Tampereen ammattikorkeakoulu

Espressokeittimen ohjausjärjestelmä

Mikko Paaso

OPINNÄYTETYÖ
Marraskuu 2023

Tieto- ja viestintätekniikan tutkinto-ohjelma
Ohjelmistotekniikka

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tieto- ja viestintätekniikan tutkinto-ohjelma
Ohjelmistotekniikka

PAASO MIKKO:
Espressokeittimen ohjausjärjestelmä

Opinnäytetyö 29 sivua, joista liitteitä 1 sivu
Marraskuu 2023

Opinnäytetyössä luotiin Rancilio Silvia -espressokeittimen ohjaus käyttäen Raspberry Pi Pico -mikrokontrolleria. Espresson uuttoon vaikuttaa pääasiassa useampi eri tekijä, kuten uutonaikainen lämpötila, uuttoaika sekä uuttopaine. Työssä laitteen eri asetusten säätäminen tapahtuu millä tahansa laitteella, jossa on webselain sekä yhteys paikalliseen WiFi-verkkoon. Kyseinen espressokeitin on manuaalikäyttöinen ja siksi hyvin altis käyttäjän virheille. Vaikka laite on yksinkertainen, uuttoprosessin optimointi on haastavaa. Työn tarkoitus oli automatisoida keitintä siten, että uuttoprosessista poistetaan inhimillisiä virheitä, kuten väärä uuttolämpötila, epätasainen uuttolämpötila, epätasainen vedenpaineen jakautuminen sekä laitteen likaantuminen.

Keittimen yhdistäminen mikrokontrolleriin tuotti ongelmia, koska laitteen osien hankinta ei onnistunut. Työssä käytettiin virtuaalista kattilaa ja testiohjaus toteutettiin lähiverkossa olevalla käyttöliittymällä. Ohjauksella tehtiin seuraavia muutoksia laitteen toimintaan: uuttolämpötilan ohjaus digitaalisena muutti lämpötilahaarukkaa alkuperäisestä laitteen noin 10 celsiusasteesta noin 0,5 celsiusasteeseen. Uuttopaineen jakautumista parannettiin kahvin esikastelulla, joka kastelee kahvin matalalla paineella jaksottaen pumppua ennen kuin paine nostetaan täyteen uuttopaineeseen. Uuton lopuksi järjestelmä viivyttää uuttovesisolenoidin vapautumista uuttotilasta, jotta äkkinäisen paineen laskun takia uutettu kahvi ei räjähdä ympäri uuttotilaa.

Projektin kehittämiseksi merkittävin jatkotoimenpide olisi mikrokontrollerin fyysinen kytkentä laitteeseen. Tämä kuului alkuperäiseen opinnäytetyösuunnitelmaan, mutta peruuntui rahoituksen puutteesta. Ohjelmisto on suunniteltu siten, että muutaman kommentin siirtäminen antaa sille valmiuden toimia yhdessä raudan kanssa. Ohjelmiston jatkokehityksessä olisi hyvä painottaa erilaisten uuttoprofiilien tallentamismahdollisuutta rerilaisia kahvilaatuja varten.

Asiasanat: espresso, silvia, raspberry, pico, mikrokontrolleri

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Programme in Information and Communication
Software Engineering

MIKKO PAASO
Espresso Machine Control System

Bachelor's thesis 29 pages, appendices 1 page
November 2023

This thesis aimed to create a control system for the Ranchilio Silvia espresso machine using Raspberry Pi Pico microcontroller. It enables adjustments via any device with a web browser and local WIFI. The Silvia machine, though simple, is manual and prone to user errors. The purpose of the work was to automate the coffee maker in a way that eliminates human errors in the brewing process in the following functions: extraction temperature, water pressure, and device cleanliness.

There was a problem in component financing, so that thesis was limited to the software. To address this, a virtual boiler was used, and test controls were implemented through a local network interface. The control system brought significant improvements: The digital temperature control narrowed the temperature ranged from approximately 10°C to about 0.5°C, ensuring a precise brewing temperature.

The improved water pressure distribution was enhanced through pre-infusion at a lower pressure before reaching the full brewing pressure. To obtain delayed pressure release to prevent splattering, a delay was introduced in releasing brewing pressure from the solenoid valve after brewing.

Future development should focus on physically integrating the microcontroller into the machine. The software’s design allows seamless integration with hardware after minor adjustments. Further software development should emphasize the ability to save diverse brewing profiles for various coffee types, enhancing brewing versatility and customization.

Key words: espresso, silvia, raspberry, pico, microcontroller

SISÄLLYS

LYHENTEET JA TERMIT

API	ohjelmointirajapinta (application programming interface)
Barista	Kahvin uuttaja. Ammattilaiset sekä harrastajat.
Git	Version hallinta järjestelmä
IDE	Integroitu kehitysympäristö
Jauhatus	Jauhettu kahvi. Viitataan yleensä jauhatuksen karkeudessa
Kakku	Espresson keitossa uuttoa varten pakattu kahvi
Kattila	Espressokeittimessä veden kuumennus yksikkö
Kori	Espressokeittimessä suodatin mihin kahvi pakataan
Open source	Vapaa lähdekoodi
Pico	Raspberry Pi Pico
Pre-infuusio	Kahvin esiuutto matalammalla paineella.
Rauta	Laitteisto
Säie	Prosessorin tiedonhallintayksikkö
TAMK	Tampereen ammattikorkeakoulu
Thonny	IDE
Uuttotila	Espresson keitossa tila missä uutto suoritetaan
VSCode	IDE

1 JOHDANTO

Tämän työn valinta johtui siitä, että opinnäytetyön tekijä on baristaharrastaja. Hänellä on myös harrastuksellinen kiinnostus sekä tarve laitemodifikaatioon ja niiden kautta optimaalisemman uuttoprosessin kautta laadukkaampaan espressokahviin.

Ranchilio Silvia sijoittuu espressokeittimessä hintaluokkaan, minkä sisällä on suhteellisen yleistä harrastajien tekemät muutostyöt keittimeen. Se on hintatasoonsa nähden hyvä espressokeitin kotibarista harrastukseen. Keittimessä on kuitenkin omat ongelmat, jotka vaikeuttavat edistyksellisemmän uuton saamista halutulle tasolle.

Keittimen toimintaa tulisi parantaa seuraavilla alueilla:

1. Lämpötilan hallinta. Espresson uutossa liian matala lämpötila johtaa aliuuttoon ja liian korkea taas yliuuttoon. Keittimen nykyinen lämpötilan hallinta on toteutettu perinteisellä termostaatilla, jonka vaihteluväli on noin 10°C. Näin suuren vaihtelun vaikutus uuttoon on todella iso. Uuttolämpötila uuton alussa tulisi saada yhden asteen tarkkuuteen. Myös uuton aikana tippuva lämpötila on yksi ongelma. Uuttoa aloitettaessa lämpötila alkaa tippua. Keitin rupeaa reagoimaan tähän vasta, kun lämpötila laskee alle termostaatin alarajan. Tätä ongelmaa olisi hyvä korjata siten, että kattilan lämmittäminen aloitetaan heti uuton alkaessa. Myös kattilan esilämmitys olisi hyvä keino vähentää lämpötilan tippumista keittoprosessin aikana. Espressokeittimen kattila on usein valmistettu raskaasta messingistä. Tämä aiheuttaa sen, että lämpötilan muutokset jatkuvat vielä pitkään lämmityselementin toiminnan muutoksen jälkeen. Tämä tulisi huomioida termostaatin toiminnassa.

   Espressokone vaatii huomattavan pitkän lämpenemisajan. Noin puoli tuntia ennen uuttoa keitin olisi hyvä käynnistää, että lämpötila tasoittuisi. Tätä aikaa voitaisiin vähentää nostamalla lämpö 130°C asteeseen laitteen käynnistyksen yhteydessä. Tämän jälkeen, kun laitteen annetaan jäähtyä uuttolämpötilaan, tulisi lämpötilan olla suhteellisen tasainen.

2. Paineen hallinta uutossa. Kun vettä aloitetaan ajaa espresson uutossa kuivan kahvikakun läpi, muodostuu kakun läpi kanavia, joista vesi pääsee virtaamaan pienemmällä vastuksella. Tällöin kanavan läheisyydessä oleva kahvi yli-uuttuu ja muualla oleva kahvi ali-uuttuu. Tähän ongelmaan on olemassa ratkaisuna menetelmä pre-infuusio. Tässä menetelmässä kahvi kastellaan pienemmällä paineella ennen varsinaista uuttoa. Matalampi paine jakautuu helpommin ympäri kakkua. Kakun läpikotaisen kastelun jälkeen uuttopaine nostetaan täyteen paineeseen. Kasteltu kakku jakaa paineen huomattavasti paremmin ja siten ehkäisee kanavoitumisen muodostumista.

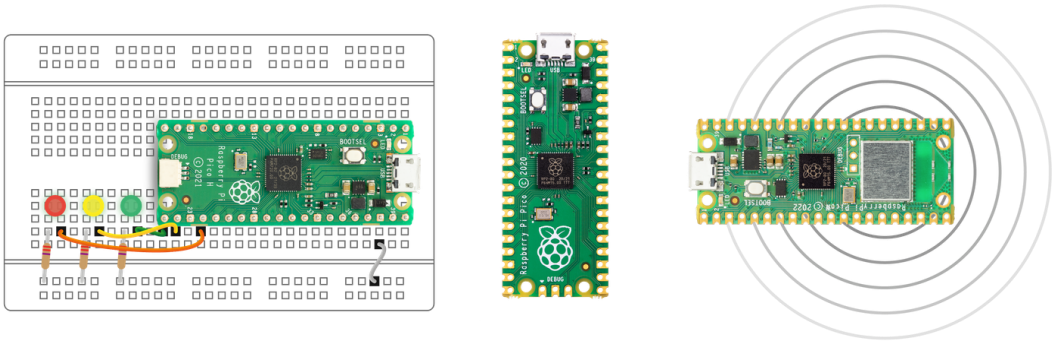
3. Paineen pehmeä lasku uuton jälkeen. Espresson uuttovaiheen jälkeen espressokone avaa vesisolenoidin välittömästi pumpun pysähtyessä. Tästä saattaa muodostua tilanne, jossa kakun sisällä oleva paine pääsee purkaantumaan liian nopeasti ja kakku räjähtää uuttotilan sisälle. Tämä tekee uuttotilan todella likaiseksi ja vaatii ylimääräistä puhdistamista. Tämä voidaan ratkaista siten, että pumpun pysähdyttyä annetaan solenoidille hetken aikaa pitää paine uuttotilan sisällä. Tällöin paine pääsee purkautumaan rauhallisesti kakun läpi, eikä sotkua pääse muodostumaan.

Työn alkuperäisenä suunnitelmana oli liittää mikrokontrolleri keittimeen, mutta se jätettiin pois osien rahoituksen puutteen vuoksi. Haasteeksi muodostui ohjelmiston kehitykseen ja testaukseen liittyvä puutteellinen vaste. Ratkaisuna keittimen toimintaa simuloidaan virtualisoimalla keittimen toimintaa.

2 TYÖKALUT

2.1 Alusta: Raspberry Pi Pico W

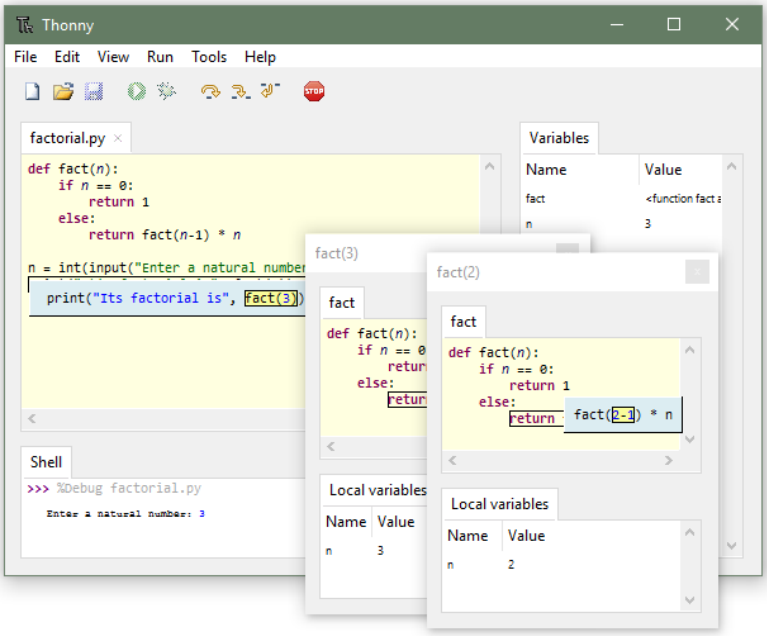
Alkuperäinen suunnitelma oli tehdä työ Arduino Nano:lla. Ajatus kuitenkin muuttui jonkin verran osittain sen takia, että ohjelmiston halutunlainen toiminta vaati kahden säikeen ajoa. Valinta kohdistui Raspberry Pi Pico W malliin (kuvio 1). W-malli lisäys sisältää myös WiFi-radion, joka helpottaa käyttöliittymän integrointia. Raspberry Pi Pico W on edullinen, kooltaan pieni sekä suuren käyttäjäyhteisön omaava mikrokontrolleri.



KUVIO 1. Raspberry Pi Pico (Raspberry Pi. n.d.)

2.2 IDE: Thonny

Thonny (kuva 1) on yksinkertainen kehitysympäristö eli IDE (Integrated Development Environment, integroitu kehitysympäristö), johon on integroitu MicroPython kehitykseen tarvittavia työkaluja. Thonny on hyvin laajalti käytetty työkalu MicroPython-kielen ohjelmointiin. Thonny yksinkertaisuudessaan tarjoaa käyttäjälle nopean käyttöönoton.



KUVA 1. Thonny IDE (Thonny. n.d.)

2.3 Kielet

2.3.1 MicroPython

Mikropython on ohjelmointikieli, joka on pythonin versiosta 3 kevennetty versio rajoitettuihin ympäristöihin. Se soveltuu hyvin käytettäväksi mikrokontrollereissa. MicroPython on avoimen lähdekoodin ohjelmointikieli (George Robotics Limited. 2014–2003). Työ on tehty suurimmaksi osaksi MicroPythonilla. Valinta pohjautui työn tekijän kokemukseen sekä mieltymykseen python kieleen.

2.3.2 HTML

HTML (Hypertext Markup Language) avoimesti standardoitu merkintäkieli, jolla voidaan kuvata hyperlinkkejä sisältävää tekstiä eli hypertekstiä. HTML:llä voidaan myös merkitä tekstin rakenne eli esimerkiksi, mikä osa tekstistä on otsikkoa ja mikä leipätekstiä. HTML tunnetaan erityisesti kielenä, jolla internetsivut on kirjoitettu (Wikipedia. 23.10.2023). Työssä käyttöliittymä on tehty HTML-pohjalle.

2.4 Kirjastot

Työssä hyödynnetään valmiita kirjastoja mahdollisuuksien mukaan. Työssä käytetyt tärkeimmät kirjastot ovat:

1. _thread kirjasto mahdollistaa monisäieajon (Damien P. George. 6.11.2023). Työssä tätä kirjastoa on käytetty käyttöliittymän sekä raudan ohjauksen säikeiden yhtäaikaiseen ajoon.
2. utime on toinen ajan käsittelyyn liittyvä kirjasto (Damien P. George. 18.4.2021). Työssä tätä kirjastoa käytettiin aikaerojen laskennassa koskien kattilan kiihtymisnopeutta sekä viiveiden luomiseen koodin käsittelyyn.
3. JSON (lyhenne sanoista JavaScript Object Notation) on yksinkertainen ja kevyt kirjasto avoimen standardin tiedostomuototiedonvälitykseen ja tallennukseen (json.org. n.d.). Työssä json:ia käytettiin tietojen tallentamiseksi tiedostoon, jotta kontrollerin uudelleen käynnistyessä vanhat tiedot olivat tallella.
4. Machine on kirjasto, joka pitää sisällään rautapuolen hallintaan liittyviä funktioita. Machinen sisältä löytyy Pin-kirjasto, jonka sisältä löytyy funktioita mikrokontrollerin sisään ja ulostulojen hallintaan (Damien P. George. 6.11.2023).
5. Socket-moduuli tarjoaa rajapinnan kommunikaatiolle verkon välityksellä päätepisteiden välille (Damien P. George. 6.11.2023).

6. Network-moduuli tarjoaa laitteelle WiFi yhteyden lähiverkkoon (Damien P. George. 6.11.2023).

3 JÄRJESTELMÄN SUUNNITTELU

3.1 Vaatimukset järjestelmälle

3.1.1 Käyttäjä

Tärkeimmät vaatimukset käyttäjän näkökulmasta ovat: laitetta on myös pystyttävä käyttää kuten se on alkuperäisesti suunniteltu käytettäväksi. Käyttöliittymään pitää päästä käsiksi lähiverkon kautta ja sen tulee olla helposti ymmärrettävä ja looginen. Käyttöliittymässä pitää pystyä säätämään: uuttolämpötilaa, höyrystyslämpötilaa, esilämmitysaikaa, pre-infuusio aikaa, paineen pehmeän laskun aikaa. Käyttöliittymään tulee myös kehitystä varten painonapit virtuaalisen laitteen ajoa varten.

3.1.2 Toiminnalliset vaatimukset

Tärkeimmät toiminnalliset vaatimukset ovat: toimintojen pitää olla toiminnaltaan parempia, kuin alkuperäiset toiminnot. Laitteen tulee käynnistäessä lämmittää itsensä käyttölämpötilaan nopeammin kuin normaalisti. Termostaatin pitää pystyä ennakoimaan laitteen lämpenemistä ja viilenemistä viiveineen. Järjestelmän pitää optimoida laitteen lämmitys siten, että kun esimerkiksi uutto on käynnissä lämpöä ei säätele termostaatti, vaan lämmitystä tehdään koko uuton ajan, jotta lämpötila tippuisi mahdollisimman vähän.

3.1.3 Tekniset vaatimukset

Tärkeimmät tekniset vaatimukset ovat: ohjelmistossa ei saa ilmetä virheitä käytössä. Laitteen käyttöliittymä ja rautahallinta tulee toimia eri säikeissä toisistaan riippumatta. Laitteen tulee pystyä palauttamaan edellisessä käytössä asetetut arvot uudellen käynnistämisen yhteydessä. Johtuen siitä, että mikrokontrolleria ei kytketä espressokoneeseen, tarvitsee ohjelmistossa olla virtuaalinen kattila toimintojen seuraamiseen.

3.2 Ominaisuudet

3.2.1 Säikeiden rinnakkainen toiminta

Ohjelmiston toiminta pohjautuu kahden säikeen rinnakkaiseen ajoon. Toinen säikeistä ohjaa käyttöliittymää ja toinen rautapuolen hallintaa. Säikeiden välinen kommunikaatio tapahtuu globaalin brew_data-luokan olion välityksellä. Konsolitulostuksiin käytetään myös globaalia lock_printer-luokan oliota. Molemmat luokat sisältävät lukitusmekanismin, joka estää muuttujiin kohdistuvan yhdenaikaisen käytön.

3.2.2 Käyttöliittymä

Käyttöliittymänä toimii WiFi-yhteydellä lähiverkkoon toteutettu API, joka tarjoaa HTML-sivua käyttäjälle. Käyttöliittymässä on seuraavat arvokentät: uuttolämpötila, höyrystyslämpötila, pre-infuusioaika, esilämmitysaika sekä paineen pehmeän laskun aika. Näiden asetusten vahvistukselle on painonappi. Testikäyttöä varten myös painonapit: brew, water ja steam.

3.2.3 Virtuaalinen kattila

Kattilan toteutus on virtuaalinen johtuen siitä, että työtä ei liitetä rautaan. Kattila jäljittelee oikeaa kattilaa siten, että sitä lämmittäessä aluksi lämpeneminen on hidasta ja kiihtyy lämmityksen jatkuessa maksimilämpenemisvauhtiinsa. kattilan lämmityksen lopuksi lämpeneminen jatkuu hidastuvasti. Myös kattilaa viilentäessä kattilan jäähtymisnopeus kiihtyy, mutta jää huomattavasti alle lämpenemisnopeuden. kattilan lämmitys ja jäähdytys toimivat funktioilla: cooldown ja heat. Funktioilla on oletusarvonsa, mutta niihin voidaan syöttää vaihtoehtoisia arvoja ottaen huomioon erilaiset poikkeukset kattilan käyttäytymiselle.

3.2.4 Termostaatti

Termostaatti pitää huolen kattilan lämpötilan säätämisestä oikealle tasolle. Termostaatti ohjaa pico:lla lämmitysreleelle lähtevää ulostulopinniä. Mikäli höyrystyskytkin on päällä, ottaa termostaatti tavoitelämpötilaksi höyrytyslämpötilan. Mikäli taas ei, valitaan tavoitelämpötilaksi uuttolämpötila. Lämpötilan muutosnopeus mitataan omalla oliolla. Termostaatti ottaa huomioon muutosnopeuden lähestyttäessä tavoitelämpötilaa ja ennakoiden joko pysäyttää tai aloittaa lämmityksen ennakkoon siten että lämpötila muutosnopeus asettuu nollaan mahdollisimman lähelle tavoitelämpötilaa.

3.2.5 Esilämmitys, preinfuusio, uutto ja pehmeä paineen lasku

Esilämmitys vähentää uuton aikana tapahtuvaa lämpötilan tiputusta. Esilämmitys ohjaa annetun arvon ajan lämmityselementin relettä ennen uuttoa siten, että lämmityselementti saadaan kuumenemaan jo ennen uuttoa. Täten uuton aikana valmiiksi kuuma elementti lämmittää vaihtuvaa vettä tehokkaammin ja jäähtyminen on pienempää. Johtuen veden lämpenemisestä, tämä tulee kuitenkin käyttäjän huomioida uuttolämpötilaa säätäessä. Uuttolämpötilaksi valitaan pienempi arvo, kuin mikä on haluttu tavoitelämpötila.

Pre-infuusio aloittaa kahvin kastelun matalammalla paineella määritetyn ajan ennen varsinaisen uuton alkamista. Matalampi paine saavutetaan siten, että kytketään jännite uuttotilan suuntaan vedenvirtauksen avaavan solenoidin releen pinniin. Tämän jälkeen pumpun releen pinniä jaksotetaan siten, että pumppu on puolisekuntia päällä ja puoli sekuntia pois päältä. Täten paine tippuu osittain tauon aikana kahvin virratessa, joten paine ei pääse puolen sekunnin pumppaussyklin aikana nousemaan liian korkealle.

Uuttotilassa tapahtuu itse kahvin uuttaminen puskemalla kuumaa vettä kakun läpi. Uuttotila käynnistyy, kun uuttokytkin on kytkettynä päälle. Varsinaista uuttoa Uuttotilassa kytketään jännite pumpun, solenoidin ja lämpöelementin releille lähteviin pinneihin. Kun uuttokytkin kytketään pois päältä, kytketään pumpun ja läm-

mitysreleen pinneiltä jännite pois. Mikäli pehmeä paineen lasku on ajastettu, suoritetaan se seuraavaksi. Muussa tapauksessa kytketään jännite pois myös solenoidin releeltä.

Pehmeä paineen lasku viivyttää solenoidin releelle johtavan pinnin jännitteen tiputusta määritetyn ajan. Kun pumppu on pysähtynyt, paine pääsee hitaasti purkautumaan kakun läpi ja kun viive on ohi, vapautetaan jäljellä oleva paine kytkemällä jännite pois solenoidin releen pinniltä. Viive on säädettävissä johtuen eri kahvilaatujen ja jauhatusten eriävistä paine vastuksista.

3.2.6 Vesitila ja Höyrytila

Vesitila on tila, jonka avulla saadaan, esimerkiksi valutettua kuumaa vettä kahvin sekaan höyrysauvasta, kun valmistetaan americano-kahvijuoma. Vesitila käynnistyy, kun vesitilan kytkin on kytkettynä päälle. Pumpun ja lämmityselementin releiden pinneihin kytketään jännite. Kun kytkin palautuu pois asentoon, jännite tiputetaan pinneiltä

Höyrytilassa höyrysauvasta tulee kuumaa vesihöyryä kahviin lisättävän vaahdotetun maidon valmistamista varten. Höyrytila käynnistyy, kun höyrytilan kytkin on kytkettynä päälle. Tavoite lämpötilaksi termostaatille asetetaan höyrytyslämpötila. Kun kytkin on kytkettynä pois, tavoitelämpötilaksi palautetaan uuttolämpötila

3.2.7 Konsolitulostus

Kehityksessä käytettävä parametrien tulostus tapahtuu omalla funktiolla, joka suorittaa tiedon tulostuksen selkeästi eri puolilla koodia. Konsoliin tulostetaan laitteen toiminnan kehitystä varten seuraavia tietoja: käyttötila, kattilan lämpötila, releille menevien pinnien asennot, kytkimien asennot.

3.2.8 Asetusten tallentaminen ja lataaminen

Jotta laite toimii uudelleen käynnistyksen jälkeen samoilla parametreilla, laite tallentaa mikrokontrollerille syötetyt uttoparametrit ja lataa ne automaattisesti käynnistyksen yhteydessä.

4 TOTEUTUS

4.1 Suunnittelu

Ensimmäisenä tehtävänä oli laatia vaatimukset järjestelmän toiminnalle. Tähän kuului ohjelmistolle tarvittavat toiminnallisuudet ja tavoitteet. Seuraava tehtävä oli rakenteen ja arkkitehtuurin suunnittelu. Alusta ja ohjelmointikielet valittiin ja tarvittavat osat etsittiin. Työn rajaus oli seuraava vaihe, joka johtui työhön tarvittavien osien rahoitusongelmista. Työtä päätettiin jatkaa vain ohjelmiston kehityksenä.

4.2 Ohjelmointi

Ohjelmointivaiheessa toteutettiin ohjelmiston toiminnallisuudet suunnitelman mukaisesti. Versiohallintaan käytettiin Git:iä. Ohjelmiston rakenne toteutettiin siten, että main.py sisältää käyttöliittymän sekä raudan hallintaan liittyvien säikeiden perusominaisuudet. Funktiot sekä luokat tallennettiin omiin tiedostoihinsa. Luokat toimivat tiedonvälittäjinä säikeiden välissä. Tulostukset ja muistinkäsittely on toteutettu lukoilla siten, että päällekkäinen toiminta on estetty virhetilojen ehkäisyä varten.

4.2.1 Termostaatti

Työssä termostaatti on toteutettu, siten että otetaan huomioon kattilan lämpötilan muutosnopeus. Muutosnopeutta laskettaessa luodaan aluksi HeatingSpeedCalculator-olio, joka sisältää funktion get_heating_speed. Olion toteutus on nähtävissä koodissa 1. Aina kun get_heating_speed -oliota kutsutaan, funktio hakee voimassa olevan ajan sekä saa parametrina ajankohtaisen lämpötilan. Termostaatti laskee lämpötila- ja aikaeron vertaamalla niitä olioon edellisellä kutsulla tallennettuihin tietoihin. Laskettu lämpötilaero jaetaan lasketulla aikaerolla ja tästä saadaan tulokseksi lämpötilan muutosnopeus. Muutosnopeuskertoimella lämpötila saadaan sovitettua kattilan ominaisuuden mukaiseksi, siten että sen

vaikutus termostaattiin on oikea suhtainen. Uudet lämpötila ja aika arvot tallennetaan olioon ja palautetaan laskettu sekä sovitettu muutosnopeus.

```
# Class to calculate heating speed
class HeatingSpeedCalculator:
    def __init__(self, utime_module, heating_speed_multiplier):
        self.temperature_begin = 0
        self.utime = utime_module
        self.time_start = self.utime.ticks_ms()
        self.heating_speed_multiplier = heating_speed_multiplier

    # Function to calculate heating speed
    def get_heating_speed(self, temperature_now):

        # Get time now
        time_now = self.utime.ticks_ms()
        # Calculate time between now and starting point
        time_between = self.utime.ticks_diff(time_now, self.time_start)
        # Set time now as start time
        self.time_start = time_now

        # Calculate temperature change between now and starting point
        temperature_between = temperature_now - self.temperature_begin
        # Set temperature now to start temperature
        self.temperature_begin = temperature_now
        # Calculate heating speed
        self.heating_speed = temperature_between / time_between * 10000 * self.heat-
ing_speed_multiplier

        # Round heating speed
        heating_speed = round(self.heating_speed, 2)

        # Remove initial spike from temperature change speed
        if heating_speed > 100:
            heating_speed = 0

        return heating_speed
```

Koodi 1. HeatingSpeedCalculator luokan toteutus.

Kun lämpötilan muutosnopeus on laskettu, voidaan suorittaa termostaatilla selvitys lämmitystarpeesta siten, että verrataan kattilan lämpötilaa tavoitelämpötilaan (koodi 2). Lisäksi huomioidaan lämmitysnopeus sekä bias. Jos kattilaa esimerkiksi ollaan lämmittämässä isolla lämpöerolla tavoitelämpötilaan, kiihtymisnopeus kasvaa suureksi. Tässä tapauksessa keitin pysäyttää lämmityksen hyvissä ajoin ennen tavoitelämpötilaa. Tässä kohtaa voidaan siis lämmitysnopeudenkertoimella säätää sopivaksi se, kuinka kauan aikaisemmin lämmitys lopetetaan. Jos lämpötila on lämmityksen aloituksessa hyvin lähellä tavoitelämpötilaa, muutosnopeuden vaikutus ennakointiin on paljon pienempi johtuen siitä, että lämpenemisnopeus ei kerkeä kiihtyä suureksi. Lämpötilahaarukka voidaan biasoinnilla säätää tarkasti oikealle alueelle.

```
# If boiler temperature is lower than target temperature - heating speed + bias
if (boiler_temperature < (target_temperature - heating_speed - bias)):

    # Start heating the boiler
    relay_heater.value(1)

    # Heat the virtual boiler
    boiler.heat_up()

    # Otherwise
    else:

        # Stop heating the boiler
        relay_heater.value(0)

        # Cool down the virtual boiler
        boiler.cooldown()
```

Koodi 2. Termostaatin toteutus

4.2.2 Uuttotilan aloitus

Uuttoprosessi koostuu kolmesta päävaiheesta: uuton esivalmistelut eli esilämmitys sekä pre-infuusio, varsinainen uutto ja pehmeä paineenlasku. Uuttotila aloite-

taan, kun uuttotilan kytkin on kytkettynä. Solenoidi kytketään uuttopaineen muodostusasentoon. Tämän jälkeen lasketaan, kuinka pitkään esilämmitystä tehdään ennen pre-infuusiota. Uuttotilan aloituksen toteutus on nähtävissä koodissa 3.

Uuttotilan eteneminen tapahtuu siten, että ensimmäisenä aloitetaan uuttotilan alustava vaihe uuton esilaskentoja varten. Tämän jälkeen suoritetaan tarvittaessa esilämmitys ja pre-infuusio esiohjelmat. Esivaiheiden jälkeen vuorossa on uuttosilmukka. Uuttosilmukka on vaihe, jossa varsinainen uutto tapahtuu. Uuttosilmukan jälkeen viimeinen vaihe on pehmeä paineen lasku. Tämä vaihe on myös valinnainen

```
if switch_brew: # (when not connected to harware)
# if switch_brew.value(): (when connected to hardware)

   # Set solenoid to brewing (pressure) mode
   relay_solenoid.value(1)

   # Calulate pre-heat time before pre-infusion
   pre_heat_time_before_pre_infusion = pre_heat_time - pre_infusion_time

    # Calculate pre-heating time for pre-infusion
    if pre_heat_time_before_pre_infusion > 0:
      pre_heat_time_on_pre_infusion = pre_heat_time - pre_heat_time_before_pre_infu
sion #
   else:
      pre_heat_time_on_pre_infusion = pre_heat_time
```

Koodi 3. Uuttotilan aloituksen toteutus

4.2.3 Esilämmitys

Esilämmitys on uuttoa edeltävä ohjelma. Esilämmitys jakautuu omaan esilämmitysohjelmaansa (katso koodi 4) sekä integroituu pre-infuusio-ohjelmaan (katso koodi 5). Mikäli valittu esilämmitysaika ylittää pre-infuusio ajan, aloitetaan esilämmitysohjelma ennen pre-infuusio-ohjelmaa, mikä kestää ajan, joka muodostuu siten, kun esilämmitys ajasta vähennetään pre-infuusio aika. Muussa tapauksessa esilämmitys tapahtuu pre-infuusio ohjelman sisällä.

```
# If pre-heat time exceeds pre-infusion time: Pre-heat before pre-infusion
if pre_heat_time_before_pre_infusion > 0:

    # Start heating the boiler
    relay_heater.value(1)

    # Create for loop for pre-heat time before pre-infusion
    for x in range(pre_heat_time_before_pre_infusion):

        # Set mode to pre-heat
        brew_data.set_mode("Pre-heat " + str(x + 1) + "s.")

        # Heat up virtual boiler
        boiler.heat_up()

        # Get boiler temperature
        boiler_temperature = boiler.get_temperature() # (when not connected to hardware)
        # boiler_temperature = sensor.read_temperature (when connected on hardware)

        # Print essential values
        print_values(lock_printer, brew_data, boiler, heating_speed, relay_heater, relay_sole-
noid, relay_pump) # (when not connected to harware)

        # print_values(lock_printer, brew_data, sensor, heating_speed, relay_heater, re-
lay_solenoid, relay_pump) # (when connected to hardware)

        # Set delay 1 second
        utime.sleep(1)

# Else stop heating the boiler
else:
    relay_heater.value(0)
```

Koodi 4. Pre-heat esiohjelman toteutus

4.2.4 Pre-infuusio

Pre-infuusio toteutetaan silmukalla, siten että silmukan alussa tarkastetaan, onko esilämmityksen aloitus ajankohtainen. Mikäli on, aloitetaan esilämmitys. Pre-infuusion matalapaine toteutetaan siten, että ensimmäinen noin puoli sekuntia pumppu on käynnissä ja toinen puolisekuntia pumppu on pois päältä. Tällöin saadaan paine laskemaan puolen sekunnin tauon aikana, siten että vesi kulkeutuu kakun sisään. Pre-infuusion toteutus on nähtävissä koodissa 5.

```
# Create loop for the time of the preinfusion
for x in range(pre_infusion_time):
    # Start the pump
    relay_pump.value(1)
    # If pre-heat time matches to the time left with the pre-infusion
    if (pre_heat_time_on_pre_infusion == pre_infusion_time - x):
        # Start pre-heating the boiler
        relay_heater.value(1)
    # If boiler is not heating set mode to "Pre-infusion"
    if relay_heater.value() == 0:
        brew_data.set_mode("Pre-infusion "+ str(x + 1) +"s.")
            # Cood down the virtual boiler
        boiler.cooldown()
    # Else set mode to "Pre-infusion + Pre-heat"
    else:
        brew_data.set_mode( "Pre-heat "+ str(pre_heat_time_before_pre_infusion + x + 1)+
"s. " + "Pre-infusion " + str(x + 1)+ "s.")
        # Cool down the virtual boiler
        boiler.cooldown(0.5)
    # Get the boiler temperature
    boiler_temperature = boiler.get_temperature() # (when not connected to hardware)
    # Set delay 0.5 seconds. this needs to be adjusted for suitable pressure when connected
to hardware
    utime.sleep(0.5)
    # Stop the pump
    relay_pump.value(0)
    ## Set delay 0.5 seconds. this needs to be adjusted for suitable pressure when con-
nected to hardware
# Start heating the boiler for brewing
relay_heater.value(1)
# Start the pump for brewing
relay_pump.value(1)
```

Koodi 5. Pre-infuusio esiohjelman toteutus

4.2.5 Uuttosilmukka

Uuttosilmukka hoitaa varsinaisen uuttamisen, jossa vesi ajetaan täydellä paineella kahvikakun läpi. Uutto-ohjelma aloittaa heti veden lämmittämisen käynnistymisen yhteydessä eikä jää odottamaan termostaattia. Uuttosilmukan toteutus on nähtävissä koodissa 6.

```
while True:

    # Get the boiler temperature
    boiler_temperature = boiler.get_temperature() # (when not connected to hardware)
    # boiler_temperature = sensor.read_temperature() (when connected on hardware)

    # Get calculated heating speed
    heating_speed = heating_speed_calculator.get_heating_speed(boiler_temperature)
    # Set mode to "Brew"
    brew_data.set_mode("Brew " + str(counter_brewing_time) + "s.")

    # Print essential values
    print_values(lock_printer, brew_data, boiler, heating_speed, relay_heater, relay_sole-
noid, relay_pump) # (when not connected to harware)
    # print_values(lock_printer, brew_data, sensor, heating_speed, relay_heater, relay_so-
lenoid, relay_pump) # (when connected to hardware)

    # Cool down virtual boiler
    boiler.cooldown()
     # Set delay 1 second (decrease as fit when connected to hardware)
    utime.sleep(1)
    # Add 1 to brewing time counter
    counter_brewing_time += 1
    # Get the state of the switches
    switch_brew = brew_data.get_brew_switch_state()

    # If brewing switch is turned off brake loop and make (optional) Pressure soft release
    if not switch_brew: # (when not connected to harware)
    # if switch_brew.value() == 0: # (when connected to harware)

        # Reset brewing time counter
        counter_brewing_time = 0
        # Stop heating the boiler
        relay_heater.value(0)

        # Stop the pump
        relay_pump.value(0)
```

Koodi 6. Uuttosilmukan toteutus

Uuton jälkeen aloitetaan mahdollinen pehmeä paineen lasku. Tämä suoritetaan viivyttämällä määritetyn ajan solenoidin kautta paineen vapauttamista, jolloin suurin osa paineesta kerkeää purkautumaan kahvikakun läpi. Pehmeän paineenlaskun toteutus on nähtävissä koodissa 7.

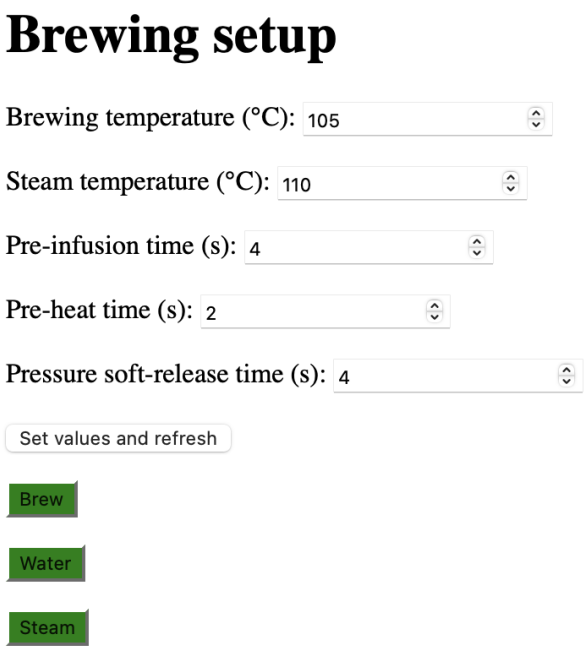
```
# Create loop for the time of pressure release
for x in range(pressure_soft_release_time):

    # Set mode to "Soft pressure release"
    brew_data.set_mode("Soft pressure release " + str(x + 1) +" s.")

    # Print essential values
    print_values(lock_printer, brew_data, boiler, heating_speed, relay_heater, relay_solenoid, relay_pump) # (when not connected to harware)
    # print_values(lock_printer, brew_data, sensor, heating_speed, relay_heater, relay_solenoid, relay_pump) # (when connected to hardware)

    # Set delay for 1 second
    time.sleep(1)

# Release brewing pressure
relay_solenoid.value(0)

# Break the brewing loop
break
```

Koodi 7. Pehmeän paineen laskun toteutus

4.2.6 Käyttöliittymä

Käyttöliittymän (kuva 2) tarkoitus on antaa käyttäjälle mahdollisuus säätää uuttoon liittyviä parametrejä uuton optimointia varten. Käyttöliittymän ja hardwaren välinen tieto kuten kytkinten tila ja asetetut arvot kulkevat brew_data olion välityksellä. Kenttiin syötetyt arvot saa tallennettua set values and refresh -napista. Brew, water ja steam -napit ovat laitteen virtuaalista käyttöä vartin. Brew-kytkin käynnistää uuttotilan, Water-kytkin vesitilan sekä Steam asettaa tavoitelämpötilaksi höyrystyslämpötilan. Painonapit ovat pois päältä -tilassa vihreät, ja kun niitä

painaa, ne menevät päälle ja muuttuvat punaisiksi. Keitin ei reagoi käynnistyksen jälkeen ennen kuin aloitukseen liittyvä pikalämmitys ohjelma on suoritettu.



KUVA 2. Käyttöliittymä. Selainnäkymä.

4.3 Työselostus jälkikäteen

Ohjelmiston kehityksen alussa ensimmäiseksi tehtäväksi muodostui ohjelman jakaminen kahteen säikeeseen. Käyttöliittymäsäikeeseen ja rautapuolen hallintasäikeeseen. Tässä tuli ongelmia thonny:n huonon yhdenaikaisen säieajon integroinnin kanssa. Selvitin ongelmaa, mutta vastausta korjaukseen ei löytynyt. Virhe piti ohittaa irrottamalla laite pariin kertaan tietokoneesta ajojen välillä. Myöskään stop-nappia ei voi käyttää ohjelman ajon pysäyttämiseen johtuen siitä, että thonny pysäyttää säikeet väärin ja se aiheuttaa virhetiloja. Virtalähteellä tätä samaa ongelmaa ei muodostunut. Tämän jälkeen ohjelmiston toteutus tapahtui suunnitelman mukaisesti käyttäen Git-versionhallintaa.

5 TESTAUS

Suurimmaksi osaksi testaus on tapahtunut kehityksen yhteydessä, jolloin itse toimintojen toimivuus sekä yhteistoiminto on testattu. Lopuksi on tehty lopputestaus. Lopputestauksessa testattiin myös sekä ominaisuuksien yksittäinen toimivuus kuten myös ristiin toimivuus. Lopputarkastus:

1. Arvojen asettamiset ja painonapit toimivat hyvin. Ohjelmisto ottaa käyttöön asetetut arvot. Tietokoneeseen ja Thonny:n yhdistettynä lähiverkko yhteyden muodostumisessa aiheutuu ajoittain ongelmia. Tämä on selvitetty ja tämä vika on Thonny:n puutteellisesta toiminnallisuudesta monisäie ajon kanssa. Erikseen virtalähteen päässä pico ei tuota samaa virhettä.
2. Termostaatissa Lämpötila asettuu virtuaaliselle kattilalle hyvin tavoitealueelle. Releen pinnin hallinta vastaa virtuaalisen kattilan lämmönsäätelyä. Jonkin verran biasointia ja kattilan lämpenemiskiihtyvyyskerrointa kuuluu säätää, kuten termostaatti on suunniteltukin käytettäväksi.
3. Uuttotila toimii suunnitellusti. Releen pinnin hallinta kuin myös virtuaalikattilan hallinta toimii suunnitellusti. Integraatio pre-infuusion sekä esilämmityksen kanssa toimii moitteetta.
4. Vesitila toimii suunnitellusti. Releen pinnin hallinta kuin myös virtuaalikattilan hallinta on suunnitelmien mukainen.
5. Höyrytila toimii suunnitellusti. Lämpötila nousee höyrytilassa höyrystyslämpötilaan.
6. Pre-infuusio toimii suunnitellusti. Releiden pinnien hallinta kuin myös ajastus toimii suunnitelmien mukaisesti. Integraatio esilämmityksen sekä uuton kanssa toimii moitteetta.
7. Esilämmitys toimii suunnitellusti. Releiden pinnien hallinta kuin myös ajastus toimii suunnitelmien mukaisesti. Integraatio pre-infuusion sekä uutonkanssa toimii moitteetta.
8. Pehmeä paineen lasku toimii moitteettomasti. Releiden pinnien hallinta kuin myös ajastus toimii suunnitelmien mukaisesti.
9. Asetusten tallennus ja lataus toimii moitteettomasti. Mikrokontrollerin uudelleen käynnistyksen yhteydessä edellisessä ajossa syötetyt uuttoparametrit pysyvät ennallaan.

6 POHDINTA

Työ toteutettiin alkuperäisestä suunnitelmista poiketen pelkästään mikrokontrollerin ohjelmistona. Rajatun suunnitelman mukaan suunnitellut tavoitteet saavutettiin. Ongelmista merkittäväksi jäi Thonny:n puutteellinen toiminto Raspberry Pi Pico:lla moniajoa ajettaessa. Tämä ongelma ei kuitenkaan vaikuta itse laitteen käyttöön, koska virhetilaa ei muodostu virtalähteeseen kytkettynä.

Tulevaisuuden hankkeena olisi mikrokontrollerin kytkentä itse keittimeen. Keittimeen kytkettynä voitaisiin työssä kehittää eteenpäin, esimerkiksi esilämmitystä. Esilämmitys toimii tällä hetkellä siten, että sen vaikutus pitää käyttäjän huomioida uuttolämpötilassa uuttolämpötilaa laskemalla matalammaksi. Tällöin esilämmitys nostaa lämmön suurin piirtein oikealle tasolle. Mittailemalla uuttoveden todellista lämpötilaa, voitaisiin esilämmityksen vaikutus kattilan lämpötilaan laskea automaattisesti termostaatin tavoitelämpötilaan. Myös pre-infuusion ja esilämmityksen yhteisvaikutus olisi hyvä selvittää.

LÄHTEET

Raspberry Pi. n.d. Raspberry Pi Pico. Verkkosivu. Viitattu 6.11.2023. https://www.raspberrypi.com/products/raspberry-pi-pico/)

Thonny. n.d. Thonny IDE. Verkkosivu. Viitattu 6.11.2023. https://thonny.org

George Robotics Limited. n.d. MicroPython. Verkkosivu. Viitattu 6.11.2023. https://micropython.org

Wikipedia. 23.10.2023. HTML. Verkkosivu. Viitattu 6.11.2023. https://fi.wikipedia.org/wiki/HTML

Damien P. George. 6.11.2023. MicroPython kirjasto _thread. Verkkosivu. Viitattu 6.11.2023. https://docs.micropython.org/en/latest/library/_thread.html

Damien P. George. 18.4.2021. MicroPython kirjasto utime. Verkkosivu. Viitattu 6.11.2023. https://docs.micropython.org/en/v1.15/library/utime.html

json.org. n.d. JSON. Verkkosivu. Viitattu 6.11.2023. https://www.json.org

Damien P. George. 6.11.2023. MicroPython kirjasto Machine -> Pin. Verkkosivu. Viitattu 6.11.2023. https://docs.micropython.org/en/latest/library/machine.Pin.html

Damien P. George. 6.11.2023. MicroPython kirjasto socket. Verkkosivu. Viitattu 6.11.2023. https://docs.micropython.org/en/latest/library/socket.html

Damien P. George. 6.11.2023. MicroPython kirjasto network. Verkkosivu. Viitattu 6.11.2023. https://docs.micropython.org/en/latest/library/network.html)

LIITTEET

Liite 1. Opinnäytetyön GIT-repository. https://github.com/Rolppe/silvia_pico