OULUN AMMATTIKORKEAKOULU

Jari Tuliniemi

TUULETINTESTAUSLAITTEISTO

TUULETINTESTAUSLAITTEISTO

Jari Tuliniemi
Opinnäytetyö
Syksy 2018
Automaatiotekniikan tutkinto-ohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Automaatiotekniikan koulutusohjelma

Tekijä: Jari Tuliniemi
Opinnäytetyön nimi suomeksi: Tuuletintestauslaitteisto
Työn ohjaajat: Timo Heikkinen (OAMK), Erkki Lehtonen (Nokia)
Työn valmistumislukukausi ja -vuosi: Syksy 2018
Sivumäärä: 40 + 1 liitettä

Tässä työssä suunniteltiin ja toteutettiin testauslaitteisto laitetuulettimille. Työ tehtiin Nokia Oyj:n lämpötestaus-tiimille. Työn tarkoituksena oli saada lämpötestaus-tiimille testauslaitteisto, jolla voitiin suorittaa laadunvarmistus täysin uusille tuuletintyypeille sekä uusille versioille jo käytössä olevista tuuletintyypeistä. Testauslaitteiston pääasiallinen tarkoitus oli tehdä On / Off -testausta, joka on testeistä pitkäkestoisin.

Opinnäytetyö jakautuu testauslaitteiston vaatimusten määritykseen, toimintaperiaatteen ja komponenttien valintaan, piirilevyn suunnitteluun ja valmistukseen ja testiohjelmiston ohjelmointiin. Vaatimusten määrityksessä käydään läpi testattavien laitetuulettimien ominaisuuksien sekä tehtävien testien muodostamat vaatimukset testauslaitteistolle. Toimintaperiaatteen ja komponenttien valinnan osiossa kerrotaan testauslaitteistoon valituista komponenteista ja perustellaan niiden valinnat. Piirilevyn suunnittelu tehdään Kicad-ohjelmistolla. Ohjelmistolla tehdään testauslaitteistolle piirikaavio ja piirilevysuunnitelma, jonka perusteella valmistetaan piirilevy ja testauslaitteisto. Ohjelmoinnin osiossa käydään läpi testiohjelmiston ohjelmointi Python-ohjelmointikielellä sekä käyttöliittymän teko Javascript-ohjelmointikielellä ja HTML-kuvauskielellä.

Lopputuloksena on toimiva tuuletintestauslaitteisto. Valmis testauslaitteisto vastaa alussa määritettyjä vaatimuksia ja sillä pystytään tekemään vaadittua On / Off -testausta.

Asiasanat: tuulettimet, suunnittelu, piirilevyt, ohjelmointi, käyttöliittymä

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Automation Engineering

Author: Jari Tuliniemi
Title of thesis: Design and Implementation of Fan Testing Equipment
Supervisors: Timo Heikkinen (OAMK), Erkki Lehtonen (Nokia)
Term and year when the thesis was submitted: Fall 2018
Pages: 40 + 1 appendices

The objective of the bachelor's thesis is to design and implement a piece of testing equipment for device fans. The thesis was commissioned by Nokia Corporation. The aim was to produce testing equipment which would make it possible for the thermal testing team to perform a quality assurance testing for the fans. The main purpose of the testing equipment was to run On / Off testing.

The thesis consists specifications of the testing equipment, its operating principles, the selection of the components, the design and the production of the printed circuit board and programming of the testing equipment. The design of the printed circuit board was made with Kicad software. The testing equipment was programmed with Python programming language and the user interface with Javascript and HTML.

In conclusion, the testing equipment worked and fulfilled the set specification. The fan testing equipment is currently in use for On / Off testing.

Keywords: fans, design, printed circuit boards, programming, user interface

ALKULAUSE

Haluan kiittää Nokia Oyj:tä kiinnostavasta opinnäytetyön aiheesta. Työ tarjosi monipuolisia haasteita ja mahdollisuuden oppia uusia asioita.

Erityiskiitokset Nokia Oyj:n puolella työtä ohjanneelle Erkki Lehtoselle, sekä lämpötestaus-tiimin Jarno Rautiolle ja Ari Vierimaalle. Lisäksi vielä kiitokset Juha Vuormalle, Marko Hiltuselle, Jarno Mettovaaralle ja Kari Nylundille lämpimästä työilmapiiristä ja avusta. Kiitos myös Oulun ammattikorkeakoulun Timo Heikkiselle opinnäytetyön ohjaamisesta.

Oulussa 16.12.2018

Jari Tuliniemi

SISÄLLYS

SANASTO

Arduino-ohjelmointiympäristö	C++-ohjelmointikieleen perustuva kokoelma apukirjastoja mikrokontrollereiden ohjelmointiin .
Avokollektorilähtö	Avolähtö, jossa transistorin kollektori toimii lähtönä
Back end	Palvelimessa suoritettava ohjelmointi
Flask	Mikrosovelluskehys Python-ohjelmointikielelle
Front end	Käyttäjälle näkyvä ohjelmoinnin osa
Gerber-tiedosto	Ohjeisto, jonka avulla piirilevy valmistetaan
GPIO	General Purpose Input/Output; yleiskäyttöinen nasta elektroniikassa
HTML	Verkkosivujen kehittämiseen tarkoitettu merkintäkieli
I2C	Sarjamuotoinen kaksisuuntainen ohjaus- ja tiedonsiirtoväylä
Javascript	Dynaamisesti tyypitetty ohjelmointikieli
jQuery	JavaScript-apukirjasto dynaamisten sivustojen tekemiseen
Kicad	Piirilevysuunniteluun tarkoitettu avoimen lähdekoodin ohjelma
Kirjasto	Kokoelma luokkia ja ohjelmia

Ohjelmistokehys	Ohjelmistotuote, jonka päälle tietokoneohjelma voidaan rakentaa
PWM	Pulssinleveysmodulaatio
Python	Tulkattava dynaamisesti tyypitetty ohjelmointikieli
Raspberry Pi	Raspberry Pi Foundationin kehittämä yhden piirilevyn tietokone
Reflow	Pintaliitoskomponenttien juotosmentelmä
Shunttivastus	Virran mittauksessa käytetty matalaresistanssinen vastus
Socket	Rajapinta tiedon lähettämiseen ja vastaanottamiseen
SQL	Kyselykieli tiedonhakuun relaatiotietokannoista

1 JOHDANTO

Opinnäytetyön tarkoituksena oli suunnitella ja toteuttaa laitetuulettimien testauslaitteisto. Työn tilaaja oli Nokia Oyj. Työn lähtökohtana oli lämpötestaus-tiimin tarve tuuletintestauslaitteistolle, jolla voitiin suorittaa laadunvarmistus täysin uusille tuuletintyypeille sekä uusille versioille jo käytössä olevista tuuletintyypeistä. Lämpötestaus-tiimin tekemät tuuletintestit olivat usein pitkäkestoisia ja niiden mittaustulokset talletettiin usein käsin. Testauslaitteiston tavoitteena oli automatisoida testi ja siten lyhentää testin alkujärjestelyihin käytettyä aikaa sekä lisätä mittausdatan tarkkuutta ja määrää. Alkuvaiheessa laitteistoa oli tarkoitus käyttää pääasiassa On / Off -testaukseen, joka on testeistä pitkäkestoisin.

Työn suoritus on jaettu neljään osaan. Ensimmäisessä osassa kuvataan testauslaitteiston vaatimukset yhdistämällä lämpötestaus-tiimin asettamat vaatimukset sekä analysoimalla testattavien tuulettimien valmistusspesifikaatiot. Toisessa osassa suunnitellaan testauslaitteiston toimintaperiaate ja valitaan testauslaitteiston vaatimukset täyttävät komponentit. Kolmannessa osassa suunnitellaan testauslaitteiston piirilevy sekä rakennetaan testauslaitteisto. Viimeisessä osassa ohjelmoidaan testauslaitteiston testiohjelmisto ja käyttöliittymä.

2 TESTAUSLAITTEISTON VAATIMUSTEN MÄÄRITYS

Laitteiston vaatimusten määritys aloitettiin aloituspalaverissa. Palaverissa lämpötestaus-tiimin jäsenet listasivat testauslaitteistolta vaadittavat ominaisuudet. Vaatimuksista tehtiin yhteenveto, joka käytiin läpi yhdessä lämpötestaus-tiimin kanssa ja siihen lisättiin puuttuvat ominaisuudet.

Laitteiston tuli kyetä testaamaan sekä 12 voltin että 48 voltin tuulettimia. Tuulettimia piti pystyä testaamaan 16 kappaletta samanaikaisesti. Laitteella tuli voida testata samanaikaisesti eri käyttöjännitteellä toimivia tuulettimia. Jokaisesta tuulettimesta oli pystyttävä mittaamaan virrankulutus ja tuulettimen nopeus. Virran mittauksen mittaustarkkuuden ei tarvinnut olla ehdottoman tarkka, mutta mittausten toistuvuuden tuli olla korkea, jolloin mahdolliset poikkeamat näkyisivät mittausdatassa. Tuulettimia piti pystyä ohjaamaan yksilöllisesti ja tarvittaessa piti pystyä määrittämään erikseen kunkin tuulettimen toiminta. Tuulettimien liittäminen laitteeseen tuli tapahtua pikaliittimillä, koska testattavissa tuulettimissa käytettiin erilaisia liittimiä. Tuulettimien asettamat vaatimukset määritettiin käyttämällä tuuletinkohtaisia spesifikaatioita.

Laitteella tuli voida mitata ulkoista lämpötilaa, koska tuulettimia testataan lämpökaapissa ja näin nähtäisiin mahdollinen lämpösyklin vaikutus tuulettimien käyttäytymiseen. Lämpötilan mittauksen tuli tapahtua käyttämällä K-tyypin lämpöparia, koska lämpötestaus-tiimi käytti niitä jo mittauksissa.

Laitteen tuli toimia itsenäisesti ilman toista tietokonetta ja sen tuli olla helposti siirrettävissä, koska lämpökaappeja ja testauslaboratorioita oli eri puolilla Nokian työtiloja. Lämpökaappien käyttö valikoitui sen mukaan, mikä kulloinkin oli vapaana. Tästä johtuen laitteen ohjauksen oli oltava mahdollista sisäverkosta käsin ja testausdatan luettavissa etänä.

Käyttöliittymästä toivottiin internet-selaimella toimivaa, jotta testauslaitteiston käyttöön ei tarvitsisi asentaa erillisiä ohjelmia. Testauksen määritys, käynnistys ja keskeytys piti pystyä tekemään käyttöliittymästä. Siitä oli välittömästi nähtävä, onko laite käynnissä sekä laitteen senhetkinen mittausdata. Mittausdatan tuli olla

helposti ladattavissa Exceliin ja mikäli mahdollista, tuli mittausdatasta saada kuvaajat valmiina.

Tulevaisuudessa testauslaitteiston tuli olla laajennettavissa erilaisiin testeihin mutta alkuvaiheessa sillä piti pystyä tekemään On / Off -testausta. Laitteita tuli tehdä aluksi kaksi kappaletta ja tarpeen mukaan lisää.

2.1 Tuulettimet

Tuulettimilla on monta valmistajaa ja niiden toiminnalle on määritetty spesifikaatio. Spesifikaatiossa määritetään tuulettimien ominaisuudet ja vaatimukset, jotka niiden tulee täyttää. Siinä on määritetty esimerkiksi tuulettimien meluominaisuudet, IP-luokitukset ja fyysiset mitat. Testauslaitteiston vaatimusten määrityksessä tarvittiin seuraavia spesifikaatiosta löytyviä vaatimuksia: käyttöjännite, virrankulutus, nopeudensäätö ja kierroslukumittaus. Testauslaitteistolla testataan sekä 12 voltin että 48 voltin tuulettimia, jolloin sen tulee täyttää kummankin tuuletintyypin asettamat vaatimukset.

Molemmat tuuletintyypit on määritetty nelijohtimisiksi. Johtimet on määritetty seuraavasti: syöttöjännite, ohjaussignaali, kierroslukusignaali ja maadoitus. On olemassa myös kolmijohtimisia tuulettimia, joissa kierroslukusignaali ja ohjaussignaali kulkevat samassa johtimessa. Testausta vaativissa laitteissa niitä ei ole käytössä, minkä johdosta testauslaitteisto voidaan suunnitella toimivaksi ainoastaan nelijohtimisten tuulettimien kanssa. (1; 2.)

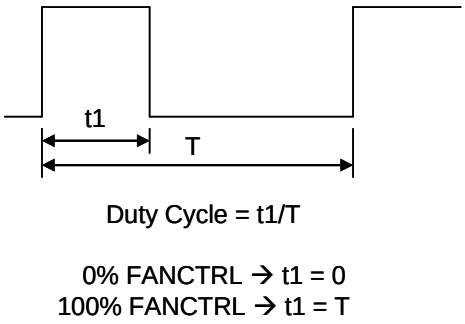
Tuulettimille on määritetty syöttöjännitteen toimintaväli (esimerkiksi 12 voltin tuulettimelle 10 - 16 VDC), jolla tuulettimen tulee toimia. Spesifikaatiossa on kuitenkin sallittu, että tuulettimien kierrosluku saa vaihdella, jos syöttöjännite muuttuu. Testauslaitteiston tulee tällöin toimia sekä 12 voltin tuulettimen minimi- että 48 voltin tuulettimen maksimisyöttöjännitteellä. Siten testauslaitteiston syöttöjännitteen toimintaväliksi muodostuu 10 - 55 VDC. (1; 2.)

Virrankulutuksen määrityksistä ilmenee, että 12 voltin tuulettimet asettavat testauslaitteiston toiminnalle suuremmat vaatimukset kuin 48 voltin tuulettimet, jolloin testauslaitteisto voidaan tältä osin suunnitella 12 voltin tuulettimien spesifi-

kaation mukaan. 12 voltin tuulettimien virrankulutukseksi niiden pyöriessä huippunopeudella ja normaalijännitteellä on määritetty maksimissaan 2,4 ampeeria. Tuulettimien käynnistyessä virrankulutuksen sallitaan nousevan suuremmaksi. Hetkellinen virrankulutus ei saa milloinkaan nousta suuremmaksi kuin 5,0 ampeeria. Suurimmat sallitut virtapiikit ovat 4,5 ampeeria 1,0 millisekunnin ja 4,5-5,0 ampeeria 50 mikrosekunnin ajaksi. (1; 2.)

2.1.1 Ohjaus

Tuulettimien ohjaus tapahtuu PWM-signaalilla eli pulssinleveysmodulaation avulla. Pulssinleveysmodulaatio tarkoittaa jännitteen säätämistä pulssisuhdetta (Duty cycle) muuttamalla. Pulssisuhde on pulssinleveyden ja jaksonajan suhde ja sitä ilmaistaan prosentteina. (3.) Kuvassa 1 näkyy tuulettimien ohjaussignaalin pulssisuhde.



KUVA 1. Ohjaussignaali (1)

Nelijohtimisessa tuulettimessa pulssinleveysmodulaatio katkaisee ja kytkee päälle tuulettimen sähkömoottorin kelalle menevää jännitettä. Pulssinleveysmodulaatio ei vaikuta tuulettimen kierroslukumittaukseen, siitä johtuen kierroslukumittaus toimii tuulettimen hidastuessa pysähdystilaan. Nelijohtimisen tuulettimen etu kolmijohtimiseen verrattuna on myös tuulettimen ohjauksen herkkyys, joka mahdollistaa tuulettimen tarkemman ohjauksen. (4.)

Ohjaussignaalin ykköstasoksi spesifikaatio määrittää > 2,0 volttia ja nollatasoksi < 0,8 volttia. Tuuletin pyörii täydellä teholla, kun PWM-signaalin pulssisuhde on alle 5% ja se on pysähdystilassa pulssisuhteen ollessa yli 92%. PWM-signaalin

toimintataajuudeksi on määritetty 1 kHz. Taulukossa 1 näkyy tuulettimen nopeus suhteessa ohjaussignaaliin. (1; 2.)

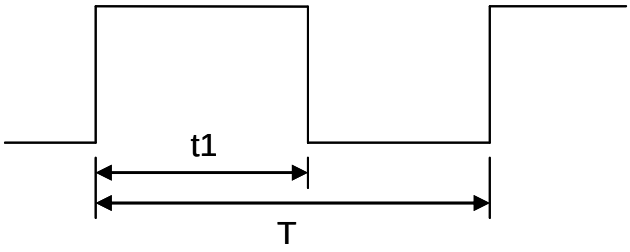
TAULUKKO 1. Pulssimodulaation vaikutus tuulettimen nopeuteen (1)

Fan Speed	PWM Duty Cycle (nominal)	PWM Duty Cycle (allowed range)	Speed Tolerance [% of max speed]
Max speed	0% to 5%	0% to 9%	+5% / -5%
Linear speed range	5% to 83%	1% to 87%	+10% / -10%
Min speed	83% to 92%	79% to 96%	+5% / -5%
Fan stopped	92% to 100%	88% to 100%	n/a

2.1.2 Kierroslukutieto

Tuulettimien kierroslukumittaus toimii Hall-anturin avulla. Anturilta saatava signaali on kanttiaaltoa. Signaalin pulssisuhde määrittää kuinka tuulettimen todellinen kierrosluku lasketaan. (5.)

Kierroslukusignaalin lähtö on määritetty avokollektorilähdöksi, minkä vuoksi testauslaitteistoon tarvitaan ylösvetovastus. Signaalin nollatasoksi on määritetty < 0,5 volttia. Kierroslukusignaalin syklin kesto T on määritetty seuraavasti: T[s] = (30 / RPM), jossa RPM ilmaisee tuulettimen pyörähdystä minuutissa. Pulssisuhde on määritetty 50 % ± 5 % eli tällöin kaksi pulssia vastaa aina yhtä tuulettimen pyörähdystä. Signaalin taajuusalue on 0 - 500 Hz. Siten tuulettimen korkeimmaksi mitattavaksi kierrosluvuksi saadaan 15000 kierrosta minuutissa. (1; 2.) Kuvassa 2 näkyy 12 voltin tuulettimen kierroslukusignaalin pulssisuhde.



Duty Cycle = t1/T = 0.5 ± 0.05

KUVA 2. Kierroslukusignaali (1)

2.2 Tietojen tallennus ja käyttöliittymä

Mittausdatan tallennus ja käyttöliittymän vaatimukset määritettiin tarkemmin lämpötestaustiimin kanssa käydyissä keskusteluissa. Keskusteluissa mitoitettiin laitteiston vaatimukset siten, että ne olisivat tehtävissä sovitussa aikataulussa. Keskustelujen pohjalta tehtiin lista vaatimuksista, jotka pyrittiin toteuttamaan laitteen suunnittelussa.

Laitteen tuli tallentaa mittausdata paikalliseen tietokantaan. Tulevaisuudessa mittausdata tallennetaan mahdollisesti erilliselle palvelimelle mutta kyseinen ominaisuus rajattiin opinnäytetyön ulkopuolelle. Tietokantaan talletetaan testaajan nimi, tuuletinvalmistaja, testisyklin numero, testin aloitusaika sekä testissä olevien tuulettimien käyttöjännite. Jokaisesta tuulettimesta tuli mitata virrankulutus ja kierrosnopeus sekä tallentaa ne tietokantaan siten, että mittaustuloksia on testisyklin eri vaiheista. Lämpötila tuli tallentaa vähintään kerran minuutissa.

Käyttöliittymän tuli toimia nykyaikaisilla internetselaimilla. Käyttöliittymässä tuli olla selkeä käynnistä- ja seis-nappi. Testiasetusten määrityksen tuli olla selkeää ja yksinkertaistettua. Pääruudulla tuli näkyä jokaisen tuulettimen sen hetkinen virrankulutus ja kierrosluku. Jo päättyneistä testeistä tuli voida ladata koko testidata, testidatan viimeiset 400 testaussykliä ja viimeisen 400 testaussyklin kuvaaja.

2.3 On / Off -testaus

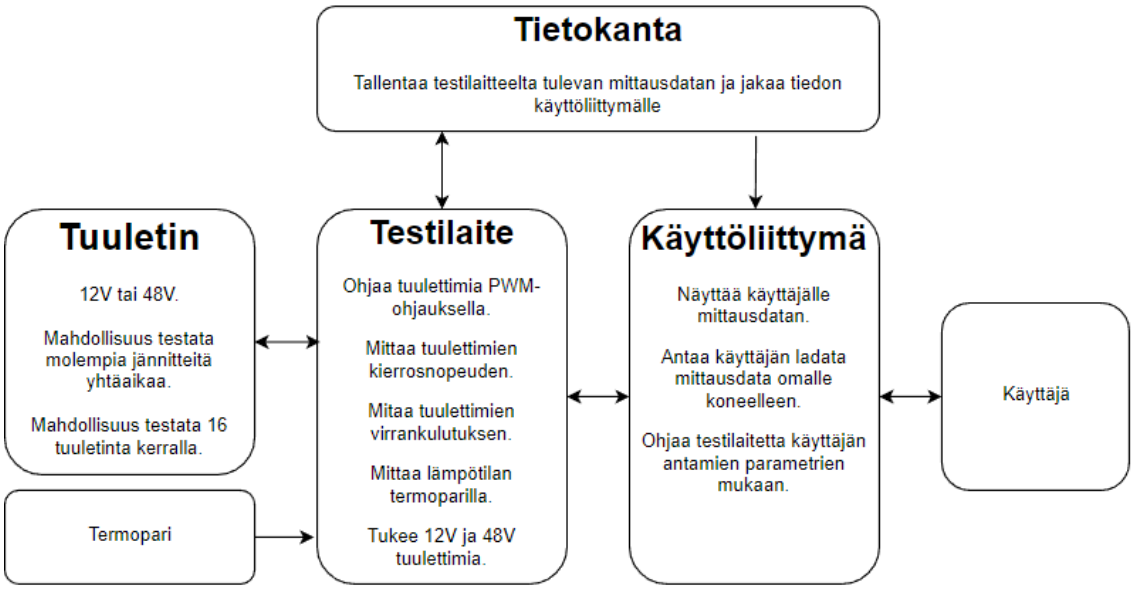
Tuuletinspesifikaatiossa tuuletintyypeille on määritetty joukko tilastollisia testejä, jotka niiden tulee läpäistä (1; 2). Työn tilaajan alkuperäinen vaatimus oli, että testauslaitteistolla voitiin tehdä On / Off -testausta. Tulevaisuudessa testauslaitteistoon voitaisiin mahdollisesti ohjelmoida lisää erilaisia testikonfiguraatioita.

On / Off -testauksessa tuulettimia ajetaan huippunopeuteen ja kytketään pois päältä. Syklin toistokerrat määritetään tuuletinspesifikaatiossa (esimerkiksi 43400 kertaa 12 voltin tuulettimelle). Testaus tapahtuu lämpökaapissa, jossa lämpötila vaihtelee -35 °C:n ja 75 °C:n välillä. Tuulettimista mitataan kierrosnopeus ja virrankulutus tuulettimen ollessa eri tiloissa, kuten kiihdytettäessä, ajettaessa huippunopeudella, hidastettaessa ja tuulettimen ollessa pysähtyneenä. Virrankulutuksen ja kierrosnopeuden tulee pysyä toistuvana testaussyklien välillä. Testaussyklin pituutta tulee voida säätää, koska 12 voltin ja 48 voltin tuulettimet kiihtyvät ja hidastuvat eri tahdissa. (1; 2.)

3 LAITTEISTON SUUNNITTELU JA KOMPONENTTIEN VALINTA

Testauslaitteiston suunnittelu aloitettiin laitteiston vaatimusten määrittämisen jälkeen. Aluksi selvitettiin, olisiko Nokialla jo olemassa valmiita ratkaisuja, joilla testilaite voitaisiin toteuttaa. Pian kävi kuitenkin ilmi, että kaikki vaatimukset täyttävää ratkaisua ei löytynyt. Siten suunnittelu ja toteutus jouduttaisiin tekemään itse.

Laitteen suunnittelu aloitettiin määrittämällä sen arkkitehtuuri vaatimusten mukaiseksi. Vaatimuksena oli, että laite toimii itsenäisesti ilman siihen liitettävää tietokonetta. Tällöin vaihtoehtoja oli kaksi: laitteessa itsessään oli oltava tarpeeksi prosessointitehoa pyörittämään web-palvelinta ja tallentamaan tiedot sisäiseen tietokantaan, tai laiteen oli keskusteltava palvelimen kanssa, johon se lataa mitattavan datan. Laitteen toteutuksessa päädyttiin ensimmäiseen vaihtoehtoon, koska projektin rajallisen ajan takia se oli nopeampi toteuttaa ja koska laitetta voitaisiin tulevaisuudessa laajentaa keskustelemaan palvelimen kanssa. Suunnitelmasta tehtiin kuvassa 3 näkyvä vuokaavio.



KUVA 3. Vuokaavio testauslaitteiston toteutuksen suunnitelmasta

Toimintaperiaatteen määrityksen jälkeen suunnittelua jatkettiin tekemällä koekytkentälevyllä prototyyppejä tuuletintestauksen eri osa-alueista. Koekytkentälevyllä

testattiin testauslaitteistossa mahdollisesti käytettävien komponenttien toimivuutta sekä yhteensopivuutta ennen varsinaisen piirilevyn tekemistä.

3.1 Testauslaitteiston ohjaus, mittausdatan tallennus ja käyttöliittymä

Työtä varten toimeksiantajalla oli Raspberry Pi 3 Model B yhden piirilevyn tietokoneita. Raspberry Pi 3:n prosessorina toimi ARM Cortex-A53 1.2 GHz moniydinprosessori, jonka teho riittäisi hyvin web-palvelimen ylläpitoon, tietojentallennukseen ja laitteiston ohjaukseen. Sen käyttöjärjestelmänä voitiin käyttää Raspbian GNU/Linux-pohjaista käyttöjärjestelmää, minkä ansiosta eri ohjelmistojen saatavuus oli hyvä. Raspberry Pi:ssä olevat 40 GPIO-pinniä eivät riittäisi tuulettimien ohjaukseen mutta niitä voisi käyttää muiden piirien ohjaukseen. (6.)

Raspberry Pi:n testauksessa huomattiin, että MicroSD-muistikorttia käyttäessä tietokannan tallennukseen web-palvelin hidastui, mutta käytettäessä USB-kiintolevyä samanlaista hidastumista ei ilmennyt. Raspberry Pi valittiin testauslaitteiston tietokoneeksi, koska USB-kiintolevyn kanssa se täytti kaikki vaatimukset. Lisäksi sitä voitiin ohjelmoida käyttäen Python-ohjelmointikieltä (7), josta työn tekijällä oli jo aiempaa kokemusta.

3.2 Kierrosnopeuden mittaus

Tuulettimien kierroslukujen mittauksen vaatimuksena oli, että 16 tuulettimelta voitiin yhtäaikaisesti mitata kierroslukusignaalia. Koekytkentälevyllä tehdyissä testeissä havaittiin, että helpoin tapa toteuttaa mittaus luotettavasti oli käyttää ulkoista keskeytystä. Sen johdosta valittavan komponentin tuli tukea vähintään 16:tta ulkoista keskeytystä. Mittauksen olisi voinut toteuttaa myös Raspberry Pi:llä, mutta sen GPIO-pinnien määrä ei välttämättä riittäisi kaikille tuulettimille, jos laitteeseen tulisi lisäksi näyttö. Siten koettiin varmemmaksi valita kierrosmittausta varten erillinen piiri. Oma toiveeni oli, että piiriä voitaisiin ohjelmoida Arduino-ohjelmointiympäristöä käyttäen, koska se oli minulle entuudestaan tuttu. Piirin tulisi myös toimia Raspberry Pi:n kanssa yhteisellä 3,3 voltin käyttöjännitteellä, jolloin piirilevyn suunnittelu yksinkertaistuisi.

Piiriksi valikoitui STM32F103C8. Piiri tukee 16:tta ulkopuolista keskeytystä, toimii 3,3 voltin jännitteellä ja pystyy kommunikoimaan Raspberry Pi:n kanssa sarjaliikenteellä (8). Se voidaan ohjelmoida Arduino-ohjelmointiympäristöllä. Arduino-tuki STM32F103C8-piirille on STM32duino-yhteisön ylläpitämä avoimen lähdekoodin projekti. (9.)

3.3 Tuulettimen ohjaus

Tuulettimien ohjauksen vaatimuksena oli, että 16:tta tuuletinta voitiin ohjata itsenäisesti, joten ohjaukseen tarvittiin kuusitoista 1 kHz:n taajuudella toimivaa PWM-signaalilähdettä. Vaihtoehtoina oli käyttää Raspberry Pi:tä tuottamaan ohjaussignaalit tai käyttää signaalin tuottoon erillistä komponenttia.

Testattavaksi hankittiin NXP:n valmistamia PCA9685-piirejä, joissa on 16 itsenäisesti ohjelmoitavaa kanavaa. Piiri toimii 24–1526 Hz:llä, sen käyttöjännite on 2,3–5,5 volttia ja sitä ohjataan sarjamuotoisella I2C-väylällä. (10.) Koekytkentälevyllä tehdyissä testeissä todettiin, että tuulettimien ohjaus oli helpompi toteuttaa PCA9685-piirillä kuin suoraan Raspberry Pi:llä. PCA9685-piirin ominaisuudet vastasivat vaatimuksia, minkä seurauksena se valittiin käytettäväksi testauslaitteistossa.

3.4 Virran mittaus

Tuulettimien virran mittaus päätettiin toteuttaa käyttämällä shunttivastusta. Shunttivastus on matalaresistanssinen vastus, jota käytetään virran mittaukseen. Virta lasketaan shunttivastuksen yli syntyvän jännitteen ja vastuksen resistanssin avulla. Shunttivastuksen valinnassa tärkeitä ominaisuuksia ovat vastuksen resistanssi, toleranssi, tehonkesto ja lämpötilankesto. (11.)

Työtä varten oli käytettävissä 0,01 ohmin (toleranssi 1 %) vastuksia. Näiden 2512-kokoisten vastusten tehonkesto oli 1 wattia. Tuulettimien maksimivirran ollessa 5 ampeeria, saatiin laskettua shunttivastukselta vaadittu tehonkesto käyttäen Ohmin lakia. Vastukseen kohdistuvaksi maksimitehoksi tuli 0,25 wattia, ja näin vastus täytti testauslaitteiston vaatimukset. (Kaava 1.)

$$P = U * I = R * I^2 = 0{,}01\Omega * 5A^2 = 0{,}25W \quad \textit{KAAVA 1}$$

Virran mittauksen tavoitteena oli mitata kaikkia 16 tuuletinta yhtä aikaa mutta koska 16-kanavaisten differentiaali AD-muuntimien hinta oli korkea, tutkittiin myös muita vaihtoehtoja. Koekytkentälevyllä tehtyjen testien perusteella analogisen differentiaalimultiplekserin läpi mitatun mittausdatan todettiin olevan riittävän tarkka testauslaitteistolle. Tällöin virran mittaus voitiin toteuttaa käyttämällä kahta kahdeksankanavaista differentiaalimultiplekseriä ja kahta analogista differentiaalituloa.

Differentiaalimultiplekseriksi valikoitui ADG707-piiri. ADG707-piirissa on kahdeksan differentiaalikanavaa, alhainen virtavuoto kanavien välillä ja nopea kanavan vaihto (12). Virran mittaus voidaan toteuttaa käyttämällä kahta ADG707-piiriä, koska yhdellä ADG707-piirillä on mahdollista ohjata kahdeksan tuulettimen virran mittaus yhteen analogiseen lähtöön. ADG707-piirin alhainen virtavuoto ja nopea kanavan vaihto lisäävät virran mittauksen tarkkuutta.

AD-muunnos toteutettiin kahdella ADS1115-piirillä. Ne toimivat 16-bittisinä AD-muuntimina ja niiden näytteenottonopeus I2C-väylää käyttäen on 860 näytettä sekunnissa, jolloin yhtä tuuletinta kohden on käytettävissä noin 100 näytteenottoa sekunnissa. Piirissä on sisäänrakennettuna ohjelmoitava vahvistin, minkä takia piirilevylle ei tarvitse suunnitella erillistä vahvistinta. (13.)

3.5 Lämpötilan mittaus

Työn vaatimuksena oli käyttää termoparia lämpötilan mittauksessa. Termoparin toiminta perustuu kahden erilaisen metallin liitoksessa syntyvään jännitteeseen. Jännitteen suuruus riippuu lämpötilasta. Mittaus perustuu lämpösähköiseen ilmiöön, jossa metallien liitoskohdan ja metallilankojen toisen pään välille syntyy lämpötilaerosta riippuvainen jännite. Ilmiötä kutsutaan Seebeck-ilmiöksi. (14.)

Vaatimuksissa oli määritetty käytettäväksi K-tyypin lämpöparia. Lämpötilan mittauksen tarkkuuden oli oltava vähintään ± 2 °C ja sen piti toimia -40 °C:n ja 80 °C:n välillä. Piirin käyttöjännitteen tuli olla 3,3 volttia ja sen oli pystyttävä kommunikoimaan Raspberry Pi:n kanssa.

Piiriksi valikoitui MAX31855K-piiri. Piirin käyttöjännite on 3,0–3,6 volttia, tarkkuus 14 bittiä, lämpötila-alue -200–1350 °C. Mittausdata luetaan sarjaportin kautta

(15). Piiri oli valmiiksi piirilevyllä, joten sen pystyi kiinnittämään suoraan Raspberry Pi:hin.

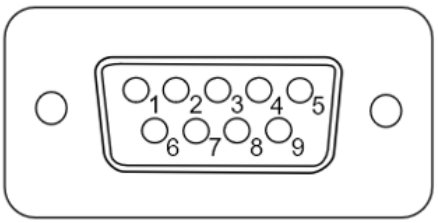
3.6 Muiden komponenttien valinta

Koekytkentälevyllä tehtyjen prototyyppien yhteydessä huomattiin, että jos useita erilaisia tuulettimia testasi yhtä aikaa, muodostui kierrosluku- ja ohjaussignaaliin runsaasti häiriöitä. Koekytkentälevylle kopioitiin Nokian tukiasemissa ollut tuuletinohjauksen rakenne, jolloin häiriöt poistuivat. Häiriön poisto, suodatus ja signaalin suojaus koostui vastuksista, kondensaattoreista, Schmitt-liipaisimesta ja transistorista. Tarkempi rakenne näkyy kuvassa 5 piirikaavion suunnitteluosiossa.

Schmitt-liipaisimeksi valittiin NXP:n valistama kuusikanavainen 74HC14-piiri. Se toimii 2,0–6,0 voltin jännitteellä (16). Transistorin valinnassa kriteereinä oli tyyppi (npn) ja jännitteen kesto. Transistoriksi valittiin BC846-transistori SOT-26 koteloinnilla. Se on npn-tyyppinen transistori ja sen jännitteen kesto on 65 volttia (17). Vastukset ja kondensaattorit valittiin siten, että niiden toleranssi oli enintään 5 % ja jännitteen kesto vähintään 55 volttia. Koteloinnin minimikooksi vastuksille ja kondensaattoreille valittiin 0805, koska suunnitteluvaiheessa ei ollut varmaa, jouduttaisiinko piirilevy juottamaan käsin.

3.7 Tuulettimien kiinnitys testauslaitteistoon

Tuulettimien kytkemiseksi testauslaitteistoon valittiin DB9-liittimet, koska niitä oli helposti saatavilla ja koska laite ei tarvinnut IP-luokitusta, voitiin DB9-liittimistä valita halvempi IP-testaamaton versio. Yhteen DB9-liittimeen voitiin kytkeä kaksi tuuletinta, jolloin laitteeseen tarvittiin kahdeksan DB9-liitintä. Kotelon puolelle valittiin naarasliittimet, koska haluttiin ehkäistä vahingossa liittimen johtimiin koskemista. DB9-liitin kytkettiin tuulettimiin kuvan 4 mukaisella tavalla.



Tuuletin 1	DB9	Tuuletin 2	DB9
VCC	1	VCC	5
PWM	2	PWM	4
RPM	7	RPM	8
Ground	6	Ground	9

KUVA 4. Tuulettimen kytkentä DB9-liittimeen

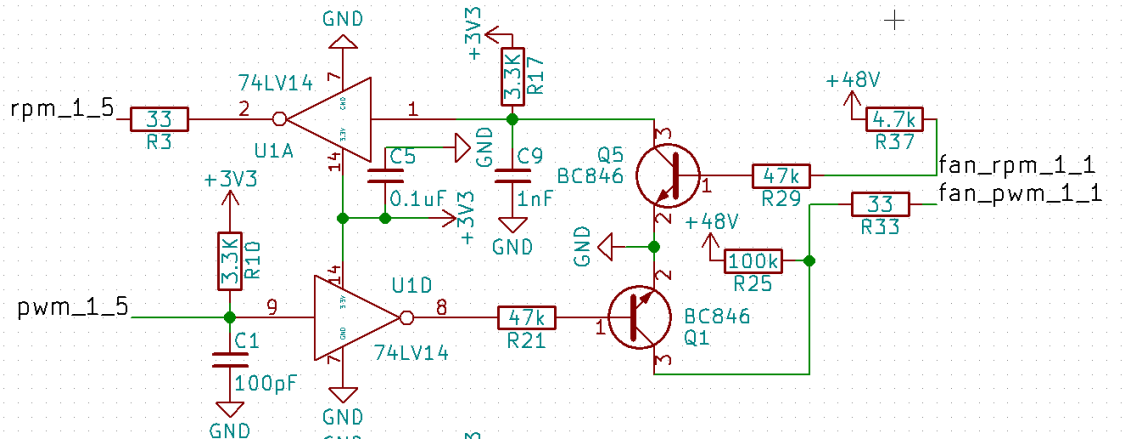
4 PIIRILEVYN SUUNNITTELU JA TOTEUTUS

Piirilevyn suunnittelu toteutettiin käyttämällä KiCad-ohjelmistoa. KiCad on avoimen lähdekoodin ohjelmisto, joka on tarkoitettu piirilevyn suunniteluun. Ohjelmiston ensimmäinen versio julkaistiin vuonna 1992 ja nykyinen versio 5.0.1 julkaistiin lokakuussa 2018. (18.) Piirikaavio ja piirilevy suunniteltiin käyttäen versiota 4.0.7.

4.1 Piirikaavio

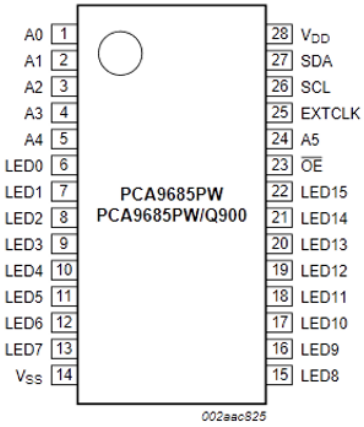
Piirikaavion tekeminen aloitettiin määrittelemällä yksi 3,3 voltin ja kaksi 10–55 voltin jännitelinjaa sekä yksi maalinja. 3,3 voltin jännitelinjan jännite saatiin Raspberry Pi:n 3,3 voltin pinnistä. Jännitelinja toimi logiikkapiirien käyttöjännitteenä. Kaksi 10–55 voltin jännitelinjaa mahdollistivat kahdella eri jännitteellä toimivien tuulettimien yhtäaikaisen testaamisen. Jännitelinjat jaettiin siten, että tuulettimet 1–8 olivat linjassa yksi ja tuulettimet 9–15 linjassa kaksi. Molempien linjojen komponentit valittiin siten, että ne kestivät vähintään 55 voltin jännitteen.

Tuulettimien ohjaussignaalin ja kierroslukusignaalin häiriön poisto ja suodatus toteutettiin kondensaattoreilla ja vastuksilla. Suodatettu signaali muutettiin takaisin kanttiaalloksi käyttäen Schmitt-liipaisinta. Logiikkapuolen jännite vedettiin vastuksella 3,3 voltin jännitelinjaan ja tuuletinpuolen jännite 10–55 voltin jännitelinjaan. Kuvassa 5 näkyy tuulettimien mittaus- ja ohjaussignaalien suodatus.



KUVA 5. Testauslaitteiston ohjaus- ja kierroslukusignaalin suodatus ja suojaus

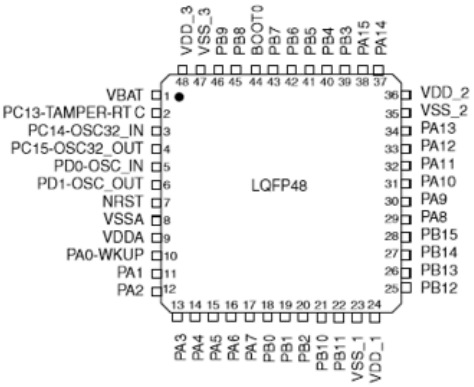
PCA9685-piiri kytkettiin I2C-väylään siten, että sen I2C-väylän osoitteeksi tuli 0x40. Tuulettimien ohjaussignaalit kytkettiin PCA9685-piirille kuvan 6 taulukon mukaan.



Tuuletin	PCA9685	Tuuletin	PCA9685
1	6	9	15
2	7	10	16
3	8	11	17
4	9	12	18
5	10	13	19
6	11	14	20
7	12	15	21
8	13	16	22

KUVA 6. Ohjaussignaalin kytkennät PCA9685-piiriin

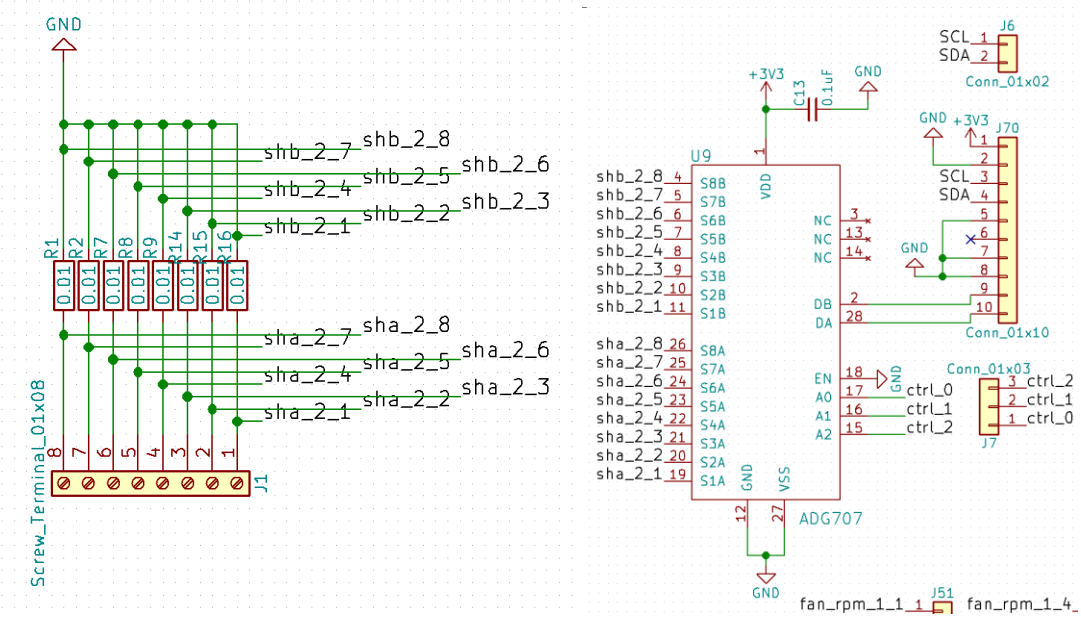
STM32F103C8-piiri kytkettiin Raspberry Pi:hin käyttämällä pinnejä PA9 (TX) ja PA10 (RX). Valitussa STM32F103C8-piirissä on mahdollista käyttää 16:tta ulkopuolista keskeytystä. Piirissä keskeytykset on jaettu kahden väylän kesken, jolloin esimerkiksi pinnien PA4 ja PB4 kanssa keskeytyksiä ei voi käyttää yhtä aikaa, koska tällöin keskeytykset tulisivat kaksinkertaisina. (8.) Kierroslukusignaali ohjattiin STM32F103C8-piirille kuvan 7 taulukon mukaisesti.



Tuuletin	STM32	Tuuletin	STM32
1	PA0	9	PB9
2	PA1	10	PB8
3	PA2	11	PB11
4	PA3	12	PB10
5	PA4	13	PB12
6	PA5	14	PB13
7	PA6	15	PB14
8	PA7	16	PB15

KUVA 7. Kierroslukusignaalin kytkennät STM32F103C8-piirille

ADG707-piirien ohjaus toteutettiin Raspberry Pi:n avulla. Raspberry Pi:stä vedettiin kolme johdinta testilaitteen piirilevylle, josta ne johdettiin molempien ADG707-piirien pinneihin A0, A1 ja A2. ADG707-piirien tuloihin ohjattiin tuulettimien shunttivastusten ylimenojännite. Molempien ADG707-piirien differentiaalilähdöt ohjattiin omille ADS1115-piireilleen. ADS1115-piirit kytkettiin Raspberry Pi:n I2C-väylään. Niiden osoitteeksi määritettiin 0x48 ja 0x49 kytkemällä toisen piirin ADDR-pinni maahan ja toisen 3,3 volttiin (13). Kuvassa 8 näkyy virranmittauksen piirikaavio.



KUVA 8. Jännitteen mittaus shunttivastuksen yli

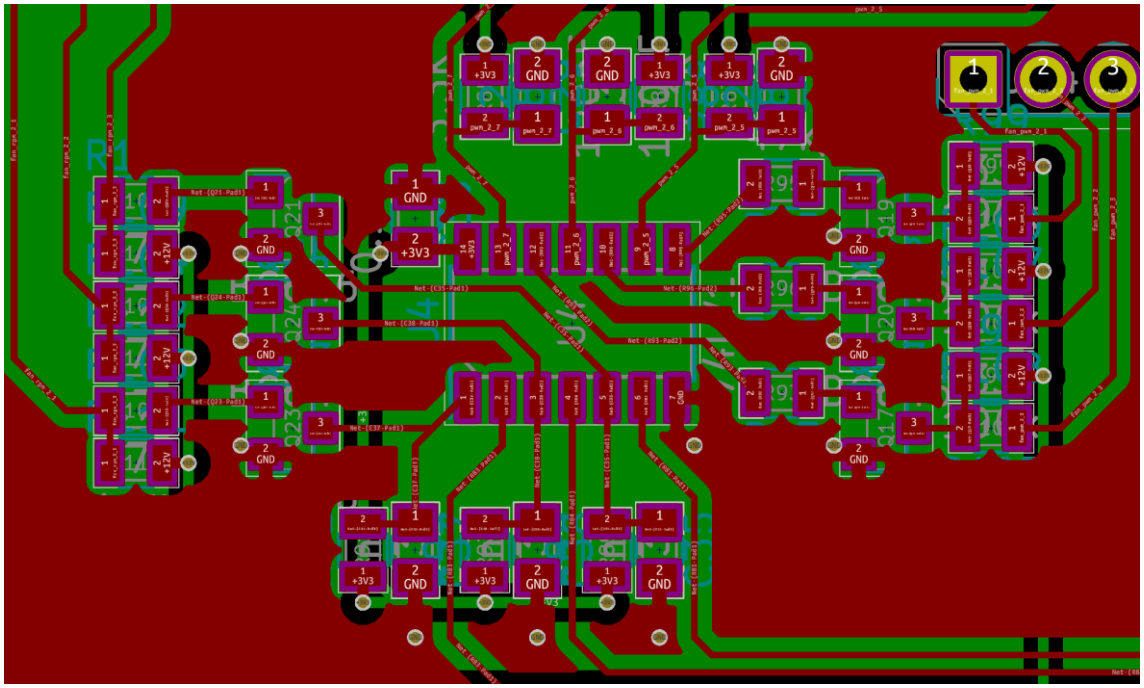
Lämpötilan mittaukseen tarkoitettu MAX31855-piiri kiinnitettiin suoraan Raspberry Pi:hin, joten sitä ei ole merkitty piirikaavioon. Lisäksi Raspberry Pi:hin kytkettiin näyttö.

4.2 Piirilevyn suunnittelu

Piirikaavion valmistuttua asetettiin piirikaaviossa käytetyille komponenteille valitut kotelotyypit. Tämän jälkeen piirikaavion pohjalta luotiin KiCad:llä piirilevylle

yhteydet eri komponenttien välille. Kun komponenttien väliset yhteydet oli merkitty, aseteltiin komponentit siten, että niiden välille pystyttiin vetämään kuparijohdinlinjat.

Piirilevyn johdinleveydet ja läpiporauksien koot määritettiin IPC-2221 standardin mukaisesti. Piirilevystä tehtiin kaksipuoleinen, jotta piirilevy pysyi yksinkertaisena. Piirilevyn koko pyrittiin pitämään pienenä, jotta se mahtuisi sille tarkoitettuun koteloon, johon piti piirilevyn lisäksi mahtua Raspberry Pi ja tuulettimien johdotukset. Piirilevyn molempien puolien tyhjät alueet täytettiin maa-alueella. Kuvassa 9 näkyy esimerkki piirilevyn suunnitteluvaiheesta.

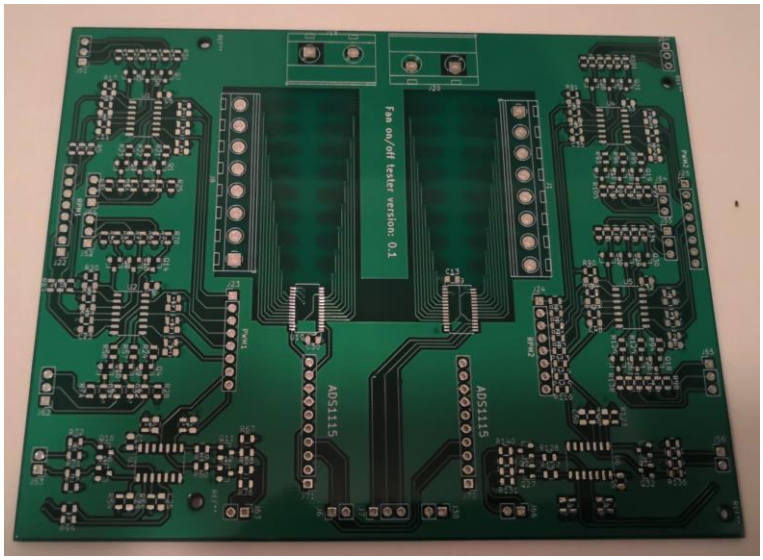


KUVA 9. Ohjaus- ja kierroslukusignaalin suodatus ja suojaus piirilevyllä

Piirilevystä tehtiin yksi versio. Piirilevystä aiottiin vielä tehdä toinen versio, jossa liittimien paikat ja merkinnät olisivat olleet paremmin sijoitettu, mutta toisen version valmistamiseen ei opinnäytetyön aikataulussa ollut aikaa. Lopullisesta piirilevystä tuli 140 mm x 110 mm. (Liite 1.)

4.3 Testilaitteen valmistus

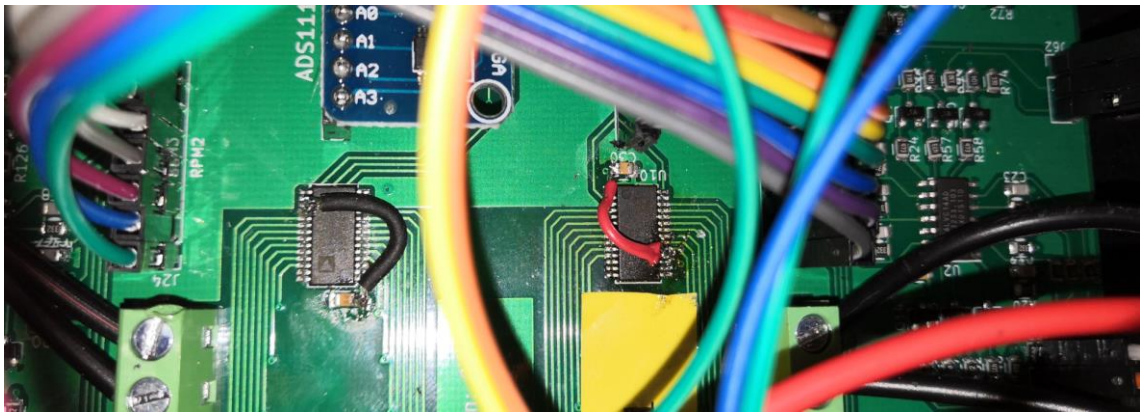
Piirilevyn suunnittelun jälkeen siitä tehtiin Gerber-tiedosto, joka sisälsi stensiilitiedoston, johdinkerrokset, piirilevymerkinnät, poraustiedostot ja juotteenestomaskin. Tiedoston avulla valmistettiin kuvassa 10 näkyvä ensimmäinen versio piirilevystä.



KUVA 10. Piirilevyn ensimmäinen versio

Piirilevyn valmistuttua siihen ladottiin komponentit käsin. Komponenttien ladonnan jälkeen levy laitettiin reflow-uunin, jossa levyä kuumennettiin käytetyn tinapastan mukaisessa profiilissa. Sen jälkeen levy tarkistettiin mikroskoopilla mahdollisten juotosvirheiden varalta.

Ennen laitteen kotelointia prototyyppilevy testattiin testiohjelmalla, joka oli tehty testilaitteen suunnitteluvaiheessa. Tällöin huomattiin, että ADG707-piirit eivät vaihtaneet kanavia, kun niitä koetettiin ohjata. Tutkimalla piirilevyä ja ADG707-piirin dokumenttia löydettiin piirilevysuunnitelmasta virhe. ADG707-piirin pinni 8 (EN) olisi tullut kytkeä 3,3 volttiin ja se oli kytkettynä maahan (12). Virhe korjattiin katkaisemalla pinniin menevä maajohdin ja lisäämällä siihen hyppylanka 3,3 volttiin kuvan 11 mukaisesti.



KUVA 11. Korjattu ADG707-liitos

Testauslaitteiston kotelointi aloitettiin prototyyppilevyn läpäistyä testiohjelman. Etupaneeliin tuli kahdeksan DB9-liitintä, joihin voidaan kiinnittää kaksi tuuletinta kuhunkin liittimeen. Lisäksi etupaneeliin sijoitettiin liitin virtalähteelle ja K-tyypin termoparille. Takapaneeliin tuli RJ45- ja MicroUSB-liitin. Kotelon päälle tehtiin paikka näytölle, jossa tulisi näkymään laitteen IP-osoite etäkäyttöä varten. Lisäksi tuulettimille tehtiin ruuvattavat pikaliittimet, joiden avulla niiden johtimet voitiin nopeasti kytkeä DB9-liittimiin. Kuvassa 12 näkyy valmis testauslaitteiston prototyyppi.

KUVA 12. Ensimmäinen prototyyppi koottuna

5 TESTIOHJELMAN JA KÄYTTÖLIITTYMÄN OHJELMOINTI

Testiohjelmisto toteutettiin käyttäen Python-ohjelmointikieltä, lukuun ottamatta kierroslukusignaalin mittausta STM32F103C8-piirillä ja käyttöliittymän käyttäjälle näkyvää osaa. Ensimmäisessä käytettiin Arduino-ohjelmointiympäristöä ja jälkimmäisessä Javascriptiä ja HTML-kuvauskieltä. Ohjelmaa muokattiin monessa vaiheessa, muun muassa komponenttien testauksen aikana. Testauslaitteiston valmistuttua ohjelma koottiin ja viimeisteltiin lopulliseen muotoonsa. Tässä luvussa kuvaillaan ohjelmiston toimintaperiaatteet.

5.1 Tuulettimien ohjaus ja mittaus

Mittausohjelmisto asetettiin käynnistymään aina Raspberry Pi:n käynnistyessä. Käynnistyttyään ohjelma aloittaa portin 56666 kuuntelemisen käyttäen Pythonin Asyncio-kirjastoa. Porttiin tulevan viestin mukaisesti ohjelma käynnistää, pysäyttää tai jatkaa testiä. Hyväksyttäviä viestejä määritettiin neljä erilaista: start käynnistää testin, stop pysäyttää testin, pause keskeyttää testin väliaikaisesti ja continue jatkaa testausta. Start-viestissä on lisäksi mukana JSON-tiedosto, joka sisältää käyttöliittymässä asetetut arvot. Viesti start käynnistää testauksen omassa säikeessään, jossa testin pituus, tuulettimien ohjausprofiilit, syklien määrä ja syklien pituus asetetaan JSON-tiedostossa olleiden arvojen mukaisesti. Mittaussäikeitä voi olla vain yksi kerrallaan toiminnassa. Testausohjelmisto jaettiin neljään osioon: kierroslukumittaus, virran mittaus, tuulettimien ohjaus ja lämpötilan mittaus.

Kierroslukumittaus toteutettiin käyttämällä piiriä STM32F103C8. Keskeytysten avulla tuulettimilta tuleva kierroslukupulssi ohjataan nousevalla reunalla lisäämään taulukossa olevaa arvoa yhdellä. Pulsseja lasketaan yhden sekunnin ajan, jonka jälkeen pulssien kuuntelu lopetetaan. Taulukon arvot lähetetään sarjaporttiin 1 (pinnit 30 ja 31). Sarjaporttiin kirjoittamisen jälkeen taulukon arvot nollataan ja keskeytysten kuuntelu aloitetaan alusta. Sarjaporttiin kirjoitettavassa datassa tuulettimet on eroteltu puolipisteellä ja rivin alku on merkitty M-kirjaimella. Data

on raakamuodossa eli siitä ei ole laskettu todellista kierrosnopeutta. Päällä olleessaan STM32F103C8-piiri mittaa tuuletinten kierroslukua jatkuvasti ja kirjoittaa mittausdatan sarjaporttiin sekunnin välein.

Raspberry Pi ohjelmoitiin lukemaan piirin STM32F103C8 lähettämää sarjaporttiliikennettä käyttämällä Pythonin Serial-kirjastoa. Ohjelma asetetaan lukemaan sarjaliikennettä M-merkistä aina rivinvaihtoon asti, jolloin saadaan edellisen sekunnin pulssien lukumäärä. Luetusta datasta erotellaan tuuletinkohtaiset tiedot puolipisteiden avulla. Datasta lasketaan tuulettimien todellinen kierrosnopeus kierroksina minuutissa kuvan 13 mukaisella tavalla. Laskutoimitusten jälkeen arvot kirjoitetaan tietokantaan kunkin tuulettimen kohdalle.

```
def format_rpm(self, reading):
    values = reading.split(":")
    values.pop(0)
    try:
        temp = [int(int(x) * 60 / 2.0 + 0.5) for x in values[:-1]]
    except:
        temp = [0] * 16
    return temp
```

KUVA 13. Kierrosnopeuden lasku ja erottelu tuuletinkohtaisesti

Virran mittaus toteutetaan käyttämällä ADS1115-piireille tehtyä Adafruit_ADS1x15-kirjastoa. Kirjaston avulla piirit määritetään toimimaan piirilevysuunnittelussa niille asetetuissa I2C-väylän osoitteissa 0x48 ja 0x49. Piireissä oli ohjelmoitava esivahvistin, joka asetettiin arvoon 4, jolloin piirien mittausalueeksi tuli ± 1,024 volttia. Tästä voitiin laskea virran mittaukseen kerroin 1,024 volttia / 32767.

Kahden rinnakkain kytketyn ADS1115-piirin mittaaman arvon lukeminen Raspberry Pi:llä kestää 0,02 sekuntia johtuen Adafruit_ADS1x15-kirjaston hitaudesta. ADS1115-piirin teoreettinen nopeus olisi 860 näytettä sekunnissa. Jokaiselta tuulettimelta mitataan kymmenen arvoa, joista poistetaan pienin ja suurin arvo. Mittausdatasta lasketaan keskiarvo, minkä jälkeen se kerrotaan virran mittauksen kertoimella. Saadusta arvosta saadaan kuvan 14 mukaisesti tuulettimen virrankulutus käyttäen Ohmin lakia

$$I = \frac{U}{R} \qquad \textit{KAAVA 2}$$

missä U on mitattuarvo ja R on laitteessa käytetyn shunttivastuksen resistanssi.

```
def calculate_current(self, ca):
    valu = self.multi * (sum(ca) / float(len(ca)))
    valu = -valu / resistor
    if valu < 0.01:
        return 0
    else:
        return valu
```

KUVA 14. Tuulettimen virrankulutuksen lasku

Seuraavaan tuuletinparin mittaukseen siirtyminen tapahtuu ohjaamalla ADG707-piirit vaihtamaan tulot seuraavaan. ADG0707-piirien ohjaus tapahtuu rinnakkain. Tästä johtuen kaikkien tuulettimien virran mittaus kestää minimissään 1,6 sekuntia. Ohjelmallisesti määritetään tuulettimien näytteenottoajan minimiksi 2 sekuntia. Näytteenoton jälkeen virran mittauksesta saadut tiedot tallennetaan tietokantaan kunkin tuulettimen kohdalle.

Tuulettimien ohjauksessa käytetään PCA9685-piirille tehtyä PWM Adafruit_PCA9685 -kirjastoa. PCA9685-piiri asetetaan toimimaan 1 kHz:n taajuudella ja piirin I2C-väylän osoitteeksi määritetään 0x40. Ohjelmisto toteutetaan siten, että jokaista tuuletinta pystyy säätämään erikseen. PCA9685-piiri on 12-bittinen, jolloin säätö tapahtuu alueella 0–4096.

On / Off -testaukselle on tehty valmis testiprofiili. Siihen on määritetty, kuinka kauan tuulettimet ovat ohjattuna pysähdystilaan ja kuinka kauan huippunopeuteen. Oletusarvoksi on asetettu 30 sekuntia, joka testien perusteella riittää molempien tuuletintyyppien kiihtymiseen huippunopeuteen ja kokonaan pysähtymiseen. Arvoa on mahdollista muokata testin käynnistyksen yhteydessä, mikäli halutaan optimoida testin kesto tuuletintyypin mukaan.

Lämpötilan mittauksessa käytettyyn MAX31855-piiriin ei löytynyt kirjastoa. Piirin ajuri toteutettiin Pythonilla, koska mittaus tapahtuu kerran kahdessa sekunnissa ja Python-kielisen ohjelman suorituskyky riittää kolmen GPIO-pinnin lukemiseen. MAX31855-piiriltä luetaan 32 tavua dataa, joka muutetaan Celsius-asteiksi

MAX31855-piirin kuvassa 15 näkyvällä tavalla. Ajuri toteutettiin käyttämällä apuna MAX31855-piirin datalehteä. Lämpötilan mittausintervallin pystyy määrittämään testikohtaisesti. Oletuksena se on 5 sekuntia. Mittauksen jälkeen tieto tallentuu tietokantaan.

```
def read_spi(self):
    bytesin = 0
    self.pi.write(19, 0)

    for i in range(32):
        self.pi.write(13, 0)
        bytesin = bytesin << 1
        if self.pi.read(26):
            bytesin = bytesin | 1
        self.pi.write(13, 1)

    self.pi.write(19, 1)
    return bytesin

def get_celcius(self, data):
    cel_data = (data >> 18) & 0x3FFF
    if cel_data & 0x2000:
        temp = ~cel_data & 0x1FFF
        temp += 1
        temp *= -1
    else:
        temp = cel_data & 0x1FFF
    return temp * 0.25
```

KUVA 15. MAX31855 ajurin toteutus

Tietokanta on toteutettu käyttäen SQLite-relaatiotietokantajärjestelmää. Tietokanta luodaan käyttäen Pythonin sqlite3-kirjastoa. Tietokannan luonti ja muokkaus tapahtuu testiohjelmistossa. Muut ohjelman osat voivat pelkästään lukea tietokannasta arvoja. Erottelu on tehty, koska SQLite-tietokanta ei salli samanaikaista tallennusta tietokantaan useasta lähteestä (19).

5.2 Käyttöliittymän toteutus

Käyttöliittymän taustajärjestelmän (back end) toteutuksessa käytetään Flask-mikrosovelluskehystä. Taustajärjestelmän toteutukseen Pythonilla oli mahdollista valita useita erilaisia sovelluskehyksiä, kuten Django ja Pyramid (20). Flaskin etuna näihin oli sen yksinkertaisuus. Palvelin voitiin toteuttaa yhdessä Python-tiedostossa, jossa luettiin tietokannasta testausdata, luotiin HTML-sivustot ja lähetettiin testiohjelmistolle käyttäjän komennot. (21.)

Flask-sovellus tarvitsee toimiakseen kaksi kansiota: templates ja static. Templates-kansioon tulevat verkkosivuston luomiseen käytetyt HTML-tiedostot. Tiedos-

toista Flask luo Jinja2-sivupohjamoottoria käyttäen selaimelle näkyvän HTML-tiedoston. Jinja2 tukee HTML-tiedostoissa Pythonin tyylistä ladontakieltä, jonka avulla voidaan dynaamisesti luoda sivustoja. Static-kansio sisältää sivuston käyttämät tiedostot, kuten CSS-, Javascript- ja kuvatiedostot. (21.)

Palvelinsovelluksessa määritetään käyttöliittymän URL-osoitteet. Sivustolle luodaan kolme pääsivua: index.html, main.html ja testdata.html. Sivustot luodaan HTML-tiedoston pohjalta käyttäen Jinja2-sivupohjamoottoria. Tämän lisäksi testdata.html -sivustoon luodaan test_data-objekti, joka sisältää tietokannassa olevat testit kuvassa 16 näkyvällä tavalla.

```
@app.route("/views/testdata.html")
def test_data():
    try:
        conn = sqlite3.connect(sqlite_file)
        c = conn.cursor()
        c.execute("""SELECT * FROM Testing_data""")
        result = c.fetchall()
        c.close()
        conn.close()
    except:
        result = [[0]*6]*4
    return render_template(
        "views/testdata.html",
        test_data=result[::-1]
    )
```

KUVA 16. testdata.html luonti ja testidatan lataus

Lisäksi on määritetty funktiot, joita käyttöliittymä käyttää ladatakseen tai lähettääkseen tiedostoja tai käskyjä käyttöliittymän ja palvelimen välillä. Funktiot start_test, stop_test, pause_test ja continue_test välittävät käyttäjältä tulevat käskyt testiohjelmistolle. Kuvassa 17 näkyy start_test funktion toiminta.

```
@app.route("/_start_test", methods=["POST"])
def start_test():
    jsdata = request.get_json()
    jsdata1 = json.dumps(jsdata).encode('utf-8')
    msg = ""
    try:
        with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
            s.connect((HOST, PORT))
            s.sendall((b'start' + jsdata1))
            msg = json.dumps({'success':True}), 200, {'ContentType':'application/json'}
    except:
            msg = json.dumps({'success':False}), 400, {'ContentType':'application/json'}
    return msg
```

KUVA 17. Testin aloituskäskyn ja asetusten ohjaus käyttöliittymästä testiohjelmistolle

Measurements-funktio hakee tietokannasta viimeisimmän mittaustuloksen kaikilta tuulettimilta ja palauttaa sen JSON-objektina. Funktio get_csv hakee valitun testin mittausdatan, muuntaa sen CSV-tiedostoksi käyttäen Pandas -kirjastoa ja palauttaa sen ladattavaksi käyttäjälle. Funktio get_csv400 toimii kuten get_csv-funktio, mutta se hakee mittausdatasta viimeisimmät 400 tehtyä mittausta. get_plot-funktio tekee 400 viimeisimmästä mittauksesta kuvaajan käyttäen Pandas- ja matplotlib-kirjastoja, pakkaa ne zip-tiedostoksi, jonka se palauttaa käyttäjälle ladattavaksi kuvassa 18 näkyvällä tavalla.

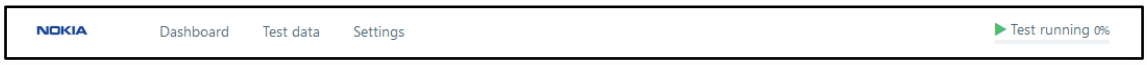
```
@app.route("/get_plot_<path:page>")
def get_plot(page):
    conn = sqlite3.connect(sqlite_file)
    c = conn.cursor()
    c.execute("SELECT * FROM Test_set_" + page + " ORDER BY Measurement DESC LIMIT 1")
    size = c.fetchone()[0]
    if size >= 400:
        database = pd.read_sql("SELECT * FROM Test_set_" + page + " WHERE Measurement > (SELECT MAX(Measurement) - 400 FROM Test_set_" + page + ")", conn)
    else:
        database = pd.read_sql("SELECT * FROM Test_set_" + page, conn)
    f = io.BytesIO()
    zi = zipfile.ZipFile(f, 'w')
    for i in range(1,17):
        rpm_file = make_plot('fan_'+ str(i) + '_rpm', database)
        cur_file = make_plot('fan_'+ str(i) + '_current', database)
        file_name_rpm = "rpm_" + str(i) + ".png"
        file_name_cur = "current_" + str(i) + ".png"
        zi.writestr(file_name_rpm, rpm_file.getvalue())
        zi.writestr(file_name_cur, cur_file.getvalue())
        rpm_file.close()
        cur_file.close()
    zi.close()
    conn.close()
    f.seek(0)
    return Response(f,
            mimetype="application/zip",
            headers={"Content-disposition":
            "attachment; filename=test_set_" + page +".zip"})

def make_plot(mes_name, data):
    temp_file = io.BytesIO()
    plt = data[[mes_name]].plot(kind='line', figsize=(18.0000, 8.000))
    fig = plt.get_figure()
    fig.savefig(temp_file, format='png', dpi=100)
    temp_file.seek(0)
    return temp_file
```

KUVA 18. Kuvaajien luonti mittausdatasta ja pakkaus zip-tiedostoksi

Käyttöliittymän edustajärjestelmä (front end) toteutettiin verkkosivuna käyttäen Bootstrap 4 -ohjelmistokehystä ja Jquery-Javascript-kirjastoa. Bootstrap 4 -ohjelmistokehys tarjoaa 12-palstaisen ruudukon, joka helpottaa sivuston asettelun luomisessa sekä useita komponentteja, joilla sivustoon voidaan lisätä ominaisuuksia. (22.) Jquery on avoimen lähdekoodin Javascript-kirjasto, joka helpottaa vuorovaikutteisten verkkosivujen tekoa (23).

Käyttöliittymä muodostuu neljästä näkymästä: yläpalkki, pääsivu, testausdata ja testin käynnistys. Yläpalkki luodaan tiedostossa index.html. Palkin avulla siirrytään käyttöliittymässä eri osa-alueille. Siinä näkyy myös testin tämän hetkinen

tila. Palkki on näkyvillä kaikissa näkymissä. Kuvassa 19 näkyy yläpalkki ja siinä olevat ominaisuudet.



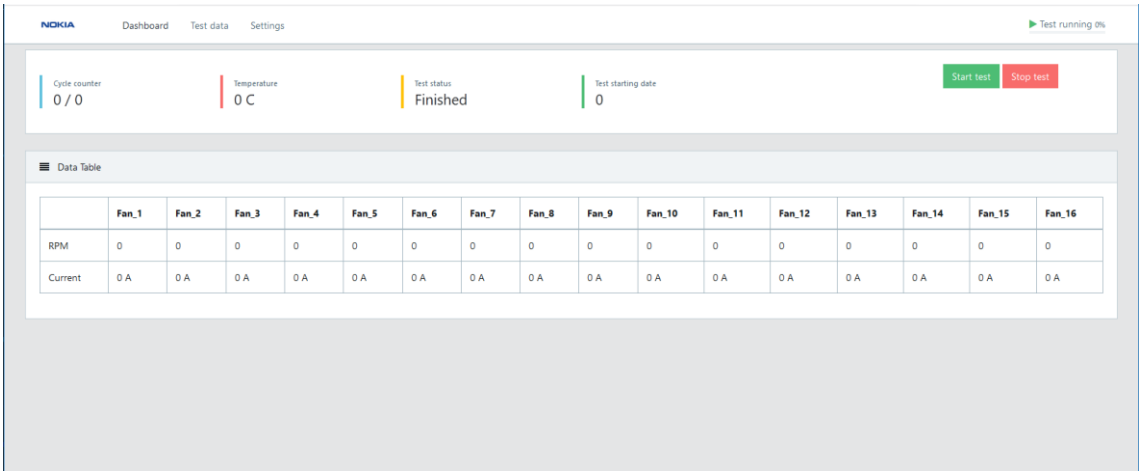
KUVA 19. Käyttöliittymän navigointipalkki

Pääsivu luodaan main.html-tiedostossa. Sivulle on luotu Jinja2-sivupohjamoottorilla taulukko testattavista tuulettimista ja niiden kierrosnopeuden ja virran mittausdatasta. Kuvassa 20 näkyy, kuinka taulukko luodaan.

```
<div class="col-lg-12">
  <div class="card">
    <div class="card-header">
      <i class="fa fa-align-justify"></i> Data Table
    </div>
    <div class="card-body">
      <table class="table table-responsive-sm table-bordered">
        <thead>
          <tr>
            <th></th>
            {% for o in range(1,17) %}
      <th> Fan_{{ o }} </th>
          {% endfor %}
          </tr>
        </thead>
        <tbody>
          {% for x in range(1,3) %}
          <tr>
            {% for number in range(0,17) %}
              {% if x == 1 and number == 0%}
              <td>RPM</td>
              {% elif x == 2 and number == 0%}
              <td>Current</td>
              {% else %}
              <td id="fan_{{number}}_{{x}}"> ? </td>
              {%endif%}
            {% endfor %}
          </tr>
          {% endfor %}

        </tbody>
      </table>
```

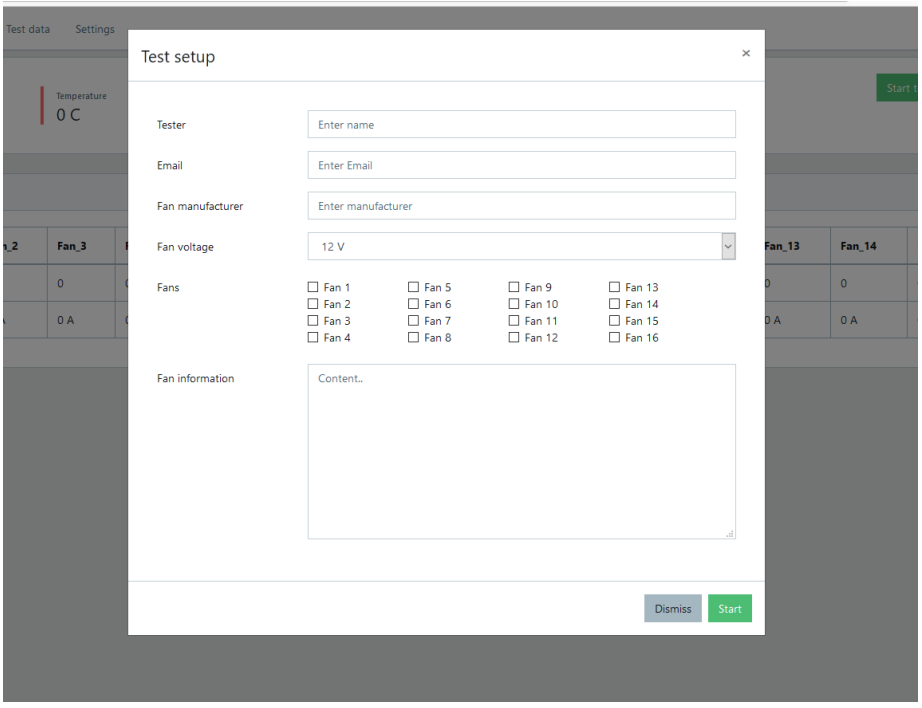
KUVA 20. Testausdatataulukon luonti Jinja2-sivupohjamoottorilla

Kierrosnopeuden ja virran mittausdata haetaan kolmen sekunnin välein Flask-palvelimelta käyttäen measurements-funktiota. Mittausdata asetetaan Jqueryn avulla taulukkoon ilman sivuston uudelleen lataamista. Samalla päivitetään sivulla näkyvä sen hetkinen lämpötila sekä testintila. Sivulla on kaksi nappia: Start test ja Stop test. Stop test pysäyttää käynnissä olevan testin. Jos testi ei ole käynnissä tai testiohjelmaan ei saada yhteyttä, Flask-palvelin lähettää käyttäjälle virheilmoituksen. Start test käynnistää testin käynnistysnäkymän. Kuvassa 21 näkyy pääsivun asettelu.



KUVA 21. Pääsivu

Testin käynnistysnäkymä on toteutettu käyttäen Bootstrap 4 -ohjelmistokehyksen modal-komponenttia. Modal-komponentti liukuu dynaamisesti pääsivun päälle niin, että pääsivu on testin käynnistysnäkymän taustalla kuvassa 22 esitetyllä tavalla. Näkymässä asetetaan testauksen tiedot kuten testaajan nimi, sähköpostiosoite, tuuletinvalmistaja, tuulettimen käyttöjännite, testattavien tuulettimien portit ja lisäinformaatio tuulettimista.



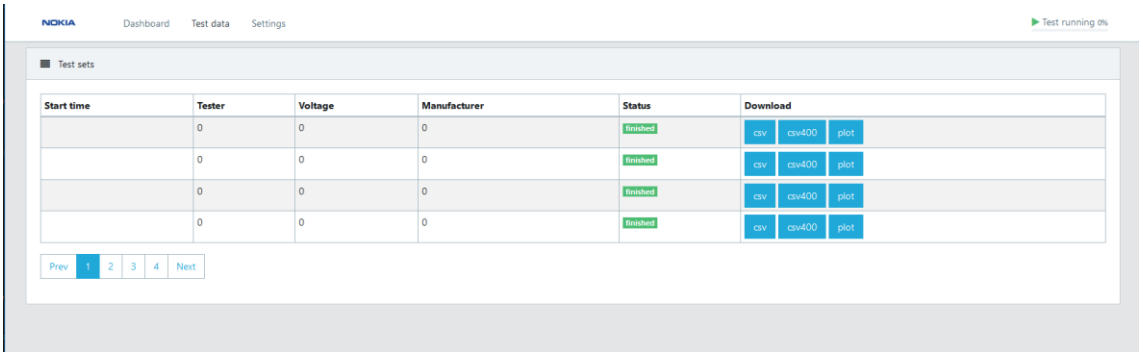
KUVA 22. Testin käynnistysnäkymä

Testi käynnistetään Start-napista. Start-nappi lähettää Flask-palvelimelle käyttäjän asettamat tiedot Jquery-kirjaston avulla kuvassa 23 näkyvällä tavalla. Jos testi on jo käynnissä tai testiohjelmaan ei saada yhteyttä, näytetään käyttäjälle virheilmoitus.

```
$("#startBtn").click(function(){
  var boxes = [];
  for (var index = 1; index < 17; index++) {
    if($("#fcheck"+index).prop('checked')) {
      boxes.push(index);
    }
  }
  $.ajax({
          url: '/_start_test',
          type: 'post',
          headers: {'content-type' : 'application/json'},
          data: JSON.stringify({
                  "tester": $("#tester-input").val(),
                  "email": $("#email-input").val(),
                  "fm": $("#fm-input").val(),
                  "info": $("#fan-info-input").val(),
                  "voltage": $("#voltage-select option:selected").text(),
                  "fans": boxes
                  }),
      success: function() {
        $('#starttest').modal('hide');
        alert('test started');
      },
      error: function(){
        $('#starttest').modal('hide');
        alert('error');
      },
  });
```

KUVA 23. Testiasetusten lähetys käyttöliittymästä Flask-sovellukseen

Testausdatanäkymässä on taulukko tehdyistä testeistä. Taulukot täytetään sivun luomisen yhteydessä Flask-palvelimelta saatujen tietojen pohjalta. Taulukoissa näkyy testitiedot ja testin tila. Kunkin testin kohdalla on kolme nappia, joista käyttäjä voi ladata koko testidatan CSV-muodossa, viimeisimmät 400 mittausta CSV-muodossa tai kuvaajan viimeisimmistä 400 mittauksesta kaikista tuulettimista. Kuvassa näkyy 24 testausdatanäkymä.



KUVA 24. Testausdatanäkymä

6 POHDINTA

Testauslaitteistolle asetetut tavoitteet täyttyivät ja sitä voi käyttää On / Off -testaukseen. Laite luovutettiin lämpötestaus-tiimille sekä opastettiin laitteen käyttämisessä. Testauslaitteistoon jäi kuitenkin paljon parannettavaa sekä elektroniikkasuunnittelun että ohjelmiston osalta.

Testauslaitteiston suojaus on tällä hetkellä heikko. Tuuletinkohtaiset sulakkeet on sijoitettu liitinadapteriin. Laitteeseen kannattaisi suunnitella erillinen piirilevy, johon sulakkeet sijoitettaisiin. Tällöin sulakkeiden vaihtaminen olisi helpompaa. Tällä hetkellä laitteistossa ei myöskään ole jännitteen vastanapaisuuden suojausta, minkä johdosta väärin kytketty virtalähde todennäköisesti rikkoo koko laitteen.

Testauslaitteiston piirilevyyn jäänyt virhe tulisi korjata piirilevysuunnitelmassa, jos levystä tehdään uusi versio. Uuteen versioon tulisi korjata myös liittimien sijoittelu tai mahdollisesti vaihtaa piikkirimaliittimet johonkin muuhun. Nykyisellä liitinten sijoittelulla testauslaitteiston johdotus vei paljon aikaa. Komponenttien määrää voisi todennäköisesti vähentää poistamalla signaalien suodatuksesta turhia komponentteja, mutta tämän työn aikana en ehtinyt tutkia, mitkä niistä olivat tarpeellisia.

Testiohjelmiston virranmittaukseen ja kierrosmittaukseen tulisi lisätä hälytysrajat. Ne voisi tällöin asettaa käyttöliittymästä testin käynnistyksen yhteydessä. Hälytysten avulla testidatan analysointi olisi nopeampaa mahdollisten poikkeamien osalta. Toinen vaihtoehto olisi tehdä skripti, joka kävisi testidatan lävitse ja ilmoittaisi poikkeamat normaalista. Lisäksi ohjelmistoon voisi lisätä uusia testiprofiileja, jolloin laitteella voitaisiin tehdä On / Off -testauksen lisäksi muitakin testejä.

Työn aikana opin paljon. Koska en ollut aiemmin tehnyt piirilevysuunnittelua, piti minun opetella kaikki alusta alkaen. Olin yllättynyt, että laite toimi heti ensimmäisestä prototyypistä lähtien, vaikka piirilevyssä oli yli 250 komponenttia. Ohjelmoinnissa en ollut aikaisemmin tehnyt web-pohjaista käyttöliittymää, mutta sen toteutus osoittautui helpoksi ja internetistä löytyi siihen paljon apua.

LÄHTEET

1. Outdoor 12V fan Specification. 2015. Nokia Oyj. Sisäinen dokumentti.

2. Outdoor 48V fan Specification. 2012. Nokia Oyj. Sisäinen dokumentti

3. BarrGroup 2001. Introduction Pulse Width Modulation. Saatavissa: https://barrgroup.com/Embedded-Systems/How-To/PWM-Pulse-Width-Modulation Hakupäivä: 17.11.2018

4. AnalogDialogue 2004. Why and How to Control Fan Speed for Cooling Electronic Equipment by Mary Burke. Saatavissa: https://www.analog.com/en/analog-dialogue/articles/how-to-control-fan-speed.html Hakupäivä: 17.11.2018

5. Digikey. Methods of Monitoring Fan Performance. Saatavissa: https://www.digikey.com/Web%20Export/Supplier%20Content/ComairRotron_25/PDF/Comair_CM_MonitoringFanPerformance.pdf Hakupäivä: 17.11.2018

6. Raspberry Pi. Specifications. Saatavissa: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/ Hakupäivä: 17.11.2018

7. Raspberry Pi. Python. Saatavissa: https://www.raspberrypi.org/documentation/usage/python/ Hakupäivä: 17.11.2018

8. STMicroelectronics 2015. STM32F103C8-datalehti. Saatavissa: https://www.st.com/resource/en/datasheet/stm32f103c8.pdf Hakupäivä: 17.11.2018

9. Arduino STM32. Saatavissa: https://github.com/rogerclarkmelbourne/Arduino_STM32 Hakupäivä: 17.11.2018

10. NXP Semiconductors 2015. PCA9685-datalehti. Saatavissa: https://www.nxp.com/docs/en/data-sheet/PCA9685.pdf Hakupäivä: 17.11.2018

11. Resistor Guide. Shunt resistor. Saatavissa: http://www.resistor-guide.com/shunt-resistor/ Hakupäivä: 17.11.2018

12. Analog Devices 2016. ADG706/ADG707-datalehti. Saatavissa: https://www.analog.com/media/en/technical-documentation/data-sheets/adg706_707.pdf Hakupäivä: 17.11.2018

13. Texas Instruments 2018. ADS1115-datalehti. Saatavissa: http://www.ti.com/lit/ds/symlink/ads1115.pdf Hakupäivä: 17.11.2018

14. Aumala, Olli 2006. Mittaustekniikan perusteet. Helsinki: Otatieto Oy

15. Maxim Integrated 2015. MAX31855-datalehti. Saatavissa: https://datasheets.maximintegrated.com/en/ds/MAX31855.pdf Hakupäivä: 17.11.2018

16. NXP Semiconductors 2015. 74HC14/74HCT14-datalehti. Saatavissa: https://assets.nexperia.com/documents/data-sheet/74HC_HCT14.pdf Hakupäivä: 17.11.2018

17. NXP Semiconductors 2012. BC846-datalehti. Saatavissa: https://assets.nexperia.com/documents/data-sheet/BC846_SER.pdf Hakupäivä: 17.11.2018

18. KiCad. About KiCad Saatavissa: http://kicad-pcb.org/about/kicad/ Hakupäivä: 17.11.2018

19. Sqlite. Faq. Saatavissa: https://www.sqlite.org/faq.html#q5 Hakupäivä: 17.11.2018

20. Python Wiki. Web Frameworks for Python. Saatavissa: https://wiki.python.org/moin/WebFrameworks Hakupäivä: 17.11.2018

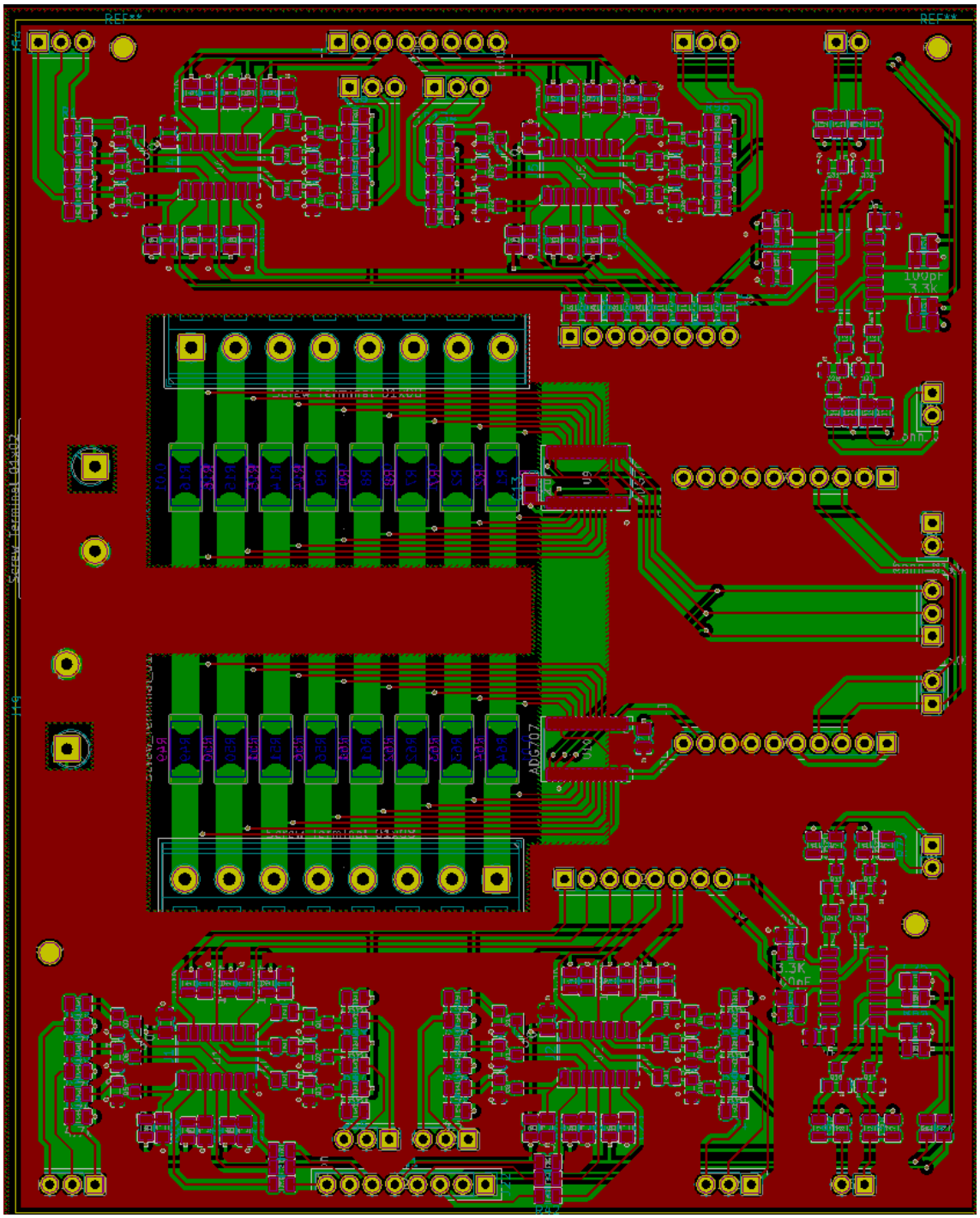
21. Flask. Foreword. Saatavissa: http://flask.pocoo.org/docs/1.0/foreword/ Hakupäivä: 17.11.2018

22. Bootstrap 2018. Getting Started. Saatavissa: https://getbootstrap.com/docs/4.0/getting-started/introduction/ Hakupäivä: 17.11.2018

23. W3Schools. Jquery intro. Saatavissa: https://www.w3schools.com/jquery/jquery_intro.asp Hakupäivä: 17.11.2018

LIITTEET

Liite 1 Piirilevy


REF**
REF**
Screw Terminal 01x02
ADG707
100pF
3.3k
3.3K
10nF