Osaamista
ja oivallusta
tulevaisuuden
tekemiseen

Juuso Helmijoki

Pienkanalan automaatio

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Sähkö- ja automaatiotekniikka

Insinöörityö

1.6.2021

Tekijä Otsikko Sivumäärä Aika	Juuso Helmijoki Pienkanalan automaatio 32 sivua 1.6.2021
Tutkinto	insinööri (AMK)
Tutkinto-ohjelma	sähkö- ja automaatiotekniikka
Ammatillinen pääaine	automaatiotekniikka
Ohjaajat	lehtori Markku Inkinen

Insinöörityön tavoitteena oli rakentaa pienkanalaan etäohjattava automaatiojärjestelmä valaistuksen ja lämmityksen säätelyyn. Kanalan toiminnan kannalta tarkoituksena oli parantaa eläinten oloja, helpottaa eläinten hoitamista sekä säästää energiaa. Automaatiojärjestelmän määrittely tehtiin yhdessä kanojen hoitajan kanssa.

Projektin aikana suunniteltiin ja rakennettiin järjestelmä, ohjelmoitiin tarvittavat toiminnot sekä testattiin järjestelmä. Järjestelmän ohjaimena toimi Raspberry Pi, ja sähkölaitteiden ohjaamiseen rakennettiin jakokeskus yhteistyössä sähköurakoitsijan kanssa. Järjestelmän ohjaamiseen tehtiin web-selaimella toimiva asetussivusto, jossa oli mahdollisuus nähdä asetettu- ja mitattuja arvoja sekä muuttaa niitä.

Automaatiojärjestelmä osoittautui testeissä täsmällisesti toimivaksi. Järjestelmä palautui onnistuneesti sähkökatkoista, ja valaistus toimi määritellysti. Projekti valmistui lämpimänä vuodenaikana, joten varsinaisia mittaustuloksia energiankulutuksesta ei projektin valmistuessa pystytty tekemään. Testiolosuhteissa lämmityksen säätö toimi halutulla tavalla. Automaatiojärjestelmä jäi valmistuttuaan heti käyttöön.

Tässä insinöörityössä kehitettyä järjestelmää on mahdollista käyttää myös moniin muihin sovelluksiin, joissa tarvitaan etähallintaa tai monipuolisia ohjelmointimahdollisuuksia. Tuotteistetulla vastaavalla järjestelmällä olisi lukuisia käyttömahdollisuuksia, mutta se vaatisi innovointia käytettävyyteen ja ohjelmoinnin yksinkertaisuuteen. Uudet ominaisuudet, kuten ilmanlaadun mittaus ja ilmanvaihdon ohjaus, täydentäisivät järjestelmää energiansäästössä ja eläinten hyvinvoinnin seuraamisessa ja parantamisessa.

Avainsanat	Automaatio, etähallinta, pienkanala, Raspberry Pi

Author Title Number of Pages Date	Juuso Helmijoki Chicken Coop Automation 32 pages 1 June 2021
Degree	Bachelor of Engineering
Degree Programme	Electrical and Automation Engineering
Professional Major	Automation Engineering
Instructors	Markku Inkinen, Senior Lecturer

The aim of the Thesis work was to develop and build an automation system for controlling chicken coop temperature and lighting. The system would improve animal well-being and animal care and save energy. The system specification was developed in co-operation with the animal attendant.

During the project, automation system was designed and built, functions were programmed, and the system was tested. Raspberry Pi was the controller of the system and for controlling of electronic devices, switchgear was built in co-operation with the electrical contractor. For the controlling of the system, web browser compatible configuration environment was created.

In the test phase, the created automation system worked precisely. The system recovered automatically from power outage and lighting functioned as described. It was not possible to measure electricity saved by the system because the project realization finished in warm season. In the test environment, controlling of the heaters worked as described. After the project was finished, the system was taken into use.

The system developed in this Thesis work is suitable for many other applications where remote controlling or versatile programming properties are needed. By productization, the system would have many applications but it would require improvements in usability and a user-friendlier programming interface. New features, such as measuring of air quality and controlling of ventilation would give improvements in energy saving and in following and improving animal well-being.

Keywords	Automation, Chicken coop, Raspberry Pi, remote control

Sisällys

Lyhenteet

1 Johdanto 1

2 Kanalan olosuhteet 2

2.1 Lämpötila 2
2.2 Valaistus 2
2.3 Tarve automaatiolle 3

3 Suunnittelu 4

3.1 Työn kulku 4
3.2 Ohjain 5
3.3 Jakokeskus 5
3.4 Ohjelmointi 5
3.4.1 Termostaattiohjelma 5
3.4.2 Valaistusohjelma 6
3.4.3 Käyttöliittymä 7
3.4.4 Lokitiedot 8

4 Toteutus 8

4.1 Ohjainkeskuksen komponentit 9
4.1.1 Raspberry Pi 10
4.1.2 Liitinkortti 12
4.1.3 Relekortti 12
4.1.4 Liittimet 12
4.1.5 Johdotukset 14
4.1.6 RTC 14
4.1.7 Lämpötila-anturit 16
4.2 Jakokeskus 18
4.3 Ohjelmisto 19
4.3.1 Flask 19
4.3.2 Cron 24

4.4 Käyttöliittymä 25
5 Yhteenveto 28
Lähteet 30

Lyhenteet

CSS	Cascade Style Sheet. Web-sivujen tyyliohje selaimelle sivuston esittämiseksi halutulla tavalla.
GPIO	General-purpose Input/Output. Raspberry Pi:n sähköiset lähdöt ja tulot.
HTML	Hyper Text Markup Language. Yleisesti web-sivujen tekemissä käytetty kieli.
PIP	Python Package Index. Python-ohjelmointikielen pakettien asentamiseen ja hallintaan tarkoitettu työkalu.
PWM	Pulse Width Modulation. Pulssinleveysmodulaatio.
RTC	Real Time Clock. Paristolla varmennettu kello, joka on varustettu paristovarmennuksella
URL	Uniform Resource Locator. Web-osoite, joka kertoo tiedon sijainnin.
WSGI	Web Server Gateway Interface. Määrittely, kuinka web-palvelin kommunikoi web-sovellusten kanssa.

1 Johdanto

Tämä insinöörityö sai aiheen tarpeesta automatisoida pienkanalan toimintoja. Kanalassa oli käytössä useita erilaisia lämmittimiä omilla termostaateilla ja valaistusta ohjattiin kellokytkimellä. Lämmittimien termostaatit eivät olleet kovin tarkkoja eikä lämpötila pysynyt vakaana. Termostaattien epätarkkuus aiheutti myös turhaa lämmitystä, joten lämmityksen sähkönkulutus oli suurta. Alueella usein toistuvat sähkökatkot myös sotkivat ajastuksen. Lämmityksen ja valaistuksen ohjauksen epätarkkuudet eivät myöskään ole kanoille hyväksi.

Markkinoilta ei löytynyt suoraan sopivaa ratkaisua automaatiojärjestelmän toteuttamiseen, joten järjestelmä oli ainakin joiltain osin rakennettava komponenteista. Automaatiojärjestelmän pohjautuminen Raspberry Pi:hin perustui sen hintaan ja mahdollisuuksiin lukea antureita, ohjata ulkopuolisia laitteita sekä ohjelmointimahdollisuuksiin ja helppoon yhdistettävyyteen.

Työn tavoitteena on parantaa eläinten hyvinvointia nykyistä tasaisemmilla olosuhteilla ja helpottaa kanojen hoitotyötä. Lisäksi tavoitteena on säästää energiaa ja sen avulla saavuttaa aiempaa pienempi hiilijalanjälki. Näitä tavoitteita varten luodaan etäohjattava järjestelmä, jolla voidaan ohjata kanalan lämmittimiä ja valaistusta. Järjestelmän on oltava myöhemmin hyödynnettävissä muihin kanalan automaation käyttötarkoituksiin. Lisäksi käyttäjälle on esitettävä käyttöliittymässä järjestelmän tilaan liittyviä tietoja.

Kanala on Pielaveden kunnassa sijaitseva pienkanala, jossa on noin 30 harvinaisen kiuruvedenkannan maatiaiskanaa. Kanala kuuluu Luonnonvarakeskuksen maatiaiskanojen säilytysohjelmaan, jonka tavoitteena on säilyttää alkuperäisrotuja ja eläingeenivarantoja. [1.]

Tässä raportissa käsitellään ensin vaatimuksia kanalan automaatiojärjestelmälle. Sen jälkeen kerrotaan suunnittelusta ja toteutuksesta. Lopuksi käsitellään tulevaisuuden mahdollisuuksia.

2 Kanalan olosuhteet

Tässä luvussa käsitellään automaatiojärjestelmän suunnitteluun vaikuttavia kanalan vaatimuksia. Eläinsuojelulaki ja maatiaiskanojen säilytysohjelma asettavat vaatimuksia kanojen olosuhteilla. Oikeat olosuhteet ovat tärkeitä eläinten hyvinvoinnin ja tuottavuuden näkökulmasta. Lisäksi olosuhteiden ylläpidon ja seuraamisen tulee olla helppoa kanalan hoitajalle.

2.1 Lämpötila

Lämmityksen tärkeä tehtävä kanalassa on luoda yhdessä ilmanvaihdon kanssa eläimen hyvinvoinnin ja aineenvaihdunnan kannalta hyvä huoneilmasto. Rehunkulutuksen kannalta lattiakanalan optimaalinen lämpötila on +20–22 °C. Maatiaiskanojen säilytysohjelmassa vielä +5 °C:n lämpötila on hyväksyttävä. Liian korkea tai matala lämpötila aiheuttaa stressiä ja muita hyvinvointiongelmia. Lämpötila on myös tärkeä tekijä kosteuden säätelyssä. [2, s. 96, 107.]

Lämpötilaa on mitattava sieltä missä kanat oleskelevat. Päivisin sisällä ollessaan kanat ovat pääasiassa lattialla kuopsuttamassa pehkua tai munintapesissä munimassa. Iltaisin kanat siirtyvät orrelle nukkumaan. [3, s. 15, 30.]

2.2 Valaistus

Sopivalla valaistuksella on suuri merkitys kanan aivolisäkkeen toimintaan, mikä vaikuttaa hormonitoimintaan. Hormonit säätelevät myös kanan munintaa, joten oikeanlaisella valaistuksella on tärkeä merkitys hyvinvoinnin lisäksi myös munintaan. Tuotantokana pysyy tuottavana ja munii vuoden ympäri, kun se saa riittävästi valoa. Lyhyt valoisa aika johtaa sulkasatoon. Maatiaiskanalassa tuotetaan kanalle sulkasato valaistusaikaa lyhentämällä. Valaistusrytmin on oltava 24 tuntia ja yhtäjaksoisen pimeän ajan on oltava vähintään kahdeksan tuntia. Valon valinnassa ja ohjauksessa on otettava huomioon, että kana havaitsee valon välkkymisen jopa 250 Hz:n taajuudella. Valolle voidaan käyttää himmennintä tai pienkanalassa voidaan käyttää myös erillistä himmeää yövaloa. [2, s. 58, 59, 97, 109.]

2.3 Tarve automaatiolle

Käytäntö on osoittanut, ettei kanaloihin myytävien lämmittimien termostaatit ole usein kovin tarkkoja. Kanalan lämpötilat ovat voineet olla yli 10 °C haluttua lämpötilaa korkeampia, jolloin energiahukka on suurta ja eläinten olot eivät pysy tasaisina. Arvio energiankulutuksen säästöstä aiempaa tarkemman ohjauksen myötä on yli 50 % kalenterivuoden aikana.

Alueella usein toistuvat sähkökatkot haittaava pistorasiaan kytkettävien ajastimien toimintaa. Automaatiojärjestelmän avulla sähkökatkoista toipuminen on tavoitteena saada automaattiseksi. Lisäksi valaistuksen ajastukseen kehitettävällä järjestelmällä olisi helppoa tehdä muutoksia kanojen päivärytmiin.

3 Suunnittelu

Tässä luvussa kerrotaan projektin vaiheista ja työn eri osien suunnittelusta. Lopussa kerrotaan, kuinka toimintoja ohjaavat ohjelmat ovat suunniteltu. Suunnittelu tehtiin yhdessä kanojen hoitajan kanssa perehtyen kanojen hoito-ohjeisiin. Suunnittelussa käytettiin AutoDesk Fusion 360-, CADMATIC-, Fritzing- ja UMLET -ohjelmia.

3.1 Työn kulku

Projektin suunnittelu- ja toteutusprosessiksi valittiin vesiputousmalli. Projektissa tehdään ensimmäinen versio laitteesta, joten on etukäteen oletettavissa, että suunnitelmaan tulee muutoksia.

Tämän insinöörityön suorittamiseen määritettiin seuraavat vaiheet:

- projektin aloituspalaveri ohjaavan opettajan kanssa
- perehtyminen vaatimuksiin ja vaatimusmäärittelyn luonti
- ohjaimen ja pistokekotelon suunnittelu
- ostot
- ohjaimen ja pistokekotelon rakentaminen
- ohjelmointi
- FAT-testi
- asennus kanalaan
- (SAT-testi)
- loppu.

Koska insinöörityön on määrä valmistua lämpimänä vuodenaikana, SAT-testauksen onnistuminen on epävarmaa. Lämmityksen ohjauksen toimintaa ei voida testata, jos lämmitystarvetta ei ole. Toiminnallisuus voidaan kuitenkin tarkastaa lämmittämällä lämpötila-anturia käsin.

3.2 Ohjain

Järjestelmälle rakennetaan ohjain asennuskoteloon, johon sijoitetaan vain pienoisjännitteisiä komponentteja. Ohjaimeen tulee neljä liitäntää 1-wire-antureille sekä liitin jakokeskuksen liittämiseksi. Ohjaimen on toivuttava automaattisesti sähkökatkoista. Järjestelmän on oltava helposti purettavissa puhdistusta tai huoltotyötä varten. Virransyötön, antureiden ja pistokekotelon kiinnitykset on toteutettava helposti kytkettävillä pistokeliittimillä. Ohjainkotelolle tuodaan 12 V:n tasoinen jännite. Relelähtöjä on oltava kahdeksan. Ohjainlaitteeksi valittiin suunnitteluvaiheessa Raspberry Pi. Ohjaimen on oltava pöly- ja roiskevesisuojattu kanalan pölyisten olosuhteiden vuoksi.

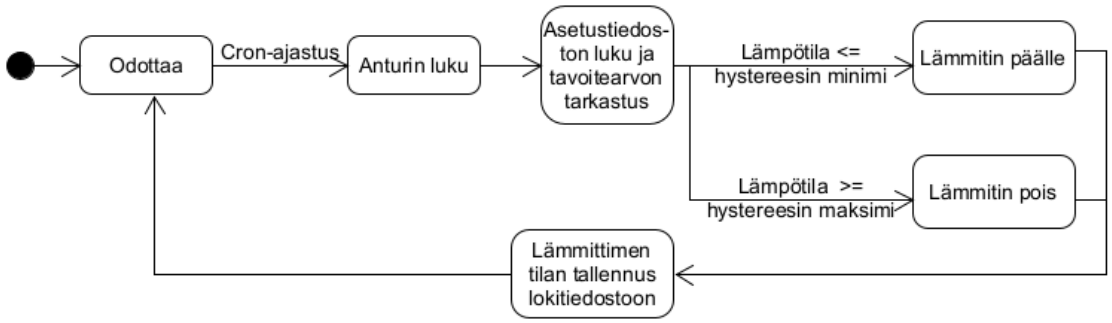
3.3 Jakokeskus

Jakokeskus tehdään yhteistyössä paikallisen sähköurakoitsijan kanssa. Jakokeskukseen asennetaan kanteen kuusi sukopistorasiaa ja kunkin pistorasian tilan osoittava merkkivalo. Koteloon asennetaan 12V:n jännitteellä ohjatut releet. Pistorasiat asennetaan kahteen ryhmään ja varustetaan kahdeksan ampeerin ylivirtasuojilla. Jakokeskuksen on oltava pöly- ja roiskevesisuojattu kanalan pölyisten olosuhteiden vuoksi.

3.4 Ohjelmointi

3.4.1 Termostaattiohjelma

Termostaatin toimintaa varten luodaan ohjelma, joka sammuttaa tai käynnistää lämmittimen asetusarvojen mukaisesti. Ohjelman toimintaa kuvaava tilakaavio on kuvassa 1. Käyttäjä määrittää käyttöliittymässä tavoitelämpötilan, joka tallennetaan asetustiedostoon. Lisäksi ohjelmaan määritetään hystereesiksi yksi celsiusaste. Kun Cron käynnistää termostaattiohjelman, anturin arvo luetaan, asetustiedosto luetaan ja lasketaan hystereesin avulla lämpötilan maksimiarvo ja minimiarvo. Maksimiarvo kuvaa lämpötilaa, jossa lämmitin sammutetaan, minimiarvo kuvaa lämpötilaa, jossa lämmitin käynnistetään.



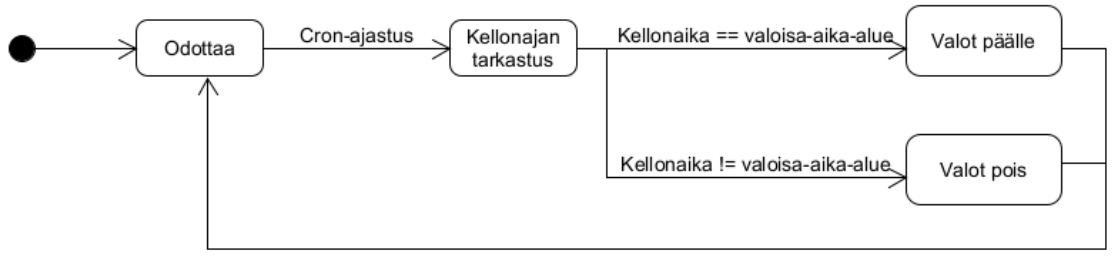
Kuva 1. Termostaattiohjelman tilakaavio

Mikäli syntyy poikkeustilanne, jossa termostaattiohjelma ei toimi halutulla tavalla, on tärkeää varmistaa riittävän lämpötilan pysyminen kanalassa. Tätä varten termostaattiohjelman tulee tarkastaa, onko lämpötilan asetusarvo tasolla, joka ei vaaranna kanojen hyvinvointia. Asetusarvoihin on koodattu minimilämpötila 0 °C ja maksimilämpötila 20 °C. Käyttäjä ei pääse näitä arvoja muokkaamaan.

Termostaattiohjelma on kirjoitettava myös asetustiedostoon: onko lämmitin päällä vai pois päältä. Lokitiedoston avulla voidaan laskea lämmittimen käyttöaste.

3.4.2 Valaistusohjelma

Valaistuksen säätämiseen on tehtävä ohjelma, jonka Cron-ajastin suorittaa. Ohjelma lukee asetustiedostosta valaistuksen syttymis- ja sammutusajat ja sytyttää tai sammuttaa valot asetusten mukaisesti. Kuvassa 2 on valaistuksen ajastus esitetty tilakaavion avulla.



Kuva 2. Valaistusohjelman tilakaavio

3.4.3 Käyttöliittymä

Jotta kanalan hoitaja voi ohjata järjestelmää, tarvitaan selkeä graafinen käyttöliittymä. Perinteinen HTML5-pohjainen sivusto toimii parhaiten tähän tarkoitukseen, koska sama sivu toimii PC:llä ja mobiililaitteella [4].

Käyttöliittymässä päädyttiin näyttämään alla listatut arvot.

- lämpötila
- lämpötilan asetusarvo
- hystereesi
- valot
- valot Päälle
- valot pois
- valoisa aika
- lämmittimien kolmen tunnin käyttöaste
- lämpötilan kolmen tunnin keskiarvo.

Käyttöliittymässä on pystyttävä helposti muuttamaan valittuja asetusarvoja. Käyttöliittymän taustalla pyörivän ohjelman on myös tarkastettava käyttäjän syöttämien asetusarvojen oikeellisuus. Alla on lueteltu käyttäjän muokattavissa olevat arvot.

- lämpötilan asetusarvo
- valot päälle
- valot pois.

Käyttöliittymässä on estettävä, ettei käyttäjä voi hätiköiden muuttaa väärää arvoa. Tämä toteutetaan käyttöliittymän pääsivulla olevilla painikkeilla, joista valitaan muutettava arvo. Painikkeesta aukeavalla uudella sivulla on kerrottava, mitä arvoa muutetaan ja minkälaiset arvot ovat sallittuja.

3.4.4 Lokitiedot

Järjestelmän on tallennettava lokitiedostoon lämpötila-arvot 15 minuutin välein sekä lämmittimen päälle- ja pois-kytkennät. Arvot tallennetaan omiin CSV-muotoisiin tiedostoihin. Yli 24h vanhat lokitiedot poistetaan tallennuksen yhteydessä.

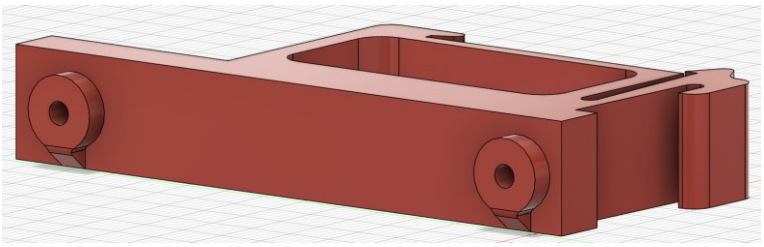
4 Toteutus

Järjestelmä rakennettiin kahteen asennuskoteloon. Ohjainkeskuksessa ovat kaikki pienoisjännitteiset komponentit. Sähköurakoitsijan kanssa yhteistyössä rakennetussa pistokekotelossa on kaikki pienjännitteiset komponentit. Komponenttien erottelulla kahteen koteloon haluttiin rakentaa kokonaisuus, jota maallikko pystyy kehittämään sekä eristämään mahdollisimman pieni komponenttimäärä sähköalan ammattihenkilön pätevyyden vaatimaan kokonaisuuteen. Ohjainkeskus ja jakokeskus on esitetty kuvassa 3 ulkoisesti.

Kuva 3. Ohjainkeskus vasemmalla ja jakokeskus oikealla

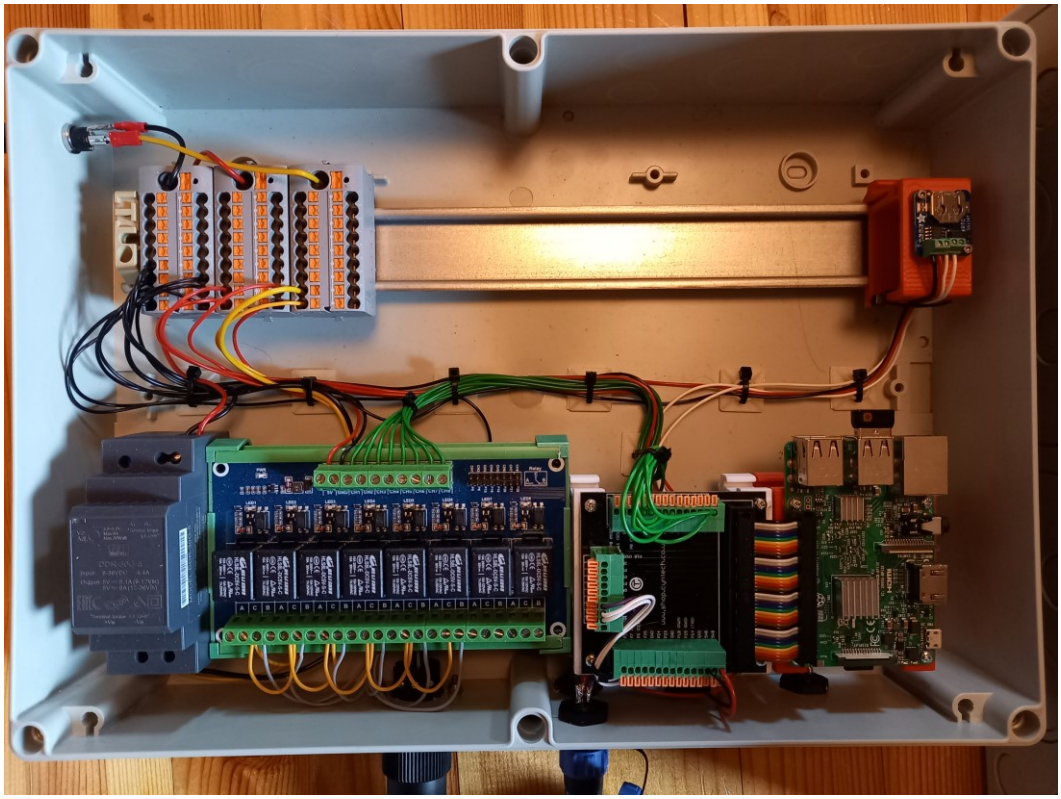
4.1 Ohjainkeskuksen komponentit

Järjestelmän ohjauskeskus rakennettiin tukuista löytyvään asennuskoteloon. Asennuskotelon pituus ja leveys olivat 361*254 mm ja syvyys 111 mm. Komponenttien paikat määritettiin sommittelemalla ne asennuskotelon pohjalle. Step-Down-virtalähde, potentiaalinjakoliittimet ja relekortti olivat DIN-kiskokiinnitteisiä. Raspberry Pi:lle, RTC:lle ja liitinkortille suunniteltiin sopivat kiinnikkeet Fusion 360 -suunnitteluohjelmalla tulostettavaksi 3D-tulostimella. Esimerkki 3D-tulostettavasta DIN-kiskokiinnikkeestä on kuvassa 4. Ohjainkeskuksen koteloon ja siihen kiinnitettäviin ulkopuolisiin osiin valittiin vähintään IP54-luokituksen komponentit, jotta vaatimus pöly- ja roiskevesitiiviydestä täyttyy.



Kuva 4. DIN-kisko kiinni Raspberry Pi:lle Fusion 360 3D -suunnitteluohjelmassa.

Kuvassa 5 on ohjainkeskuksen kotelon kansi avattuna. DIN-kiskolle on jäänyt hyvin tilaa myöhemmin mahdolllisesti tehtäville päivityksille.



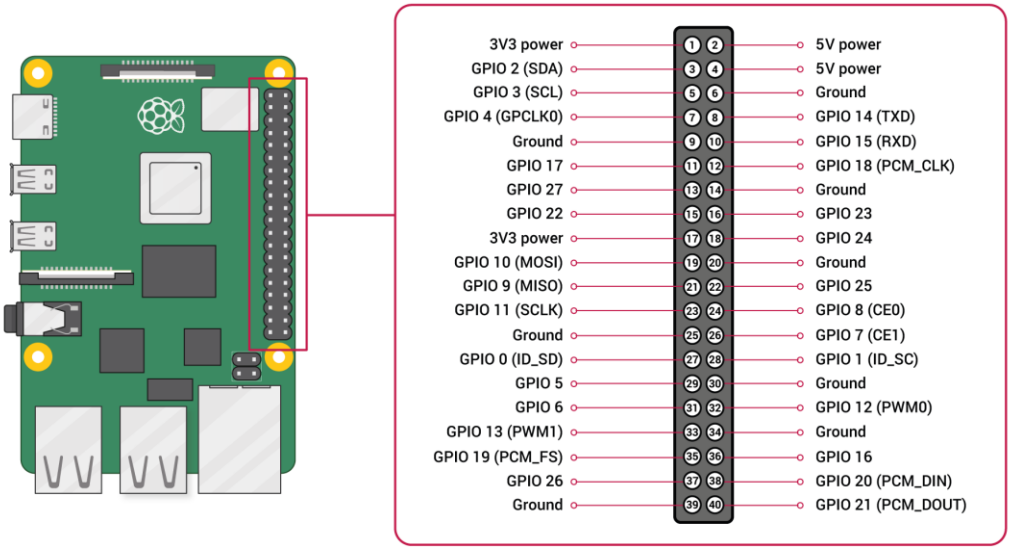
Kuva 5. Ohjainkeskuksen kotelon kansi avattuna.

4.1.1 Raspberry Pi

Raspberry Pi on tietotekniikan kouluissa opettamiseen kannustavan Raspberry Pi Foudationin luoma pieni yhden piirilevyn tietokone, josta on ilmestynyt useita versioita [5]. Raspberry Pi:tä käytetään yleisesti myös kohdeyleisön ulkopuolella ja siitä on tullut hyvin yleinen harrastelijoiden ja tutkijoiden keskuudessa [6, 7]. Lisäksi Raspberry Pi:stä on tehty kaupallisia tuotteita [8]. Virallinen tuettu käyttöjärjestelmä on Raspberry Pi OS, jonka käyttäjä asentaa itse [9].

Raspberry Pi:ssä on 40 pinniä, joilla voidaan kytkeä laitteeseen muita komponentteja. Pinneistä 12 on virtapinnejä; loput pinneistä on yleiskäyttöisiä pinnejä. Pinnejä voidaan käyttää kytkiminä, käyttää pulssinleveysmodulaatioon ja kommunikointiin erilaisilla väylillä. [10.] Kuvassa 6 on pinnien käyttötarkoituksia havainnollistava kuva. Taulukossa 1 esitellään tässä projektissa käytetyt pinnit.

Raspberry Pi valittiin tähän projektiin, koska sitä on käytetty paljon vastaavissa sovelluksissa ja siihen löytyy paljon sopivia lisäosia. Avoimuuden avulla rajoitteita ei juuri ole, ja hinta harrasteprojektiin on sopiva. Vasta-argumenttina Raspberry Pi:n valinnalle oli ei-teollisuuslaatu sekä toiminta vikatilanteissa. Jokainen laite voi vikaantua, mutta Raspberry Pi:lle löytyy tarvittaessa korvaaja tai uusi muistikortti toisesta projektista, joten Raspberry Pi:n katsottiin toimivaksi tähän projektiin. Viimeisessä luvussa on käsitelty varautumista vikaantumistilanteisiin.



Kuva 6. Raspberry PI:n GPIO [10]

Taulukko 1. GPIO-lähtöjen/-tulojen käyttö

Pinni	**Käyttö**	**Käyttö**
GPIO 2 (SDA)	RTC SDA	RTC-tiedonsiirto
GPIO 3 (SLA)	RTC SDL	RTC-tiedonsiirto
GPIO 4	1-wire	Väylä antureiden lukuun
GPIO 5	Relekanava #1	Jakokeskus/Lämmitin
GPIO 6	Relekanava #2	Jakokeskus/Valaistus
GPIO 13	Relekanava #3	Jakokeskus/Varalla
GPIO 16	Relekanava #4	Jakokeskus/Lämmitin
GPIO 19	Relekanava #5	Jakokeskus/Varalla
GPIO 20	Relekanava #6	Jakokeskus/Varalla
GPIO 21	Relekanava #7	Varalla
GPIO 26	Relekanava #8	Varalla

4.1.2 Liitinkortti

Raspberry Pi:n piikkirimaliitinten luotettavuus voi olla heikkoa, koska kokemusperäisesti piikkirimasta liittimet irtoavat helposti. Liittimen tahottoman irtoamisen välttämiseksi projektiin päätettiin asentaa erillinen GPIO-lisäosa, jossa on jousiliittimet johdoille. Jousiliitinten selkeä merkitseminen auttoi kytkentöjen tekemisessä.

Liitinkortti liitettiin Raspberry Pi:hin lattakaapelilla. Lattakaapeli lyhennettiin sopivan mittaiseksi. Asennuksen ja muokkauksen onnistuminen varmistettiin mittaamalla liitinkortilta ennen muiden laitteiden kytkentää.

4.1.3 Relekortti

Releiden ohjaamiseen tarvitaan erillisiä komponentteja GPIO-lähtöjen suojaamiseksi [11]. Kahdeksankanavaisella relekortilla ohjauskoteloon saadaan tarvittava määrä lähtöjä pistokekotelon pistokkeiden ohjaamiseen sekä kaksi ylimääräistä relelähtöä tulevaisuuden tarpeita varten. Virransyöttö toteutettiin ulkopuolisella 12 V:n muuntajalla ja sisäisellä 5V DC-DC step-down -muuntajalla.

Pistokekotelon ohjausta varten asennettiin kahdeksankanavainen relekortti, jossa jokainen rele on varustettu optoerottimella [12]. Relekortissa olvati valmiina ruuviliittimet johtojen kiinnitystä varten. Relekortin ohjaus johdotettiin suoraan Raspberry Pi:n pinneistä. Relekortin 12 V:n lähdöt kytkettiin pistokekoteloon lähtevälle liittimelle. Relekortissa oli valmiina piikkirimakiinnitys Raspberry Pi:tä varten. Tätä osaa ei kuitenkaan työssä haluttu käyttää, koska GPIO-lähtöihin käytettiin erillistä liitinkorttia. Tämä osa piirilevystä oli helposti poistettavissa taittamalla piirilevy esivalmisteltua katkaisukohtaa pitkin tilan säästämiseksi.

4.1.4 Liittimet

Kanalan siivoamisen ja laitteen puhdistamisen, kehittämisen ja korjauksen ajaksi ohjainlaitteet on mahdollisesti poistettava kanalasta. Ohjaimeen valittiin tämän vuoksi helposti irrotettavat liittimet. Liittimien valintaan vaikuttivat napojen määrä, hinta ja saatavuus. Antureiden ja releiden ohjauksen virrat eivät asettaneet vaatimuksia liittimille.

Jakokeskuksen kytkemiseksi ohjaimeen käytettiin seitsemännapaista Binder-liitintä, joka näkyy kuvassa 7. Liittimellä on IP67-luokitus ja siinä on kierrelukitus [13]. Liittimen napoja 1–6 käytettiin relekortin kanavien numeroinnin mukaisesti. Liitintä 7 käytettiin 0V-johtimena.

Kuva 7. 7-napainen Binder-liitin [13]

One Wire -antureiden liittämiseksi ohjaimeen käytettiin kolme napaisia SP13-liittimiä (kuva 13). Liittimet ovat IP68-luokiteltuja [14]. One Wire-väylän vaatima ylösvetovastus juotettiin yhden paneeliliittimen 2- ja 3-napoihin kiinni taulukon 2 mukaisesti.

Kuva 8. SP13-liittimen napojen numerointi [14]

Taulukko 2. One Wire -antureiden liittimien numerointi

Liitin	**Käyttö**
1	0V
2	One Wire
3	+5V

Jakokeskuksen virransyöttö toteutettiin 3x1,5 mm^2 H05RN-F SUKO -laitejohdolla. Ohjaimen virransyöttöä varten koteloon asennettiin DC 2,1/5,5 -liitin. Ohjaimen ja jakokeskuksen välisessä johdossa käytettiin ÖLFLEX CLASSIC 110 7x0,75 -ohjauskaapelia.

4.1.5 Johdotukset

Jakokeskuksen sisäisissä johdotuksissa käytettiin MKEM-kaapeleita, joiden värit täyttävät standardin SFS-EN 60446 vaatimukset. Johtimien päät holkitettiin luotettavan liitoksen varmistamiseksi [15].

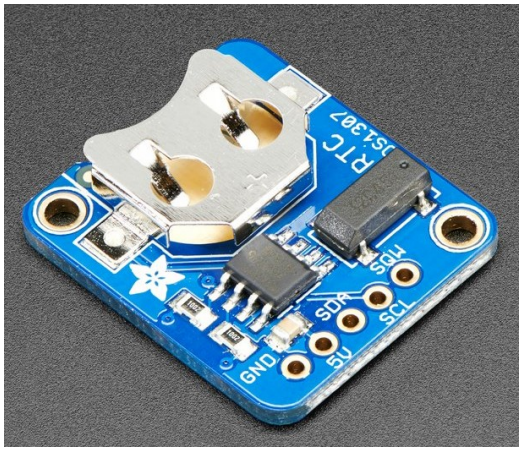
Ohjaimen johdotukset tehtiin pääasiassa PVC-päällisteisellä hienosäikeisellä asennusjohdoilla, joiden halkaisijat ovat 0,50 mm² tai 0,35 mm². Johtimien päät tinattiin. Värikoodaus on tehty taulukon 3 mukaisesti.

Taulukko 3. Ohjaimen johtimien värikoodaus

Väri	**Väri**	**Käyttö**
	Musta	0V
	Oranssi	+3,3V
	Punainen	+5V
	Keltainen	+12V
	Sininen	INPUT
	Vihreä	OUTPUT
	Valkoinen	I²C-väylä
	Harmaa	Releiden ohjaukset
	Violetti	OneWire

4.1.6 RTC

Raspberry Pi:ssä ei ole paristolla varmistettua RTC-kelloa, joten virtojen jostain syystä katketessa kellonaika menetetään. Kellonajan varmistamiseen käytettiin erillistä DS1307-piiriin perustuvaa piirikorttia (kuva 9). Raspberry Pi OS -käyttöjärjestelmän oletusasennukseen sisältyy ohjelmallinen valekello, joka poistetaan RTC-kellon asennuksessa [16].



Kuva 9. DS1307-piirikortti [16]. Alaosaan juotettiin ruuviliittimet.

Piirikortille syötetään 5V-jännitettä. Kommunikointi Raspberry Pi:n kanssa tapahtuu I²C-väylällä. Piirikortin SDA- ja SLA-liittimet kytketään Raspberry Pi:n vastaaviin. [16.] Piirikorttiin juotettiin piikkirimaliittimen asemesta ruuviliittimet luotettavan liitoksen aikaansaamiseksi.

Raspberry Pi tarvitsee python-smbus- ja i2c-tools-paketit, jotka sisältävät tarvittavat ohjelmat DS1307-piirin kanssa kommunikointiin I²C-väylällä [16]. I²C-väylään kytketyt laitteet voidaan etsiä i2cdetect-komennolla. Komento palauttaa käyttäjälle väylään kytketyt laitteen ja niiden osoitteet. Osoitetaulukko on esitetty kuvassa 10. [17.]

```
     @       :/dev $ sudo i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- 68 -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

Kuva 10. I2cdetect-komennon listaus löydetyistä I²C-väylään kytketyistä laitteista.

RTC-piiri pitää lisätä *device tree* -määrittelyyn. Se tehdään lisäämällä hakemistossa /boot/config.txt sijaitsevaan tiedoston loppuun teksti dtoverlay=i2c-rtc,ds1307. RTC-piiri on käytettävissä järjestelmän uudelleenkäynnistyksen jälkeen. [16.]

Raspberry Pi OS oletusasennuksessa ollut ohjelmallinen ”valekello” on poistettava, jotta RTC-piirin kellonaikaa voidaan käyttää. Fake-hwclock-paketti poistetaan paketinhallinta-ohjelmalla. Updare-rc.d-komennolla poistetaan fake-hwclock käynnistysskriptistä sekä systemctl-komennolla poistetaan järjestelmän palveluista. [16.]

Fake-hwclock-paketin poistamisen jälkeen on vielä poistettava viittaukset ”vanhaan” kelloon /lib/dev/hwclock-set -asetustiedostosta kommentoimalla esimerkkikoodissa 1 esitetyt rivit [16].

```
#if [ -e /run/systemd/system ] ; then
# exit 0
#fi
---
#/sbin/hwclock --rtc=$dev --systz --badyear
---
#/sbin/hwclock --rtc=$dev -systz
```

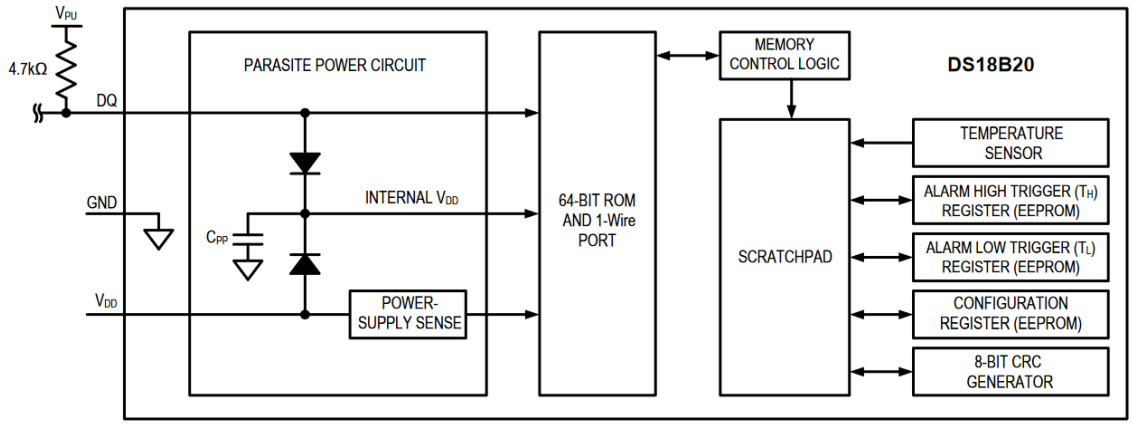
Esimerkkikoodi 1. /lib/dev/hwclock-set-asetustiedostosta kommentoitavat rivit

Kun Raspberry Pi on kytketty internetiin, järjestelmän aika päivitetään oletuksena automaattisesti. RTC-kellon aikaa hallitaan hwclock-komennolla. Hwclock-komennon -r-funktiolla tulostetaan RTC-kellon kellonaika ja -w-funktiolla järjestelmän kellonaika päivitetään uudeksi RTC-kellonajaksi. [18.]

4.1.7 Lämpötila-anturit

Lämpötila-anturiksi valittiin DS18B20-anturit. Kyseinen anturi tuottaa digitaalisen 9- tai 12-bittisen lukeman. Anturi kommunikoi Raspberry Pi:n kanssa 1-Wire-väylällä. Jokaisella DS18B20-anturilla on uniikki 64-bittinen sarjanumero. Uniikin sarjanumeron ansiosta samaan 1-wire-väylään voidaan kytkeä useita DS18B20-antureita ja erottaa antureiden lukemat toisistaan ohjelmallisesti. [19.] Anturista löytyy versioita, jotka on koteloitu metalli- tai muovikoteloon ja sisältävät sähköjohdon. Tähän projektiin valittiin metallikoteloon paketoitu anturi viiden metrin johdolla.

1-wire-väylään tarvitaan ylösvetovastus, joka kiinnitettiin ohjaimen liittimiin 1-Wire-datalinjan DQ ja jännitetulon V_{DD}-välille. Sopivan vastuksen arvo on 4,7 KOhm [19]. V_{DD} liitettiin ohjaimen 3V-liittimeen. GND liitettiin ohjaimen 0V-liittimeen.



Kuva 11. DS18B20-sensorin toimintaa kuvaava lohkokaavio [19]

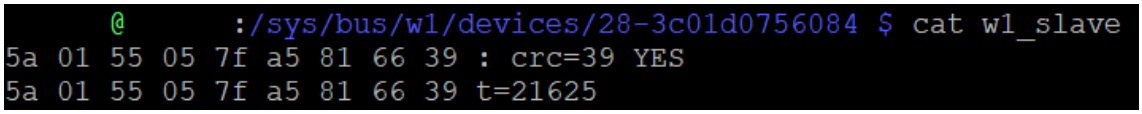
1-Wire-väylän käyttöönottamiseksi se on aktivoitava Raspberry Pi:n device tree -määrittelyyn. Se tehdään lisäämällä hakemistossa /boot/config.txt sijaitsevan tiedoston loppuun teksti dtoverlay=w1-gpio,gpiopin=4. 1-Wire-väylä on käytettävissä järjestelmän uudelleenkäynnistyksen jälkeen. Vielä on varmistettava, että sopivat Linux-ytimen ajonaikaiset w1-gpio-[30] ja w1-therm [20] -modulit ovat ladattu. Moduulit ladataan käyttöön Modprobe-komennolla [21].

Hakemistoon /sys/bus/w1/devices on listattu kaikki kytketyt ja tunnistetut 1-wire-laitteet hakemistoina. Antamalla komennon ls-hakemistossa järjestelmä tulostaa listan kaikista 1-wire laitteiden sarjanumeroista, kuten kuvassa 12 on esitetty.

```
      @       :/sys/bus/w1/devices $ ls
28-3c01d0756084  28-3c01d075867b  28-3c01d075a473  w1_bus_master1
```

Kuva 12. Esimerkkituloste 1-wire-antureista

Siirtymällä cd-komennolla haluttuun hakemistoon voidaan cat q1_slave -komennolla tulostaa anturin lähettämä lämpötilalukema, kuten kuvassa 13 on esitetty. Jälkimmäisellä rivillä lämpötila on ilmaistu asteen tuhannesosina t=-merkin jälkeen. Lämpötila saadaan normaaleiksi celsiusasteiksi kertomalla tuhannella.



Kuva 13. Cat w1_slave-komennon tulostamat anturin lähettämät tiedot.

4.2 Jakokeskus

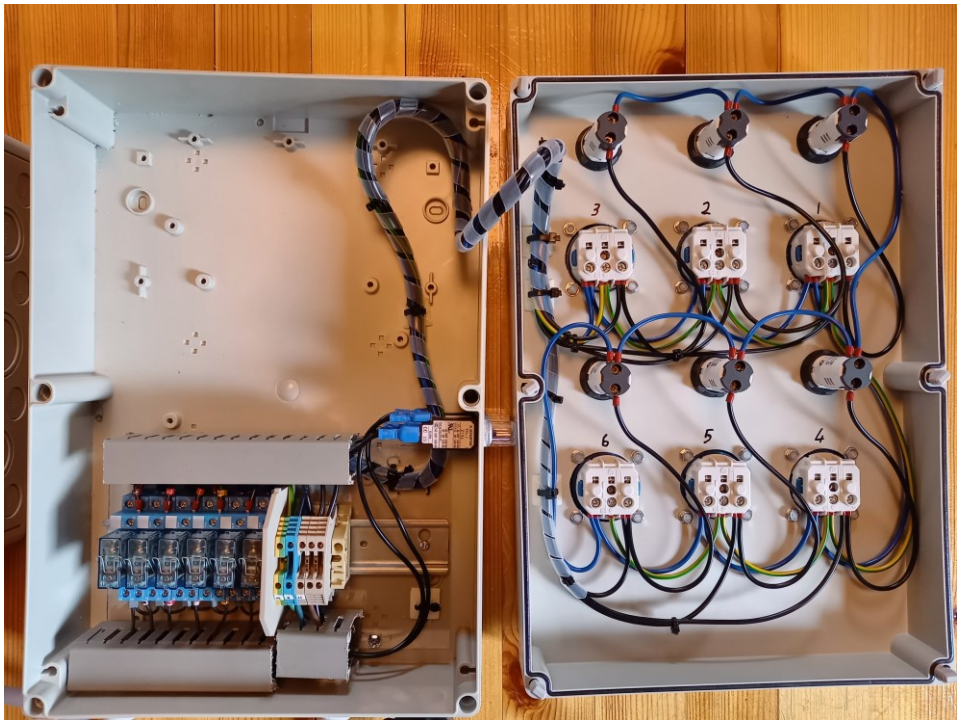
Työssä tarvitsee ohjata pienjännitteisiä lämmittimiä ja valaisimia. Pienjännitteisten laitteiden rakentaminen ei kuulu maallikon oikeuksiin, joten jakokeskus rakennettiin yhteistyössä sähköurakoitsijan kanssa [22]. Yhteistyöstä sähköurakoitsijan kanssa sovittiin, että työhön tehdään opiskelijatyönä tarvittavat suunnitelmat, hankitaan tarvittavat osat ja rakennetaan jakokeskus ilman käyttöönottoa. Sähköurakoitsija tarkastaa suunnitelmat ja asennuksen sekä kiinnittää arvokilven, kun vaatimukset täyttyvät.

Jakokeskuksen suunnittelu aloitettiin kotelon kanteen asennettavien suko-pistorasioiden ja merkkivalojen paikkojen suunnittelulla. Suunnittelu tehtiin CADMATIC-ohjelman Draw-sovelluksella. Seuraavaksi työhön suunniteltiin piirikaavio CADMATIC-ohjelman Piirikaavio-sovelluksella.

Osaluettelo tehtiin kopioimalla CADMATIC-ohjelmasta laitelista Exceliin. Excelissä lisättiin laitelistaan puuttuvat komponentit, kuten vedonpoistajat. CADMATIC:in suunnitelmia ei ollut tehty tukkureiden valikoiman perusteella, joten jokaiselle komponentille etsittiin toimittaja. Osalistan avulla tarvittavat osien hinta- ja saatavuusvertailu oli helppoa. Lisäksi tilaaminen oli osalistan avulla joutuisaa.

Jakokeskuksen rakentaminen oli suoraviivaista suunnitelmien avulla. Kotelon kanteen tehtiin reiät akkuporakoneella kalusteita varten. Koteloon kiinnitettiin taustalevy. Taustalevyyn kiinnitettiin DIN-kisko releitä ja riviliittimiä varten sekä johtokourut. Johdotukset tehtiin 1,5 mm^2:n MKEM-johtimilla. Johtimien päissä käytettiin puristusholkkeja hyvän liitoksen varmistamiseksi. Asennus viimeisteltiin lopuksi nippusiteillä ja johtospiraalilla.

Ennen käyttöönottoa jakokeskus toimitettiin tarkastettavaksi sähköurakoitsijalle. Sähköurakoitsijan hyväksynnän ja arvokilven kiinnittämisen jälkeen jakokeskus voitiin ottaa käyttöön. Kuvassa 14 on esitetty jakokeskus kotelon kansi avattuna.



Kuva 14. Jakokeskus kotelon kansi avattuna

4.3 Ohjelmisto

Opinnäytetyössä käytettiin työtä varten tehtyjen ohjelmien lisäksi muutamia valmiita ohjelmia. Flask-mikrosovelluskehystä käytettiin web-palvelimena ja web-sivujen renderöintiin. Cron-ajastinohjelmaa käytettiin ohjelmien oikea-aikaiseen ajamiseen.

4.3.1 Flask

Flask on Python-pohjainen mikrosovelluskehys web-sovelluksien luomiseen. Se pohjautuu kahteen muuhun sovellukseen. Ensimmäinen niistä on Werkzeug, joka on WSGI-rajapinta, joka ohjaa URL-pyyntöjä olioille. Jinja on kehitetty web-sivujen mallipohjien luontiin. Jinja'a käytetään renderöimään HTML-sivuja, joille Jinja voi välittää muuttujia Python-koodista. Flaskissa on myös muita enemmän taustalla toimivia ohjelmakirjastoja/sovelluksia [23]. Ympäristöön, jossa Pythonin pakettienasennus (PIP) on valmiiksi asennettu, asennettaan Flask PIP:n install-komennolla [24].

Sovellusta varten luodaan kansio, jolle luodaan kaksi alikansiota. Templates-alikansioon sijoitetaan web-sivujen esimerkiksi HTML-koodilla luodut mallipohjat ja static-kansioon sijoitetaan CSS-tyylitiedostot [23].

Reititys toteutetaan tässä opinnäytetyössä Flaskissa route()-dekoraattoria käyttäen. Dekoraattorin avulla yhdistetään tietty URL tiettyyn funktioon. Index()-funktiossa olevan templateData-sanakirjan arvot välitetään Jinjalle web-sivun muuttujiin sijoitettaviksi [23].

Alla esiteltävissä esimerkkikoodeissa 2–4 esitellään esimerkki Python-ohjelmasta, joka reitittää ja renderöi index.html-sivun. Koodiesimerkeissä 4–5 esitellään uusien arvojen lukeminen käyttäjältä. Esimerkit ovat yksinkertaistettuja esimerkkejä työssä käytetyistä toteutuksista. Esimerkkikoodin selainnäkymä on esitetty kuvassa 15 ja ohjelman ajo terminaalissa on esitetty kuvassa 16.

```
from flask import Flask, render_template
from datetime import datetime

app = Flask(__name__)

@app.route("/")
def index():

    time_now = datetime.now()
    time_string = time_now.strftime("%H:%M:%S %d.%m.%Y")

    templateData = {
        "teksti" : "Hello World!",
        "time" : time_string
    }
    return render_template("index.html", **templateData)

if __name__ == '__main__':
    app.run(debug=True, host='0.0.0.0')
```

Esimerkkikoodi 2. Python-ohjelma, joka ohjaa URL-kutsun halutulle web-sivulle.

Edellisen koodiesimerkin alussa tuodaan tarvittavat modulit ja annetaan Flask-konstruktorille ohjelman nimi muuttujana. Tämän jälkeen luodaan route()-dekoraattori, joka kertoo ohjelmalle, millä URL-polulla käynnistetään index()-funktio. Index()-funktiossa templateData-sanakirjalle annetaan kahdelle avaimelle arvoksi merkkijonot, ”Hello World!” ja järjestelmän tämän hetkinen kellonaika. Funktion paluuarvona on render_template-funktio, jonka parametreiksi on annettu web-sivun tiedostonimi ja templateData-sanakirja web-sivun muuttujan arvoiksi. Koodin lopuksi määritellään debug-tila käynnistämään ohjelmaa suoritettavaksi ja lopuksi asetetaan palvelin näkymään verkossa. Yllä olevaa koodia tulee käyttää vain verkossa, jonka kaikkiin käyttäjiin voi luottaa.

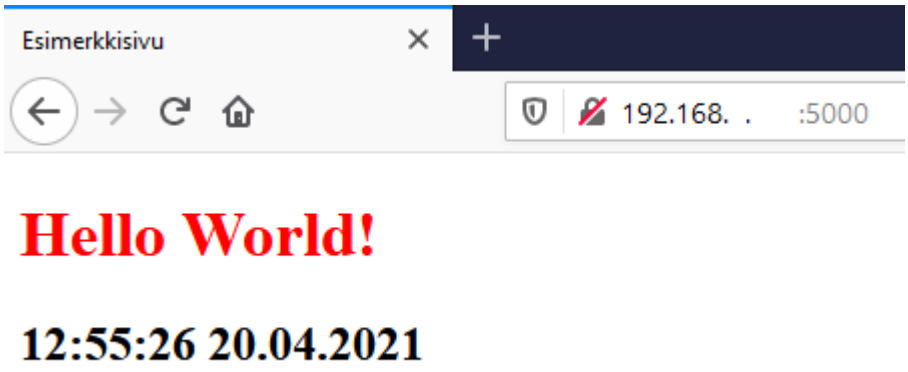
```
<!DOCTYPE html>
<html>

<head>
    <title>Esimerkkisivu</title>
    <link rel="stylesheet" type="text/css" href="{{ url_for('static',file-
name='style.css') }}">
</head>

<body>
    <h1>{{ teksti }}</h1>
    <h2>{{ time }}</h2>
</body>

</html>
```

Esimerkkikoodi 3. HTML-sivu, jonka esimerkkikoodi 2 renderöi.

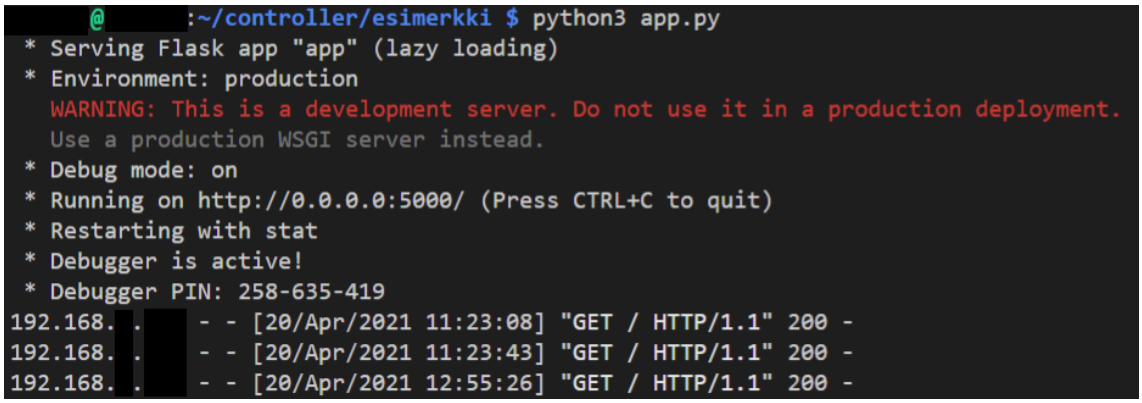
Edellisessä koodiesimerkissä on aaltosulkeissa esitetty kohdat, joihin tuodaan ulkopuolista tietoa. Tässä esimerkissä tuodaan CSS-tyylisivun osoite sekä teksti- ja time-muuttujat templateData-sanakirjasta.

```
h1 {
    color: red;
}
```

Esimerkkikoodi 4. CSS-sivu, joka on linkitetty esimerkkikoodin 3 HTML-sivuun. H1-elementin tekstin väri on määritetty punaiseksi.



Kuva 15. Esimerkin mukainen Flask-sivu



Kuva 16. Esimerkkisivun tuottavan app.py-sovelluksen ajaminen SSH-yhteyden avulla Linuxin komentorivillä.

Edellisessä esimerkissä lähetettiin tietoa palvelimelta käyttäjälle. Myös käyttäjä voi lähettää tietoa palvelimelle [23]. Opinnäytetyössä tätä ominaisuutta käytettiin lämpötila- ja kellonaika-asetusten muuttamiseen.

HTTP POST -metodilla voidaan lähettää tietoa palvelimelle [23]. HTML-sivulle tehtyyn lomakkeeseen syötetään arvo, joka välitetään POST-metodilla palvelimelle. Python-koodi hoitaa käyttäjän arvon tallentamisen muuttujaan. Seuraavassa koodiesimerkissä 5 ja 6 käyttäjä syöttää lomakkeeseen arvon ja arvo tallennetaan Python-ohjelman sanakirjaan. Näkymä selaimessa on havainnollistettu kuvassa 17. Työssä HTML-sivulta lukeva funktio tallentaa arvon myös asetustiedostoon.

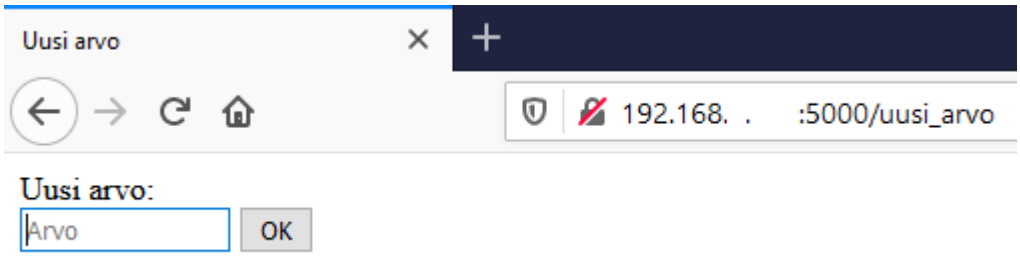
```
@app.route('/uusi_arvo', methods=["POST", "GET"])
def uusi_arvo():
      if request.method == "POST":
            sanakirja["avain"] = request.form["kayttajan_arvo"]
            return redirect(url_for("index"))
      return render_template("uusi_arvo.html")
```

Esimerkkikoodi 5. Ensimmäisellä rivillä annetaan route()-funktiolle metodeiksi POST ja GET.

Edellisessä koodiesimerkissä määritellään uusi_arvo-funktio, joka palautusarvona on uusi_arvo.html-sivun renderöinti. Jos uusi_arvo.html sivulta lähetetään POST-metodilla palvelimelle tietoa, sanakirjaan tallennetaan käyttäjän antama arvo ja käyttäjä palautetaan aloitussivulle.

```
<form action="#" method="post">
      <label>Uusi arvo:</label></br>
      <td><input type="text" placeholder="Arvo" name="kayttajan_antama_arvo"
      size="13"> </td>
      <td><input type="submit" value="OK"></td>
</form>
```

Esimerkkikoodi 6. Esimerkkikoodissa luodaan HTML-lomake, jossa yksi kenttä – käyttäjän syöttämä arvo. Arvo lähetetään POST-metodilla, kun OK-nappia painetaan.



Kuva 17. Esimerkin mukainen sivu uuden arvon lähettämiseksi palvelimelle.

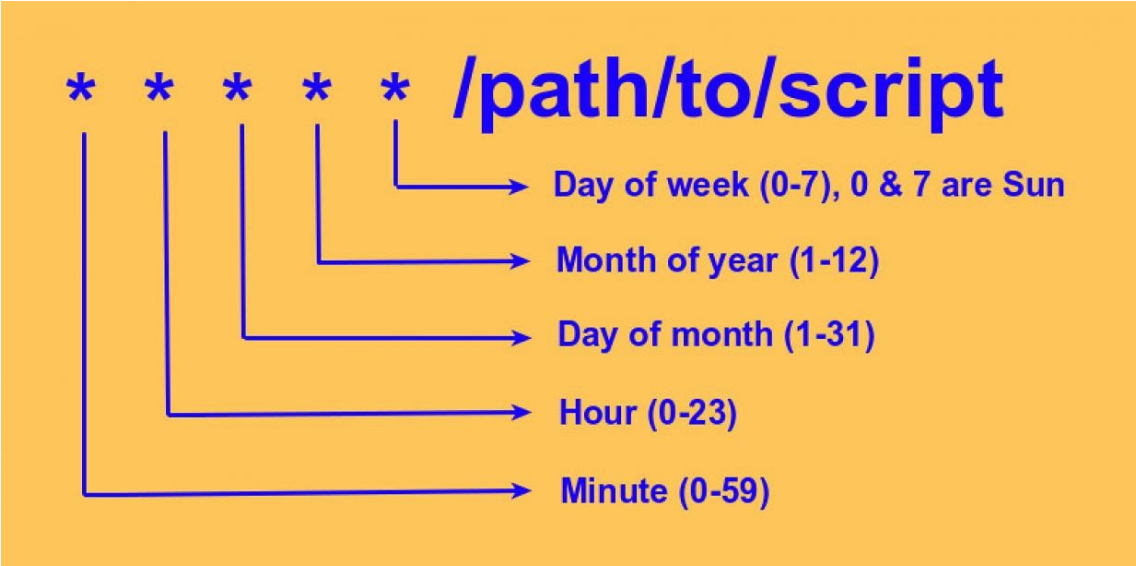
4.3.2 Cron

Cron-ajastuspalvelua käytettiin insinöörityössä ajastamaan termostaatin ja valaistuksen ohjauksen sekä lämpötilalokin kirjoittamiseen. Cronia päädyttiin käyttämään työssä, jotta työtä varten tehdyn ohjelmien ajastuksesta ei tarvitse huolehtia. Cronia käytettäessä ohjelma suoritetaan aina tarvittaessa. Tätä ominaisuutta käytettiin, koska ohjelman jumiutuminen esimerkiksi muistiylivuodon seurauksena on epätodennäköisempää. [25.]

Cron etsii määritellyistä kansioistan crontab-tiedostoja ja tarkastaa jokaisen tehtävän sen selvittämiseksi, tarvitseeko työ suorittaa. Kun tehtävä on suoritettu, lähetetään käyttäjälle siitä sähköpostiin tieto tai vaihtoehtoisesti tieto voidaan kirjoittaa järjestelmän lokiin. [26.]

Crontab-tiedosto sisältää ohjeet Cronille muodossa ”suorita tehtävä tähän aikaan tänä päivänä”. Tehtävän suoritusaika voidaan määrittää minuuttien tai tuntien mukaan sekä päivämäärän ja kuukauden mukaan. Myös viikonpäivän ajastus on mahdollista. Tehtävä on myös mahdollista suorittaa uudelleenkäynnistyksen yhteydessä. [27.]

Kuvassa 18 on havainnollistava kuva crontab-tiedoston asetusrivin muodostamisesta. Asteriski tarkoittaa kaikkia mahdollisia arvoja, joten sijoittamalla ainoastaan asteriskejä aikamuuttujiin suoritetaan haluttu skripti joka minuutti. Aikamuuttuja voi sisältää myös useita arvoja pilkuilla eroteltuna tai arvoalueen väliviivalla erotettuna. On myös mahdollista suorittaa jokin toimenpide viidentoista minuutin välein asteriskin ja kenoviivan yhdistelmällä (/*15). Esimerkkikoodi seitsemässä suoritetaan Python-ohjelma 15 minuutin välein. [27.]



Kuva 18. Crontab-asetusrivi muodostuu aikamuuttujista ja polusta suoritettavaan skriptiin. [28]

Python-ohjelmaa suoritettaessa joudutaan Cronille kertomaan, mistä löytyy ajettava tiedosto ja mistä löytyy Python-tulkki. Esimerkkikoodissa 7 ajastuksen määrittämisen jälkeen siirrytään ensin cd-komennolla hakemistoon, jossa suoritettava tiedosto sijaitsee. &&-operaattoria käytetään seuraavan komennon suorittamiseen, jos ensimmäinen komento on suoritettu onnistuneesti. Tässä tapauksessa cd-komennolla hakemistoon siirtymisen pitää olla suoritettu onnistuneesti, jotta seuraava komento voidaan suorittaa. &&-operaattoria seuraava komento käynnistää Python 3 -tulkin ja suorittaa sillä logger.py-ohjelman. [27, 29.]

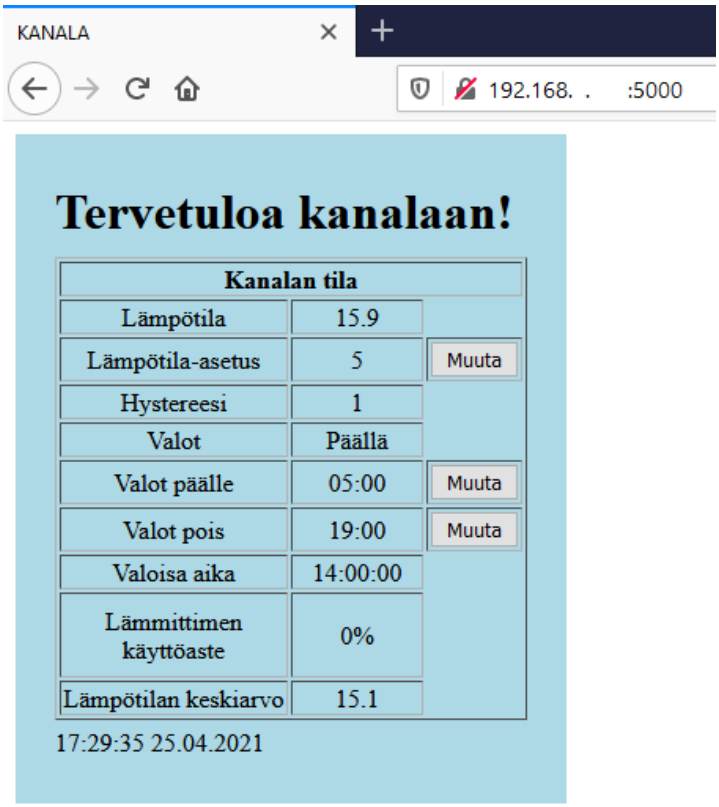
```
*/15 *  *   *   *     cd /home/user/controller/ && /usr/bin/python3 logger.py
```

Esimerkkikoodi 7. Crontab-asetustiedoston rivi ohjelman suorittamiseksi viidentoista minuutin välein.

4.4 Käyttöliittymä

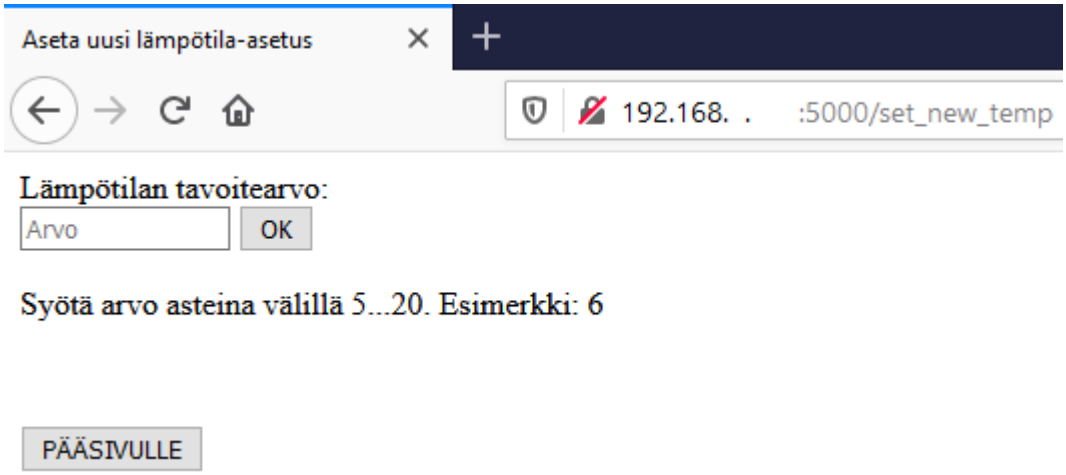
Valaistukseen ja lämmitykseen on tarve tehdä muutoksia aika-ajoin sekä käyttäjän on tärkeää saada tietää, mikä on kanalan lämpötila tai loppuuko lämmittimen kapasiteetti kesken. Lisäksi kanalan hoitajalla on helppo tarkastaa valoisan ajan pituus ja mahdollinen virheellinen asetus on helpommin havaittavissa. Tätä tarvetta varten tehtiin HTML5-

pohjainen web-selaimessa toimiva käyttöliittymä. Käyttöliittymän ulkoasu on esitetty kuvassa 19.



Kuva 19. Kanalan asetussivu, jossa näytetään käyttäjälle nykyiset asetukset ja kanalan tilatietoja.

Kuvassa 20 on kuvakaappaus lämpötila-arvon muuttamisesta. Käyttöliittymässä on mahdollista muuttaa haluttua arvoa painamalla ”muuta”-painiketta. Mikäli käyttäjä huomaa olevansa väärällä asetussivulla, on mahdollista palata pääsivulle. Käyttäjä ohjataan uudelle sivulle, jossa kerrotaan, mitä arvoa voi muuttaa ja mitkä ovat mahdollisia arvoja. Käyttäjän syötettyä uuden arvon hänet palautetaan etusivulle. Uusi arvo päivittyy välittömästi, mikäli arvo mahdollinen ja ohjeiden mukainen.



Kuva 20. Uuden lämpötilan tavoitearvon syöttäminen selaimessa

5 Yhteenveto

Työssä valmistui kaksi laitetta: ohjain ja jakokeskus. Järjestelmä ohjaa lämmitystä ja valaistusta sekä mittaa lämpötilaa. Järjestelmä on säädettävissä etäyhteydellä ja laajennusvaraa jäi runsaasti.

Valmistuttuaan projekti vastasi asetettuja tavoitteita. Laite sopii ulkoisesti ja teknisesti kanalaolosuhteisiin ja sen asentaminen käyttöpaikalle oli helppoa. Kirkkaat pistorasian tilan osoittavat LED-valot jouduttiin osin peittämään 3D-tulostetuilla kansilla valomäärän vähentämiseksi. Kanalan hoitaja onnistui järjestelmän avulla säätämään ja tarkastamaan kanalan tilasta kertovia tietoja haluamallaan tavalla.

Valaistuksen ohjaus toimii tarkasti ja halutunlaisesti. Lämmityksen ohjausta ei ollut mahdollista testata lämpimän vuodenajan vuoksi, joten lämmityksen ohjaus voidaan vasta kylmänä vuodenaikana. Lämmityksen ohjaus toimi halutulla tavalla lämpöanturia käsin lämmitettäessä. Tarkemman ohjauksen avulla eläinten hyvinvointi paranee ja energiankulutusta pystytään pienentämään. Sähkökatko simuloitiin testauksessa katkaisemalla sähköt. Järjestelmä palautui sähkökatkosta automaattisesti. Automaatiojärjestelmä jäi heti testauksen jälkeen tuotantokäyttöön.

Valmiita kaupallisia tuotteita, jotka täyttäisivät olosuhteiden vaatimukset, ei ole onnistuttu löytämään. Valmiista kiinteistöautomaatiojärjestelmistä olisi pystynyt rakentamaan perustoiminnot suorittavan kokonaisuuden. Valmiin järjestelmän hinta olisi kuitenkin ollut huomattavasti korkeampi ja vapautta vaikuttaa etähallintaan ja kytkettäviin lisäosiin oltaisiin menetetty. Kaupallinen ratkaisu olisi parempi vaihtoehto, kuin tässä työssä rakennettu, jos järjestelmää joutuisi myöhemmin ohjelmoimaan ulkopuolinen henkilö.

Seuraaviksi kehitysaskeleiksi jää usean anturin luku ja lämmittimien ohjaus. Tällä uudella ominaisuudella olisi mahdollista ohjata kanan poikas- tai nuorikkovaiheen lämmittimiä tai keskittämään lämmitystä orsille tai lattialle. Valojen ohjaukseen on mahdollista tuoda toiminnot valaistuksen asteittaiseksi lyhentämiseksi sulkasatoa varten ja valaistuksen asteittainen palautus muninnan aloittamista varten.

Tulevaisuudessa vastaavaan automaatiojärjestelmään kehitettäviä ominaisuuksia voisi olla esimerkiksi kanojen tunnistaminen RFID-tunnisteen avulla. RFID-tunnisteen luku

munintapesissä mahdollistaisi tiedon keräämisen kanan muninnasta. Rehu-, vesi- ja kalkkiautomaattien painon seuraamisella voi valvoa, että vettä ja ravintoa on aina saatavilla esimerkiksi juoma-automaatin vuototilanteessa. Lisäksi alla on luetteloitu kanalassa mitattavia kaasuja, joita voi käyttää ilmanlaadun valvomiseen ja ilmanvaihdon ohjaamiseen.

- happi
- hiilidioksidi
- ammoniakki
- rikkivety
- suhteellinen kosteus

Kanalassa on myös hyvin paljon mahdollisuuksia muidenkin toimintojen automatisointiin. Joidenkin toimintojen automatisoinnin mielekkyys on kuitenkin kyseenalainen, koska ihmisen tekemien hoitotoimenpiteiden yhteydessä tulee myös tarkasteltua eläinten hyvinvointia ja esimerkiksi rakenteiden toimivuutta.

Rakennettua järjestelmää on mahdollista käyttää pienin muokkauksin moniin erilaisiin käyttötarkoituksiin, joissa halutaan ohjata lämpötilan tai muiden arvojen pohjalta pienois- tai pienjännitteellä toimivia laitteita. Kaupallinen käyttö voisi olla mahdollista tuotteistamisen avulla, mutta tuotteistaminen yksinkertaiseksi kuluttajatuotteeksi muuttaisi järjestelmän luonnetta. Sopiva välimalli voisi löytyä mahdollisuudella graafiseen ohjelmointiin, kuten Scratch. Tuotteistamisessa tietoturvan ottaminen huomioon on tärkeää.

Lähteet

1 Kanakuulutus: alkuperäinen maatiaiskana tarvitsee lisää kasvattajia. 2021. Verkkoaineisto. Luke. <https://www.luke.fi/uutinen/kanakuulutus-alkuperainen-maatiaiskana-tarvitsee-lisaa-kasvattajia/>. Luettu 20.4.2021.

2 Ojanne, Tarja. 2018. Opas pienkanalan hoitajalle. 4. painos. Jokioinen: Suomen Siipikarjaliitto.

3 Bestman, Monique; Ruis, Marko; Heijmans, Jos & van Middelkoop, Koos. 2012. Kanahavaintoja: käytännönläheinen opas siipikarjanpitoon, jossa lintu pääosassa. Forssa: Suomen Siipikarjaliitto.

4 Mobile HTML5. 2015. Verkkoaineisto. Mobile HTML5 <https://mobilehtml5.org/>. Luettu 20.4.2021.

5 What is Raspberry Pi. Verkkoaineisto. Raspberry Pi Foundation <https://www.raspberrypi.org/help/what-%20is-a-raspberry-pi/>. Luettu 20.4.2021.

6 The Raspberry pi in scientific research. Verkkoaineisto. Raspberry Pi Foundation <https://www.raspberrypi.org/blog/the-raspberry-pi-in-scientific-research/>. Luettu 20.4.2021.

7 Ten millionth raspberry pi new kit. Verkkoaineisto. Raspberry Pi Foundation. <https://www.raspberrypi.org/blog/ten-millionth-raspberry-pi-new-kit/>. Luettu 20.4.2021.

8 Revolution Pi. Verkkoaineisto. KUNBUS GmbH <https://revolution.kunbus.com/revolution-pi-series/>. Luettu 20.4.2021.

9 Raspberry Pi OS. Verkkoaineisto. Raspberry Pi Foundation <https://www.raspberrypi.org/software/>. Luettu 20.4.2021.

10 GPIO. Verkkoaineisto. Raspberry Pi Foundation <https://www.raspberrypi.org/documentation/usage/gpio/README.md>. Luettu 20.4.2021.

11 Flyback diode. Verkkoaineisto. Wikipedia <https://en.wikipedia.org/wiki/Flyback_diode>. Luettu 5.4.2021.

12 RPI Relayboard Schematics. Verkkoaineisto. Waveshare <https://www.waveshare.com/w/upload/d/db/RPi_Relay_Board_%28B%29_Schematic.pdf>. Luettu 5.4.2021.

13 Serie 693. Verkkoaineisto. Binder Steckverbinder <https://www.elfadis-trelec.fi/Web/Downloads/_d/-e/xg693_data_d-e.pdf >. Luettu 5.4.2021.

14 SP13 series. Verkkoaineisto. Weipu <https://weipuconnector.com/product/37>. Luettu 5.4.2021.

15 Pääteholkit. Verkkoaineisto. Phoenix Contact <https://www.phoenixcontact.com/online/portal/fi?1dmy&urile=wcm:path:/fifi/web/main/products/subcategory_pages/Tools_P-25/8180ec6c-b52b-4694-a5a6-ec0f28558052/8180ec6c-b52b-4694-a5a6-ec0f28558052>. Luettu 5.4.2021.

16 Limor Fried, 2012. Adding a Real Time Clock to Raspberry Pi. Verkkoaineisto. Adaruit < https://learn.adafruit.com/adding-a-real-time-clock-to-raspberry-pi?view=all>. 31.8.2012. Luettu 5.4.2021.

17 Frodo Looijaard, Mark D. Studebaker, Jean Delvare. i2cdetect(8) - Linux man page. Verkkoaineisto. die.net<https://linux.die.net/man/8/i2cdetect>. Luettu 5.4.2021.

18 Bryan Henderson. 1996. hwclock(8) - Linux man page Verkkoaineisto. die.net < https://linux.die.net/man/8/hwclock>. Luettu 5.4.2021.

19 DS18B20 datasheet1. 2019. DS18B20 datasheet1. Maxim integrated Rev. 6, 7/2019.

20 Evgeniy Polyakov. Kernel driver w1_therm. Verkkoaineisto. Linux Kernel Organization < https://www.kernel.org/doc/html/latest/w1/slaves/w1_therm.html> Luettu 5.4.2021.

21 Rusty Russel. 2002. Verkkoaineisto. IBM Corporation <https://linux.die.net/man/8/modprobe>. Luettu 5.4.2021.

22 Sähköpätevyydet ja työalueet. Verkkoaineisto. Tukes <https://tukes.fi/sahko/sahkotyot-ja-urakointi/sahkopatevyydet-ja-tyoalueet#4831468c>. Luettu 20.4.2021.

23 Quickstart. Verkkoaineisto. Flask <https://flask.palletsprojects.com/en/1.1.x/quickstart/>. Luettu 20.4.2021.

24 Installation. Verkkoaineisto. Flask <https://flask.palletsprojects.com/en/1.1.x/installation/>. Luettu 20.4.2021.

25 Mohamed Said. 2021. Avoiding Memory Leaks When Running Laravel Queue Workers. Verkkoaineisto. Diving Lavarel<https://divinglaravel.com/avoiding-memory-leaks-when-running-laravel-queue-workers>. Luettu 20.4.2021.

26 Paul Vixia, Marcela Mašláňová, Colin Dean, Tomáš Mráz, 2013 cron(8) — Linux manual page Verkkoaineisto. <https://man7.org/linux/man-pages/man8/cron.8.html>. Luettu 20.4.2021.

27 Paul Vixie. 2012. crontab(5) — Linux manual page. Verkkoaineisto. Man7.org <https://man7.org/linux/man-pages/man5/crontab.5.html>. Luettu 20.4.2021.

28 Ken Hess. 2019. Automate your Linux system tasks with cron. Verkkoaineisto. Red Hat<https://www.redhat.com/sysadmin/automate-linux-tasks-cron>. Luettu 20.4.2021.

29 BASH manual - List of Commands. Verkkoaineisto. Free software Doundation <https://www.gnu.org/software/bash/manual/html_node/Lists.html>. Luettu 1.5.2021.

30 Ville Syrjälä. Kernel driver w1-gpio. Verkkoaineisto. <https://www.kernel.org/doc/html/latest/w1/masters/w1-gpio.html>. Luettu 1.5.2021.