OULUN AMMATTIKORKEAKOULU

Lauri Aunio

TUOTANNONTESTAUKSEN PARANTAMINEN IoT-LAITTEELLE

TUOTANNONTESTAUKSEN PARANTAMINEN IoT-LAITTEELLE

Lauri Aunio
Opinnäytetyö
Syksy 2022
Tietotekniikan tutkinto-ohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietotekniikan tutkinto-ohjelma, ohjelmistokehityksen suuntautumisvaihtoehto

---

Tekijä: Lauri Aunio
Opinnäytetyön nimi: Tuotannontestauksen parantaminen IoT-laitteelle
Työn ohjaajat: Janne Kumpuoja OAMK, Kimmo Korhonen Bittium Oy
Työn valmistumislukukausi ja -vuosi: Syksy 2022 Sivumäärä: 32

Opinnäytetyön tavoitteena oli parantaa ja kehittää tuotannontestausta Bittium Oy:n IoT-laitteelle. Tätä aloittaessa työn kohteena olevan laitteen tuotannontestaus oli muutamia puutteita lukuun ottamatta valmis ensimmäisiin tuotantosarjoihin. Työn aikana tuotannontestausjärjestelmää päästiin koeponnistamaan tuotantolinjalla, josta saatuja tuloksia huomiotiin kehitystyössä.

Tuotannontestaus on laadunvalvontatoimintaa, jolla halutaan varmistaa valmistettavan tuotteen virheettömyys tuotannon eri valmistusvaiheissa sekä ennen luovuttamista loppukäyttäjälle. Opinnäytetyössä käydään myös katsaus ohjelmisto- ja tuotannontestauksen yleisistä menetelmistä.

Työssä syvennytään tarkemmin ledi- ja summeritestien tapauksiin piirilevyntestauksessa. Tavoitteena on saavuttaa testeille massatuotanto kelpoisuus parantamalla testien suorituskykyä sekä luotettavuutta. Työn tuloksena saatiin parannettua testit massatuotantokelpoisiksi.

Asiasanat: Python, Robot Framework, tuotannontestaus, testiautomaatio

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Information Technology, Option of Software Development

---

Author: Lauri Aunio
Title of thesis: Tuotannontestauksen parantaminen IoT-laitteelle
Supervisors: Janne Kumpuoja OAMK, Kimmo Korhonen Bittium Oy
Term and year when the thesis was submitted: Autumn 2022
Number of pages: 32

The aim of the thesis was to improve and develop production testing for Bittium Oy's IoT device. When this started, the production testing of the device under work was ready for the first production series, except for a few shortcomings. During this work, the production testing system was put to the test on the production line, the results of which were considered in the development work.

Production testing is a quality control activity that aims to ensure the flawlessness of the manufactured product in the various stages of production and before handing over to the end user. The thesis also reviews the general methods of software and production testing.

In the work, we delve more deeply into the cases of LED and buzzer tests in circuit board testing. The goal is to achieve mass production eligibility for the tests by improving the performance and reliability of the tests. As a result of the work, the tests were improved to be suitable for mass production.

Keywords: Python, Robot Framework, Production testing, Test automation

SISÄLLYS

1 JOHDANTO

Työn tarkoituksena oli parantaa tuotannontestausta Bittium Oy:n uudelle IoT-laitteelle. Tätä aloittaessa työn kohteena olevan laitteen tuotannontestaus oli muutamia puutteita lukuun ottamatta valmis ensimmäisiin tuotantosarjoihin. Tuotannontestauksessa erityisesti laitteen ledien ja summerin testaus oli puutteellinen. Tässä työssä ei käsitellä tuotteen koko tuotannontestausprosessia vaan keskitytään piirilevyntestaukseen sekä syvennytään tarkemmin ledi- ja summeritestien tapauksiin. Työn tavoitteena on parantaa näiden testien suorituskykyä sekä luotettavuutta, jotta testausjärjestelmä saavuttaa tuotantokelpoisuuden.

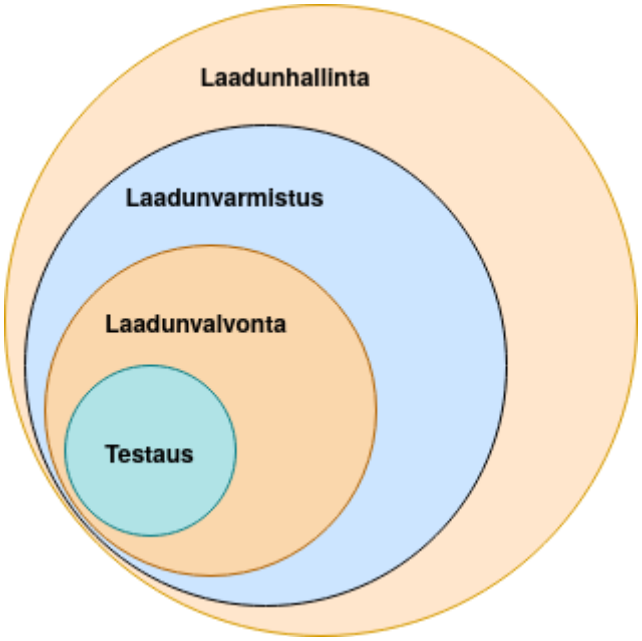
Työ jakaantuu karkeasti kahteen osaan. Ensimmäisessä osassa käsitellään laadunhallintaa ja testauksen teoriaan. Toisessa osassa käsitellään työn kohteena olevaa testausjärjestelmää sekä itse työn käytännön toteutusta. Käytännön kehitystyön aikana tuotannontestausta suoritettiin laitteen esituotantosarjaan tuotantolinjalla, josta saatiin havaintoja testitapausten kehittämiselle sekä testiaseman käytettävyyden parantamiselle.

2 LAATUJÄRJESTELMÄ

Tämän työn aihe keskittyy tuotannontestaukseen, mutta ensin tarkastellaan testauksen tarpeellisuutta laadun näkökulmasta. Testaus on yksi laadunvalvonnan osa-alue laadunhallinnassa, jonka hyöty nähdään tuotteen laadussa, kustannustehokkuudessa ja lopulta asiakastyytyväisyydessä. Laatu-käsitteellä on erilaisia määritelmiä, mutta tuotteen laadun lopullisena mittarina voidaan pitää vaikkapa virheettömyyden ansiosta saavutettua asiakastyytyväisyyttä. Tuotetta tekevän yrityksen näkökulmasta pelkästään korkean laadun tavoitteleminen ei riitä. Täydellisyyteen pyrkivä laadukkuus voi tulla yritykselle lopulta taloudellisesti kannattamattomaksi saatuun hyötyyn nähden, jolloin laatutason saavuttaminen tulee hoitaa tehokkaasti. (Haikala & Märijärvi 2004, 192.)

2.1 Laadunhallinta

Laadun varmistamiseksi ja parantamiseksi käytetään laadunhallintajärjestelmää, jonka eri osa-alueita on esitelty ASQ:n (American Society for Quality) mukaan kuvassa 1. Laadunhallintajärjestelmällä tarkoitetaan joukkoa työkaluja ja prosesseja, jotka varmistavat tuotteen vastaavan haluttua laatua. (American Society for Quality, 2022a.)



KUVA 1. Testauksen suhde laadunhallintajärjestelmässä (American Society for Quality 2022a.)

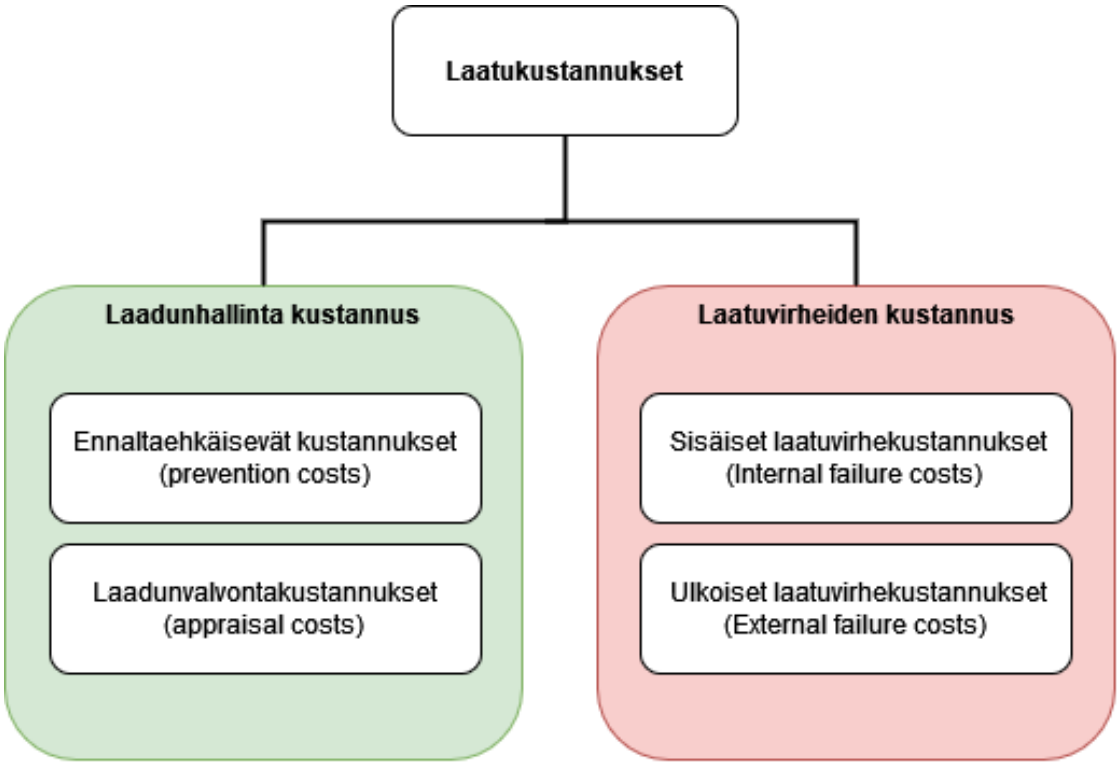
Laadunvarmistus keskittyy laadunhallinnan prosessien laatuun ja ongelmien ennaltaehkäisyyn. Se määrittelee käytännön laadunvalvonnalle laatu- ja tarkastussuunnitelmat sekä testaukselle vaatimusmäärittelyn. (American Society for Quality 2022a.)

Laadunvalvonta keskittyy laatuvaatimusten täyttymiseen operatiivisella tasolla. Laadunvalvonta tekee käytännön tarkistusta, joka tarjoaa tiedot ja mittarit laadunvarmistukselle. Testaus on laadunvarmistuksen määrittelemä prosessi, jota laadunvalvonta toteuttaa käytännössä. Tässä työssä keskitytään ainoastaan laadunvalvonnan osaan laadunhallinnan kokonaisuudessa. (American Society for Quality 2022a.)

2.2 Laadun kustannukset

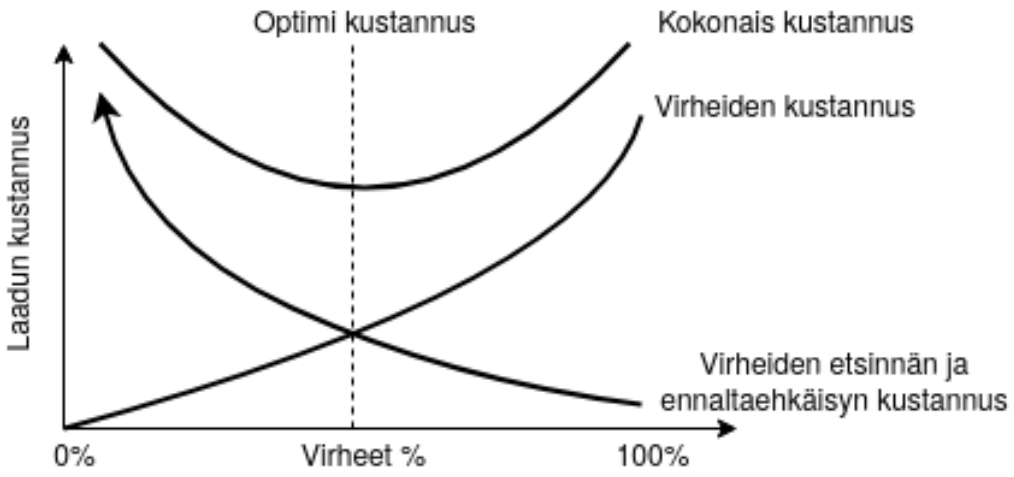
Laadukkaan työn tekemisen, laadun parantamisen ja tavoitteiden saavuttamisen kustannuksia olisi hyvä hallita laadun kustannusanalyysin (COQ, Cost of Quality) kautta. Tällainen analyysi tarjoaa menetelmän laadunhallinnan tehokkuuden arvioimiseksi ja keinon määrittää ongelmat sekä priorisoida toimintaa. (American Society for Quality 2022b.)

Kuvassa 2 olevan ASQ:n COQ mallin mukaan laatukustannukset jakaantuvat ennaltaehkäiseviin kustannuksiin, laadunvalvontakustannuksiin sekä sisäisten ja ulkoisten laatuvirheiden kustannuksiin.



KUVA 2. Laatukustannusten muodostuminen (American Society for Quality 2022b.)

Tuotannon laatutason ja laatukustannusten suhdetta voidaan kuvata kuvan 3 tavalla. Kuvaajasta nähdään, että virhetason laskiessa vikakustannukset pienenevät ja ennaltaehkäisyn kulut nousevat. Laadunhallinnan kustannusten ja vikakustannusten summasta saadaan tietyllä virhetasolla laadun kokonaiskustannusten kuvaaja. Tämä laadunkustannuskäyrä osoittaa, että kokonaiskustannus laskee virhemäärän laskiessa tiettyyn tasoon asti ja alkaa sitten kasvaa virhemäärän vähentyessä. Tämän laadun kustannusanalyysin perusteella voidaan löytää laadulle optimaalinen taso, joka perustuu taloudellisiin tekijöihin. (Krishnamoorthi, K.S., Krishnamoorthi, V.R. & Pennathur, A. 2018, luku 1.7.2.)



KUVA 3. Virhemäärän ja virheiden ennaltaehkäisyn suhde kustannuksiin. (Krishnamoorthi ym. 2018, luku 1.7.2.)

3 TESTAUSMENETELMÄT

Yleisesti testauksella tarkoitetaan lähes mitä tahansa kokeilemista. Testausta voidaan pitää suunnitelmallisena virheiden etsintänä, joten olennainen osa ennen varsinaista testausta on testauksen suunnittelu. (Haikala & Märijärvi 2004, 284.)

3.1 Testauksen tasot

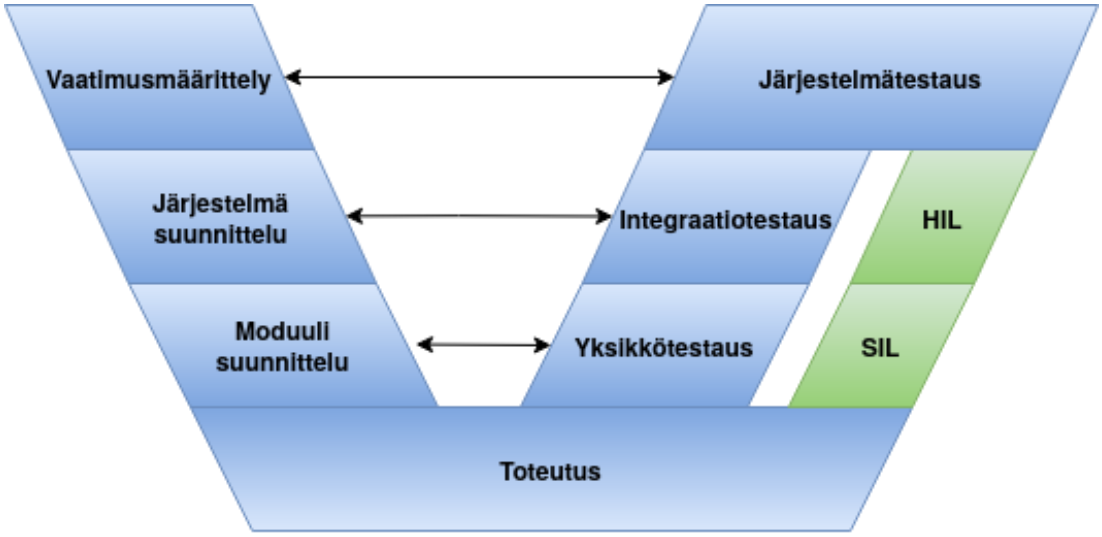
Kattavassa testauksessa on tarkasteltava ja testattava järjestelmää eri arkkitehtuurin tasoilla yksittäisestä ominaisuudesta aina kokonaiseen järjestelmään saakka. Testaustoiminnot, jotka liittyvät tiettyyn arkkitehtuuri tasoon, tunnetaan testaustasona. Ohjelmistotestauksen tasot voidaan jakaa karkeasti neljään eri tasoon, jotka ovat yksikkötestaus (unit testing), integrointitestaus (integration testing), järjestelmätestaus (system testing) ja hyväksyntätestaus (acceptance testing). (Haikala & Märijärvi 2004, 288.)

Yksikkötestaus (unit testing) on testauksen alin taso, joka testaa järjestelmän yksittäisen ominaisuuden tai toiminnon huomioimatta muuta kokonaisuutta. Nämä testit ovat rakenteeltaan yksinkertaisia ja helppoja hallita. Tyypillisesti testi antaa testattavalle menetelmälle tietyt parametrit ja vertaa tämän palautusarvoa odotettuun tulokseen. Yksikkötestit ovat nopeita suorittaa. Kattavien yksikkötestien jälkeen muutostöiden tekeminen on helpompaa, kun saadaan nopeasti palaute mahdollisista virheistä. (Aniche 2022, luku 1.4.4.) Integrointitestauksen (integration testing) tarkoituksena on testata järjestelmän komponentteja yhdessä keskittyen niiden välisiin riippuvuuksiin ja yhteyksiin. Järjestelmätestaus (system testing) testaa järjestelmää kokonaisuudessaan lopullisessa ympäristössään. Tyypillisesti nämä testit eivät välitä suoraan järjestelmän sisäisestä toiminnasta, vaan tarkoituksena on testata järjestelmälle annettujen vaatimusten toimintaa. (Aniche 2022, luku 1.4.) Hyväksyntätestauksen (acceptance testing) on testauksen viimeinen vaihe, jossa lopullinen tuote testataan. Tarkoituksena on löytää mahdolliset viat ja puutteet, jotka eivät täytä alkuperäisiä vaatimuksia. (Haikala & Märijärvi 2004, 290.)

Ohjelmistotestauksen säännöt soveltuvat myös sulautettujen järjestelmien testausprosesseihin. Sulautettujen järjestelmien testauksessa on kuitenkin otettava huomioon tietyt erityisvaatimukset, kuten testausympäristö. Joskus sulautetun järjestelmän kehittäminen on aloitettava, ennen kuin

varsinainen laitteisto on saatavilla. Tällöin testausympäristö on luotava osittain tai kokonaan simuloimalla. (Haikala & Märijärvi 2004, 289.)

Sulautettujen järjestelmien testauksessa integraatiotestaus voidaan jakaa integraatiotestaukseen ilman laitteistoa (SIL) ja integraatiotestaukseen laitteiston kanssa (HIL). Software-In-The-Loop (SIL) -menetelmässä integrointitestaaminen ja validointi tehdään simuloidulla laitteistolla. Hardware-In-The-Loop (HIL) -menetelmässä integrointitestauksessa on mukana todellinen laitteisto. (Kuva 4.)

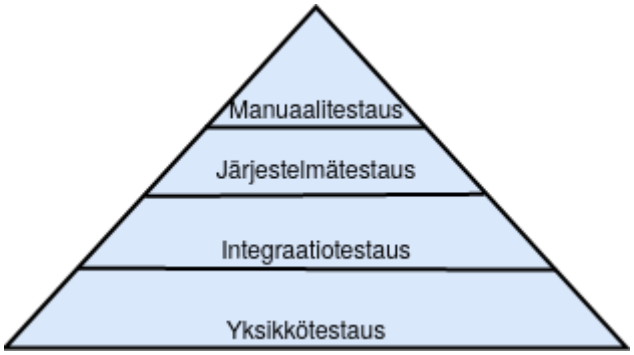


KUVA 4. Testauksen V-malli. (Haikala & Märijärvi 2004, 289, mukaillen.)

3.2 Testausstrategia

Kun eri testaustasojen vahvuudet tiedetään, voidaan analysoida, mitä testaustasoja on järkevä painottaa. Testien ajaminen vain lopullisessa ympäristössään eli järjestelmätason testaus paljastaa kyllä yleensä lähes kaikki virheet, mutta pelkästään tuolla tasolla testaaminen ei ole järkevää. Jos virheen korjauksen tai muutostöiden jälkeen joudutaan testaus suorittamaan vain lopullisessa toimintaympäristössä, on testaaminen vaivalloista, hidasta ja kallista. Tämän vuoksi järjestelmää halutaan testata alemmilla tasoilla, kuten integraatio- ja yksikkö -testeillä. (Aniche 2022, luku 1.4.)

Kuvassa 5 on yksi melko yleinen näkemys testaustasojen painotuksessa. Yksikkötestejä halutaan painottaa niiden helppouden ja nopeuden takia. Pyramidin lohkon koko kuvastaa testaustason haluttua kattavuutta. Matalan tason testaus on halvempaa ja nopeampaa kuin korkean tason järjestelmätestaus. (Aniche 2022, luku 1.4.4.)



KUVA 5. Testipyramidi (Aniche 2022, luku 1.4.4.)

3.3 Tuotannontestaus

Tuotannontestaus on laadunvalvontatoimintaa, jolla halutaan varmistaa valmistettavan tuotteen virheettömyys tuotannon eri valmistusvaiheissa sekä ennen luovuttamista loppukäyttäjälle. Tuotannontestaus on laadultaan järjestelmätestausta. Kuten aiemmin kuvassa 4 esiteltiin vaatimusmäärittelyn ohjaavan järjestelmätestausta, myös tuotannontestaus tehdään määriteltyjä vaatimuksia vasten.

Laitevalmistuksen yhteydessä tehtävällä laadunvalvonnalla on kustannusten kannalta merkittävää havaita mahdolliset virheet mahdollisimman aikaisin. Erityisesti laitevalmistuksessa ennaltaehkäisevät toimet kuten kattava testaaminen ovat huomattavasti edullisempia kuin mahdolliset laitteiden takaisinvedot asiakkailta. (American Society for Quality 2022b.)

Testaus pyritään suorittamaan mahdollisuuksien mukaan mahdollisimman automatisoidusti. Automatisoinnilla pyritään paitsi nopeuteen myös testauksen toistettavuuteen ja tarkkuuteen. Tuotannontestausta varten laadunvarmistusprosessit määrittelee tarkastussuunnitelmat ja testauksen vaatimusmäärittelyn eli testispesifikaation. Testauksessa havaittu virhe on poikkeama spesifikaatiosta, joten ilman spesifikaatiota testauksen oikeellisuutta ei voida varmistaa. (Haikala & Märijärvi 2004, 283–285.)

Tuotevalmistuksessa tuotannontestaus ei keskity tuotteen ei-toiminnallisiin ominaisuuksiin, kuten suorituskykyyn, laatuun tai käytettävyyteen. Tuotannontestauksen kohteena on itse laite ja sen toiminnan varmistaminen, kuten valmistetun piirilevyn testaus. Tyypillisiä piirilevyjen tuotannossa ilmeneviä vikoja voivat olla seuraavat:

- komponentti puuttuu
- komponentti väärinpäin
- oikosulut
- kylmäjuotokset
- rikkinäinen komponentti.
  (Proto-electronics 2022.)

Kuvassa 6 on esimerkki piirilevylle tehtävistä testausmenetelmistä. AOI (Automated optical inspection) ja AXI (Automated X-ray inspection) ovat nopeita testausmenetelmiä, jotka tehdään heti piirilevyn valmistuksen jälkeen. AOI perustuu konenäön avulla tehtävään havainnointiin ja sen avulla voidaan löytää esimerkiksi mahdolliset komponenttipuutteet. AXI perustuu röntgenkuvausmenetelmään, jonka avulla on mahdollista tarkastella komponenttipuutteiden lisäksi myös juotosten laatua. (Proto-electronics 2022.)

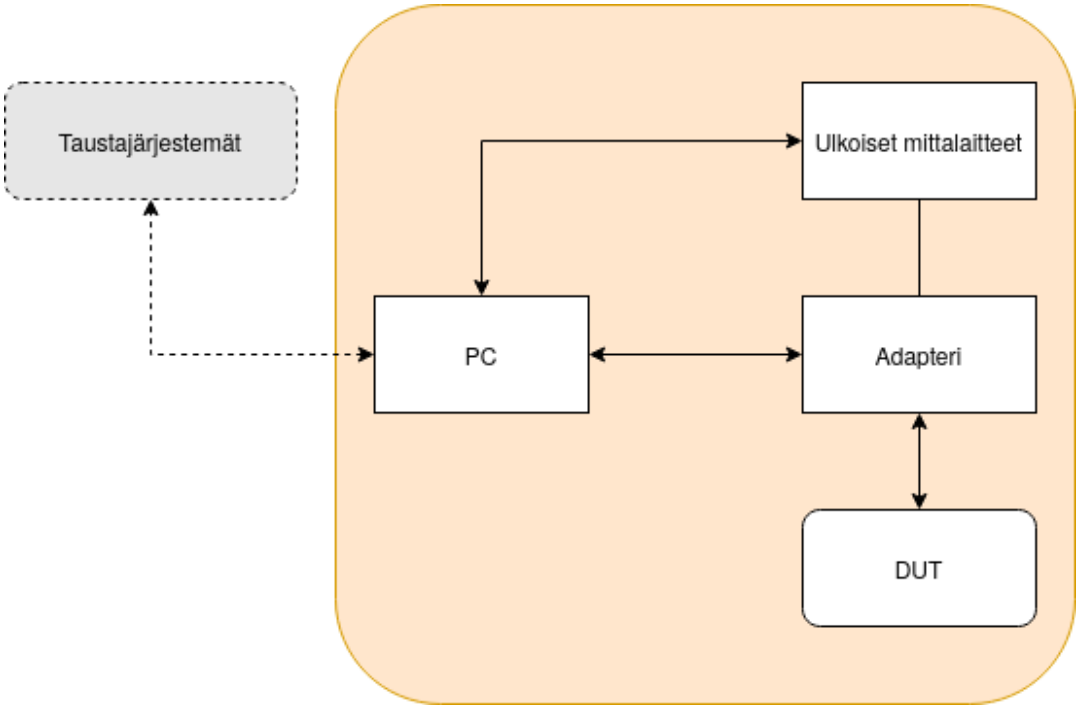
KUVA 6. Tyypilliset piirilevyn testauksessa käytettävät menetelmät

4 TESTAUSJÄRJESTELMÄ

Tässä luvussa esitellään testausjärjestelmää ja testiautomaatiota yleisesti sekä tässä työssä jo käytössä olevia testausjärjestelmän ratkaisuja.

4.1 Testausjärjestelmän rakenne

Testausjärjestelmä käsittää kaiken sen ympäristön, mitä testausvaiheen suorittamiseen vaaditaan (ISTQB Glossary, hakusana testing environment). Kuvassa 7 kuvataan tyypillistä testausjärjestelmää. Testattava DUT (Device Under Testing) on yhdistetty välikappaleen kautta testausympäristöön. Testattavan laitteen tyypistä riippuen välikappale eli testijigi ja mittalaitteet vaihtelevat. Testiympäristön PC:n tehtävä on suorittaa testisarjat DUT:lle sekä raportoida saadut tulokset taustajärjestelmälle, joka huolehtii tietojen tallettamisesta.



KUVA 7. Yleiskuvaus tyypillisestä testausjärjestelmästä

4.2 Testausohjelmisto laitteelle

Tuotannontestausta varten laitteelle ladataan erityinen tuotannontestausohjelmisto (PTSW, Production Testing Software), joka antaa testausympäristölle pääsyn laitetason komponentteihin sekä

asettaa laitteen testauksessa haluttuun tilaan. PTSW tarjoaa testiympäristölle pääsyn laitteistoon komentorajapinnan kautta. Näin testiympäristö voi ohjata ja lukea laitteelta tietoa, jonka perusteella toimintaa voidaan arvioida laitteen ulkopuolelta.

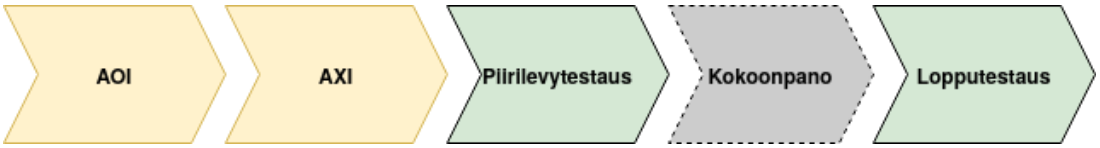
4.3 Testiautomaatio

Testausautomaatio on ohjelmisto, joka suorittaa tai tukee testaustoimintaa (ISTQB Glossary, Testing automation). Testausautomaatio nopeuttaa toistuvaa testaamista ja vapauttaa resursseja tärkeämpään toimintaan. Testitoimintojen automatisoinnilla saavutetaan suurin hyöty toistoja vaativissa testeissä. (Dustin, Rashka & Paul 1999, luku 1.)

Tuotevalmistuksessa yleensä pyritään valmistamaan tuote mahdollisimman tehokkaasti. Tuotannontestaus on osa tuotevalmistuksen prosessia, jossa tehokkuutta usein haetaan testiautomaatiosta. Testisautomaatio tuotannontestauksessa usein painottuu testisarjojen suorittamiseen. Tuotannontestauksen automatisoinnilla voidaan lisäksi parantaa testauksen tarkkuutta ehkäisemällä inhimillisiä virheitä tulosten tulkinnassa.

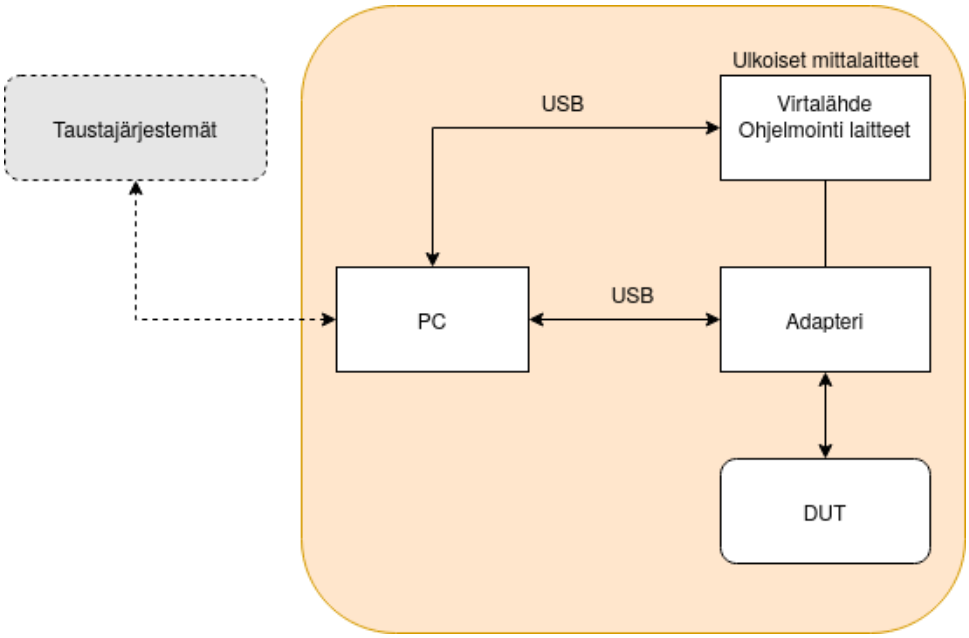
4.4 Testausjärjestelmän lähtötilanne

Tässä osiossa esitellään tuotannontestausjärjestelmää, johon on tässä työssä tarkoitus tehdä parannuksia. Tarkastelussa olevan IoT-laitteen tuotannontestaus on toteutettu kahdella erillisellä testausasemalla. Tätä laitetta koskeva tuotantokaavio näkyy kuvassa 8. Tässä työssä tarkastellaan erityisesti piirilevyn testausta.



KUVA 8. Laitteen tuotantokaavio

Kuvassa 9 on kuvattuna piirilevyn testausjärjestelmä, jonka laitteistopuoleen kuuluvat testi-PC, virtalähde, ohjelmointilaitteita ja adapteri eli testijigi.

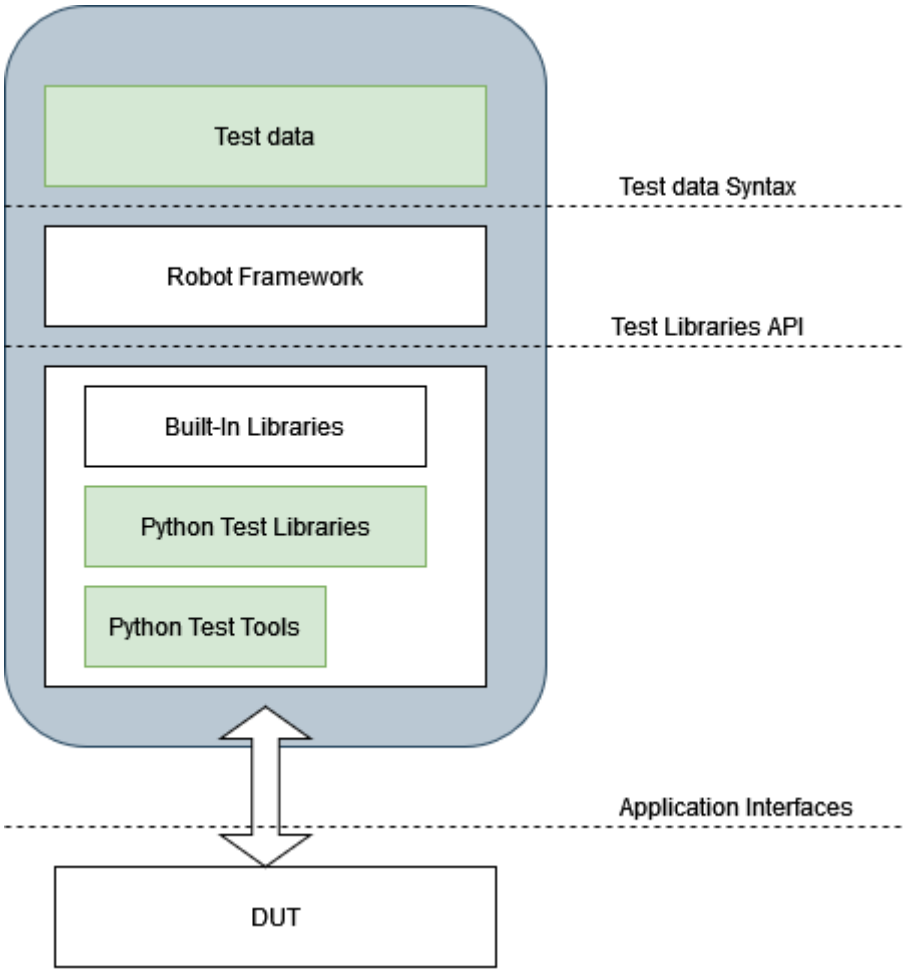


KUVA 9. Testausjärjestelmän yleiskuvaus

4.5 Käytössä oleva testiautomaatio

Testausjärjestelmä käyttää avainsanaohjattua Robot Framework -testiautomaatiokehystä. Robot Framework soveltuu käytettäväksi muun muassa hyväksyntätestauksessa ja robottiprosessiautomaatiossa. Robot Framework ei ole käyttöjärjestelmäriippuvainen ja sillä voidaan testata hyvin erityyppisiä järjestelmiä.

Kuvassa 10 on vihreällä pohjalla ne Robot Frameworkin osat, joihin testitapauksia kehittäessä käytännön työ kohdistuu. Suoritettavia testikokonaisuuksia kutsutaan testidataksi ja ne kirjoitetaan tekstitiedostoon Robot Frameworkin omalla syntaksilla. Varsinaiset testikirjastot tarjoavat Robot Frameworkin testausominaisuudet. Valmiita testikirjastoja löytyy paljon, mutta monimutkaisemmissa testausjärjestelmissä usein on tarpeen toteuttaa itse tarvittavat testikirjastot ja työkalut. Robot Framework itsessään on kirjoitettu Pythonilla, joten tarvittavat laajennukset on luonnollista tehdä samalla kielellä. (Robot Framework Foundation 2022.)



KUVA 10. Robot Frameworkin arkkitehtuurikuvaus (Robot Framework User Guide, 2022.)

5 TOTEUTUS

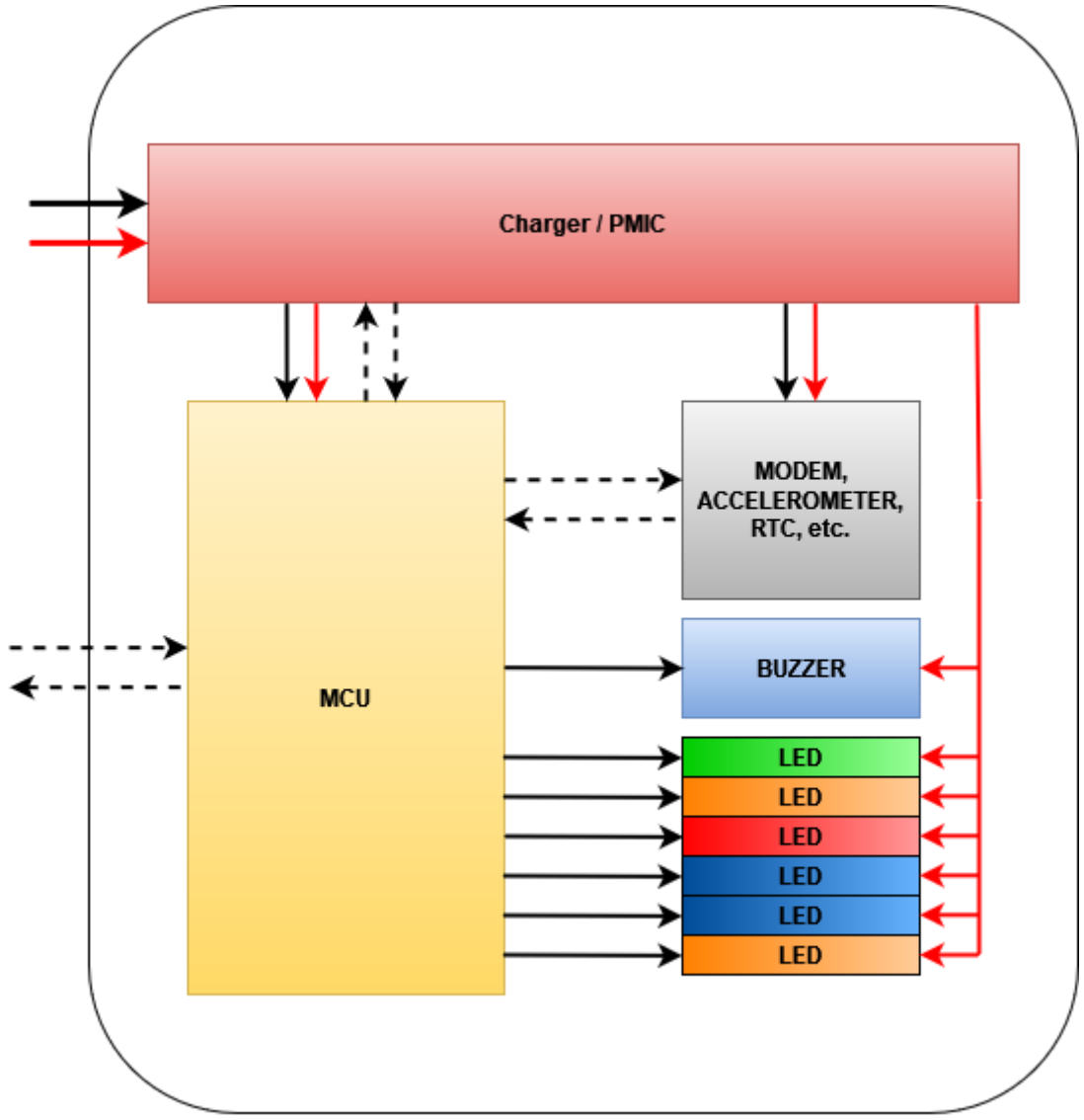
Tässä osiossa käydään läpi työn käytännön osuutta eli testien suunnittelua, toteutusta ja tuloksia. Työ toteutettiin osana tuotannontestausta kehittävän tiimin kanssa. Lähtötilanteessa lähes kaikki testit olivat automatisoituja ja tavoitteena oli saada loputkin testit mahdollisimman automatisoiduiksi. Tiedossa olevat parannustavoitteet piirilevyn testaukseen ovat ledi- ja summeritestien automatisointi. Näiden toimintojen testaamiseen on olemassa valmiina testit, mutta ne vaativat manuaalista tarkistusta ja vastausta testausoperaattorilta.

5.1 Testivaatimukset ja automatisointi

Tuotannon testauksessa pyritään suorittamaan mahdollisimman kattavat testit aikaisessa vaiheessa tuotannon testausprosessia, sillä mitä aikaisemmassa vaiheessa virheet löydetään, sitä vähemmän se aiheuttaa kustannuksia. Näiden tarkastelussa olevien ledien ja summerin testaus on tarkoitus suorittaa piirilevy- sekä lopputestauspaikoilla. Vaikka samat toiminnallisuudet testataan useampaan kertaan, ei piirilevyn testausta voida jättää tekemättä, ettei viallinen piirikortti pääse kokoonpanovaiheeseen. Ei voida myöskään olettaa, että jos toiminnallisuus on toiminut aikaisemmassa piirilevytestauksessa, se toimisi vielä kootulla laitteella, sillä laitekokoonpanon jälkeen voi tulla mekaniikasta tai kokoonpanosta johtuvia virheitä. Testien tulisi toimia itsenäisesti ilman testausoperaattoria vaativaa tarkistusta eli testien tulee olla testiautomaatiokelpoisia.

5.2 Testattava laitteisto

Kuvassa 11 on esiteltynä testattavan laitteen keskeiset osat, jossa kuvataan yksinkertaistettuna komponenttien kytkennät laitteella. Datalinjat on kuvattu katkoviivalla ja sähköiset kytkennät yhtenäisillä viivoilla. Testejä suunnitellessa on huomioitava, etteivät testit pääse vaikuttamaan toistensa toimintaan ja jokainen yksittäinen testi jättää laitteen edeltävään testiä edeltävään tilaan.



KUVA 11. Testattavan laitteen keskeiset osat

5.3 Testien rakenne

Suunnittelussa on pyritty selkeyteen ja yhdenmukaisuuteen testien rakenteessa. Testitapaukset noudattavat nelivaiheista (Four-Phase) testimenetelmää, jossa testit on rakennettu neljästä erillisestä osasta, jotka suoritetaan peräkkäin. Testin muodostamat vaiheet ovat alkuvalmistelut (Setup), toimeenpano (Exercise), tuloksen varmistus (Verify) ja testin purku (Teardown). (Meszaros 2007, luku 19.)

DUT:n ja ympäristön konfigurointi testin vaatimaan tilaan voi olla monivaiheinen prosessi, joka yhdistettynä testilogiikan kanssa voi olla vaikeaselkoinen luettavuudeltaan. Suunnittelemalla testit

selkeisiin osakokonaisuuksiin luettavuus ja uudelleenkäytettävyys paranevat. Testivaiheiden selkeällä erottamisella voidaan pitää itse testin logiikka selkeänä.

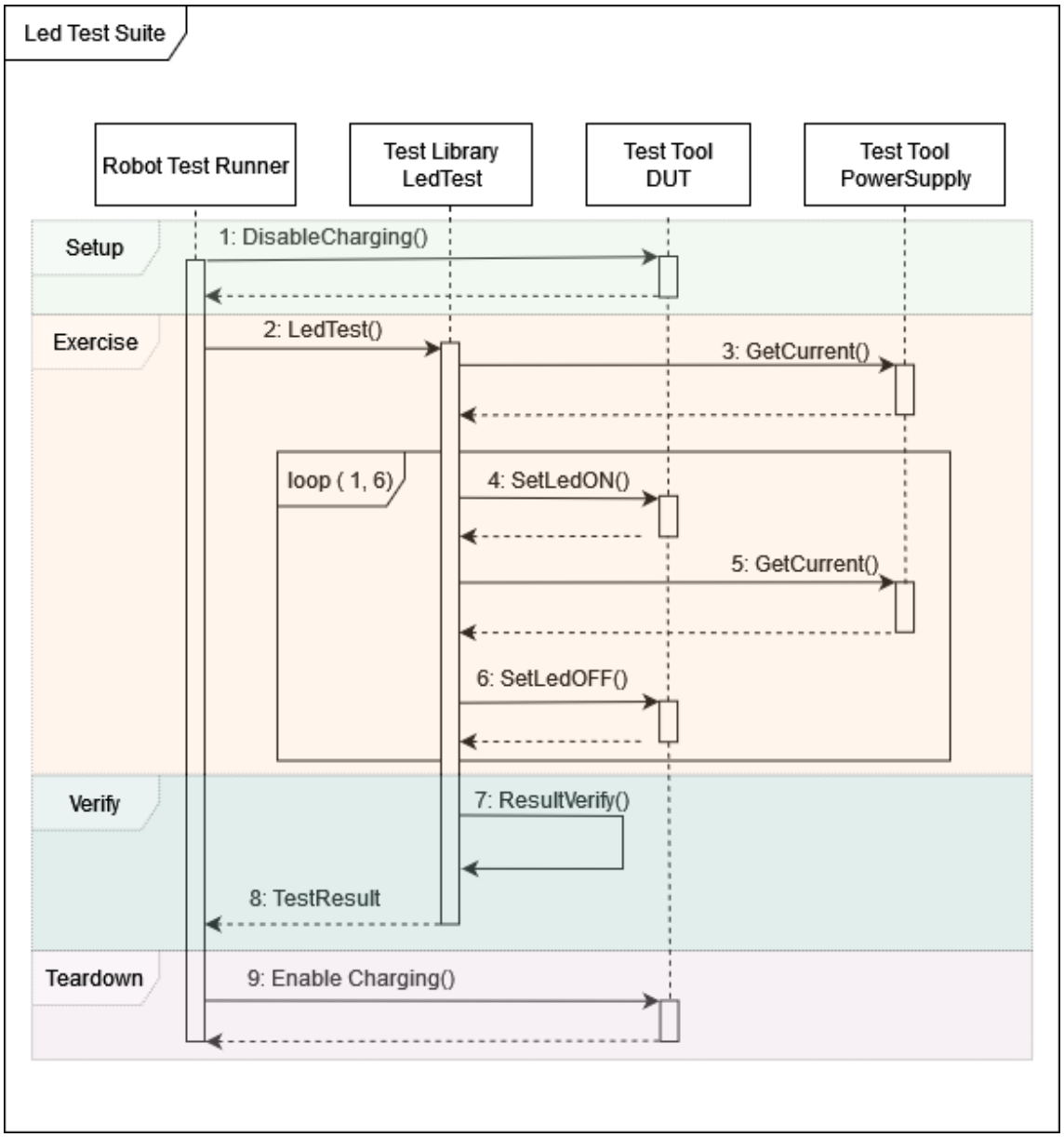
5.3.1 Ledien testaus

Ledien testausmenetelmäksi pohdittiin paria testausautomaatioon soveltuvaa vaihtoehtoa. Valovasteellinen mittalaite olisi ollut yksi mahdollinen vaihtoehto ledien tarkistukseen. Tätä vaihtoehtoa ei kovin pitkälle lähdetty suunnittelemaan, sillä mittaustekniikan järjestäminen piirilevytesterille olisi tarvinnut suuria mekaniikkamuutoksia. Piirilevytesterissä oli ennestään virtalähde tarkalla virranmittausominaisuudella, joten testausta virranmittausmenetelmällä selvitettiin. Todettiin, että virranmittausmenetelmä pystyy havaitsemaan virrankulutuksen muutoksen, kun ledi ohjataan päälle. Virrankulutusmenetelmän laskenta kaavassa (kaava1) I_{LED_OFF} kuvaa laitteen pohjavirrankulutusta ja I_{LED_ON} laitteen virrankulutusta ledin ollessa päällä. Testattavassa laitteessa on 6 lediä, joiden väri ja kirkkaus vaihtelee. Ledin tyypistä riippuen virrankulutus vaihtelee 0,7 mA:n ja 1,6 mA:n välillä. Virranmittausmenetelmä vaatii kuitenkin erityistä huomiota myös testattavan laitteen tilaan, jotta testattavan laitteen virrankulutus saadaan vakioitua tarpeeksi tasaiseksi. Testattavan komponentin kannalta tarpeettomat laitteet ja komponentit on tarpeen pitää sammutettuna tai vakiotilassa. Tässä on huolehdittava, että laitteen ylimääräiset toiminnot kuten modeemi on sammutettu.

$$I_{LED} = I_{LED\_ON} - I_{LED\_OFF}$$

KAAVA 1. Ledin virrankulutuksen laskenta kaava

Kuvassa 12 on kuvattu leditestin sekvenssiä sekä eri testivaiheiden tehtäviä. Kuvassa on myös eriteltyinä nelivaiheisen testimenetelmän osat. Robot Frameworkin ominaisuuksiin kuuluvat Suite Setup- ja Suite Teardown -toiminnot, joiden avulla voidaan määritellä ennen- ja jälkeen-toiminnot testisarjalle. Itse testin suoritus ja verifikaatio tapahtuu Python-testikirjastolla.

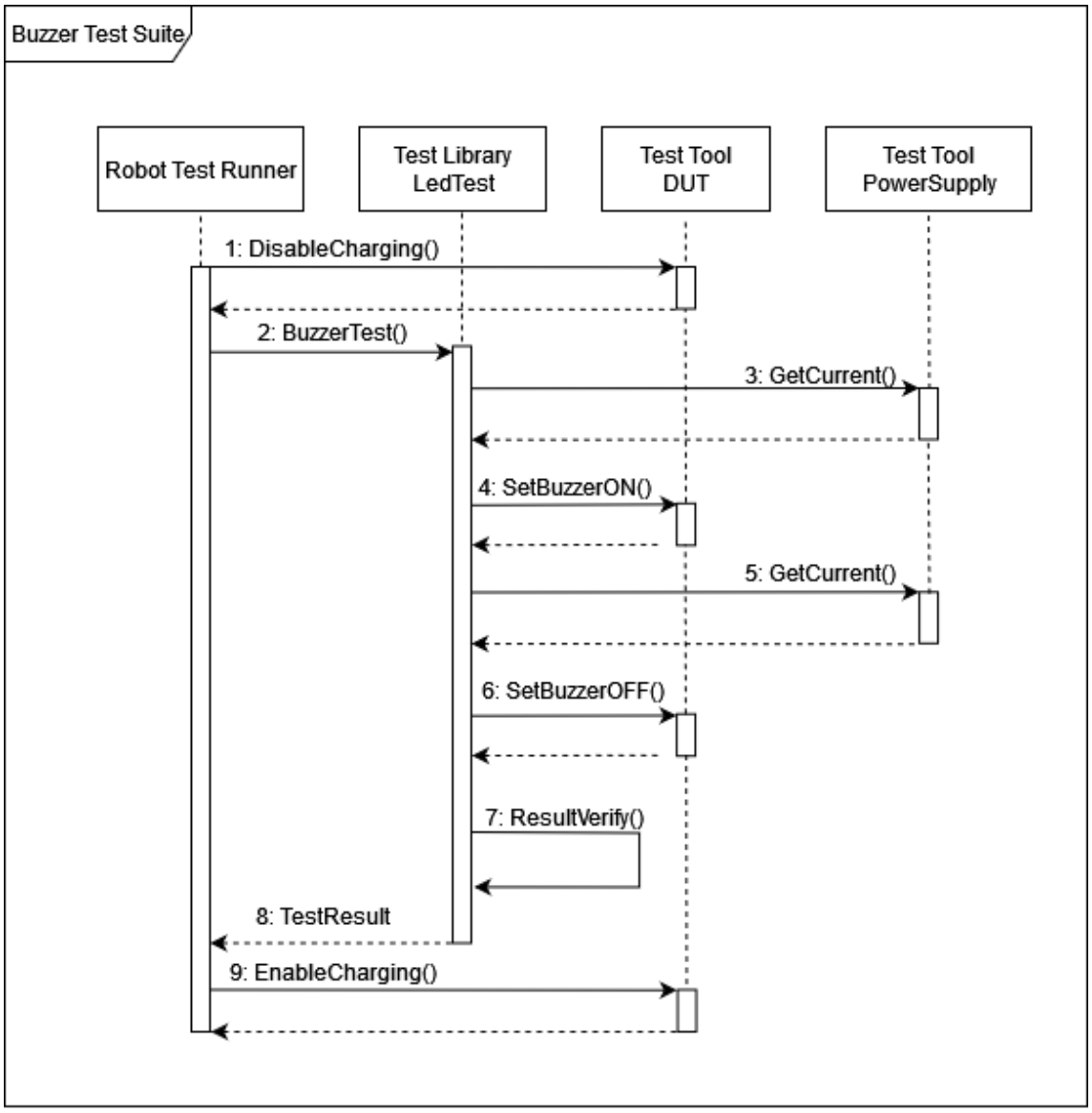


KUVA 12. Leditestin sekvenssikuvaus

5.3.2 Summerin testaus

Summerin testaus toteutettiin lopulta myös virran mittaukseen perustuen. Yksi vaihtoehto, jota myös kokeiltiin käytännössä, olisi ollut mikrofoniin perustuva menetelmä. Tässä tapauksessa virranmittausmenetelmä valikoitui järkevämmäksi testausasemasta löytyvän virranmittausominaisuuden perusteella. Summerin virrankulutus on noin 48 mA, joten toiminnan tulkinta on selkeä, kun laitteen taustavirrankulutus on noin 5 mA.

Kuvassa 13 summeritestin sekvenssi, joka on rakenteeltaan samankaltainen leditestien kanssa.

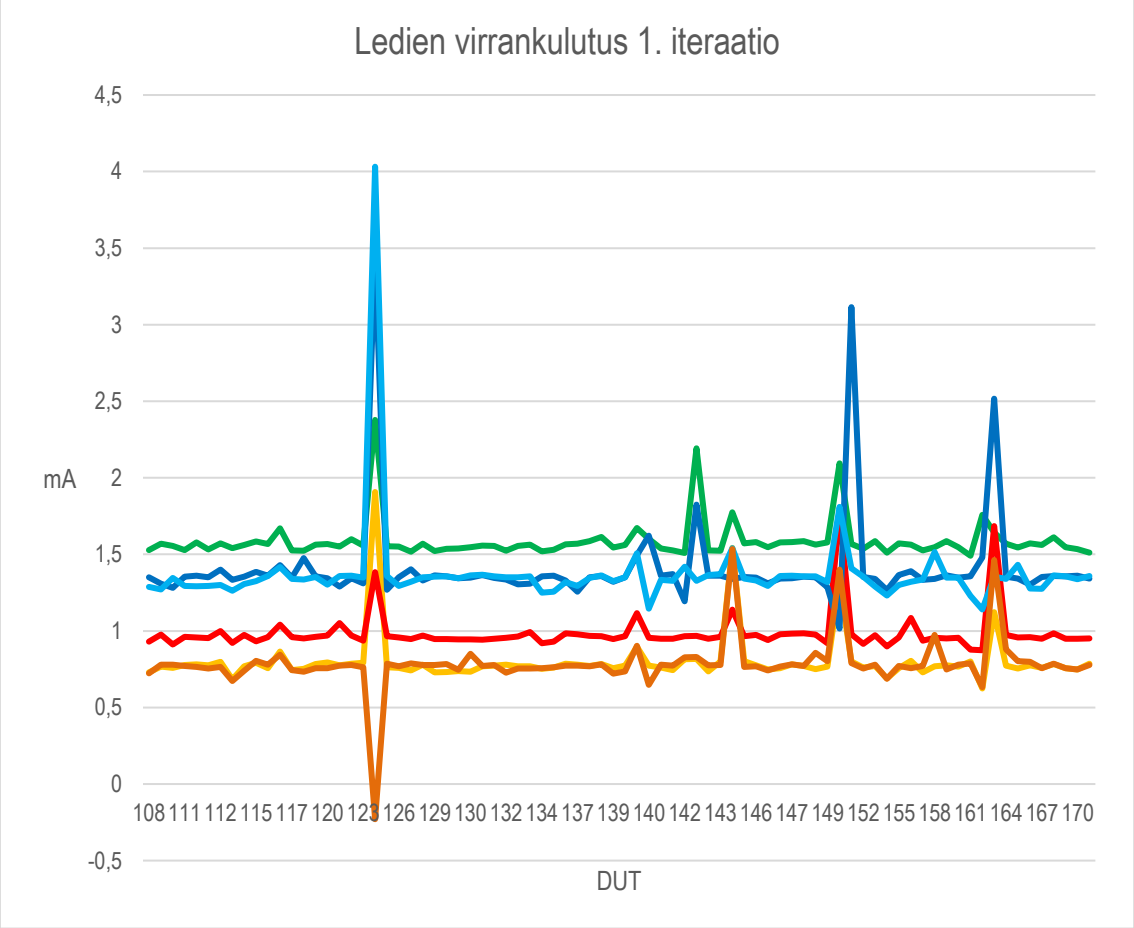


KUVA 13. Summeritestin sekvenssikuvaus

5.4 Ensimmäinen iteraatio

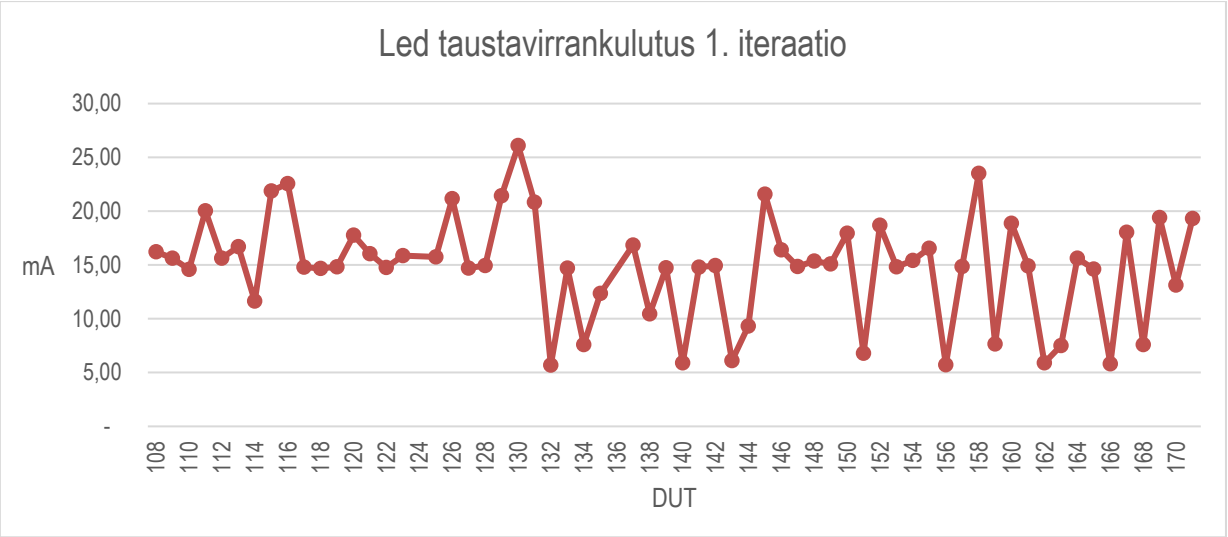
Testausmenetelmät saatiin toimivaksi kehityskäytössä olleilla muutamalla laitteella, mutta tehtaalla suuremman esisarjan testauksen aikana havaittiin ongelmia ledien testauksessa, jotka oireilivat virheellisinä testituloksina. Ongelmaa tutkiessa havaittiin laitteiden testituloksissa rajuja vaihteluja ledien virrankulutuksessa. Kuvassa 14 leditestin keräämiä virranmittaustuloksia, joista voidaan havaita virheelliset mittaukset. Virrankulutuksen kuvaajista voidaan nähdä, että yhden laitteen osalta

mitattu virrankulutuksen tulos on vaihdellut rajusti ja ollut jopa alhaisempi kuin mitattu virrankulutuksen vertailuarvo ledit sammutettuna. Tästä voidaan päätellä, että laitteen taustavirrankulutus ei ole ollut tasainen ja virrankulutus on päässyt vaihtelemaan testin aikana. Leditestin suoritusaika on kokonaisuudessaan noin sekunnin, joten virrankulutuksen vaihtelut ovat olleet nopeita. Noin 50 laitteen otannalla virheellisiä tuloksia löytyi 6 kappaletta, joten ongelma esiintyy toistuvasti. Näiden löydösten perusteella virheellisiä tuloksia antava leditesti ei ole tämmöisenään tuotantokäyttöön sopiva.



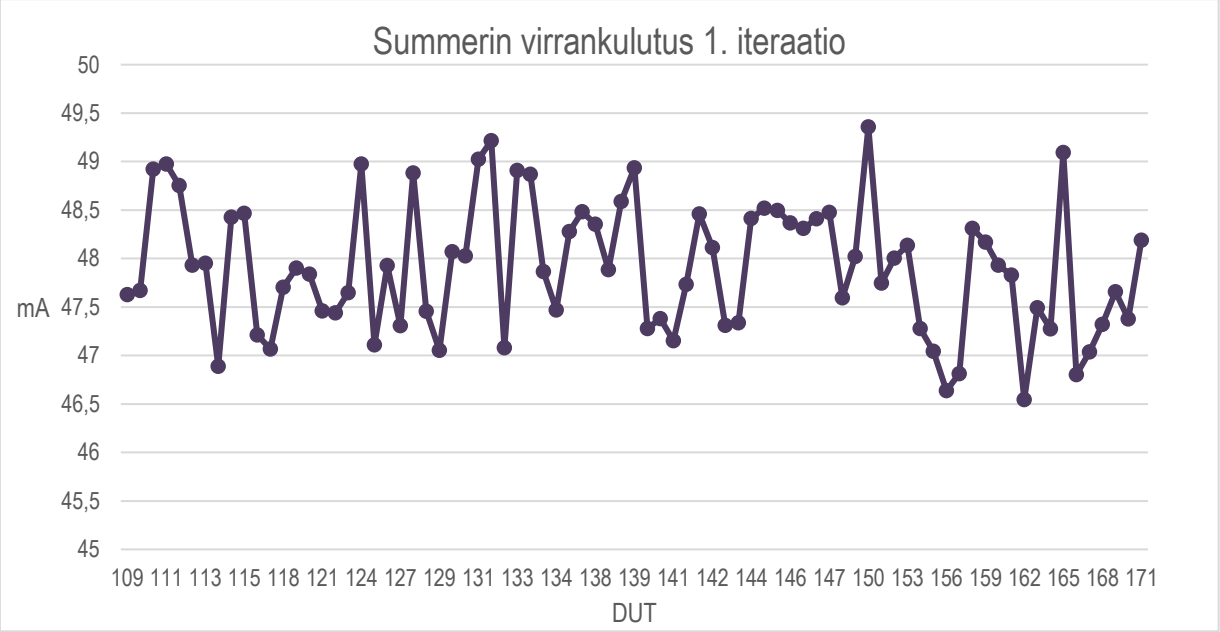
KUVA 14. Ensimmäisen iteraation ledien virrankulutusmittauksia

Ongelmaa selvitellessä havaittiin laitekohtaista vaihtelua testin mittaamissa pohjavirrankulutuksen mittauksessa. Kuvassa 15 leditestin mittaamia virrankulutuksia ledit sammutettuina. Kuvaajasta nähdään, että laitteiden virrankulutuksessa on suurta vaihtelua. Tästä voidaan päätellä että, laitteet eivät ole joka testikierroksella samassa vakioidussa tilassa.



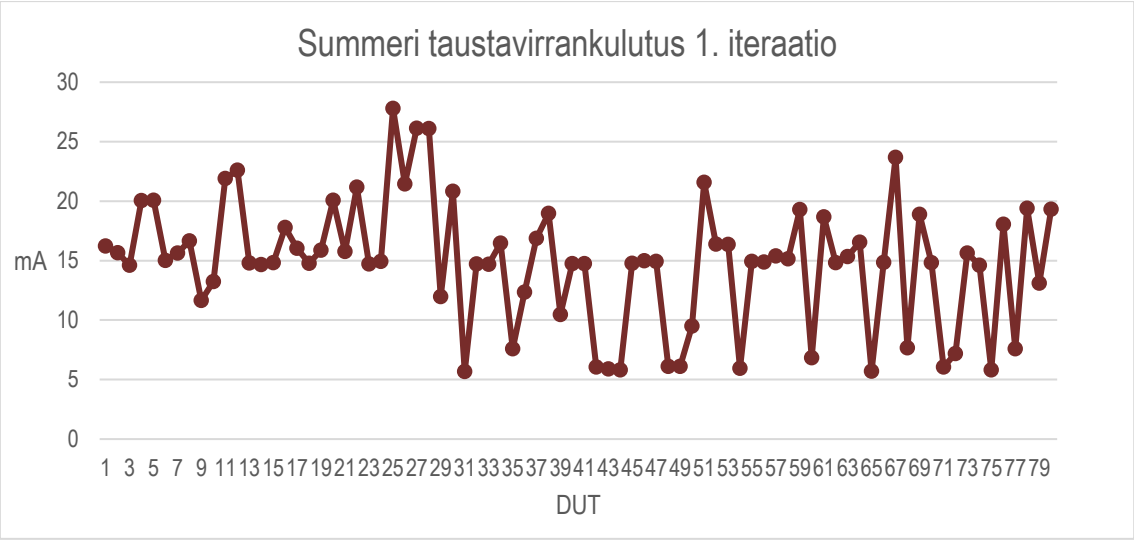
KUVA 15. Leditestin aikana mitatut virrankulutuksen vertailuarvot, kun virrankulutus on epästabiili

Summeritestin mittaustuloksista pystyttiin havaitsemaan sama heiluntailmiö kuin leditesteillä, mutta itse testi tulokseen sillä ei ole vaikutusta, koska summerin aiheuttama muutos virrankulutukseen on heiluntaa suurempi. Kuvassa 16 on summerin virranmittaustuloksia 1. iteraatiosta.



KUVA 16. Summerin virrankulutuksen mittaustuloksia 1. iteraatiosta

Kuvassa 17 summeritestin aikana mitattuja virrankulutuksen vertailuarvoja, joista havaittavissa heilumista vastaavalla vaihteluvälillä kuin leditesteillä.



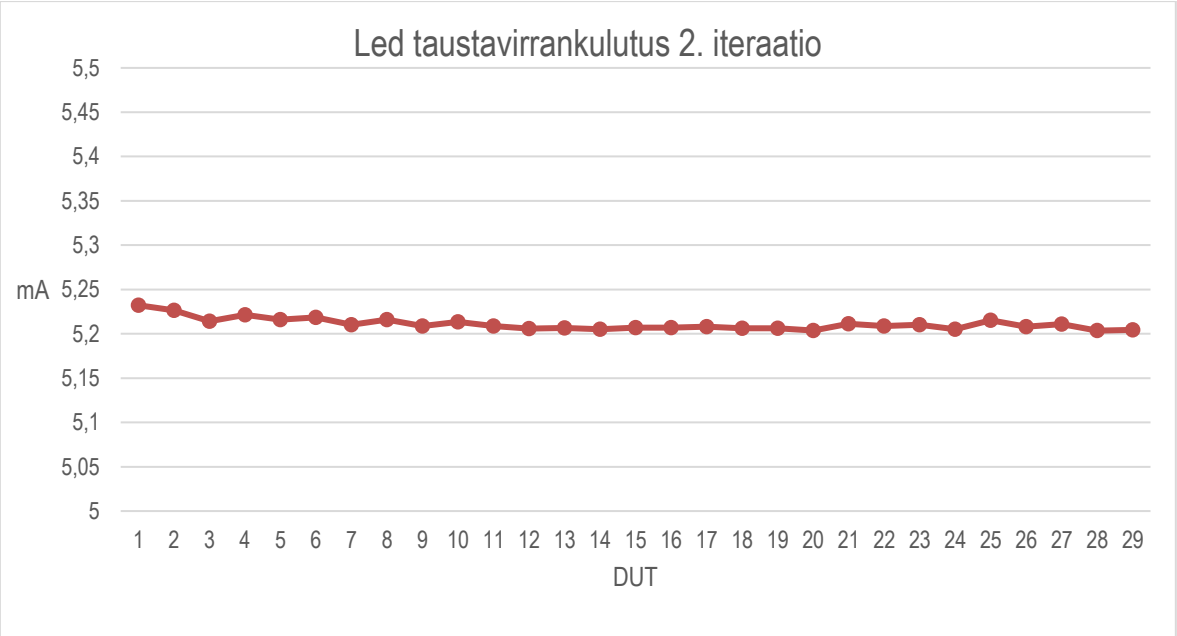
KUVA 17. Summeritestin aikana mitatut virrankulutuksen vertailuarvot, kun virrankulutus on epästabiili

Ongelmaksi havaittiin laitteen jäävän epästabiiliin tilaan edeltävien testien jälkeen. Epästabiili tila on seurausta puutteellisista testien purkutoimista, kun yksittäinen testi konfiguroi DUT:n tilaa ja jättää laitteen konfiguroimaansa tilaan siirtyessään seuraavaan testiin. Leditestien suoritus järjestys oli testiautomaation loppupäässä, jolloin edellä on suoritettu jo kymmeniä testejä. Testien suoritus järjestystä vaihtelemalla voitiin haarukoida puutteelliset testit. Hieman puutteellisia testejä löydettiin useampia.

Yleisin havaittu puute oli testeissä konfiguroidut gpio:t (general purpose I/O), joita yksittäisen testin jälkeen ei palautettu testiä edeltävään tilaan. Yksittäisen virheellisen gpio tilan aiheuttama vaikutus virrankulutukseen oli hyvin pieni, mutta useiden virheellisten gpio tilojen yhteisvaikutus oli kokonaisuudessaan noin 5 mA:n luokkaa. Suurimpana vaikutuksena virrankulutuksen vaihteluun löydettiin SD-kortista, joka on testauksen jälkeen jätetty päälle.

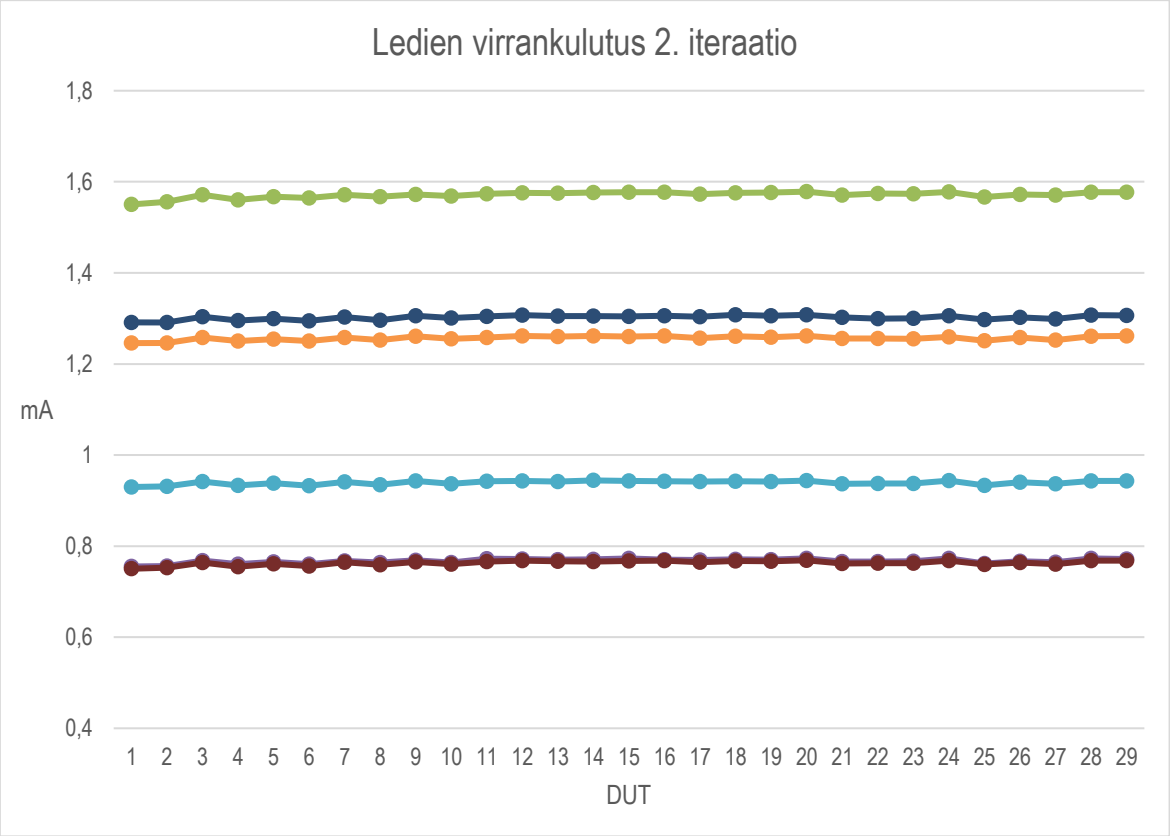
5.5 Toinen iteraatio

Vaikka ensimmäisen iteraation perusteella löydetyt puutteet saatiin enimmäkseen korjattua, päätettiin ledi ja summeritestien suoritus siirtää testiautomaation alkupäähän. Muutosten jälkeen ledi ja summeritestien mittaama taustavirrankulutuksen vertailuarvo tasoittui noin 5,2 mA:n tasoon. Kuvassa 18 leditestin mittaamia taustavirrankulutuksia ledit sammutettuna.



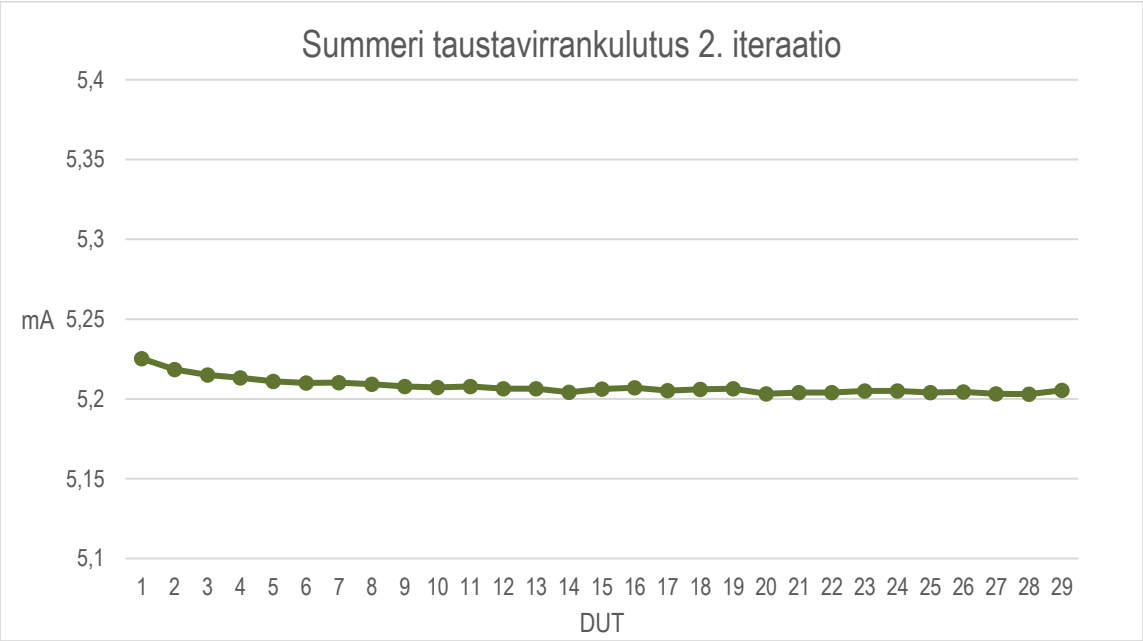
KUVA 18. Leditestin stabiloitunut taustavirrankulutus

Laitteen virrankulutuksen vakioiduttua tasaiseksi myös itse leditestin tulokset paranivat huomattavasti ja mittaustuloksista hävisi epämääräinen heilunta kokonaan. Kuvassa 19 testaus tuloksia korjausten jälkeen stabiililla virrankulutuksella.



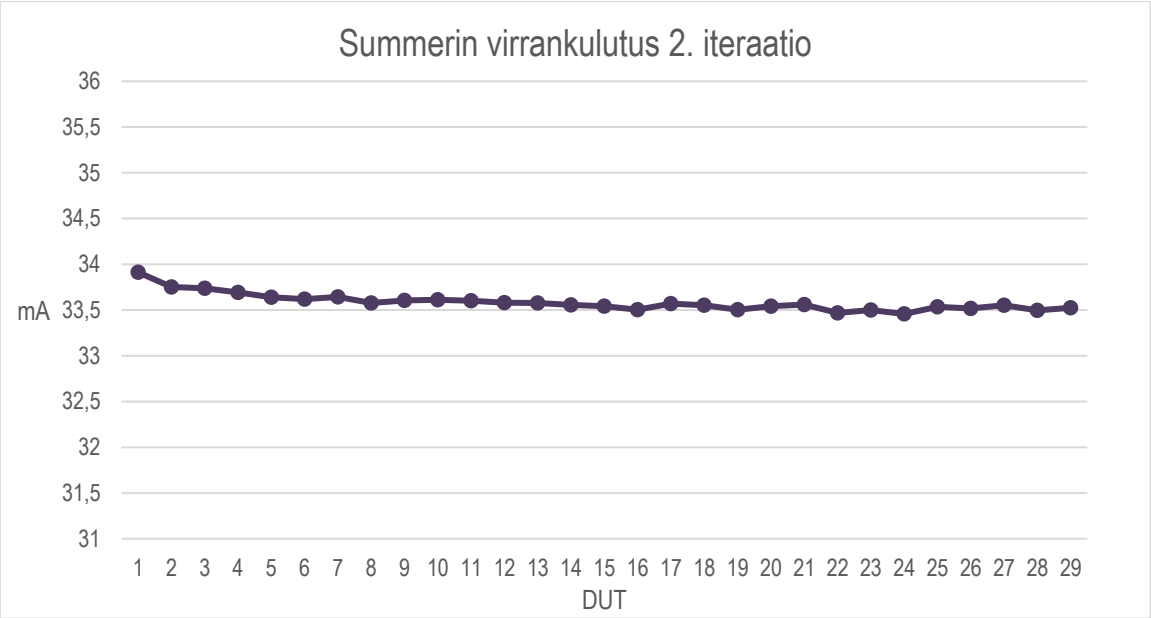
KUVA 19. Ledien virrankulutusmittauksia stabiililla pohjavirrankulutuksella

Kuvassa 20 summeritestin mittaama stabiloitunut laitteen taustavirrankulutus.



KUVA 20. Summeritestin stabiloitunut taustavirrankulutus

Summeritestin virranmittaus tulosten vaihtelu tasoittui myös huomattavasti. Kuvassa 21 summerin virrankulutus mittauksia stabiililla virrankulutuksella. Laitteen testausohjelmistoon tulleen muutoksen myötä summerille ajettavan PWM signaalin pulssisuhdetta muutettu, joten summerin virrankulutuksen taso on hieman laskenut verrattuna 1. iteraation mittauksille.



KUVA 21. Summerin virrankulutusmittauksia stabiililla pohjavirrankulutuksella

6 POHDINTA

Tämän opinnäytetyön tavoitteena oli parantaa Bittiumin uuden laitteen tuotannontestausta ja toteuttaa siitä puuttuvat ledien ja summerin testaukset massatuotantoon sopivaksi. Työn tavoitteet saavutettiin hyvin ja tuloksena saatiin testausautomaatioon soveltuvat testit, jotka ovat suorituskyvyltään nopeita ja luotettavia. Vaikka testausjärjestelmän käytössä esisarjan tuotannossa havaittiin ongelmia testituloksissa, oli itse suunniteltu testausmenettely varsin onnistunut. Ensimmäisen iteraation tuotannosta saatujen tulosten perusteella osattiin toteuttaa tarvittavat korjaukset testausjärjestelmään.

Ledi- ja summeritesteissä käytetty virranmittaukseen perustuva testausmenetelmä on suoritukseltaan nopea ja helppo toteuttaa, sillä se ei vaadi erillistä mittalaitetta testausadapteriin. Virranmittausmenetelmällä ei voida testata kuuluuko summerista ääntä, tuleeko ledistä valoa tai mikä on ledin väri. Testattavan komponentin virrankulutuksesta voidaan kuitenkin tietyllä tasolla päätellä komponentin toimintaa. Tämän testauksen tason arvioitiin olevan riittävä piirilevyn testauksessa sillä ledien ja summerin toiminta testataan myös lopputestauksessa. Teoriaosuudessa käsiteltiin testauskattavuutta kustannusten näkökulmasta, joka auttoi riittävän testauskattavuuden arviointiin, sillä toteutetuille testeille ei ollut määritelty tarkkoja vaatimuksia.

Kokemukset virrankulutukseen perustuvasta testausratkaisusta olivat hyvät ja menetelmää voisi hyödyntää myös muilla tuotteilla. Testit suunniteltiin alun perin hyvin yleiskäyttöisiksi rakenteeltaan, joten pienellä jatkokehityksellä testejä voisi uudelleen käyttää eri laitteiden testauksessa. Kuten testien sekvenssi kuvista voidaan havaita ovat testien rakenteet hyvin samankaltaiset (kuvat 12 ja 13). Yksi jatkokehityksen mahdollisuus voisi olla testimenetelmän parametrisointi, jolloin voitaisiin samaa testipohjaa käyttää eri testeille.

Työssä kohdatut ongelmat saatiin ratkaistua ja havaittujen ongelmien syihin voidaan jatkossa kiinnittää paremmin huomiota. Esiin nousseita ongelmia tutkiessa havaittiin testeissä puutteita testien purkutoimissa, jotka vaikuttivat muiden testien suoritukseen. Yksittäisen testin on jätettävä laite samaan tilaan kuin se oli testin alkaessa. Jatkossa tulisikin kiinnittää huomiota, että testien rakenne noudattaa huolellisesti neljän vaiheen testimenetelmää, etenkin testin purkutoimien osalta. Myös testispesifikaatioiden kattavuuteen tulisi kiinnittää huomiota, jotta testauksen toteutus on yksiselitteinen testien toteuttajalle.

Kokemukseni testauksesta ja testijärjestelmistä oli melko vähäistä tätä opinnäytetyötä aloittaessani. Työn alkuun syvennyin teorian tasolla testaamiseen ja testauksen tavoitteisiin laadunhallinnan näkökulmasta. Tästä teoriakatsauksesta sain hyvät perusteet itse testien suunnitteluun ja toteutukseen.

LÄHTEET

American Society for Quality 2022a. Quality Assurance & Quality control. Hakupäivä 6.5.2022 https://asq.org/quality-resources/quality-assurance-vs-control.

American Society for Quality 2022b. Cost of Quality (COQ). Hakupäivä 6.5.2022. https://asq.org/quality-resources/cost-of-quality.

Aniche, M. 2022. Effective Software Testing: A Developer's Guide. Manning Publications. Hakupäivä 9.5.2022. O´Reilly eBooks. Vaatii käyttöoikeuden.

Dustin, E., Rashka, J., & Paul, J. 1999. Automated software testing: introduction, management, and performance. Addison-Wesley Professional. Hakupäivä 9.7.2022. O´reilly eBooks. Vaatii käyttöoikeuden.

Haikala, Ilkka & Märijärvi, Jukka 2004. Ohjelmistotuotanto. Hämeenlinna: Karisto Oy

ISTQB Glossary. International Software Testing Qualifications Board. Hakupäivä 9.5.2022. https://glossary.istqb.org.

Krishnamoorthi, K.S., Krishnamoorthi, V.R. & Pennathur, A. 2018. A First Course in Quality Engineering: Integrating Statistical and Management Methods of Quality (3rd ed.). CRC Press. Hakupäivä 7.5.2022. O´Reilly eBooks. Vaatii käyttöoikeuden.

Meszaros, G. 2007. xUnit test patterns: Refactoring test code. Addison-Wesley Professional. Hakupäivä 9.7.2022. O´Reilly eBooks. Vaatii käyttöoikeuden.

Proto-electronics 2022. How to Find Defects on a PCB? Hakupäivä 9.7.2022. https://www.proto-electronics.com/blog/how-to-find-defects-on-a-pcb.

Robot Framework Foundation 2022. Robot Framework User Guide. Hakupäivä 9.5.2022. https://robotframework.org/robotframework/5.0/RobotFrameworkUserGuide.html.