at det ikke var spenning på den og konkluderte da med at det var software feil. Etter å ha testet på en dummy mikrokontroller fant vi ut at det manglet en linje kode som vi fikk skrevet inn og lastet opp programmet på nytt til anlegget. Når vi nå skulle slå på motor #1 og ventil #1 gikk motor #2 og ventil #2 på. Dette var en feil i kabeltrekkingen som vi byttet om på og etter det fungerte systemet akkurat som det skulle.



Figur 31: Blokkskjema fullt system



Figur 32: AC-Rele Nedre

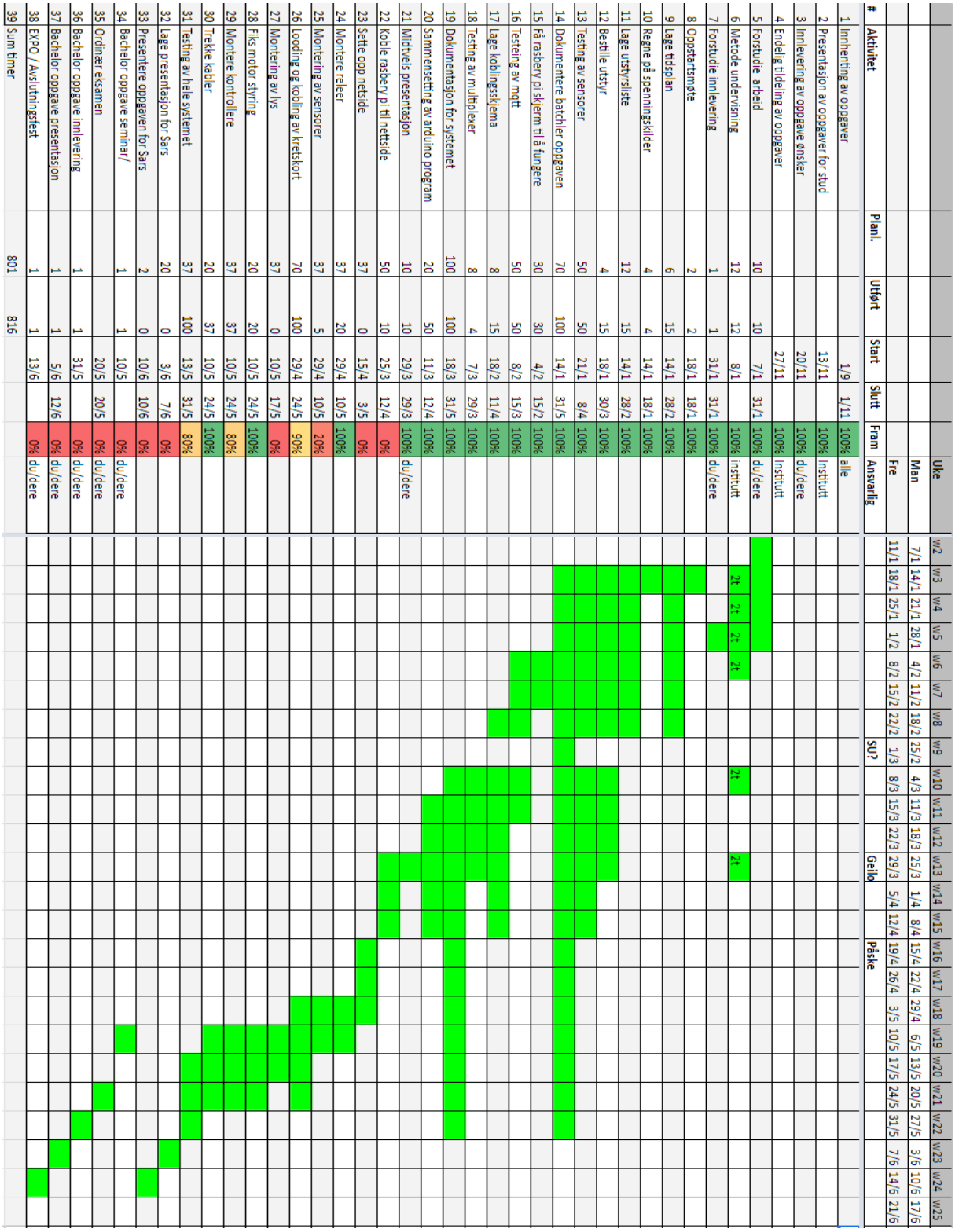


Figur 33: DC-Rele nedre

10 Diskusjon

Vi startet med en ganske dårlig fremdriftsplan som var lite komplett og få oppdelinger i. Dette gjorde at vi i starten ikke brydde oss så mye om hva som skulle bli gjort i forhold til fremdriftsplanen. Men istedenfor hadde en intern muntlig plan om hva og hvordan ting skulle gjøres. Rett før forstudien skulle leveres lagde vi en bedre fremdriftsplan som vi brukte ut resten av prosjektet. Denne fulgte vi ganske bra og den hjalp oss å få oversikt over alt som måtte gjøres. Allerede fra starten havnet vi litt bak skjemaet fordi vi ikke fikk alle delene så fort som vi hadde håpet. Det var også en del problemer med sensorene som var uventet. Først var det temperatursensoren som ikke virket. Vi brukte da unødvendig lang tid på å feilsøke i koden og oppkoblingen når vi skulle teste sensoren. Så var det salt sensoren som vi måtte kjøpe ny av fordi den første vi fikk hadde et måleområde som ikke passet med hva vi trengte å måle.
Når vi prøvde å ta igjen tiden som var tapt gjorde vi en bestillings feil. Da bestilte vi feil multiplexer og feil transformator. Dette var ikke så stort problem i forhold til fremdriftsplanen, men en unødvendig kostnad. Vi greide også å ødelegge 24VDC strømforsyningen vårt ved å koble den i 230V når input bryteren var i 110V og ikke i 230V. Dette var samme tiden som vi feilbestilte slik at det ikke gikk ut over fremdriftsplanen men igjen en unødvendig kostnad. Etter alle disse feilene greide vi nesten å hente oss helt inn igjen på tidsskjemaet fordi Node-Red var veldig intuitivt og lett å bruke. Men når vi nærmet oss tidskjemaet fikk vi nye problemer. Da var det NodeMCUen som ikke fungerte slik vi trodde. Den har en spesiell oppstartsprosedyre som vi ikke var klar over som ga oss mye ekstra jobb når det kom til laging av kretskortet vårt. Flere av IO pinsene blir slått på, av eller er usikre. Noen av IO pinsene kan ikke få spenning og noen må ha spenning ved oppstart. Dette gjorde at vi måtte endre kretskortet vårt helt og gjorde at vi brukte mye lengre tid på lodding og planlegging av kretskortet. I tillegg til dette kom det en software oppdatering på biblioteket til NodeMCUen som gjorde at det ferdige programmet vårt ikke fungerte i det hele tatt. Begge disse problemene gjorde at vi kom skjevt ut i forhold til fremdriftsplanen. Fordi begge problemene kom så seint i prosjektet var det vanskelig å hente inn tiden. Da ble det ferdige systemet litt redusert. Og fikk ikke lagd mer enn en tank og tok vekk lydsignalet som skulle være i systemet.

10.1 GANT- skjema



#	Aktivitet	Planl.	Utført	Start	Slutt	Fram	Ansvarlig	Uke Man	Uke Fre
1	Innhenting av oppgaver			1/9	1/11	100%	alle		
2	Presentasjon av oppgaver for stud			13/11		100%	Institutt		
3	Innlevering av oppgave ønsker			20/11		100%	du/dere		
4	Endelig tildeling av oppgaver			27/11		100%	Institutt		
5	Forstudie arbeid	10	10	7/1	31/1	100%	du/dere		
6	Metode undervisning	12	12	8/1		100%	institutt		
7	Forstudie innlevering	1	1	31/1	31/1	100%	du/dere		
8	Oppstartsmøte	2	2	18/1	18/1	100%			
9	Lage tidsplan	6	15	14/1	28/2	100%			
10	Regne på spenningsskilder	4	4	14/1	18/1	100%			
11	Lage utstyrsliste	12	15	14/1	28/2	100%			
12	Bestille utstyr	4	15	18/1	30/3	100%			
13	Testing av sensorer	50	50	21/1	8/4	100%			
14	Dokumentere batchler oppgaven	70	100	14/1	31/5	100%			
15	Få rasbery pi skjerm til å fungere	30	30	4/2	15/2	100%			
16	Testeing av mqtt	50	50	8/2	15/3	100%			
17	Lage koblingsskjema	8	15	18/2	11/4	100%			
18	Testing av multiplexer	8	4	7/3	29/3	100%			
19	Dokumentasjon for systemet	100	100	18/3	31/5	100%			
20	Sammensetting av arduino program	20	50	11/3	12/4	100%			
21	Midtveis presentasjon	10	10	29/3	29/3	100%	du/dere		
22	Koble rasbery pi til nettside	50	10	25/3	12/4	0%			
23	Sette opp netside	37	0	15/4	3/5	0%			
24	Montere releer	37	20	29/4	10/5	100%			
25	Montering av sensorer	37	5	29/4	10/5	20%			
26	Loading og kobling av kretskort	70	100	29/4	24/5	90%			
27	Montering av lys	37	0	10/5	17/5	0%			
28	Fiks motor styring	20	20	10/5	24/5	100%			
29	Montere kontrollere	37	37	10/5	24/5	80%			
30	Trekke kabler	20	37	10/5	24/5	100%			
31	Testing av hele systemet	37	100	13/5	31/5	80%			
32	Lage presentasjon for Sars	20	0	3/6	7/6	0%			
33	Presentere oppgaven for Sars	2	0	10/6	10/6	0%			
34	Bachelor oppgave seminar/	1	1	10/5		0%	du/dere		
35	Ordinær eksamen			20/5	20/5	0%	du/dere		
36	Bachelor oppgave innlevering	1	1	31/5		0%	du/dere		
37	Bachelor oppgave presentasjon	1	1	5/6	12/6	0%	du/dere		
38	EXPO / Avslutningsfest	1	1	13/6		0%	du/dere		
39	Sum timer	801	816						

Figur 34: GANT-skjema

11 Konklusjon

Vi begynte tidlig med å teste og kalibrere sensorer, men vi manglet noen sensorer og vi jobbet da parallelt med dataoverføring. Vi fikk til MQTT protokoll overføring mye tidligere enn forventet og fant også ut at Node-Red støttet MQTT på en enkel og intuitiv måte. Node Red var en software vi ikke visste noe om før vi begynte, men som dukket opp under utforsking av brukergrensesnitt for RPI. Node Red gjorde at mye av den tunge kodingen forsvant og vi fikk fokusert mer på å lage et godt helhetlig system.
Vi fikk laget den første prototypen før vi fikk den riktige saltsensoren. Det var også en stor milepæl, men det ble gjort en del endringer fra denne prototypen til det endelige designet med saltsensor. Dette grunnet saltsensorkortet tok en del plass på kretskortet og måtte bruke en annen inngang enn forventet.
Etter mye planlegging, testing og bytte av defekte og feile komponenter og sensorer fikk vi tilslutt testet og kalibrert alle sensorene som skulle inngå i prosjektet. Dette var en stor milepæl for vi kunne da ferdigstille planen for oppbyggingen av kretskortet til mikrokontrolleren.
Vi hadde som plan å levere et fullt system for 16 tanker som var ferdig oppkoblet kalibrert og fungerende. Etterhvert som tiden gikk og vi møtte på diverse problemer med måleområde, kalibrering og softwareoppdateringer ble fokuset endret mer mot å få én fungerende modul med god dokumentasjon. Vi har laget en modul med mikrokontroller som er enkel å duplisere. Systemet er enkelt skalerbart til så mange tanker du ønsker. Den eneste software endringen du må gjøre for hver modul er å endre “ID” i Arduino koden.
Når vi hadde fått laget og testet det endelige kretskortet fikk vi montert alle komponentene i sine respektive bokser og monter boksene på plass i anlegget. Testen av hele systemet gikk som planlagt med gode resultater og vi har et fungerende anlegg med overvåkning og styring til én tank.

Fortsettelsen i arbeidet med anlegget er å lage et kretskort og 14 kretskort, montere dem i sine bokser, overføre program med korrekt ID til mikrokontrolleren og trekke de ledningene som trengs til hver boks. Dette skal enkelt kunne gjøres med den dokumentasjonen som fremstår i bacheloroppgaven. Vi rakk heller ikke montere lysene og lys-releene. Programkoden for dette er allerede skrevet men fordi vi ikke har fått lysene enda har vi ikke hatt muligheten til å koble dem opp.

12 Referanser

[1] M. Ford, «Forward.com,» 02 07 2018. [Internett]. Available: https://bit.ly/2JTOO7J. [Funnet 05 05 2019].

[2] Hobby Elecectronics, «Hobbyelectronics.co.uk,» 01 01 2019. [Internett]. Available: https://bit.ly/2Ydk58t. [Funnet 10 02 2019].

[3] K. Dimitrov, «Arduino Project Hub,» 22 11 2016. [Internett]. Available: https://bit.ly/2jVMq1T. [Funnet 25 01 2019].

[4] D. C. Caballero, «Scidle,» 10 03 2017. [Internett]. Available: https://bit.ly/2wvbd2d. [Funnet 06 02 2019].

[5] T. HAREENDRAN, «Elctro Schematics,» 18 03 2019. [Internett]. Available: https://bit.ly/2Xd0rJS. [Funnet 10 04 2019].

[6] Atlas Scientific, «Atlas Scientific,» 01 01 2019. [Internett]. Available: https://bit.ly/2MWeKOy. [Funnet 26 04 2019].

[7] goodtft, «github.com,» 04 12 2018. [Internett]. Available: https://bit.ly/2GOB3pY. [Funnet 05 03 2019].

[8] bennutal, «github.com,» 21 03 2017. [Internett]. Available: https://bit.ly/2JOi1k5. [Funnet 15 02 2019].

[9] T. Varnish, «instructables.com,» 19 02 2018. [Internett]. Available: https://bit.ly/2lyGi0X. [Funnet 02 03 2019].

[10] SeinSmart, «Amazon.com,» 10 05 2015. [Internett]. Available: https://amzn.to/2swehZU. [Funnet 16 03 2019].

[11] SeinSmart, «Amazon.com,» 07 12 2011. [Internett]. Available: https://amzn.to/30W0XxU. [Funnet 16 03 2019].

[12] M. Blackstock, «noderedguide.com,» 03 12 2016. [Internett]. Available: https://bit.ly/2Xx6XLP. [Funnet 14 04 2019].

[13] Phillips, «elfadistrelec.com,» 01 01 1990. [Internett]. Available: https://bit.ly/30Y9P6k. [Funnet 02 04 2019].

[14] monkeyysups, «github.com,» 30 02 2019. [Internett]. Available: https://bit.ly/2EKBBK2. [Funnet 30 02 2019].

[15] J. Puls, «Stackoverflow.com,» 06 02 2009. [Internett]. Available: https://bit.ly/2Mhi4ag. [Funnet 14 05 2019].

Appendiks A Forkortelser og ordforklaringer

24VAC	24 volt vekselspenning
24VDC	24 volt Likespenning
Add-on	Tilleggsmodul
Attach-interrupt	Er en programkode for å gjøre noe med en gang noe skjer
Autofullfør	Fullfører setninger/variabelnavn
Booting	Oppstart av program
Databussystem	Et en-linjes dataoverføringssystem
ESP8266	Et annet ord for mikrokontrolleren
HW	Hardware, fysiske komponenter
IC	Integrated Circuit, elektronisk komponent
IO pins	Inngangs- og utgangs- pinner på NODEMCU-en
IOT	Internet of Things, nettverk av identifiserbare elektroniske gjenstander
NODEMCU	mikrokontrolleren vi valgte å bruke
Metning	Når noe får mer spenning enn maks nivået
Mosquitto	Et program se 8.2.4.2
MUX	Multiplexer
On-site	På anlegget
Open-source	Offentlig programkode
RPI	Raspberry Pi
SD-kort	Solid disk lagringskort
Single-board	Ett-bretts kretskort
Subsystemer	Under-systemer, delsystemer
SQLite	Bibliotek som implementerer en Database
SW	Software, programvare

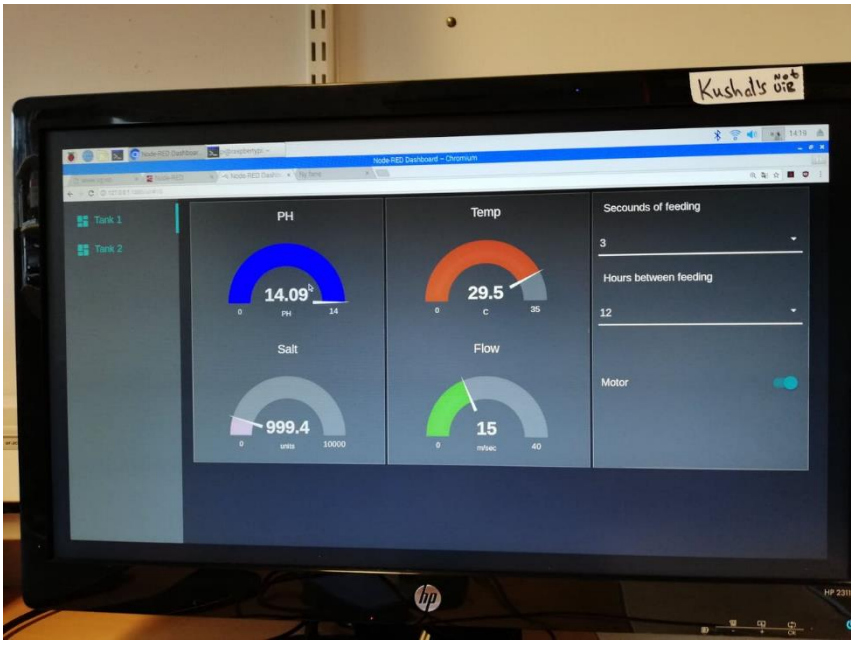
Appendiks B Brukerdokumentasjon

B.1 Brukerdokumentasjon

Når du ser på RPI skjermen vil du se på hovedmenyen som består av en marg på venstre side hvor du kan velge hvilken tank du vil se nærmere på. Du vil se saltverdien i anlegget og minimums og maksimumsverdien det siste døgnet. Her kan du også slå av og på alle ventiler og motorer samtidig. Når du trykker inn på for eksempel tank 1, vil du få opp ett bilde ca. som i *Figur 35.* Her ser du fortsatt den samme margen hvor du kan velge tanker. Du får opp fire måleverdier: Temperatur, PH, turbiditet og vannflyt for den tanken du ser på. Du har også mulighet til å styre blandingsmotoren av og på ved hjelp av en knapp. Med en rullegardinmeny kan du velge hvor mange sekunder matingsventilen skal holde åpen, og en rullegardinmeny hvor du kan velge hvor mange timer det skal gå mellom hver mating.

B.2 Drift og vedlikehold

Når du skal se på sensorverdiene systemet har laget, må du koble fra den eksterne harddisken og koble den til en vanlig pc. PCen må ha et SQL-databaseprogram slik at du kan åpne databasefilen som heter S13WQDB. Da kan du søke og finne alle verdier for systemet som er logget.
Om ingen av tankene viser verdier må du sjekke Wi-Fi tilkoblingen til RPI. Da må du koble til tastatur til RPIen. Trykke alt+F4, da vil du se hjemskjermen til RPIen og et lite Wi-Fi symbol oppe til høyre. Der sjekker du om den er koblet til nettet LAB13, om den ikke er det velg dette nettverket.
Ved strømbrudd på systemet koble til strøm igjen. Se på RPIen at den er koblet til Wi-Fi LAB13 og koble til om ikke. Trykk på terminalen og vent til systemet virker som normalt.
Hver 6. måned skal sensorene renses. Eneste du trenger å gjøre er å løfte dem ut av tanken. Skylle dem i rent vann og tørke av dem.



Figur 35: Brukergrensesnitt V0.8

Appendiks C Utstyrsliste

Navn	Type	Antall	Link	Pris Dollar	Pris NOK	Sum NOK
NodeMCU V3 ESP8266	Mikrokontroller	16	https://amzn.to/2Cjkm0m	6,5		936
Raspberry pi 3	Single Board Computer	1	Montert i anlegget			0
Raspberry pi 3 skjerm	3.5" LCD screen	1	Montert i anlegget			0
74HCT4051N	Multiplex	16	https://bit.ly/2UyFM1x	1,00		144
Power Supply	230VAC-24VDC 360W	1	https://amzn.to/2FGN3rn	25,00		225
Power Supply	230VAC-5VDC 300W	1	https://amzn.to/2BkiNzC	19		171
Transformator	230VAC-24VAC	1	https://bit.ly/2F81Ieq		470	470
PH Sensor	E201-BNC, PH-4502C	16	https://ebay.to/2XfHlml	14,6		2102,4
Temperatur sensor	DS18B20	16	https://amzn.to/2AQ8Ae5	3		432
Vannflyt sensor	YF-S201	16	https://amzn.to/2RAIYwb	9		1296
Saltnivå sensor	Conductivity Probe k1.0	1	https://bit.ly/2Dajxra	119		1071
Saltnivå kort	Conductivity Circuit	1	https://bit.ly/2sgIFqA	60		540
Turbiditet sensor	Gravity	16	https://bit.ly/2MCm0gQ	9,9		1425,6
Relé	solidstate rele DC	2	https://amzn.to/2swehZU	43		774
Relé	solidstate rele AC	2	https://amzn.to/2CJOYIy	21		378
Relé	Relé for lys	2	https://amzn.to/2Pr92FA	11,51		207,18
Led lys	SMD Led Strip	1	https://bit.ly/2WEcDq4		499	499
Mat tilførsels ventiler	Zirai Z350A 24V 9W	16	Montert i anlegget			0
Blandemotor	Crouzet X0231343 24V 8W	16	Montert i anlegget			0
Kretskort	PCB brett	2	https://bit.ly/2W4pb6g		99,9	199,8
boks	LeMotech 7.9"x 6.1" x 3.1"	2	https://amzn.to/2DuYikE	13,99		251,82
boks	LeMotech 15.7" x 13.8" x 4.7"	1	https://amzn.to/2PoUk1M	49,99		449,91
boks	LeMotech 3.9" x 3.9" x 2.8"	16	https://amzn.to/2PtlxjP	9,49		1366,56
kontakt	BNC-kabinetthunn	1	https://bit.ly/2IFiC78		39,9	39,9
Strømkabel	Apparatkabel 5m	1	https://bit.ly/2Gug2h0		89,9	89,9
Ledning	RKUB 0,75 mm² Svart	20	https://bit.ly/2URjSKU		12	240
Ledning	RKUB 0,75 mm² Rød	20	https://bit.ly/2yAWQvH		12	240
Ledning	Jumper Wires	4	https://amzn.to/2sv3grX	11,99		431,64
Rekkeklemme	2pin	2	https://amzn.to/2DbH3VZ	6,85		123,3
Rekkeklemme	3pin	3	https://amzn.to/2RNlcft	10,70		288,9
Resistans	4700 ohm	16	Finnes på Sars			
Resistans	1000 ohm	16	Finnes på Sars			
Resistans	690 ohm	16	Finnes på Sars			
Gjengetapp	1/2" NPT	1	https://bit.ly/30NzheC		200	200
Sub Total	2					14592,91

Appendiks D Figurliste