LIUOTUSNESTEEN SEURANTA JA SÄÄTÖAUTOMAATIO

Prototyypin määrittely ja rakentaminen

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Konetekniikan opinnäytetyö

Konetekniikka, insinööri AMK, Riihimäen kampus

Syksy 2021

Mikko Nurmi

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Konetekniikka, insinööri AMK — Tiivistelmä

Riihimäen kampus

Tekijä	Mikko Nurmi	Vuosi 2021
Työn nimi	Liuotusnesteen seuranta ja säätöautomaatio, prototyypin määrittely ja rakentaminen	
Ohjaajat	Timo Karppinen	

TIIVISTELMÄ

Opinnäytetyön tavoitteena on suunnitella ja valmistaa kuvauskalvojen liotusaltaan pH seuranta- ja säätöautomatiikka. Muiden liuotusnesteen ja -prosessin tilannetta seuraavien antureiden lisäämistä tarkastellaan ja toteutetaan mahdollisuuksien rajoissa. Tarkoituksena on määritellä käytettävien antureiden ja laitteiden ominaisuudet sekä ohjelmoida ja koota sulautettu automaatiojärjestelmä osaksi liuotusprosessia.

Järjestelmän suunnittelu aloitettiin toimeksiannon ja prosessikuvauksen pohjalta. Prototyypille suunniteltiin tarvittavat anturit, näppäimet, näyttö, releet ja kotelo. Prototyypin mikro-ohjain ohjelmoitiin tätä tehtävää varten ja prototyyppi rakennettiin valmiiksi, rujoksi laitteeksi, jolla voidaan arvioida vaaditut muutokset ja kehityskohdat tai käyttää laitetta sellaisenaan osana liuotusprosessia.

Mikäli projektin järkevyyttä ajatellaan taloudellisesti, tulee tehdasvalmisteinen järjestelmä varmasti edullisemmaksi. Valmis laite kuitenkin toimii ja voidaan pienillä muutoksilla ottaa käyttöön tuotannossa.

Avainsanat Automaatio, pH mittaus, sulautettu järjestelmä, prototyyppi

Sivut 34 sivua ja liitteitä 16 sivua

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Machinery, engineer
Riihimäki Campus

Abstract

Author Mikko Nurmi Year 2021

Subject Dissolution process monitoring and controlling automation, prototype definition and building

Supervisor Timo Karppinen

ABSTRACT

The goal of my thesis work was to design and build an automation system that would measure and adjust pH in a dissolution process. Other monitoring and adjusting elements in the dissolution process were to be evaluated and implemented if elements were required and if it was possible to realize them with reasonable effort. The intention was to define features for sensors and other units and program and build an embedded system prototype as part of the existing process.

System designing started based on defining the assignment and the process. The required sensors, buttons, display, relays and casing were designed for this prototype. A micro controller was programmed for this prototype and the process and prototype were built. The intention was to evaluate the changes and development steps for next generation prototypes or use this version as a part of the dissolution process.

Financially this project was not as reasonable as buying a ready-made device would have been. The prototype however works and with minor changes can be used in the process.

Keywords Automation, pH measuring, embedded system, prototype
Pages 34 pages and appendices 16 pages

Sisällys

Liitteet

1 Johdanto

Arvometallien kierrättämisestä ja keräämisestä on tullut houkuttelevaa, kun metallien louhimisesta on tullut epäeettisempää eikä monien arvometalleja sisältävien komponenttien kierrätyksestä ole välitetty tai toiminta ei ole ollut kannattavaa. Kiertotalousajattelun ja kierrätykseen liittyvien lakien ja ohjeiden myötä arvometallien talteenotosta on tullut eettisempää, tiedostavampaa ja kannattavampaa.

Pinnoitetut kalvot sisältävät pieniä määriä hopeaa. Kalvojen liottamisen prosessi sisältää käsillä tehtävää työtä ja -arviointia. Tämän opinnäytetyön tarkoituksena on selvittää liuotusnesteen pH arvon arviointia ja säätöä sekä toteuttaa tähän tehtävään automaatio. Liuotusnesteessä olevan hopean määrää on myös ajateltu seurattavan, jotta tiedetään oikea aika ottaa hopea liuoksesta talteen. Opinnäytetyössäni kartoitan automaation mahdollisuutta myös tähän toimintaan ja mahdollisesti myös toteutan sen, mikäli tehokas arviointikeino löytyy. Opinnäytetyöni keskittyy näiden prosessien arviointiin, kehittämiseen ja toteuttamiseen osana kokonaisprosessia.

Tavoitteena määrittää ja tuottaa toimeksiantajalle valmis automaatiokokonaisuus. Kokonaisuus voidaan liittää sellaisenaan osaksi kokonaisprosessia. Automaatiokokonaisuuden tulee toimia itsenäisesti ja ohjata prosessin käyttäjää toimimaan tehokkaasti.

Laitekokonaisuuden tehtävänä on tarkkailla liuotusnesteen happamuutta. Nesteen happamuus ilmoitetaan pH arvona. Prosessinhoitajan tule olla mahdollista asettaa haluttu pH tavoitearvo, jota laite mittaa. Mitattua pH arvoa verrataan tavoiteltuun arvoon ja tehdään mahdollinen säätö, mikäli pH arvo ei ole halutuissa rajoissa. Tavoitteena on vaatia prosessitilasta vain sähkövirta ja tuottaa muutoin itsenäisesti toimiva laite.

Opinnäytetyöhöni piiriin ei kuulu liuotus- ja käsittelyprosessin muiden vaiheiden kehittäminen tai muokkaaminen. Myös

2 Teoriaosuus

2.1 Prosessin kuvaus

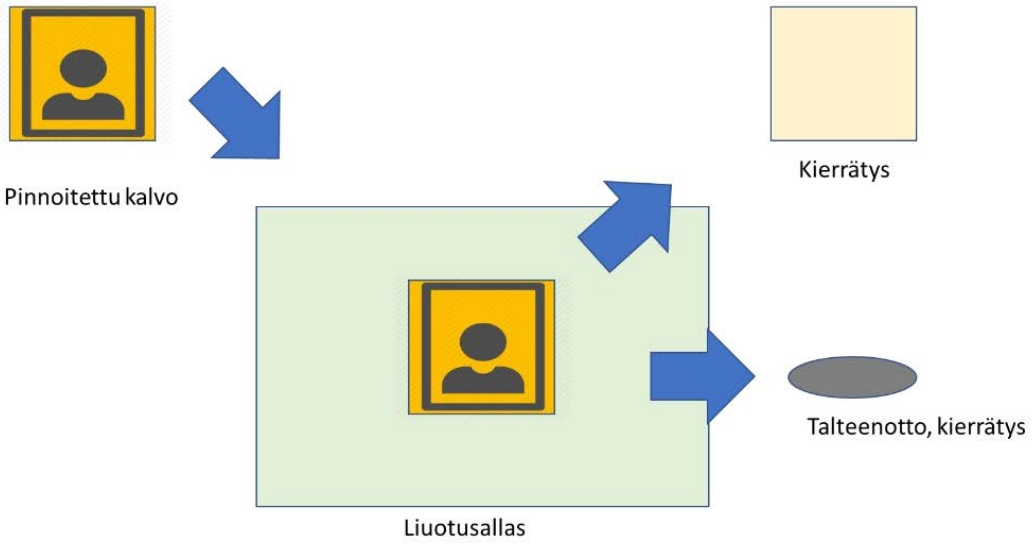
Hopeaa on perinteisesti liottaa kalvoista valkaisuaineella (Natriumhypokloriitti (NaClO)). Kalvoja on myös poltettu, jolloin tuhkasta on saatu kerättyä hopea talteen. (Radiopaedia, 2021).

Natriumhypokloriitilla tehtävä liotus kuluttaa paljon resursseja ja prosessin hyötysuhde on huono. Valkaisuaineella tehtävä prosessi kuluttaa kymmeniä litroja valkaisuainetta saatavaa hopea grammaa kohden. Tällöin taloudelliset hyödyt prosessissa pienenevät. Valkaisuaineella tehtävässä prosessissa ongelmaksi muodostuu myös suuren valkaisuainemäärän hävittäminen tai kierrättäminen.

3R Cycle Oy:n käyttää prosessissa luonnollisista aineista koostuvaa liuosta. Samaa liuosta voidaan käyttää pitkään. Liuoksen hävittämistä tai kierrättämistä ei tarvita. Liuoksen tehokkuutta tarkkaillaan pH arvoa mittaamalla ja liuotusnestettä lisäämällä, pH arvon laskiessa liian alhaiseksi. Prosessissa vanha kalvo upotetaan liuotusaltaaseen. Kalvossa olevat metallit liukenevat nesteeseen ja puhdas kalvo voidaan kierrättää. (Kuva 1). Kun liuos ei enää vastaanota metalleja tehokkaasti, otetaan metallit talteen liuoksesta. (Rilla, haastattelu, 18.11.2020).

Liuoksen pH arvoa mitataan tällä hetkellä käsin, joka on aikaa vievää ja hidasta. Vaivalloisuuden takia mittauksia tehdään harvoin. Kun mittauksia tehdään harvoin, myös prosessin tehokkuus heikkenee ja liuotusajat vaihtelevat. (Rilla, haastattelu, 18.11.2020).

Kuva 1. Kalvon liotus ja keräys.



2.2 Tarvittavien mittausten määrittely

Liuoksesta tarvitaan pH arvo. Liuokselle tarvitaan riittävän korkea pH arvo, jotta liotusprosessi toimii. Tarkan pH arvon määrittämiseksi tarvitaan liuoksen lämpötila. Prosessin pH arvo vaihtelee arviolta välillä 7-11.

Liuotusprosessin redoxpotentiaalia ja liuotusnesteessä olevan hopean määrää ei vielä tarkkailla. (Rilla, haastattelu, 18.11.2020). Redoxpotentiaali kuvaa nesteen valmiutta ottaa elektroneja vastaan ja pelkistyä. Redoxpotentiaalia kutsutaan myös pelkistymispotentiaaliksi.

Hopean määrän tarkkailu mahdollisesti sähkönjohtavuuden kautta voitaisiin tehokkaammin määritellä oikea ajankohta hopean poistamiselle liuotusnesteestä. Liuoksen sisältämän hopean määrää mitataan parhaiten sähkönjohtavuuden arvolla. Kun sähkönjohtavuus nousee riittävän korkealle tasolle, voidaan valmistautua hopean talteenottoon. Tällöin lisättäisiin prosessin tehokkuutta, kun liuotettavien kalvojen määrää ei tarvitsisi tarkkailla. Hopeaa ei myöskään kerättäisi talteen liian usein sillä, hopean erotus keskeyttää liuotusprosessin. (Rilla, haastattelu, 18.11.2020).

Koska liuoksen lisätään pH:ta sääteleviä nesteitä, ilman että liuosta välttämättä haihtuu samaa vauhtia, tulee varmistua liotusaltaan ylivuodoilta nestepinnan korkeutta tarkkailemalla.

Prosessissa pH vaihtelee tyypillisesti 7 ja 11 välillä. Nesteen tehokkuuden turvaamiseksi pyritään nesteen pH arvo pitämään yli 9. Prosessi ei ole kovintaan tarkka pH arvon suhteen vaan säätötarkkuudeksi riittää 0,5. Mittauksen ja säädön tulee kuitenkin olla toistettavaa, jotta ihmisen ei tarvitse olla välissä tarkkailemassa ja säätämässä liotusnestettä. Tällöin henkilö voi keskittyä muihin tehtäviin ja vaihtamaan liotettavia röntgenkalvoja. (Rilla, haastattelu, 18.11.2020).

2.3 Tarvittavien ohjausten määrittely

Prosessin käynnistyminen vaatii sähkövirrankytkemisen. Tällöin antureille, ohjaimelle ja säätimille saadaan virta. Kuvasta 2 käy ilmi hahmotelma järjestelmästä antureineen.

Hätä-seis kytkin tarvitaan käsin painettavana kytkimenä, jolloin prosessi on helppo katkaista, mikäli virhetila huomataan. Hätä-seis kytkin keskeyttää pH säätönesteiden pumppujen releet. Tällöin säätönesteiden virtaus altaaseen lakkaa. Vaikka jätä-seis painiketta painetaan, jatkuu nesteen arvojen mittaaminen silti. Prosessi tulee kytkeytyä pois päältä myös, mikäli nestepinta kohoaa liian korkeaksi.

Liuoksen pH arvon mittaukseen tarvitaan anturi, jolta saadaan analogisia arvoja nesteestä. Jatkuva mittaus ja pidempi kestoikä ovat toivottavia.

Liuotusnesteen pH arvon säätöön käytetään kahta nestettä. Nesteiden lisäys tapahtuu ulkoisista astioista pumpuilla. Pumput saavat virran pistorasiasta (230V). Pumppujen virransaantia ohjataan releellä.

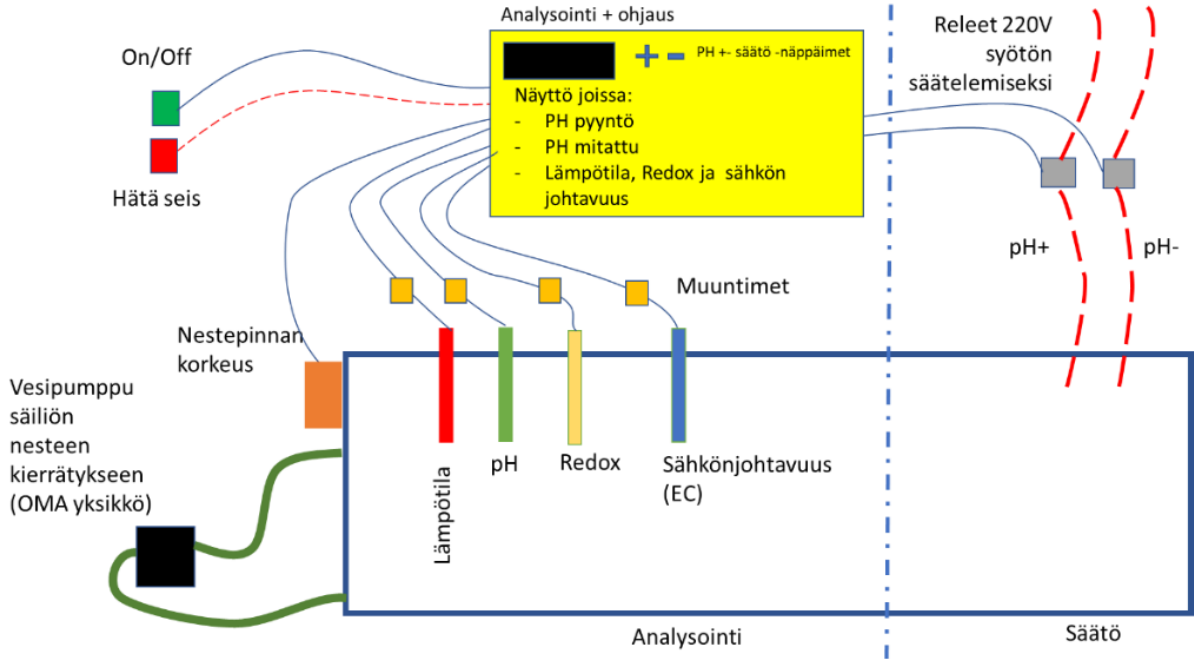
Liuotusaltaaseen on asennettu oma kierrätyspumppu, jonka tarkoituksena on saada neste liikkumaan myös liuotettavien kalvojen välissä. Pumppu toimii 230V virralla itsenäisesti ja kokoaikaisesti.

Mittausten ja säätöjen tulee toimia jatkuvasti vaikkakin hitaasti. Liuotusprosessi vie aikaa, jolloin reagointinopeus esim. pH säädön suhteen voi olla rauhallinen.

Liuoksen sähkönjohtavuusanturi toimii omana kokonaisuutenaan ja sähkönjohtavuutta mitataan manuaalisesti niin tahdottaessa. Sähkönjohtavuudesta saatavan arvon todellinen merkitys löytynee vasta useiden liuotuserien jälkeen. Anturilta tarvitaan analoginen tieto, kuitenkin tieto, että onko arvo paljon vai vähän löytynee ajan kanssa opettelemalla sillä alkuun ei ole tietoa liuotusnesteen sähkönjohtavuudesta tai että millä arvolla liuotusnesteestä saadaan paras hyötysuhde.

Antureita ja säätöjä varten tarvitaan ohjainyksikkö tai kaksi, mikäli sähkönjohtavuus halutaan erottaa kokonaan omaksi järjestelmäksi. Ohjainyksikön lisäksi tarvitaan näyttö, jonka kautta arvoja voidaan asettaa, voidaan tarkkailla mitattuja arvoja sekä tarkkailla prosessin kehittymistä.

Kuva 2. Havainnekuva.



2.4 Käytettävät ohjauksen ja säädön säännöt, logiikka ja matemaattiset algoritmit

Liuoksen pH arvon mittaus ja säätö sekä redox potentiaalin mittaus kulkee seuraavasti:

1. Sähkövirta kytketty.
2. Tarkistetaan nestemäärä anturilta, ettei liuotusnesteen määrä ei ylitä annettua arvoa.
3. Tarkistetaan käyttäjän esiasettama haluttu arvo pH:lle.
4. Mitataan liuotusnesteen lämpötila.
5. Mitataan pH arvo viisi kertaa ja lasketaan mitattujen arvojen keskiarvo, joka toimii pH-mittausarvona.
6. Kirjoitetaan haluttu pH arvo sekä mitatut pH ja lämpötila-arvo näytölle.
7. Lasketaan pH arvo ja verrataan arvoa haluttuun arvon.
    a. Mikäli mitattu arvo on pienempi kuin asetettu arvo → käynnistetään pH+ pumppu.
    b. Mikäli mitattu arvo on suurempi kuin asetettu arvo → käynnistetään pH- pumppu.
8. Mitataan redoxpotentiaalin arvo viisi kertaa ja lasketaan mittausten keskiarvo, joka toimii redox-mittausarvona.
9. Näytetään näytöllä mittauspäivän kymmenen korkeimman mittaustuloksen keskiarvo, joka toimii päivän korkeimpana mittausarvona.
10. Mitataan liuoksen sähkönjohtavuus.
11. Näytetään näytöllä päivän korkein sähkönjohtavuuden mittausarvo sekä edelliset mittausarvot ja niiden päivämäärät.
12. Mittaus lopetetaan, mikäli nestepinnan korkeus kasvaa liian suureksi tai hätäseis painiketta painetaan.

5V käyttöjännitteen analogisilta antureilta saadaan mittaustulos 0-1023. Kun mittaustulos jaetaan 204,6:lla saadaan mitattu jännite.

3 Toteutuksen suunnittelu

Liuotusprosessi kestää tunteja ja on tiedossa, että pH arvo muuttuu liotuksen aikana joitakin yksiköitä. Mittausprosessi voi olla jatkuvaa, ei kuitenkaan vaadita säätöä joka sekunti vaan mittaus voi tapahtua esim. 10 min välein otettavan mittausten keskiarvona.

Mittauksen halutaan tapahtuvan automaattisesti ja ennalta määrätyllä syklillä, sillä prosessin halutaan toimia mahdollisimman itsenäisesti siten, ettei henkilön tarkkailla prosessia.

Liuoksen redox arvo muuttuu lämpötilan ja pH arvon myötä. Tässä mittauksessa redoxpotentiaalin mittauksella on tietoa lisäävä vaikutus, redoxpotentiaalin arvo ei vaikuta automaatioprosessiin.

Liuoksen kyllästyneisyyttä mitataan sähkönjohtavuusanturilla. Tällöin saadaan tieto kuinka paljon sähköä johtavia aineita (hopea) on lionneena liuokseen ja milloin hopea kannattaa ottaa talteen.

Antureiden tulee kestää osittainen upotus nesteeseen, jonka pH pitoisuus voi olla 5-13.

Järjestelmän toteutetaan 5V käyttöjännitteellä.

3.1 PH Anturi ja muunninpiiri

PH anturilta vaaditaan 1-13 mittausalue ja toiminta 15-35˚C lämpötilassa. Anturin tarkkuuden tulee erottaa ±0,1 vaihtelu pH arvossa. Mittaukset tulee olla toistettavia ±0,1 tarkkuudella kuukauden jakson ajan ilman kalibrointia. Reagointiajaksi riittää 30 sekuntia.

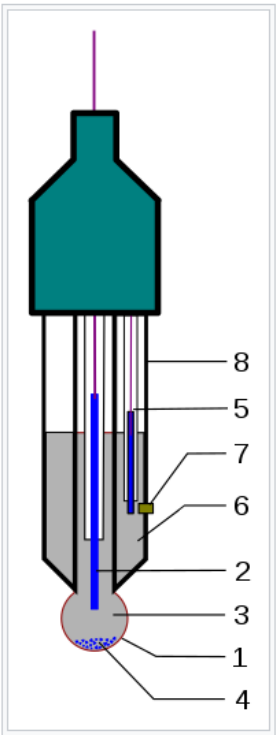
Anturivahvistimen tulee sisältää potentiometri, jolla anturin mittaustulosta voidaan säätää tehtyjen kalibrointien mukaan.

PH anturin mittauspään tulee kestää jatkuvaa upotusta nesteeseen.

3.1.1 PH anturin toiminta

PH anturissa tyypillisesti kaksi tai kolme anturia, joista toinen tekee mittausta (mittauselektrodi) ja toinen anturi toimii vertailuanturina (referenssielektrodi). Referenssielektrodi on pH mittarin sisällä ja upotettuna nesteeseen, jonka pH arvo tiedetään tarkasti. Mittausanturin muuttuvaa vastusta verrataan vertailuanturiin, jonka jälkeen mitatun nesteen pH arvo voidaan laskea. (Kuva 3.)

Kuva 3 - PH anturin leikkauskuva (Wikipedia, 2021.-a)



pH-mittari, jolla on anturina yhdistelmäelektrodi:
1–lasielektrodin reagoiva lasi
2–sisäinen elektrodi (AgCl)
3–1,0·10^{-7} mol/l HCl-liuos
4–AgCl (kiinteä)
5–referenssielektrodi (AgCl)
6–0,1 mol/l KCl-liuos
7–kapilaarinen aukko liuokseen
8–anturin runko (ionitiivis)

Kaikkien pH-elektrodien toiminta perustuu Nernst:n yhtälöön. Nernst:n yhtälössä jännitelukema (mV) muutetaan ionikonsentraatioksi (tai pH:ksi). Tämä korrelaatio on suoraan verrannollinen. PH-elektrodeille teoreettinen mV-arvo on 0 mV, kun pH on 7 ja kulmakerroin on 59,16 mV/pH. Tämä tarkoittaa sitä, että aina, kun pH-muuttuu yhden yksikön, niin jännite muuttuu 59,16 mV. Kaikki tämä on teoriaa, sillä elektrodien

kulmakerroin muuttuu niiden ikääntyessä. Todellisuudessa elektrodit saattavat käyttäytyä hieman eritavoin, kuin teoriassa (esim. kulmakerroin voi olla 58,2 mV/pH ja poikkeama 8 mV). (Pietiko, 2021).

3.1.2 PH anturimuuntimen toiminta ja mittaustiedon siirtyminen mikro-ohjaimeen

Anturin tuottama analoginen tieto muutetaan anturimuuntimessa digitaliseen muotoon, jota mikro-ohjain osaa tulkita. PH anturi liitetään muuntimeen esimerkiksi BNC liittimellä. Anturimuuntimessa olevalla potentiometrillä, voidaan tehdä säätö mittaustulokseen, kun anturia kalibroidaan.

3.2 Redox-potentiaalianturin toiminta

Redox-potentiaali on sähköinen potentiaali, joka tarvitaan elektronien siirtämiseksi aineesta toiseen. Redox potentiaalin mittaus tapahtuu pH mittauksen tapaan. Mittaus- ja referenssielektrodien välinen jännite-ero ilmaiseen liuoksen hapettumis- pelkistymisreaktion tasapainon. Positiivisempi redox-arvo ilmaisee liuoksen olevan vahvempi hapetin, kun taas negatiivisempi arvo ilmaisee heikomman hapettimen. (Wikipedia, 2021 -b).

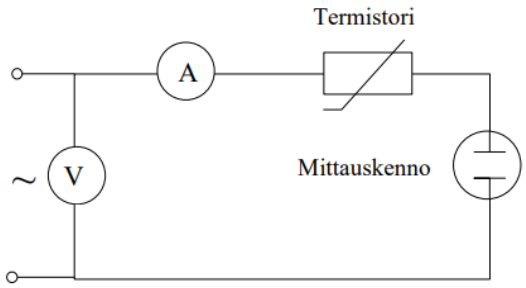
Redox-potentiaalin mittausalue tulee olla 0-1000mV ja erotuskyvyn 1mV. Tarkkuuden tulee olla ±1%

Redox-potentiaalianturin mittaustieto siirtyy mikro-ohjaimeen vahvistimelta. Anturi kytketään vahvistimeen esim. BNC liittimellä.

3.3 Sähkönjohtavuusanturin toiminta

Johtavuusanturin klassinen rakenne on yksinkertainen. Anturissa on kaksi metallielektrodia sijoitettuna eristemateriaalista valmistettuun kammioon. Anturin valmistaja ilmoittaa kullekin anturille kennovakion arvon. Kennovakio (k) on elektrodien välisen etäisyyden (l) ja pinta-alan (A) suhde. Kun anturin kennovakio tunnetaan, johtavuus saadaan mittaamalla elektrodien välinen resistanssi. Johtavuutta mitataan kuvassa 1 esitetyllä resistanssimittauksella. (Oulu, 2021).

Kuva 4. Johtokyvyn mittaus (Oulu, 2021).



Sähkönjohtavuuden mittaustulos riippuu liuoksen lämpötilasta.

Tulosten laskeminen. Astiavakion arvo voidaan joko asettaa suoraan mittariin, jolloin laite ilmoittaa tuloksen suoraan tai tulos saadaan kertomalla mittarin lukema astiavakiolla γ = kG, jossa

γ = sähkönjohtavuus mS/m
k= astiavakio m-1
G= konduktanssi eli laitteen antama lukema mS
(Oulu, 2021)

Tulosten tarkkuus ja toistettavuus ovat riippuvaisia laitteistosta ja näytteet mittauksesta. Hyvällä suoritustavalla ja laitteella on mahdollisuutta päästä 5% tarkkuuteen ja toistettavuuteen. (Opetushallitus, 2021.-b).

Taulukossa 1 on veden sähkönjohtavuuden korjauskertoimet lämpötilaan 25°C. Kertoimet perustuvat 0,01 mol/l KCl- ja 0,01 mol/l NaNO3-liuokseen.

Taulukko1. Veden sähkönjohtavuuden korjauskertoimet lämpötilaan 25°C. (Opetushallitus, 2021.-b).

t /°C	kerroin	t /°C	kerroin	t /°C	kerroin
32	0,89	22	1,06	12	1,30
31	0,90	21	1,08	11	1,33
30	0,92	20	1,10	10	1,36
29	0,93	19	1,12	9	1,39
28	0,95	18	1,14	8	1,42
27	0,97	17	1,16	7	1,46
26	0,98	16	1,19	6	1,50
25	1,00	15	1,21	5	1,54
24	1,02	14	1,24	4	1,58
23	1,04	13	1,27	3	1,62

Sähkönjohtavuusanturilla on oma vahvistimensa. Sähkönjohtavuusanturi liitetään vahvistimeen esimerkiksi BNC -liittimellä. Sähköanturin tulee pystyä mittaamaan alueella 0-2000 µs/cm. Erotuskyvyn tulee olla 1% muutos ja mittausvirhe saa olla 5%. (Opetushallitus, 2021.-b).

3.4 Lämpötila-anturi

Vastuslämpötila-anturi mittaa liuoksen lämpötilaa. Vastuslämpötilamittarin mittaus perustuu anturin sähkön resistanssin muuttumiseen. Lämpötilan noustessa anturin sähkönvastus (resistanssi) kasvaa.
Lämpötila-anturin mittausalueen tulee olla 15-30˚C. Erotustarkkuuden 0,5˚C ja mittaustarkkuuden ±1%.

3.5 Nestepinnan tunnistinanturi

Hankitaan anturi, joka tunnistaa vain ylivuodon. On hyvin epätodennäköistä, että säiliössä tulisi olemaan liian vähän nestettä, jolloin pelkkä ylivuoto on riski. Nestepinnan anturin tulee tunnistaa pinnan korkeuden nousu asetetun arvon ylitse. On parempi, jos anturi saadaan asennettua liotusaltaan ulkopuolelle, jottei altaan sisällä ole turhia osia häiritsemässä prosessia. Anturiksi voisi soveltua esimerkiksi altaan ulkopuolelle asennettava kapasitiivinen anturi. Kapasitiivinen anturi vertaa sähkökentän muutosta väliaineessa, permittiivisyyttä.

Anturi havaitsee muutoksen sähkökentässä. Anturia voidaan käyttää esim. muovisten, lasisten tai lasikuituisten altaiden kanssa. Anturi ei sovellu käytettäväksi metallisissa altaissa.

3.6 Mikro-ohjain

Valittavan mikro-ohjaimen tulee olla universaali ohjelmoitava ja laajalti tunnistettava. Mikäli säätöjä tai laajennuksia tehdään tulevaisuudessa, tulee ohjelmointi olla mahdollista muidenkin kuin laitteen rakentajan toimesta. Ohjelmointi kielen tulee olla mielellään C++ tai C Python.

3.7 Ulkoinen muisti

Selvitetään mahdollisuutta tallentaa mittaustietojen historiaa. Mittaustietoa halutaan tallentaa, jolloin tarvitaan SDRAM muistia.

Muistille tulee tallentaa määreet: mittausajankohta, pH mittausarvo, lämpötila, redox arvoa ja sähkönjohtavuusarvo. Esimerkiksi Arduino Uno liukuluku -muuttuja on 32-bittinen arvo.

Mikäli pH anturilta otetaan mittaustietona 5 mittausta 10 min välein, joista lasketaan keskiarvo, saadaan muistille tarvittavaksi summaksi 8410 kilotavua /vuodessa (32 bittiä mittauksesta x 5 mittausta x 6 kertaa tunnissa x 8760 tuntia vuodessa). Mikäli halutaan tallentaa myös mittausajankohta, lämpötila, redox ja sähkönjohtavuusarvot niin tarvitaan vuoden mittauksille 42048 kt muistia. Kun varaudutaan 5 vuoden mittauksille, tulee muistia varata vähintään 32 Mt. Muistipaikkana voidaan käyttää SD korttia, jolloin muistin luku tietokoneellakin onnistuu helposti.

3.8 Näyttö

Näytöllä tulee olla tiedot: valittu pH arvo, mitattu pH arvo, liuoksen lämpötila, mitattu sähkönjohtavuus aikaisemmin, päivämäärä, jolloin arvo on mitattu sekä nyt mitattu sähkönjohtavuuden arvo. Kosketusnäyttöä ei toivota, sillä työskentely prosessissa tehdään suojahanskat kädessä. Tarvittaneen kaksi kappaletta 4 rivistä ja 20 merkkistä LCD näyttöä. Mikro-ohjaimen pinnimäärän käytön minimoimiseksi toteutetaan LCD näyttöjen ohjaaminen

I2C muuntimen kautta. I2C muunnin varaa mikro-ohjaimelta vain kaksi analogista ulostulopinniä.

LCD näyttöjen käyttöjännite on oltava 5V.

3.9 Ohjaus releet

PH arvon säätöön käytetään nesteitä, joita pumpataan liuokseen. Pumppuja ohjataan releillä, joita mikro-ohjain käskyttää. Releen tulee pystyä ohjaamaan 5V järjestelmällä 230V pumppuja.

3.10 Muut osat

Hätä-seis painikkeella tulee pystyä pysäyttämään prosessi ja pumput.

Tarvitaan kaksi painiketta pH arvon valintaan sekä yksi painike arvojen hyväksymiseen.

Mikro-ohjain tarvinnee roiskeilta suojaavan kotelon. Kotelon sisään tulee mahtua mikro-ohjaimen lisäksi kytkentälevyt, releet sekä sähkökaapeloinnit.

4 Toteutus

4.1 Valitut komponentit

Prototyyppiä lähdettiin hahmottelemaan edullisilla komponenteilla, joita oli saatavilla nettikaupoista sekä Etelä-Suomessa sijaitsevista elektroniikka-alan kaupoista. Osa materiaalista oli olemassa jo ennen projektin aloittamista.
Yleisesti anturit haluttiin BNC-liittimellä antureiden vaihdettavuutta ja asennusta ajatellen. Kaikille antureille oli myös oma anturimuunnin, jonka vuoksi laitteen rakentaminen ja helpottui.

4.1.1 PH anturi

Anturiksi valikoitui edullinen, geneerinen DIY More-merkkinen anturi ja anturimuunnin PH 4502-C. Anturi ja anturimuunnin oli hankittu jo ennen tarkkaa anturimäärittelyä ja koska anturin tekniset ominaisuudet riittivät ja anturi on suoraan yhteensopiva Arduino-mikro-ohjaimen kanssa. Päätettiin käyttää tätä anturi. Kalibrointiohjetta ei valmistajan sivulta löydy. Myöskään tietoa jatkuvasta upottamisesta ei mainita valmistajan tiedoissa. Anturin todetaan soveltuvan veden pH mittaukseen esim. uima-altaissa ja akvaarioissa. Kuvissa 6 ja 7 pH anturi ja pH anturimuunnin.

DIY More pH anturin tekniset ominaisuudet:

- mittausalue: 0-14 pH
- toleranssi ±1 (25 ˚C)
- toimintalämpötila on 5-60˚C
- neutraali piste: 7 ±0,5
- sisäinen vastus <250MΩ

Kuva 4. PH anturi (DIY More, 2021.-a).

Anturimuuntimen hyvänä puolena on valmiiksi integroitu potentiometri, jolla anturin antamaa jännitettä (mittaustulos) voidaan säätää ja näin anturin kalibroiminen on helppoa ja nopeaa. Anturimuuntimen käyttöiäksi valmistaja arvioi 3 vuotta.

Anturimuunnin PH 4502-C tekniset ominaisuudet:

- virrankulutus 5mA
- mittausalue 0-14 pH
- käyttölämpötila: -10-50˚C
- vasteaika: alle 5s
- materiaali: ABS
- suojaus: IP67
- kaapelin pituus: 50cm

Kuva 5. PH anturin anturimuunnin (DIY More, 2021.-b).

4.1.2 Redox anturi

Anturiksi valikoitui myös edullinen Grove ORP anturi ja anturimuunin, joka on suoraan soveltuva Arduino-järjestelmään. Anturi liitetään anturimuuntimeen BNC liittimellä. Anturin tekniset tiedot riittivät määriteltyyn vaateeseen. (Kuva 8).

Redox anturin tekniset ominaisuudet:

- elektrodin potentiaali 245-270mv (15-30 ˚C)
- referenssi elektrodin sisäinen vastus ≤10k
- elektrodin resoluutio ±8mv
- käyttölämpötila: 5-70˚C
- käyttöjännite: 3,3V/5V

Kuva 6. Redox anturi (Seeedstudio, 2021.-b).

4.1.3 Sähkönjohtavuus anturi

Anturioksi valikoitui edullinen Grove EC sensor, joka oli saatavilla samasta nettikaupasta kuin Redox-anturi. Yhtä lailla anturin tekniset tiedot riittivät määriteltyihin arvoihin ja anturin mukana tuli anturimuunnin johon anturi yhdistyy BNC- liittimellä. (Kuva 9).

Grove EC sähkönjohtavuusanturin tekniset ominaisuudet:

- käyttöjännite: 3,3V/5V
- mittausalue_ 0-2000µs/cm
- resoluutio: ±5mv
- käyttölämpötila: 5-80˚C
- vasteaika: <10s

Kuva 7. Sähkönjohtavuusanturi (Seeedstudio, 2021.-a).

4.1.4 Lämpötila-anturi

Anturiksi valikoitui RTD PT100, kolmijohtoinen, ruostumattomasta teräksestä valmistettu anturi, joka liitettiin MAX31865 anturivahvistimeen. Anturin nimi tulee sanoista Resistance Temperature Detector ja numero 100 viittaa siihen, että anturin vastus 0˚C on 100mΩ. (Kuva 10).

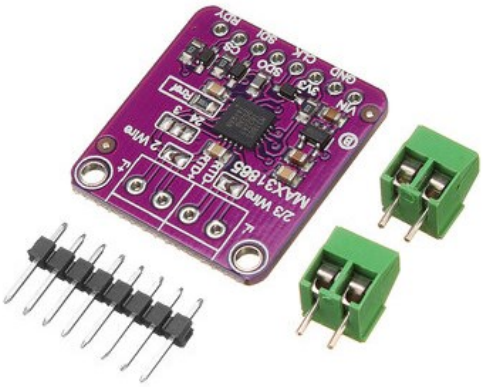
Anturivahvistin liitettiin Arduino Uno mikro-ohjaimen SPI-väylään. Anturivahvistin toimii 5V tai 3.3V järjestelmässä. MAX 31865 anturivahvistimessa on ylijännitesuojaus ±45V saakka. (Kuva 11).

PT 100 anturin tekniset ominaisuudet:

- vastus 100Ω 0˚C lämpötilassa
- vastuksen muutos 0,385Ω /°C

Kuva 8. PT 100 lämpötila-anturi (Circuits DIY, 2021)

Kuva 9. MAX31865 anturivahvistin (DIY More, 2021.-c).



4.1.5 Nestepinnan anturi

Anturiksi valittiin altaan ulkopuolelle sijoitettava XKC-Y25-NPN anturi, joka toimii 5-12V jännitteellä. Anturi soveltuu käytettäväksi muovisten, lasisten tai keraamisten astioiden kanssa, joiden paksuus voi olla jopa 20mm. Anturi tunnistaa induktion eli sähkömagneettisen muutoksen. Anturin herkkyyttä on mahdollista säätää anturimoduulin ”set”-painikkeella. Anturi tulee asentaa liuotusaltaaseen siten, ettei prosessin osat tai esim. prosessin hoitajan käsi tule liian lähelle anturia ja näin pysäytä prosessia. (Kuva 12).

Kuva 10. XKC-Y25-NPN nestepinnantunnistin (IC Station, 2021).

Koska liuotusastia on muovia, voidaan nestekorkeus tunnistaa anturilla altaan läpi. Näin altaan sisälle ei tarvita anturia (esim. uimuri) ja liotusprosessi ei häiriinny. Kun nestepinnan anturi tunnistaa sähkömagneettisen säteilyn määrän muutoksen, lähettää anturi signaalin mikro-ohjaimelle. Mikro-ohjaimen ohjelmoinnilla päätetään mitä tapahtuu, kun signaali saapuu.

YKC-Y25-NPN anturin tekniset ominaisuudet:

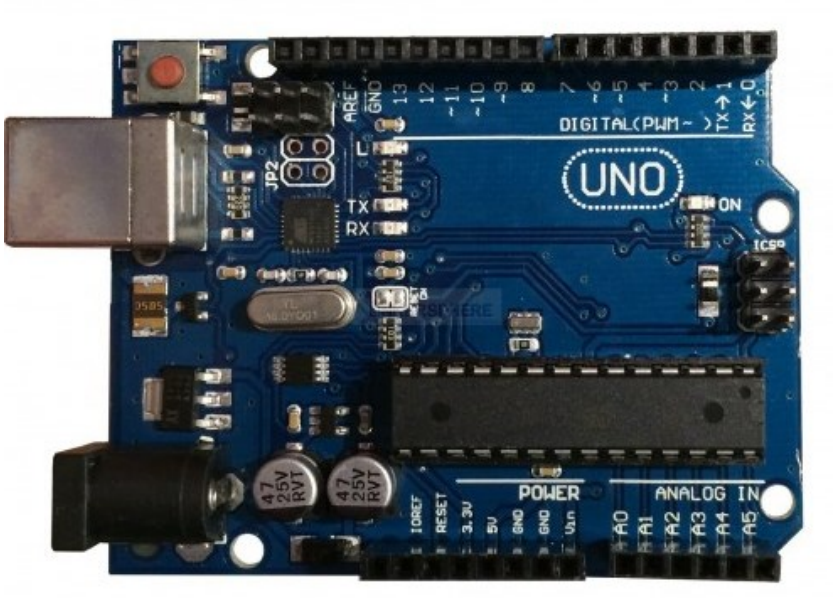
- käyttöjännite: DC 5-12V
- virrankulutus 5mA
- lähtövirta: 1-200mA
- vasteaika: 500ms
- käyttölämpötila: 0-100˚C
- tunnistuspaksuuden väli: 0-20mm
- liitäntä: NPN
- kosteus: 5-100%
- materiaali: ABS
- suojaus: IP67
- kaapelin pituus: 50cm

4.1.6 Mikro-ohjain

Arduino ympäristö oli alkujaankin selvä valinta sen helpon ohjelmoitavuuden ja edullisuuden vuoksi. Arduino ympäristö on luotu prototyyppienrakentamiseen ja muokkaamiseen. Antureiden liittäminen on helppoa. Valitut anturit, anturivahvistimet ja -muuntimet tukevat Arduino käyttöä. Valituista antureista löytyy lisäksi valmiita koodikirjastoja Arduinolle. Arduinolla tehtyä järjestelmää on tulevaisuudessa helppo muokata. Muokkausta voi helposti tehdä myös joku muu henkilö.

Kun antureiden, releiden ja näyttöjen tarvitsemat liitäntämäärät tiedettiin, valittiin mikro-ohjaimeksi Arduino Uno. (Kuva 13). Mikro-ohjaimen pinnit riittävät juuri ja Uno on edullinen. Aiemmin oli hankittuna Arduino Uno V3 kopio, joka päätettiin ottaa käyttöön. Arduino Uno on yksi tunnetuimmista ja edullisimmista mikro-ohjaimista. Mikäli antureiden ja liitettävien laitteiden liitinmäärä olisi vielä kasvanut, olisi siirrytty Arduino Mega 2560 mikro-ohjaimeen.

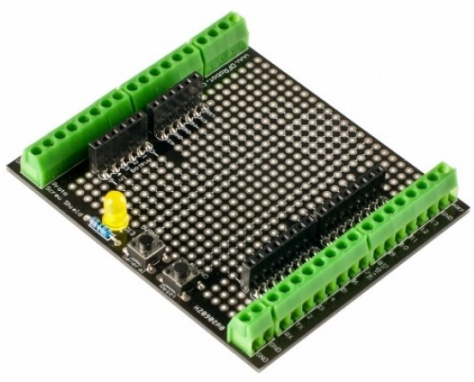
Kuva 11. Mikro-ohjain (Tinkersphere, 2021).



Liitäntöjä varten tarvittiin lisää virran ulostuloja (5V) ja maadoituspinnejä. Lisäksi kaapeleiden liittämiset haluttiin tehdä ruuviliitoksin, jotta liitoksia voidaan muuttaa tarpeen mukaan. Alkuperäisessä mikro-ohjaimessa ei ole ruuvattavia liittimiä. Tähän tarkoitukseen

hankittiin erillinen Arduino Unon päälle asennettava liitin levy. Shield: DFRobot Proto Screw Shield-Assembled (Arduino Compatible). (Kuva 14).

Kuva 12. Asennuslevy ruuviliitoksilla (DFRobot, 2021).



4.1.7 Näyttö

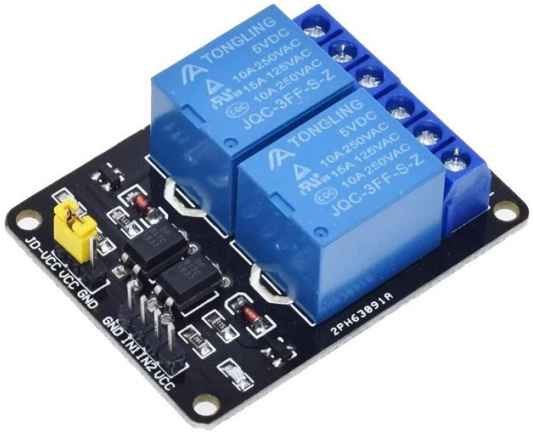
Tuotantoprosessin yhteydessä toimivassa laitteessa tarvitaan selkeät merkit ja kosketusnäytön sijaan näppäimet ovat parempi vaihtoehto. Näin laitetta voidaan käyttää käsineet kädessä ja suojalasit silmillä. Tämän vuoksi näytöksi valikoitui 20 merkkiä leveä ja neljä rivinen LCD näyttö. Näyttö liitetään I2C väylästä Arduinon SPI väylään. Näytön etuna on edullisuus, kirkkauden säätö ja helppo ohjelmoitavuus. (Kuva 15).

Kuva 13. LCD näyttö 20x4 merkkiä (Elektroniikkaosat, 2021)

4.1.8 Releet

Järjestelmään tullaan yhdistämään pumput, jotka mikro-ohjaimen käskystä käynnistyvät ja sammuvat. Tätä varten tarvittiin kaksi relettä, yksi molemmille pumpuille. Releiksi valittiin kahden JQC-3FF-5-Z releen moduuli. Tuote: Tongling JQC-3FF-5-Z 5V. Releitä voidaan käyttää 5V jännitteellä ja releet kestävät 10A ja 250V jännitteen. Releitä voidaan ohjata suoraan mikro-ohjaimella. (Kuva 16).

Kuva 14. Releet (Ihmevekotin, 2021).



4.1.9 Muut komponentit

Koteloksi hankittiin muovinen asennuskotelo. Kotelon mitat ovat 310mm x 240mm x125mm. Koteloon saadaan helposti reikiä liittimille ja kotelo on varmasti tarpeeksi tilava. (Kuva 17).

Kuva 15. Kotelo (Puuilo, 2021).

Projektin aikana päätettiin luopua ylimääräisestä SD muistista. Todettiin, ettei mittaushistoria ole tarpeellinen tällaisessa prototyypissä eikä mittaushistoria tuo lisäarvoa prosessiin.

Painikkeet ja verkkovirtamuuntaja löytyivät itseltä jo aiemmin hankittuina.

Antureiden siistin ja toiminnallisen käsittelyn varmistamiseksi liuotusaltaassa, mallinnettiin antureille pidin. Pidin 3D tulostettiin PETG muovista, jolla on hyvä kesto esim. ultraviolettivaloa vastaan. Anturit kiinnitetään pitimeen, jolloin anturit pysyvät yhdessä eikä liotusprosessia häiritä.

4.2 Komponenttien liittäminen

Järjestelmä saa käyttövirran verkkovirrasta 7,5V muuntajan kautta. Käyttövirta kulkee Arduino mikro-ohjaimen läpi ja Arduinon ohjelmointi ja virran ulostuloilla voidaan määritellä komponenttien saama virta.

Johdotukset anturimuuntimiin tehtiin juottamalla ja lisäämällä juotoksen ympärille kutistesukka. Kutistesukan tarkoitus on suojata ja vahvistaa juotosliitosta sekä estää paljaiden johtojen toisiinsa osuminen, joka voi aiheuttaa signaali tai sähköongelmia. Arduinoon johdot liitettiin ruuvaamalla johdotukset kiinni Arduino Proto shield’in, jossa pinneihin voidaan liittyä ruuviliitoksin. Ruuviliitoksen hyvä puoli on, että muutokset

onnistuvat helposti ja komponenttien vaihtaminen, lisääminen tai poistaminen onnistuu helposti.

4.3 Mikro-ohjaimen ohjelmointi

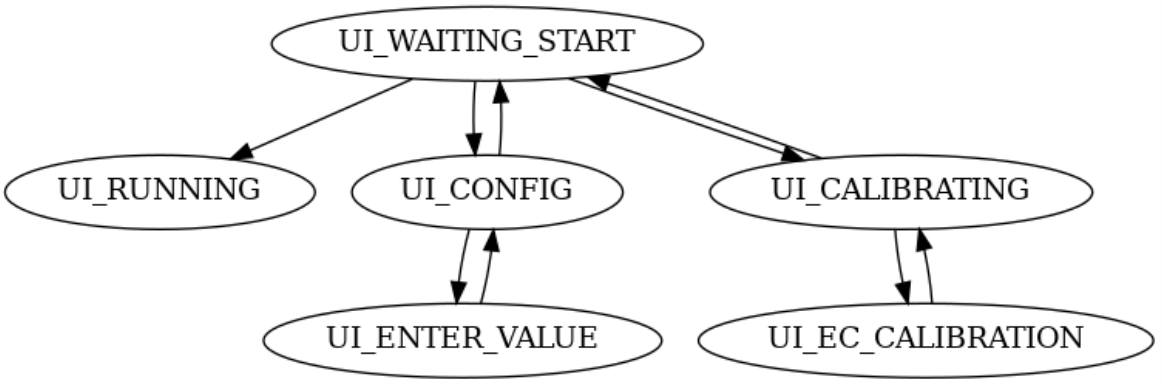
Arduino järjestelmän ohjelmointikielenä käytetään C++ johdannaista kieltä. Ohjelmointi tehtiin Arduinon ohjelmistolla. Valmisantureille löytyy valmiita kirjastoja, joita käytettiin hyväksi. Valmiskirjastojen ongelman kuitenkin on, että valmiiseen ohjelmaan liittyy usein paljon muutakin, kokonainen järjestelmä, jolloin valmiskoodia jouduttiin lyhentämään ja muokkaamaan.

4.3.1 Tilakone

Prosessi logiikaksi valittiin ”tilakone” (state machine) tyylinen ohjelmointitapa. Koska järjestelmästä tuli melko monipuolinen, antureiden ja laitteiden määrän vuoksi, oli helpompi hahmotella toimintaa tilakone logiikan kautta. Tilakone mahdollistaa prosessin seurattavuuden. Ohjelmointilogiikkaa mutkisti prosessin ja näytön eri tehtävät. Mittaus- ja säätöprosessit toimivat koko ajan ja näytöllä näytetään vain osa mittauksen antamista arvoista ja tietoa prosessin tapahtumista. Toisaalta näytöllä piti pystyä tekemään halutun pH arvon asetus ja pH toleranssin valinta, tarvittiin tilakoneeseen myös oma prosessi näiden asettamiseksi.

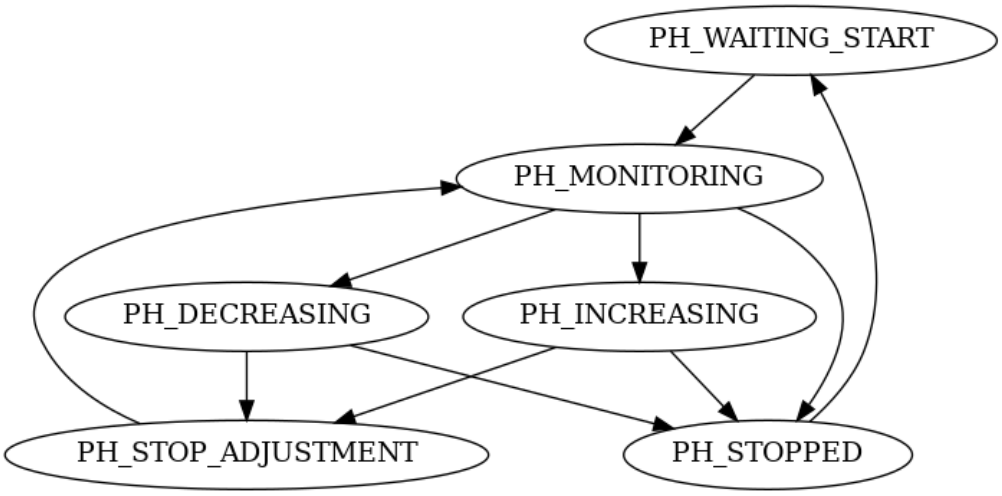
Käyttöliittymä (UI, user interface) siirtyy käynnistettäessä automaattisesti ”UI Waiting Start”-tilaan jossa käyttäjällä on vaihtoehtona siirtyä käyttöön ”UI Running”, muuttaa haluttuja pH raja-arvoja valikossa ”UI Config” tai siirtyä sähkönjohtavuusanturin kalibrointiin ”UI Calibrating”. Kaikista näistä tiloista pääse takaisin alkutilaan. (Kuva 18).

Kuva 16. Käyttöliittymän tilakone.



PH mittauksessa ja säädössä ohjelmassa toistetaan samaa tilakone logiikkaa. Alkutilanteessa ”PH Waiting Start” laitteisto odottaa mittaukseen siirtymistä. Mittaus tapahtuu ”PH Monitoring” tilassa. Mittaustuloksen perusteella mikro-ohjain voi antaa käskyt pH:n lisäämiseksi ”PH Increasing” tai vähentämiseksi ”PH Degreasing”, jolloin määrätyt releet käynnistävät ulkoiset pumput. PH:n säädöistä siirrytään ”PH Stop Adjustment” tilan kautta takaisin ”PH Monitoring” -tilaan. Vikatilassa, kun esim. nestepinnankorkeuden anturi havaitsee nestepinnan nousun, siirtyy pH mittaus ja säätöprosessi ”PH Stopped” -tilaan, josta vasta käyttäjän kuittauksella ja hyväksynnällä prosessi käynnistyy uudelleen. (Kuva 19).

Kuva 17. PH mittauksen ja säädön tilakone.



4.3.2 Halutun pH arvon ja toleranssin asettaminen

Valittiin mahdollisuudeksi asettaa haluttu pH arvo ja tälle säätötoleranssi laitteen käyttäjän toimesta. Parhaaksi vaihtoehdoksi valikoitui ohjelmoinnilla tehtävä säätö, kun potentiometrien tai vääntimien käyttö nähtiin hankalana. Järjestelmän käynnistyessä laitteen käyttäjällä on mahdollisuus valita haluttu pH arvo ja valita myös toleranssi, joka sallitaan ilman, että säätöä tapahtuu.

Alkalimetallit voivat häiritä mittausta antaen liian pieniä pH:n arvoja, jos pH- arvo on suurempi kuin 10. Häiriö voidaan poistaa käyttämällä elektrodia, jonka natriumvirhe on vähäinen.(Opetushallitus, 2021.-a).

Lämpötila vaikuttaa pH-mittaukseen kahdella tavalla:
- elektrodipotentiaali on lämpötila riippuvainen
- dissosioituminen näytteessä on lämpötila riippuvainen
(Opetushallitus, 2021.-a).

Mikäli mahdollista pH-arvo mitataan lämpötilassa 25±2°C

Lämpötila °C	pH-arvo 4,01	pH-arvo 6,87	pH-arvo 9,18
15	4,00	6,90	9,28
20	4,00	6,88	9,23
25	4,01	6,87	9,18

(Opetushallitus, 2021.-a).

4.4 Valmis järjestelmä

Järjestelmän ulkoasusta tuli prototyyppimainen ja rujon yksinkertainen. Järjestelmän toiminta on suoraviivaista.

Järjestelmän materiaalikustannukset ovat arviolta 170 €. Lisäksi postituskuluja kertyi arviolta 60€ arvosta. Työtunteja järjestelmän ohjelmointiin ja rakentamiseen kului arviolta 130 tuntia. Suuren työmäärän vuoksi laitteisto ei ole hinnaltaan kilpailukykyien tehdasvalmisteisten järjestelmien kanssa. (Taulukko 2).

Taulukko 2. Järjestelmän kustannuserittely.

Kustannukset		**2021**
Mikro-ohjain Arduino Uno kopio		6 €
Liitos-shield		16 €
20x4 LCD näyttö I2C		16 €
PH anturi ja anturimuunnin		8 €
Redox anturi ja anturimuunnin		35 €
Sähkönjohtavuusanturi ja anturimuunnin		35 €
Lämpötila-anturi ja signaalivahvistin		8 €
Nestekorkeusanturi		5 €
Releet 2 kpl		7 €
Painikkeet		8 €
Kotelo		22 €
Ohjelmointityö	100 h	
Rakennustyö	30 h	
Postituskulut		60 €
	Yhteensä	226 €

4.5 Järjestelmän toiminta ja säätö

Oikein toimiviin elektrodeihin liittyy tiettyjä numeroita. Kulmakerroin ja poikkeama ovat tuttuja arvoja, joilla voit mitata elektrodisi toimintaa. Nämä arvot ovat pääteltävissä kalibroinnin aikana. Poikkeama on yksinkertaisesti mV-arvo pH 7 kalibrointiliuoksessa ja kulmakerroin mV-arvon muutos / pH-yksikkö. Monissa mittareissa näitä arvoja voi katsoa Hyvien labrakäytäntöjen -näytöltä ”Good Laboratory Practice (GLB)”. Toimivien elektrodien kulmakerroin on normaalisti 85 – 105% ideaalisesta arvosta ja poikkeaman tulisi olla ±30mV. (Pietiko, 2021).

Seeedstudio nettisivustolla ohjeistetaan Redox anturin puhdistaminen: ”Jos elektrodin platinalanka on tahriintunut rasvalla, se voidaan puhdistaa imukykyisellä puuvillalla ja asetonilla tai alkoholilla. Ja jos se kastuu liukenemattomaan epäorgaaniseen aineeseen, 30-50% suolahappo toimii myös. Lisäksi se voidaan puhdistaa myös vanhemmalla wc -paperilla kerran päivässä.” (Seedstudio, 2021).

4.6 Antureiden kalibrointisuunnitelma

Antureiden mittaukset tarkistetaan kuukausittain. Antureiden mittausarvoja verrataan referenssinesteisiin, joiden tarkat arvot tunnetaan tai vertaamalla tulosta toiseen anturin tulokseen. Mikäli anturia ei saada kalibroitua tai mittaustulokset vaihtelevat jatkuvassa mittauksessa ylös alas, tulee anturi vaihtaa.

4.6.1 PH Anturin kalibrointi

Seuraavat näytteet voivat muuttaa elektrodin pH-herkkyyttä kerrostumalla elektrodin pinnalle:
- öljy ja rasva
- proteiinit
- veteen niukasti liukenevat yhdisteet
(Opetushallitus, 2021.-a).

Elektrodin likaantumisen huomaa siitä, että mittausherkkyys vähenee ja pH-mittari tarvitsee pitemmän ajan pysyvän arvon osoittamiseen. Likaantuneet elektrodit puhdistetaan käyttöohjeen mukaan. (Opetushallitus, 2021.-a).

Anturin mittauspää puhdistetaan tähän tarkoitukseen hankitulla liuoksella. Puhdistuksen jälkeen anturi upotetaan referenssiliuokseen. Referenssiliuosten pH arvot tunnetaan tarkasti. Verrokkeina käytetään esim. liuoksia, joiden pH on 1,413; 7,01 sekä 12,88. Näitä nesteitä on helposti saatavissa esimerkiksi akvaariotarvike- tai kukkakaupoista. Tuloksen säätö tapahtuu anturimoduulin potentiometrin ruuvia säätämällä. Ruuvia käännetään, kunnes mittaustulos osoittaa oikeaa.

Tarkastuksen jälkeen pH anturi puhdistetaan tarkoitukseen tehdyllä liuoksessa ja käsitellään pH anturille tarkoitetulla KCl- liuoksella. Näin saavutetaan anturin pidempi käyttöikä ja tarkemmat mittaustulokset.

4.6.2 Redox anturin kalibrointi

Redox anturin kalibrointi tapahtuu vertaamalla anturin mittaustulosta tunnetusta referenssiliuoksesta. Referenssiliuoksena voi toimia esim. 146 mV, 220 mV sekä 468 mV referenssineste.

4.6.3 Sähkönjohtavuusanturin kalibrointi

Sähkönjohtavuusanturin kalibrointi suoritetaan mikro-ohjaimella. Sähkönjohtavuusanturin ohjelmistosta löytyy valmiskirjaston koodi, jossa kalibrointi on mukana. Kalibrointiin päästään suoraan ohjainpaneelin valikoista.

Kalibrointi toteutetaan laittamalla anturi kalibrointinesteeseen 1400µS/cm tai 700 µS/cm. Ohjelma päättelee, kumpi neste on kyseessä ja tekee automaattisen kalibroinnin tähän arvoon. Tämän jälkeen anturi on valmis käytettäväksi ja kalibrointi voidaan vielä tarkistaa toisella nesteellä ja vertaamalla mittaustulosta tarkistusnesteen arvoon. (Taulukko 3).

Taulukko 3. Lämpötilan vaikutus sähkönjohtavuuteen (VP Lux, 2021).

Temp. (°C)	µS/cm
15	1147
20	1278
21	1305
22	1332
23	1359
24	1386
25	1413
26	1440
27	1467
28	1494
29	1521
30	1548

4.6.4 Muiden antureiden tarkistaminen

Nesteen lämpötila-anturi on esikalibroitu eikä kalibrointia voida tehdä ilma, että koodiin tehdään muutoskerroin. Lämpötilan oikeellisuus tehdään vertaamalla mittaustulosta toisen lämpömittarin tulokseen. Mikäli eroa löytyy yli 1˚C tarkistetaan tulos vielä uudestaan ja tarvittaessa kolmannella laitteella. Mikäli mittausvirhe on ilmeinen, tehdään koodiin muutoskerroin.

Nestekorkeusanturin toiminta voidaan varmistaa asettamalla käsi anturin eteen. Mikäli käden ollessa anturin edessä prosessi seisahtuu, on anturi kunnossa. Viallinen anturi vaihdetaan uuteen.

5 Kehitys

Laitteistoa voitaisiin tulevaisuudessa kehittää langattoman tarkkailun mahdollistavaksi siten, että mittaustiedot voitaisiin lukea etänä. Teknisesti toteutus voitaisiin tehdä vaihtamalla nykyinen mikro-ohjain langattoman tiedonsiirron (Wi-Fi) sisältäväksi mikro-ohjaimeksi. Arduino mikro-ohjaimiin liitettävät Wi-Fi shield'it rajautuvat tässä tapauksessa pois, sillä mikro-ohjaimeen on jo nyt liitetty ruuviliitos shield. Langaton seuranta voisi vähentää henkilön paikalla olon määrää. Langattoman käyttöjärjestelmän lisääminen on kuitenkin melko suuri työ, sillä myös datan vastaanottava ohjelmisto tulee rakentaa. (Kuva 20).

Kuva 18. Mikro-ohjain Wi-Fi lähettimellä (Amazon, 2021)

Yksi suunta kehittää järjestelmää on lisätä toimintoja. Sähkönjohtavuusanturin arvoilla voitaisiin esim. sytyttää huomiovalo. Sähkönjohtavuusanturi voisi myös käynnistää talteenottoprosessin, joka toimisi ehkä omana automaatiojärjestelmänä.

Laitteen seuraava kehitysaskel olisi todennäköisesti antureiden vaihtaminen teollisuusympäristöön suunnitelluiksi antureiksi. Mikäli antureiden kalibrointia saataisiin vähennettyä, liuotusprosessi nopeutuisi ja helpottuisi. Teollisuudessa käytettävien antureiden haasteena on hintaluokkien ja varmasti myös antureiden laadun vertailu. Mikäli antureita vaihdetaan, tulee selvitettäväksi myös ohjainlaitteen ja koko laitteen vaihtaminen tehdasvalmisteiseksi. Teolliseen prosessiin valmistetun laitteiston etuna voisi olla antureiden jatkuvan upottamisen mahdollisuus, takuu sekä käyttötuki. (Kuva 21).

Kuva 19. Prominent Dulcontrol (Prominent, 2021).

Kun prosessi saadaan testattua toimivaksi, voidaan mahdollisesti korvata Arduino käyttöympäristö teollisilla laitteilla ja pienemmillä logiikkaohjaimilla, jotka olisivat mahdollisesti yhteensopivia suurempien järjestelmäkokonaisuuksien kanssa. Esimerkiksi ABB AWT420 kaksi kanavainen vastaanotin voisi olla hyvä alusta kehittämiselle. (Kuva 22). AWT 420 hinta ilman antureita on arviolta 1230 USD. (Quicktime online, 2021).

Kuva 20. Kaksikanavainen vastaanotin (ABB, 2021).

Mikäli laitteistosta halutaan edelleen edullinen, kevyt ylläpitää ja tehdasvalmisteinen, niin yhtenä vaihtoehtona on ottaa käyttöön pelkästään pH mittaus- ja säätöjärjestelmä, joita on saatavilla kuluttajille tarkoitetuista kasviviljely- tai uima-altaita tarjoavista kaupoista.

Tällainen valmis, melko kompakti järjestelmä löytyy esim. Pavunvarsi myymäläketjulta hintaan 269 € (sis. alv). Kyseisen kaltaisessa laitteessa on tarvittavat toiminnot pH säädön ylläpitämiseksi. Heikkoutena on, että tarkkaillaan ja säädetään vain yhtä parametriä (pH) ja muut mittaukset tulee toteuttaa muilla laitteilla. Lisäksi esim. Milwaukeen MC720 mittaria tulee kalibroida ja anturipäätä vaihtaa tarpeen mukaan (á 79 €). Laitetta ei myöskään välttämättä voi käyttää pitkään upotettuna tai ainakin laitteen myyjän mukaan anturin elinikä saattaa lyhentyä. (Kuva 23).

Kuva 21. PH mittaus ja säätöjärjestelmä (Pavunvarsi, 2021).

6 Yhteenveto

Tehtävä vaikutti mielenkiintoiselta ja monipuoliselta, jonka takia innostuin aiheesta ja lähdin tehtävää työstämään. Tehtävä osoittautui kuitenkin haastavaksi ja aikaa vieväksi erityisesti siksi, etten aiemmin ollut rakentanut automaatioon liittyviä laitteistoja tai ohjelmoinut mikro-ohjainta näin laajasti.

Alkumäärittelyt, prosessin toiminta ja laitteiston toiminta osana prosessia oli helppo ja nopea hahmottaa. Suunnittelu sujui ongelmitta. Suunnittelun vajavaisuus ilmeni laitteiston rakentamiseen ryhdyttäessä. Koska itselläni ei ollut aiempaa kokemusta laitteiston rakentamisesta erillisistä komponenteista, huomasin rakennusvaiheessa tarvitsevani lisää komponentteja, esim. anturivahvistimia. Toteutusvaiheessa myös tarvittavien toimintojen vaatimat liitinmäärät mikro-ohjaimelta tarkentuivat. Alkusuunnitelmassa en ymmärtänyt ottaa kaikkia pienimpiä seikkoja huomioon, joka johtui kokemattomuudestani. Esimerkiksi alustavasta ajattelin käyttäväni mikro-ohjaimena Arduino Unoa, joka toteutusvaiheessa ilmeni riittämättömäksi ja myöhemmin toteutusvaiheessa taas riittäväksi. Tähän tilanteeseen johti rakentamisen aikana tehdyt muutokset näyttöjen määrään, painikkeisiin ja antureihin.

Tehtävän aikana opin ennakkosuunnittelun ja aihealueen osaamisen tärkeydestä. Pääsin rakentamaan prototyyppiä alusta alkaen ja sain käsityksen projektin kokonaisesta läpiviennistä. Ohjelmointiin käytin arviolta sata tuntia aikaa, vaikka antureille oli käytettävissä valmista koodia. Mikäli projektin järkevyyttä ajatellaan taloudellisesti, tulee tehdasvalmisteinen järjestelmä varmasti edullisemmaksi. Valmis laite kuitenkin toimii ja voidaan pienillä muutoksilla ottaa käyttöön tuotannossa. Samoin projektin kautta olen oppinut paljon uutta, joten arvioin tehtävän onnistuneeksi.

Lähteet

ABB. (7.4.2021). *AWT 420 Dual input transmitter.* https://new.abb.com/products/measurement-products/analytical/continuous-water-analysis/transmitters/awt420-universal-4-wire-dual-input-transmitter

AHaa elektroniikka. (7.4.2021). *LCD näyttö*. https://www.elektroniikkaosat.com/c-46/p-4020/LCD-naytto-4x20-merkkia-i-useita-vaihtoehtoja-i.html

Amazon. (7.4.2021). *Arduino WiFi shield.* https://www.amazon.com/HiLetgo-ATmega328P-ESP8266-NodeMCU-USB-TTL/dp/B0834JQ5Y5/ref=sr_1_9?dchild=1&keywords=arduino+wifi&qid=1630221617&sr=8-9

Circuits DIY. (7.4.2021). *PT 100 sensor module.* https://circuits-diy.com/pt100-rtd-sensor-module/

DF Robot. (7.4.2021*). Prote screwshield.* https://www.dfrobot.com/product-468.html

DiIY More. (7.4.2021.-a). *Aquarium hydrophonic ph controller electrode.* https://www.diymore.cc/collections/sensor-module/products/aquarium-hydroponic-ph-controller-electrode-probe-bnc-connector-sensor-ph-meter

DIY More. (7.4.2021.-b). *PH4502C Liquid pH value detection detect sensor.* https://www.diymore.cc/products/diymore-liquid-ph-value-detection-detect-sensor-module-monitoring-control-for-arduino-m

DIY More. (7.4.2021.-c). *PT100 sensor amplifier module.* https://www.diymore.cc/products/max31865-pt100-to-pt1000-rtd-to-digital-converter-board-temperature-thermocouple-sensor-amplifier-module-3-3v-5v-for-arduino

IC Station. (7.4.2021). *Water level detector.* http://www.icstation.com/contact-liquid-level-sensor-ip67-waterproof-output-water-level-detector-p-12292.html

Ihmevekotin. (7.4.2021). *2x5V relekortti.* https://ihmevekotin.fi/kytkimet-ja-releet/566-2x5v-relekortti-kodin-saehkoelaitteiden-ohjaukseen.html

Oulun yliopisto (2011). *Johtokyvyn mittaus ja säätö.* https://www.oulu.fi/sites/default/files/content/johtokyky_v1.6_0.pdf

Opetushallitus. (25.4.2021.-a). *PH vedestä.* http://www03.edu.fi/oppimateriaalit/laboratorio/ymparistoanalyysit_ph_vedesta.html

Opetushallitus (11.4.2021.-b). *Veden sähkönjohtavuus.* http://www03.edu.fi/oppimateriaalit/laboratorio/ymparistoanalyysit_veden_sahkonjohtavuus.html

Pavunvarsi. (7.4.2021). *Milwaukee MC720 pH kontrolleri.* https://pavunvarsi.fi/ph-ja-ec-arvon-seuranta/ph-mittarit/milwaukee-mc720-ph-kontrolleri.html

Pietiko (7.4.2021). *PH mittaus.* https://www.pietiko.fi/mittaustietoa/ph-mittaus/

Prominent. (7.4.2021). *Dulcotrol.* https://www.prominent.fi/fi/Tuotteet/Tuotteet/Mittaus-ja-s%C3%A4%C3%A4t%C3%B6tekniikka-anturitekniikka/Paneeliasennettavat-mittaus-ja-s%C3%A4%C3%A4t%C3%B6j%C3%A4rjestelm%C3%A4t/p-dulcotrol-waste-water.html

Puuilo. (7.4.2021). *Asennuskotelo.* https://www.puuilo.fi/famatel-asennuskotelo-310x240x125-ip55

Quicktime online (2.10.2021). *ABB dual channel transmitter.* https://www.quicktimeonline.com/awt420a1a1c2y0y0e5-abb-dual-channel-transmitter,

Radiopaedia. (24.11.2021). *Silver recovery.* https://radiopaedia.org/articles/silver-recovery

Seeedstudio. (7.4.2021.-a). *Grove ED sensor kit.* https://www.seeedstudio.com/Grove-EC-Sensor-Kit-DJS-1C-Black-p-4576.html

Seeedstudio. (7.4.2021.-b). *Grove ORP sensor kit.* https://www.seeedstudio.com/Grove-ORP-Sensor-Kit-501Z-p-4575.html

Tinkersphere. (7.4.2021). *Uno R3.* https://tinkersphere.com/boards/953-uno-r3-arduino-compatible.html

VP Lux. (11.4.2021). *Kalibraationeste*. https://www.vplux.fi/fi/product/aquamaster-kalibraationeste-1413-100ml/13475

Wikipedia. (7.4.2021.-a). *PH mittari.* https://fi.wikipedia.org/w/index.php?title=PH-mittari&oldid=19675414

Wikipedia. (7.4.2021.-b). *Redoxpotentiaali*. https://fi.wikipedia.org/w/index.php?title=Redoxpotentiaali&oldid=18599266

Liite 1: Mikro-ohjaimen koodi

```
#include <EEPROM.h>
#include <Adafruit_SPIDevice.h>
#include <Adafruit_MAX31865.h>
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 20, 4);
#if defined(ARDUINO) && ARDUINO >= 100
#define printByte(args) write(args);
#else
#define printByte(args) print(args, BYTE);
#endif

// Custom characters
uint8_t plusminus[8] = {0x4, 0x04, 0x1f, 0x04, 0x04, 0x0, 0x1f};
uint8_t uparrow[8] = {0x4, 0xe, 0x15, 0x0, 0x4, 0x0, 0x4};
uint8_t downarrow[8] = {0x4, 0x0, 0x4, 0x0, 0x15, 0xe, 0x4};

//PINS
#define BUTTON_PLUS 2
#define BUTTON_MINUS 3
#define BUTTON_ENTER 4
#define BUTTON_STOP 7
#define PH_PLUS_RELAY 9
#define PH_MINUS_RELAY 8
#define orpPin A3

#define ON 1
#define OFF 0

struct eeprom_config
{
  int sentinel;
  float target_ph;
  float tolerance;
  float kvalue_low;
  float kvalue_high;
} configuration;

// Utility functions
void set_relay(int relay, int mode)
{
  if (ON == mode)
  {
    digitalWrite(relay, HIGH);
  }
  else
  {
    digitalWrite(relay, LOW);
  }
}

// Setup measurements to "uninitialized"
float target_ph;
float current_ph;
float current_ec;
float current_temp;
double current_orp;
boolean liquid_level;
```

```
// Read liquid level
#define LL_PIN 5
boolean read_liquid_level()
{
  return digitalRead(LL_PIN);
}

Adafruit_MAX31865 thermo = Adafruit_MAX31865(10, 11, 12, 13);
#define RREF 430.0     //Preset value for PT100 sensor
#define RNOMINAL 100.0 //Preset value for PT100 sensor

// Temperature reading
float read_temp_sensor()
{
  static unsigned long last_millis = 0;
  static float temperature = 0.0;
  unsigned long current_millis = millis();

  if ((current_millis - last_millis) >= 1000U)
  {
    uint16_t rtd = thermo.readRTD();
    uint8_t fault;

    // Time to update the temperature
    last_millis = current_millis;
    if (fault)
    { // in case of fault return last good read
      thermo.clearFault();
      return temperature;
    }
    temperature = thermo.temperature(RNOMINAL, RREF);
  }

  return temperature;
}

// ORP reading
#define ArrayLength 40
#define VOLTAGE 5
#define OFFSET 0

double avergearray(int *arr, int number)
{
  int i;
  int max, min;
  double avg;
  long amount = 0;
  if (number <= 0)
  {
    return 0;
  }
  if (number < 5)
  { //less than 5, calculated directly statistics
    for (i = 0; i < number; i++)
    {
      amount += arr[i];
    }
    avg = amount / number;
    return avg;
  }
  else
```

```
  {
    if (arr[0] < arr[1])
    {
      min = arr[0];
      max = arr[1];
    }
    else
    {
      min = arr[1];
      max = arr[0];
    }
    for (i = 2; i < number; i++)
    {
      if (arr[i] < min)
      {
        amount += min; //arr<min
        min = arr[i];
      }
      else
      {
        if (arr[i] > max)
        {
          amount += max; //arr>max
          max = arr[i];
        }
        else
        {
          amount += arr[i]; //min<=arr<=max
        }
      } //if
    }   //for
    avg = (double)amount / (number - 2);
  } //if
  return avg;
}

double read_orp_sensor()
{
  static int orpArray[ArrayLength];
  static int orpArrayIndex = 0;
  static unsigned long orpTimer = millis();
  unsigned long current_time = millis();
  double value = current_orp;

  if ((current_time - orpTimer) >= 20) //read an analog value every 20ms
  {
    orpTimer = current_time;
    orpArray[orpArrayIndex++] = analogRead(orpPin);
    if (orpArrayIndex == ArrayLength)
    {
      orpArrayIndex = 0;
    }
    value = ((30 * (double)VOLTAGE * 1000) - (75 * avergearray(orpArray, ArrayLength) * VOLTAGE *
1000 / 1024)) / 75 - OFFSET; //convert the analog value to orp according the circuit
  }
  return value;
}

// EC reading, code adapted from the DFRobot library as the calibration step
// would have required serial port access to be used properly and the calibration
// value storage was hardcoded to write to fixed EEPROM address that was conflicting
// with our configuration
```

```
#define EC_PIN A2
#define RES2 820.0
#define ECREF 200.0

float ec_kvalue = 1.0;
float ec_kvaluelow = 1.0;
float ec_kvaluehigh = 1.0;

float read_ec_voltage()
{
  return analogRead(EC_PIN) / 1024.0 * 5000;
}

// reads new value every seconds, if called more frequently returns the last value read
float read_ec_sensor()
{
  static unsigned long timepoint = millis();
  static float ecValue;

  if (millis() - timepoint > 1000U) //time interval: 1s
  {
    float voltage = read_ec_voltage();
    float _raw_value = 1000 * voltage / RES2 / ECREF;
    float _tmp_value = _raw_value * ec_kvalue;

    timepoint = millis();

    //automatic shift process
    //First Range:(0,2); Second Range:(2,20)
    if (_tmp_value > 2.5)
    {
      ec_kvalue = ec_kvaluehigh;
    }
    else if (_tmp_value < 2.0)
    {
      ec_kvalue = ec_kvaluelow;
    }
    ecValue = _raw_value * ec_kvalue;                         //calculate the EC value after automatic shift
    ecValue = ecValue / (1.0 + 0.0185 * (current_temp - 25.0)); //temperature compensation
  }

  return ecValue;
}

bool calibrate_ec_sensor()
{
  float temperature = current_temp;
  float voltage = read_ec_voltage();
  float _raw = 1000 * voltage / RES2 / ECREF;
  float solution = 0.0;
  float *kvaluep = NULL;

  // determine buffer solution, apply temperature compensation
  // and determine if the compensation is for high or low kvalue
  if ((_raw > 0.9) && (_raw < 1.99))
  {
    solution = 1.413 * (1.0 + 0.0185 * (temperature - 25.0)); //temperature compensation
    kvaluep = &configuration.kvalue_low;
  }
  else if ((_raw > 9.0) && (_raw < 16.8))
  {
    solution = 12.88 * (1.0 + 0.0185 * (temperature - 25.0)); //temperature compensation
```

```
    kvaluep = &configuration.kvalue_low;
  }
  else
  {
    return false;
  }
  *kvaluep = RES2 * ECREF * solution / 1000.0 / voltage;
  return true;
}

// PH FSM
#define PH_WAITING_START 0
#define PH_MONITORING 1
#define PH_INCREASING 2
#define PH_DECREASING 3
#define PH_STOP_ADJUSTMENT 4
#define PH_STOPPED 5

int ph_state = PH_WAITING_START;

bool invalid_ph()
{
  if ((current_ph < 2.0) || (current_ph > 13.0))
  {
    return true;
  }
  return false;
}

// Constants controlling ph measurement reading
float read_ph_sensor()
{
#define PH_SENSOR A0
#define PH_SAMPLES 10
#define ADC_RESOLUTION 1024.0
  static int measurings = 0;
  static int last_read = millis();
  static int samples = 0;
  static float ph_value = -1.0;

  int current_millis = millis();

  if (!samples || ((current_millis - last_read) > 10U))
  {
    measurings += analogRead(PH_SENSOR);
    samples++;
    last_read = current_millis;
    if (samples >= PH_SAMPLES)
    {
      float voltage = 5 / ADC_RESOLUTION * measurings / samples;
      ph_value = 7 + ((2.5 - voltage) / 0.18);
      samples = 0;
      measurings = 0;
    }
  }
  return ph_value;
}

void ph_state_next(int state)
{
  if (ph_state == PH_STOPPED)
  {
```

```
    if ((state == PH_WAITING_START) || (state == PH_MONITORING))
    {
      ph_state = state;
    }
    return;
  }
  ph_state = state;
}

// UI FSM
#define UI_WAITING_START 0
#define UI_RUNNING 1
#define UI_CONFIG 2
#define UI_ENTER_VALUE 3
#define UI_CALIBRATING 4
#define UI_EC_CALIBRATION 5

int ui_state = UI_WAITING_START;

void ui_state_next(int state)
{
  ui_state = state;
}

float ph_upper_limit = 0.0;
float ph_lower_limit = 14.0;

void ph_state_exec()
{
  if (ph_state == PH_WAITING_START)
  {
    return;
  }

  // If current ph does not make sense
  // do not make any adjustments
  // (ph display should show error)
  if (invalid_ph())
  {
    return;
  }

  if (ph_state == PH_MONITORING)
  {
    if (current_ph < ph_lower_limit)
      ph_state_next(PH_INCREASING);
    if (current_ph > ph_upper_limit)
      ph_state_next(PH_DECREASING);
    return;
  }
  if (ph_state == PH_INCREASING)
  {
    set_relay(PH_PLUS_RELAY, ON);
    if (current_ph >= target_ph)
      ph_state_next(PH_STOP_ADJUSTMENT);
    return;
  }
  if (ph_state == PH_DECREASING)
  {
    set_relay(PH_MINUS_RELAY, ON);
    if (current_ph <= target_ph)
      ph_state_next(PH_STOP_ADJUSTMENT);
```

```
    return;
  }
  if (ph_state == PH_STOP_ADJUSTMENT)
  {
    set_relay(PH_MINUS_RELAY, OFF);
    set_relay(PH_PLUS_RELAY, OFF);
    ph_state_next(PH_MONITORING);
    return;
  }
  if (ph_state == PH_STOPPED)
  {
    set_relay(PH_MINUS_RELAY, OFF);
    set_relay(PH_PLUS_RELAY, OFF);
    return;
  }
}

void show_measurements()
{
  static int last_show = millis() - 1001U;
  int current_millis = millis();

  if ((current_millis - last_show) > 1000U)
  { // LDC show values code goes here eventually
    lcd.setCursor(0, 0);
    lcd.print("pH ");
    if (invalid_ph())
    {
      lcd.print("Err");
    }
    else
    {
      lcd.print(current_ph);
    }
    lcd.print("/");
    lcd.print(target_ph);
    lcd.printByte(7);
    lcd.print(ph_upper_limit - target_ph);
    lcd.setCursor(19, 0);
    if (invalid_ph())
    {
      lcd.print("!");
    }
    else
    {
      if (ph_state == PH_INCREASING)
      {
        lcd.printByte(8);
      }
      else if (ph_state == PH_DECREASING)
      {
        lcd.printByte(9);
      }
      else if (ph_state != PH_MONITORING)
      {
        lcd.printByte('W');
      }
      else
      {
        lcd.printByte('=');
      }
    }
```

```
    lcd.setCursor(0, 1);
    lcd.print("Temp: ");
    lcd.print(current_temp);
    lcd.setCursor(0, 2);
    lcd.print("ORP: ");
    lcd.print(current_orp);
    lcd.setCursor(12, 2);
    lcd.print("mV");
    lcd.setCursor(0, 3);
    lcd.print("EC: ");
    lcd.print(current_ec);
    lcd.setCursor(12, 3);
    lcd.print("ms/cm");

    last_show = current_millis;
  }
}

boolean button_pressed(int button)
{
  int i = digitalRead(button);
  return digitalRead(button) == LOW;
}

// return button pressed (UI)
// button must be released to reregister reading it
int get_button()
{
  static int last_button = 0;

  if (button_pressed(BUTTON_PLUS))
  {
    if (!last_button)
    {
      last_button = BUTTON_PLUS;
      return BUTTON_PLUS;
    }
  }
  else if (button_pressed(BUTTON_MINUS))
  {
    if (!last_button)
    {
      last_button = BUTTON_MINUS;
      return BUTTON_MINUS;
    }
  }
  else if (button_pressed(BUTTON_ENTER))
  {
    if (!last_button)
    {
      last_button = BUTTON_ENTER;
      return BUTTON_ENTER;
    }
  }
  else
  {
    last_button = 0;
  }
  return 0;
}

void print_if_entering(const char *s, int row, int current_row)
```

```
{
  if ((ui_state == UI_ENTER_VALUE) && (row == current_row))
  {
    lcd.print(s);
  }
  else
  {
    lcd.print(" ");
  }
}

void stateful_print_value(const char *label, float value, int row, int current_row)
{
  lcd.setCursor(0, row);
  lcd.print(label);
  print_if_entering("[", row, current_row);
  lcd.print(value);
  print_if_entering("]", row, current_row);
}

void configure_values()
{
  static bool cleared = false;
  static int row = 0;
  int button;

  if (!cleared)
  {
    lcd.clear();
    cleared = true;
  }
  if (button = get_button())
  {
    if (button == BUTTON_MINUS)
    {
      if (ui_state == UI_ENTER_VALUE)
      {
        if (row == 0)
        {
          configuration.target_ph = configuration.target_ph - 0.5;
        }
        else if (row == 1)
        {
          configuration.tolerance = configuration.tolerance - 0.1;
        }
      }
      else if (row < 2)
      {
        lcd.setCursor(19, row);
        lcd.printByte(' ');
        row++;
      }
    }
    if (button == BUTTON_PLUS)
    {
      if (ui_state == UI_ENTER_VALUE)
      {
        if (row == 0)
        {
          configuration.target_ph = configuration.target_ph + 0.5;
        }
        else if (row == 1)
```

```
        {
          configuration.tolerance = configuration.tolerance + 0.1;
        }
      }
      else if (row)
      {
        lcd.setCursor(19, row);
        lcd.printByte(' ');
        row--;
      }
    }
    if (button == BUTTON_ENTER)
    {
      lcd.setCursor(19, row);
      lcd.printByte(' ');
      if (row == 2)
      {
        ui_state_next(UI_WAITING_START);
        cleared = false;
        write_config();
        read_config();
        return;
      }
      if (ui_state == UI_ENTER_VALUE)
      {
        ui_state_next(UI_CONFIG);
      }
      else
      {
        ui_state_next(UI_ENTER_VALUE);
      }
    }
  }
  stateful_print_value("PH target:", configuration.target_ph, 0, row);
  stateful_print_value("tolerance:", configuration.tolerance, 1, row);
  lcd.setCursor(0, 2);
  lcd.print("[BACK]");
  lcd.setCursor(19, row);
  lcd.printByte('<');
}

int start_screen()
{
  static bool cleared = false;
  static int row = 1;
  int button;

  if (!cleared)
  {
    lcd.clear();
    cleared = true;
  }
  lcd.setCursor(0, 0);
  if (ph_state == PH_STOPPED)
  {
    lcd.print("Stopped");
  }
  else
  {
    lcd.print("Waiting start");
  }
  lcd.setCursor(0, 1);
```

```
  lcd.print("Start");
  lcd.setCursor(0, 2);
  lcd.print("Setup");
  lcd.setCursor(0, 3);
  lcd.print("Calibrate");
  if (button = get_button())
  {
    if (button == BUTTON_PLUS)
    {
      lcd.setCursor(19, row);
      lcd.printByte(' ');
      row--;
      if (row < 1)
      {
        row = 3;
      }
    }

    if (button == BUTTON_MINUS)
    {
      lcd.setCursor(19, row);
      lcd.printByte(' ');
      row++;
      if (row > 3)
      {
        row = 1;
      }
    }

    if (button == BUTTON_ENTER)
    {
      lcd.setCursor(19, row);
      lcd.printByte(' ');
      lcd.clear();
      cleared = false;

      if (row == 1)
      {
        return UI_RUNNING;
      }
      if (row == 2)
      {
        return UI_CONFIG;
      }
      /* row == 3 */
      return UI_CALIBRATING;
    }
  }
  lcd.setCursor(19, row);
  lcd.printByte('<');
  return UI_WAITING_START;
}

int calibration_screen()
{
  static bool cleared = false;
  int button;
  static int last_show = millis() - 1001U;
  int current_millis = millis();
  static int row = 3;

  if (!cleared)
```

```
  {
    lcd.clear();
    cleared = true;
  }

  if ((current_millis - last_show) > 1000U)
  {
    lcd.setCursor(0, 0);
    lcd.print("pH ");
    lcd.print(current_ph);
    lcd.setCursor(0, 1);
    lcd.print("Temp: ");
    lcd.print(current_temp);
    lcd.setCursor(0, 2);
    lcd.print("CALIBRATE EC");
    lcd.setCursor(0, 3);
    lcd.print("END CALIBRATION");
    lcd.setCursor(19, row);
    lcd.printByte('<');
    last_show = current_millis;
  }

  button = get_button();
  if ((button == BUTTON_PLUS) || (button == BUTTON_MINUS))
  {
    lcd.setCursor(19, row);
    lcd.printByte(' ');
    if (row == 2)
    {
      row = 3;
    }
    else
    {
      row = 2;
    }
  }
  if (button == BUTTON_ENTER)
  {
    cleared = false;
    if (row == 3)
    {
      return UI_WAITING_START;
    }
    return UI_EC_CALIBRATION;
  }
  return UI_CALIBRATING;
}

int ui_ec_calibration()
{
  static bool cleared = false;
  static bool good_read = false;
  static int tries = 0;
  static int last_show = millis() - 1001U;

  int button;
  int current_millis = millis();
  static int row = 3;

  if (!cleared)
  {
    lcd.clear();
```

```
  cleared = true;
}

// Calibration is done by placing the probe in the solution after
// which the calibration can be started
if ((current_millis - last_show) > 1000U)
{
  lcd.setCursor(0, 0);
  lcd.print("EC calibration");
  // Wait to start the calibration
  if (!good_read)
  {
    if (tries > 0)
    {
      lcd.setCursor(0, 1);
      lcd.print("FAILED     ");
    }
  }
  else
  {
    lcd.setCursor(0, 1);
    lcd.print("CALIBRATED ");
  }
  lcd.setCursor(0, 2);
  if (!good_read)
  {
    if (tries == 0)
    {
      lcd.print("Start");
    }
    else
    {
      lcd.print("Retry");
    }
  }
  else
  {
    lcd.print("Save ");
  }
  lcd.setCursor(0, 3);
  lcd.print("Cancel");
  lcd.setCursor(19, row);
  lcd.printByte('<');
}
button = get_button();
if ((button == BUTTON_PLUS) || (button == BUTTON_MINUS))
{
  lcd.setCursor(19, row);
  lcd.printByte(' ');
  if (row == 2)
  {
    row = 3;
  }
  else
  {
    row = 2;
  }
}
if (button == BUTTON_ENTER)
{
  if (row == 3) // Cancel
  {
```

```
      cleared = false;
      return UI_CALIBRATING;
    }
    // row == 2, start/retry or save
    if (good_read)
    {
      cleared = false;
      write_config();
      return UI_CALIBRATING;
    }
    good_read = calibrate_ec_sensor();
    tries++;
  }
  return UI_EC_CALIBRATION;
}

void ui_state_exec()
{
  if ((ui_state == UI_CONFIG) || (ui_state == UI_ENTER_VALUE))
  {
    configure_values();
    return;
  }

  if (ui_state == UI_WAITING_START)
  {
    ui_state_next(start_screen());
    if (ui_state == UI_RUNNING)
    {
      ph_state_next(PH_MONITORING);
    }
    return;
  }

  if (ui_state == UI_CALIBRATING)
  {
    ui_state_next(calibration_screen());
    return;
  }

  if (ui_state == UI_EC_CALIBRATION)
  {
    ui_state_next(ui_ec_calibration());
    return;
  }

  show_measurements();
  if (ui_state == UI_RUNNING)
  {
    if (liquid_level)
    {
      ph_state_next(PH_STOPPED);
      ui_state_next(UI_WAITING_START);
    }
    else if (ph_state == PH_STOPPED)
    {
      ph_state_next(PH_MONITORING);
    }
  }
}

#define CONFIG_VERSION 0x45
```

```
bool read_config()
{
  int i;
  byte *p;
  bool valid = true;

  for (p = (byte *)&configuration, i = 0; i < sizeof(configuration); i++, p++)
  {
    *p = EEPROM.read(i);
  }

  // check if the eeprom has valid configuration
  if ((configuration.sentinel != CONFIG_VERSION) || (configuration.target_ph <= 1.0) ||
(configuration.target_ph > 14.0))
  {
    configuration.target_ph = 7.0;
    configuration.tolerance = 1.0;
    configuration.kvalue_low = 1.0;
    configuration.kvalue_high = 1.0;
    valid = false;
  }
  // copy values from configuration
  target_ph = configuration.target_ph;
  ph_lower_limit = target_ph - configuration.tolerance;
  ph_upper_limit = target_ph + configuration.tolerance;

  return valid;
}

void write_config()
{
  int i;
  byte *p;

  // copy values from settings
  configuration.sentinel = CONFIG_VERSION;

  for (p = (byte *)&configuration, i = 0; i < sizeof(configuration); i++, p++)
  {
    EEPROM.write(i, *p);
  }
}

void setup()
{
  Serial.begin(115200); // debug console
  Serial.println("Starting up");

  //LCD setup
  Serial.println("Initialize LCD");
  lcd.init();
  lcd.backlight();
  lcd.clear();
  lcd.createChar(7, plusminus);
  lcd.createChar(8, uparrow);
  lcd.createChar(9, downarrow);

  pinMode(BUTTON_PLUS, INPUT_PULLUP);
  pinMode(BUTTON_MINUS, INPUT_PULLUP);
  pinMode(BUTTON_ENTER, INPUT_PULLUP);
  pinMode(BUTTON_STOP, INPUT);
```

```
  pinMode(PH_PLUS_RELAY, OUTPUT);
  pinMode(PH_MINUS_RELAY, OUTPUT);
  pinMode(LL_PIN, INPUT);
  pinMode(orpPin, INPUT);

  // Setup starting state
  // Turn off relays
  Serial.println("Turn off relays");
  set_relay(PH_PLUS_RELAY, OFF);
  set_relay(PH_MINUS_RELAY, OFF);

  Serial.println("Initialize sensors");
  //read initial sensor values
  thermo.begin(MAX31865_3WIRE);
  for (; current_temp < 0;)
    current_temp = read_temp_sensor();
  Serial.println("temp sensor initialized");
  for (; current_ph < 0;)
    current_ph = read_ph_sensor();
  Serial.println("ph sensor initialized");
  for (; current_ec < 0;)
    current_ec = read_ec_sensor();
  Serial.println("ec sensor initialized");
  for (; current_orp < 0;)
    current_orp = read_orp_sensor();
  Serial.println("orp sensor initialized");
  liquid_level = read_liquid_level();

  Serial.println("Initialize state machines");
  // Initialize state machines
  ph_state_next(PH_WAITING_START);
  if (read_config())
  {
    ui_state_next(UI_WAITING_START);
  }
  else
  {
    // we do not have a valid config in EEPROM
    ui_state_next(UI_CONFIG);
  }

  Serial.println("Running");
}

void loop()
{
  liquid_level = read_liquid_level();
  current_temp = read_temp_sensor();
  current_orp = read_orp_sensor();
  current_ph = read_ph_sensor();
  current_ec = read_ec_sensor();

  ui_state_exec();
  ph_state_exec();
}
```