LEVÄREAKTORIN PH-ANTUREIDEN KALIBROINTIOMINAISUUDEN TOTEUTTAMINEN TYÖPÖYTÄSOVELLUKSEEN

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Ammattikorkeakoulututkinnon opinnäytetyö

Tietojenkäsittelyn koulutusohjelma

Hämeenlinna, syksy 2017

Janina Koskela

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Tietojenkäsittelyn koulutusohjelma
Hämeenlinna

Tekijä Janina Koskela **Vuosi** 2017

Työn nimi Leväreaktorin pH-antureiden kalibrointiominaisuuden toteuttaminen työpöytäsovellukseen

Työn ohjaaja Tommi Lahti

TIIVISTELMÄ

Opinnäytetyön toimeksiantaja oli Hämeen ammattikorkeakoulun Älykkäät Palvelut-tutkimusyksikkö, joka kehittää elinkeinoelämän ja yhteiskunnan digitalisaatiota ja palveluja.

Tavoitteena oli toteuttaa Hämeen ammattikorkeakoulun biotaloustutkimusyksikössä käytössä olevan levänkasvatusreaktorin pH-antureiden kalibrointiominaisuus jo olemassa olevaan työpöytäsovellukseen. Kalibrointiominaisuus tuli toteuttaa siten, että kalibroinnin voisi tehdä kuka tahansa ilman teknistä osaamista. Kalibrointiominaisuuden toteuttamisen ohella leväreaktorin tekninen arkkitehtuuri tuli dokumentoida selkeästi ja ymmärrettävästi. Tekniseen arkkitehtuuriin kuuluu leväreaktoriin liitetyt tekniset laitteet ja ohjelmat.

Teoriaosuuteen kerättiin tietoa mm. haastattelemalla levänkasvatusreaktorin parissa työskenteleviä henkilöitä. Käytännön osuudessa suunniteltiin ja toteutettiin uusi kalibrointiominaisuus työpöytäsovellukseen ja dokumentoitiin leväreaktorin tekninen arkkitehtuuri.

Opinnäytetyön tuloksena saatiin toimiva kalibrointiominaisuus työpöytäsovellukseen, joka otettiin käyttöön HAMKin biotalouden tutkimusyksikössä. Tuloksena syntyi myös selkeä dokumentaatio leväreaktorin teknisestä arkkitehtuurista.

Avainsanat fotobioreaktori, Arduino-kehitysalusta, kalibrointi

Sivut 31 sivua, joista liitteitä 2 sivua

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Degree Programme in Business Information Technology
Hämeenlinna

Author Janina Koskela **Year** 2017

Subject The Implementation of the Calibration Feature of Algae Photobioreactor's pH Sensors to the Windows Desktop Application

Supervisor Tommi Lahti

ABSTRACT

The commissioner of this thesis was Häme University of Applied Sciences' Smart Services research unit, which advances digitalization and services of local businesses.

The objective was to implement a calibration feature of the algae photobioreactor's pH sensors to an already existing Windows desktop application. The algae photobioreactor is in use at the bio economics research unit of Häme University of Applied Sciences. The calibration feature had to be implemented so that anyone would be able to calibrate the algae photobioreactor's pH sensors, even without technical knowledge. In addition to the implementation of the calibration feature, the algae photobioreactor's technical architecture had to be documented coherently. The technical architecture includes technical devices and programs of the algae photobioreactor.

Information in the theoretical part of the thesis was collected, among other things, by interviewing people working with the algae photobioreactor. In the practical part of thesis, the calibration feature was planned and implemented to the Windows desktop application and algae photobioreactors technical architecture was documented.

The outcome of the thesis was a functional Windows desktop application's calibration feature, which was introduced in bio economics research unit of Häme University of Applied Sciences. Another result was a comprehensible technical architecture documentation of algae photobioreactor.

Keywords algae photobioreactor, Arduino development platform, calibration

Pages 31 pages including appendices 2 pages

SISÄLLYS

1 JOHDANTO

Levä on yllättävän monipuolinen raaka-aine. Sitä voidaan käyttää kalankasvatuksessa ja siitä voidaan valmistaa kosmetiikkaa, lisäravinteita, ja mikä tärkeintä, biopolttoainetta. Maailmassa fossiiliset polttoaineet hupenevat hupenemistaan ja uusiutuvia energianlähteitä tarvitaan kipeästi. Levän hehtaarituotto on moninkertainen verrattuna muihin kasviöljyihin. Levän tuottaminen on kuitenkin kallista ja leväöljyn hinta on kilpailukyvytön (Neste Oil 2014). Kun levänkasvatusta saadaan tutkittua ja tuotantoa tehostettua, leväöljystä saattaa tulla merkittävä uusiutuva energianlähde.

Vaikka levänkasvatus on monimutkainen prosessi, suurin osa kasvatusajasta kuluu passiivisesti odotellen kasvun tasaantumista ja sitä hetkeä, jolloin levämassa voidaan seuloa vedestä. Jotta levänkasvatusta saadaan tehostettua globaalisti, täytyy tutkia, miten tähän vaiheeseen päästään mahdollisimman nopeasti tarjoamalla levälle juuri oikeat kasvuolosuhteet. Levä ei vaadi montaa eri muuttujaa kasvaakseen nopeasti, vaan tärkeintä on kasvuolosuhteiden tasaisuus ja vakaa elinympäristö. Automatisoimalla levänkasvatuksen passiivinen vaihe voidaan säästää sekä aikaa että rahaa. Levänkasvatusta kannattaa automatisoida niin paljon kuin on teknisesti mahdollista. Esimerkiksi levänkasvatusliuoksen sekoituksen ja hapensyötön, hiilidioksidin syötön ja valon saannin voi automatisoida edullisesti ja pienellä vaivalla.

Mikrokontrollerit ja korttitietokoneet ovat mullistaneet tekniikkaharrastelun ja tee se itse -nikkaroinnin ja niillä voi oman mielikuvituksen ja osaamisen rajoissa tehdä melkein mitä tahansa. Näillä luottokortin kokoisilla ja äärimmäisen edullisilla laitteilla voi itse tehdä esimerkiksi etäohjattavan kahvinkeittimen tai kukkienkasteluautomaatin. (Jääskeläinen 2016). Arduino-kehitysalusta on saavuttanut maailmanlaajuisen suosion edullisuudellaan ja yksinkertaisuudellaan. Sekä Arduino-ohjelmistot että -laitteistot ovat täysin avoimia, joten kuka tahansa voi kehittää uusia ohjelmia ja laitteita Arduino-kehitysalustalla käytettäväksi.

Opinnäytetyön toimeksiantaja on Hämeen ammattikorkeakoulun Älykkäät palvelut -tutkimusyksikkö, joka tukee lähialueen yrityksiä mm. digitalisaatioon ja palveluliiketoiminnan kehittämiseen liittyvissä muutosprosesseissa.

Työn tavoitteena on leväreaktorin pH-antureiden kalibrointiominaisuuden toteuttaminen jo olemassa olevaan työpöytäsovellukseen. Työpöytäsovelluksen kalibrointiominaisuus helpottaisi suuresti levänkasvatusprosessia, sillä muuten se ei onnistuisi ilman teknistä osaamista tai ohjelmoijan apua. Tarkoituksena on toteuttaa kalibrointiominaisuus siten, että kalibroinnin voisi suorittaa helposti ja vaivattomasti kuka tahansa.

Aihe on rajattu käsittelemään Hämeen ammattikorkeakoulussa käytössä olevaa leväreaktorikokonaisuutta, johon kuuluu leväreaktorin lisäksi levänkasvua ohjaava Arduino Uno -mikrokontrolleri sekä dataa käsittelevä työpöytäsovellus. Opinnäytetyössä tutkitaan yleisesti levänkasvatuksen automatisointia, sekä Arduino-kehitysalustaa ja Arduinon ohjelmoimista.

Teoriaosuudessa perehdytään Hämeen ammattikorkeakoulun leväreaktorin toimintaan ja levänkasvatuksen tarkoitusperiin, lyhyesti laitteiden kalibroimiseen, sekä Arduino-kehitysalustaan ja Arduinon ohjelmoimiseen. Käytännön osuudessa suunnitellaan uusi ominaisuus työpöytäsovellukseen ja pohditaan, mitä tulee ottaa huomioon uutta ominaisuutta suunniteltaessa ja toteuttaessa.

Opinnäytetyön tutkimuskysymyksiä ovat: Miten levänkasvatusreaktorin kalibrointiominaisuus toteutetaan työpöytäsovelluksena? Miten Arduinon ja tietokoneen saa keskustelemaan keskenään? Miten pH-anturi kalibroidaan?

2 LEVÄREAKTORI

2.1 Mikä on leväreaktori?

Leväreaktori on laite, jolla kasvatetaan levää (Nokkonen 2017). Hämeen ammattikorkeakoulussa biotalouden tutkimusyksikössä käytössä oleva reaktori on 30 litran suljettu säiliöreaktori. Levä kasvaa kasvatusliuoksessa, joka koostuu vedestä ja ravinteista (Nokkonen 2017).

Leväreaktori on osa LEVARBIO-hanketta, jossa Hämeen ammattikorkeakoulu on mukana osatoteuttajana. LEVARBIO-hankkeen tavoite on edistää levän mahdollisuuksia uusiutuvana energianlähteenä ja rehu-, elintarvike, energia-, sekä lannoitekäytössä. Hanke tuottaa tietoa sivu- ja jätevirtojen hyödyntämisestä levänkasvualustoina, jotta levä saataisiin osaksi ravinnekierrätystä, sekä levänkasvatuksen eri mahdollisuuksista ja vaatimuksista. (HAMK n.d.). Tarkoituksena on löytää teollisuudesta sivuvirtoja, joissa pystyttäisiin kasvattamaan levää. Levän kasvattaminen kemiallisesti on kallista, joten jos levä pystyisi hyödyntämään ja käyttämään kasvuunsa ravinteita esimerkiksi jätevesistä, laskisivat sekä leväkasvatuksen, että jätevesien puhdistuksen kustannukset (Nokkonen 2017).

Leväreaktorin on alun perin suunnitellut ja toteuttanut kaksi Hämeen Ammattikorkeakoulun sähkö- ja automaatiotekniikan opiskelijaa kesällä 2016 (Heikkilä 2016), mutta sitä on jatkokehitetty HAMkin biotalousyksikössä yhteistyössä HAMKin Älykkäät Palvelut-tutkimusyksikön kanssa (Saarinen 2017).

2.2 Levänkasvatus leväreaktorilla

Hämeen ammattikorkeakoulun leväreaktorilla kasvatetaan levää panosperiaatteella. Kasvatus aloitetaan laittamalla leväreaktoriin kasvatusliuos, sekä kasvatettava levä. Levän kasvussa on neljä vaihetta: alku, kiihtyvän kasvun vaihe, tasaisen kasvun vaihe ja kuolema. Tasaisen kasvun vaiheessa eli noin viikon kuluttua kasvatuksen aloittamisesta, levämassa sentrifugoidaan eli erotetaan vedestä, jonka jälkeen tehdään tarvittavat mittaukset ja analyysit (Nokkonen 2017). Analyysejä ja mittauksia tehdään myös kasvatuksen aikana.

Levä tarvitsee kasvaakseen paljon ravinteita, happea, hiiltä, valoa ja oikeanlaisen pH-tasapainon. Ravinteensa levä saa kasvatusliuoksesta, johon on lisätty typpeä ja fosforia. Typestä levä käyttää ensisijaisesti ammoniumtyppeä. (Nokkonen 2017).

Happea levä saa tavallisesta huoneilmasta, jota leväreaktoriin pumpataan jatkuvasti leväreaktorin ilmasäiliöstä. Leväreaktorin pohjalla on ilmastuskiviä, joiden rakenne tuottaa pieniä ilmakuplia, ja happi, sekä muut huoneil-

massa olevat aineet liukenevat kasvatusliuokseen. Ilmakuplat toimivat samalla levämassan ja kasvatusliuoksen sekoittimena. Liukenematon huoneilma nousee kuplina ylhäällä olevaan kaasutilaan ja putkea pitkin takaisin ilmasäiliöön. (Nokkonen 2017).

Levän kasvaessa kasvatusliuoksen pH-arvo nousee. Sitä voidaan laskea hiilidioksidilla. Hiilidioksidin syöttö on automatisoitu leväreaktorissa siten, että kun Arduino Uno -mikrokontrolleriin liitetyn pH-anturin antama arvo nousee yli määritellyn pH-arvon, Arduino lähettää hiilidioksidipulloon liitetylle venttiilille signaalin avautua (Heikkilä 2016). Hiilidioksidia syötetään leväreaktorin ilmasäiliöön ja hiilidioksidi painuu ilmasäiliön pohjalle Nokkonen 2017). Pohjalla oleva pumppu syöttää hiilidioksidipitoisen ilman leväreaktorin ilmastuskiviin ja se sekoittuu kasvatusliuokseen. Liukenematon hiilidioksidi nousee ilmakuplina leväreaktorin kaasutilaan ja putkea pitkin takaisin ilmasäiliöön. Ilmasäiliö ei ole täysin ilmatiivis, mutta suurin osa kaasuista jää vielä leväreaktorin käytettäväksi.

Valoa tuotetaan levälle LED-valoilla. Valot ovat päällä 16 tuntia vuorokaudessa ja ne toimivat ajastimella. LED-valot tuottavat levälle lämpöä, ehkä jopa liian paljon. Ne ovat aivan kiinni leväreaktorissa, sillä levän kasvaessa kasvatusliuos sakenee ja sitä myötä valon läpäisykyky heikkenee (Nokkonen 2017).

2.3 **Leväreaktorin tekninen arkkitehtuuri**

Opinnäytetyön toisena tavoitteena kalibrointiominaisuuden rinnalla oli dokumentoida leväreaktorikokonaisuuden tekninen arkkitehtuuri selkeästi ja ymmärrettävästi. Helpoin tapa teknisen arkkitehtuurin dokumentoimiseen oli kaavio, josta selviää leväreaktorissa käytetyt laitteet, ohjelmistot ja tiedonsiirtorajapinnat. Kaavio selityksineen on tämän opinnäytetyön liitteenä (LIITE 1).

Leväreaktorikokonaisuuteen kuuluu kasvatussäiliö, ilmasäiliö, hiilidioksidipullo, LED -valot, Arduino Uno -mikrokontrolleri ja anturit, sekä tietokone ohjelmineen. Leväreaktori on rakennettu Cooking Hacks -verkkokaupasta saatavaa OpenGarden Hydroponics -settiä hyödyntäen.

Koko leväreaktorin ydin on Arduino Uno -mikrokontrolleri. Se säätelee kasvatusliuoksen pH-tasapainoa, johon koko leväreaktorin toiminta nojautuu. Arduino Uno -piirilevy on omassa suojaavassa laatikossaan erillään itse kasvatussäiliöstä, jotta sen päälle ei roiskuisi esimerkiksi vettä (Heikkilä 2016). Arduinoon on liitetty kolme erilaista anturia: pH-anturi, veden sähkönjohtavuusanturi ja lämpöanturi. Anturit viedään kasvatussäiliön kannen läpi kasvatusliuokseen.

Toinen oleellinen asia leväreaktorin toiminnassa on hiilidioksidipullo, jonka venttiiliin on liitetty sähkömekaaninen kytkin eli rele. Arduino lähettää sähkösignaalin releelle, joka avaa magneettisen venttiilin ja hiilidioksidi pääsee vapautumaan hiilidioksidipullosta ilmasäiliöön johtavaan putkeen.

Ilmasäiliö ei ole ilmatiivis, mutta hiilidioksidi pysyy hyvin säiliön sisällä, sillä se on paljon ilmaa painavampaa ja painuu säiliön pohjalle, jossa oleva ilmapumppu pumppaa hiilidioksidi-ilmasekoituksen kasvatussäiliöön johtavaan putkeen. Kasvatussäiliön kaasutilassa eli säiliön yläosasta lähtee putki takaisin ilmasäiliöön, täten ilma pääsee kiertämään leväreaktorissa tauotta.

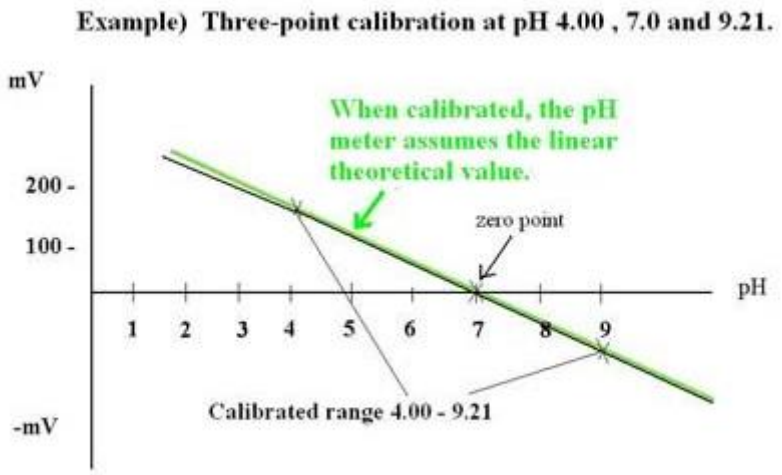
Arduino Uno -mikrokontrolleri on liitetty USB -kaapelilla tietokoneeseen. Levänkasvatuksen aikana tietokoneessa on käynnissä työpöytäsovellus, joka kerää Arduinon sarjaliikenteen avulla lähettämää dataa. Työpöytäsovellus muuntaa datan erillisiksi arvoiksi, jotka se tallentaa .csv-tiedostoon paikallisesti tietokoneen kovalevylle. Sovellus lähettää arvot myös HTTP POST-pyynnön avulla internetin yli Power BI-analysointipalveluun. Tiedot tallennetaan siis sekä paikallisesti että pilveen.

Leväreaktorin kasvatussäiliön ympärille on kiinnitetty LED -valoja, joista levä saa tarvitsemansa valon. Kasvatussäiliö tyhjennetään säiliön pohjalla olevasta venttiilistä ja kasvatussäiliön päällä on kansi, josta se päästään uudestaan täyttämään.

3 KALIBROINTI

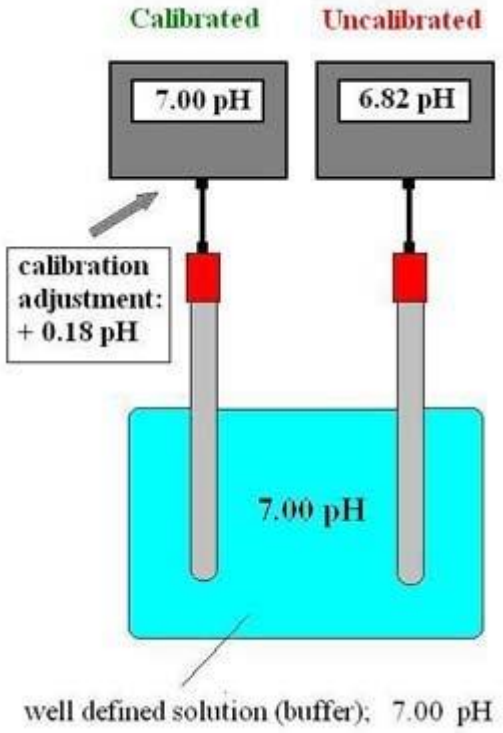
Kalibroinnin virallinen määritelmä on (SFS 5223): ”Toimenpiteet, joiden avulla annetuissa olosuhteissa saadaan mittalaitteen, mittausjärjestelmän tai kiintomitan näyttämien arvojen ja mittasuureen vastaavien arvojen välinen yhteys” (Lehtonen & Sihvonen 2004, 98). Kalibroinnin tavoite on selvittää laitteen toimintatarkkuus ja se on selvitettävä ennen tai jälkeen jokaista käyttökertaa.

Kalibroitavan laitteen toimintatarkkuus selvitetään vertailumateriaalin, esimerkiksi kalibrointiliuoksen, avulla. Kalibrointiliuoksen pitoisuus tiedetään tarkasti. Eri pitoisuuksia ja laitteen antamaa signaalia eri pitoisuuksilla verrataan keskenään ja niistä muodostetaan kalibrointikäyrä. Jos kalibrointikäyrä ei ole täysin suora, on joko kalibrointiliuoksessa tai laitteessa vikaa. Yleensä ensimmäisenä suoritetaan laitteen uudelleenkalibrointi ja vasta sen jälkeen kalibrointiliuoksen vaihto, jos kalibrointikäyrä ei suoristu. (Jaarinen & Niiranen 2005, 18–21).



Kuva 1. Kalibrointikäyrä pH-mittarin kalibroinnista. Mittarin signaalia millivoltteina verrataan pH-arvoon (All about pH n.d.).

3.1 Leväreaktorin pH-antureiden kalibrointi



Kuva 2. pH-anturin kalibrointi (All about pH n.d.).

Liuokset muuttavat pH-anturin antamaa arvoa ja leväreaktorin pH-anturi tulisi kalibroida jokaisen kasvatuserän lopussa. Kalibrointiin käytetään pH-arvon 4, 7 ja 10 kalibrointiliuoksia. (Nokkonen 2017). Arduinoon ajetaan erillinen kalibrointikoodi, joka lukee sekunnin välein pH-anturin antaman arvon millivoltteina ja lähettää sen sarjaporttiin (Saarinen 2017). Sarjaportista tulevat arvot näkee Arduino IDEn sarjaporttimonitorin avulla ja viidestä viimeisestä arvosta lasketaan käsin keskiarvo. Sama toistetaan kaikilla kalibrointiliuoksilla. Kun keskiarvot millivolttiarvoista on saatu, ne asetetaan manuaalisesti suoraan Arduinon leväreaktoriohjelman koodiin. Tällä hetkellä kalibrointi onkin monimutkainen ja hidas prosessi, jota ei voi suorittaa ilman ohjelmoijan apua.

4 ARDUINO -KEHITYSALUSTA

”Arduino on avoin fyysisen tietojenkäsittelyn alusta” (Banzi 2011, 1). Fyysisen tietojenkäsittelyn ”avulla suunnitellaan laitteita, joiden kanssa ihminen voi olla vuorovaikutuksessa sensorien ja käyttölaitteiden avulla ja joiden toimintaa ohjaa mikrokontrolleri (pienen tietokoneen sisältämä mikropiiri)” (Banzi 2011, 3). Ihmiset, joilla ei ole tietoteknistä kokemusta, kuten taiteilijat tai suunnittelijat, voisivat hyödyntää tietojenkäsittelyä ja luoda uusia ratkaisuja ja innovaatioita. Fyysisen tietojenkäsittelyn tarkoitus on luoda yhteys fyysisen maailman ja tietokoneen virtuaalisen maailman välille. (O'Sullivan & Igoe 2004, 17–19).

Arduino koostuu kahdesta osasta: Arduino-piirilevystä ja Arduino IDE -ohjelmistosta. (Banzi 2011, 1). Sekä laitteisto että ohjelmisto ovat täysin avointa lähdekoodia, jotta niitä voidaan käyttää, muokata ja levittää vapaasti (Nussey 2013, 16).

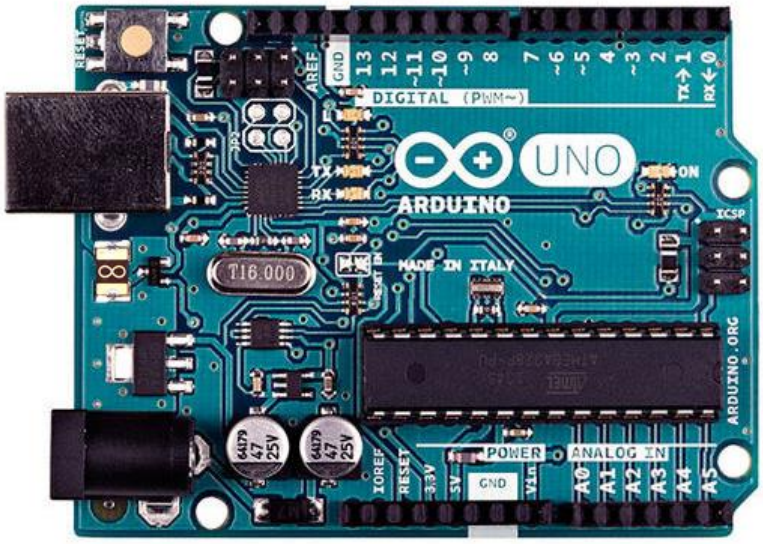
Arduino-projekti sai alkunsa vuonna 2005 Italiassa, Interaction Desing Institute Iverassa (IDII), joka on vuorovaikutussuunnitteluun erikoistunut koulu. Arduinon kehitti Mazzimo Banzi ja David Cuartellies (Bayle 2013, 9; Nussey 2013, 8) ja projektin tavoitteena oli luoda helppokäyttöisiä ja edullisia laitteita vuorovaikutussuunnittelun opiskelijoille. Se pohjautuu kahteen aikaisempaan projektiin: Processing ja Wiring.

Processing-projektin aloittivat Casey Reas ja Benjamin Fry vuonna 2001. Ideana oli suunnitella ”digitaalinen luonnosvihko”, jonka avulla kuka tahansa pääsisi kokeilemaan ohjelmointia. Arduino IDE -ohjelmisto on ottanut paljon vaikutteita Processing-luonnosvihkosta. Esimerkiksi Arduino-ohjelmia kutsutaan sketseiksi eli luonnoksiksi, niin kuin Processing-projektissakin. (Nussey 2013, 7–8).

Hernando Barragán kehitti vuonna 2003 Wiring-nimisen mikrokontrollerin, joka on Arduino-mikrokontrollerien edeltäjä. Wiring-projektin tavoitteena oli luoda edullinen mikrokontrolleri ei-teknisille ihmisille, kuten taiteilijoille ja suunnittelijoille. (Nussey 2013, 9–10).

4.1 Arduino Uno -piirilevy

Arduino Uno (Kuva 1) on yksinkertainen piirilevy, joka sisältää erilaisia analogisia ja digitaalisia liittimiä ja komponentteja. Tärkein osa on kuitenkin ATmega328-mikrokontrolleri, joka on koko piirilevyn sydän. (Banzi 2011, 17–18).



Kuva 3. Arduino Uno-piirilevy (Arduino n.d.c).

ATmega328-mikrokontrolleri on Atmelin valmistama mikropiirilevy, joka sisältää tietokoneelle tyypilliset osat kuten prosessorin, muistin sekä sisään- ja ulostuloliittimet. Mikrokontrollerissa on 32 kB:n flash-muisti, 2 kB:n SRAM-muisti ja 1 kB:n EEPROM-muisti, sekä 14 digitaalista ja 6 analogista I/O-pinniä (Nussey 2011, 19; Bayle 2013, 7; Langbridge 2015, 10).

Mikrokontrollerissa on 28 jalkaa (Banzi 2011, 17), jotka on yhdistetty piirilevyn reunoilla oleviin mustiin liittimiin eli pinneihin. Pinnit on nimetty ja numeroitu käyttötarkoituksen mukaan analogisiksi-, digitaalisiksi- ja virtapinneiksi. Analogiset pinnit lähettävät ja vastaanottavat vain analogisia signaaleja, jotka voivat olla mitä tahansa 0 ja 5 voltin väliltä. Digitaaliset pinnit lähettävät ja vastaanottavat digitaalisia signaaleja. Digitaalisella signaalilla on vain kaksi arvoa: 5 tai 0 volttia, eli joko päällä tai pois päältä. (Nussey 2013, 20). Arduino-piirilevyyn voidaan syöttää virtaa tietokoneesta USB-liittimen kautta tai tasavirtamuuntajan virtapinnien kautta. Jos piirilevyn virtapinneihin on liitetty jokin ulkoinen virtalähde, piirilevy alkaa toimia sen kautta automaattisesti. Muussa tapauksessa käytetään USB-liitintä. (Banzi 2011, 18).

Arduino on jo itsessään tehokas laitteisto ja sisältää lukuisia sisään- ja ulostuloliittimiä, mutta se on vasta hyvä alku. Koska Arduino on avoimen lähdekoodin järjestelmä, monet eri elektroniikkayritykset ovat kehittäneet lisäpiirilevyjä (eng. shields), joilla voidaan laajentaa Arduinon ominaisuuksia. Lisäpiirilevy asetetaan Arduino-piirilevyn päälle ja yhdistetään I/O-pinneihin. Esimerkiksi mikro-SD-kortinlukija voisi olla hankala toteuttaa pelkästään Arduinon kytkentälevyn avulla. Markkinoilta löytyy valmiita kortinlukija-lisäkortteja, jotka kytketään Arduino-piirilevyyn ja jotka vaativat vain muutaman rivin koodia toimiakseen. Muita mahdollisia ominaisuuksia

lisäkorteilla voi olla mm. Ethernet- tai WiFi-verkkoyhteys. Suurinta osaa lisäkorteista voidaan käyttää yhdessä, mutta jotkin lisäkortit vaativat samoja I/O-liittimiä. (Langbridge 2015, 20–21).

Arduino käyttää kolmea erilaista muistiteknologiaa: RAM-, flash- ja EEPROM-muistia. RAM-muisti on lyhytaikainen muisti ja sinne tallennetaan ohjelmakoodin käyttämiä muuttujia ja muita komponentteja. RAM-muisti tyhjenee aina, kun Arduinosta katkaistaan virta. Kun Arduinoon ladataan ohjelma, se tallentuu flash-muistiin ja edellinen ohjelmakoodi pyyhkiytyy pois. Flash-muisti sisältää myös alkulatausohjelman, jonka avulla ohjelmakoodi käynnistyy. EEPROM-muisti on pitkäaikainen, eikä pyyhkiydy pois, kun uusi ohjelmakoodi ladataan Arduinoon. Arduino Unossa on EEPROM-muistia yhteensä 1 024 tavua ja muistiin voidaan tallentaa arvoja ohjelmakoodin avulla. Tietojen tallentaminen muistiin kuitenkin vahingoittaa joka kerralla hieman EEPROM-muistilaitetta, mutta suurin osa laitteista tukee yli 100 000 kirjoituskertaa. EEPROM-muisti on tarkoitettu vain sellaisten arvojen tallentamiseen, joita ei tarvitse vaihtaa kovinkaan usein. (Langbridge 2015, 103–104).

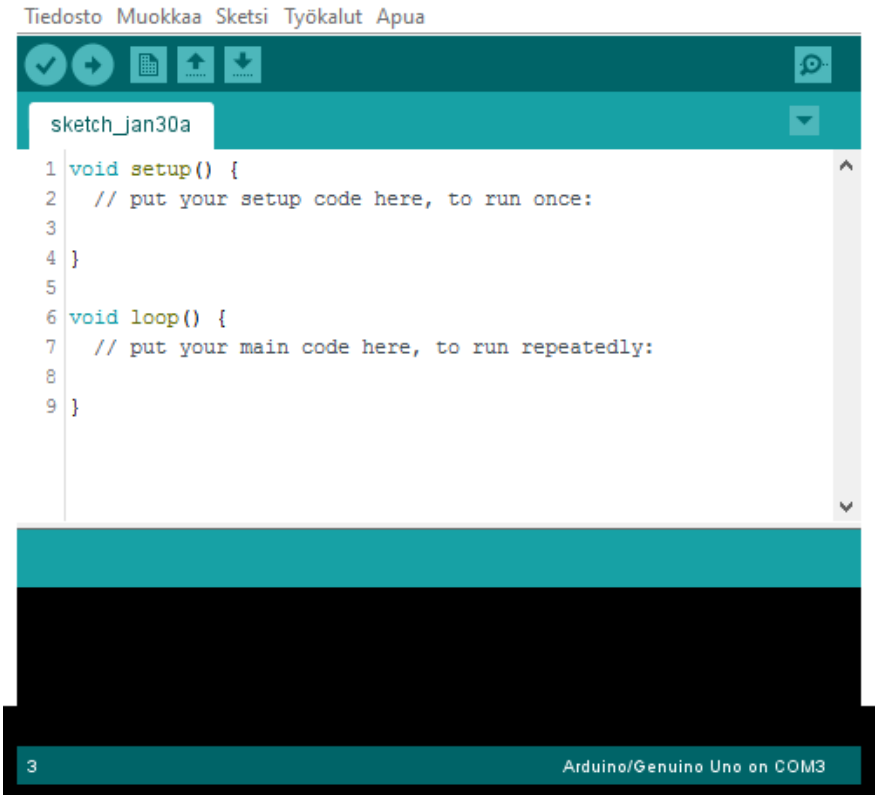
4.2 **Arduino IDE -kehitysympäristö**

Arduino IDE on integroitu kehitysympäristö (Integrated Development Environment, IDE), eli ohjelmisto, jolla luodaan, testataan ja ladataan ohjelmia Arduino-piirilevyn mikrokontrollerille. Arduino IDEn voi ladata ilmaiseksi osoitteesta www.arduino.cc/en/Main/Software. Se on saatavilla Windows-, Macintosh OS X- ja Linux-käyttöjärjestelmille.

Arduino IDEn graafinen käyttöliittymä on yksinkertainen ja sillä on helppo tehdä ohjelmia. Ohjelmia, joita Arduino IDEllä luodaan, kutsutaan sketseiksi. Nimeämistapa on kulkeutunut Processing-projektista. (Banzi 2011, 20; Nussey 2013, 33, 39–40). Ennen ohjelmoimisen aloittamista täytyy määritellä käytettävä Arduino-piirilevy ja sarjaportti, jossa piirilevy on kiinni.

Arduino-piirilevyn mallin voi varmistaa sen takapuolella olevasta tekstistä. Malli määritellään Arduino IDEn valikkopalkista Tools→Board. Listasta valitaan käytettävä malli kaikkien Arduino IDEn tukemien piirilevyjen joukosta.

Sarjaportti (eng. serial port) on yhteys, jonka avulla tietokone ja Arduino voivat kommunikoida keskenään. Arduino liitetään tietokoneeseen USB-liitännällä, joka tässä tapauksessa toimii sarjaporttina. Sarjaportti valitaan Arduino IDEssä valitsemalla ylhäältä valikkopalkista Tools → Serial Port. Listassa näkyy portit, joihin on kytketty jokin laite, johon voidaan ottaa yhteyttä sarjaportin kautta. Portit nimetään COM1, COM2, COM2 jne. Jos Arduino on juuri äskettäin kytketty tietokoneen USB-porttiin, se näkyy listassa ensimmäisenä suurimmalla sarjaportin numerolla (esim. COM3). (Nussey 2013, 43–45).



Kuva 4. Kuvakaappaus Arduino IDE:n käyttöliittymästä.

Kun tarvittavat määrittelyt on tehty, voidaan aloittaa Arduinon ohjelmoiminen. Arduinoa ohjelmoidaan C-kielellä, joka on yksi yleisimmistä ohjelmointikielistä. Kun ohjelmakoodi on laadittu, sen syntaksi täytyy varmistaa. Arduino IDE tarkistaa ohjelmakoodin ja etsii siitä virheitä. Jos varmistus onnistuu, ohjelmakoodi käännetään mikrokontrollerin ymmärtämäksi binaaritiedostoksi. Binaaritiedosto ladataan mikrokontrollerille USB-portin kautta. Aina, kun uusi ohjelma ladataan mikrokontrollerille, aikaisempi pyyhitään pois. Mikrokontrolleri siis suorittaa aina vain yhtä ohjelmaa kerrallaan. (Banzi 2011, 20; Nussey 2013, 47–50; O'Sullivan & Igoe 2004, 55).

Arduino IDE -kehitysympäristössä on sarjamonitori (Kuva 2: suurennuslasi oikeassa yläkulmassa), jonka avulla voidaan lähettää ja vastaanottaa dataa Arduino-mikrokontrollerista. Sarjamonitori tarkkailee määriteltyä sarjaporttia. (Nussey 2013, 40)

4.3 **Ohjelmointi**

Arduinoa ohjelmoidaan C- ja C++-kieleen perustuvalla ohjelmointikielellä. C-kieli on prosessoririippumaton ohjelmointikieli, koska C-kielen kääntäjiä on melkein kaikissa prosessoreissa. Sillä voi helposti olla vuorovaikutuksessa eri laitteistojen kanssa ja sen takia Arduinon ohjelmointikieli perustuu C-kieleen. (Bayle 2013, 70–72).

Jokaisessa Arduino-sketsissä täytyy määritellä funktiot void setup() ja void loop(), vaikka ne olisivat tyhjiä. Ilman niitä sketsi ei käänny Arduino-ohjelmaksi. Setup-funktio ajetaan aina kerran, kun Arduino käynnistyy. (Nussey 2013, 53-54). Setup-funktiossa alustetaan esimerkiksi tarvittavat muuttujat, käytettävät pinnit ja muut koodissa käytettävät komponentit. Koodin luettavuuden helpottamiseksi esimerkiksi muuttujien määrittelyt kannattaa tehdä setup()-funktiossa sen sijaan, että ne määrittelisi myöhemmin juuri ennen käyttöä. (Langbridge 2015, 35). Loop-funktio kiertää nimensä mukaisesti samaa silmukkaa yhä uudelleen ja uudelleen, niin kauan, kunnes Arduino käynnistetään uudelleen. Se on koko koodin ydin, koko Arduinon toiminta perustuu periaatteessa loop-funktioon. On hyvä muistaa, että jos loop-funktiossa määrittelee muuttujia, ne ylikirjoitetaan aina seuraavalla kierroksella. (Nussey 2013, 56; Landbridge 2015, 36).

Muuttujia käytetään tiedon tallentamiseen ja ne täytyy määritellä ennen käyttöä. Muuttujaa määriteltäessä sille täytyy antaa muuttujan tietotyyppi, sen nimi tai tunniste. Tietotyypit täsmentävät muuttujan käyttökohteen ja luonteen. Arduinon muisti on rajallinen, joten on hyvä muistaa, kuinka paljon eri tietotyypeille täytyy varata tilaa. (Bayle 2015, 90–95).

Taulukko 1. Tietotyypit Arduinon ohjelmointikielessä. (Bayle 2013, 91–93; Banzi 2011, 95–98).

Tietotyyppi	**Määritelmä**	**Koko**
void	Käytetään funktioiden määrittelyssä.	
boolean	Totuusarvotietotyyppi, jonka arvona voi olla joko true tai false.	1 tavu
char	Yksittäinen merkki, kuten ’A’, tallennettuna ASCII-taulukon mukaan numerona. Numeerinen arvo voi olla välillä -127 ja 127.	1 tavu
byte	Voi tallentaa numerot 0–255. Vie muistia vain yhden tavun verran.	8 bittiä
int	Voi tallentaa numerot -32,768–32,767. Käyttää kaksi tavua muistitilaa. Eniten Arduinossa käytetty tietotyyppi.	2 tavua
unsigned int	int-tietotyypin kaltainen kahden tavun tietotyyppi, mutta etuliite unsigned tarkoittaa, ettei siihen voi tallentaa negatiivisia lukuja. Lukualue on 0–65,535.	2 tavua
long	Voi tallentaa numerot -2,147,483,648–2,147,483,647. Käyttää 4 tavua muistitilaa.	4 tavua

	Kooltaan kaksinkertainen int-tietotyyppiin verrattuna. Voi olla int-tietotyypin tavoin myös unsigned.	
float	Tietotyyppi, joka voi sisältää liukulukuja (desimaalilukuja). Käyttää neljä tavua muistia. Float-tietotyyppiä kannattaa käyttää säästeliäästi, sillä liukulukujen käsittelyyn tarvittavat funktiot vievät runsaasti muistitilaa.	4 tavua
double	Liukulukutietotyyppi, jonka maksimiarvo voi olla kaksi kertaa tarkempi kuin float-tietotyypin maksimiarvo.	4 tavua
Array	Taulukkotietotyyppi, jonka arvoihin voi viitata ja muokata erityisen indeksin avulla. Tallennetaan taulukkomuotoista tietoa.	taulukkoelementtien määrä x taulukon tietotyypin koko
string	Merkkijono, eli tietotyyppi joka sisältää joukon ASCII-merkkejä. Käyttää yhden tavun muistia merkkiä kohden. Merkkijonon lopussa on null-merkki, joka kuvastaa merkkijonon päättymistä.	1 tavu x merkkijonon pituus

Helpoin tapa saada Arduino ja tietokone kommunikoimaan keskenään on käyttää sarjaliikennettä. Sarjaliikenteen avulla Arduino ja tietokone lähettävät ja vastaanottavat digitaalisia impulsseja eli bittejä. Jotta sarjaliikennettä voidaan käyttää, täytyy osapuolilla olla yhteinen protokolla tai määritelmä siitä, miten bitit sarjaportissa liikkuvat. Sarjayhteyden luomiseksi täytyy määritellä käytettävä sarjaportti, sekä bittien siirtonopeus ja lähetettävien datapakettien rakenne (O'Sullivan & Igoe 2004, 137). Siirtonopeus (eng. baud rate) sarjaliikenteessä tarkoittaa yksinkertaisesti sitä kuinka nopeasti dataa voidaan siirtää. On tärkeää, että keskustelevilla laitteilla on sama siirtonopeus, sillä jos toinen laitteista lähettää dataa nopeammin, kuin mitä toinen pystyy lukemaan, tieto ei välity oikealla tavalla. Arduino käyttää oletuksena 9600 bps:n siirtonopeutta, mutta se voi toimia myös nopeamminkin.

Arduinon ja tietokoneen välillä käytetään RS-232-sarjaprotokollaa eli USB-porttia. RS-232-protokolla ei vaadi jatkuvaa datansiirtoa, vaan se sallii sarjaliikenteen aloittamisen ja lopettamisen, milloin tahansa. (Langbridge 2015, 84). Jotta sarjaliikenteen vastaanottaja tietäisi missä ollaan menossa, data lähetetään sarjaporttiin kapseloituina paketteina. Paketit koostuvat aina aloitusbitistä, kahdeksasta databitistä, sekä lopetusbitistä. Aloitusbitti merkitsee paketin alun, databitit ovat varsinainen data, jota siirretään ja lopetusbitti määrää paketin lopun (Langbridge 2015, 85). Kun

data lähetetään paketteina, vastaanottaja pysyy helposti synkronoimaan vastaanottamaansa datavirtaa.

Arduino on suljettu järjestelmä, ohjelmakoodin debuggaus eli virheenkorjaus ja käyminen läpi vaihe vaiheelta ei ole mahdollista ilman, että ohjelmoija itse huomioi tämän (Langbridge 2015, 86). Debuggaus helpottuu huomattavasti sarjaliikennettä hyödynnettäessä. Ohjelmakoodiin voidaan lisätä koodirivejä, jotka lähettävät sarjaporttiin informaatiota esimerkiksi muuttujista tai funktioiden suorittamisesta. Sarjamonitorilla voidaan seurata sarjaportista tulevaa dataa ja huomata helposti mahdolliset virheet.

Aina ennen sarjaliikenteen käyttöä Arduinossa, sarjaporttiyhteys täytyy alustaa ja määrittää siirtonopeus Serial.begin()-funktiolla. Funktiota kutsutaan setup()-funktion sisällä, koska siirtonopeutta ei yleensä muuteta kesken kaiken. (Langbridge 2015, 87).

Taulukko 2. Arduinossa yleisimmin käytetyt sarjaliikennefunktiot. (Banzi 2011, 107–109).

Serial-funktio	**Selitys**
Serial.begin(siirtonopeus)	Tarvitaan sarjaporttiyhteyteen. Täytyy kutsua, ennen kuin sarjaporttia voidaan käyttää. Siirtonopeus on yleensä 9600.
Serial.print(data)	Lähettää dataa sarjaportille. Voi lähettää mitä tahansa merkkejä.
Serial.println()	Sama kuin Serial.print(), mutta lisää loppuun rivinvaihdon (\r\n).
int Serial.available()	Palauttaa lukemattomien tavujen määrän kokonaislukuna. Jos tavut on luettu lukufunktiolla, palauttaa arvon 0.
int Serial.read()	Lukee tavu kerrallaan sarjaportista tulevaa dataa.
Serial.flush()	Arduino säilyttää kaiken tulevan datan puskurissa. Dataa voi tulla nopeammin sisään, kuin ohjelma ehtii sitä käyttää, joten jos puskuriin tarvitaan tuoreempaa tietoa, voidaan käyttää flush()-funktiota puskurin tyhjentämiseen.

Ohjelmointikirjasto on kokoelma C- tai C++-kielellä kirjoitettuja ohjelmointikomponentteja ja -resursseja. Ne voivat sisältää ohjeita ja lähdeaineistoja, aliohjelmia ja uudelleenkäytettäviä funktioita, sekä luokkia ja tyyppimääritelmiä. Useimmiten ohjelmointikirjasto on vain kokoelma funktioita, jotka on jo kertaalleen kirjoitettu ja mitä voi käyttää uudelleen omassa ohjelmakoodissa. (Bayle 2013, 72, 529-530; Langbridge 2015, 378). Ohjelmakirjastojen käyttäminen säästää ohjelmoijan aikaa ja Arduinon vähäistä muistia. Koodista tulee selkeämpi ja lyhyempi, sillä koodissa voidaan vain kutsua eri funktioita ilman, että ne pitäisi kirjoittaa erikseen koodiin. (Langbridge 2015, 378). Ohjelmakirjasto otetaan käyttöön #include<>-komennolla. Arduinon omat ohjelmointikirjastot tuovat jo valtaisan määrän lisäominaisuuksia käytettäväksi alustalla, mutta ohjelmiin voidaan lisätä myös kolmannen osapuolen ohjelmointikirjastoja. (Langbridge 2015, 377). Ennen kuin kolmannen osapuolen ohjelmointikirjasto voidaan ottaa käyttöön koodissa, se pitää lisätä Arduino IDE-kehitysympäristöön. Ohjelmointikirjasto voidaan lisätä Arduino IDE:n avulla tai manuaalisesti (Langbridge 2015, 382). Manuaalisesti lisättäessä ohjelmointikirjaston tiedostot kopioidaan Arduinon ohjelmointikirjasto kansioon tietokoneella. Se löytyy yleensä Tiedostot-kansion Arduino-alikansiosta. Jos kirjasto on lisätty onnistuneesti, sen pitäisi näkyä Arduino IDE:ssä valikkokohdassa Sketsi→Include Library.

Tietojen lukemien ja tallentaminen pitkäaikaiseen EEPROM-muistiin onnistuu EEPROM-kirjaston funktioiden avulla. Kirjasto lisätään ohjelmakoodiin #include<>-komennolla. (Langbridge 2015, 104). EEPROM-kirjasto koostuu ainoastaan viidestä eri funktiosta, jotka perustuvat EEPROM-muistin lukemiseen ja kirjoittamiseen.

EEPROM-kirjaston funktiot ovat: read(muistipaikka), write(muistipaikka, data), put(muistipaikka, data), update(muistipaikka, data), get(muistipaikka, data). (Arduino n.d.).

Funktiot read() ja write() käsittelevät dataa tavuina. Funktio read() lukee datan määritellystä muistipaikasta ja se palauttaa datan tavuna. Write()-funktio toimii samaan tapaan, kuin read(), mutta lukemisen sijaan se kirjoittaa tavun määrättyyn muistipaikkaan.

```
#include <EEPROM.h>

int address = 0;
byte value = B10010;
byte valueFromEEPROM;

void setup() {

//Kirjoitetaan tavu muistipaikkaan 0:
EEPROM.write(address, value);

//Luetaan tavu muistipaikasta 0:
valueFromEEPROM = EEPROM.read(address);

}

void loop(){}
```

Kuva 5. Esimerkkikoodi yhden tavun kirjoittamisesta ja lukemisesta EEPROM-muistiin.

Jos EEPROM-muistiin halutaan tallentaa kokonaisia merkkijonoja tai lukuja, täytyy funktioiden käyttöä hieman soveltaa. Merkkijonojen tallentamiseksi, merkkijonosta täytyy tehdä char-tietotyypin taulukko ja tallentaa taulukon tiedot solu kerrallaan. Lukujen tallentaminen on hieman monimutkaisempaa, koska kokonaisluvut eli int-tietotyypin muuttujat ovat aina kahden tavun mittaisia ja yhteen EEPROM-muistipaikkaan mahtuu vain tavun verran dataa (Langbridge 2015, 104-105, 108–109). Tämän vuoksi int-muuttujat täytyy purkaa kahteen osaan ja sen jälkeen tallentaa osat EEPROM-muistin kahteen peräkkäiseen muistipaikkaan.

Helpompi tapa tallentaa dataa EEPROM-muistiin on käyttää put()-funktiota ja datan lukemiseen get()-funktiota. Put()-funktion avulla voidaan tallentaa mikä tahansa tietotyyppi tai objekti EEPROM-muistiin. Tietoja ei tarvitse tallentaa tavu kerrallaan, niin kuin write()-funktiolla. Get()-funktio toimii samaan tapaan, mutta sen avulla voidaan lukea mikä tahansa tietotyyppi EEPROM-muistista. Parametreina put()-funktiolle annetaan muistipaikka, johon data tallennetaan, ja muuttuja, joka tallennetaan. Get()-funktion parametrin ovat muistipaikka, josta tieto haetaan, sekä muuttuja, johon haettu tieto tallennetaan. (Arduino n.d.a; Arduino n.d.b).

4.4 Open Garden

Open Garden on avoin alusta eri kasvien kasvun tarkkailuun. Se on suunniteltu Arduino -kehitysalustalle ja sen avulla voidaan kerätä tietoa esimerkiksi kasvien kasvualustan lämpötilasta ja kosteudesta sekä lähettää saadut tiedot WiFin, GPRS:n tai 3G:n välityksellä verkkopalvelimelle. Siitä on kolme erilaista settiä tai kokonaisuutta, jotka soveltuvat eri kasvien kasvattamiseen eri kasvualustoilla. Leväreaktorissa on käytössä Open Garden

Hydroponics-setti (Kuva 6), joka on tarkoitettu vesiviljelyn tarkkailuun. (Cooking Hacks n.d.). Vastaavanlaisia vesiviljelyyn tarkoitettuja avoimia alustoja on paljon muitakin, mutta Open Gardenissa on juuri ne levänkasvatukseen tarvittavat anturit ja ominaisuudet.

Hydroponics-settiin kuuluu Open Garden -lisäpiirilevy, sekä Hydroponics-laajennuspiirilevy Arduino Uno -mikrokontrollerille, sekä pH-, sähkönjohtavuus- ja lämpötila-anturit johtoineen ja liittimineen.

Open Garden -lisäpiirilevyyn on liitetty lämpötila-anturi, jonka avulla voidaan mitata kasvatusliuoksen lämpötilaa. Hydroponics-laajennuspiirilevyn avulla mitataan kasvatusliuoksen pH-arvoa ja sähkönjohtavuutta. Piirilevyt on asennettu Arduinoon analogisten ja virtaliittimien avulla.

Open Gardenille on oma C++-kirjasto, jonka avulla anturien lukeminen Arduinolla on helpompaa. Kirjasto on täysin avointa lähdekoodia, joten kuka tahansa voi muuttaa tai parannella sitä tarpeen mukaan (Cooking Hacks n.d.). Kirjasto lisätään Arduino-projektiin kopioimalla kirjaston sisältämät tiedostot Arduino IDEn Sketchbook-kansioon ja tuomalla ne sen jälkeen Arduino IDEn avulla Arduino-projektiin.

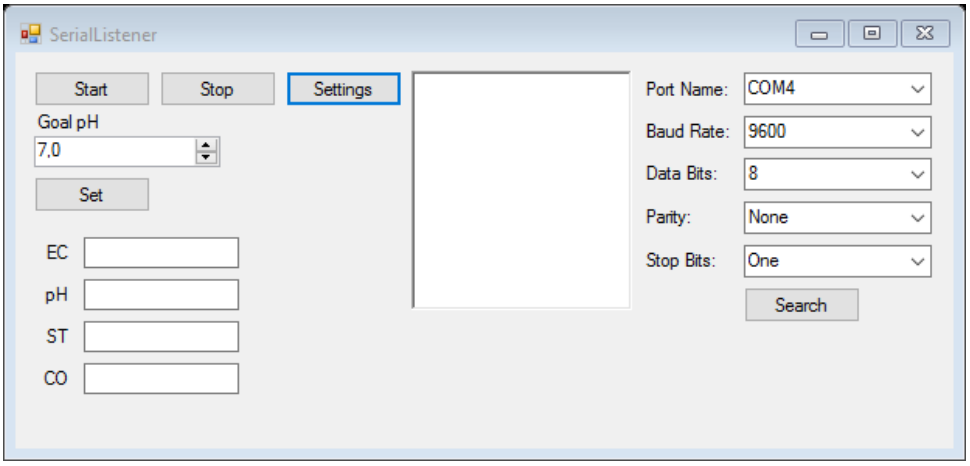
Kuva 6. Open Garden Hydropnics (Cooking Hacks n.d.).

5 TYÖPÖYTÄSOVELLUS

5.1 Alkuperäinen sovellus

Alkuperäinen sovellus on tehty hallitsemaan levänkasvatusta ja keräämään levänkasvatuksesta saatavaa dataa. Sen on suunnitellut ja toteuttanut Hämeen Ammattikorkeakoulun Älykkäät palvelut-tutkimusyksikön teknologia-assistentti. Sovellus on englanninkielinen, koska sen parissa saattaa työskennellä suomea osaamattomia.

Sovelluksen ominaisuuksia kirjoitushetkellä ovat: tavoite-pH:n asettaminen ja lähettäminen Arduinoon, sarjaporttiyhteyden luominen ja katkaiseminen, sarjaporttiyhteyden asetusten muokkaaminen, Arduinosta saatavien tietojen tallentaminen paikallisesti Excel-taulukkoon (c:\\temp -kansioon) ja Arduinosta saatavien tietojen lähettäminen Power BI-visualisointiohjelmaan.



Kuva 7. Alkuperäinen työpöytäsovellus.

Alkuperäinen työpöytäsovellus on toteutettu Windows Forms -luokkakirjastolla Visual Studiolla. Käyttöliittymä on hieman monimutkainen ja sovellus koostuu vain yhdestä ikkunasta. Yhteys Arduinoon luodaan painamalla ”Start”. Sovellus luo sarjaporttiyhteyden, jonka avulla pystytään lähettämään ja vastaanottamaan dataa. Yleensä sovelluksen sarjayhteysasetuksia (Settings) ei tarvitse muokata tai edes valita ennen yhteyden luomista, sillä sovellus hakee sopivat yhteysasetukset automaattisesti. Ainoa asetus, jota saattaisi joutua joskus vaihtamaan, on ”Port Name”, eli sarjaportin nimi, jossa Arduino on yhdistettynä.

Kun yhteys Arduinoon on luotu, työpöytäsovelluksella voidaan asettaa toivottu pH-arvo (pH 4–10), jota kasvatusliuoksen pH-arvo ei saisi ylittää. Arvo valitaan pudotusvalikosta. Haluttu pH-arvo lähetetään Arduinoon sarjayhteyden avulla painamalla ”Set” ja Arduino käsittelee sen ja muuttaa tavoite-pH-arvoaan.

Muutaman sekunnin välein Arduino lähettää sarjayhteyden kautta merkkijonon, jonka työpöytäsovellus erittelee ja tallentaa saadut arvot eri muuttujiksi. Arvot merkkijonossa ovat järjestyksessä: sähkönjohtavuus, pH-arvo, lämpötila, hiilidioksidin syöttö -laskurin arvo, sekä sen hetkinen tavoite-pH. Merkkijono voisi olla esimerkiksi: ”a;45;6.7;19.38;1;7.0;o”. Merkki ”;” on arvojen erotusmerkki ja merkit ”a” ja ”o” toimivat merkkijonon aloitus- ja lopetusmerkkeinä, joista työpöytäsovellus tunnistaa merkkijonon olevan tallennettavaksi sopiva. Kun merkkijonon erittely on tehty, sovellus näyttää saadut arvot erillisissä laatikoissaan sovellusikkunan vasemmassa laidassa. Jotta Arduinosta saatava data pysyisi varmasti tallessa, arvot tallennetaan paikallisesti sekä koneen kiintolevylle csv-tiedostoon, että myös pilveen Power Bi-datavisualisointipalveluun (Saarinen 2017).

6 KALIBROINTIOMINAISUUDEN SUUNNITTELU JA TOTEUTUS

6.1 Uuden ominaisuuden suunnittelu alkuperäiseen työpöytäsovellukseen

Baylen (2013, 67) mukaan ”ohjelmointi on tietokoneohjelmien lähdekoodin suunnittelun, kirjoittamisen, testaamisen, virheenkorjauksen ja ylläpitämisen sisältämä prosessi”. Sovellus täytyy suunnitella ennen, kuin voi alkaa edes miettiä sen ohjelmointia. Suunnitella voi piirtämällä, kirjoittamalla tai tekemällä kaavioita sovelluksen ominaisuuksista. Halutuista toiminnoista saa helposti hyvän käsityksen kirjoittamalla sovelluksen ominaisuudet ja loogiset vaiheet pseudokoodina. Pseudokoodi on vähän niin kuin oikeaa koodia, mutta sen kirjoittamisessa käytetään selkeitä, kokonaisia lauseita. Pseudokoodin avulla saa paremman käsityksen siitä, mikä on sovelluksen toimintaperiaate ja miten toiminnot voidaan saavuttaa. (Bayle 2013, 67-68).

Leväreaktorin kalibrointiominaisuuden suunnittelussa käytettiin apuna eräänlaista pseudokoodia. Suunnittelu auttoi ymmärtämään, mitkä asiat tulee sovelluksen toteutuksessa ottaa huomioon. Kalibrointiominaisuudesta tehtiin kaksi erillistä pseudokoodia: työpöytäsovelluksen pseudokoodi (Kuva 8) ja Arduinon pseudokoodi (Kuva 9).

```
//Käyttäjä painaa kalibrointinappia
lähetetään Arduinoon käsky kalibroida
avataan kalibrointi-ikkuna
kalibroi = true
pH4 = null
pH7 = null
pH10 = null

WHILE (kalibroi = true) THEN
    //Käyttäjä asettaa pH-anturin kalibrointiliuokseen
    arvo = sarjaportista saatava pH-arvo
    kalibroitava = käyttäjän valitsema arvo
    IF ("Tallenna"-nappia painetaan) THEN
         IF (kalibroitava = 4) THEN
            pH4 = arvo
         END IF
         IF (kalibroitava = 7) THEN
            pH7 = arvo
         END IF
         IF (kalibroitava = 10) THEN
            pH10 = arvo
         END IF
    END IF
    IF ("Kalibroi"-nappia painetaan) THEN
         IF (pH4 != null && pH7 != null && pH10 !=null) THEN
            lähetetään kalibrointiarvot sarjaporttiin
            kalibroi = false
         ELSE
            jatka
         END IF
    END IF
END WHILE
```

Kuva 8. Työpöytäsovelluksen toimintalogiikka pseudokoodina.

```
FUNCTION setup()
  IF (EEPROM ei ole tyhjä) THEN
    lue pH-arvot EEPROMista
    kalibroi pH-anturit
  ELSE
    kalibroi pH-anturit oletusarvoilla
  END IF
END FUNCTION

FUNCTION loop()
    lue sarjaporttia
    IF (sarjaporttiin tulee merkkijono "cal") THEN
         boolean kalibroi = true
    END IF
    IF (kalibroi = true) THEN
      WHILE (sarjaporttiin ei tule dataa) THEN
         lue pH-arvo anturista
         lähetä pH-arvo sarjaporttiin
      END WHILE
      IF (sarjaporttiin tulee merkkijono) "stop" THEN
         kalibroi = false
         break
      ELSE
         lue sarjaportista tuleva merkkijono
         pilko merkkijono erillisiksi muuttujiksi
         kalibroi pH-anturi muuttujien arvoilla
         tallenna muuttujat EEPROMiin
      END IF
    END IF
    IF (kalibroi = false) THEN
      jatka
    END IF
END FUNCTION
```

Kuva 9. Arduinon toimintalogiikka pseudokoodina

Sovelluksen ulkoasuun ei kiinnitetty paljoakaan huomioita, eikä ulkoasun suunnitteluun paneuduttu sen enempää. Ulkoasussa tärkeintä on, että se on siisti ja ymmärrettävä, mutta mitään käyttöliittymäsuunnitteluun liittyviä ohjeita ei otettu huomioon.

6.2 **Uuden ominaisuuden toteutus**

Olennaisin osa kalibrointiomaisuuden implementointia oli saada Arduino ja tietokone keskustelemaan keskenään. Tämä onnistui sarjaporttiliikenteen avulla tietokeen USB-portin kautta. Sarjaporttiyhteydellä lähetetään erilaisia merkkijonoja puolin ja toisin, ja Arduinon ohjelman suoritus riippuu pitkälti siitä, mitä arvoja se tietokoneesta vastaanottaa. Arduino suorittaa perusohjelmaansa, eli lukee antureiden sen hetkiset arvot ja lähettää ne tietokoneeseen, jos tietokone ei lähetä sarjaporttiin mitään arvoja.

Kun arvoja lähetetään, ohjelman suoritus muuttuu ja sitä lähdetään suorittamaan vastaanotetun arvon perusteella, eli joko muuttamaan tavoite-pH-arvoa tai kalibroimaan pH-antureita.

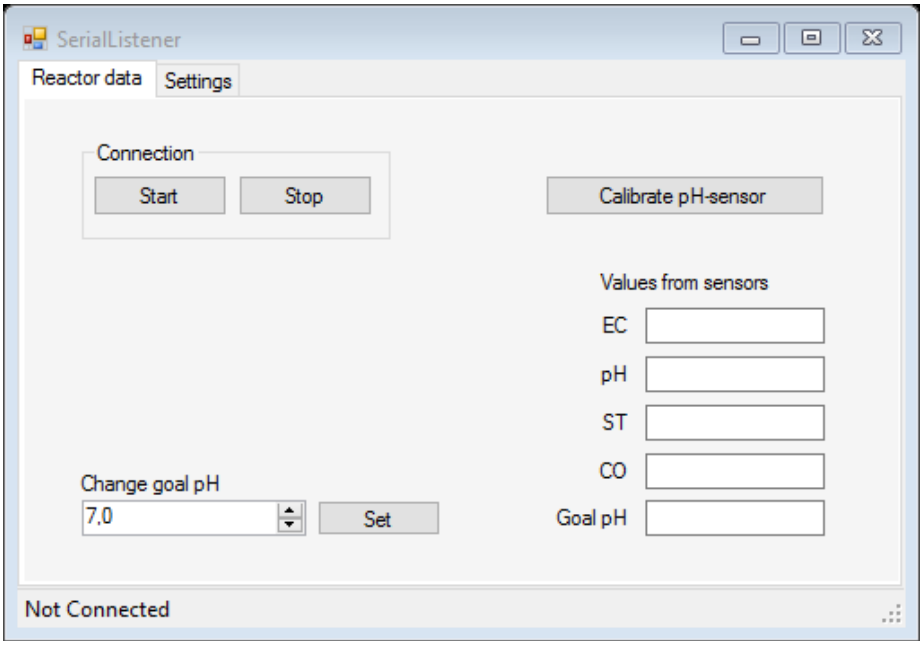
Alkuperäisessä työpöytäsovelluksessa oli vain yksi luokka, johon koko sovelluksen toiminta perustui. Sarjaliikenne oli alustettu tämän luokan sisälle eikä omassa luokassaan, mikä olisi ollut hyvä jatkokehitystä varten. Jotta sarjaliikenne saatiin toimimaan vaivattomasti myös uudessa kalibrointiluokassa, sarjaporttiliikenteelle tehtiin uusi singleton-luokka, jonka avulla sarjaporttiin kirjoittaminen ja sarjaporttiin tulevan datan lukeminen onnistui ongelmitta. Tätä singleton-luokkaa voi käyttää kaikkiin sovelluksiin, jotka käyttävät sarjaliikennettä (Heisway Nrird 2016).

Singleton on luokka, joka sallii vain yhden instanssin luomisen itsestään (C# in Depth n.d.) ja siihen pääsee mistä tahansa helposti käsiksi. Singleton-luokan käsittelemiseen käytetään vain yhtä ja samaa oliota ja se voidaan jakaa eri luokkien kesken. Samaa sarjayhteyttä voi käyttää vain yksi säie samanaikaisesti, joten sarjaportin lukemisessa päädytään ongelmiin, jos sarjayhteyttä yritetään lukea esimerkiksi monesta eri luokasta samaan aikaan. Tässä tapauksessa sarjayhteyden kanssa jouduttiin ongelmiin, koska kalibrointi-luokka ei päässyt käsiksi sarjaporttiin. Sarjaportin lukeminen oli vielä toisessa luokassa kesken ja ohjelma kaatui.

Jos Arduino vastaanottaa sarjaportista merkkijonon ”cal”, se lähtee suorittamaan ohjelman kalibrointiominaisuutta, eli lähettämään sarjaporttiin pH-anturista saamaansa arvoa, niin kauan, kunnes se vastaanottaa joko kalibroitavat arvot tai kalibroinnin peruutusta vastaavan merkkijonon ”stop”. Kalibrointiarvot lähetetään sarjaporttiin yhtenä merkkijonona, jonka aloitus- ja lopetusmerkkinä toimivat ”<” ja ”>”. Kalibrointiarvojen erotusmerkkinä on ”;”. Merkkijono voisi olla esimerkiksi: ”<2345;2567;2890>”. Arduinon ohjelma pilkkoo arvot näitä aloitus-, lopetus-, ja erotusmerkkejä hyödyntäen, kalibroi pH-anturin saamillaan lukemilla ja tallentaa lukemat Arduinoon talteen.

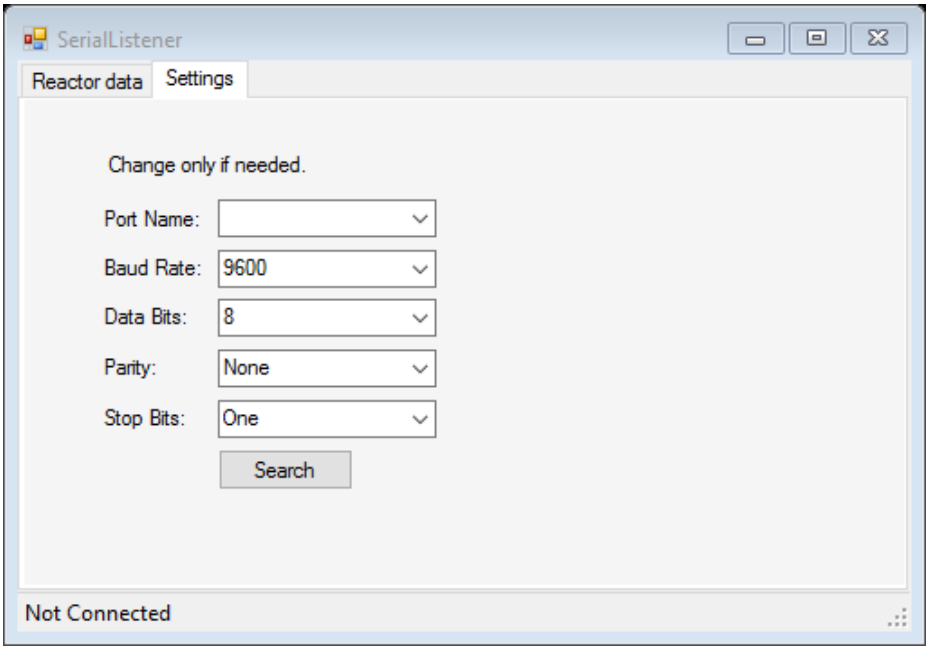
Työpöytäsovellus koki suuren muodonmuutoksen, koska opinnäytetyön tavoitteena oli luoda selkeä ja yksinkertainen tapa kalibroida leväreaktorin pH-anturit. Alkuperäisen työpöytäsovelluksen ulkoasu oli hieman monimutkainen, joten sitä oli hyvä yksinkertaistaa ja selkeyttää. Ulkoasua voisi kuitenkin vieläkin hioa enemmän. Uusi työpöytäsovellus koostuu kolmesta eri ikkunanäkymästä: perusnäkymästä (Kuva 8), asetusnäkymästä (Kuva 9) ja kalibrointinäkymästä (Kuva 11).

Työpöytäsovelluksen perusnäkymä (Kuva 8) tulee esiin, kun ohjelma käynnistetään. Perusnäkymässä avataan sarjaporttiyhteys Arduinoon, mikä on työpöytäsovelluksen tärkein vaihe. Ilman yhteyttä sovellus ei toimi. Kun yhteys on luotu, työpöytäsovellus ryhtyy lukemaan sarjaportista tulevaa dataa eli Arduinon lähettämiä merkkijonoja. Merkkijonot pilkotaan ja näytetään käyttäjälle, samoin kuin alkuperäisessä työpöytäsovelluksessa.

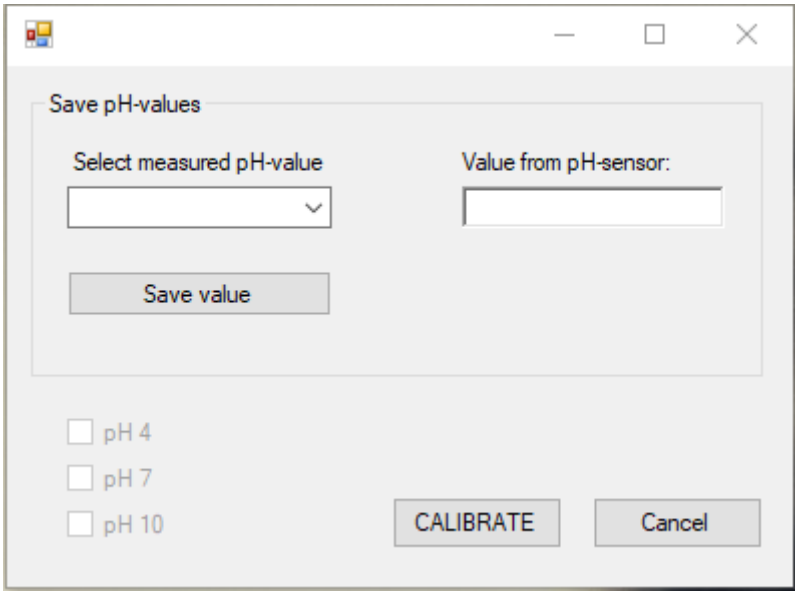


Kuva 10. Uuden työpöytäsovelluksen perusnäkymä.

Työpöytäsovelluksen asetusnäkymä (Kuva 9) siirrettiin omaan välilehteensä, koska asetuksiin ei tavallisesti tarvitse koskea. Ne ovat silti hyvä olla ohjelmassa mukana, jotta asetuksia voi yrittää säätää ongelmatilanteissa. Asetukset liittyvät sarjayhteyden muodostamiseen ja niillä säädetään sarjayhteyden portin nimeä, siirtonopeutta, lopetusbittiä, databittien määrä ja sarjayhteyden virheiden havainnointia, eli pariteettitarkastusta. Säädettäviä asetuksia tai asetuksien vaihtoehtoja ei lisätty tai poistettu.



Kuva 11. Uuden työpöytäsovelluksen asetusnäkymä.



Kuva 12. Uuden työpöytäsovelluksen kalibrointinäkymä.

Kalibrointinäkymässä (Kuva 10) Arduinon pH-anturista tulevat arvot näkyvät ”Value from pH-sensor” -tekstin alla laatikossa. Toinen lukemista on pH-anturin ilmoittama millivolttiarvo ja toinen millivolttiarvosta muunnettu pH-arvo. Pudotusvalikosta ”Select measured pH-value” tekstin alla valitaan kalibroitava kalibrointipiste; pH 4, pH 7 tai pH 10. Kalibrointipiste valitaan käytössä olevan kalibrointiliuoksen mukaan. Kun pH-anturista tuleva arvo tasaantuu ja kalibrointipiste halutaan tallentaa kalibroitavaksi, painetaan ”Save value” -nappia. Ohjelma laskee viiden viimeisen pH-anturin ilmoittaman lukeman keskiarvon ja tallentaa sen ohjelman muistiin, sekä piirtää valintamerkin kalibroitavan kalibrointipisteen kohdalle ikkunan vasempaan alareunaan. Sama toistetaan kaikilla kalibrointiliuoksilla. Kun kaikki kalibrointipisteet on tallennettu, voidaan kalibroida pH-anturi painamalla ”CALIBRATE” -nappia. Työpöytäsovellus lähettää tallennetut kalibrointipisteiden arvot Arduinoon sarjaliikenteen avulla ja Arduino pilkkoo saamansa arvot omiksi lukemikseen ja kalibroi pH-anturin, sekä tallentaa lukemat EEPROM -muistiin.

Parempaa käyttökokemusta silmällä pitäen työpöytäsovelluksen alalaitaan lisättiin teksti, joka kertoo, onko Arduino yhdistettynä vai ei. Arduino ja sarjaporttiyhteys on verrattain hidas reagoimaan muutoksiin ja vastaamaan pyyntöihin ja varsinkin ensimmäisen merkkijonon saapumisessa tietokoneeseen yhteyden luonnin jälkeen kestää kauan. On siis hyvä osoittaa käyttäjälle, että jotain on tapahtumassa, ja Arduino on onnistuneesti yhdistetty tietokoneeseen.

7 JATKOKEHITYS

Arduinon pH-anturin toiminnassa saattaa ilmetä ongelmia, jos anturin kalibroi väärillä arvoilla esimerkiksi niin, että käyttää pH-arvon 4 kalibrointipisteen kalibroimiseksi pH-arvon 7 kalibrointiliuosta. Jos näin tapahtuu, pH-anturi menee sekaisin ja sensori lukee liuoksien pH-arvot väärin tai ei ollenkaan. Tätä tilannetta on hankala korjata ilman ohjelmoijan apua, koska uudelleenkalibrointi työpöytäsovelluksen kalibrointiominaisuuden avulla ei onnistu väärin kalibroidulla pH-anturilla. Jos tilanne tulee eteen nyt, Arduinon EEPROM-muisti pitää tyhjentää erillisellä tyhjennyskoodilla ja ladata leväreaktorissa käytettävä koodi takaisin laitteeseen. Samalla laite käynnistyy uudelleen ja pH-anturi kalibroituu kovakoodatuilla oletusarvoilla, koska EEPROM-muisti on tyhjä.

Leväreaktoria jatkokehittäessä työpöytäsovellukseen kannattaisi tehdä sellainen ominaisuus, joka tyhjentäisi Arduinon EEPROM-muistin ja kalibroisi pH-anturit joillain suuntaa antavilla oletusarvoilla. Näin vältyttäisiin ongelmatilanteilta, joissa pH-anturi ei toimi tai ilmenee muita ongelmia. Työpöytäsovelluksessa voisi olla jokin nappi, jota painamalla sovellus lähettäisi Arduinoon sarjaliikenteen avulla käskyn siirtyä koodilohkoon, joka tyhjentäisi EEPROM-muistin tyystin (Kuva 7) ja kirjoittaisi päälle jotkin oletusarvot.

```
#include <EEPROM.h>

void setup() {
  // initialize the LED pin as an output.
  pinMode(13, OUTPUT);

  /***
    Iterate through each byte of the EEPROM storage.

    Larger AVR processors have larger EEPROM sizes, E.g:
    - Arduno Duemilanove: 512b EEPROM storage.
    - Arduino Uno:        1kb EEPROM storage.
    - Arduino Mega:       4kb EEPROM storage.

    Rather than hard-coding the length, you should use the pre-provided length function.
    This will make your code portable to all AVR processors.
  ***/

  for (int i = 0 ; i < EEPROM.length() ; i++) {
    EEPROM.write(i, 0);
  }

  // turn the LED on when we're done
  digitalWrite(13, HIGH);
}

void loop() {
  /** Empty loop. **/
}
```

Kuva 13. Koodi EEPROM-muistin tyhjennykseen.

Leväreaktorin LED-valot tuottavat paljon lämpöä, ja liian korkea lämpötila haittaa levän kasvua. Lämpötilan alentamiseksi olisi hyvä keksiä jokin kustannustehokas ratkaisu. Leväreaktori tulisi ensinnäkin sijoittaa hyvin ilmastoituun ja viileään huoneeseen, mikä jo osaltansa laskisi lämpötilaa. Olisi hyvä, jos LED-valot voisi korvata jollain vähemmän lämpöä tuottavilla valoilla tai kasvattaa vain sellaista levää, joka ei vaadi niin paljon valoa kasvuunsa. Karjalaisen (2015) opinnäytetyössä leväkasvatuksessa käytettiin *Chlorella protothecoides* -nimistä mikrolevää, joka on hyvin samankaltainen kuin Hämeen ammattikorkeakoulussa nyt kasvatettavat levät, mutta viihtyy hyvin myös pimeässä. Se käyttää kasvuunsa kasvatusliuoksessa olevia orgaanisia yhdisteitä, esimerkiksi sokeria, mutta tuottaa kasvaessaan sivutuotteena happamia aineksia, jolloin kasvatusliuoksen pH lähtee laskemaan ja levän kasvu häiriintyy. Tämä voitaisiin kuitenkin ratkaista samanlaisella järjestelmällä, jota käytetään pH-arvon nousun kontrolloimiseen hiilidioksidilla, mutta hiilidioksidin syöttämisen ilmasäiliöön sijasta Arduino syöttäisi emäksistä liuosta suoraan leväreaktorin kasvatussäiliöön. Ilman lämmittäviä LED-valoja kasvatusliuoksen lämpötilaa voisi säätää akvaariolämmittimellä (Karjalainen 2015) tietokoneen työpöytäsovelluksen kautta.

8 YHTEENVETO

Opinnäytetyön tavoite oli toteuttaa ja liittää leväreaktorin pH-antureiden kalibrointiominaisuus jo olemassa olevaan työpöytäsovellukseen. Ominaisuuden toteutus onnistui hyvin ja uudistettu työpöytäsovellus, sekä uusi Arduino-sketsi otettiin käyttöön Hämeen ammattikorkeakoulun biotalousyksikössä. Kalibrointiominaisuus valmistui annetuissa aikamääreissä. Opinnäytetyön raportti myöhästyi todella paljon, mutta varmasti kaikki asianosaiset oppivat tapahtuneesta jotain.

Arduino Uno -mikrokontrollerin ohjelmoimisessa on tärkeää muistaa miten mikrokontrolleri suorittaa sketsiään eli ohjelmaansa. *()setup*- ja *()loop*-funktiot ovat välttämätön osa Arduino-sketsiä ja ne on pakko olla jokaisessa sketsissä vaikka ne olisivatkin muuten tyhjiä. *()setup*-funktio suoritetaan vain kerran, kun ohjelman suoritus alkaa ja sen sisällä kannattaa alustaa käytettävät kirjastot tai muuttujat, jotka halutaan säilyttää sellaisenaan koko ohjelman suorituksen ajan. *()loop*-funktio on sketsin sydän ja eräänlainen silmukka, jota suoritetaan yhä uudelleen ja uudelleen. Koko Arduinon ohjelmalogiikka perustuu *()loop*-funktioon.

Automatisoimalla mikrokontrollerin avulla osa levänkasvatusprosessia, säästetään kustannuksissa ja saadaan tarkempaa tietoa levän kasvusta eri olosuhteissa ja kasvuun vaikuttavista elementeistä. Arduino lukee antureita muutaman sekunnin välein, ja tarpeeksi tarkoilla antureilla pienimmänkin muutoksen huomaa Arduinon keräämästä datasta helposti. Dataa on myös saatavilla paljon enemmän, kuin jos sitä kerättäisiin vain manuaalisesti. Koska kaikki data tallentuu pilveen digitaaliseen muotoon, siihen pääsee käsiksi mistä tahansa ja siitä on helppo tehdä erilaisia visualisointeja ja kaavioita.

Suurimmat ratkaistavat ongelmat työpöytäsovelluksen jatkokehityksessä olivat sarjaporttiyhteys ja merkkijonon manipulointi Arduinolla. Sarjaporttiyhteys oli tekijälle täysin vieras käsite, ja sen ominaisuuksien ymmärtämisessä kului monia työtunteja. Haastavaa oli myös Arduinon ja työpöytäsovelluksen toimintalogiikan yhtenäistäminen niin, ettei Arduino jäänyt esimerkiksi jumiin kalibrointisilmukkaan.

LÄHTEET

All about pH. (n.d.). pH calibration procedure. Haettu 29.2.2017 osoitteesta: http://www.all-about-ph.com/ph-calibration.html

Arduino. (n.d.). *EEPROM Library.* Haettu 28.2.2017 osoitteesta: https://www.arduino.cc/en/Reference/EEPROM

Arduino. (n.d.a). *EEPROM put().* Haettu 28.2.2017 osoitteesta: https://www.arduino.cc/en/Reference/EEPROMPut

Arduino. (n.d.b). *EEPROM get().* Haettu 28.2.2017 osoitteesta: https://www.arduino.cc/en/Reference/EEPROMGet

Arduino. (n.d.c). *Arduino Uno.* Haettu 29.2.2017 osoitteesta: http://www.arduino.org/products/boards/arduino-uno

Banzi, M. (2011). *Arduino – perusteista hallintaan*. 2. painos. Suom. Mäenpää, Y. Hämeenlinna: Robomaa.com.

Bayle, J. (2013). *C Programming for Arduino*. Olton, Iso-Britannia: Packt Publishing.

Cooking Hacks. (n.d.). *Open Garden-Hydroponics & Garden Plants Monitoring for Arduino*. Haettu 29.2.2017 osoitteesta: https://www.cooking-hacks.com/documentation/tutorials/open-garden-hydroponics-irrigation-system-sensors-plant-monitoring/#step3

C# in Depth. (n.d.). Implementing the Singleton Pattern in C#. Haettu 5.9.2017 osoitteesta: http://csharpindepth.com/Articles/General/Singleton.aspx

HAMK (n.d.). *HAMKIN HANKKEITA. LEVARBIO-LEVIEN ARVOKKAIDEN BIOMOLEKYYLIEN JA BIOMASSAN HYÖDYNTÄMINEN RAVINNOSSA, REHUNA JA ENERGIANA*. Haettu 26.1.2017 osoitteesta: http://www.hamk.fi/tyoelamalle/hankkeet/Sivut/default.aspx?RepoProject=7727422

Heikkilä, V. (2016). *Summer project PhotoBioreactor*. YouTube-video 18.8.2016. Haettu 19.1.2017 osoitteesta: https://www.youtube.com/watch?v=56_bO1aeExY

Heisway Nrird. (2016). *C#-Simple SerialPort singleton class.* Haettu 16.2.2017 osoitteesta: https://heiswayi.nrird.com/2016/csharp-simple-serialport-singleton-class

Jaarinen, S. & Niiranen, J. (2005). *Laboratorion analyysitekniikka.* Helsinki: Edita Prima Oy.

Jääskeläinen, O. (2016). Luottokortin kokoiset monitaiturit. *Mikrobitti* 8/2016, 14-27.

Karjalainen, J. (2015). *Leväpolttoaine-kasvatusreaktorin automatisointi: testireaktorin elinolosuhteiden ylläpito.* Opinnäytetyö. Elektroniikka. Turun ammattikorkeakoulu. Haettu 26.7.2017 osoitteesta: http://urn.fi/URN:NBN:fi:amk-201505259919

Langbridge, J. (2015). *Arduino Sketches.* Yhdysvallat: John Wiley & Sons, Incorporated.

Lehtonen, P. & Sihvonen, M. (2004). *Laboratorioalan analyyttinen kemia*. 1. painos. Helsinki: Edita Prima Oy.

Neste Oil. (2014). *Levääöjy on lupaava uusiutuvan dieselin raaka-aine – Nesto Oil varmistaa saantiaan ehdollisilla ostosopimuksilla.* Haettu 26.7.2017 osoitteesta https://www.neste.com/fi/fi/lev%C3%A4%C3%B6ljy-lupaava-uusiutuvan-dieselin-raaka-aine-neste-oil-varmistaa-saantiaan-ehdollisilla

Nussey, J. (2013). *Arduino For Dummies*. Somerset, Iso-Britannia: For Dummies.

O'Sullivan, D. & Igoe, T. (2004). *Physical Computing: Sensing and Controlling the Physical World with Computers*. Boston: Thomson Course Technology PTR.

Purdum, J. (2012). *Beginning C for Arduino: Learn C Programming for the Arduino and Compatible Microcontrollers*. Yhdysvallat: Apress.

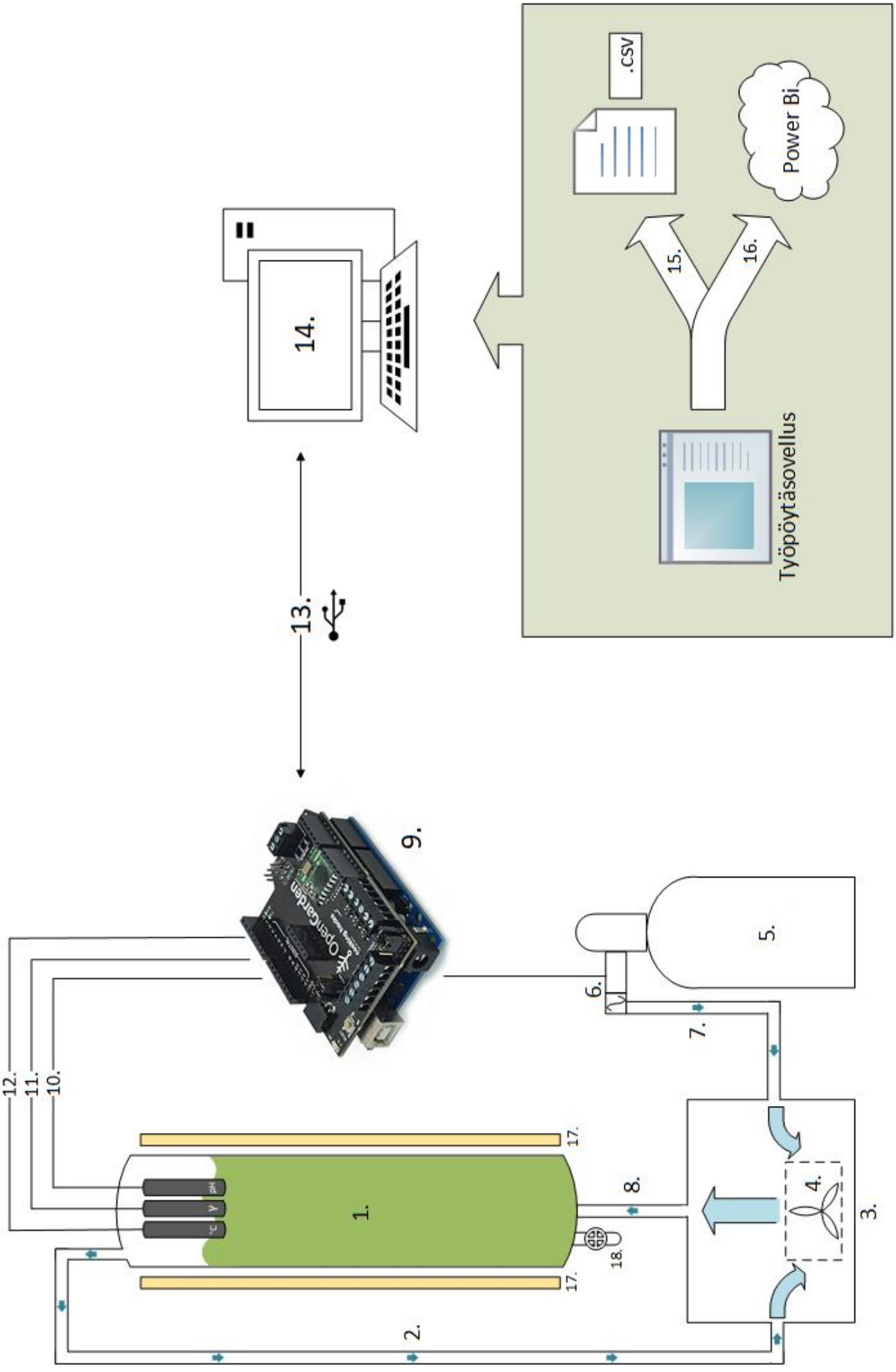
HAASTATTELUT

Nokkonen, S. (2017). Projektityöntekijä, HAMK. Haastattelu 24.1.2017, Hämeenlinna.

Saarinen, J. (2017). Teknologia-assistentti, HAMK Älykkäät Palvelut. Haastattelu 27.1.2017, Hämeenlinna.

Liite 1

LEVÄREAKTORIN TEKNINEN ARKKITEHTUURI



LIITE 1

Kuvan selitteet:

1. Leväreaktorin 30 litran kasvatussäiliö, jossa ovat kasvatusliuos ja levä.
2. Putki kasvatussäiliön yläosasta ilmasäiliöön, jota pitkin ilma ja veteen liukenematon hiilidioksidi pääsee kulkemaan. Ilman kulkusuunta on esitetty nuolin.
3. Ilmasäiliö, jossa hiilidioksidi ja tavallinen huoneilma sekoittuvat.
4. Ilmapumppu, joka pumppaa ilmaseoksen kasvatusliuokseen.
5. Hiilidioksidipullo.
6. Rele, joka Arduinon ohjaamana syöttää hiilidioksidia hiilidioksidipullosta ilmasäiliöön.
7. Putki, jota pitkin hiilidioksidi kulkee ilmasäiliöön.
8. Putki, jota pitkin ilmaseos kulkee kasvatussäiliöön.
9. Arduino Uno -mikrokontrolleri ja Open Garden -lisäpiirikortti. Säätelee kasvatusliuoksen pH-tasapainoa ja lähettää antureista saatavaa dataa USB-johdon kautta tietokoneeseen.
10. pH-anturi. Mittaa kasvatusliuoksen pH-arvoa.
11. Sähkönjohtavuus-anturi. Mittaa kasvatusliuoksen sähkönjohtavuutta.
12. Lämpöanturi. Mittaa kasvatusliuoksen lämpötilaa.
13. USB-kaapeli A-B Arduinon ja tietokoneen välillä.
14. Tietokone, jossa pyörii työpöytäsovellus, jonka avulla voidaan säädellä kasvatusliuoksen pH-arvon ylärajaa. Kerää myös Arduinon antureista sarjaliikenteen avulla saatavaa dataa.
15. Työpöytäsovellus tallentaa antureista saatavan datan paikallisesti .csv-tiedostoon.
16. Työpöytäsovellus lähettää HTTP POST -pyyntönä antureista saatavan datan pilveen Power Bi-datavisualisointi-ohjelmaan.
17. LED-valot. Päällä ajastimella 16 tuntia vuorokaudesta.
18. Venttiili, josta kasvatussäiliö tyhjennetään.