Nataliya Bauer

Kamu urbaaniviljelijän apulaisena

Metropolia Ammattikorkeakoulu
Insinööri (AMK)
Sähkö- ja automaatiotekniikka
Insinöörityö
1.12.2022

Tiivistelmä

Tekijä:	Nataliya Bauer
Otsikko:	KaMu urbaaniviljelijän apulaisena
Sivumäärä:	18 sivua + 1 liite
Aika:	1.12.2022
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Sähkö- ja automaatiotekniikka
Ammatillinen pääaine:	Elektroniikka
Ohjaajat:	Ohjaava opettaja Heikki Valmu Työnohjaaja Raine Hermans

---

KaMu on kutsumanimi Aeropod Oy:n aeroponiselle kasvatusmoduulille, jossa kasvatetaan mukulakasveja. KaMun ohjaustapa on ollut tähän saakka hyvin yksinkertainen ja yksipuolinen, joten Aeropod lähti etsimään ratkaisuja sen parantamiseen. Tässä opinnäytetyössä on suunniteltu ja rakennettu web-pohjainen automaatio-ohjaus, jonka tavoitteena on olla viljelijän tukena viljelyssä.

Ohjattavina laitteina ovat vesipumppu ja kasvilamput, ja ohjaus perustuu enimmäkseen käyttäjän määräämiin asetusarvoihin. Tärkeänä osana ovat myös anturit, joista osa on tarkoitettu datan keräystä varten ja osa tunnistamaan vikatilanteet ja huollon tarve Järjestelmään on myös suunniteltu autonominen vikatila, joka lähettää hälytystä käyttäjälle ja itsenäisesti sulkee osittain tai kokonaan systeemin ilman käyttäjän ohjausta.

Päämääränä on rakentaa kevyt- ja helppokäyttöinen ohjaus sekä sen lisäksi suunnitella varapiiri, joka korvaa pääjärjestelmän ongelmatilanteissa. Varapiiri on pääosin analoginen, eikä sisällä etäohjausta ja -luentaa, mutta on myös autonominen sekä ohjattavissa manuaalisesti käsin. Sitä voi käyttää erikoisolosuhteissa tai pääjärjestelmän rikkoutuessa. Pääjärjestelmän ytimenä on CONTROLLINO MAXI, joka on Arduino-pohjainen relemoduuli. Moduulissa on sisäinen kello sekä Ethernet-liitäntä, mikä on välttämätöntä kasvatussysteemin toiminnalle. Koodikielenä on C ja C++.

Piirien suunnittelussa referensseinä toimivat nykyiset ammatti- ja harrastajakäyttöön tarkoitetut järjestelmät, sekä aiemmin innovaatioprojektissa suunniteltu demoversio Aeropodin järjestelmälle. Opinnäytetyön lopputuloksena on toimiva demoversio, jota kehitellään lisää ennen asennusta järjestelmään. Kuitenkin kaikki oleelliset toiminnallisuudet on testattu toimiviksi.

Avainsanat: aeroponiikka, anturi, CONTROLLINO, mqtt, node-red

Abstract

Author:	Nataliya Bauer
Title:	KaMu as urban farmer assistant
Number of Pages:	18 pages + 1 appendix
Date:	1 December 2022
Degree:	Bachelor of Engineering
Degree Programme:	Electrical and Automation Engineering
Professional Major:	Electronics
Supervisors:	Heikki Valmu, Senior Lecturer Raine Hermans, Project manager

---

KaMu is a cultivation module by Aeropod Oy, used in aeroponic cultivation of potatoes. So far controlling KaMu has been very simple and crude, so Aeropod wanted to improve the functionality of the design. The purpose of the thesis work was to design and build the automation for the aeroponic cultivation system, system control and monitoring of conditions with a web-based user interface. The main goals were to design easy and functional system control, alarms in case of disturbances, and data collection and reading from sensors.

The devices that need controlling are water pumps and greenhouse lights. Controlling is based on values specified by the user. Some of sensors collect data and some are used to notice alarms and the need for maintenance. System for autonomous alarm sends a notification to the user and automatically and shuts down the system partially or completely without any user input. The aim was to design a system that is a lightweight and easy-to-use control system, and a backup circuit that replaces the main system when it breaks down.

CONTROLLINO MAXI operates system and has an internal clock, as well as an Ethernet interface, which is necessary for the operation of the growing system. The code language was C and C++. In the circuit design, the references were the current systems for professional and amateur use, as well as the demo version of the Aeropod system previously designed in an innovation project. The result of the thesis work is a functional demo version, which is developed further before installing it into the system.

Keywords: aeroponic, sensor, CONTROLLINO, mqtt, node-red

Sisällys

Lyhenteet

DIN: *Deutsches Institut für Normung*, standardoidu kisko.

IDE: *Integrated Development Environment*. Ohjelma, jossa voi kirjoittaa laiteohjelmiston koodia.

I^2C: *Inter-Integrated Circuit,* tiedonsiirtoväylä.

KaMu: Kasvatusmoduulin lyhenteestä muotoiltu nimitys.

MQTT: *Message Queuing Telemetry Transport*, laitteiden välisen kommunikaation protokolla.

PID: *Proportional-Integral-Derivative*, vahvistimilla rakennettu suljetun silmukan ohjaus.

RTC: *Real Time Clock*. Reaaliaikainen kello, johon sisältyy useampaan digitaaliseen kehitysalustaan kuten Photon, ESP32 tai CONTROLLINO.

SPNa: Suurpainenatriumlamppu.

SPI: *Serial Peripheral Interface*, sarjamuotoinen oheislaiteväylä.

WIFI: *Wireless Local Area Network*, langaton lähiverkko.

1 Johdanto

Aeropod Oy on startup-yritys, joka on lähtenyt tutkimaan aeroponiikkaa eli ilmaviljelyä yksittäisen kasvatusmoduulin kehityksen myötä. Siinä kasvatetaan pääosin perunaa, mutta on tehty myös koekasvatusta bataatilla. Moduulin nimeksi on annettu KaMu (lyhenteenä kasvatusmoduulista) ja siihen sisältyy iso arkku, johon istutetaan taimet ja joka toimii samalla vesisäiliönä, puukehikko, johon asennetaan valaisimet ja kasvien tukirakenteet, sekä kastelujärjestelmä ja sen ohjaus. Tähän asti kastelujärjestelmän ja valaistuksen ohjaaminen on perustunut paikalliseen analogiseen ohjaukseen. Tilanneseuranta on ollut manuaalista ja perustunut viljelijän omiin merkintöihin. Kasvattajat eivät ole saaneet automaattisesti kerättyä informaatiota kasvikauden olosuhteista tai ajankohtaista tietoa laitteiston kunnosta, mikä on erittäin tärkeää vikatilanteiden ennakoimiseksi.

Yksi tärkeimmistä Aeropodin kehityssuunnista on KaMun monipuolistaminen automatisoinnin myötä. Monipuolisuus tulee näkyväksi järjestelmän helppokäyttöisyydellä, tiedonkeräyksellä sekä ongelmatilanteiden havaitsemisella ja ennakoimisella. Tämä auttaa luomaan tuotteesta erittäin yhtenäisen ja viimeistellyn, joten käyttäjän ei tarvitse huolehtia mistään ylimääräisestä saadakseen haluttuja tuloksia.

Viimevuotisessa innovaatioprojektissa oli aloitettu suunnittelemaan ja toteuttamaan kasvatusjärjestelmän automatisointia, joten siinä luotu vaatimusmäärittely sekä osa ratkaisuista on hyödynnetty tässä opinnäytetyössä. Muutokset koskevat muun muassa ohjaustapaa sekä käyttöliittymää, sillä opinnäytetyössä keskitytään järjestelmän vakauteen ja käyttäjäystävällisyyteen, kun taas innovaatioprojektin päämääränä oli luoda yhtenäinen systeemi. Yhtenäisyyden tavoittelu jatkuu nykyisessä työssä, ja sen rinnalle toisena tärkeänä ominaisuutena tulee helpon ja nopean skaalauksen mahdollistaminen.

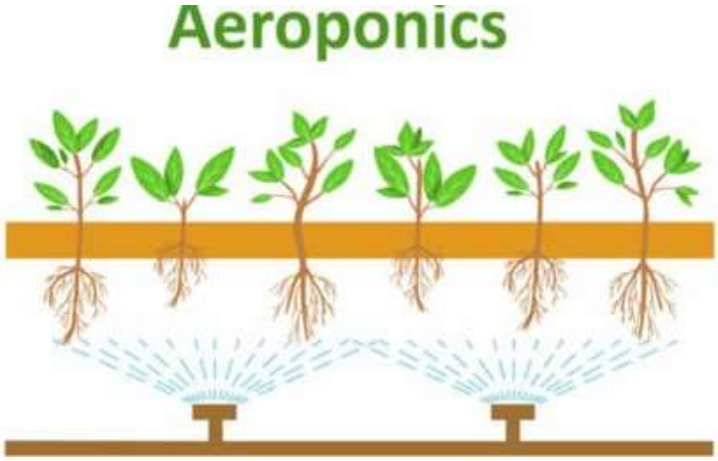
2 Vaatimusmäärittely ja aiemmat ratkaisut

Tässä luvussa käsitellään aeroponiikan perusperiaatetta ja vaatimuksia toimivaan aeroponisen systeemiin. Yleiskatsauksen lisäksi tarkastellaan olemassa olevia yritysratkaisuja eri puolilta maailmaa niiden järjestelmiä, käyttötarkoituksia sekä perusteluja aeroponiikan käytölle. Lopuksi kuvataan hieman tarkemmin innovaatioprojektia, sen ratkaisujen vahvoja ja heikkoja puolia.

2.1 Aeroponiikka

Aeroponinen viljely on yksi urbaaneista viljelykeinoista hydroponisen ja akvaponisen viljelyn rinnalla. Hydroponiikka perustuu pitkälti vesikiertoon, ja akvaponiikassa yhdistetään sekä kasvi- että kalaviljelyä. Kaikkia järjestelmiä yhdistää se, että ne ovat eristyneitä luonnollisista kasvutekijöistä täysin tai osittain. Jos hydroponiikkaviljelyssä voi valita kasvualustaksi luonnollisen materiaalin, turpeen, tai täysin keinotekoisen, esimerkiksi perliitin, niin aeroponiikassa ei ole ollenkaan kiinteää kasvualustaa.

Aeroponisia järjestelmiä on paljon erinäköisiä, mutta perusperiaate pysyy pitkälti samana kasvualustaa, voidaan jakaa ulko- ja sisäpuoleen ja raja vedetään kasvin ulkoisen osan ja juurten mukaan. Kasvi sijoitetaan alustaan, jossa sen joko ylä- tai keskiosaa tuetaan, ja juuriosa piilotetaan valolta ja liialliselta lämmöltä suljettuun moduuliin, joka voi toimia samalla vesisäiliönä. Säiliön sisällä juuret saavat roikkua vapaasti ilmassa, ja tästä tuleekin viljelytavan nimitys. Ravinne toimitetaan juurille yleisemmin kasvuliuosta sumuttamalla tai harvemmin ultraäänikostuttimen avulla (kuva 1).



Kuva 1. Aeroponiikan perusrakenne [1].

Aeroponiikka parhaimmillaan voi olla täysin riippumaton luonnollisista tekijöistä, mikä selittyy sillä, että menetelmä oli kehitelty jo vuonna 1980 NASAn toimesta, kun etsittiin avaruusalukseen sopivaa viljelykeinoa, ja luopuminen maaperän käytöstä sekä pieni vedenkulutus olivat keskeisimpiä tavoitteita.

2.2 Erilaiset järjestelmät ja niiden ohjaustavat

Aeroroots

Aeroroot sijaitsee Nepalissa ja on muutaman vuoden vanha. Aeroponiikan perusyksikkönä on tynnyri, jossa on kuusikymmentä reikää kasveja varten. Tämän lisäksi yrityksellä on muita aeroponisia ratkaisuja, ja kaikissa viljellään yrttejä, salaatteja tai muita pieniä ruokakasveja. [2.]

Aeroroots korostaa aeroponiikan hyötyjä vertaamalla sitä klassiseen viljelyyn. Tärkeintä yrityksen kannalta on säästyminen hyönteiskarkoitteilta, mikä tosiaan on ollut klassisen viljelyn suuri ongelma sen myrkyllisyyden takia. Aeroroots tarjoaa ongelmaratkaisijaksi aeroponiikkaa sillä perusteella, että suljettu kasvualusta ennakoivasti estää tuhohyönteisten tuloa. [2.]

LettUsGrow

LettUsGrow on britannialainen yritys, joka on perustettu vuonna 2015, yhdistää aeroponiikan ja vertikaaliviljelyn yrttien ja salaattien kasvatuksessa. LettUsGrow myös mainitsee hyönteiskarkoitteiden ongelmaan ratkaisun menetelmällään sekä sen lisäksi muita hyötyjä. Yrityksen visiona on ekologinen lähiruoka, jonka kasvatustapa mahdollistaa helpon ja näppärän kasvatusalustan skaalauksen tarvittaessa. Parempi vedenkulutuksen ja sisällön säätely vähentää veden saastumista. [3.]

Aerobloom

Aerobloom on pohjoisamerikkalainen yritys, joka nostaa omien aeroponisten järjestelmiensä tasoa monimutkaisella ja älykkäällä teknologialla. Osana järjestelmää ovat lukuisat anturit ja niiden keräämän tiedon perusteella toimivat algoritmit, jotka luovat tehokkaan kasvuprosessin sekä havaitsevat ongelmatilanteet ja parhaimmillaan myös ratkaisevat niitä. Aerobloomille aeroponiikka voittaa sekä klassisen että hydroponisen viljelyn pienemmän vedenkulutuksen ansiosta, minkä mahdollistaa tehokas mutta tarkasti ajastettu ravinneliuoksen sumutus juuristoon. Yritys myös pitää tärkeänä omien kasvualustojensa helpon ja edullisen skaalauksen mahdollisuutta. [4.]

Aeropod

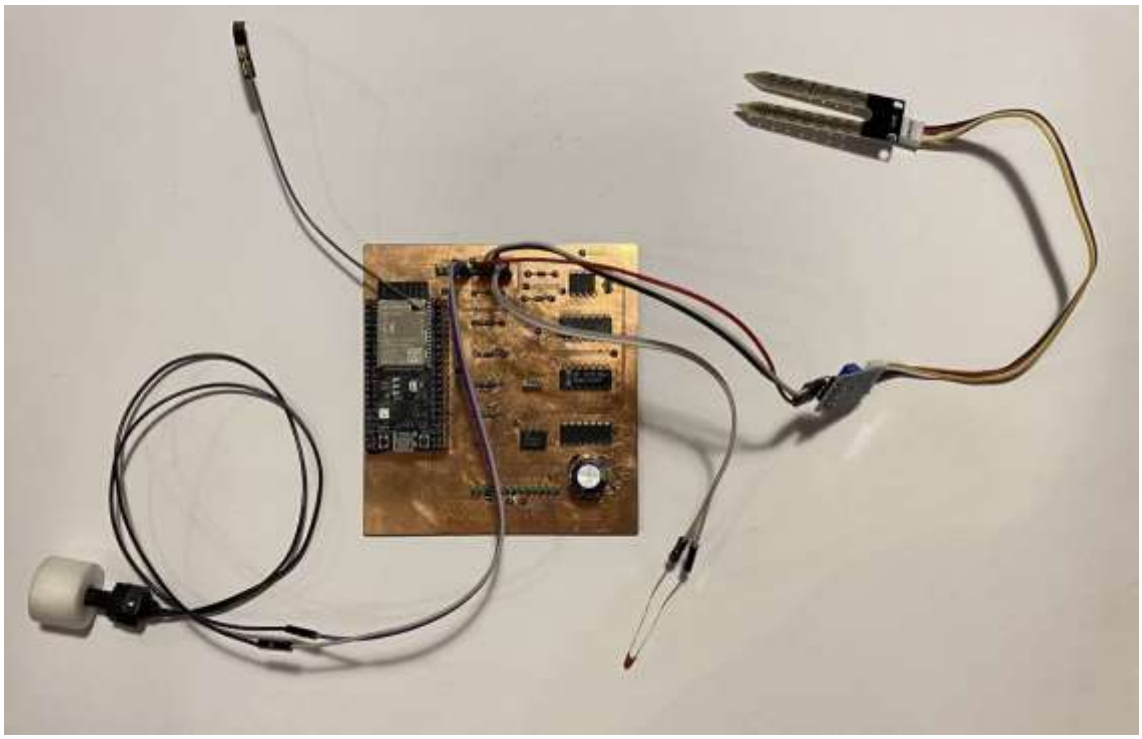
Aeropod on suomalainen startup-yritys, joka kasvattaa aeroponisesti mukulakasveja, mikä on haastavampaa kuin yrttien tai salaattien kasvatus lähinnä perunakasvin rakenteen takia. Esimerkiksi toisin kuin klassisesti viljellyllä perunalla on paksu runko ja yläosa saa maata vapaasti maan päällä, aeroponisen perunakasvin varsi on paljon ohuempi ja tarvitsee tukea, jonka saatuaan se pyrkii kasvamaan ylös niin paljon kuin suinkin pääsee, kasvu loppuu lähinnä tukirakenteiden ja katon korkeuden rajoittamana, minkä latvoja joudutaan ohjaamaan kasvamaan sivulle. Myös arkun sisällä kasvava juuristo vaatii runsaasti tilaa pystysuunnassa, sillä ilmassa roikkuvia juuria vetovoima vetää suoraan alaspäin, ja pohjalla on ravinneliuos, johon juuret eivät saa osua. Nämä kaikki kasvuominaisuudet vaativat tarpeeksi korkean tukikehikon sekä syvän arkun, joten

järjestelmästä tulee kokonaisuudessaan suhteellisen kookas. Yrityksen käyttämä kasvatusmoduuli on samankaltainen, mitä käyttää Suomen siemenperunakeskus omien puhtaiden siemenperunoidensa kasvatukseen, joten juuri tämä ratkaisu on osoittanut toimivuutensa.

2.3 Innovaatioprojekti

Innovaatioprojektissa rakennettiin ensimmäinen demoversio KaMusta ja siihen sisältyi anturiluenta, laitteiston ohjaus ja hälytysjärjestelmä. Mikroprosessorin roolissa toimi alun perin Particle Photon alusta, jonka ympärille suunniteltiin koko ohjaus, mutta myöhemmin se oli korvattu samankaltaisella alustalla ESP32-S2-SAOLA, joka oli hyvä valinta siihen asennettuun RTC-kellon ja WIFI-moduulin ansiota (kuva 2). Tarkkaa kellonaikaa tarvittiin valaisinohjaukseen, ja WIFI-moduulilla sai tiedot kulkemaan käyttäjälle myös tämän ollessa pois järjestelmän luota.

Käytössä olleet anturit olivat pitkälti samat kuin nykyisessä opinnäytetyössä, joten oikeastaan suurin ero molempien projektien välillä on opinnäytetyön raskas koodipainotteisuus, sekä se, että opinnäytetyössä käytetyssä CONTROLLINO MAXI -ohjaimessa on sisäänrakennettuna releet ja ESP32 tarvitsee ulkoiset releet verkkovirtalaitteiden ohjaamiseen.



Kuva 2. Innovaatioprojektin piirilevy.

Opinnäytetyötä hieman helpotti se, että tarvittava vaatimusmäärittely automatisoinnille oli tehty jo innovaatioprojektin aikana. Aiemmassa projektissa pääsi myös kokeilemaan rohkeasti erilaisia ratkaisuja, jotka eivät välttämättä olleet tehokkaimpia, joten opinnäytetyössä voi keskittyä lopputuloksen turvallisuuteen ja tehokkuuteen tuhlaamatta aikaa epäkäytännöllisiin toteutustapoihin.

2.4 KaMu-järjestelmän käyttötarkoitus

KaMu asettuu kasvatusalustan ja viljelijän väliin ja yhdistää itsessään ohjauksen, tilanneseurannan ja viestittelyn käyttäjän kanssa, eli se on käytännössä kasvatusalustan aivot ja hermosto. Aivoina toimii mikroprosessori ja hermostona useat anturit, ja ongelmatilanteissa systeemi osaa lähettää avunpyynnön huoltajalleen. Nämä ominaisuudet helpottavat viljelijän työskentelyä huomattavasti, ja on helpompaa tehdä kehityssuunnittelua runsaan kerätyn tiedon pohjalta.

Järjestelmä on myös helposti skaalattavissa ja muokkaantuu eri tarpeisiin, mikä on tärkeää etenkin kasvaville yrityksille, jotka aikovat laajentaa viljelyään. Järjestelmän helppo muokkaaminen helpottaa uudistuksien tekemisessä.

2.5 Käyttäjäystävällisyys

Urbaani viljely on lähtökohtaisesti haastavaa, etenkin aloittelijalle, jolla kokemuksen sijaan on vain innokkuutta oppia kasvattamaan itsenäisesti ruokaa itselleen. Liian yksinkertaiset tai toisaalta liian monimutkaiset ja raskaat ohjaus- ja automaatiojärjestelmät vaikeuttavat viljelyä, ja virheiden määrä lisääntyy. Ohjauksen ollessa liian yksinkertainen käyttäjä joutuu pitämään huolta hyvin monesta asiasta, kuten veden tason seuraamisesta tai lämpötilan korkeudesta, mikä vie huomion pois itse kasvin hoidosta.

Monimutkaisemmassa systeemissä, jossa lähtökohtaisesti on paljon erilaisia ominaisuuksia, kasvaa käyttäjävirheen todennäköisyys. Ylimääräiset ominaisuudet voivat myös häiritä ohjaamista ja dataseurantaa. Hyvä käyttöliittymä asettuu keskelle; se on tarpeeksi yksinkertainen ja informatiivinen kokemattomalle käyttäjälle, mutta siinä on mahdollisuus laajentumiseen ja monipuolistumiseen, jos käyttäjälle tulee tarvetta siihen, yhden kasvualustan laajentuessa kasvihuoneeksi.

Tämän opinnäytetyön käyttöliittymä on rakennettu Node-Red-nimisellä kehitysalustalla. Yksinkertaisuuteen voi päästä helposti, sillä käyttöliittymää rakennetaan pieni pala kerralla ja toiminnallisuuden voi rajoittaa vain muutamaan ominaisuuteen.

3 Laitteet, varusteet ja metodit

Opinnäytetyössä on käytetty useita eri laitteita ja menetelmiä, joten tässä luvussa kuvataan niitä kaikkia tarkemmin myös antamalla teknisiä yksityiskohtia.

Päämääränä on antaa yleiskuva siitä, millä työkaluilla on työskennelty, jotta tulevaa jatkokehitystä tekevät asiantuntijat omaavat tarpeeksi tietoa siitä järjestelmästä, mitä he ovat lähdössä kehittämään.

Tärkeintä on löytää ohjauslaite, joka kykenee ohjaamaan kasvatusalustan laitteistoa. Yhteensopivuus saavutetaan ottamalla huomioon teknisiä piirteitä sekä laitteiden alkuperäistä käyttötarkoitusta.

3.1 Ohjattava laitteisto

KaMun tärkeimpiä prosesseja ovat sumutus ja valaistus. Sumutusta varten on rakennettu sumutusjärjestelmä, joka sijoittuu arkun sisälle, ja siihen sisältyy pumppu, putkisto ja suuttimet. Sumutusta ohjaamalla ohjataan siis suoraan pumpun toimintaa kytkemällä se päälle ja pois tietyin väliajoin. Valon riittävää määrää ylläpitävät kasvilamput, ja niiden ohjaaminen on aikaan sidottua, sillä kasveille yritetään luoda luonnollinen päivä- ja yösykli.

Vesipumppu

Nykyisessä järjestelmässä on käytössä kalvopumppu (kuva 3), jonka tehokkuus on kolmesataa wattia ja valmistaja on Elpumps [5].

Kuva 3. Kalvopumppu VP300 [5].

Pumppu on tarkoitettu kaivoihin, joten sen nostokorkeus on useita kymmeniä metrejä. Tämä on hyödynnetty aeroponiikassa, sillä perunan juuristo tarvitsee erittäin hienoa vesisumua, ja mitä korkeampi paine, sitä hienompaa sumua tulee suuttimista. Pumpun heikkoina puolina voi pitää suurta äänekkyyttä ja huonoa soveltumista ravinneliuoksen pumppaamiselle.

Valaisimet

Aeropod käyttää järjestelmissään kahdenlaisia valaisimia, LED-kasvilamppuja ja suurpainenatriumlamppuja (SPNa-lamput). Sekä LED-lampuilla että natriumlampuilla on vahvat ja heikot puolensa. LED-valaisimet eivät lämmitä ilmaa liikaa, mikä on tärkeää optimaalisen ilmalämpötilan ylläpitoa varten, kun taas SPNa-lamput kuumenevat ja kuumentavat itseään ympäröivää ilmaa. Toisaalta SPNa-lamput ovat edullisen hintaan nähden erittäin tehokkaita, kun taas yhtä tehokas LED-valaisin on huomattavasti kalliimpi. Perunakasvi vaatii tehokkaampaa valoa mitä esimerkiksi yrtit ja salaatit, joten natriumlamppujen käytössä on siinä tapauksessa järkeä, kun haluaa saada tehokkuutta pienellä budjetilla. SPNa-lamppujen tehot ovat yleensä 200–800 wattia, ja tärkeänä teknisenä

ominaisuutena on se, että ne eivät kestä useaa peräkkäistä päälle- ja poiskytkentää. Tämä on siis otettava huomioon ohjauksessa.

Muut laitteet

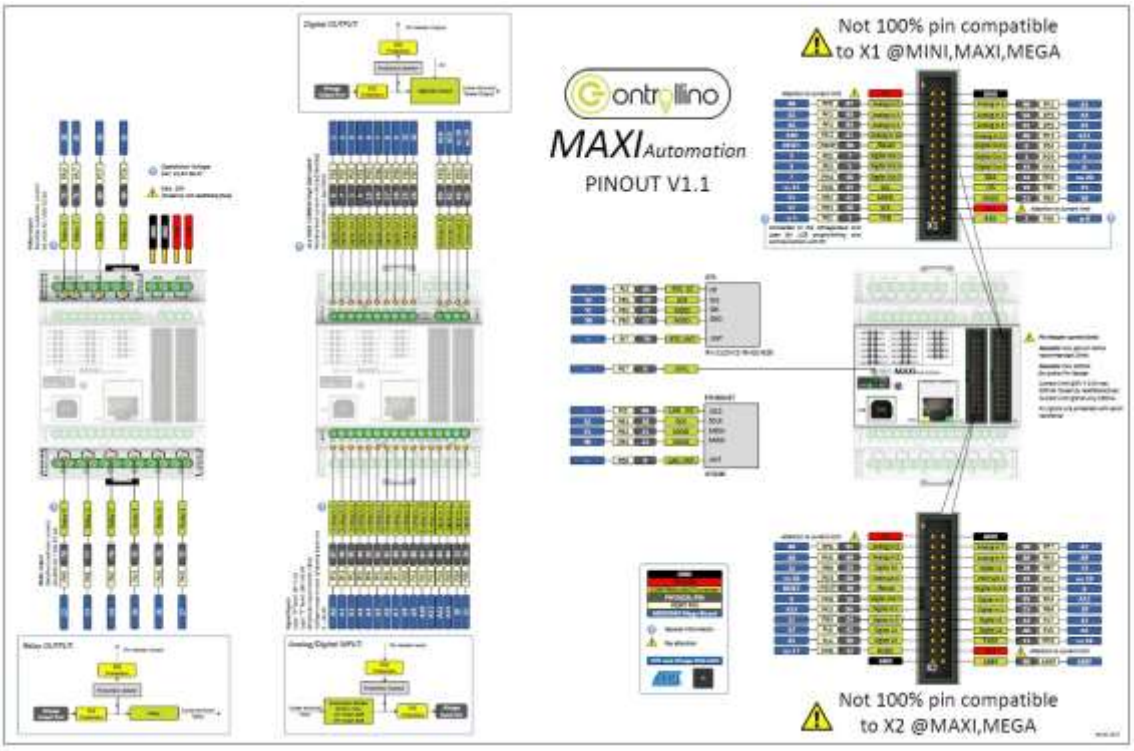
Tällä hetkellä päälaitteiden rinnalla on käytössä ilmapumppu sekä tuulettimet. Ilmapumpun tarkoituksena on välittää veteen happea, eikä siihen tarvita erillistä ohjausta, sillä pumpun on hyvä olla päällä aina kun koko systeemi on päällä. Tuulettimilla yritetään ylläpitää tasaista ulkolämpötilaa, joten ne tarvitsevat asetusarvon ja anturiluentaan perustuvan PID-ohjauksen, joka ei kuulu tähän opinnäytetyöhön mutta on tarkoitus suunnitella tulevaisuudessa.

3.2 Ohjausyksikkö

Ohjausalustan valinta on ollut riippuvainen useasta muuttujasta, mutta tärkeimpinä ovat WIFI/Ethernet ja RTC ominaisuudet. CONTROLLINO MAXI:lla, joka lopulta tuli valituksi, oli myös yksi tärkeä etulyöntiasema verrattuna muihin alustoihin: moduuliin kuului useampi rele, joilla voi ohjata verkkovirtalaitteita. Oli myös mahdollista ottaa käyttöön tavallinen Arduino ja erikseen releet, mutta Controllinon tarjoama ratkaisu on erittäin näppärä ja turvallinen. Se on myös suunniteltu virallista käyttöä varten, joten demoversio voidaan asentaa kasvatusalustaan kätevästi.

Tekniset tiedot

CONTROLLINO MAXI Automation on yksi CONTROLLINO-perheen kehitysalustoista. Se on muotoiltu kiinnittymään DIN-kiskolle, joten moduuli voidaan asentaa koteloon muun automatiikan rinnalle. Sen operointijännite on tyypillisesti 5 volttia, mutta releitä ohjatakseen CONTROLLINO:n on kytkettävä 24 volttia käyttöjännitettä (kuva 4).



Kuva 4. CONTROLLINO MAXI:n sisään- ja ulostulojen kartta [6].

CONTROLLINO MAXI:n yleisempiä ominaisuuksia ovat:

- prosessori ATmega 2560
- 12 sisääntuloa + 12 ulostuloa
- 10 relettä
- Ethernet-portti
- I^2C
- SPI
- RTC.

Controllinon koodausalustana voi toimia Arduino IDE, jossa saa kätevästi lisättyä kaikki tarvittavat kirjastot. Koodikielenä on C kieli, sekä usein C++ kieli. [6.]

3.3 Anturit

KaMu-järjestelmään tulevat anturit voi jakaa kolmen ryhmään niiden tarkoituksen perusteella: tiedonkeruuanturit, ohjausanturit ja vikatila-anturit. Tiedonkeruuantureiden rooli on hyvin passiivinen: ne keräävät dataa ympäristön olosuhteista ja toimittavat tulokset prosessorille, josta tieto kulkeutuu Ethernet-yhteydellä käyttäjän laitteeseen.

Ohjausantureiden toiminta on monimutkaisempi, ja tässä opinnäytetyössä niitä ei vielä otettu käyttöön. Sekä sumutus että valaistus ovat aikaohjattuja, mutta ohjausanturin käyttöönotto mahdollistaisi sen, että tietyissä olosuhteissa esimerkiksi liiallisessa kuivuudessa juuristossa aktivoituu ylimääräinen pumpun päällä käyminen. Vikatila-anturit ovat käytännössä myös ohjausantureita ja niiden havaitessa ongelmatilanteen pääohjaus kytkeytyy pois päältä ja käyttäjälle tulee viesti vikatilasta. Alla on tarkempi tekninen kuvaus käytetyistä antureista.

3.3.1 DHT11

DHT11 sisältää yhdessä kuoressa sekä lämpötila- että kosteusanturin. Anturi on digitaalinen, ja käyttöjännitteenä sille on tarkoitettu viisi volttia. Lämpötilaa se lukee parhaimmillaan 0–50 °C (tarkkuus kaksi astetta), ja kosteuden mittaustarkkuus on noin viisi prosenttia. DHT22 olisi tarkempi anturi, ja siinä lämpötilaa mitataan puolen asteen tarkkuudella. DHT11 ja DHT22 antureille on luotu Arduinoon yhteinen kirjasto. Kyseinen anturi asennetaan järjestelmän ulkopuolelle lähelle lehdistöä seuraamaan lämpötilaa ja kosteutta. [7.]

3.3.2 DS18B20

DS18B20 on digitaalinen lämpötila-anturi, jonka mittausväli on -55–125 °C ja tarkkuus noin puoli astetta. Anturia varten on kehitelty lukuisia kirjastoja, mutta

välttämätön on OneWire.h ja sen lisäksi tässä projektissa on käytetty DallasTemperature.h kirjastoa helpottamaan työskentelyä.

KaMuun asennetaan kaksi tällaista anturia mittaamaan sisälämpötilaa arkussa sekä ravinneliuoksen lämpötilaa. Anturit on tehty vedenkestäviksi, joten ne soveltuvat hyvin tähän tarkoitukseen. [8.]

3.3.3 Kelluntakytkin

Kelluntakytkin on kellukkeella ohjattava kytkin, joka siirtyy päälle ja pois kellukkeen noustessa tai laskiessa, joten sitä voi käytännössä käyttää digitaalisena anturina. Tämän anturin käyttö on erittäin helppoa sen yksinkertaisuuden ansiota. Kellukkeen materiaali soveltuu erityyppisiin ja lämpöisiin nesteisiin [9].

Etuna muihin vedentasoantureihin on se, että tilanmuutos tapahtuu mekaanisesti eikä sähköisesti. Mekaaninen ratkaisu on luotettavampi kuin elektroninen, joten tällainen anturi on hyvä valinta vikatilan havaitsemiseen.

3.4 Vikatila

Vikatilaksi sanotaan tilannetta, jossa jokin asia häiritsee tai tulee pian häiritsemään koko järjestelmän toimintaa. KaMussa on kaksi tärkeintä vikatilavaihtoehtoa ja molemmat koskevat sumutusjärjestelmän toimintaa: vedentaso sekä pumpun toiminta. Veden tason laskettua liian alhaiseksi tulee vaaratilanne sekä kasveille että koko järjestelmälle, koska sen lisäksi että juuristo ei saa vettä, myös vesipumppu alkaa pumppaamaan ilmaa, mikä rikkoo pumpun hyvin nopeasti. Tätä välttääkseen on hyvä, että käyttäjällä on mahdollisuus sulkea pumppu myös etänä heti saatuaan ilmoituksen matalasta vedentasosta.

Pumpun toiminnan tarkka ja luotettava seuranta on hieman monimutkaisempi, ja on päädytty siihen, että paras tapa tähän on asentaa pumpun ja suutinten väliin virtausgeneraattori, joka toimisi käytännössä analogisena anturina. Tätä

opinnäytetyötä varten ei saatu ajoissa virtausgeneraattoria, joten se lisätään järjestelmään tulevaisuudessa.

Muita vikatilaan liittyviä tilanteita ovat esimerkiksi erittäin suuret tai liian pienet lämpötilat, niiden havaitseminen ei suoraan kytke pois mitään prosesseja, vaan käyttäjä itse päättää, miten hän saa korjattua tilanteen.

Vikatilaan siirtymisestä tulee ohjaamoon ilmoitus käyttäjälle. Alun perin tavoitteena oli saada myös tapa lähettää käyttäjälle puhelimeen ilmoitus ongelmatilanteesta, mutta sitä ei saatu toimimaan Node-Redin avulla. Tämä on siis otettava huomioon koodissa, jos käyttäjä ei reagoi ongelmaan millään tavalla, niin järjestelmä itse ohjaa kriittiset prosessit pois päältä.

4 Käyttöliittymä ja koodi

KaMun automaatio-ohjauksen voi jakaa varsinaiseen koodiin, jolla ohjataan suoraan CONTRILLINO:lla ja siihen kytkettyjä antureita ja laitteita, sekä käyttöliittymään, joka on rakennettu Node-Red-nimisellä alustalla. Molemmat osat ovat tiiviissä yhteistyössä toistensa kanssa MQTT-protokollan avulla.

Laiteohjelmaa on kirjoitettu Arduino IDE -nimisessä ohjelmassa ja ladattu CONTROLLINO:lle USB-portin kautta. Node-Redissä koodattiin korkeamman tason prosesseja, muun muassa käyttöliittymän ulkoasua, tietojen keräystä ja tallentamista.

4.1 Arduino IDE

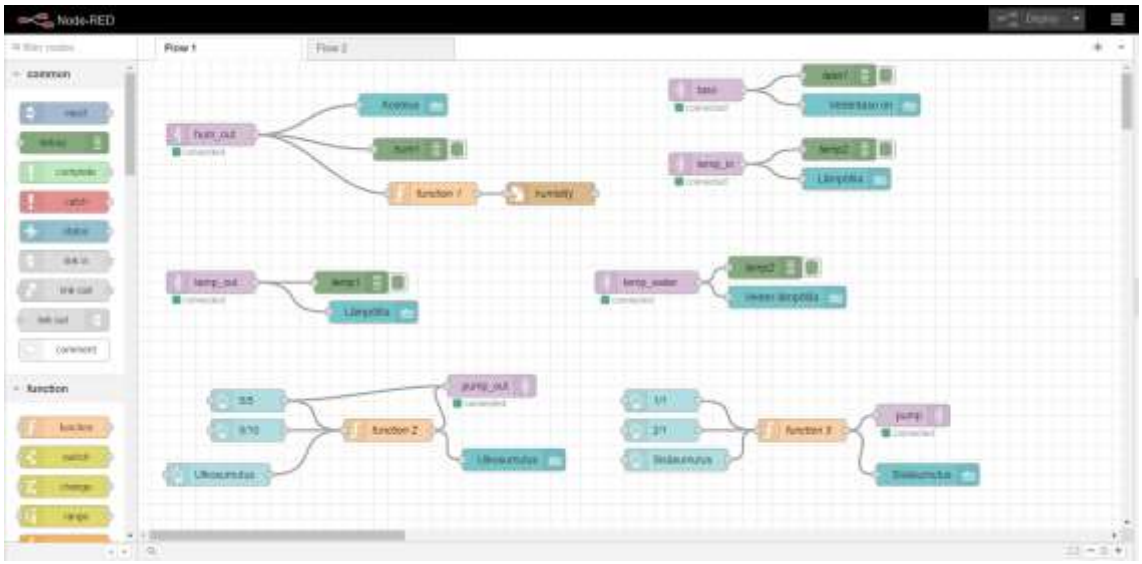
Arduino IDE on koodausalusta, jolla kirjoitetaan laiteohjelmistoa, ja se tukee useita eri laitteita (Arduino, ESP, Controllino yms.). Kieli on C:tä ja C++:a, ja ohjelmaan on sisäänrakennettu tarkistustoiminto, joka etsii koodissa virheitä. Ohjelmassa on myös hyvin kätevää hakea ja ladata erilaisia kirjastoja, joita tarvitsee omassa projektissa. ”Serial monitor” toiminnolla saa tulostettua näytölle tarvittua informaatiota, esimerkiksi anturiarvoja.

Opinnäytetyön koodi suorittaa seuraavia tehtäviä: lukee anturiarvoja, keskustelee Node-Redin kanssa MQTT:n avulla lähettämällä ja vastaanottamalla dataa ja käskyjä sekä määrittää sumutuksen ja valaistuksen ajastusta. Ajastettu ohjaus sumutuksessa on toteutettu syklimäisesti, ja sitä varten on koodissa käytetty millis-toiminto.

Aikaa laskettaessa C++ kielessä voi käyttää delay-komentoa tai millis-struktuuria. Vaikka delay on erittäin nopea ja helppo tapa luoda muun muassa syklejä, ja sen käytössä on yksi ratkaiseva haittapuoli, kun delay komentoa suoritetaan, koko ohjelma jäätyy siksi aikaa eikä pysty lukemaan anturidataa, keskustelemaan Node-Redin kanssa tai mitään muuta. Sumutusta varten tarvitaan syklejä, joissa lepoväli saattaa olla useita minuutteja pitkä. Jos otetaan käyttöön delay-komento, se tarkoittaa, että järjestelmä olisi suurimman osan ajasta jäätynyt, eikä käyttäjällä olisi mitään mahdollisuutta vastaanottaa tietoa ja lähettää komentoja. Millis sen sijaan ei pysäytä koko ohjelmaa tietyksi ajaksi, vaan asettaa aikalaskurin tietystä hetkestä ja suorittaa annetun komennon määräajan kuluessa. Tämä tapa ajastaa suoritusta ei jäädytä koko ohjelmaa, ja tämän takia millis on paljon turvallisempi käyttää.

4.2 Node-Red

Node-Red-alustaa päädyttiin käyttämään, sillä siinä saa hyvin nopeasti ja vaivattomasti tehtyä rakenteita ja muokattua niitä. Koodikielenä on JavaScript, mutta sitä tarvitsee, kun tekee hieman monimutkaisempaa järjestelmää. Visuaalisuus helpottaa työskentelyä, pitää vain valita tarvittavat solmut ja yhdistää ne toisiinsa, ja yhdistely voi tapahtua usealla eri tavalla (kuva 5). Node-Redin solmut jakautuvat aihealueisiin, joista osa on tiedon vastaanottoa ja -lähetystä varten, osalla on funktionaalinen tarkoitus, joka liittyy muiden solmujen ohjaamiseen, osa kääntää tiedot yhdestä muodosta toiseen ja osa tallentaa tiedostoja. Voi myös ladata lisää solmuryhmiä, ja yksi tärkeimmistä ovat käyttöliittymän luontiin tarkoitetut solmut. Niillä saa rakennettua koko käyttöliittymän lisäämällä nappeja, kytkimiä, tekstikenttiä, kuvaajia yms.



Kuva 5. Node-Redin kehitysalusta.

Opinnäytetyötä tehtäessä on keskitytty nimenomaan testaamaan Node-Redissä rakennetun käyttöliittymän soveltuvuutta KaMulle, joten demoversiossa ei ole kaikkia ominaisuuksia ja ulkoasun parantamiseen ei käytetty aikaa. On kuitenkin päädytty siihen, että kyseinen tapa rakentaa käyttöliittymää on hyvin kätevä ja sitä on helppoa laajentaa tai muuttaa. Kaikkien muutosten pitää olla myös laiteohjelmakoodissa otettu huomioon.

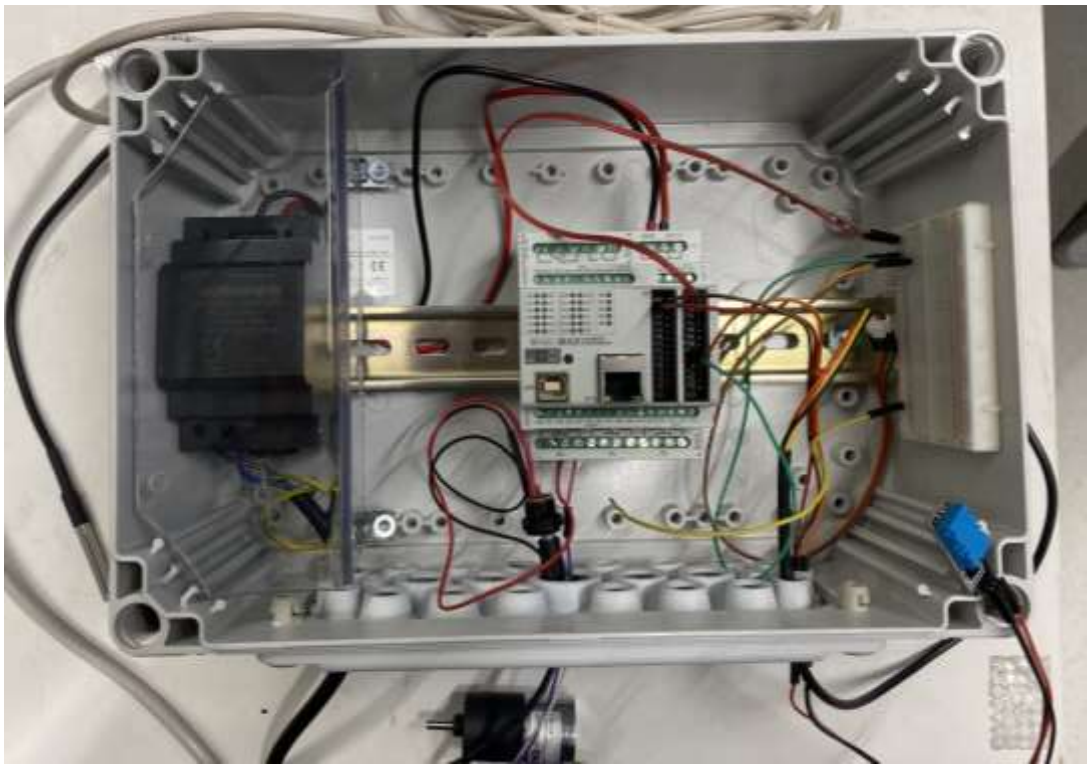
5 Pää- ja varajärjestelmä

Opinnäytetyön alkuperäisenä tarkoituksena on ollut suunnitella pääjärjestelmä, joka pysyisi yksinkertaisena ja tarjoaisi etäohjauksen ja -luennan mahdollisuuksia, sekä sen rinnalle varajärjestelmä, joka korvaisi ensimmäisen sen rikkoutuessa. Varajärjestelmä oli tarkoitus suunnitella suurimmaksi osaksi analogiseksi ja olla käyttämättä mikrokontrolleria.

5.1 Pääjärjestelmä

CONTROLLINO MAXI on asennettu koteloon läpinäkyvällä kannella. Kuten on jo mainittu, ohjatakseen releitä CONTROLLINO tarvitsee käyttöjännitteeksi 24

volttia, joten asennuskoteloon oli lisättävä sopiva muuntaja (kuva 6). Muuntaja on turvallisuussyistä eristetty muusta kotelon tilasta muovisella suojalla, sillä käyttäjällä, jolla saattaa olla tarve päästä ohjausyksikköön vaihtamaan anturia, ei saisi olla pääsyä koskemaan verkkovirtapuoleen.



Kuva 6. Ohjausyksikön koekytkentä

5.2 Varajärjestelmä

Varajärjestelmä on ainoa osa opinnäytetyöstä, jota ei pystytty suunnittelemaan tarkasti alusta loppuun. Kuitenkin on selkeä näkemys siitä, millainen sen pitäisi olla pääpiirteittäin. Järjestelmän päätoiminnallisuuksina ovat sumutuksen ajastettu ohjaus sekä vikatilan tunnistaminen ja reagointi siihen.

Tuleva piiri suunnitellaan mahdollisimman yksinkertaiseksi, sillä tavoitteena on saada lopputulos, jonka korjaaminen tulee edulliseksi ja vaivattomaksi. Ajastettu ohjaus toteutetaan kellopiirin avulla, vikatila tunnistetaan operaatiovahvistimella

ohjatun anturin avulla ja kaikki ohjaussignaalit vahvistetaan releitä varten, jotka kytketään suoraan laitteisiin.

6 Johtopäätökset ja yhteenveto

Vaikka opinnäytetyön käytännön osuus jäi demovaiheeseen eikä kaikkia ominaisuuksia päässyt toteuttamaan ja testaamaan, on kuitenkin tehty merkittävä työ Aeropodin yksittäisen kasvatusmoduulin automatisoinnin luomiseksi. Projektissa on otettu käyttöön täysin uusia laitteita ja menetelmiä, mikä on tuonut sekä uusia mahdollisuuksia että haasteita kokemattomuuden vuoksi. Yrityksen puolesta valitut ratkaisut ovat onnistuneita ja soveltuvat myös tuleviin ratkaisuihin. Projekti on saatettu siihen vaiheeseen, että sen jatkokehittely ja parantaminen ovat helposti toteutettavissa työprosessiin ja materiaaleihin tutustumisen jälkeen.

Aeroponisen järjestelmän automatisointi aiheena on hyvä kokonaisuus opinnäytetyötä varten, sillä se pakottaa opiskelijaa tutustumaan muihin aloihin, eikä anna vaan keskittyä omaan osaamisalueeseen. Työskentely yrityksen kanssa on myös kannattavaa etenkin ammattikorkeakoulun opiskelijalle, jonka opinnot ovat olleet käytäntöpainotteisia. Mahdollisten tulevan työpaikan lisäksi saa aitoa kokemusta työskentelystä asiakkaan kanssa, mikä on tulevalle insinöörille arvokas kokemus.

Lähteet

1 Lilkin, How Does Aeroponics Work? Modern Farmer. Verkkoaineisto. <https://modernfarmer.com/2018/07/how-does-aeroponics-work/>. Luettu 9.11.2022.

2 About Aeroroots. Aeroroots. Verkkoaineisto. <https://aeroroots.com/>. Luettu 9.11.2022.

3 Aeroponic Technology. LettusGrow. Verkkoaineisto. <https://www.lettusgrow.com/aeroponic-technology>. Luettu 9.11.2022.

4 Introduction to Aeroponics. Aerobloom. Verkkoaineisto. <https://aerobloom.com/what-is-aeroponics/>. Luettu 9.11.2022.

5 KALVOPUMPPU VP300 300W. IKH. Verkkoaineisto. <https://www.ikh.fi/fi/kalvopumppu-vp300-300w-el018>. Luettu 9.11.2022.

6 CONTROLLINO MAXI. Controllino. Verkkoaineisto. <https://www.controllino.com/product/controllino-maxi/>. Luettu 9.11.2022.

7 DHT11, DHT22 and AM2302 Sensors. Adafruit Learning system. Verkkoaineisto. <https://learn.adafruit.com/dht>. Luettu 9.11.2022.

8 DS18B20, Programmable Resolution 1-Wire Digital Thermometer. Maxim Integrated. Verkkoaineisto. <https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf>. Luettu 9.11.2022.

9 Kelluntakytkin. Partco. Verkkoaineisto. <https://www.partco.fi/fi/rakennussarjat/crowtail/23732-ect-ct008315s.html>. Luettu 9.11.2022.

Kamu järjestelmän koodi

```
//////LIBRARIES/////////
#include <DHT.h>
#include <SPI.h>
#include <Controllino.h>
#include <DallasTemperature.h>
#include <OneWire.h>
#include <Ethernet.h>
#include <ArduinoMqttClient.h>
//////////////////////////
///SENSORS///
#define ONE_WIRE_BUS CONTROLLINO_PIN_HEADER_SDA
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);

#define DHTPIN 6     // Digital pin connected to the DHT sensor
#define DHTTYPE DHT11   // DHT 11
DHT dht(DHTPIN, DHTTYPE);

//////////////
///ETHERNET////

EthernetClient net;
byte mac[] = { 0x00, 0xAA, 0xBB, 0xCC, 0xDE, 0x02 };
byte ip[] = {192, 168, 8, 101};
//192.168.8.101
////////////////

///MQTT/////
MqttClient mqttClient(net);
StaticJsonDocument<400> doc;

char broker[] = "test.mosquitto.org";
int port = 1883;

//create topic name

const char topic[]  = "temp_out";
const char topic1[]  = "hum_out";
const char topic2[]  = "temp_in";
const char topic3[]  = "pump";
const char topic4[]  = "taso";
const char topic5[]  = "pump_out";
const char willTopic[] = "arduino/will";

String hum1;
String temp1;
String temp2;
String pump1;
String taso1;
String pump_out1;

bool willRetain = true;
int willQos = 1;

/////////////////////////////////////////////

///DEFINITIONS///
////SISÄSUMUTUS////
```

```
int R0 = CONTROLLINO_R0;
int DO2 = CONTROLLINO_DO2;
int DO0 = CONTROLLINO_DO0;
int sumutus1;
int sumutus2;
//////////////////
///ULKOSUMUTUS
int R1 = CONTROLLINO_R1;
int DI2 = CONTROLLINO_DI3;
int DO7 = CONTROLLINO_DO7;
int sumutus3;
int sumutus4;

int DO6 = CONTROLLINO_DO6;

int taso;

int numberOfDevices;

//ajastus ja sumutusvälit

const long interval = 3000;
unsigned long previousMillis = 0;
unsigned long previousMillis2 = 0;
unsigned long previousMillis3 = 0;

const long interval1 = 3000;
const long interval2 = 5000;
const long interval3 = 1000;
const long interval4 = 2000;

const long interval5 = 5000;
const long interval6 = 10000;
const long interval7 = 5000;
const long interval8 = 15000;

int count = 0;

////////////////////////////////
////SETUP///

void setup() {

  Serial.begin(9600);
  Ethernet.begin(mac, ip);
  dht.begin();
  sensors.begin();
  pinMode(R1, OUTPUT);
  digitalWrite(R1, LOW);
  pinMode(R0, OUTPUT);
  digitalWrite(R0, LOW);

  mqttClient.beginWill(willTopic, willPayload.length(), willRetain, willQos);
  mqttClient.print(willPayload);
  mqttClient.endWill();

  if (!mqttClient.connect(broker, port)) {
    Serial.print("MQTT connection failed! Error code = ");
    Serial.println(mqttClient.connectError());

    while (1);
    }

  mqttClient.onMessage(onMqttMessage);

  int subscribeQos = 1;
  int subscribeQoSs = 0;
```

```
mqttClient.subscribe(topic, subscribeQos);
mqttClient.subscribe(topic1, subscribeQos);
  mqttClient.subscribe(topic2, subscribeQos);
  mqttClient.subscribe(topic3, subscribeQoSs);
  mqttClient.subscribe(topic4, subscribeQos);
  mqttClient.subscribe(topic5, subscribeQos);

  Serial.print("Waiting for messages on topic: ");
  Serial.println(topic);
  Serial.println();

}

void onMqttMessage(int messageSize) {

  char teksti[20];
  int i = 0;
  while (mqttClient.available()) {

    teksti[i++] = (char)mqttClient.read();
    }

  teksti[i] = 0;
  if(strcmp(teksti, "2/1")==0) {

    digitalWrite(DO2, HIGH);
    digitalWrite(DO0, LOW);
  }

 else if(strcmp(teksti, "SULJE_SISÄ")==0)

    {digitalWrite(DO2, LOW);}

 if(strcmp(teksti, "1/1")==0)

    {
    digitalWrite(DO0, HIGH);
    digitalWrite(DO2, LOW);
    }

  else if(strcmp(teksti, "SULJE_SISÄ")==0)

    {digitalWrite(DO0, LOW);}

  if(strcmp(teksti, "5/10")==0) {

    digitalWrite(DI2, HIGH);
    digitalWrite(DO7, LOW);
  }

 else if(strcmp(teksti, "SULJE_ULKO")==0)

    {digitalWrite(DI2, LOW);}

 if(strcmp(teksti, "5/5")==0)

    {
    digitalWrite(DO7, HIGH);
    digitalWrite(DI2, LOW);
    }

  else if(strcmp(teksti, "SULJE_ULKO")==0)

    {digitalWrite(DO7, LOW);}

}
void loop() {
```

```
unsigned long currentMillis = millis();

  if (currentMillis - previousMillis >= interval)
    {

      mqttClient.poll();

      float h = dht.readHumidity();

      float t = dht.readTemperature();

      sensors.requestTemperatures();
      float tempC = sensors.getTempCByIndex(0);

      taso = digitalRead(DO6);

      hum1 = String(h);
      temp1 = String(t);
      temp2 = String(tempC);
      taso1 = String(taso);
      bool retained = false;
      int qos = 1;
      bool dup = false;

      mqttClient.beginMessage(topic, temp1.length(), retained, qos, dup);
      mqttClient.print(temp1);
      mqttClient.endMessage();

      mqttClient.beginMessage(topic1, hum1.length(), retained, qos, dup);
      mqttClient.print(hum1);
      mqttClient.endMessage();

      mqttClient.beginMessage(topic2, temp2.length(), retained, qos, dup);
      mqttClient.print(temp2);
      mqttClient.endMessage();

      if (taso == HIGH)

        {
          mqttClient.beginMessage(topic4, retained, qos, dup);
          mqttClient.print("jees");
          mqttClient.endMessage();
        }

      else if (taso == LOW)

        {
          mqttClient.beginMessage(topic4, retained, qos, dup);
//taso1.length()
          mqttClient.print("MATALA");
          mqttClient.endMessage();
        }

      sumutus1 = digitalRead(DO2);
      sumutus2 = digitalRead(DO0);
      sumutus3 = digitalRead(DI2);
      sumutus4 = digitalRead(DO7);

      if (sumutus1 == HIGH)
      {

        unsigned long CurrentMillis = millis();

if (CUrrentMillis - previousMillis3 >= interval5 && CUrrentMillis - previ-
ousMillis3 < interval6)  //interval1 = 2000, interval11 = 3000
          {

            digitalWrite(R0, HIGH);
```

```
          }

        else if (CUrrentMillis - previousMillis3 >= interval6)
          {

            digitalWrite(R0, LOW);
            previousMillis3 = CUrrentMillis;
          }

        else







{
            digitalWrite(R0, LOW);
          }
      }
      else if (sumutus2 == HIGH)
      {
        unsigned long CUrrentMillis = millis();

        if (CUrrentMillis - previousMillis3 >= interval7 && CUrrentMillis -
previousMillis3 < interval8)  //interval1 = 2000, interval11 = 3000
          {

            digitalWrite(R0, HIGH);

          }

        else if (CUrrentMillis - previousMillis3 >= interval8)
          {

            digitalWrite(R0, LOW);
            previousMillis3 = CUrrentMillis;
          }

        else
          {
            digitalWrite(R0, LOW);
          }
      }

else
          {
            digitalWrite(R1, LOW);
          }

    }

  Serial.println();
 }
```