OULUN AMMATTIKORKEAKOULU

Tiia Viitanen

JÄRJESTELMÄTESTAUS TUOTEKEHITYSPROJEKTISSA

JÄRJESTELMÄTESTAUS TUOTEKEHITYSPROJEKTISSA

Tiia Viitanen
Opinnäytetyö
Syksy 2024
Tietotekniikan tutkinto-ohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietotekniikan tutkinto-ohjelma, ohjelmistokehityksen suuntautumisvaihtoehto

---

Tekijä(t): Tiia Viitanen
Opinnäytetyön nimi: Järjestelmätestaus tuotekehitysprojektissa
Työn ohjaaja(t): Harri Määttä
Työn valmistumislukukausi ja -vuosi: Syksy 2024 Sivumäärä: esim. 42 + 4 liitettä

Opinnäytetyössä tutkittiin järjestelmätestauksen suunnittelua ja toteutusta tuotekehitysprojektissa. Projektissa testaamisen kohteena oli toimeksiantajan Haltian Oy:n kehittämä IoT-laite. Tavoitteena oli varmentaa, että tuote vastaa asetettuja vaatimuksia, sekä minimoida jatkotestauksessa havaittavat ongelmat. Järjestelmätestauksen toteuttamiseen ei ole yhtä oikeaa tapaa, joten työssä keskityttiin yleisimpiin menetelmiin. Opinnäytetyö on jaettu kahteen osaan: teoria ja toteutus. Teoria osuudessa käytettiin tieteellisiä tutkimuksia, artikkeleita sekä toimeksiantajan sisäisiä dokumentteja.

Testaus suunniteltiin ja toteutettiin toimeksiantajan toimitiloissa. Toteutuksen aikana testiympäristössä käytettäviä laitteita vaihdettiin sujuvan testaamisen sekä luotettavien testituloksien takaamiseksi. Toteutuksen alussa suoritettiin testien suunnittelu. Työssä käytetyt työkalut, menetelmät sekä testitapaukset määräytyivät tuotekehitysprojektin projekti- ja testisuunnitelman mukaisesti. Testitapaukset suoritettiin vaiheittain ja jokainen testitapaus raportoitiin määritellyllä tavalla. Testituloksien kirjaamiseen käytettiin TestLink- ja Excel-työkalua. IoT-laitteesta testattiin kernel-tason ominaisuudet ja rajapinnat. Testaamisella varmistettiin tuotteen alatason toiminta ja suorituskyky. Tuloksissa huomioitiin myös testiympäristön mahdollinen vaikutus.

Kokonaisuudessa suoritetuissa testeissä havaittiin vain yksi heikkous, joka liittyi WLAN-moduulin Dual Stack-ominaisuuteen. Havainnon vuoksi tuotteessa siirryttiin käyttämään Single Stack-menetelmää. Testauksen jälkeen tuote siirtyi jatkokehitykseen, jossa suoritetaan kattavampia testejä uudessa testiympäristössä.

Asiasanat: ohjelmistotestaus, Python, Linux, manuaalinen testaus, testitapaus, testisuunnitelma

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Bachelor of Engineering, Option of Software Development

---

Author(s): Tiia Viitanen
Title of thesis: System testing in product development project
Supervisor(s): Harri Määttä
Term and year when the thesis was submitted: Spring 2024
Number of pages: 42 + 4 appendices

The target for this thesis was to plan and execute system tests in a product development project. Main goal was to make sure that IoT-device can proceed to further development. Commissioner of thesis was Haltian Oy. System testing focused on IoT-devices kernel-level systems and interfaces.

This thesis was sectioned into two separate sections: theory and practical work. Because there is more than one correct way to execute system testing, this work focuses on the most known and used practices. Sources used in the theory part were scientific research, articles and commissioners internal documentations.

Different test cases were planned and executed in Haltian office space. In addition to the system tests, this project also included developing a light production test tool. That tool was used to test multiple different devices in the beginning stages of the project. During the project only one weakness was found. This was WLAN chips Dual Stack features range in office space. This was noted and reported accordingly.

Keywords: system testing, Python, Linux, manual testing, test case, test plan

SISÄLLYS

KÄSITTEET

AP	Access Point, tukiasema
BAT	Business Acceptance Testing, liiketoiminnan näkökulmasta suoritettu hyväksymistestaus
CAN	Controller Area Network, CAN-väylä
CAT	Contract Acceptance Testing, tuotteen julkaisemisen jälkeinen hyväksymistestaus
DD	Decimal Degrees, Desimaaliasteet
DDM	Degrees Decimal Minutes, asteet ja desimaaliminuutit
eMMC	Embedded MultiMediaCard, laitteen emolevyyn upotettu muistikortti.
GNSS	Global Navigation Satellite System, satelliittipaikannusjärjestelmä
HW	Hardware, laitteisto
IDE	Integrated Development Environment, ohjelmointiympäristö
IoT	Internet of Things, Esineiden Internet
IP	Internet Protocol, internetprotokolla
MTP	Master Test Plan, kokonaistestisuunnitelma
OAT	Operational Readiness Testing, tuotantoa edeltävä hyväksymistestaus
PC	Personal Computer, tietokone
QA	Quality Assurance, laadunhallinta
RAT	Regulation Acceptance Testing, lakien ja säädöksien näkökulmasta suoritettu hyväksymistestaus
RF	Radio Frequency, radiotaajuus
Rx	Receiver, vastaanotin
SCP	Secure Copy Protocol, tiedostojen kopiointi SSH-protokollan yli.
SDK	Software Development Kit, ohjelmistokehityspaketti
SSH	Secure Shell, etäkäyttöohjelmisto, jota käytetään järjestelmien välisiin salaisiin yhteyksiin.
SSID	Service Set Identifier, langattoman lähiverkon verkkotunnus
STA	Station, asema
SCTP	Stream Control Transmission Protocol, siirronohjauksen yhteyskäytäntö
SW	Software, ohjelmisto
TCP	Transmission Control Protocol, siirronohjauksen yhteyskäytäntö
TTFF	Time-To-First-Fix, tarkkaan paikanmääritykseen kuluva aika

Tx	Transmitter, lähetin
UART	Universal Asynchronous Receiver/Transmitter, sarjapiiri
UAT	User Acceptance Testing, käyttäjän hyväksymistestaus
UDP	User Datagram Protocol, siirronohjauksen yhteyskäytäntö
VIN	Voltage In, syötetty virta
WLAN	Wireless Local Area Network, langaton lähiverkko

1 JOHDANTO

Tämän opinnäytetyön tarkoituksena on suunnitella ja suorittaa järjestelmätestausta Haltian Oy:n tuotekehitysprojektissa olevalle IoT (Internet of Things) -laitteelle. Työssä keskitytään tuotteen Linux-käyttöjärjestelmän ytimen (kernel) toiminnallisuuksien testaamiseen. Opinnäytetyössä vastataan kysymyksiin ”Mitä ohjelmistotestauksen eri menetelmiä tuotekehitysprojekti sisältää?” ja ”Mitä työkaluja järjestelmätestauksen suunnittelu ja toteutus tuotekehitysprojektissa vaatii?”.

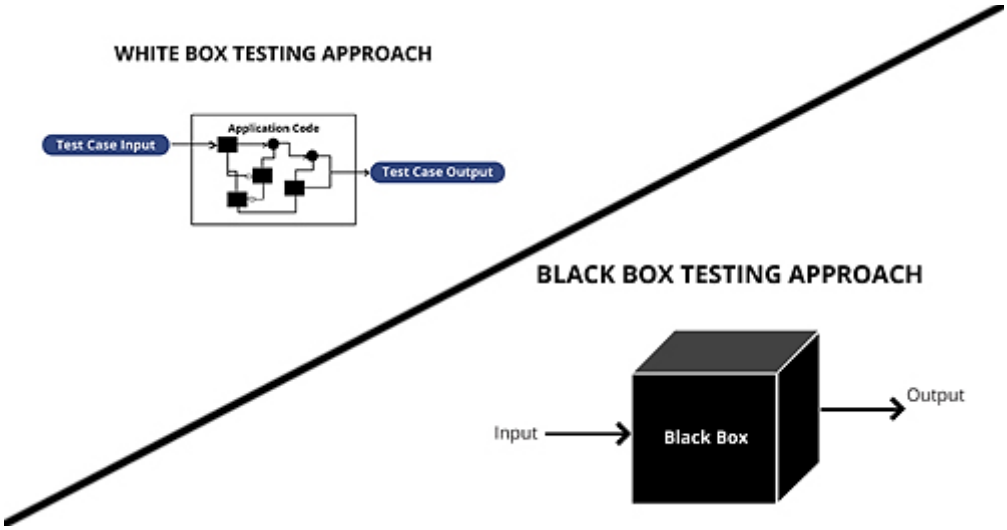
Työssä kerrotaan, miten ohjelmointitestaus tuotekehitysprojektissa toteutetaan ja mitä työkaluja järjestelmätestien suunnittelu ja toteutus vaatii. Tässä käytetään toimeksiantajan laatimia laadunhallintaprosessiin (QA) liittyviä aineistoja. Työhön liitetyt dokumentit on kirjoitettu englanniksi, koska se on toimeksiantajan käyttämä kieli virallisissa raporteissa ja ohjeissa. Opinnäytetyö on jaettu kolmeen osaan. Ensimmäisessä osassa käsitellään testauksen eri menetelmiä ja tasoja sekä vaiheita, joista tuotekehitysprojektin ohjelmistotestaus koostuu. Toinen osa selittää testauksen suunnittelemisen sekä testien toteuttamisen. Kolmannessa osassa käsitellään testien tulokset ja työn lopputuloksen. Tavoitteena työssä on tehdä laitteelle mahdollisimman automaattisesti ajettava tuotannontesti sekä suunnitella ja suorittaa toiminnallisia ja ei-toiminnallisia testejä, joiden tuloksia voidaan hyödyntää laitteen jatkokehityksessä.

2 TESTAUSPROSESSI

Tuotekehitysprojektissa tuotetta testataan koko projektin aikana monella eri tavalla. Testausprosessi sisältää testauksen tasoja, joiden sisällä on erilaisia testausmenetelmiä testauksen toteuttamiseen. (Kasurinen 2013, 64.) Esimerkiksi Haltian pyrkii seuraamaan kaikissa projekteissa yhtenäistä QA-prosessia, jotta yritys voi pitää laaduntason mahdollisimman korkeana (Haltian Oy 2024a). Tuotteisiin suoritetaan myös laite- eli HW-testejä, mutta tässä luvussa avataan, miten testauksen vastuualueet jakautuvat, suunnitellaan ja toteutetaan ohjelmisto- eli SW-puolella sekä mitä eri testausmenetelmiä on käytössä. Ohjelmistotestauksessa ei kuitenkaan ole yhtä totuutta ja asiantuntijoilla on eriäviä mielipiteitä, mitä osa-alueita ja tasoja testaukseen kuuluu ja mitä termejä näistä käytetään. (Valagroup 2022.)

2.1 Testausmenetelmät

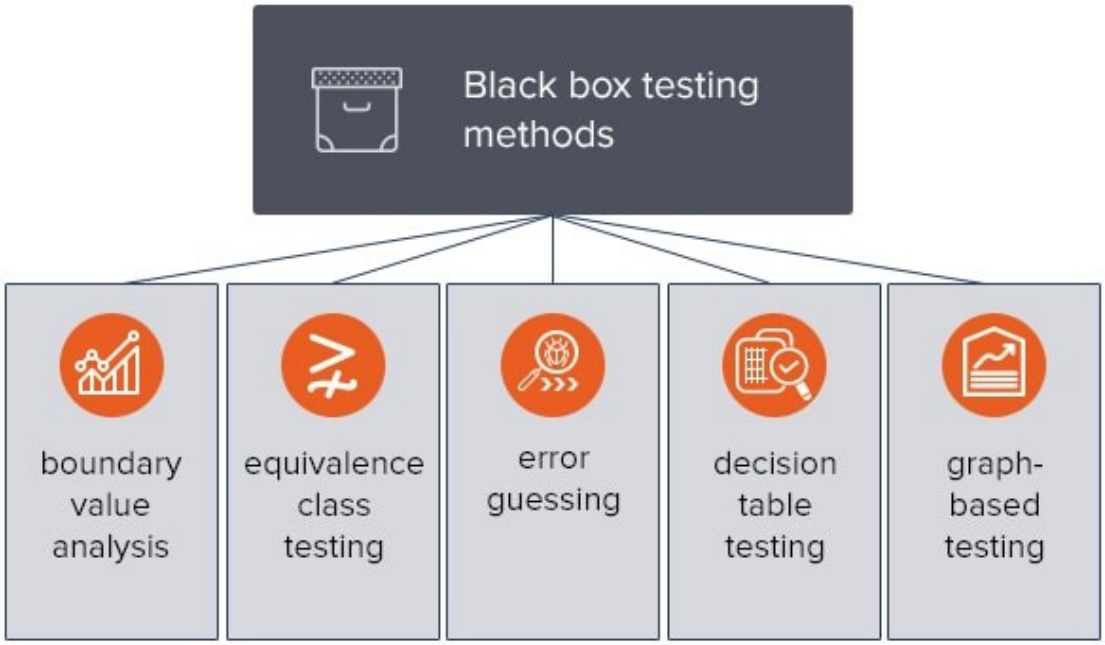
Testaamisen eri tasojen lisäksi on käytettävissä eri testausmenetelmiä. Nämä menetelmät kuvailevat, kuinka testausta suoritetaan ja miksi. Ohjelmistotestauksen käsikirjan mukaan menetelmät voidaan jakaa eri osiin, staattinen ja dynaaminen, musta laatikko- (black box), lasilaatikko- (white box, glass box, clear box) sekä harmaa laatikko- (grey box) testaamiseen. (Kasurinen 2013, 65–68.)



KUVA 1. Kuvaus white box- ja black box- testaamisen lähestymistavoista (Khandelwal 2019)

Staattisella testaamisella tarkoitetaan testausta, jossa ohjelmistoa tai sen osia ei suoriteta, kun taas dynaamisella testauksella tarkoitetaan testausta, jonka suoritettavia osia ajetaan useasti ja eri tavoilla (Mathur 2008, 41). Staattisella menetelmällä pyritään löytämään helposti havaittavia ongelmia ennen kuin ohjelmistoa käynnistetään. Dynaamisen menetelmän tavoite on löytää virheitä, jotka esiintyvät ohjelmiston suorittamisen tuloksena. (Kasurinen 2013, 65.) Näistä menetelmistä staattinen testaus on kustannustehokkaampi, sillä mahdolliset virheet löydetään aikaisemmassa vaiheessa. Staattisella testaamisella löydetään noin 30–70 % virheistä. (Farooq 2011.)

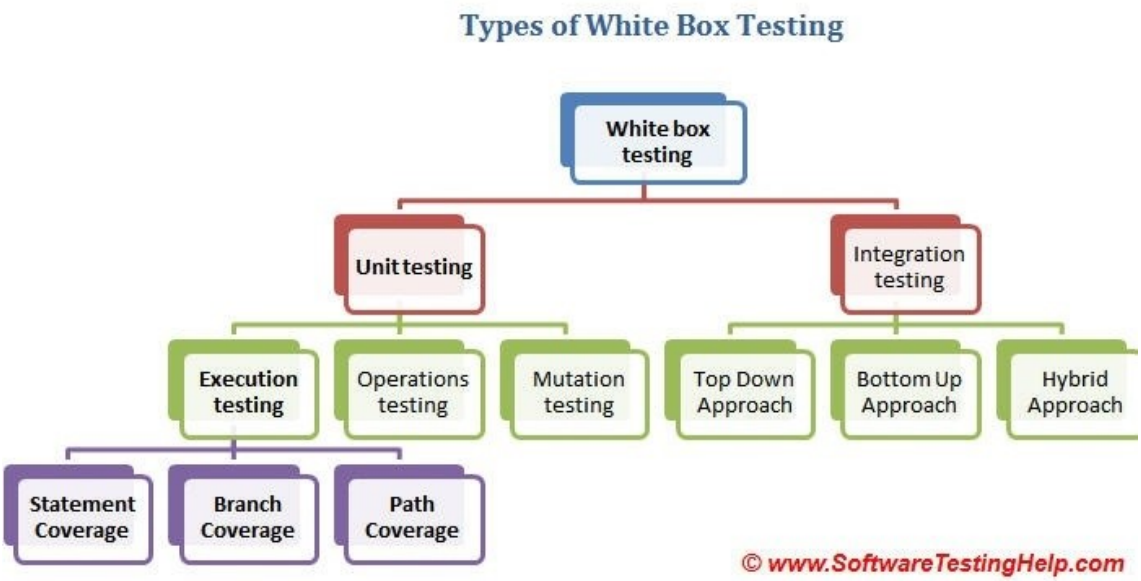
Kasurisen (2013, 65) mukaan perinteisin testausmenetelmä on black box-testaus. Menetelmällä annetaan ohjelmistolle syötteitä, mutta ei seurata sisäistä toimintoa (Kasurinen 2013, 65). Termi black box tulee siitä, ettei käyttäjä tai testaaja näe ohjelmiston tai ”laatikon” sisään (kuva 2) (Ashtari 2022). Menetelmää voidaan käyttää jokaisella testauksentasolla, missä on saatavilla suoritettava ohjelmisto (Kasurinen 2013, 66). Black box-testaamisen yksi vahvuus on Impervan mukaan se, että tällä menetelmällä testaajalla ei tarvitse olla ohjelmointitaitoja sekä väärien tuloksien mahdollisuus on matalampi. Huonoja puolia on muun muassa testauksen automatisoinnin ja epäonnistuneen testin virheen havaitsemisen vaikeus. (Imperva 2024a.) Vaikka, black box-testaamisessa ohjelmalle syötettään eri arvoja, jokaisen arvon testaaminen ei kannata. Tämän vuoksi testaamisessa hyödynnetään eri metodeja (kuva 2), joiden avulla voidaan saavuttaa riittävän hyvä testauskokonaisuus. (QATestLab 2020.)



KUVA 2. Black box-testaamisen metodit (QATestLab 2020)

White box-termi voidaan ajatella samalla periaatteella, kuin black box. White box-testaamisessa tarkastellaan testin aikana, miten ohjelmisto reagoi ja mitä tuloksia saadaan (kuva 1), mutta myös katsotaan, mitä ohjelmiston sisällä tapahtui testin aikana. (Kasurinen 2013, 67.) White box-testaaminen vaatii tietoa ohjelmiston arkkitehtuurista ja sisäisten komponenttien toiminnallisuuksista. Useimmiten tämä testaaminen suoritetaan ohjelmistokehittäjän toimesta yksikkötestauksen ja integraatiotestauksen tasolla (kuva 3). (Khandelwal 2019.)

Grey box-testaamismenetelmä on yhdistelmämenetelmä. Tässä yhdistyy black box- ja white box-menetelmät, jossa yhtäaikaisesti testataan mallikattavuutta sekä koodikattavuutta. Tavoitteena on varmistaa, että ohjelmistolle asetetut vaatimukset täyttyvät ja lähdekoodi on tarkistettu vaaditulla tavalla. (Kasurinen 2013, 68.) Tehokkaimmin tätä testaus tapaa voidaan käyttää esimerkiksi integraatiotestauksessa tai turvallisuus arvioinnin tekemisessä. Menetelmä vaatii tietoa ohjelmiston sisäisestä toiminnasta, mutta Impervan mukaan testaajan ja ohjelmistokehittäjän välille kannattaa tehdä selkeä rajaus, jotta saadaan mahdollisimman puolueettomia ja luotettavia testituloksia. (Imperva 2024b.)



KUVA 3. White box-testaamisen tasot (Khandelwal 2019)

2.2 Testisuunnitelma

Hyviin toimintatapoihin kuuluu testisuunnitelman laatiminen jokaisen projektin alussa. Testisuunnitelman tekemiseen osallistuvat useimmiten testi päällikkö, testaaja, asiakas, ohjelmistokehitys tiimi, tuotepäällikkö ja projektipäällikkö. (GeeksforGeeks 2023a.) Roolien ja vastuiden jako on projektin ja tiimin koosta riippuvainen. Hyvin tehty suunnitelma strukturoi, ohjaa ja antaa selkeyttä testaajille. (Holota 2023.) Aroran (2024) mukaan testisuunnitelmia on kolmea eri tyyppiä: MTP eli kokonaistestisuunnitelma (master test plan), testausvaihesuunnitelma (test phase plan) ja spesifinen testisuunnitelma (spesific test plan) (Arora 2024).

MTP on nimensä mukaisesti projektissa ”korkean tason” dokumentti. Tässä suunnitelmassa käydään läpi sekä laite- että ohjelmistopuolen testaus suunnitelmat ja vaatimukset. Suunnitelman tarkoitus on määritellä

- ketkä ovat projektissa mukana
- mikä on projektin tavoite tai kokonaispäämäärä
- mitkä osa-alueet testataan
- minkä tason testejä suoritetaan ja miksi
- mitä menetelmiä testauksessa käytetään
- mitä työkaluja käytetään
- mitä mahdollisia riskejä on ja miten niiltä voi välttyä

- miten vastuu jakautuu
- miten tulokset raportoidaan
- mitä vaaditaan, että tuote voidaan siirtää tuotantoon.

Hyvin suunnitellussa projektissa MTP toimii kontrolli dokumenttina. (Borg 2017.)

Testausvaihesuunnitelmassa määritellään tarkemmin eri testauksen tasojen aikataulua, mitä työkaluja käytetään, mitä välietappeja testaaminen vaatii ja mitä ei kokonaistestisuunnitelmassa ole käyty läpi. Viimeinen suunnitelmatyyppi eli spesifinen testisuunnitelma sisältää käytävät testitapaukset ja testauksen tasot. Suunnitelmaan kirjataan, mitä testejä esimerkiksi suorituskykytestaaminen sisältää. Tässä kirjataan, millä työkaluilla testejä suoritetaan, mikä on testausympäristö sekä mitkä ovat halutut tai odotetut lopputulokset. (Arora 2024.)

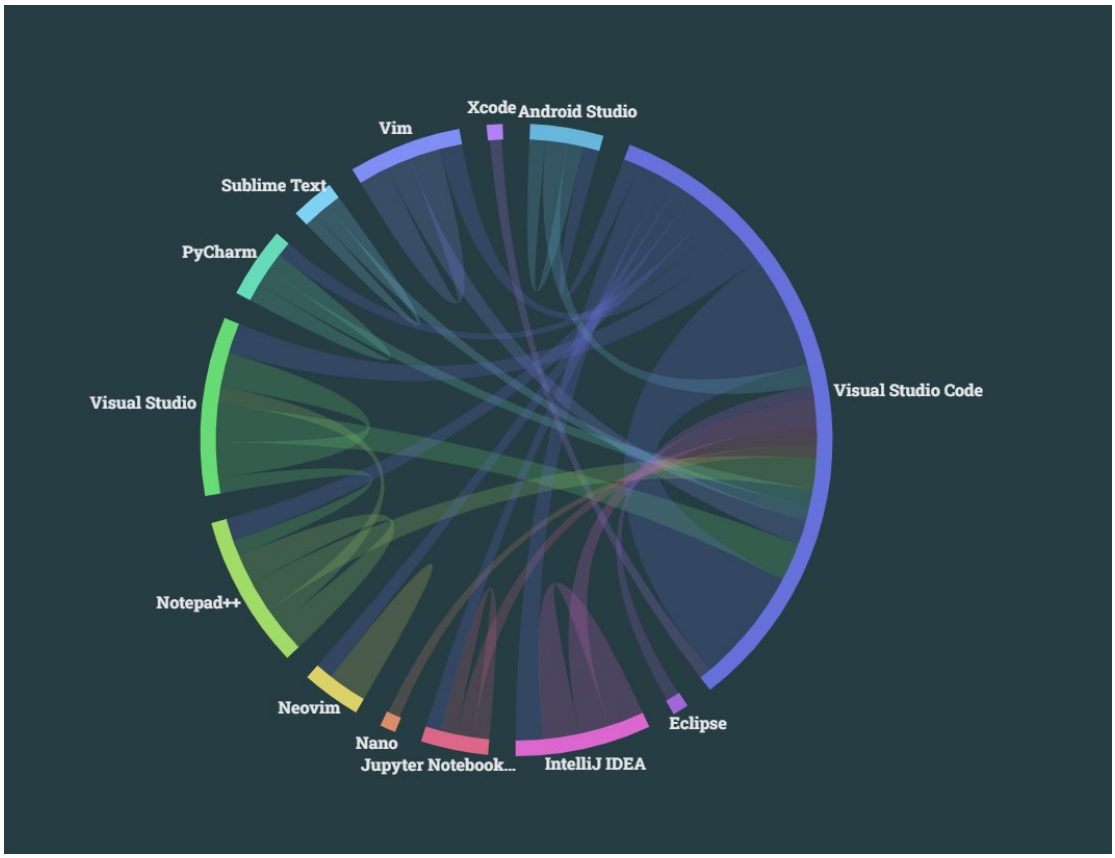
MTP:stä Valagroup (2023b) on käyttänyt termiä testausstrategia, sillä siinä keskitytään koko projektin isoon kuvaan ja ohjelmiston testaussuunnitelma (liite 1) keskittyy puhtaasti SW puolen vaatimuksiin (Valagroup 2023b). Projektin aikana tapahtuvien muutoksien vuoksi testauksessa seurataan suunnitelmaa, josta voidaan tarkastaa muutosten vaikutus testaamiseen sekä projektin aikatauluun (Rice 2024).

2.3 Työkalut testaamiseen

Ohjelmistotestauksen työkaluja on useampia erilaisia, mutta Kasurisen (2013, 84–88) kirjoituksen mukaan suurin osa testaamisesta suoritetaan ohjelmistokehityksen kanssa samoilla työvälineillä. Riippuen projektista, voi organisaatio myös itse kehittää omia työvälineitä testaamiseen. Kasurinen on jakanut työkalut kuuteen eri teemaan. Nämä teemat ovat kehitysympäristöt ja yleiset työvälineet, testausautomaation työkalut, vikatietokannat ja bugiraportit, tyngät, analysaattorit sekä dokumenttipohjat. Dokumenttipohjilla tässä tapauksessa viitataan mihin tahansa lomakepohjaan, johon kirjataan ylös testien tavoitteet ja tulokset. Kehitysympäristöllä viitataan samaan työkaluun, jota on ohjelmiston tekemiseen käytetty. (Sama.) Näistä työkaluista voidaan myös käyttää lyhennettä IDE eli Integrated Development Enviroment. IDE tyypillisesti toimii yhden graafisen käyttöliittymän (graphical user interface, GUI) kautta. IDE vaihtoehtoja on monia ja valintaa tehdessä tulee tietää, mitä työkalulla halutaan saavuttaa ja mitä ohjelmointikieliä käytetään. (Wasay 2024.)

Stackoverflown (2023) vuosittaisen kyselyn mukaan kaikista suosituin IDE on selkeästi Visual Studio Code (VS Code) ja sitä käyttää 73,71 % kyselyyn vastanneista (kuva 4). Kyselyssä vastattiin kysymykseen ”Mitä ohjelmointi ympäristöjä käytit säännöllisesti viimeisen vuoden aikana ja millä haluat työskennellä seuraavan vuoden aikana?”. (Sama.)

Uspenskin (2023) mukaan VS Coden suosio voidaan selittää muutaman ominaisuuden avulla. Ensimmäisenä VS Code tukee satoja erilaisia ohjelmointikieliä, eikä projektissa ole tarvetta käyttää useampaa työkalua ohjelmointikielien vaihtuessa. Toisena ominaisuutena on laaja käyttäjäyhteisö, joka on muodostunut ohjelmiston laajennuksien ympärille. VS Codessa on saatavilla suuri määrä laajennuksia, joita myös käyttäjät voivat itse kehittää. Laaja käyttäjäyhteisö myös parantaa VS Coden kehittymistä kokonaisuutena. Kolmas ominaisuus on monikäyttöisyys. VS Code toimii tekstieditorina, joka sisältää myös terminaali-ikkunan, integroidun vikojen etsinnän sekä Git-versionhallinnan. Viimeisimpänä ominaisuutena on hinta. VS Code on täysin ilmainen ohjelmointiympäristö ja sitä voi käyttää kaikilla käyttöjärjestelmillä. (Sama.)



KUVA 4. Stackoverflown 2023 vuonna tehdyn kyselyn mukaan käytetyimmät IDE-työkalut (Stackoverflow 2023)

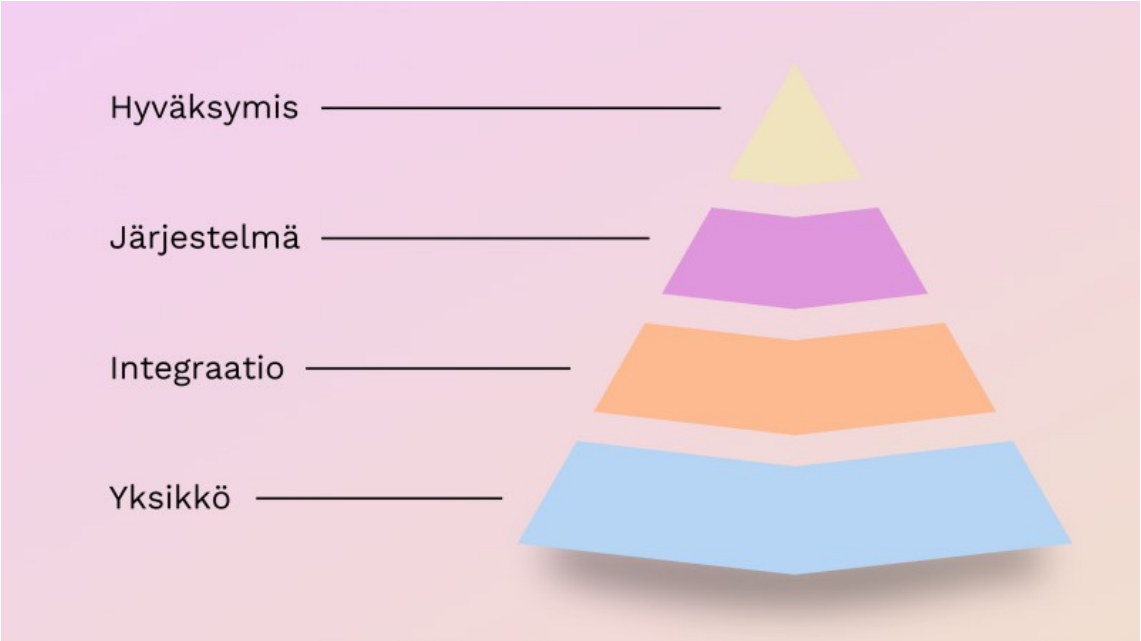
Testausautomaation työkalut ovat usein organisaation itse rakentamia työkaluja, jotka koostuvat useammasta sarja-ajettavasta testiajurista. Automaatiota käytetään monesti regressiotestaukseen, laadunvarmennukseen sekä kuormitustestaukseen. (Kasurinen 2013, 86.) Työkalu on osa ohjelmistoa, jonka avulla voidaan määritellä testaamisen tehtäviä, joiden ajamiseen myöhemmin tarvitaan mahdollisimman vähän ihmisen manuaalista työskentelyä (Testim 2022).

Vikatietokannalla tarkoitetaan työkalua, jota käytetään vikojen raportointiin. Monesti tähän käytetään samaa työkalua, jota projektissa käytetään myös projektinhallintaa varten. (Kasurinen 2013, 86.) Vaihtoehtoja on satoja ja oikean työkalun valintaan vaikuttaa useampi tekijä. Kuitenkin yksi tunnetuimmista ja käytetyimmistä työkaluista on Jira. (Singh 2023.) Jira on Atlassianin 2020 kehittämä ketterän kehityksen työkalu, jota yli 65 000 yritystä käyttää (Atlassian 2024).

Testityökaluista puhuttaessa tyngällä kuvataan sijaiskomponenttia. Näillä mahdollistetaan yksittäisten osien testaaminen, kun koko ohjelmiston testaaminen ei ole vielä mahdollista tai edes ajankohtaista. Yleisesti testitynkiä käytetään yksikkötestaamisessa sekä integraatiotestauksessa yksittäisten komponenttien lisäksi. (Kasurinen 2013, 87.) Analysaattoreita käytetään staattisen testauksen työkaluna. Static Application Security Testing (SAST) työkalulla tarkoitetaan ohjelmistoa, joka lähde koodia ajamatta käy läpi koodin ja tarkistaa, että koodi täyttää asetetut vaatimukset. (Owasp 2024.)

2.4 Testauksen tasot

Ohjelmistotestaus voidaan jakaa neljään eri tasoon (Tuleap 2024). Pyramidimalli (kuva 5) on yksi tapa havainnollistaa testauksen tasot (Valagroup 2022).



KUVA 5. Pyramidimalli (Valagroup 2022)

Vaikka testausmenetelmät vaihtelevat asiantuntijan mukaan, nämä mallit ovat suosituimpia ohjelmistokehitysprojekteissa. Kirjassa Pragmatic Software Testing Black mainitsee myös havaitsemastaan code and fix-mallista. Tässä mallissa prototyypin rakentamisen sijasta tähdätään rakentamaan välittömästi valmista tuotetta, jolloin projektin ohjelmistokehitys ja testaaminen etenee seuraavanlaisesti: Ohjelmistokehittäjät koodaavat ja korjailevat vikoja sitä mukaan, kun projekti edistyy. Tämän vaiheen jälkeen koostetaan suoritettava ohjelmisto, jonka testaaminen siirtyy epäviralliselle testaus tiimille. Blackin sanojen mukaan tämä tiimi koostuu usein juniori ohjelmistokehittäjistä, jotka eivät ole saaneet töitä ohjelmistokehittäjinä, eikä heillä ole virallisesti kokemusta testaamisesta ollenkaan. Kokemuksen puutteesta huolimatta he löytävät vikoja, joille tehdään pikakorjauksia ja usein myös tuotteen mennessä asiakkaalle löytävät he myös vikoja. Black kertoo, kuinka tästä mallista olisi tärkeä luopua. (Black 2007, 28–29.)

Eri vaiheiden suunnitteleminen ja toteutus jaetaan ohjelmistokehittäjien ja testaajien välille. Osa testausvaiheista voi kuulua molemmille ja näiden tehtävien jako on projektikohtaista, mutta seuraavissa luvuissa selkeytyy, miten vastuualueet usein jakautuvat. (Valagroup 2022.)

2.4.1 Yksikkötestit

Yksikkötestaus keskittyy ohjelmiston yksittäisiin osiin tai komponentteihin. Testauksen tarkoituksena on varmistaa, että ohjelmisto toimii suunnitellulla tavalla ja täyttää vaatimukset.

(GeeksforGeeks 2023b.) Yksikkötestien kirjoitus ja toteutus kuuluu usein ohjelmistokehittäjien tehtäviin. Testejä kirjoitetaan koodin kirjoittamisen ohella, jotta yksittäiset osiot tulevat testatuiksi kehittämisvaiheessa ja mahdolliset yksilötason virheet tulevat esille. Tätä testausta voidaan kutsua white box-testaukseksi, koska testien kirjoittaja tuntee ohjelmiston rakenteen parhaiten. Ohjelmistokehittäjien suorittama testaus on hyvin usein white box-testaamista. (Valagroup 2022.)

Yksikkötestit usein automatisoidaan ja yhdistetään koodin versiohallintaan, jotta testit voidaan ajaa aina kun uusia yksiköitä ja komponentteja halutaan lisätä. Testausta varten on olemassa erilaisia työkaluja, joilla voidaan kirjoittaa ja automatisoida testejä. Työkalun valintaan vaikuttaa ohjelmistokehittäjän käyttämä ohjelmointikieli sekä henkilökohtainen preferenssi. (Valagroup 2023c.)

2.4.2 Integraatiotestaus

Yksikkötestauksien jälkeen voidaan suorittaa integraatiotestausta. Integraatiotestauksessa tarkistetaan, kuinka yksittäiset komponentit tai moduulit toimivat yhdessä. Ohjelmistokehittäjillä on eriäviä tapoja rakentaa koodia, joten on tärkeä testata eri funktioiden yhteensopivuus. (Awati 2022.) Integraatiotestausta voidaan lähestyä muutamilla eritavoilla, kuten esimerkiksi Big Bang- tai inkrementaalinen lähestymistapa. Big Bang-lähestymistavassa testaus aloitetaan vasta, kun kaikki tai suurin osa komponenteista on kehitetty. Tässä tavassa testaus suoritetaan koko järjestelmälle kokonaisuudessaan. (Valagroup 2023a.)

Inkrementaalisessa testauksessa ohjelmiston moduulit ja komponentit jaetaan pienempiin osiin, jolloin nämä integroidaan sekä testataan vaiheittain. Tämä lähestymistapa voidaan suorittaa joko ylhäältä alas -, alhaalta ylös - tai yhdistelmämenetelmällä. Vertaillessa näitä kahta lähestymistapaa, on Big Bang-malli yksinkertaisempi ja se sopii pienille projekteille, joissa on vain rajoitettu määrä moduuleja ja komponentteja, kun taas inkrementaalinen malli voi vaatia enemmän aikaa useamman testausvaiheen vuoksi. Big Bang-lähestymistavassa yksittäisten ongelmien havaitseminen on haastavampaa, koska järjestelmä testataan isona kokonaisuutena. Inkrementaalisessa tavassa testausta tehdään pienissä osissa, jolloin vikoja havaitaan nopeammin sekä ongelman sijainti on helpompi paikantaa. (Sama.)

2.4.3 Järjestelmä- eli systeemitestaus

Järjestelmätestauksessa tarkkaillaan ja testataan, toimiiko ohjelmisto vaatimusten mukaisesti. Järjestelmätestaus jaetaan yleensä kahteen isompaan osa-alueeseen: toiminnalliseen ja ei-toiminnalliseen testaamiseen. Järjestelmätestauksessa varmistetaan asetettujen vaatimusten saavuttaminen sekä ohjelmiston heikkouksien havainnoiminen. Tämän testaus tason suunnittelu ja toteutus kuuluu QA testaajalle. Jotta testien tulokset olisivat mahdollisimman luotettavia, nämä suunnitellaan ja toteutetaan erillisessä testiympäristössä. Ohjelmistokehittäjät eivät osallistu testien suunnitteluun tai suorittamiseen, sillä heidän tietonsa ohjelmiston toiminnasta ja rakenteesta saattaisivat vaikuttaa lopputulokseen. (Indeed 2023.)

Toisin kuin yksikkötestaus, tämä osa-alue toteutetaan black box testausmenetelmällä (Ashtari 2022). Toiminnallinen testaaminen perustuu projektissa asetettuihin vaatimuksiin, kun taas ei-toiminnallisessa testaamisessa käsitellään projektissa asetettuja odotuksia (GeeksforGeeks 2023c).

2.4.4 Hyväksymistestaus

Hyväksymistestaus on testauksen viimeinen taso. Sen tavoitteena on varmistaa, että ohjelmisto tai laite on luotettava ja toimii odotetusti, se täyttää kaikki asetetut toiminnalliset ja ei-toiminnalliset vaatimukset sekä on turvallinen ja helppo käyttää. (Testsigma 2023.) Aiemmat testauksen tasot suoritetaan yleisesti hallitussa testausympäristössä, joka kopioi kohdeympäristöä, mutta hyväksymistestauksessa järjestelmää pyritään käyttämään kohdeympäristössä (Kasurinen 2013, 57).

Collinsin (2022) mukaan hyväksymistestauksen eri tapoja ovat ainakin 7. UAT, Alfa testaus, Beta testaus, RAT, CAT, OAT ja BAT. UAT eli User Acceptance Testing on käyttäjän suorittama testaus metodi, jossa tavoitteena on tarkistaa, että kehitetty ohjelmisto toimii määritellyllä tavalla. (Collins 2022.) Hyväksymistestauksen tavat määritellään seuraavanlaisesti:

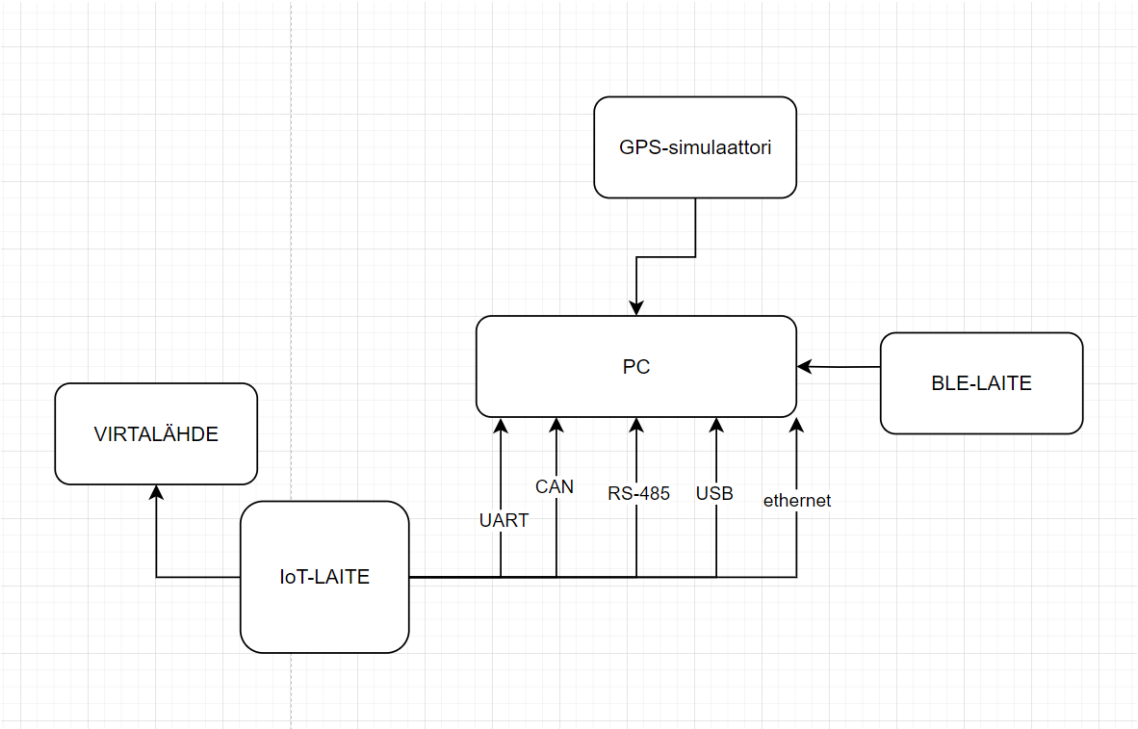
- Alfa testaus
    - Tämän tyyppisellä testaamisella peilataan mahdollisimman tarkasti loppukäyttäjän käytöstä. Koska testaajilla on tietoa ohjelmiston ja laitteen toiminnasta, on mahdollista käyttää black box- ja white box-menetelmiä sekä ohjelmistokehittäjät

pystyvät korjaamaan mahdolliset viat nopeasti. Alfa testaamisen toteutus kuuluu QA tiimille. (PractiTest 2024.)

- Beta testaus
    - Toteutetaan loppukäyttäjillä ja testaustyylin tavoitteena on vastata kysymykseen ”Pitävätkö asiakkaat tuotteesta?”. Tässä tavassa on tärkeää saada ohjelmiston käyttämisestä palautetta, jotta mahdolliset viat voidaan korjata. (PractiTest 2024.)
- RAT
    - Regulation Acceptance Testing methodilla varmistetaan, että tuote vastaa käytettyjä säännöksiä. Tavoitteena on välttyä sakoilta ja rangaistuksilta. (Natchimuthu 2022.)
- CAT
    - Contract acceptance testing metodissa varmistetaan, että tuote läpäisee julkaisemisen jälkeen asetetut testit. Service level agreement (SLA) eli palvelutasosopimus määrittää, että maksu tuotteesta tapahtuu vasta kun CAT testi on mennyt hyväksytysti läpi ja sopimuksen ehdot täyttyvät. (Collins 2022.)
- OAT
    - Operational Acceptance Testing on ohjelmiston tai tuotteen käyttöönotto valmiuden testausta. Tavoitteena on varmistaa, että tuote voidaan hyväksyä tuotannon tasolle. (Codium AI 2024.)
- BAT
    - Business Acceptance Testing metodissa testaaminen suunnitellaan liiketoiminnan näkökulmasta. Tämä tarkoittaa sitä, että vikojen löytämisen lisäksi tutkitaan, mitä se tarkoittaa rahallisesta näkökulmasta katsottuna. Jos UAT:n aikana löytyy vikoja, se hidastaa tuotteen lopullista hyväksymistä, joka vaikuttaa kustannusten nousemiseen. BAT testaamisessa tulee olla tietoa tuotteesta kokonaisuutena, mutta ymmärrystä myös asiakkaan toimintatavoista. (Collins 2022.)

3 TESTAUSYMPÄRISTÖ

Ensimmäinen askel työssä oli testausympäristön perustaminen. Testausympäristö sisälsi kaikki tarvittavat laitteet ja työkalut, joita tuotteen tai ohjelmiston testaaminen vaati. Tässä tapauksessa IoT-laitteen testaamiseen vaadittiin erillinen virtalähde, GPS-simulaattori, Bluetooth low energy-mainostaja (BLE) sekä testaamista varten oleva tietokone (PC) (kuva 6).

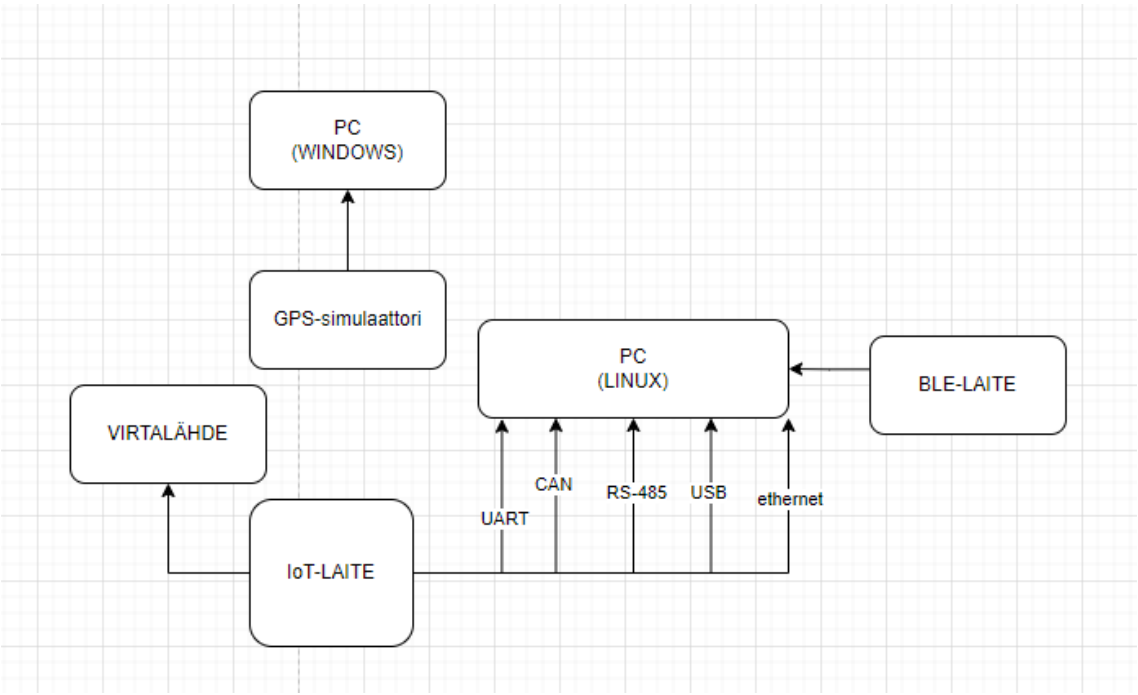


KUVA 6. Testausympäristössä käytetyt kytkennät

Laitteen ominaisuuksiin kuuluivat Universal Asynchronous Receiver/Transmitter (UART), CAN-väylä (CAN), RS-485 (Modbus), USB, Ethernet, WLAN ja Bluetooth. Kuvassa 6 näkyviä ominaisuuksia varten kytkettiin jokaiselle oma liitäntä testausta varten tarkoitettuun tietokoneeseen. Testausympäristö muuttui projektin aikana, koska alkuperäisesti ympäristössä käytetyssä GPS-toistajassa sekä testitietokoneessa havaittiin puutteita. Työn alussa asennettiin GPS:n testaamista varten GPS-toistaja, joka myöhemmin vaihdettiin GPS-simulaattoriin. GPS-toistaja vahvisti ulos asennetun antennin avulla GPS-signaaleja, mutta signaalin voimakkuutta täytyi säätää toistuvasti eikä tällöin testien tuloksiin voinut luottaa. Tämän vuoksi käyttöön otettiin GPS-simulaattori, joka ohjelmoitiin replikoimaan aitoa GPS-signaalia. Simulaattorin käytöllä testien tuloksien tarkkuus ja luotettavuus saatiin varmistettua.

Testauksessa käytettiin ensin tietokonetta, jossa käyttöjärjestelmä oli Windows 11. Tuotannontestauksen toteutuksen jälkeen havaittiin, että Linuxin käyttäminen testaamisessa olisi kannattavampaa työn tekemisen sekä ohjaamisen kannalta. Tätä varten tietokoneeseen asennettiin Oracle VM VirtualBoxilla ajettava Linuxin käyttöjärjestelmää käyttävä virtuaalikone. Oracle VM VirtualBox on cross-platform-virtualisointiohjelmisto (Oracle 2021).

Käyttöjärjestelmän lisäksi tietokoneeseen liitettiin erillinen USB-jakaja, jotta kuvan 6 mukaiset kytkennät olisivat mahdollisia. Virtuaalikoneen käyttöä varten tietokoneeseen liitettiin myös WLAN- ja Bluetooth-adapteri. USB-kytkentöjen määrä osoittautui haasteeksi virtuaalikonetta käyttäessä, koska molemmat järjestelmät eivät voineet käyttää portteja yhtäaikaisesti. Virtuaalikonetta käyttäessä valittiin manuaalisesti käytettävät portit, jotka täytyi virtuaalikoneen sammuttamisen jälkeen luovuttaa takaisin pääkäyttöjärjestelmän haltuun. Valitettavasti järjestelmää vaihtaessa porttien määrä saattoi aiheuttaa niin sanotun blue screen of deathin. Blue screen of death (BSOD) -termillä tarkoitetaan koko käyttöjärjestelmän kaatumista Windowsin kernel-tasolla. Tämä johtuu usein ajuri- tai HW-viasta. (Patrizio 2021.) Tämän vuoksi alkuperäinen tietokone vaihdettiin täysin Linuxin käyttöjärjestelmää käyttävään koneeseen. Tietokoneen vaihdoksen jälkeen muuttui testausympäristö kuvan 8 mukaiseksi. Linux-konetta käytettiin testien suunnitteluun ja toteutukseen, kun Windows-koneella ohjattiin GPS-simulaattorin skenaarioita. Windows-kone jätettiin käyttöön, koska GPS-simulaattorin ohjelmistoa ei ollut mahdollista asentaa Linuxin käyttöjärjestelmään.



KUVA 7. Testausympäristö tietokoneen vaihtamisen jälkeen

4 TESTAUKSEN TOTEUTUS

Testaamisen osa-alueisiin kuuluivat tuotannontestaus sekä järjestelmätestaus. Tuotannontestauksesta oli saatavilla valmiit manuaalisen testauksen ohjeet, mutta työn nopeuttamista varten kehitettiin yksinkertainen testiohjelma. Järjestelmätestauksen vaiheeseen kuului kolme osa-aluetta. Testauksen suunnittelu, toiminnallinen ja ei-toiminnallinen testaus. Seuraavissa luvuissa on tarkemmin esitetty, miten työn aikana osa-alueet toteutettiin. Luvussa 5 esitetään testaamisen tulokset.

4.1 Tuotannontestaus

Tavoitteena oli testata useampi laite ja varmistaa, että kaikki ovat vaaditussa kunnossa. Tuotannontestauksen manuaalin avulla luotiin lyhyempi ja nopeammin suoritettava tuotannontestiohjelma, jolla testattiin pieniä laite määriä. Työn aikana testattiin yli 50 laitetta.

Tuotannontesti sisälsi kaksi osa-aluetta: laiteohjelmiston (engl. firmware) asennus sekä ominaisuuksien testaus. Laiteohjelmiston asennukseen tarvittiin UUU (Universal Update Utility) - ja Android SDK (Software Development Kit) -työkaluja. UUU on NPXn kehittämä komentorivityökalu, jota käytetään laiteohjelmiston asennuksessa (github-actions 2024). Kuvassa 8 näkyy, miltä UUU komentorivillä näyttää. Android SDK-työkalut sisältävät adb- ja fastboot-rajapinnat, joista fastboot-rajapintaa käytetään asentamaan uutta laiteohjelmistoa. (Android Developers 2024.)

```
uuu (universal update utility) for nxp imx chips -- libuuu-1.0.1-gffd9837

Succues:0       Failure:3               Wait for Known USB Device Appear...

1:11     5/5 [                            ] SDP: jump -f u-boot-dtb.imx - ↩
   ivtinitramf....
2:1      1/5 [===>                        ] SDP: boot -f u-boot-imx7dsabresd_sd. ↩
   imx ....
```

KUVA 8. UUU-työkalun esimerkkitulostus NPX:n pdf-tiedostosta (github-actions 2024)

Laiteohjelmiston asennusta varten IoT-laite kytkettiin USB-kaapelilla tietokoneeseen. Ennen asennusta varmistettiin, että laite on uuu-tilassa. Tämä tarkastettiin komentoriviltä antamalla komento ”uuu -lsusb”. Jos laiteen tiedot ei tulostu komentoriville, laite irrotettiin virtalähteestä. Uuu-tila aktivoitiin painamalla laitteen USB-Boot painiketta ennen kytkemistä takaisin virtalähteeseen.

Tuotannontestin suorittamisen alussa varmistettiin, että kaikki laitteeseen tarvittavat kaapelit ovat kytketty oikein sekä tarvittavat ohjelmistot käynnistetty. Testissä tarvittaviin ohjelmistoihin kuului PCAN-View, iPerf3 sekä endpoint.py. PCAN-View käynnistettiin CAN-väylän kommunikoinnin välittämistä varten. iPerf3-komentorivityökalua käytetään aktiivisen pakettiliikenteen luomiseen, jonka avulla se mittaa yhteyden siirtokapasiteettia. iPerf3 tukee TCP (Transmission Control Protocol) -, UDP (User Datagram Protocol) - ja SCTP (Stream Control Transmission Protocol) -protokollia. (Dugan ym. 2015.) TCP, UDP ja SCTP on tietoliikenneprotokollia. TCP-protokolla luo yhteyden vastaanottajan ja lähettäjän välille ennen tiedonsiirtoa. UDP-protokolla on yhteydetön ja vasta tiedon siirtyessä, luo yhteyden vastaanottajan ja lähettäjän välille. (Zieniūtė 2024) SCTP-protokolla perustuu TCP- ja UDP-protokolliin, mutta toisin kuin TCP:ssä, SCTP mahdollistaa datan lähetyksen useammalla eri jonolla (Awati 2021). Tämän työkalun avulla tarkastettiin, onko käytetyissä yhteyksissä häiriötä. RS-485:n testaamiseen luotiin erillinen endpoint.py-tiedosto, jonka tarkoitus oli vastaanottaa ja lähettää merkkijono. Python tiedostojen ajaminen toteutettiin komentoriviltä. Ohjelmaa ajaessa argumentiksi lisättiin RS-485:n käyttämä portti.

Koska tuotannontestausohjelma jaettiin kahteen osaan, testiä ajettaessa käytettiin tarkentavia argumentteja. Argumenteilla määriteltiin: asennetaanko laiteohjelmistoa, laitteen versio sekä sarjanumero. Jos tuotannontestauksen alussa uusi laiteohjelmisto asennetaan, pyytää ohjelma asennuksen jälkeen käyttäjää irrottamaan USB-kaapelin laitteesta. USB-kaapelin irrottamatta jättäminen vaikuttaa ominaisuuksien testituloksiin. Tuotannontestausta varten ohjelmistokehittäjä suunnitteli ja kehitti testaus osion, joka päivittyy laitteeseen laiteohjelmiston asennuksen aikana. Testaamisen lopputulokset tallentuivat automaattisesti laitekohtaisesti nimettyyn CSV-tiedostoon saman kansion alle, jossa tuotannontestausohjelma sijaitsi.

4.2 Testien suunnittelu

Tuotannontestauksen jälkeen aloitettiin järjestelmätestaamisen suunnittelu. Testien suunnitteluun käytettiin TestLink-työkalua, johon kirjoitettiin ylös toteutettavat testitapaukset, käytettävät työkalut ja komennot sekä odotetut lopputulokset. TestLink-työkalussa oli mahdollista suoritettujen testien raportit eritellä suoritettavan koodin version mukaan, jotta tuloksista pystyi seuraamaan mitä eroavaisuuksia ilmeni. Tämän lisäksi käytössä oli tavallinen Excel taulukko (liite 2, liite 3), johon merkittiin ajetut testit, verisointi sekä lopputulokset.

Testien suunnitteluvaiheessa tutkittiin, mitä komponentteja laitteessa käytettiin sekä, miten niiden ohjaaminen toteutetaan. Suunnitellessa tarkistettiin projektin alussa kirjoitetun testisuunnitelman (liite 1) rajaukset. Testien suunnittelu jaettiin kolmeen osaan: smoke testit, toiminnalliset ja ei-toiminnalliset testit. Smoke testien suunnittelussa keskityttiin laitteen perustoimintoihin. Tavoite oli tehdä automatisoitava testi, joka ajetaan uusimmalle ohjelmistoversiolle. Toiminnallisten testien kohdalla katsottiin pintaa syvemmälle ja testattiin, toimiiko osat vaaditulla tavalla ja kirjattiin ylös halutut tulokset. Viimeisenä suunniteltiin ei-toiminnalliset testit, joiden tavoitteena oli tarkistaa, toimiiko komponentit luvatulla tavalla.

4.3 Toiminnallinen testaus

Toiminnallisessa testauksessa ensimmäisenä suoritettiin smoke testaus. Smoke testauksessa ohjelmointikielenä käytettiin Pythonia ja testiohjelmassa hyödynnettiin Pythonin pytest-kirjastoa, jolla testattiin tarvittavat komponentit sekä ominaisuudet. Tulevaisuudessa smoke testejä olisi mahdollista ajaa täysin automaattisesti. Smoke testauksen tavoitteena oli testata pääominaisuudet, kuten tunnistaako tietokone USB:lla kytkettyä laitetta. Smoke testien kirjoittamiseen käytettiin VS Codea. Testiohjelmaan luotiin kymmenen suoritettavaa testitiedostoa. Näiden lisäksi kehitettiin viisi taustalla toimivaa apuohjelmaa. Jotta pytestillä olisi mahdollista ajaa testattavat Python-tiedostot sekä niiden sisältämät funktiot, täytyi nimeäminen tehdä vaatimusten mukaisesti. Nimeäminen täytyi olla joko ”test_*.py” tai ”*_test.py” muotoa. Esimerkiksi, jos tiedoston nimi on usb.py ei tiedostoa ajeta. Nimen täytyi olla muodossa ”test_usb.py” tai ”usb_test.py”. Testit suoritettiin automaattisesti ylhäältä alas menevässä järjestyksessä.

Smoke testien tarkoituksena oli:

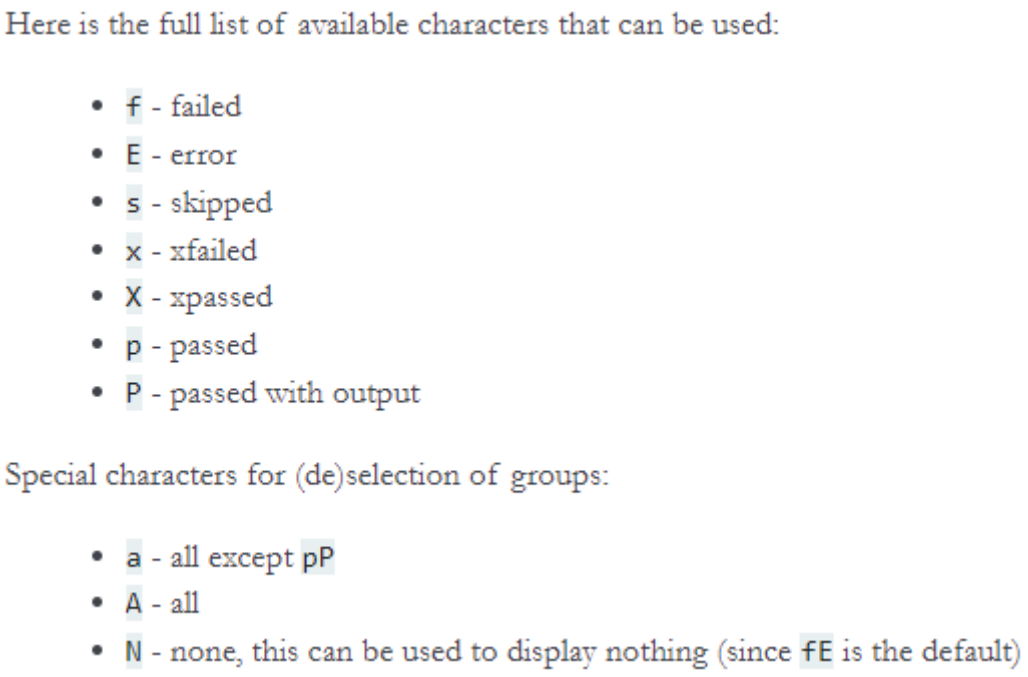
- testata ja käyttää Secure Copy Protocol (SCP) yhteyttä kopioimaan Modbusin testaamista varten kirjoitettu apuohjelma laitteeseen
- varmistaa, että laitteen Ethernet on yhdistettynä
- ottaa Secure Shell (SSH)-yhteys laitteeseen ja varmistaa, että laite palauttaa oikean käyttäjätunnuksen
- varmistaa, että laitteessa USB-tilan vaihto toimii. Laitteessa on kaksi USB-tilaa, modeemi ja automaattinen

- varmistaa, että laitteessa oleva modeemi on mahdollista käynnistää, sekä verkkoon yhdistäminen onnistuu
- yhdistää laite ulkoiseen WLAN reitittimeen sekä yhdistää PC laitteen verkkoon. Tällä testattiin laitteen Access Point (AP) ja Station (STA) tilaa
- varmistaa, että laite saa haettua GPS koordinaatteja
- varmistaa, että laitteesta pystyi lähettämään viestin Modbussille ja Modbus lähetti viestin laitteelle
    - tämä testi vaati erillistä .py tiedostoa, jonka avulla testi toteutettiin
- varmistaa, että laite löytää ulkoisia bluetooth laitteita
- varmistaa CAN-väylän kautta kulkeva kommunikointi
- ajaa SubGHz:tä varten luotu erillinen .py ohjelma, jonka tuloksista tarkistettiin toimivuus.

Testejä suoritettiin yksi kerrallaan tai kaikki yhden kansion sisällä olevat testit järjestyksessä. Testaaminen toimi antamalla komentoriville käsky testin ajamisesta. Yhden testifunktion ajaminen toteutettiin ”pytest -k ”*”- komennolla, jossa tähden kohdalle kirjoitettiin testattavan tiedoston tai funktion nimi. Kaikkien testien ajaminen yhtäjaksoisesti toimi komennolla ”pytest”. Pytestin oletusasetuksena terminaaliin tulostuu lyhyt yhteenvetoraportti, joka sisältää testitiedostojen nimet, epäonnistuneet testit sekä virhekoodit. Testien suunnittelu, koodaus ja testaaminen toteutettiin yksi testitiedosto kerrallaan. Tällä menetelmällä huomattiin virheet sekä virheiden alkuperä nopeammin. Koodaamisen aikana testejä ajaessa käytettiin eri tarkkuutta yhteenvetoraportissa. Pytestiä käyttäessä on useampia käskyjä (kuva 9), joiden avulla muutettiin testien yhteenvetoraportin tarkkuus. Oletuksena pytest tulostaa raporttiin epäonnistuneet testit ja niiden vikakoodin (Pytest 2015). Testien kirjoittamisen aikana lisättiin komentoriville ”-rA” komento, jonka avulla havaittiin testien epäonnistumisen alkuperä.

Testien koodaamisen aikana käytettiin myös Pythonin vikojenetsintä työkalua PDB:tä (Python Debugger). Työssä PDB-työkalua käytettiin yksittäistä testitiedostoa testatessa. Tällöin komentoriville kirjoitettiin komento: ”pytest --pdb test_000_copyhelper.py”. Työkalu avasi komentoriville vikojenetsintä työkalun välittömästi, jos testissä ilmeni ongelmia. Kyseisellä työkalulla hahmotettiin testeistä roikkumaan jääneet funktiot ja toiminnallisuudet. PDB ilmoitti, mihin riville testi pysähtyi ja miksi. Esimerkiksi CAN-väylän ja RS-485:sen testaamista varten käytettiin erillistä Python-ohjelmaa, jonka sammuttaminen vaati manuaalista käskyä. Koska testifunktio jäi

odottamaan ohjelman päättymistä, jäi testi roikkumaan. Tällaiset ongelmat hahmotettiin PDB-työkalun avulla.


Here is the full list of available characters that can be used:

- `f` - failed
- `E` - error
- `s` - skipped
- `x` - xfailed
- `X` - xpassed
- `p` - passed
- `P` - passed with output

Special characters for (de)selection of groups:

- `a` - all except `pP`
- `A` - all
- `N` - none, this can be used to display nothing (since `fE` is the default)

KUVA 9. Pytest yhteenveto raportin komentorivikäskyt (Pytest 2015)

Smoke testauksen suorittamisen jälkeen siirryttiin tekemään tarkempia toiminnallisia testejä. Näiden aikana testattiin laajemmin, toimiiko laitteessa olevat ominaisuudet halutulla tavalla. Testit tehtiin järjestelmällisesti, yksi osa-alue kerrallaan.

4.3.1 WLAN ja Bluetooth

Testattaviin osiin IoT-laitteessa kuului Quectel FC909A WLAN- ja Bluetooth-moduuli. FC909A moduulin yhtenä ominaisuutena oli Dual Stack-tila. Dual Stack-tilalla viitataan, kun STA- sekä AP-tilaa on mahdollista käyttää yhtäaikaisesti. Tämä tarkoittaa sitä, että AP-tilassa laite toimii reitittimen tavoin ja IoT-laitteen verkkoon voi yhdistää ulkoisen laitteen. STA-tilassa laite yhdistää itsensä olemassa olevaan langattomaan verkkoon, samalla tavalla kuin puhelimen voi yhdistää kotona WLAN-verkkoon. Toiminnallisuudessa testattiin, voiko WLAN-moduulista sulkea AP- tai STA-tilan sekä onko laitteen WLAN konfiguraatiot muokattavissa. WLANin hallitseminen laitteessa onnistui applikaation avulla sekä nmcli (Network Manager) komennoilla. FC909A moduulissa on myös mukana Bluetooth ominaisuus. Bluetoothista testattiin ulkoisten Bluetooth laitteiden skannaaminen, yhteyden laadun taso, datavirran laatu sekä onnistuuko Bluetoothin uudelleen

formatointi prosessin sammuttamisen jälkeen. Bluetoothin testaamisen alussa laitteen Bluetooth ominaisuudet formatoitiin hciattach-työkalun avulla. Testaamisessa käytössä oli ulkoinen Bluetooth low energy-mainostaja sekä oma mobiililaite.

4.3.2 Modeemi ja GPS

Laitteessa käytettiin Quectelin EG-21GL LTE-moduulia, joka sisältää myös Global Navigation Satellite System (GNSS)-vastaanottimen. Modeemin ja GNSS-vastaanottimen käynnistäminen ohjattiin laitteelle tehdyn applikaation välityksellä. Modeemista testattiin yhteyden luominen sekä verkon Reference Signal Received Power (RSRP)- ja Reference Signal Received Quality (RSRQ)-tasot. Näiden avulla tarkistettiin verkkoyhteyden laatua, mutta tuloksia kirjatessa huomioitiin laitteen puuttuvien osien vaikutus. EG-21GL kuuluu LTE-kategoriaan LTE Cat 1, jolloin RSRP- ja RSRQ-arvoille oli suuntaa antava taulukko (kuva 10), jonka mukaan testituloksia arvioitiin.

Table

Parameter	Description	Very Poor (Cell Edge)	Poor (Mid Cell)	Good	Very Good (Excellent)
RSRP (dBm)	Average power from a single Reference signal	-140 to -115	-115 to -105	-105 to -95	-95 to -44
RSRQ (dB)	Quality of the received signal	-19.5 to -15	-15 to -10	-10 to -5	-5 to -3
RSSI (dBm)	Total received power including co-channel power and noise	-110 to -90	-90 to -80	-80 to -70	-70 to -44
SINR/SNR (dB)	Ratio of signal power to interference plus noise power	0 to 5	5 to 10	10 to 20	20 to 30

KUVA 10. LTE Cat 1 suuntaa antavat signaali arvot (Haltian Oy 2024)

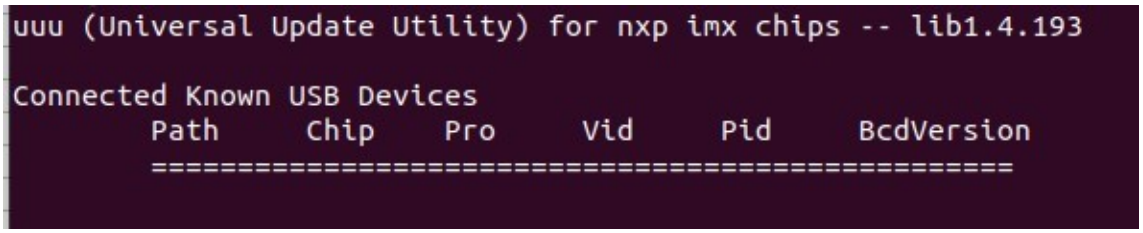
Sekä modeemin, että GNSS-vastaanottimelle annettiin AT-komentoja (liite 4), joiden avulla testattiin ja haettiin moduulista tietoja. Vastaanottimen yhteyksien testaamisessa hyödynnettiin myös mmcli-komentorivityökalua (ModemManager). AT-komentojen käyttäminen toimi UART-rajapinnan yli käyttämällä microcom-terminaalityökalua. AT-komennot yleisesti seuraavat standardisoitua syntaksia, joita on kolmea eri mallia: kirjoittaminen, lukeminen ja testaaminen. Näiden kolmen lisäksi on joissain tapauksissa mahdollista käyttää suoritusmallia. AT-komentojen

rakenne vaihtelee moduulin toimittajan mukaan, esimerkiksi Quectelin AT-komento sisältää Q-kirjaimen. Yleinen rakenne on ”AT+<komento>”, esimerkiksi ”AT+CFG”. Quectelin komennossa rakenne on ”AT+Q<komento>”, eli ”AT+QCFG”. (Ernst 2022.) Liitteessä 3 on listattu yleisimpiä testauksen aikana käytettyjä komentoja.

Kuvassa 6 on näkyvillä laitteesta tietokoneeseen kulkeva UART-kaapeli, jonka avulla luettiin laitteessa kulkevaa sarjaliikennettä. Minicom-terminaalityökalu oli käytössä projektin aikana sarjaliikenteen seuraamiseen, sekä myös SSH-yhteyden luomiseen. GNSS-vastaanottimesta testattiin paikannuksen tarkkuudentaso.

4.3.3 Muut ominaisuudet

IoT-laitteen muista ominaisuuksista testattiin vielä laitteen toimintatila sekä eMMC status. Laitteen toimintatiloja oli kolme: uuu, fastboot sekä operational. Kahta ensimmäistä tilaa käytettiin laiteohjelmiston asennukseen. Tilan tarkastaminen suoritettiin testitietokoneen komentorivillä. Käytettäviin komentoihin sisältyi ”uuu -lsusb”, ”fastboot devices” sekä ”lsusb”. Viimeinen käsky on Linuxin oma komento, jolla voidaan nähdä tietokoneeseen liitetyt laitteet sekä niiden lisätiedot. Laitteen ollessa käyttövalmiina eli operational-tilassa, tulostui laitteen tiedot ”lsusb” komentoa käyttäessä. Komento ”uuu -lsusb” tulostaa komentoriville tietokoneeseen liitetyt laitteet, jotka ovat saatavilla uuu-tilassa. Jos IoT-laite ei ole uuu-tilassa, tulostuu komentoriville kuvan 11 mukainen näkymä. Komento ”fastboot devices” toimii samalla periaatteella.



KUVA 11. UUU-työkalun terminaali näkymä

Viimeisenä ominaisuutena laitteesta tarkistettiin eMMC:n statuksia. Koska eMMC on NAND-flash-muisti, testataan poistosykli lohkojen sekä lohkokokonaisuuksien arvioidun elinkaaren status. NAND-flash-muistilla tarkoitetaan haihtumatonta (non-volatile) muistia, joka ei tarvitse virtaa tietojen säilyttämiselle (Bigelow 2023). Flash-muistissa lohkoihin voidaan kirjoittaa sekä poistaa vain tietyn syklimäärän verran ennen muistin elinkaaren loppua. Statuksien testaamiseen käytettiin Linuxin mmc-työkalua.

4.4 Ei-toiminnallinen testaus

Ei-toiminnallisissa testeissä selvitettiin, yltääkö laiteen kernel ja siihen kuuluvat komponentit haluttuihin tavoitteisiin. Testatessa käytiin läpi suorituskykyä, vikasietokykyä sekä resurssien käyttöä. Testien suunnittelussa tavoitteiden ja rajausten asettamiseen käytettiin projektin alussa kirjoitettua testisuunnitelmaa, sekä komponenttien valmistajien asettamia arvoja.

4.4.1 WLAN ja Bluetooth

FC909A moduulin ei-toiminnalliseen testaamiseen suunniteltiin seitsemän testitapausta. Neljään ensimmäiseen testitapaukseen kuului WLAN-verkon kantavuuden mittaaminen laitteen ollessa Dual Stack-, AP- ja STA-tilassa. Testi toteutettiin toimistotilassa, jossa on useampi WLAN-verkko. Tämän vaikutus otettiin huomioon testaamisen aikana, jolloin testatessa konfiguroitiin laitteen sekä testissä käytetyn reitittimen kanavia. Testaamisessa käytettiin erillistä mobiililaitetta, johon asennettiin PingTools-sovellus. PingTools-sovellusta käytettiin pingaamiseen ja verkkojen voimakkuuksien seuraamiseen. Pingaamisella tässä tapauksessa tarkoitetaan sitä, kun mitataan tietokoneen ja palvelimen välistä viivettä (Kaspersky 2024).

Dual Stack-tilan testaamisessa suoritettiin kaksi erillistä testitapausta, joissa testattiin AP- ja STA-tilan suorituskykyä molempien tilojen ollessa aktiivisena. Vaikka Dual Stack-tilassa kumpaakaan yhteyttä ei sammuteta, testasin AP- ja STA-yhteyden suorituskykyä yksitellen. Kyseisten testitapauksien jälkeen testasin, miten toisen tilan sammuttaminen vaikuttaa kantavuuteen. AP-tilan testissä PingTools-sovelluksella pingattiin laitteen IP-osoitetta, kun taas STA-tilassa yhdistettiin IoT-laite langattomaan reitittimeen. STA-yhteyden testissä laitteen omasta terminaalista pingattiin reitittimen IP-osoitetta. Kaikissa testitapauksissa verkkoon yhdistämisen jälkeen seurattiin pingauksen tuloksia useammalta etäisyydeltä. Tuloksia seurattiin 1–20 metrin etäisyydeltä. WLAN-verkon ominaisuuksista testattiin myös verkkoyhteyden laatu. Tämän testaamisessa hyödynnettiin Linuxin iwconfig-työkalua (kuva 13). Bluetooth:in suorituskyvystä testasin virran kulutusta sekä katkenneesta yhteydestä palautumista. Virran mittaamiseen käytin erillistä jännite monitoria, jonka avulla havainnoitiin reaaliaikaisesti laitteen käyttämää sähkövirtaa.

4.4.2 Modeemi ja GPS

Quectelin EG-21GL LTE-moduulista testattiin ensimmäisenä GPS ominaisuuksia, kuten Time-To-First-Fix (TTFF) ja warm fix. TTFF-arvolla tarkistetaan, kuinka kauan laitteen GNSS-vastaanottimella kestää löytää vähintään kolme GPS-satelliittia. TTFF-arvon laskeminen suoritettiin ensin käynnistämällä laitteen modeemi, GPS-antenni sekä GPS-datan lukeminen. Käynnistämisen jälkeen laitteessa käynnistettiin cgps-työkalu, jolla reaaliaikaisesti tulostettiin kaikki saatavilla oleva GPS-data. Kun laite lukee ensimmäisen kerran GPS-dataa, voidaan suorittaa warm fix ajan testaaminen. Kyseiseen testiin käytettiin myös cgps-työkalua.

Modeemista testattiin laitteen kykyä tunnistaa ulkoinen SIM-kortti. Testaamisessa SIM-kortin tunnistamiseen käytettiin ”AT+CIMI” komentoa. SIM-kortin ollessa paikalla, terminaaliin tulostuu SIM-kortin numero sekä ”CIMI: OK” teksti. Jos SIM-korttia ei ole laitteessa tulee terminaaliin ilmoitus ”CIMI: ERROR”. Testitapauksen tavoitteena on varmistaa laitteen kyky huomata vaurioitunut tai poistettu SIM-kortti.

4.4.3 Muut ominaisuudet

Laitteen ominaisuuksista testattiin myös käynnistymiseen kuluva aika sekä, mikä on Ethernet-yhteyden nopeus. Käynnistykseen kuluvaa aikaa seurattiin laitteen UART-yhteyden avulla. Minicom-terminaalissa sarjaliikenteen mukana tulostui aikaleima, josta tarkkailtiin käynnistyksen kestoa. Ethernet-yhteyden testaamiseen käytettiin iPerf3-työkalua, jota käytettiin laskemaan, kuinka kauan Ethernetin yli liikkuvan datan lähettämiseen ja vastaanottamiseen menee. Testissä IoT-laite toimi iPerf3-serverinä, joka vastaanotti tietokoneelta lähetettyä dataa.

5 TESTITULOKSET

Selkeiden testiraporttien teko kuului osana projektiin. Raportit sisälsivät suoritetut testit sekä saavutetut tulokset. Tulokset raportoitiin Excel-taulukkoon (liite 2, liite 3), sekä TestLinkiin. Testien alussa määriteltiin tavoiteltavat tulokset. Testi kuitattiin onnistuneeksi, jos tulokset vastasivat määriteltyjä arvoja. Testauksessa ajettiin kaikki tarvittavat testit, jonka jälkeen laite ja applikaatio luovutettiin eteenpäin. Seuraaviin lukuihin on jaoteltu mitkä olivat tuotannon -, toiminnallisen - ja ei-toiminnallisen-testauksen tulokset.

5.1 Tuotannontesti

Tuotannontesteissä testattiin eMMC (embedded MultiMediaCard), modeemi, GPS, Bluetooth, LED, WLAN, CAN, Ethernet lähetin (Tx) sekä vastaanotin (Rx), RS-485, SubGHz sekä VIN (Voltage In) raja. Testattavia laitteita oli kahta versiota ja suurin eroavaisuus oli LED:ien määrä sekä RF (Radio Frequency) -suojat. Testejä ajaessa osa laitteista läpäisi tuotannontestin ensimmäisellä yrittämällä. Kuitenkin muutamassa laitteessa havaittiin ongelmia GPS-signaalin, Bluetooth RSSI-tasojen tai Ethernetin kanssa.

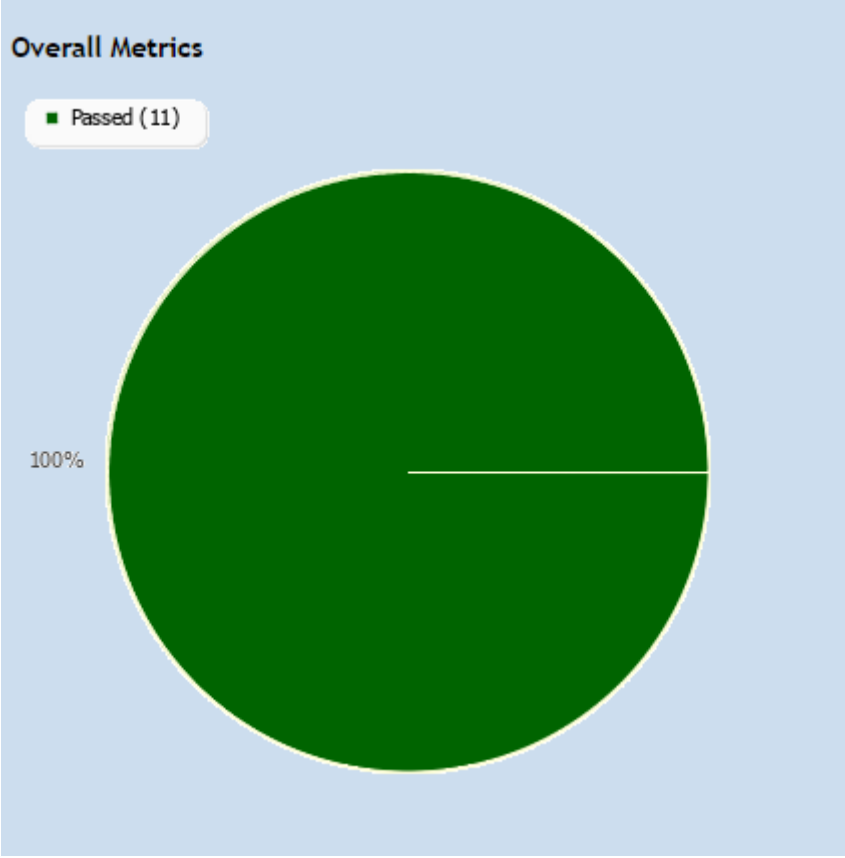
GPS testausta hankaloitti sisätiloissa käytettävän GPS-toistajan toiminta. GPS-toistajan signaalin voimakkuutta sekä paikkaa joutui vaihtamaan manuaalisesti usein, jotta testit menisivät läpi. GPS-toistajan signaalin toistovoimakkuus oli ajoittain liian matala, mutta tasoa nostaessa saattoi signaali nousta liian korkealle. Varovaisella hienosäädöllä kaikki IoT-laitteet läpäisivät GPS-signaalin testit. Vaikka tuotannontestin käyttötarkoituksena oli havaita ongelmia laitteessa, pystyi sillä havaitsemaan myös ongelmia käytettävästä testausympäristöstä. Toisena ongelmana havaittiin satunnaisesti Bluetooth RSSI-tason olevan hyväksyttävän rajan ulkopuolella. Jos testatessa RSSI ylitti -60 dBm rajan, testi epäonnistui. RSSI tasoon vaikutti osasta IoT-laitteista puuttuva RF-suoja. RF-suojan tarkoitus on suojata komponentteja sähkömagneettiselta häiriöltä (Cadence PCB Solutions 2024). Tuotekehitysprojektissa käytettävät lopputuotteet tulisivat kuitenkin kaikki sisältämään RF-suojat, jotta voidaan taata tuotteen toimivuus.

Viimeisimpänä havaittiin ongelma testatessa Ethernetin Tx ja Rx toimintaa. Tämä ongelma ilmeni myöhemmässä vaiheessa, jolloin vian syyksi paljastui viallinen Ethernet kaapeli. Tuotannontestejä

ajaessa, täytyi testeissä käytettävät kytkennät irrottaa ja liittää useamman kerran. Tämän vuoksi Ethernet kaapeli vaurioitui ja testaamisessa käytetty kaapeli vaihdettiin uuteen. Lopputuloksena tuotannontestauksen aikana ei ilmentynyt vakavia ongelmia, jotka vaikuttaisivat projektin etenemiseen.

5.2 Toiminnalliset testit

Toiminnallisten testien tuloksista kirjattiin ylös kolme erillistä Excel-taulukkoa (liite 2, liite 3). Liitteessä 1 on nähtävissä smoke testauksen tulokset. Testejä ajettiin useammalle koodi versiolle, eikä testien aikana ilmennyt vikoja. Excel-taulukon lisäksi TestLinkin raportti osion koostui ympyrädiagrammi (kuva 12) testien tuloksista. Koska smoke testauksessa ajettiin vain yksinkertaisia testejä, ei erillistä raporttia testeistä kirjoitettu.



KUVA 12. Ympyrädiagrammi TestLinkin raportista

WLAN- ja Bluetooth-ominaisuuksien testaamisen aikana havaittiin, että WLAN-verkkojen etsimistä varten käytettävä nmcli-työkalu ei toimi. Tästä raportoitiin virhetiketti Jiraan, jotta muut projektissa saavat tiedon viasta. Syyn alkuperäksi ilmeni FC909A moduulin Dual Stack-ominaisuus.

Luotettavaa tulosta varten, verrattiin työkalun toimivuutta suoritettavan koodin aikaisemmasta versiosta, jossa ei ollut käytössä Dual Stack-ominaisuutta. Testauksen loppuajaksi siirryttiin käyttämään iwlist-työkalua.

Bluetooth yhteyden testaamisessa huomioksi kirjattiin yhteyden näkyvän katkaistuna, vaikka linkki tietokoneeseen saataisiin. Tämä ei kuitenkaan tarkoittanut yhteyden olevan viallinen, koska testi protokollaa ajaessa yhteys laitteiden välillä toimi vaaditulla tavalla. Laitteesta testattiin myös kuuluvuusalueella olevien ulkoisten Bluetooth-laitteiden löytämistä. Bluetooth-laitteiden hakeminen toimi laitteessa, mutta huomioksi kirjattiin ulkoisten laitteiden skannaamisen olevan hidasta. BLE-laitteiden skannaaminen oli nopeampaa, mutta testissä käytettävien laitteiden löytäminen terminaalista oli vaikeampaa suuren tietomäärän vuoksi. Bluetoothin testitapauksiin kuului myös Bluetooth-asetuksien uudelleen konfigurointi. Bluetooth-rajapinnan käynnistämisen jälkeen konfiguroidaan laitteeseen tarvittavat tiedot. Tämä konfigurointi käynnistää prosessin laitteen taustalle. Uudelleen konfigurointia varten käynnistynyt prosessi on lopetettava tai laite sammutettava. Testillä varmistettiin prosessin hallittu alas ajaminen, jonka jälkeen laite oli mahdollista konfiguroida uudelleen. FC909A moduulin testaamisessa tulleet huomiot kirjattiin ylös ja raportoitiin eteenpäin, mutta muutoin moduulin testitapauksien tulokset olivat vaatimusten mukaiset.

GPS:n toiminnallisessa testaamisessa keskityttiin sijainnin hakemiseen ja sen tarkkuuteen. Tarkkuutta testattiin ajamalla laitteen oman applikaation GPS-datan haku ominaisuutta. Applikaatio tulosti terminaaliin saadun GPS-datan yhteyden laadun, kellonajan sekä koordinaatit. Tarkkuus laskettiin vertailemalla laitteen ilmoittamia koordinaatteja GPS-simulaattorin lähettämiin koordinaatteihin. Eroavaisuuden laskemisessa otettiin huomioon simulaattorin näyttävän koodinaatit asteina ja desimaaliminuutteina (DDM), mutta IoT-laite ilmoittaa saadut koordinaatit desimaaliasteina (DD). GPS-signaalin paikannuksen vaatimus oli päästä 100 metrin tarkkuuteen. Laitteen GPS-paikannus ilmoitti sijainnin viiden metrin tarkkuudella.

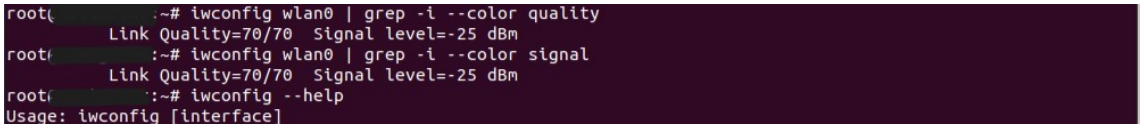
Modeemin LTE-verkosta testattiin yhteyden muodostaminen modeemin käynnistyksen jälkeen, sekä yhteydenlaatu. Testaamisessa käytettiin ”AT+CREG” sekä ”AT+QENG=”servingcell”” komentoja. CREG-komennolla varmistettiin, onko laite muodostanut yhteyden verkkoon. QENG-komennolla testattiin, mikä on muodostetun yhteyden status. Yhteyden muodostamisessa ei havaittu ongelmia, mutta RSRP- sekä RSRQ-arvot olivat heikot. Hyvää yhteyttä varten RSRQn on oltava yli -10 dB ja RSRP yli -105 dBm. Testaamalla parhaimmat mitatut arvot olivat -11 dB ja -109

dBm, jotka viittasivat heikkoon yhteyteen. Heikko yhteys voi johtua monesta erinäisestä syystä. Yleisimpiä syitä on etäisyys läheisimpään tukiasemamastoon, testaaminen ympäristössä, joissa on useampi aktiivinen laite sekä laitteen altistuminen häiriölle. Tässä tapauksessa testissä käytetystä laitteesta puuttui suojaavia osia ja tämä vaikutti testin lopputulokseen. Viimeisenä toiminnallisessa testauksessa varmistettiin laitteen toimintatilojen toimiminen sekä flash-muistin status, eikä kyseisissä testitapauksissa havaittu ongelmia.

5.3 Ei-toiminnalliset testit

FC909A-moduulin seitsemästä testitapauksesta vain viidestä saatiin vaaditut lopputulokset. WLAN-verkon STA-yhteyden kantavuudessa Dual Stack-tilassa havaittiin 20 metrin etäisyydellä 10 %:n paketti hävikki ja yhteyden keskiarvo Round Trip Time (RTT) oli 13.445 ms. Samalla etäisyydellä AP-tilan ollessa sammutettuna tulokseksi saatiin 5 %:n paketti hävikki ja RTT keskiarvoksi 11.511 ms. Testiympäristön huomioon ottaen STA-yhteyden tulokset vastasivat vaadittuja arvoja. AP-yhteyden kantavuus ei yltänyt vaaditulle tasolle. Dual Stack-tilassa metrin etäisyydellä ilmeni 10 %:n paketti häviötä ja kuudessa metrissä paketti häviön määrä kasvoi 33 %:iin. STA-tilan sammuttamisen jälkeen havaittiin noin kahdeksan metrin etäisyydellä 25 %:n häviö, sekä laskettu RTT-arvo oli 180 ms.

Testien tuloksien tarkkuus todistettiin ajamalla testejä eri WLAN-kanavilla. Tällä suljettiin testiympäristön vaikutus testien lopputuloksiin. Tarkemmat mittatulokset raportoitiin erilliseen tiedostoon (kuva 14, kuva 15) myöhempää vertailua varten. Kantavuuden ongelma johtui moduulin Dual Stack-ominaisuudesta, ja tämä huomioitiin kehitysprojektin jatkovaiheissa. Kuvassa 13 näkyy WLAN-verkon yhteyden laadun testauksessa saatuja tuloksia, kun IoT-laite yhdistettiin ulkoiseen WLAN-reitittimeen.



KUVA 13. iwconfig-työkalun näkymä WLAN testin aikana

Bluetooth ominaisuuden ei-toiminnalliset testit olivat yksinkertaiset projektin laadun ja IoT-laitteen Bluetooth vaatimuksien vuoksi. Virran kulutuksen testissä ei ollut havaittavissa mitään poikkeavaa. Katkenneen Bluetooth yhteyden testaamisessa täytyi varmistaa, miten laitteessa Bluetooth

ominaisuutta on tarkoitus käyttää sekä miten laite käyttäytyy yhteyden katketessa. Bluetooth yhteyttä varten avattiin RFCOMM-yhteys, jonka avulla laitteiden välinen data liikkui. Yhteyden palautumista testatessa katkaistiin tietokoneen Bluetooth, jonka vuoksi tietokoneen terminaaliin avattu yhteys katkesi. IoT-laitteeseen ilmestyi ilmoitus katkenneesta yhteydestä, jolloin laite jäi odottamaan data liikennettä. Tietokoneen Bluetoothin uudelleen käynnistämisen jälkeen avattiin uusi yhteys, johon laite yhdistyi uudestaan välittömästi.

```
AP test:
DUAL MODE
1 m - 10% packet loss rtt min/avg/max/mdev = 2/27/73/22.4 ms
6 m - 33% packet loss rtt min/avg/max/mdev = 29/34/38/4.1 ms

SINGLE MODE
6 m - 0% packet loss rtt min/avg/max/mdev = 6/26/62/20.2 ms
8 m - 25% packet loss rtt min/avg/max/mdev = 30/180/1314/425.4 ms
```

KUVA 14. AP-yhteyden mitatut arvot

```
STA test:
DUAL MODE
+20 m - 10% packet loss rtt min/avg/max - 2.87./13.445/84.502

SINGLE MODE
+20 m - 5% packet loss rtt min/avg/max/ - 1.812/11.511/37.669
```

KUVA 15. STA-yhteyden mitatut arvot

Modeemissa SIM-kortin lukemisen testeissä ei ollut havaittavissa mitään poikkeavaa. IoT-laite reagoi vaaditulla tavalla SIM-kortin poistamiseen sekä asentamiseen. GNSS-vastaanottimen warm fix- ja TTFF-testit suoritettiin useampaan otteeseen, jotta tulokset olisivat luotettavat. Testien aikana GPS-simulaattorissa käytettiin samaa skenaariota testituloksien vertailemista varten. Molemmat testitapaukset toistettiin onnistuneesti, eikä GNSS-vastaanottimen toiminnassa ollut havaittavissa ongelmia. Myöskään IoT-laitteen käynnistysajan tai Ethernetin testitapauksissa ei ollut havaittavissa puutoksia. Ei-toiminnallisen testien lopputulokset raportoitiin TestLinkiin sekä Excel-taulukkoon (liite 3).

6 POHDINTA

Opinnäytetyössä käsiteltiin järjestelmätestausta tuotekehitysprojektissa. Työn aikana IoT-laitteelle ja sen ohjelmistolle suoritettiin 36 eri testitapausta. Lopputuloksena löytyi yksi WLAN-verkon suorituskykyyn vaikuttava ominaisuus, joka huomioitiin tuotteen jatkokehityksessä.

Työn alussa esitettiin kysymykset ”Mitä ohjelmistotestauksen eri menetelmiä tuotekehitysprojekti sisältää?” sekä ”Mitä työkaluja järjestelmätestauksen suunnittelu ja toteutus tuotekehitysprojektissa vaatii?”. Ensimmäiseen kysymyksen vastausta pohjustettiin teoriaosuudessa, jossa esiteltiin tunnetuimpia testausmenetelmiä sekä yleisimpiä työkaluja. Käsitteenä järjestelmätestaus voi tarkoittaa useammalle ihmiselle eri asioita. Tämä korostui teoria osuutta koostaessa, kun yhdellä termillä on mahdollista kuvata useampaa asiaa. Myös testaamisessa käytettyjä menetelmiä ja vaiheita on useampia. Tämä teki tutkimuksesta mielenkiintoista sekä haastavaa. Tässä opinnäytetyössä testauksessa käytettiin pääosin grey box- menetelmää. Tämä johtui siitä, että testattavana oli kernel-tason ominaisuudet sekä ulkoa käsiteltävät rajapinnat. Tuotekehitysprojektissa käytetyt työkalut määräytyivät projektin sekä toimeksiantajan mukaan.

Työn toteutus eteni alkuun hitaammin johtuen aiemman testauskokemuksen puutteesta. Testitapausten suunnittelemista kuitenkin auttoi selkeä testisuunnitelma. Testisuunnitelmassa näkyi kirjallisena rajatut tehtävät sekä laitteen ominaisuuksien vaatimukset. Näiden avulla määritettiin testitapauksien laajuus sekä saavutettavat tavoitteet. Vaikka ominaisuuksille on mahdollista tehdä laajempia testitapauksia, olivat työssä kirjatut tapaukset vaatimusten mukaiset. Ilman selkeää dokumentointia testitapausten määrä sekä vaaditut tulokset olisi ollut miltei mahdotonta määritellä.

Testien toteutuksen aikana haettiin jatkuvasti lisää tietoja testattavista ominaisuuksista sekä esimerkkejä testaamisessa käytettyihin työkaluihin. Kokonaisuudessaan opinnäytetyössä vastattiin alussa esitettyihin kysymyksiin tarkasti ja työn toteutukseen sisällytettiin tuotekehitysprojektin järjestelmätestauksen eteneminen.

LÄHTEET

Android Developers 2024. SDK Platform Tools release notes. Android Studio. Hakupäivä 24.4.2024. https://developer.android.com/tools/releases/platform-tools.

Arora, Simran Kaur 2024. Test Plan in Software Testing: Types and Steps to Create One. Knowledgehut. Hakupäivä 8.4.2024. https://www.knowledgehut.com/blog/software-testing/test-plan-in-software-testing.

Atlassian 2024. Welcome to Jira. Atlassian. Hakupäivä 15.4.2024. https://www.atlassian.com/software/jira/guides/getting-started/introduction#what-is-jira-software.

Awati, Rahul 2021. Stream Control Transmission Protocol (SCTP). TechTarget. Hakupäivä 24.4.2024. https://www.techtarget.com/searchnetworking/definition/SCTP.

Awati, Rahul 2022. Integration testing or integration and testing (I&T). TechTarget. Hakupäivä 5.4.2024 https://www.techtarget.com/searchsoftwarequality/definition/integration-testing.

Bigelow, Stephen J. 2023. NAND flash memory. TechTarget. Hakupäivä 13.5.2024. https://www.techtarget.com/searchstorage/definition/NAND-flash-memory.

Black, Rex 2007. Pragmatic Software Testing. Wiley Publishing, Inc. Indianapolis.

Cadence PCB Solutions 2024. The Importance of RF Interference Shielding. Cadence. Hakupäivä 26.4.2024. https://resources.pcb.cadence.com/blog/2022-the-importance-of-rf-interference-shielding.

Codium AI 2024. OAT (Operational Acceptance Testing). Codium AI. Hakupäivä 11.4.2024. https://www.codium.ai/glossary/oat-operational-acceptance-testing/.

Collins, Tom 2022. What is Acceptancce Testing? (Importance, Types & Best Practices). Browserstack. Hakupäivä 11.4.2024. https://www.browserstack.com/guide/acceptance-testing.

Dugan, John, Elliot, Seth, Mah Bruce A., Poskanzer Jeff, Prabhu Kaustubh 2015. What is iPerf / iPerf3?. iPerf. Hakupäivä 24.4.2024. https://iperf.fr/.

Ernst, Constantin 2022. AT Commands Guide for GSM. Emnify. Hakupäivä 23.4.2024. https://www.emnify.com/developer-blog/at-commands-for-cellular-modules#what-are-at-commands.

Farooq, Sheikh Umar, Quadri, S.M.K 2011. 3W's of Static Software Testing Techniques. Global Journal Of Computer Science & Technology. Hakupäivä 15.4.2024. https://globaljournals.org/GJCST_Volume11/8-3Ws-of-Static-Software-Testing-Techniques.pdf.

GeeksforGeeks 2022. Difference between Functional and Non-functional Testing. GeeksforGeeks. Hakupäivä 5.4.2024 https://www.geeksforgeeks.org/differences-between-functional-and-non-functional-testing/.

GeeksforGeeks 2023a. Test plan – Software Testing. GeeksforGeeks. Hakupäivä 8.4.2023. https://www.geeksforgeeks.org/test-plan-software-testing/.

GeeksforGeeks 2023b. Unit Testing – Software Testing. GeeksforGeeks. Hakupäivä 4.4.2024. https://www.geeksforgeeks.org/unit-testing-software-testing/.

GeeksforGeeks 2023c. White box Testing – Software Engineering. GeeksforGeeks. Hakupäivä 4.4.2024. https://www.geeksforgeeks.org/software-engineering-white-box-testing/.

Haltian Oy 2024a. QA process instructions. Sisäinen lähde.

Haltian Oy 2024b. Software Testing Plan. Sisäinen lähde.

Holota, Olha 2023b. How to Create a Test Plan. Medium. Hakupäivä 8.4.2024. https://medium.com/@case_lab/how-to-create-a-test-plan-77874e88888f.

Imperva 2024a. Black Box Testing. Imperva. Hakupäivä 15.4.2024. https://www.imperva.com/learn/application-security/black-box-testing/.

Imperva 2024b. Gray Box Testing. Imperva. Hakupäivä 15.4.2024. https://www.imperva.com/learn/application-security/gray-box-testing/.

Kaspersky 2024. Ping-arvon pienentäminen ja verkkopelien suorituskyvyn optimoiminen. Kaspersky. Hakupäivä 30.4.2024. https://www.kaspersky.fi/resource-center/preemptive-safety/how-to-improve-game-performance.

Kasurinen, Pekka Jussi 2013. Ohjelmistotestauksen käsikirja. Saarijärven Offset Oy, Saarijärvi.

Khandelwal, Abhik 2019. Difference between Black Box and White Box Testing. Testing Genez. Hakupäivä 12.04.2024. https://testinggenez.com/black-box-and-white-box-testing/.

Microsoft 2023. What is Visual Studio? Microsoft. Hakupäivä 15.4.2024. https://learn.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide?view=vs-2022.

Natchimuthu, Akil 2022. 7 Types of UAT: Definition, Best practices and FAQs. Disbug. Hakupäivä 10.4.2024. https://disbug.io/en/blog/types-of-uat.

Owasp 2024. Source Code Analysis Tool. Owasp. Hakupäivä 15.4.2024. https://owasp.org/www-community/Source_Code_Analysis_Tools.

Patrizio, Andy 2021. blue screen of death (BSOD). TechTarget. Hakupäivä 19.4.2024. https://www.techtarget.com/searchwindowsserver/definition/blue-screen-of-death-BSOD.

PractiTest Team 2024. Alpha Testing vs Beta Testing. PractiTest. Hakupäivä 12.04.2024. https://www.practitest.com/resource-center/article/alpha-testing-vs-beta-testing/.

Pytest 2015. Managing pytest's output. Pytest. Hakupäivä 29.04.2024. https://docs.pytest.org/en/7.1.x/how-to/output.html.

QATestLab 2020. Black-Box Testing: Algorithm & Methods. QA TestLab. Hakupäivä 12.04.2024. https://qatestlab.com/resources/knowledge-center/black-box-testing/.

Rice, Randall W. 2024. What is a Test Plan? The Complete Guide for Writing a Software test Plan. PractiTest. Hakupäivä 12.4.2024. https://www.practitest.com/resource-center/article/write-a-test-plan/.

SHIFT ASIA 2020. Difference between V-model and W-model in software testing. Medium. Hakupäivä 8.4.2024. https://medium.com/qa-moments/difference-between-v-model-and-v-model-in-software-testing-2b5f4f6ff575.

Shyniaieva, Kateryna 2024. The Most Popular IDEs for Developers in 2024. Outstaff Your Team. Hakupäivä 15.4.2024. https://outstaffyourteam.com/articles/most-popular-ides-for-developers.

Singh, Harpreet 2023. Top 30 Bug Tracking Software Platforms for 2024. Launchable. Hakupäivä 15.4.2024. https://www.launchableinc.com/blog/top-30-bug-tracking-software-platforms-for-2024/.

Stackoverflow 2023. 2023 Developer Survey. Stackoverflow. Hakupäivä 15.4.2024. https://survey.stackoverflow.co/2023/.

Testim 2022. Test Automation Tool: Definition and 5 Best Ones. Testim. Hakupäivä 15.4.2024. https://www.testim.io/blog/what-is-a-test-automation-tool/.

Testsigma 2023. What, Why, Types & how to Do? Teststigma. Hakupäivä 8.4.2024. https://testsigma.com/guides/acceptance-testing/#What_is_Acceptance_Testing.

Tuleap 2024. Software Quality: understanding the different types of software testing. Tuleap. Hakupäivä 25.4.2024. https://www.tuleap.org/software-quality-different-types-software-testing.

Uspenski, Anastasija 2023. Why VS Code remains a developer favorite, year after year. ShiftMag. Hakupäivä 15.4.2024. https://shiftmag.dev/vs-code-171/.

Valagroup 2022. Mitä on ohjelmistotestaus ja mitä hyötyä siitä on. Valagroup Hakupäivä 4.4.2024. https://www.valagroup.com/fi/blogi/mita-on-ohjelmistotestaus-ja-mita-hyotya-siita-on/.

Valagroup 2023a. Integraatiotestaus: Mitä se tarkoittaa ja mitä hyötyä siitä on? Valagroup. Hakupäivä 5.4.2024 https://www.valagroup.com/fi/blogi/integraatiotestaus/.

Valagroup 2023b. Opas testaussuunnitelman laatimiseen ja ilmainen malli. Valagroup. Hakupäivä 12.4.2024. https://www.valagroup.com/fi/blogi/opas-testaussuunnitelman-laatimiseen-ja-ilmainen-malli/.

Valagroup 2023c. Yksikkötestaus: mitä se on ja miksi se on tärkeää?. Valagroup. Hakupäivä 12.4.2024. https://www.valagroup.com/fi/blogi/yksikkotestaus/.

Wasay, Abdul 2024. Top Most Popular IDE Picks for Developers in 2024. SkillReactor. Hakupäivä 19.4.2024. https://www.skillreactor.io/blog/best-most-popular-ides/.

Zieniūtė, Ugnė 2024. Mitä TCP ja UDP ovat? Yksinkertainen selitys. NordVPN. Hakupäivä 24.4.2024. https://nordvpn.com/fi/blog/tcp-udp-protokolla/.

LIITTEET

Haltian Oy SW testisuunnitelma mallin sisällysluettelo liite 1

Smoke testauksen tulokset liite 2

Järjestelmätestauksen tulokset liite 3

AT-komennot liite 4

Software Test Project Plan

SMOKE TESTAUKSEN TULOKSET

LIITE 2

Smoke test			
Test Case	**Build**	**Expected result**	**Actual result**
test_000_copyhelper.py	2024.03.28.3_rnd	Device gets help.py	help.py in device
test_001_ethernet.py	2024.03.28.3_rnd	successfull ping	succesfull ping
test_002_ssh.py	2024.03.28.3_rnd	remote connection to device	connection successfull
test_003_usb.py	2024.03.28.3_rnd	lsusb returns device	device found on lsusb
test_004_modem.py	2024.03.28.3_rnd	modem echo 0	modem echo 0
test_005_wifi.py	2024.03.28.3_rnd	device finds wifi networks	device finds wifi networks
test_006_gps.py	2024.03.28.3_rnd	gps location found	gps location found
test_007_rs485.py	2024.03.28.3_rnd	modbus sends and receives	modbus sent and received
test_008_bluetooth.py	2024.03.28.3_rnd	bluetooth scan returns devices	other BT devices found
test_009_can.py	2024.03.28.3_rnd	CAN bus sends and receives string	CAN bus sent and received string
test_010_SubGHz.py	2024.03.28.3_rnd	SubGHz test.py echo 0	SubGHz test.py echo 0

JÄRJESTELMÄTESTAUKSEN TULOKSET

LIITE 3

F/N	Test Case	Build	Preconditions	Expected result	Actual result	Notes
F	WiFi on STA mode	2024.04.25.1_rnd	WiFi on	wlan1 down	wlan1 down	
F	WiFi on AP mode	2024.04.25.1_rnd		wlan0 down	wlan0 down	
F	WiFi channel	2024.04.25.1_rnd		device iwlist scan	device iwlist scan	
F	BT data flow	2024.04.25.1_rnd	Bluetooth on	RFCOMM watch	RFCOMM watch	
F	Verify BT scanning	2024.04.25.1_rnd	Bluetooth on	hcitool returns	hcitool returns	
F	Reinitialize BT	2024.04.25.1_rnd	Bluetooth	hciattach stops	hciattach stops	
F	Verify LTE network	2024.04.25.1_rnd	Modem on.	CREG: 0,1	CREG: 0,1	new SIM -> CREG:
F	Verify network quality status LTE	2024.04.25.1_rnd	Modem on.	acceptable level (RSRQ -10 to -3 &&	RSRQ = -11 | RSRP = -109	Tested device not in real mechanics
F	GPS location	2024.04.25.1_rnd	GPS on, GPS	location within	location within 5	
F	Read eMMC status	2024.04.25.1_rnd	Device on	PRE-EOL → 0x01	PRE-EOL → 0x01	
F	Set device to uuu	2024.04.25.1_rnd		uuu -lsusb returns	uuu -lsusb returns	
F	Set device to fastboot flashing	2024.04.25.1_rnd		fastboot devices returns device	fastboot devices returns device	
N-F	Remove SIM	2024.05.15.1_rnd	Modem on.	CIMI: ERROR	CIMI: ERROR	
N-F	SIM attached	2024.05.15.1_rnd	Modem on.	CIMI: OK	CIMI: OK	
N-F	WiFi STA mode	2024.05.15.1_rnd	WiFi on, AP mode	range over 10m	range over 10m	
N-F	WiFi AP mode	2024.05.15.1_rnd	WiFI on, STA down	range over 10m	range under 6m	Dual Stack issue
N-F	WiFi STA range in	2024.05.15.1_rnd	WiFi on, AP & STA	range over 10m	range over 10m	
N-F	WiFi AP range in	2024.05.15.1_rnd	WiFI on AP & STA	range over 10m	range under 10m	Dual Stack issue
N-F	WiFi quality tests	2024.05.15.1_rnd	WiFi on.	signal strength over -70 dBm, link	-30 dbm, 70/70, max rtt 84 ms	
N-F	Verify GPS TTFF	2024.05.15.1_rnd	GPS on, GPS	TTFF under 1 min	TTFF 30 sec	
N-F	GPS warm fix	2024.05.15.1_rnd	GPS on, GPS	TTFF under 3 min	TTFF 2 min	
N-F	BT connection	2024.05.15.1_rnd				
N-F	Device boot up	2024.05.15.1_rnd		Under 1 min	19 s	
N-F	Ethernet data transfer speed	2024.05.15.1_rnd		min bitrate 1 Gbit/sec	send bitrate 5.76 Gbits/sec | receive	

AT COMMANDS

A/	Repeat previous command
AT+CIMI	IMSI
AT+CREG?	Network Registration Status
AT+CSQ	RSSI & Channel bit error rate
AT+QNWINFO	Query Network Information
AT+QENG="servingcell"	Query the information of serving cells
AT+GMM	Modem model
AT+GMR	Modem Firmware
AT+GSN	IMEI
AT+QGPS?	GPS enabled
AT+QGPSEND	GPS disabled