Lauri Kirvesniemi

Puoliautomaattisen mäskäys- ja keittolaitteiston suunnittelu sekä toteutus

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Automaatiotekniikka

Insinöörityö

16.5.2017

Metropolia

Tekijä(t) Otsikko Sivumäärä Aika	Lauri Kirvesniemi Puoliautomaattisen mäskäys- ja keittolaitteiston suunnittelu sekä toteutus 37 sivua + 2 liitettä 16.5.2017
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Automaatiotekniikka
Suuntautumisvaihtoehto	
Ohjaaja(t)	Hallituksen puheenjohtaja Erno Iipponen Lehtori Markku Inkinen

Tämän insinöörityön tarkoituksena on puoliautomaattisen mäskäys- ja keittolaitteiston suunnittelu sekä toteutus. Valmis työ on pienpanimokäyttöön tarkoitettu prototyyppi. Työ tehtiin Lapinlahti Beer Fellows ry:lle. Työssä käydään läpi laitteistolta vaadittavat prosessit sekä itse laitteiston valmistusprosessi alusta loppuun.

Työn aloitus painottui mekaaniselle rakentamiselle, joka piti sisällään kattilakokonaisuuden sekä ohjauskärryn suunnittelun, rakentamisen sekä kompponenttien asentamisen. Johdotus suoritettiin tämän jälkeen, ja työn sähkökuvat piirrettiin tietokoneella työn loppuvaiheessa.

Työn keskivaiheissa käsitellään laitteistoon asennettuja komponentteja, mikrokontrollerin ohjelmointia, PID-säätimen viritystä sekä mobiiliapplikaation ohjelmointia. Työn loppuvaihe sisältää laitteiston testausta sekä pohdintaa työstä.

Laitteiston tavoitteena oli suoriutua sille asetetuista panimoprosessien vaatimuksista. Työvaiheiden lopputuloksena rakentui sulautettu prosessinohjausjärjestelmä, joka kykenee ohjaamaan kolmivaihevirtaa PID-ohjauksella käyttäjän mobiiliapplikaatiolla valitsemien parametrien mukaisesti.

Avainsanat	Ohjelmointi, PID, Android, Arduino, sulautettu järjestelmä

Author(s) Title Number of Pages Date	Lauri Kirvesniemi Design and Implementation of a Semi-Automatic Brewing Device 37 pages + 2 appendices 16 May 2017
Degree	Bachelor of Engineering
Degree Programme	Automation Engineering
Specialisation option	
Instructor(s)	Erno Iipponen, Chairman of the Board Markku Inkinen, Senior Lecturer

The purpose of this study was to design and implement a semi-automatic brewing device. The completed device is a prototype for small scale breweries. The work was completed for a registered association called Lapinlahti Beer Fellows ry. This thesis covers required processes concerning the device and the whole manufacturing process of the device.

The beginning of this thesis covers mechanical construction that consisted of designing and building of mashing vessel, boiling vessel, controlling cart, and installations to these. Wiring was performed after these steps, and electrical drawings were made with computer during later phases of the project.

The middle part of this thesis covers installed components, microcontroller programming, PID-controller tuning and mobile application programming.
The final part of this thesis covers device testing and reflection on the work.

The objective of this device was to perform under the requirements of the designated brewing processes. As the result of the work phases, an embedded process control system that is capable of controlling three phase current with a PID-controller by the parameters set by user from a mobile application was constructed.

Keywords	Programming, PID, Android, Arduino, embedded system

Sisällys

Lyhenteet ja sanaselitykset

1 Johdanto 4

2 Laitteistolla suoritettavat prosessit 4

2.1 Mäskäysprosessi 4
2.2 Keittoprosessi 5

3 Mäskäys- ja keittokattila 6

3.1 Lämmitysvastus 7
3.2 Lämpötila-anturi 8

4 Ohjauskärry 11

4.1 Kojeistus ja kytkennät 11
4.2 Vierrepumppu 12
4.3 Bluetooth-moduuli 13

5 Arduinon ohjelmointi 15

5.1 Lämpötiladata 16
5.2 PID-kirjasto 17
5.3 Arduinoon tuleva data 18
5.4 Arduinosta lähtevä data 19

6 PID-säädin 20

6.1 Ziegler-Nichols menetelmät 22
6.1.1 Askelvastekoe asetusarvon muutoksella 22
6.1.2 Askelvastekoe ulostuloarvon muutoksella 25
6.2 Viritysarvojen vertailu 25

7 Android-ohjelma 27

7.1 Ohjelman komponentit 27
7.2 Tiedon vastaanotto Arduinosta 29
7.3 Tiedon lähetys Arduinoon 30

7.4 Graafinen käyttöliittymä 31

8 Laitteiston testaus 33

9 Pohdintaa 34

Lähteet 36

Liitteet

Liite 1. Sähkökuvat osa 1

Liite 2. Sähkökuvat osa 2

Lyhenteet ja sanaselitykset

ASCII	American Standard Code for Information Interchange. Tietokoneissa yleisesti käytettävä merkistö.
Bluetooth	Avoin standardi laitteiden langattomaan kommunikointiin lähietäisyydellä.
EEPROM	Electronically Erasable Programmable Read-Only Memory. Haihtumatonta puolijohdemuistia jota voidaan uudelleenkirjoittaa n.10 000–100 000 kertaa.
MAC-osoite	Media Access Control-osoite. Yksilöllinen verkkosovittimen osoite ethernet-verkossa.
Mäski	Jäljelle jäänyt sivutuote, joka on yleisimmin valmistettu mallasrouheista.
PID-säädin	Proportional-integral-derivative-säädin, joka muodostuu kolmesta termistä: suhde, integroiva ja derivoiva.
SSR-rele	Solid-state-rele on puolijohderele, joka ei sisällä mekaanisesti liikkuvia osia. Tavanomaisen releen kärjet ovat puolijohdereleissä korvattu useimmiten transistoreilla ja tyristoreilla.
ULWD	Ultra low watt density. Kuvaa lämmitysvastuksen tehojakaumaa sen pinta-alan suhteen. ULWD-vastuksia käytetään erityisesti panimolaitteistoissa pienentämään vierteen kiinnipalamisen riskiä.
Vierre	Panoprosessin lopputuote, joka jää jäljelle mäskäämisen ja keiton jälkeen. Vierre sisältää mm. maltaista irronneet sokerit.

1 Johdanto

Tämän insinöörityön aiheena on puoliautomaattisen mäskäys- ja keittolaitteiston suunnittelu sekä toteutus. Työ toteutettiin Lapinlahti Beer Fellows ry:lle.

Pien- ja mikropanimot ovat lisääntyneet Suomessa viime vuosina hyvin paljon, ja kiinnostus käsityöläisoluisiin, kotioluen valmistukseen ja sen vaatimaan laitteistoon on kasvanut samalla huomattavasti [1].

Tämän laitteiston tarkoituksena on pystyä valmistamaan 50 litraa vierrettä, jota fermentoimalla pystytään valmistamaan valmista olutta. Laitteisto koostuu keitto- ja mäskäyskattilasta sekä ohjauskärrystä, johon laitteistoa ohjaavat komponentit on asennettu. Laitteisto toimii kolmivaihesähkövirralla, jota ohjataan keittokattilassa sijaitsevalle lämmitysvastukselle puolijohdereleellä PID-ohjauksen avulla. Laitteistoa ohjataan Android-mobiililaitteeseen asennetulla applikaatiolla Bluetooth-yhteyden kautta. Valmis laitteisto on prototyyppi pienpanimokäyttöön, ja laitteiston kehitystä on tulevaisuudessa tarkoitus jatkaa eteenpäin yhteistyössä yhdistyksen kanssa.

Lapinlahti Beer Fellows ry. on olutharrastajien yhteenliittymä, ja sen jäsenet ovat kotipanijoiden roolissa olevia yksityishenkilöitä. Yhdistys toimii Lapinlahden sairaalassa Helsingissä. [2.]

> Yhdistyksen tarkoituksena on edistää ja levittää olutkulttuuria yleisesti, kohottaa käsityöläisoluiden ja oluttyylien tuntemusta, jäsentensä oluentietämystä yleensä, herättää heissä oluthenkisyyttä ja pysyvää harrastuneisuutta laadukkaiden ja monipuolisten oluiden parissa sekä tarjota jäsenilleen puitteet oluen parissa viihtymiseen. Tavoitteena on myös olutharrastukseen liittyvien positiivisten mielikuvien ja tiedon lisääminen olueen liittyvässä keskustelussa. [2.]

2 Laitteistolla suoritettavat prosessit

2.1 Mäskäysprosessi

Mäskäyksessä idätettyjä jyviä eli maltaita pidetään rouhittuina tietyssä lämpötilassa vedessä, jolloin maltaiden entsyymitoiminta aktivoituu ja sen seurauksena maltaiden sisältämät ainesosat alkavat muokkautua. Rouhitut maltaat lisätään mäskikattilaan, jossa on suodatuslevyllä varustettu valepohja. Valepohja mahdollistaa lämpimän veden

virtaamisen mäskin lävitse ilman mallasrouheen pääsyä nesteeseen. Mäskäyksellä pyritään saamaan maltaista irti pääasiassa käymiskelpoisia sokereita. [3, s. 130.]

Yleisimmin kotipanimoissa käytetään yksivaiheista infuusiomäskäystä, jossa vettä pidetään oluen reseptistä riippuen tietyssä lämpötilassa 45–90 minuutin ajan maltaiden ollessa vedessä. Yleisesti lämpötilaa pidetään n. 65–70 °C:ssa. Tänä aikana maltaista irtoaa reseptistä riippuen pääasiassa käymiskelpoisia sokereita vierteeseen maltaiden entsyymitoiminnan tuloksena. Tämä tekniikka toimii useimmissa olutresepteissä hyvin, sillä nykyään maltaat ovat hyvin modifioituineita. Yksivaiheisella infuusiomäskäyksellä niistä saadaan hyvällä hyötysuhteella irti sokereita, sekä säästetään aikaa verrattuna monivaiheinfuusiomäskäykseen. [3, s. 149.] Laite on varustettu vierrepumpulla, joka pumppaa vastuksilta kuuman veden kattilan pohjalta suoraan mallaspatsaan päälle. Tällöin maltaiden lämpötila pysyy tasaisena koko mäskäysprosessin ajan.

Mäskäys voidaan suorittaa myös monivaiheisesti eri lämpötilojen avulla, jolloin eri lämpötila-alueilla aktivoidaan eri entsyymejä. Tällöin mäskin lämpötilaa pidetään halutun entsyymin lämpötila-alueella reseptin vaatima aika, jonka jälkeen siirrytään seuraavan entsyymin lämpötila-alueelle. Entsyymien toimintakyky tuhoutuu, jos lämpötila nousee liikaa niiden optimilämpötilasta. [3, s. 131.] Tämän takia tarkka lämpötilakontrolli on ehdottoman tärkeä mäskäyksen aikana.

Kun mäskäysaika on tullut täyteen, mäski useimmiten vielä huuhdotaan ja valutetaan, jotta saataisiin viimeiset mäskiin jääneet sokerit keittokattilan nesteeseen. Tällä laitteistolla sen voi toteuttaa nostamalla mäskikattila keittokattilan yläpuolelle lepäämään apukannen päälle, jolloin vierteestä erotetun mäskin sokerit on helppo liuottaa irti mäskistä kaatamalla kuumaa vettä mäskipatsaan päälle. Tämän seurauksena kuuma vesi valuu mäskin, valepohjan, sekä apukannen lävitse keittokattilassa olevaan nesteeseen sisältäen mäskin sokerijäämiä. [3, s. 154.]

2.2 Keittoprosessi

Mäskäyksen jälkeen keittokattilaan syntynyttä nestettä keitetään puolesta tunnista kahteen tuntiin. Prosessin aikana tapahtuu keiton humalointi, sterilointi sekä nesteen kondensoitumista höyrystymisen seurauksena. Kun neste on keittämisen aikana steriloitunut, tulee vierteen käsittelyssä tämän jälkeen käyttää ainoastaan steriloituja

välineitä kontaminaatioriskin minimoimiseksi. Keittämisen jälkeen myös vierteen jäähdytys tulee toteuttaa mahdollisimman nopeasti, koska kontaminaatioriski on korkeampi vierteen ollessa vielä lämmintä. [3, s. 77.]

3 Mäskäys- ja keittokattila

Kattilakokonaisuus sisältää 70 litran keittokattilan, joka seisoo säädettävillä jaloilla, sekä 50 litran mäskäyskattilan, joka asettuu joko keittokattilan sisälle tai päälle (kuva 1). Keittokattilaan on asennettu lämmitysvastus, lämpötila-anturi sekä suodattimilla varustettu läpivienti vierrepumpulle. Mäskäyskattilan ollessa keittokattilan sisällä mäskäyskattilan pohja asettuu 10 cm:n korkeudelle keittokattilan pohjasta jättäen näin riittävästi tilaa vastuksille, pumpun liitännälle suodattimineen sekä lämpötila-anturille. Mäskäyskattilan pohja on leikattu osittain auki ja siihen on jätetty tukiristikko tukemaan maltaiden painoa (kuva 1). Ristikon päälle asetetaan suodatinlevy, joka estää mallasrouheen päätymistä keittokattilaan päästäen kuitenkin sieltä tulevan nesteen läpi. Kummatkin kattilat sekä niissä olevat liitännät ovat ruostumatonta terästä, josta ei liukene prosessin aikana mitään ylimääräistä nesteeseen. Keittokattilan lämmitysvastukselle tulee kolmivaihevirtakaapeli, jonka toisessa päässä oleva pistotulppa kytketään ohjauskärryyn. Lämpötila-anturi on kytketty RJ11-kaapelilla, joka on mahdollista irrottaa keittokattilasta. Näiden lisäksi pumpulle on ulostuloletku, jolle on pikakiinnitys ohjauskärryn alaosassa vierrepumpun sisääntulossa.

Kuva 1. Vasemmalla mäskäyskattilan pohja plasmaleikkauksen jälkeen. Oikealla keittokattila, jonka päällä mäskäyskattila pysyy apukannen avulla

Keittokattilan jalat ovat hitsattu kiinni, ja hitsaussaumat ovat käsitelty Polinox-P Rapid -peittaustahnalla ruostumattoman teräksen ominaisuuksien säilyttämiseksi. Myös mäskäyskattilan leikkaussaumat on käsitelty tällä aineella. Mäskäyskattilan yläosaan sijoitetut tuet lepäävät keittokattilan reunojen päällä mäskäyskattilan ollessa keittokattilan sisällä (kuva 2). Tukiin on kiinnitetty keskityspalat, jotka varmistavat mäskäyskattilan asettumisen keittokattilan keskelle.

Kuva 2. Mäskäyskattila keittokattilan sisällä tukien varassa

3.1 Lämmitysvastus

Vastuksen valinnassa tärkeimmät kriteerit olivat valmistusmateriaali, lämmitysteho, sekä mahdollisimman matala teho vastuksen pinta-alan suhteen. Vastuksessa on jokaiselle vaiheelle oma silmukkansa, joista jokainen ottaa 8 A:n virtaa 230 V:n vaihtovirtajännitteellä. Vastuksen kokonaisteho kolmivaihevirralla tähteen kytkettynä on 5500 W, joka riittäisi hieman suuremmankin laitteiston lämmitykseen. Vastus on erityisesti panimokäyttöön tarkoitettu Ultra Low Watt Density -vastus, mikä viittaa vastuksen tehoon sen pinta-alaan suhteutettuna. Tässä vastuksessa se on 13 W/cm². Vastussilmukoiden pintamateriaali on Incoloy 800 -metallia, jota kutsutaan

metallurgiassa superseosmetalliksi. Materiaalin ominaisuuksiin kuuluu erittäin korkea lämmönkesto, sekä korkea vastustuskyky hapettumista ja kulumista vastaan. Tämän tyyppisiä vastuksia käytetäänkin erityisesti sovelluksissa, joissa materiaali altistuu korkeille lämpötiloille sekä erilaisille nesteille. [4, s. 2.] Vastuksen kanta ja kytkentärasia on valmistettu SS304-tyyppisestä ruostumattomasta teräksestä. Vastuksen tiiviste on elintarvikehyväksyttyä korkeisiin lämpötiloihin soveltuvaa silikonia. [5.] Kuvassa 3 on vastuksen kytkentä.

Kuva 3. Vasemmalla lämmitysvastuksen tähtikytkentä keittokattilassa. Oikealla valmis liitäntä kolmivaihepistotulpalla

3.2 Lämpötila-anturi

Lämpötila-anturiksi työhön valikoitui muutamien testien jälkeen digitaalinen lämpötila-anturi DS18B20, joka soveltuu tähän käyttötarkoitukseen hyvin. Anturin tarkkuus on 0,5 °C mittauslämpötilojen ollessa -10...+85 °C, ja sen mittausalue on -55...+185 °C [6, s. 1,5]. Anturin tarkkuus riittää tähän projektiin hyvin, sillä prosessille kriittiset mäskäyslämpötilat jäävät anturin suurimman tarkkuuden piiriin. Lisäksi anturilta saa lämpötilatiedot suoraan digitaalisessa muodossa mikrokontrollerille. Sopivaa anturia valittaessa on testeissä käytetty muutamaa eri DS18B20-anturia, jotta niiden tehdaskalibroinnin yhdenmukainen laatu pystyttiin varmistamaan. Anturi on kiinnitetty

SS304-tyyppisestä ruostumattomasta teräksestä valmistettuun suojakuoreen, jonka kiinnityksen tiivisteet ovat elintarvikehyväksyttyjä korkeisiin lämpötiloihin soveltuvia silikonista valmistettuja o-renkaita (kuva 4).

Kuva 4. Lämpötila-anturi ja läpivienti pumpulle

DS18B20-anturi hyödyntää Dallas Semiconductor Corporationin kehittämää 1-Wire-väyläteknologiaa, jonka avulla anturi kykenee sekä lähettämään että vastaanottamaan tietoa samalla johtimella. Samaan 1-Wire-väylään voidaan asentaa useita antureita, joilla kaikilla on oma yksilöllinen 64-bittinen numerosarja jonka master-laite, eli tässä tapauksessa Arduino, pystyy tunnistamaan. Kaikki väylän anturit toimivat slave-laitteina. Koska antureilla on uniikki numerosarja tunnistusta varten, niitä voisi periaatteessa sijoittaa 1-Wire-väylään rajattomasti, ja kustakin anturista saisi erikseen kutsumalla dataa Arduinoon. [6, s. 5.] Tähän työhön riitti kuitenkin vain yksi anturi, joka on sijoitettu keittokattilan alaosaan, jossa veden lämmitys tapahtuu. Anturin datajohtimen lisäksi anturiin menee käyttöjännitejohdin sekä maajohdin. Datajohtimen ja käyttöjännitejohtimen välille tulee asentaa heikko ylösvetovastus, joka tässä työssä on 4.7 kΩ:n vastus. Heikolla ylösvetovastuksella tarkoitetaan vastusta, jonka kautta tulee vain vähän virtaa, eli mitä pienempi vastus sitä vahvempi ylösveto. Vastusta tarvitaan koska 1-wire-väylä käyttää ”open drain” -lähtöä, jonka tulee ylläpitää loogista

”1”-tilaa silloinkin kun väylä ei ole aktiivinen. Toisin sanoen väylän ”idle”-tila on ”high”-tila, jonka vastuksen kautta tuleva loisjännite nostaa väylään. [6, s. 10] Väylän kommunikointi toimii aina yhteen suuntaan kerrallaan, ja se alkaa alkaa aina alustusviestiketjulla, jossa master-laitteelta tulee reset-pulssi joka vetää Arduinon ulostulon ”0”-tilaan. Tämä vetää samalla väylän ”low”-tilaan kun väylä kytkeytyy maihin. Tämän vaiheen tulee kestää vähintään 480 µs, jonka jälkeen master-laite vapauttaa väylän. Tällöin Arduinon ulostulo asetetaan loogiseen ”1”-tilaan, jolloin väylä nousee jälleen ”high”-tilaan. Edellä mainittujen pulssiviestien jälkeen master-laite on siirtynyt datan vastaanottotilaan (Rx). Kun DS18B20-anturi tunnistaa väylän nousevan reunan, se odottaa 15–60 µs. Tämän jälkeen anturi puolestaan vetää väylän alas 60–240 µs:n ajaksi ilmoittaakseen olevansa väylässä, ja olevansa valmis toimimaan. Myös muu viestintä väylässä noudattaa samanlaista pulsseilla tapahtuvaa tiedonsiirtoa yksi bitti kerrallaan. [6, s.15.]

Anturia voisi käyttää myös pelkästään nolla- sekä datajohtimella. Tällöin anturin toiminta on pelkästään datajohtimeen vastuksen kautta tulevan loisvirran varassa. Anturi lataa datajohtimesta anturissa olevalle kondensaattorille sähkövarausta, jota anturi käyttää muun muassa lämpötiladatan konversion suorittamiseksi ja sen tallentamiseksi anturin EEPROM-muistiin. Kun anturia käytetään kahdella johtimella, sen saatavuus pienenee, koska konversio vie aikaa suurimmalla 12 bitin resoluutiolla jopa 750 ms, ja tänä aikana anturia ei voi kutsua lainkaan datajohtimen ollessa varattuna virransyöttöön. Toimenpiteen jälkeen lämpötila-arvoa voi taas kutsua Arduinosta normaalisti. Tähänkin konfiguraatioon tulee asentaa ylösvetovastus Arduinon 5v-linjan ja datajohtimen välille. Vastuksen arvon kriittisyys kasvaa suuresti loisjännitteellä käytettäessä, mikäli väylässä on useampia antureita, tai anturin kaapeli on pitkä. Tällöin tulee varmistua, että lämpötila-anturi saa varmasti tarpeeksi jännitettä koko 750 ms:n ajan. Loisjännitteellä toimiessaan anturin tulisi saada 3–5,5 V:n tasajännitettä esimerkiksi 12 bitin konversioon vaadittavan 750 millisekuntin ajan. Tänä aikana anturi imee suurimman virran väylästä. 4.7 kΩ:n vastuksella jännite saattaa tuona aikana laskea liian alas, mikäli väylässä on useampi anturi tai pitkä kaapeli. Anturia ei käytetä loisvirralla tässä työssä, koska sitä käytettäessä reaaliaikaisen lämpötilan saanti ei ole yhtä varmaa, eikä työssä ollut tarvetta rajoittaa anturille meneviä johtimia. Tarpeettomaksi havaittiin myös anturin 12 bitin resoluutiolla käyttö, ottaen huomioon anturin oikean tarkkuuden sekä konversioajat (kuva 5). 10 bitin resoluutiolla lämpötila-arvo ilmoitetaan 0,25 celciusasteen portailla, joka on tähän työhön riittävä resoluutio. [6, s. 7.]

(-55°C to +125°C; V_{DD} = 3.0V to 5.5V)

PARAMETER	SYMBOL	CONDITIONS		MIN	TYP	MAX	UNITS
Temperature Conversion Time	t_{CONV}	9-bit resolution	(Note 12)			93.75	ms
		10-bit resolution				187.5	
		11-bit resolution				375	
		12-bit resolution				750	

Kuva 5. DS18B20 lämpötila-anturin konversioajat [6, s. 3]

4 Ohjauskärry

Ohjauskärryn runko on valmistettu hitsaamalla teräksisestä suorakulmaisesta putkesta. Sen korkeus on 113,5 cm, leveys 30 cm ja pituus 50 cm. Kärryssä on neljä pyörää liikuttamista varten. Etupyörät on mahdollista lukita, jolloin kärry pysyy paikoillaan laitteen ollessa käytössä. Ohjauskärryn mitat on suunniteltu tarvittavien kytkentärasioiden, kolmivaiheliitännän, sekä pumpun viemän tilan perusteella. Kärryn suunnittelussa myös tukevuus oli tärkeä tekijä käyttöturvallisuuden kannalta, joten siihen on hitsattu tukiraudat kärryn ala- ja yläpalkkien välille. Lisäksi kärry on mitoitettu pituudeltaan riittäväksi suhteessa kärryn korkeuteen.

4.1 Kojeistus ja kytkennät

Ohjauskärryssä on kolme kytkentärasiaa, jotka on yhdistetty tiivistetyillä läpivienneillä. Rasioiden lisäksi kärryssä on kolmivaihepistorasia sekä vierrepumppu putkistoineen. Kojeistus on suunniteltu siten, että ylimpään kytkentärasiaan tulee verkkoliitäntä, ja alimmasta kytkentärasiasta lähtee kaapelit pumpulle, lämmitysvastukselle sekä lämpötila-anturille. Ohjauskärryn virta tulee kolmen metrin 5 x 2,5 mm²:n kumikaapelilla 3 x 16 A:n kolmivaihepistokkeesta. Kaapelin sisääntulo ohjauskärryn ylimmässä kytkentärasiassa on varustettu tiivistetyllä vedonpoistolla ja se on tuettu runkoon asennetulla kiinnikkeellä, jottei painava kaapeli rasittaisi rasian liitäntää.

Ensimmäisessä kytkentärasiassa syöttökaapeli on kytketty C-tyypin 3 x 10 A:n johdonsuojakatkaisijaan, jonka kautta virta kulkee A-tyypin 40 A:n vikavirtasuojalle. Maadoitusjohtimella on ensimmäisessä kytkentärasiassa oma maadoitusriviliitin, jonka

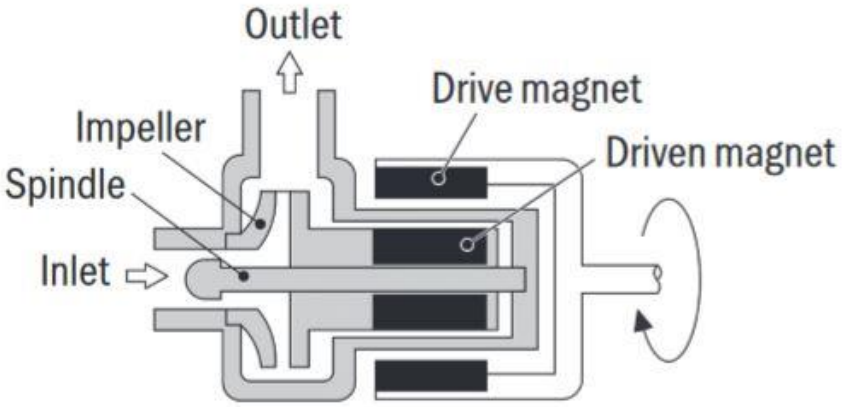
lisäksi ohjauskärryn runko on maadoitettu kahdesta kohdasta 2,5 mm²:n suojamaajohtimilla. Vikavirtasuojasta vaiheet menevät seuraavassa kytkentärasiassa sijaitsevalle puolijohdereleelle, jolla keittokattilan vastuksille menevää virtaa ohjataan. Puolijohdereleeksi on valittu PQLYT PQSSR-3DA -puolijohderele, jonka kautta voi ohjata virtaa enintään 20 A per vaihe. Tässä työssä vastuksille menevä virta on vain 8 A per vastus. Puolijohdereleen ohjausjännite on 3 – 32 V:n tasavirtaa, joten sitä voi ohjata suoraan Arduinon digitaalisella ulostulolla, josta tulee 5 V:n tasajännitettä ”high”-tilassa. Puolijohdereleen sisäinen vastus sen ollessa kytkeytyneenä on melko korkea ja tämä muuttuu lämmöksi, joten releeseen on asennettu jäähdytyssiili siirtämään tuotettu lämpö kytkentärasioiden ulkopuolelle [7]. Jäähdytyssiili on asennettu kytkentäkotelon taakse, ja releen pohjan sekä siilin väliin on asennettu metallilevy joka on tiivistetty rasian pohjaa vasten. Siilin ja releen pohjan pintoihin on laitettu hopeatahnaa edistämään lämmönsiirtoa. Siili on lisäksi maadoitettu.

Kolmannen kytkentärasian tarvitsema vaihtovirta on otettu ensimmäisestä vaiheesta vikavirtasuojan jälkeen. Vaihe menee kolmannen rasian riviliittimille nolla- ja suojamaajohtimen kanssa, joista syötetään virtaa tasavirtalähteelle sekä vierrepumpulle. Ensimmäisen vaiheen ottama kokonaisvirta kasvaa tällä kuormalla maksimissaan 8,59 A:n suuruiseksi. Kolmannessa kytkentärasiassa sijaitsee näiden lisäksi releohjauskortti, Arduino sekä Bluetooth-moduuli. Virtalähteessä on hienosäätötrimmeri, jolla jännite on säädetty 11,9 V:iin, koska tästä syötetään Arduinolle käyttöjännitettä ja suurin sille suositeltava käyttöjännite on 12 V. Relekortissa on 2 relettä, joissa on 5 V:n kelat ja ne kykenevät ohjaamaan 230 V:n vaihtovirtajännitettä enintään 10 A:n virralla. Releohjauskortin ohjauksen sisääntulot ovat invertoituja, mikä tuli ottaa huomioon ohjelmointivaiheessa. Releen kelan vaatima kytkentäjännite saadaan suoraan Arduinon digitaalisesta ulostulosta.

4.2 Vierrepumppu

Vierrepumpuksi työhön on valittu kotipanimokäyttöön tarkoitettu Brewferm Magnetic Pump’in 15 -pumppu, koska siinä on sopivalla halkaisijalla olevat liitännät sekä pumpun virtaama ja nostovoima ovat laitteelle sopivat. Magneettipumpun etuna on myös se, että ulostulovirtausta voi kuristaa venttiilillä ilman, että pumppu vahingoittuisi. Virtauksen säätö on ehdottoman tärkeää mäskäyksessä, jossa tulee varmistua, että mäskäyskattila saa imettyä lävitsensä kaiken sinne virtaavan veden eikä ylivuotoa

tapahdu. Mikäli ylivuotoa tapahtuu, niin keittokattilan nesteen sekaan pääsee mallasrouhetta, joka lopulta voi tukkeuttaa pumpun suodattimet. Pumpun maksimilämmönkesto on 90 °C, ja Arduinon ohjelma sammuttaa sen automaattisesti kun lämpötila on yli 85 °C. [8, s. 7.] Pumppu jaksaisi pumpata nesteen 1,5 metrin korkeudelle, mikä riittää mainiosti, koska tässä työssä nestettä tarvitsee nostaa vain 72 cm:n korkeuteen pumpusta. Pumpun maksimivirtaus on 8 – 12 l/min. [9, s. 1.] Kuvassa 6 näkyy magneettipumpun etuosan läpileikkaus.



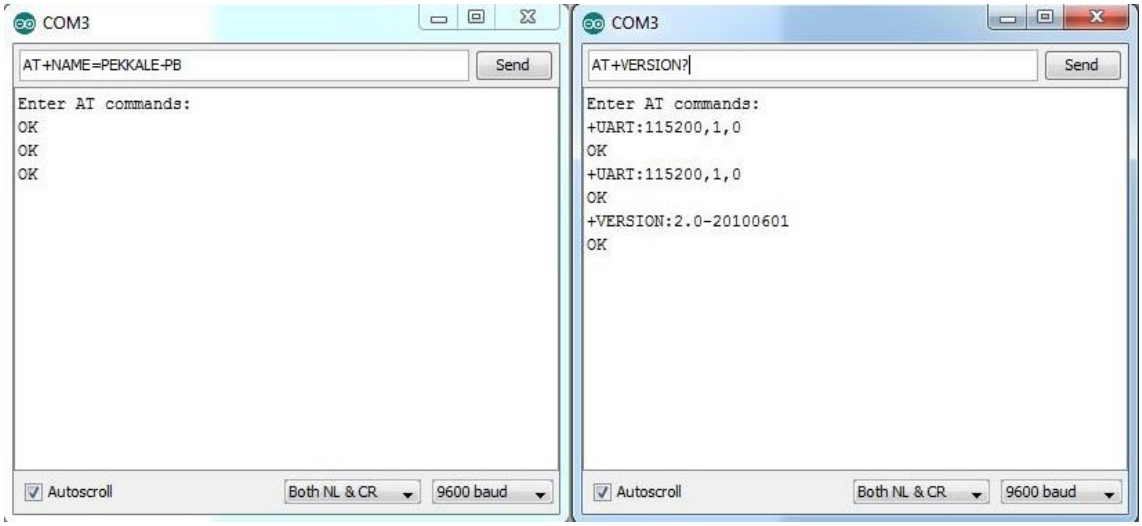
Kuva 6. Magneettipumpun etuosan läpileikkaus. Itse sähkömoottori on kiinni drive magnetin akselissa, jolloin myös siipipyörän magnetoitu akseli pyörii drive magnetin liikkeen mukana [8, s. 8]

4.3 Bluetooth-moduuli

Työssä käytetään Arduinon sarjaväylään yhteensopivaa HC-05 Bluetooth -moduulia. Moduuli käyttää Bluetooth 2.0 -rajapintaa, ja sen kantama on 10 metriä ympäristössä, jossa ei ole häiriöitä Bluetoothin käyttämillä radiotaajuuksilla [10].

Ennen Bluetooth-moduulin varsinaista käyttöönottoa sille on määritelty haluttu sarjaväylän nopeus. Nopeus tulee asettaa samaksi myös Arduinon ohjelmassa, jotta data tulisi ymmärrettävässä muodossa perille. Tehdasasetuksissa tämä oli määritetty 9600 bittiin sekunnissa, joka on melko hidas nopeus. Väylä kykenee toimimaan paljon nopeamminkin, joten väylänopeutta nostettiin. Tämän lisäksi moduulin nimi sekä PIN-koodi vaihdettiin. Näiden muutoksien tekemiseksi Bluetooth-moduuli tulee saattaa AT-tilaan, jossa voi muuttaa moduulin tehdasasetuksia, sekä tehdä kyselyitä moduulilta.

Arduino tarvitsee ohjelman joka mahdollistaa keskusteluyhteyden Bluetooth-moduulin kanssa sen ollessa AT-tilassa. Tämän ohjelman pyöriessä käyttäjä voi kirjoittaa haluttuja AT-komentoja Arduinon sarjaväylämonitoriin, sekä saada sinne moduulilta takaisin vastauksia (kuva 7). Arduinon ohjelma jolla pääsee Bluetooth-moduulin AT-tilaan määrittää aluksi sarjaväylän tiedonsiirtonastat, sekä muuttaa niiden sarjaväylän nopeuden 38 400 bittiin sekunnissa, jonka moduulin AT-tila vaatii. Ohjelma ladataan Arduinon muistiin, jonka jälkeen Bluetooth-moduulin KEY-nasta tulee kytkeä ohjelmassa määritettyyn nastaan, jonka ohjelma alkaessaan saattaa ”high”-tilaan. Ohjelman latauksen jälkeen Bluetooth-moduulista tulee irroittaa virransyöttöjohto, sekä Arduinosta USB-kaapeli jolloin se ei saa enää virtaa. Tämän jälkeen tulee kytkeä USB-kaapeli Arduinoon takaisin, jolloin ohjelma lähtee käyntiin ja KEY-nasta siirtyy ”high”-tilaan. Kun ohjelma on käynnissä, niin Bluetooth-moduulin virransyöttö voidaan kytkeä takaisin jolloin se siirtyy AT-tilaan. Moduuli vilkuttaa tästä merkiksi lediään 2 sekunnin välein. [11, s. 1-2.] Arduinon AT-mode-ohjelman loop()-funktiossa ohjelma lukee Arduinon sarjaväylämonitoriin lähetetyt tiedot, lähettää ne Bluetooth-moduulille, lukee Bluetooth-moduulista tulevat tiedot sekä lähettää ne Arduinon sarjaväylämonitoriin. AT-komennot löytyivät internetistä moduulin mallin perusteella [11].



Kuva 7. Vasemmanpuolimmaisessa ikkunassa määritetään Bluetooth-moduulille uusi nimi. Oikeanpuolimmaisessa ikkunassa on määritetty väylän tiedonsiirtonopeus 115 200 bittiin sekunnissa, sekä pysäytysbitti yhteen, jolloin se on moduulin AT-komentojen dokumentin mukaisesti kaksi bittiä sekä parity eli tarkistusbitti nollaksi, joten se on pois käytöstä. Lisäksi moduulin versio tarkistettiin

5 Arduinon ohjelmointi

Arduino on avoimeen laitteeseen perustuva mikrokontrolleri/elektroniikan kehitysalusta. Arduinon oma ohjelmointikieli perustuu C++:aan, ja sen ohjelmiin voi liittää C++ -kielellä kirjoitettuja kirjastoja. Arduinosta on olemassa lukuisia eri malleja eri valmistajien tuottamina. [12.] Työssä on käytössä Arduino Mega -mikrokontrolleri.

Tyypillisessä Arduinon ohjelmassa määritellään aluksi ohjelmaan sisällytettävät kirjastot, kehitysalustasta käytettävien kytkentänastojen numerot sekä niiden nimet (kuva 8). Näiden lisäksi yleensä määritellään myös muita ohjelmassa tarvittavia muuttujia (kuva 8). Seuraavaksi tulee ensimmäinen pääfunktio: setup()-funktio (kuva 9), jossa alustetaan laitteen asetukset. Tämä pääfunktio ajetaan vain kerran kun ohjelma käynnistetään. Ohjelmassa tulee tämän jälkeen useimmiten toinen pääfunktio: loop(), jonka sisältämät funktiot toistetaan ohjelman mukaisesti mikrokontrollerin virran sammuttamiseen asti. [12.]

```
#include <PID_v1.h>
#include <OneWire.h>
#include <DallasTemperature.h>

#define ONE_WIRE_BUS 10
#define SSR_PIN 2
#define PUMP_PIN 3

OneWire ourWire(ONE_WIRE_BUS);

//
DallasTemperature sensors(&ourWire);



double Setpoint, Input, Output;


double Kp=50, Ki=0.85, Kd=10
;
PID myPID(&Input, &Output, &Setpoint, Kp, Ki, Kd, DIRECT);

int Pulssiikkuna = 5000;
unsigned long Aloitusaika;
byte Tulevatieto;
//
```

Kuva 8. Määrittelyt ohjelman alussa

```
void setup()
{
  pinMode(SSR_PIN, OUTPUT);
  pinMode(PUMP_PIN, OUTPUT);
  digitalWrite(PUMP_PIN, HIGH);
  Serial.begin(115200);

  Serial1.begin(115200);
  Aloitusaika = millis();

  //Ohjelman käynnistyessä setpoint asetetaan 0 asteeseen.
  Setpoint = 0;

  //Kerrotaan PID:lle ulostulon minimi ja maksimi -arvot.
  //Tämän PID suhteuttaa pulssi-ikkunan kokoon
  myPID.SetOutputLimits(0, 100);


  myPID.SetMode(AUTOMATIC);
  sensors.begin();
  sensors.setResolution(10);
}
```

Kuva 9. Setup()-funktio ohjelmassa

5.1 Lämpötiladata

Ohjelmointi on aloitettu kirjoittamalla ohjelma, jossa haetaan lämpötilatietoja lämpötila-anturilta, muunnetaan ne celciusasteiksi (kuva 10) ja kirjoitetaan lämpötilatieto Arduinon sarjaväylään.

```
sensors.requestTemperatures();
Input = sensors.getTempCByIndex(0);
```

Kuva 10. Lämpötilan haku, muunnos ja muunnoksen määrittäminen PID-säätimen Input-muuttujaan

DS18B20 käyttää 1-Wire-väylää, joten ohjelmaan sisällytettiin Onewire C++ -kirjasto. Lisäksi siihen on sisällytetty Dallastemperature C++ -kirjasto, joka muuntaa haetun lämpötiladatan celsiusasteiksi. Muunnostyön voi myös vaihtoehtoisesti kirjoittaa suoraan Arduinon koodiin ilman Dallastemperature-kirjastoa 1-Wire-väylästä saatavalla datalla. Tällä ohjelmalla suoritettiin testejä eri lämpötila-antureilla, jossa anturit asetettiin vuorotellen eri lämpöisiin nesteisiin ja niiden tuloksia vertailtiin mittaamalla nesteen lämpötila samanaikaisesti kahdella nestelämpömittarilla, jotta varmistuttiin anturien yhteneväisistä lukemista. Lukemat digitaalisilla antureilla olivat yhteneväisiä,

mutta nestelämpömittarin näyttämään verrattuna pieniä heittoja ilmeni muutamilla lämpötila-alueilla. Tämä on korjattu ohjelmoimalla lämpötilankorjaus, joka tapahtuu heti lämpötilan haun jälkeen ohjelmakierrossa (kuva 11). Tässä vaiheessa on testattu myös lämpötila-anturista saatavan datan eri resoluutiot ja anturi on asetettu toimimaan 10 bitin resoluutiolla.

```
 if (Input > 47 < 58)
{ Input = Input + 0.75; }

 if (Input > 41 < 47)
{ Input = Input + 0.5; }
```

Kuva 11. Korjaukset anturin ilmoittamiin lämpötiloihin eri lämpötila-alueilla if-funktioiden avulla

5.2 PID-kirjasto

Seuraava vaihe oli PID-kirjaston lisääminen ohjelmaan. Kirjaston lisäyksen jälkeen on määritetty muuttujat, alustavat viritysarvot sekä näiden formaatit. Nämä ovat input-, setpoint-, output-, Kp-, Ki-, ja Kd-muuttujat. PID-ohjelman input-muuttuja määritettiin tulemaan Dallastemperature-kirjaston avulla saatavasta lämpötilamuunnoksesta, jonka jälkeen tapahtuu vielä muuttujan arvon hienosäätö. Ohjelman ulostulo on skaalattu välille 0 - 100, joka kuvastaa lämmitysvastuksille syötettävän tehon määrää. Koska PID-säätimellä ohjataan tässä tapauksessa puolijohderelettä Arduinon digitaalisella ulostulolla, eikä vastuksen suoraa tehoa, niin ohjelmaan tarvittiin tapa jolla pulssitetaan vastuksille syötettävää virtaa suhteessa PID-säätimen ulostuloarvoon. Tällöin säätimen ulostuloarvo suhteutetaan ohjelmassa määritettyyn pulssi-ikkunaan, joka on asetettu 5000 ms:n pituiseksi. Tämä tarkoittaa sitä, että mikäli säädin laskee ulostuloarvoksi 50, niin ohjelma pitää lähtöä puolijohdereleelle 2500 ms ”high”-tilassa ja 2500 ms ”low”-tilassa. Tällöin vastus toimii pulssi-ikkunan kokoon suhteutettuna puolella teholla. Tähän toimintaan ohjelmassa käytetään kahta eri laskuria, jotka määrittävät pulssin koon (kuva 12).

```
{
  myPID.Compute();
   }
  }
  else{
    myPID.Compute();
    unsigned long Nykyhetki = millis();
    if(Nykyhetki - Aloitusaika>Pulssiikkuna)
    {
      Aloitusaika += Pulssiikkuna;
    }
    if((Output*(Pulssiikkuna/100)) > Nykyhetki - Aloitusaika) digitalWrite(SSR_PIN,HIGH);

    else digitalWrite(SSR_PIN,LOW);
  }
```

Kuva 12. PID-laskennan aloitus, sekä pulssin määrittely

Ohjelma sisältää ehtoja joilla PID-laskenta käynnistetään. Mikäli asetusarvo vähennettynä haetusta ja kalibroidusta lämpötilasta on suurempi kuin 20 °C, niin ohjelma kytkee puolijohdereleelle menevän lähdön ”high”-tilaan, eikä PID-laskentaa käytetä ollenkaan. Jos taas vertailun tulos on pienempi kuin 21 °C, niin PID-laskenta käynnistetään (kuva 13). Tämä vertailu tehdään joka ohjelmakierrolla.

```
   if((Setpoint - Input)>20){ /
    digitalWrite(SSR_PIN,HIGH);
    if ((Setpoint - Input)<21)
{
  myPID.Compute();
   }
```

Kuva 13. PID-laskennan käynnistyksen ehdot

5.3 Arduinoon tuleva data

Android-laitteesta Arduinoon Bluetooth-moduulin kautta tuleva data kulkee mikrokontrollerin ensimmäisen sarjaväylän 0->RX0 -nastaan. Ohjelma tarkistaa jokaisella kierrolla mikäli dataa on tullut Androidista käsiteltäväksi ja se suorittaa enintään yhden saadun käskyn per ohjelmakierto. Mikäli käskyjä ehtii ohjelmakierron aikana tulla enemmän, niin ne jäävät jonoon odottamaan seuraavia ohjelmakiertoja. Mikäli käskyjä ei ole tullut, niin vertailua ei suoriteta ollenkaan. Jos käskyjä on tullut,

niin ohjelma tallentaa ensimmäisen jonossa olevan käskyn Tulevatieto -nimiseen muuttujaan. Tämän jälkeen ohjelma suorittaa vertailun, jossa vertaillaan muuttujaan asetettua tietoa ohjelmasta löytyviin komentoihin switch()-komennon avulla. Mikäli saatua tietoa vastaava komento löytyy, niin ohjelma suorittaa sen ja siirtyy eteenpäin ohjelmakierrossa. Switch()-komennon alla on eri vertailutilanteet case:-komennoilla eroteltuina. Komento ajetaan tiedon täsmätessä break;-komentoon asti, jolloin kyseisen ohjelmakierron vertailu on suoritettu loppuun (kuva 14). Ennen vertailun aloittamista ohjelma tarkastaa mikäli tullut arvo on alle 101, sillä Androidista tulevia alle 101:n suuruisia arvoja käytetään muuttamaan PID-säätimen asetusarvoa (kuva 14). Kaikki muut Arduinoon tulevat komennot ovat suuruudeltaan yli 100.

```
if (Serial.available()>0) {Tulevatieto =Serial.read(); Serial.println(Tulevatieto);}

if (Tulevatieto < 101) {Setpoint = Tulevatieto;}
 switch(Tulevatieto){

   case 109:
   digitalWrite(PUMP_PIN, LOW);
   break;



   case 110:
   digitalWrite(PUMP_PIN, HIGH);
   break;
   //Pumppu päälle

   case 111:
   Setpoint = Setpoint + 5.00;
   break;
   //Setpointin muutos
```

Kuva 14. Arduinoon tulevan datan määritys muuttujaan, tarkistus mikäli kirjoituskentästä on lähetetty asetusarvo sekä painonappien komentojen vertailu ohjelmasta löytyviin komentoihin

5.4 Arduinosta lähtevä data

Arduinosta Androidiin lähtevä data kulkee mikrokontrollerin toisen sarjaväylän 18<-TX1 -nastasta Bluetooth-moduuliin RXD -nastaan. Dataa lähetetään jokaisella ohjelmakierrolla ja se sisältää viimeisimmän lämpötilatiedon, PID-säätimen asetusarvon sekä PID-säätimen ulostulon arvon (kuva 15). Lähetettävän datan ensimmäinen merkki on aina ”<”-merkki, ja viimeinen merkki on aina ”>”-merkki, jotta

Android-sovellus käsittelisi saapuvan datan aina alusta loppuun asti. Aloitusmerkin jälkeen lähetetään lämpötilatieto jota seuraa ”|”-merkki. Sen tehtävänä on erotella lämpötilatieto erotusmerkin jälkeen tulevasta asetusarvosta. Samaa merkkiä käytetään myös asetusarvon ja ulostuloarvon erottamiseksi. Tietojen lähettämisen jälkeen ohjelmassa on 700 millisekunnin tauko, jotta tietojenvaihto toimii synkronoidusti Android-applikaation kanssa. Tämä on ohjelmakierron suurin hidastaja, mutta ohjelman toimintaan tällä viiveellä ei ole juurikaan vaikutusta. Data siirtyy alle sekunnissa mikä on laitteistolle varsin riittävä nopeus.

```
Serial1.print('<'); //
Serial1.print('|');
Serial1.print(Input); //
Serial1.print('|');
Serial1.print(Setpoint);
Serial1.print('|');
Serial1.print(Power);
Serial1.print('|');
Serial1.print('>');
delay(700);
```

Kuva 15. Arduinosta lähtevä data

6 PID-säädin

PID-säätimen tarkoitus tässä työssä on saattaa laitteiston veden lämpötila käyttäjän valitsemaan lämpötilan asetusarvoon. Tämä tapahtuu ohjaamalla lämmitysvastuksille menevää virtaa puolijohdereleellä, jota PID-säätimen ulostulo ohjaa. Säätimen prosessimuuttuja on lämpötila ja se saa tiedon lämpötila-anturista. Lämpötilatietoa verrataan valittuun lämpötilan asetusarvoon, ja näiden arvojen erosuuretta kutsutaan säätövirheeksi e. PID-säädin käyttää säätövirheen arvoa laskeakseen korjausarvon, jolla ohjataan säätimen ulostuloa. Ulostulon laskemiseen PID-säädin käyttää kolmea termiä: suhdetta, integraalia sekä derivaattaa. P-termi, eli suhde tarkkailee prosessin nykyhetkeä ja sen arvo on suhteessa sen hetkiseen säätövirheeseen. I-termi, eli integraali tarkkailee prosessin historiaa ja sen arvo on integraali tapahtuneista säätövirheistä. D-termi, eli derivaatta ennustaa prosessin tulevia säätövirheitä tarkkailemalla prosessin säätövirheen muutosnopeutta, ja sen arvo on derivaatta muutoksista säätövirheessä. [13, s. 3–6.] Näiden kolmen termin positiivisia kertoimia

kutsutaan PID-säätimen viritysarvoiksi. Ne ovat Kp, Ki ja Kd [14]. PID-säätimen ohjelmaa ajetaan Arduinossa tasaisesti, ja sen toimintaa ohjelmassa voidaan esittää kaavalla:

$$u = K_c \left(e(t) + \frac{1}{T_i} \int_0^t e(t) d\tau + T_d \frac{dPV}{dt} \right) \quad (1)$$

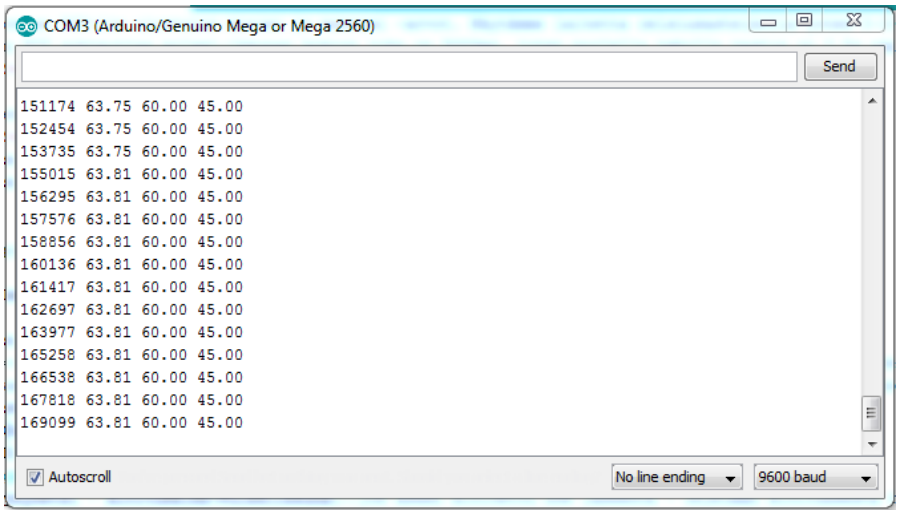
jossa

- u= Säätimen ulostulo
- K_c=Ohjauksen vahvistus ($K_p = 0{,}6K_c$)
- e(t)=Virhe ajassa t
- T_i=Integraaliosan aikavakio
- T_d=Derivaattaosan aikavakio
- PV=Prosessin muuttuja, eli lämpötila
- dt=Tahti jolla PID-ohjelmaa ajetaan

ja jonka aikavakioiden muunnoskaavat viritysarvoiksi ovat

- $K_I = \frac{K_p}{T_i}$
- $K_D = K_p T_D$

Säätimen viritystä varten laitteiston ohjelmasta on muokattu versio, jonka avulla pystyy suorittamaan säätimelle erilaisia viritystestejä. Ohjelmaan on jätetty vain PID-kirjasto sekä lämpötiladatan haku. Ohjelma ilmoittaa lämpötilan, asetusarvon, ohjelman käynnistämisestä kuluneen ajan sekä säätimen ulostulon arvon. Ohjelmaan on luotu muuttuja joka mittaa kulunutta aikaa millisekunteina ohjelman käynnistyksestä sekä siihen on lisätty 700 ms:n viive, jottei sarjaväylämonitori tukkeutuisi datasta, ja jotta viive olisi sama kuin laitteessa normaalisti käytettävässä ohjelmassa. Muut tarvittavat arvot löytyivät jo valmiiksi ohjelmasta. Datan jälkikäsittelyä helpottaa tietojen jakaminen välilyönneillä sekä rivierottelu. Ohjelma tulostaa tiedot Arduinon sarjaväylämonitoriin mistä ne saa kopioitua analysointia varten (kuva 16).



Kuva 16. Viritysdata Arduinon sarjaväylämonitorissa

Ohjelmasta saatavan datan avulla pystyy Excel- sekä Logger Pro -ohjelmien avulla määrittämään säätimen viritysarvot Ziegler-Nichols menetelmää hyödyntämällä.

6.1 Ziegler-Nichols menetelmät

Ziegler-Nichols menetelmät perustuvat joko systeemin askelvasteen, tai kriittisen värähtelyn tutkimiseen. Menetelmien avulla saadaan suuntaa antavat viritysarvot säätimelle, ja useimmissa tapauksissa ohjausta on vielä tarpeen hienosäätää. [15, s. 17-20.] Työssä käytettiin PID-säätimen viritysparametrien saamiseen askelvastemenelmää. Menetelmässä on tarkoitus ensin saattaa systeemi tasapainotilaan, jonka jälkeen ohjaukseen tehdään askelmainen muutos. Testistä saatuja tietoja analysoimalla voidaan määrittää tarvittavat arvot säätimen viritysarvojen laskemiseksi. Testit suoritettiin automaatilla säätäen asetusarvoa sekä manuaalilla säätäen ulostuloa. Tärkeää testeissä oli että ne suoritettiin tyypillisillä mäskäyslämpötila-alueilla ilman systeemiin kohdistuvia häiriöitä. Testien jälkeen säätimen toimintaa vertailtiin saaduilla viritysarvoilla.

6.1.1 Askelvastekoe asetusarvon muutoksella

Ensimmäinen testi suoritettiin aloitusasetusarvolla 60 säätimen ollessa automaatilla. Lämpötilan tasaannuttua ohjelma käynnistettiin uudestaan, jolloin ohjelma nosti asetusarvon 66 -asteeseen 100 sekunnin kuluttua ohjelman aloituksesta. Tähän

käytettiin samaa ajastinta joka tulostaa sarjaväylämonitoriin tiedon ohjelman kestosta. Kun lämpötila oli asettunut lopulliseen arvoonsa, niin Arduinon sarjaväylämonitoriin tulleet tiedot kopioitiin talteen ja siirrettiin Exceliin, jossa tiedot jaettiin omille sarakkeilleen. Tämän jälkeen tiedot siirrettiin Logger Pro -ohjelmaan datan analysointia varten (kuva 17). Analysointi aloitettiin sovittamalla tangentti vasteen jyrkimpään kohtaan, jonka avulla saatiin nousuaika (Tn). Kuollut aika (Tk) laskettiin asetusarvon muutosajasta ensimmäiseen nousseeseen lämpötila-arvoon (kuva 18). Aikavakio (T) saatiin laskemalla nousuajan alkamiskohdasta kohtaan, jossa noususta oli tapahtunut 63%. Vahvistus (K) laskettiin lämpötilan muutoksen suhteesta asetusarvon muutokseen. Saadut arvot sijoitettiin muunnoskaavataulukoihin. Arvot on laskettu käyttämällä ensin nousuaikaa ja sitten aikavakiota.

Taulukko 1. Ziegler-Nichols kaavat laskettaessa nousuajalla

	Kp	*Ti*	*Td*
PID	1,2Tn/(Tk*K)	2Tk	0,5Tk

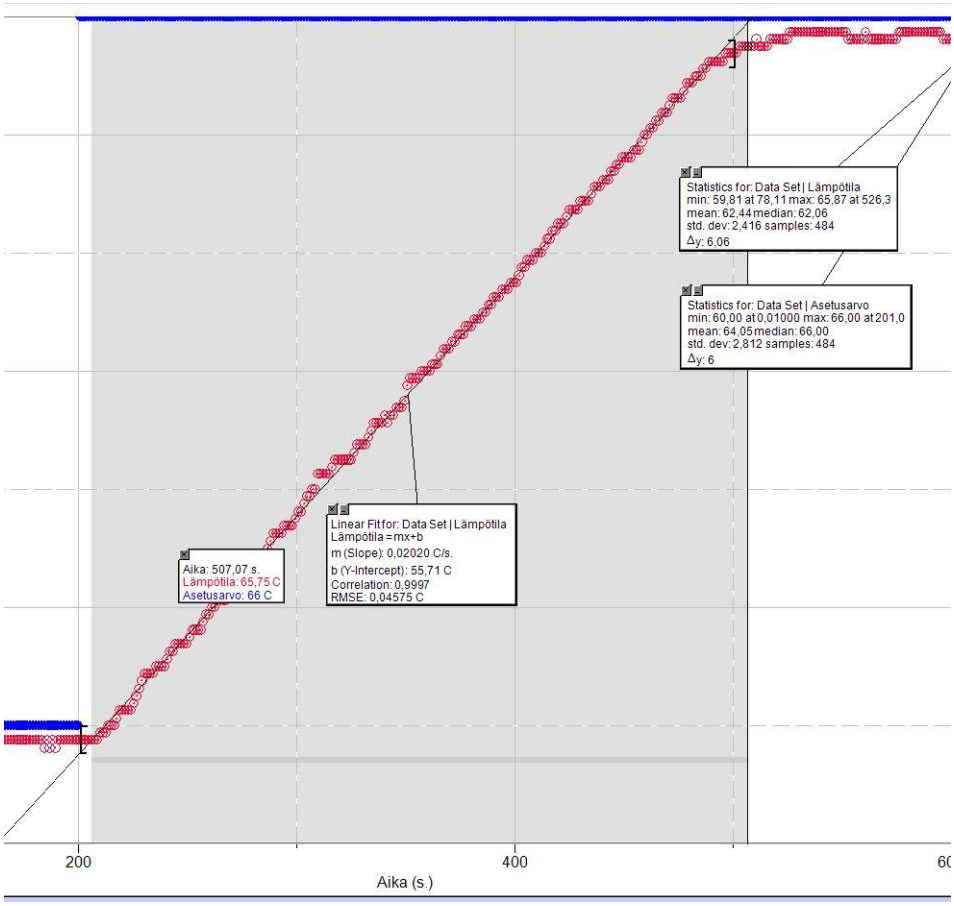
Taulukko 2. Ziegler-Nichols kaavat laskettaessa aikavakiolla

	Kp	*Ti*	*Td*
PID	1,2/K*(T/Tk)	2Tk	0,5Tk

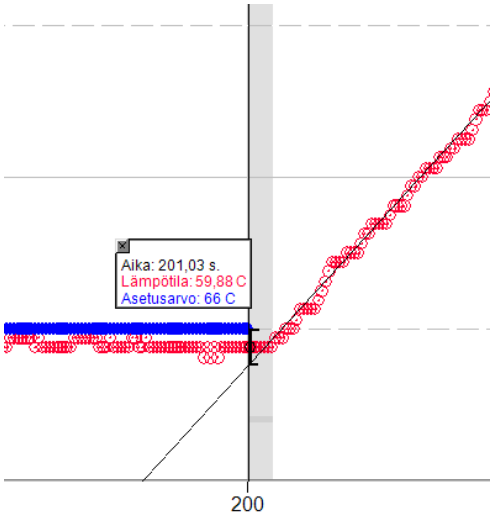
Taulukko 3. Ensimmäisestä askelvastekokeesta saadut viritysarvot

Kuollut aika Tk (s.):	6,4
Nousuaika Tn (s.):	302,88
Vahvistus K:	1,01
Aikavakio T (s.):	201,50

Nousuajalla		Aikavakiolla	
Kp:	56,23	Kp:	37,41
Ki:	4,39	Ki:	2,92
Kd:	179,9	Kd:	119,71



Kuva 17. Tietojen analysointi Logger Pro -ohjelmassa, jossa sininen esittää asetusarvoa ja punainen lämpötilaa. Harmaalla merkitty alue esittää nousuaikaa, joka alkaa ensimmäisestä nousseesta lämpötila-arvosta sekä loppuu asetusarvon ja tangentin leikkauspisteeseen lopullisessa asetusarvossa



Kuva 18. Harmaalla merkitty aika asetusarvon muutoksesta lämpötilan nousuun edustaa kuollutta aikaa

6.1.2 Askelvastekoe ulostuloarvon muutoksella

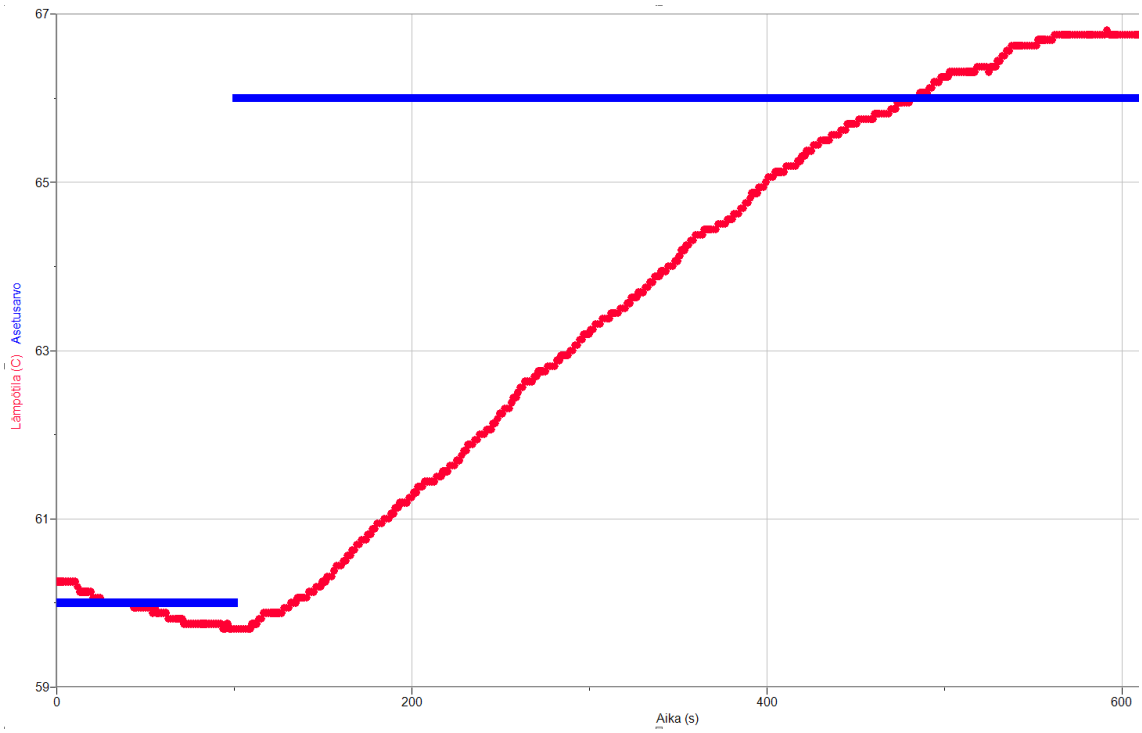
Testissä suoritettiin ulostuloarvon muutos 30:stä 45:een säätimen ollessa manuaaliajolla. Tänä aikana veden lämpötilassa tapahtui 8 °C:n muutos 61 °C:sta 69 °C:seen, mikä kesti 2 tuntia ja 42 minuuttia. Prosessin hitauden vuoksi testaus osoittautui hyvin ongelmalliseksi, eikä ulkoisten häiriötekijöiden vaikutusta testiin voitu täysin sulkea pois. Haasteellista testissä oli myös tangentin asettaminen nousuajan laskemiseksi, suuren mittausotannan ja erittäin loivan nousun takia. Lisäksi testissä kuollut aika kasvoi todella suureksi, jolloin derivaattatermin viritysparametri Kd kasvoi aivan liian suureksi. Lopputuloksena on, että testistä saadut viritysarvot ovat käyttökelvottomia ohjaimen säätämiseen.

Taulukko 4. Toisesta askelvastekokeesta saadut viritysarvot

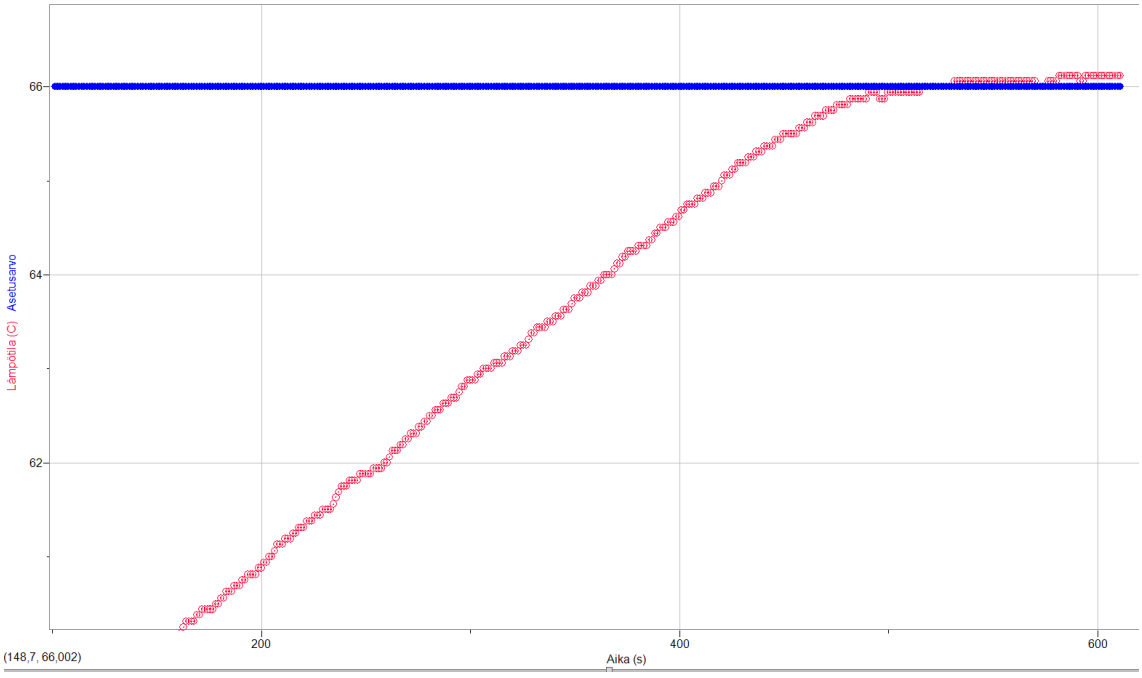
Kuollut aika Tk (s.):	28		
Nousuaika Tn (s.):	5142,2		
Vahvistus K:	0,73		
Aikavakio T (s.):	3005		
Nousuajalla		Aikavakiolla	
Kp:	160,88	Kp:	176,4
Ki:	2,8	Ki:	3,15
Kd:	2251,2	Kd:	2469,6

6.2 Viritysarvojen vertailu

Arvojen vertailussa on käytetty nousuajalla laskettuja viritysarvoja. Ensimmäisestä testistä saadut viritysarvot ylittivät lämpötilan asetusarvon, jonka jälkeen lämpötila tasaantui erittäin hitaasti kohti asetusarvoa (kuva 19). Arvoja hienosäätämällä toiminta stabiloitui hieman, mutta lopulta parempi tulos syntyi virittämällä säädin uudelleen. Säädin on viritetty uudelleen kasvattamalla Kp-arvoa riittävään nousuun, jonka jälkeen Ki- ja Kd-arvoja on nostettu hyvin maltillisesti. Virityksessä on käytetty säätimen ulostuloarvoa kertomaan lämmityspulssin pituus lämpötilan lähestyessä asetusarvoa, ja tehtyjä muutoksia on arvioitu tämän perusteella. Näillä viritysarvoilla vesi saavuttaa asetusarvon 66 samasta lähtölämpötilasta hitaammin kuin askelvastekokeesta saaduilla arvoilla, mutta se on erittäin stabiili eikä juurikaan ylitä asetusarvoa (kuva 20).



Kuva 19. Kuvaajassa punainen väri on lämpötilan muutos ensimmäisestä askelvastekokeesta saaduilla viritysarvoilla. Sininen väri esittää asetusarvoa



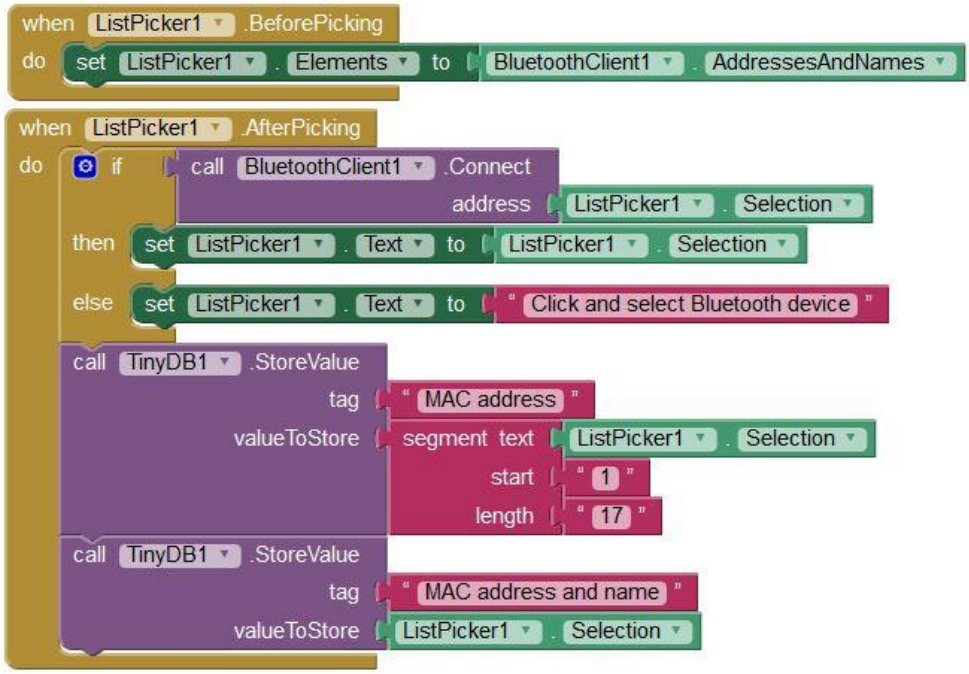
Kuva 20. Uudet käytössä olevat viritysarvot, joilla asetusarvon suurin ylitys oli 0,12 °C kun asetusarvo oli 66 °C

7 Android-ohjelma

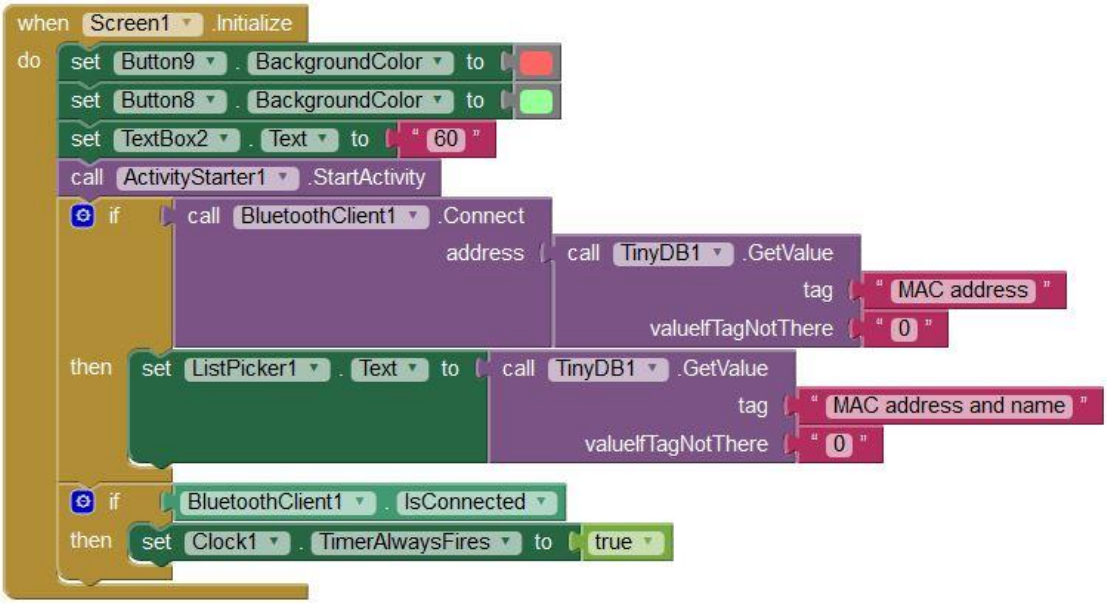
Android-ohjelma on kehitetty MIT App Inventor 2 -web-alustalla. MIT App Inventor 2 on avoimen lähdekoodin web-applikaatio, joka on alun perin Googlen kehittämä, mutta nykyisin Massachusetts Institute of Technologyn ylläpitämä Android-applikaatioiden kehittämisohjelma. Ohjelmointi applikaatiolla tapahtuu logiikkalohkojen avulla, joita voidaan liittää toisiinsa. [16.]

7.1 Ohjelman komponentit

Applikaatioon voidaan lisätä näkyviä sekä näkymättömiä komponentteja. Näkymättömiä komponentteja ohjelmassa on esimerkiksi: Bluetooth-server, Bluetooth-client, ajastimet, äänet, tietokanta ja tekstintallennus. Näkyviä komponentteja ovat taas kaikki graafisessa käyttöliittymässä näkyvät komponentit. Kehitysalustassa on kaksi näkymää: designer- sekä block-näkymät. Applikaation kehittäminen alkoi lisäämällä designer-näkymään tarvittavat komponentit, joiden avulla Bluetooth-yhteyden luonti tapahtuu. Tämän lisäksi applikaatioon on lisätty TinyDB-komponentti, joka toimii applikaatiossa tietokantana. Tietokantaa käytetään tallentamaan jo muodostetut yhteydet, jotta applikaatio yhdistäisi avautuessaan aikaisemmin valittuun Bluetooth-laitteeseen automaattisesti (kuva 22). Tietokantaan tallennetaan tiedot tag-tunnisteilla, joilla niitä voi myöhemmin hakea tietokannasta. Tietokannan tiedot tallentuvat applikaation salattuun kansioon puhelimen muistiin. Lisäksi applikaatioon on lisätty Listpicker-komponentti, jolla yhdistettävän Bluetooth-laitteen valinta tapahtuu, sekä ActivityStarter-komponentti, jonka avulla voidaan suorittaa siihen liitettyjä toimintoja kun komponentti on saanut käynnistyskäskyn. Komponenttien lisäysten jälkeen siirryttiin block-näkymään, jossa itse logiikkaohjelmointi tapahtuu (kuva 21).



Kuva 21. Block-näkymässä määritetty Bluetooth-yhteyden muodostaminen sekä yhteyden ja sen nimen tallennus TinyDB-tietokantaan. MAC-osoitteet ovat vakiopituisia, joten ne on helppo erotella segment text -logiikkalohkolla



Kuva 22. Vanhan Bluetooth-yhdeyden hakeminen sekä siihen yhdistäminen ohjelman käynnistyessä. Lisäksi ActivityStarter-komponentille annetaan käynnistyskäsky, sekä Clock-ajastinkomponentti asetetaan päälle, mikäli Bluetooth-yhteys on saatu päälle. Näiden lisäksi määritellään graafisen näkymän komponenttien ominaisuuksia

7.2 Tiedon vastaanotto Arduinosta

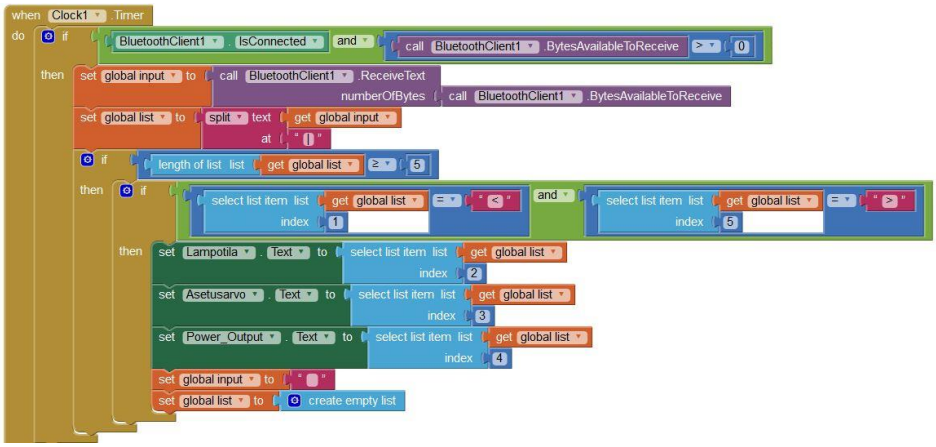
Tietoja haetaan Arduinosta aina kun Clock-ajastinkomponentin laskema aika tulee täyteen. Ajastimen viiveeksi on määritetty puolet vähemmän kuin Arduinon ohjelmassa määritettyyn viiveeseen. Seuraavien ehtojen tulee täyttyä jotta Arduinosta lähetettyjä tietoja noudetaan applikaatioon:

- Bluetooth-yhteyden täytyy olla päällä.
- Arduinosta lähetetyn datan määrän tulee olla enemmän kuin nolla tavua.

Kun nämä ehdot ovat täyttyneet, niin Arduinosta tullut tieto tallennetaan global input -nimiseen muuttujaan. Global list -muuttuja hakee tekstin global input -muuttujalta ja jakaa tekstin |-merkkien perusteella listaksi. Ehtoja joiden täytyy täyttyä, jotta vastaanotettua dataa käytetään:

- Vastaanotetusta datasta tehdyssä listassa täytyy olla viisi kohtaa.
- Vastaanotetun datan ensimmäisen merkin on oltava ”<”-merkki.
- Vastaanotetun datan viimeisen merkin on oltava ”>”-merkki.

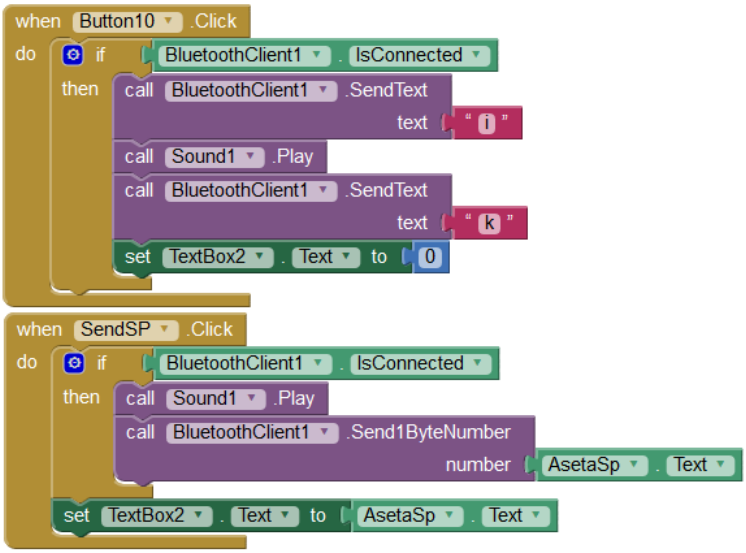
Mikäli ehdot täyttyvät, saadut arvot listan kohdista kaksi, kolme sekä neljä asetetaan niille osoitettuihin tekstikenttiin (kuva 23). Listan ensimmäiseen kohtaan on sijoitettu ”<”-merkki ja viimeiseen kohtaan ”>”-merkki, eikä niitä enää käytetä. Tämän jälkeen global input- ja global list -muuttujat nollataan. Muuttujat nollataan myös sovelluksen aloituksessa.



Kuva 23. Tiedon saanti Arduinosta

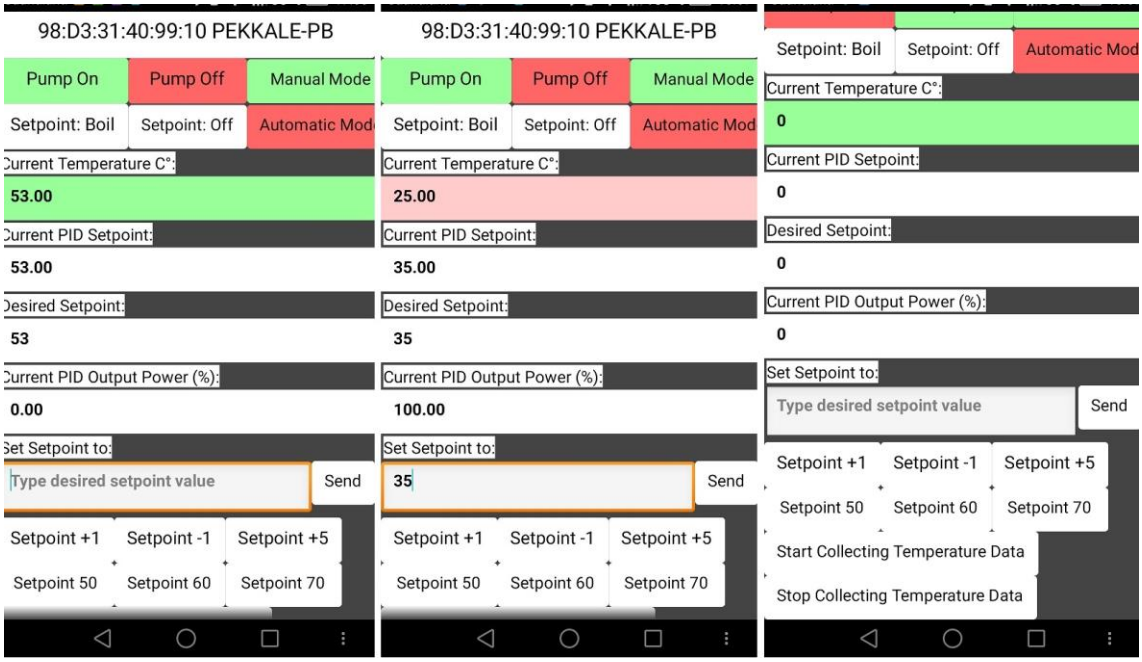
7.3 Tiedon lähetys Arduinoon

Tiedon lähetys Arduinoon tapahtuu applikaatiossa olevien painonappien sekä asetusarvolle varatun kirjoituskentän avulla. Ehtona datan lähetykselle on se, että Bluetooth-yhteyden tulee olla päällä. Nappeja painettaessa niistä kuuluu ääni, joka indikoi napin painalluksen tapahtuneen. Myös tekstikenttään kirjoitetun asetusarvon hyväksynnästä kuuluu sama ääni. Lähetettävä tieto painonapeista tapahtuville komennoille on kirjaimina, jotka näkyvät Arduinoon saapuessaan ASCII-muunnoksesta johtuen desimaalilukuina. Applikaatiosta painonapeilla lähetettäviin komentoihin käytetään vain sadan ylittäviä desimaalilukuja, koska alle sadan lukuarvot säästetään kirjoituskentästä lähetettävälle asetusarvon valintatiedolle (kuva 24). Kun painonapin kirjain on lähetetty Arduinoon, niin sen perään lähetetään myös k-kirjain, joka on desimaalilukuna 107. Tämä tehdään siksi, jotta Arduino suorittaisi valitun komennon vain kerran, sillä luku 107 ei ole sidottu mihinkään Arduinon ohjelman käskyyn. Mikäli tätä ei käytettäisi, niin Arduino suorittaisi viimeisen saamansa komennon jokaisella ohjelmakierrollaan. Toinen vaihtoehto olisi myös pyyhkiä Arduinon Serial0-sarjaväylä puhtaaksi, mutta silloin myös jonossa olevat komennot pyyhkiytyisivät pois, eikä laitteisto toimisi käyttäjän odottamalla tavalla.



Kuva 24. Tiedon lähetys Arduinoon. Ylemmässä lohkossa Arduinoon lähetetään komento painonapin painalluksen jälkeen. Alemmassa lohkossa Arduinoon lähetetään kirjoituskentän asetusarvo lähetyspainikkeen painalluksen jälkeen

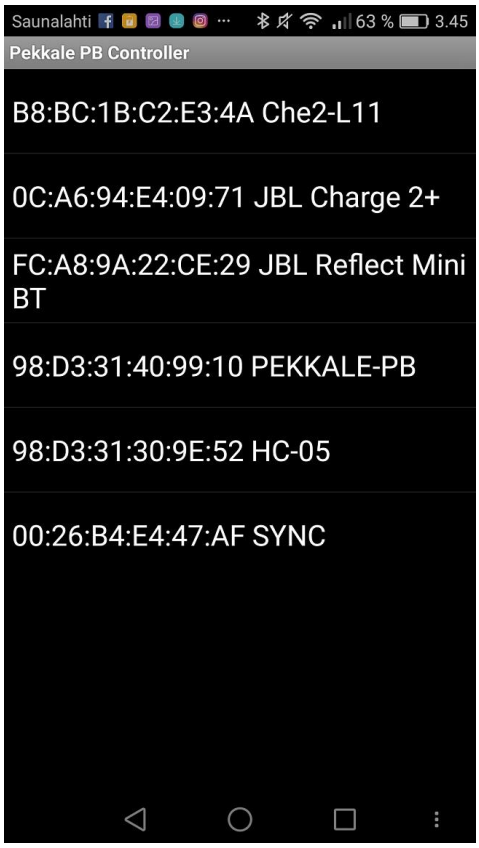
7.4 Graafinen käyttöliittymä



Kuva 25. Applikaation graafisen käyttöliittymän versio 1.0

Kuvassa 25 näkyvä laitteiston graafinen käyttöliittymä sisältää painonappeja sekä yhden kirjoituskentän laitteiston ohjaukseen. Käyttöliittymä näyttää nesteen lämpötilan, PID-säätimen asetusarvon sekä PID-säätimen ulostuloarvon alle sekunnin viiveellä. Arvot on Arduinon ohjelmassa määritetty neljän tavun Double precision floating point number -formaatiksi, jolloin arvot voidaan applikaatiossa esittää sadasosan tarkkuudella. Lämpötilalle varattu tekstikenttä muuttaa värinsä vihreäksi kun asetusarvon mukainen lämpötila on saavutettu. Lisäksi pumpun painonappien värit indikoivat pumpun kyseistä tilaa. Asetusarvon valinnalle tehtyyn kirjoituskenttään pystyy syöttämään vain numeroita, ja ainoastaan lukuarvot nollasta sataan vaikuttavat säätimen asetusarvoon. Käyttöliittymän alareunassa on myös painikkeet datan keräyksen aloittamiselle sekä lopettamiselle. Datan keräys tapahtuu ottamalla tekstikentistä tiedot applikaation ajastimen mukaan, ja tallentamalla ne tekstitiedostona ohjelman kansioon kun datan keräyksen lopetuspainiketta on painettu. Käyttöliittymän ylimmässä kentässä näkyy yhdistetyn Bluetooth-laitteen MAC-osoite sekä nimi. Mikäli käyttöliittymää ei ole vielä yhdistetty Bluetooth-laitteeseen, niin kentässä lukee ”Select Bluetooth device”. Kenttää painamalla voi valita Bluetooth-laitteen, johon käyttöliittymä yhdistetään (kuva 26). Bluetooth-laite pitää tätä ennen määrittää Android-laitteesta

laitepariksi, jotta se näkyisi listassa. Tässä vaiheessa laite pyytää myös siihen AT-komennoilla asetettua PIN-koodia.



Kuva 26. Bluetooth-laitteen valintaikkuna. Ikkunassa näkyy kaikki käytössä olevan Android-laitteen muodostamat Bluetooth-laiteparit

8 Laitteiston testaus

Kuva 27. Laitteiston kolmivaihetestausta Lapinlahdessa

Laitteiston testaukset alkoivat ennen ensikäynnistystä tapahtuneilla sähkökytkentöjen tarkastuksilla. Oikosulkujen mahdollisuus tehdyissä kytkennöissä poissuljettiin mittaamalla kaikki johtimet erikseen. Maadoituksien resistanssit mitattiin kaikkien maadoituspisteiden ja laitteiston rungon väliltä mahdollisimman monesta kohdasta. Näiden jälkeen alkoivat laitteiston varsinaiset käyttötestaukset yhdellä vaiheella. Laitteiston testaus oli jatkuvaa ja se muodostuikin edellytykseksi laitteiston valmistumiselle, sillä ilman jatkuvaa testausta laitteiston saattaminen tähän pisteeseen olisi kestänyt huomattavasti kauemmin. Käyttötestien alkuvaiheessa ohjelmallisten puutteiden ja vikojen korjaaminen koski enemmän Arduinon ohjelmaa, kun taas loppuvaiheessa ne koskivat enemmän Android-ohjelmaa.

Laitteston kolmivaihetestaukset aloitettiin vasta kun Android-sovellus oli käyttökunnossa, jotta laitteen käyttäytymistä pystyi havainnoimaan paremmin nopeilla ohjausmuutoksilla (kuva 27). Tässä vaiheessa tuli esille myös monia käyttöliittymän parannusideoita, kuten applikaation näyttämä PID-säätimen ulostulon arvo. Ulostulon

tehon monitoroinnilla kykenee ymmärtämään paremmin PID-säätimen toimintaa ja sen tarkkailusta lämpötilan ohella oli suuri hyöty viritysparametrien hienosäädössä. Kaikkein voimakkaimmin laitteiston kehitystä lopulta ohjasi nimenomaan laitteiston testauksessa ilmenneet puutteet ja parannusideat.

9 Pohdintaa

Insinöörityön suurimmat haasteet olivat aikataulu ja työn laajuus. Työssä tuli jatkuvasti eteen odottamattomia ongelmia sekä työvaiheita joihin oli vaikea varautua. Näihin kului arviolta lähes yhtä paljon aikaa kuin muuhun työhön. Työn tekemisen aikatauluttaminen oli tästä syystä erityisen haastavaa. Ongelmat aikatauluttamisen kanssa tulivat ilmi etenkin osia tilattaessa, joita jouduin tilaamaan ulkomailta useampaan otteeseen. Osien saapuminen kesti usein pitkään, jolloin useita työvaiheita ei voinut suorittaa ilman puuttuvia osia.

Työn aiheen sisältämä laajuus konkretisoitui kirjoitusvaiheessa, jonka aikana hahmotin selkeän kuvan työn monista keskenään hyvin erilaisista vaiheista. Monista pienemmistä työvaiheista jäikin aiheen laajuuden vuoksi kirjoittamatta. Suurin osa työn kirjoitusosuudesta muotoutui lopulta käsittelemään laitteiston ohjelmointia, tietoliikennettä sekä säätöä.

Säätimen viritys askelvaste-menetelmällä tuotti Ziegler-Nichols menetelmälle tyypillisen agressiivisen säädön, jossa lämpötila hetkellisesti ylitti asetusarvon ja vaikka ylitys olikin enintään 0,7 °C, niin säätimen toiminta ei ollut tyydyttävä. Sain viritysarvoja hienosäätämällä säätimen toimimaan stabiilimmin, mutta lopulta parhaimman tuloksen sain kuitenkin tekemällä virityksen uudestaan eri menetelmällä.

Kaikkia yhdistyksen kanssa sopimiani laitteiston ominaisuuksia en ehtinyt saamaan tähän prototyyppiin valmiiksi. Laitteistoon suunniteltua reseptin hakua ja automaattiajoa en saanut applikaation tämänhetkiseen versioon, mutta kaikki automaattiajolta vaadittavat ominaisuudet löytyvät jo Arduinon ohjelmasta. Laitteen integroitu jäähdytin puuttuu vielä, mutta liitännät ja venttiilit sille ovat jo valmiiksi asennettu. Vierteen jäähdytys tapahtuu toistaiseksi immersiojäähdyttimellä. Näiden lisäksi järjestelmän tiedonkeruuominaisuus kaipaa vielä kehitystä. Laitteisto on kuitenkin jo tällä hetkellä täysin kykenevä hallitsemaan oluen panemiseen vaadittavia prosesseja, joten se siirtyy

yhdistykselle testikäyttöön. Laitteiston kehitysmahdollisuudet vastaaviin kaupallisiin laitteisiin verrattuina ovat varsin suuret, joten laitteiston kehitys yhteistyössä yhdistyksen kanssa tulee jatkumaan.

Insinöörityö oli hyvin monipuolinen ja opettavainen kokonaisuus, jonka aikana opin sulautetun prosessinohjausjärjestelmän rakentamisesta, säädöstä sekä ohjelmoinnista laaja-alaisesti. Opin myös ymmärtämään suunnittelutyön tärkeyden ja sen vaikutukset projektin aikatauluun. Vaikka en pidä laitteistoa vielä täysin valmiina, niin olen silti erittäin tyytyväinen sen tämänhetkiseen tilaan ja pidän sitä varsin onnistuneena laitteistona. Työn lopputuloksena syntyi myös into kehittää vastaavanlaisia järjestelmiä tulevaisuudessakin.

Lähteet

1 Uutinen pienpanimoista. Verkkosivu. Luettu 6.5.2017. <http://www.ess.fi/uutiset/kotimaa/art2290203/>

2 Lapinlahti Beer Fellows ry. Yhdistyksen perustamisasiakirja.

3 Palmer, John. 2006. How to Brew. Brewers Publications.

4 Tietoa Incoloy 800-metalliseoksesta. Verkkodokumentti. Luettu 5.5.2017. <http://www.specialmetals.com/assets/documents/alloys/incoloy/incoloy-alloy-800.pdf/>

5 Lämmitysvastuksen tiedot. Verkkosivu. Luettu 10.4.2017. <https://store.brewpi.com/mashing/stainless-steel-heating-elements/incoloy-800-3-phase-heating-element-5500w/>

6 DS18B20 anturin datasheet. Verkkodokumentti. Luettu 12.4.2017. <https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf/>

7 Tietoa puolijohdereleistä. Verkkosivu. Luettu 24.4.2017. <https://en.wikipedia.org/wiki/Solid-state_relay/>

8 Vierrepumpun ohjeet. Verkkodokumentti. Luettu 18.4.2017. <https://www.brouwland.com/content/docs/018/018%20Wijn%20-%20en%20bierpompen/018.600.x_MAN_Pump'in.pdf/>

9 Vierrepumpun tietoja. Verkkodokumentti. Luettu 19.4.2017. <https://www.brouwland.com/en/pdf/018.600.15.pdf/>

10 Bluetooth moduuli. Verkkosivu. Luettu 20.4.2017. <http://wiki.keyestudio.com/index.php/Ks0097_keyestudio_Bluetooth_Transmission_Module_for_Arduino_with_Bottom_HC-05_Master_and_Slave#Specification/>

11 Bluetooth moduulin AT-komennot. Verkkodokumentti. Luettu 20.4.2017. <http://eskimon.fr/wp-content/uploads/2014/10/commandes_AT_HC05.pdf/>

12 Yleistietoa Arduinosta. Verkkosivu. Luettu 19.4.2017. <https://fi.wikipedia.org/wiki/Arduino/>

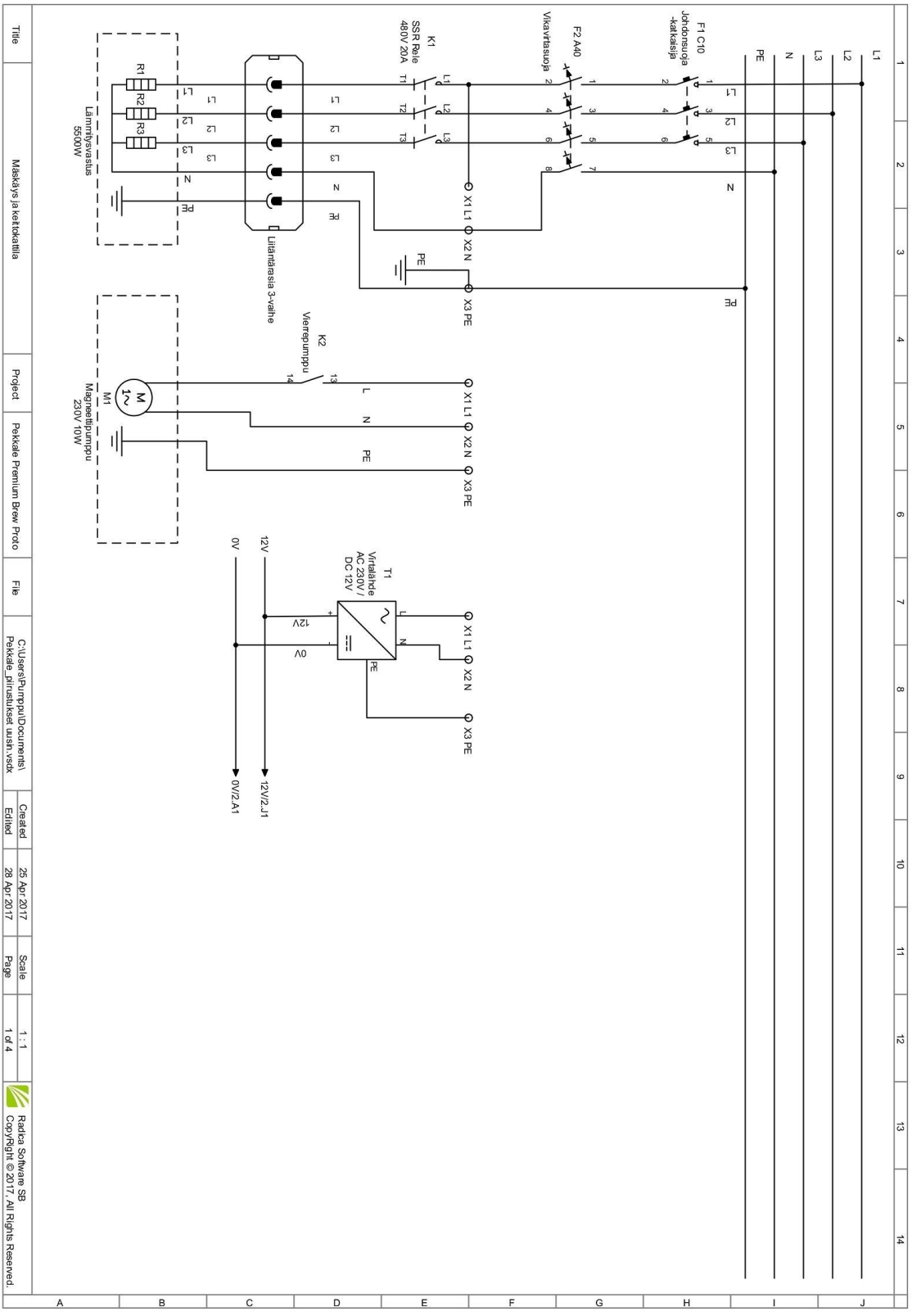
13 Visioli, Antoni. 2006. Practical PID control. Springer. PhD in Advances in Industrial Control. E-kirja.

14 Yleistietoa PID-säätimestä. Verkkosivu. Luettu 27.4.2017. <https://en.wikipedia.org/wiki/PID_controller/>

15 Tietoja PID-säätimistä. Verkkodokumentti. Luettu 4.5.2017. <https://www.cds.caltech.edu/~murray/courses/cds101/fa02/caltech/astrom-ch6.pdf/>

16 Yleistietoa MIT App Inventorista. Verkkosivu. Luettu 18.4.2017. <https://en.wikipedia.org/wiki/App_Inventor_for_Android/>

Sähkökuvat osa 1



Sähkökuvat osa 2