Opinnäytetyö (AMK)

Tieto- ja viestintätekniikka

2021

Jimi Österholm

RADIOMODEEMIEN TESTIASETUSTEN TUOTTAMINEN GRAAFISEN KÄYTTÖLIITTYMÄN AVULLA

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tieto- ja viestintätekniikka

2021 | 52 sivua

Jimi Österholm

RADIOMODEEMIN TESTIASETUSTEN TUOTTAMINEN GRAAFISEN KÄYTTÖLIITTYMÄN AVULLA

Radioreitittimiä käytetään maailmanlaajuisesti eri teollisuuskohteissa. Radioreitittimien tehtävä on varmistaa luotettava ja toimintavarma tiedonsiirto. Esimerkkejä radioreitittimien käyttökohteista ovat SCADA, koneohjaus, GNSS, ympäristömonitorointi ja teollinen internet.

Radioreitittimien ohjelmistotestausta varten Satel on luonut automaatiotestausjärjestelmän, joka pohjautuu Jenkins-automaatiopalvelimeen ja sen avulla ajettaviin Robot Framework -testeihin. Testeissä on mahdollista asettaa laitteisiin asetuksia JSON-tiedostojen avulla.

Radioreitittimien laitteistoasetuksia sisältävät JSON-tiedostot luodaan komentorivipohjaisella JavaScript-ohjelmistolla. JavaScript-ohjelmiston käyttö vaatii asetuksien kirjoittamisen erilliseen CSV-taulukkoon. Kirjoitettuja asetuksia ei tarkisteta JSON-tiedoston luomisen aikana ja se mahdollistaa virheelliset asetukset. Virheet huomataan usein vasta testivaiheessa. Testit epäonnistuvat virheellisien asetuksien takia.

Opinnäytetyössä toteutettiin graafista käyttöliittymää hyödyntävä sovellus. Sovelluksella korvattiin aiempi käytössä oleva radioreitittimien laiteasetuksien luomiseen käytetty komentorivipohjainen JavaScript-ohjelmisto. Toteutettu sovellus luotiin Python-ohjelmointikielen ja PySimpleGUI-kirjaston avulla. Python valittiin ohjelmointikieleksi sen monipuolisuuden, sekä useiden olemassa olevien kirjastojen takia.

Radioreitittimien testiasetusten luomiseen kehitetyssä Python-sovelluksessa mahdollistettiin laitteistoasetuksien valitseminen listoista. Käyttäjän valinnoista syntyvät asetuskombinaatiot tarkistetaan ja virheelliset asetukset jätetään JSON-tiedostojen luomisprosessissa huomioimatta. Sovellukseen lisättiin mahdollisuus kirjata asetuksia myös käsin. Käsin kirjoittamista varten luotiin erillinen ikkuna, jossa mahdollistettiin harvinaisempien ja tuotekohtaisempien asetuksien käyttö. Sovellus viimeisteltiin PyInstallerin avulla ja siitä paketoitiin yksittäinen erikseen ajettava exe-tiedosto. Tämän exe-tiedoston avulla sovellusta on mahdollista käyttää itsenäisesti, ilman ohjelmistoriippuvuuksia.

Kokonaisuutena työssä päästiin asetettuun tavoitteeseen. Sovelluksen käytettävyyden ja toimintojen kehitys jatkuu opinnäytetyön jälkeen.

ASIASANAT:

Python, graafinen käyttöliittymä, JSON, radiotekniikka, testiautomaatio

BACHELOR´S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information and Communications Technology

2021 | 52 pages

Jimi Österholm

GENERATING RADIO MODEM TEST SETTINGS USING A GRAPHICAL INTERFACE

Radio modems are used worldwide in industries to ensure reliable data transmission between different applications. The main use applications of radio modems include SCADA, automation, GNSS, environmental monitoring, and the industrial internet.

The reliability of radio modems must be tested in various ways. Testing devices manually is time-consuming, but testing is possible to be mostly automated.

Satel has created a test automation platform for radio modem firmware testing. The platform is based on a Jenkins automation server and it uses various tests written with Robot Framework. Radio modem device settings used in these tests can be set using JSON files.

JSON files used in radio modem testing are created using a terminal-based JavaScript program. Using this program requires the user to write device settings manually in a CSV file, which often leads to mistakes. Written settings are not automatically checked for faultiness during JSON file generation. Faulty settings in JSON files lead to failed tests.

The main idea behind this thesis was to create an application, that uses a user interface. The created application replaces the terminal-based JavaScript program. The application was created using Python programming language and its PySimpleGUI library. Python was chosen as the programming language because of its diversity and multiple existing libraries.

The Python program created for radio modem test setting generation made it possible to choose device settings from lists. The combinations of the chosen settings are automatically checked for faultiness. When faulty settings are found, they are discarded during JSON file generation. A window containing manual functionality similar to the old JavaScript-based program was also created for more product-specific settings. The program was finalized using PyInstaller and it was packaged into one executable file. The executable file allowed the program to be used independently, without any software dependencies.

The set requirements for the thesis were achieved successfully. The development of the program, its user interface, and functionalities continue after this thesis.

KEYWORDS:

Python, graphical interface, JSON, radiotechnology, test automation

SISÄLTÖ

KUVAT

TAULUKOT

KÄYTETYT LYHENTEET

CSV	Tiedostomuoto yksinkertaisen taulukkomuotoisen tiedon tallentamiseen. (Comma-Separated Values)
CTS	Sarjaliikenteessä käytettävä toiminto, joka varmistaa suurempien pakettien lähetyksen ja vastaanoton. Vastaanottava laite ilmoittaa, kun se on valmis vastaanottamaan dataa. (Clear To Send)
FEC	Tiedonsiirrossa virheenkorjaukseen käytettävä teknologia, joka korjaa lähetyksessä ilmeneviä virheitä ilman uudelleenlähettämisen tarvetta. (Forward Error Correction)
FGPA	Kentällä erikseen kustomoitava integroitu piiri. Kustomointi/ohjelmointi tapahtuu pääosin Hardware Description Language (HDL), ohjelmointikieltä käyttäen. (Field-Programmable Gate Array)
HTML	Standardoidu selainten avulla avattavien sivustojen näyttämiseen käytettävä merkkauskieli. (HyperText Markup Language)
JSON	Yksinkertainen tiedon tallentamiseen ja siirtämiseen käytetettävä tiedostonmuoto. Sen sisältämä teksti on helposti luettavaa. (JavaScript Object Notation)
RTS	Sarjaliikenteessä käytettävä toiminto, joka varmistaa suurempien pakettien lähetyksen ja vastaanoton. Lähettävä laite kysyy, onko vastaanottava laite valmis tiedonsiirtoon. (Request To Send)
RSSI	Arvioitu mittaus radiosignaalia vastaanottavan laitteen signaalinvahvuudesta. (Received Signal Strength)
UART	Konfiguroitava sarjaliikenneväylä tiedonsiirtoon. Yleensä osana integroituja piirejä. (Universal Asynchronous Receiver-Transmitter)

1 JOHDANTO

Radiomodeemit mahdollistavat lyhyen, sekä pitkän kantaman tietoyhteydet luotettavasti. Tietoyhteyksien luotettavuus on otettava huomioon erilaisissa teollisuusympäristöissä, joissa laitteistojen sisäiset vikatilat voivat katkenneen tietoyhteyden seurauksena jäädä huomaamatta ja korjaamatta. Luotettavan tiedonsiirron lisäksi radiomodeemit mahdollistavat yksityisen radioverkon muodostamisen. Yksityisen radioverkon käyttäjä ei ole riippuvainen tietoyhteyksiä tarjoavista kaupallisista operaattoreista. (Satel 2021a.)

Radiomodeemien luotettavuus on testattava monin tavoin. Laitteiden testaaminen manuaalisesti vaatii paljon aikaa, mutta nykypäivänä testaus on mahdollista pääosin automatisoida. Satelin radiomodeemien ohjelmistotestauksessa käytettävä testiautomaatiojärjestelmä perustuu Jenkins-automaatiopalvelimeen, sekä Robot Framework-kirjastoa käyttäviin testeihin. Reiman on vuonna 2017 kertonut Satelin siirtymisestä testiautomaation käyttöön Pro gradu -teoksessaan ”Radiomodeemin ohjelmistotestauksen automatisointi”. Testiautomaatiojärjestelmä on kuitenkin muuttunut ja kehittynyt teoksen valmistumisen jälkeen ja tämän opinnäytetyön kehittämisen aikana.

Satelin luomassa testiautomaatiojärjestelmässä radiomodeemeihin on mahdollista asettaa asetuksia JSON-päätteisillä tiedostoilla. JSON-tiedostojen tuottamista varten Satel on kehittänyt komentorivipohjaisen JavaScript-sovelluksen. Sovellus vaatii käyttäjältä paljon manuaalista työtä. Asetukset kirjataan erilliseen CSV-taulukkoon ja siihen kirjoitettuja mahdollisia virheitä ei tarkisteta. Asetuksissa ilmenevät virheet huomataan usein vasta testivaiheessa. Virheet aiheuttavat turhaan epäonnistuneita testejä.

Tämän opinnäytetyön tavoitteena on luoda uusi ja helppokäyttöisempi sovellus vanhan tilalle. JavaScript-ohjelmointikielestä luovutaan. Uusi sovellus tuotetaan Python-ohjelmointikielellä sen monipuolisuuden ja useiden saatavilla olevien kirjastojen takia. Vanha komentorivipohjainen toiminta poistetaan. Komentorivipohjaisuuden tilalle luodaan graafinen käyttöliittymä PySimpleGUI-kirjaston avulla. Käyttöliittymään luodaan kätevät valintaruudut asetuksille. Käyttäjän valintojen yhteensopivuus tarkistetaan ja virheelliset asetukset jätetään huomioimatta. Käyttäjälle luodaan mahdollisuus tallentaa ja ladata asetuksia myöhempää käyttöä varten. Kustomointia varten luodaan oma ikkuna, joka mahdollistaa asetuksien syöttämisen manuaalisesti. Lisäksi ohjelmassa mahdollistetaan asetustiedostojen nimeäminen, sekä niiden sisällön helppo tunnistaminen. Valmiista ohjelmasta luodaan suoritettava exe-tiedosto PyInstaller-sovelluksen avulla.

2 SATEL OY-YRITYS

Satel Oy on Salossa sijaitseva yksityinen, radioteknisiä ratkaisuja tuottava yritys. Satel valmistaa ja myy laadukkaita, yksityiseen radioverkkoihin tarkoitettuja radiolaitteita, jotka varmistavat turvallisen ja luotettavan tietoyhteyden. Valmistetut tuotteet ovat tyyppihyväksyttyjä ympäri maailman. Satel valmistaa tuotteita sekä lyhyen, että pitkän kantaman tiedonsiirtoon. Reitittimiä on olemassa eri taajuusalueille 100–900 MHz. Osa käytetyistä taajuusalueista ovat luvanvaraisia, osa taasen vapaasti käytettävissä. (Satel 2021a; Reiman 2017.)

2.1 Tuotteet

Satelin radiomodeemit ovat jaettu neljään eri luokkaan, jotka sisältävät eri tuoteperheitä:

- Ethernet ja sarjaväylän kautta toimivat radioreitittimet
- vain sarjaväylän kautta toimivat radioreitittimet
- erikseen asennettavat ja pienemmät radiomoduulit
- mobiiliverkoissa toimivat reitittimet.

Kuvassa 1 on esiteltynä jokaisesta edellä mainitusta luokasta tuotteita. Kuvassa esiintyy vasemmalta oikealle SATEL-GW120 (Mobiiliverkko), SATEL XPRS (Ethernet ja sarjaväylä), SATEL TRx (Radiomoduuli), ja SATEL PROOF-TRx (Sarjaväylä) tuotteet. (Satel 2021b.)

Kuva 1. SATEL-GW120, SATEL XPRS, SATEL TRx ja SATEL PROOF-TRx -tuotteet.

2.2 Radioreitittimien käyttökohteet

Radioreitittimiä käytetään maailmanlaajuisesti eri teollisuuskohteissa. Radioreitittimien tehtävä on varmistaa luotettava ja toimintavarma tiedonsiirto. Esimerkkejä radioreitittimien käyttökohteista ovat Supervisory Control and Data Acquisition (SCADA), koneohjaus, Global Navigation Satellite System (GNSS), ympäristömonitorointi ja teollinen internet. (Satel 2021c.)

Supervisory Control and Data Acquisition (SCADA) on järjestelmä, jolla valvotaan ja ohjataan eri teollisuuslaitosten toimintaa paikallisesti tai etänä. Näiden järjestelmien toiminta on oltava äärimmäisen luotettavaa. Järjestelmän havaitsemat vikatilanteet on voitava havaita, sekä korjata nopeasti. SCADA-järjestelmä rakentuu useista eri komponenteista kuten tehdaslaitteista, antureista ja mikrokontrollereista. (Inductive Automation 2021.)

Koneohjaus on laitteistojen ohjaamiseen ja paikantamiseen käytetty teknologiamuoto. Siinä käytetään esimerkiksi Real Time Kinematic - Global Navigation Satellite System (RTK-GNSS) -satelliittipaikannusta parantamaan paikantamisen tarkkuutta. Koneohjauksella edistetään työmaakoneitten automatisointia ja seurantaa. RTK-GNSS:n avulla on mahdollista saavuttaa senttimetrien tarkkuus laitteen ohjauksessa. (Novatron 2021.)

Global Navigation Satellite System (GNSS) on satelliittisignaaleja hyödyntävien järjestelmien yleiskäsite. Käsitteen alle kuuluvat kaikki paikantamiseen satelliittisignaaleja käyttävät teknologiat. GNSS-järjestelmät eivät ole sidottu yksittäiseen satelliittiin, vaan käytössä oleva satelliitti vaihtuu tarpeen mukaan. Vaihdoksen syynä voi olla esimerkiksi yhteyden menetys. (Symmetry Electronics 2021.)

Ympäristömonitorointia käytetään ympäristöolosuhteiden valvomiseen. Sitä käytetään sään tutkimiseen, sekä ilman- maaperän- ja vedenlaadun seuraamiseen. Käyttökohteita ovat ilmatieteenlaitokset, ympäristöarvioinnit ja ympäristöhaittariskin sisältävät alueet. (Satel 2021e.)

Teollinen internet on nimike, jolla yhdistetään koneet, analytiikka ja ihmiset. Teolliseen internettiin lukeutuvat esimerkkinä automatisoidut ja itseohjautuvat ajoneuvot, sekä monimutkaiset ja itseään energiatehokkaaksi optimoivat tehtaat. Teollisen internetin mahdollistaa asioiden internet (IoT, Internet of Things), joka terminä sisältää erilaiset älykkäät esineet ja laitteistot. (Satel 2021f; Oracle 2021.)

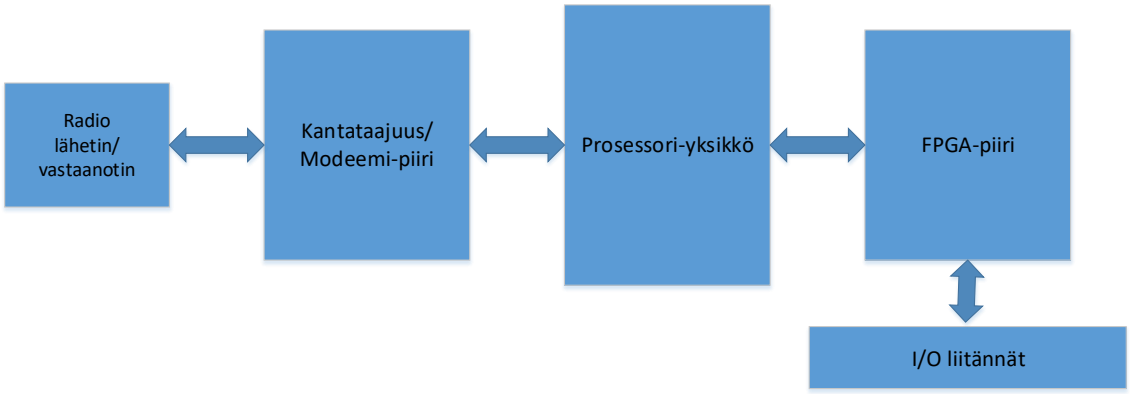
3 RADIOMODEEMIN RAKENNE JA TESTAUS YLEISESTI

Tässä opinnäytetyössä keskitytään pääasiassa radiomodeemin ohjelmistotestauksen ja erityisesti sen automatisoinnin helpottamiseen. Ohjelmistotestauksen lisäksi radiomodeemi käy useiden testivaiheiden läpi. Tässä luvussa käsitellään radiomodeemia yleisesti. Samalla käydään läpi radiomodeemin eri testivaiheita.

3.1 Modeemin toiminta ja rakenne

Radiomodeemin ensisijainen tehtävä on välittää näkymättömien radiosignaalien välityksellä tietoa. Radiomodeemeille on mahdollista asettaa eri toimintatehoja, joilla on suora vaikutus niiden toiminnan kantosäteeseen. Kantaman riittämättömyyttä voidaan ehkäistä toistimien avulla. Radiomodeemeilla on mahdollista luoda yksityinen radioverkko, jolla varmistetaan pieni siirtoviive ja reaaliaikainen tiedonsiirto eri tehdas- ja laitteistoympäristöjen sisällä. Yksityisen radioverkon toiminta on riippumaton muista laajemmista yleisessä käytössä olevista verkoista. Radiomodeemeilla on tavallisiin verkkoihin verrattaessa mahdollista saavuttaa useiden kymmenien kilometrien kantama. Tästä syystä radiomodeemit ovat ensisijainen valinta toiminnoille, joissa vaaditaan suurta toimintasädettä ja luotettavuutta. (Reiman 2017.)

Radiomodeemi rakentuu mikroprosessorin ja laitemuistin sisältävästä mikrokontrollerista. Field-Programmable Gate Array (FGPA) -piiri lisätään mukaan tarvittaessa, mikäli tarvitaan laajempaa toiminnallisuutta. FGPA-piiri ei kuitenkaan ole välttämätön. Itse radioyhteyden ja tiedonsiirron muodostuminen tapahtuu modeemipiirin välityksellä. Modeemipiiri sisältää radiolähettimen ja vastaanottimen. Radiomodeemin liitännät koostuvat yhdestä tai useammasta Universal Asynchronous Receiver-Transmitter (UART)-sarjaliikenneväylästä, sekä yleiskäyttöisistä Input/Output digitaalilinjoista. Kuvassa 2 on esitetty radiomodeemin toimintalohkot. Lohkojen välinen nuoli kuvaa kaksisuuntaista toiminnallisuutta lohkojen välillä. (Reiman 2017.)



Kuva 2. Radiomodeemin toimintalohkot. (Reiman 2017.)

3.2 Radiomodeemin laitteistotestaus

Radiomodeemien laitteistotestausta tehdään useassa eri vaiheessa tuotekehityksen aikana. Eri piirikytkentöjen arvot mitataan ulkoisilla mittalaitteilla ja niiden toimivuus varmistetaan erillisellä kytkentäalustalla. Prototyyppilaitteille testauksella ja mittauksilla varmistetaan sen eri osien toiminta itsenäisenä, sekä yhdessä. (Reiman 2017.)

3.3 Radiomodeemin ohjelmistotestaus

Radiomodeemien ohjelmistotestauksessa varmistetaan uusien, sekä vanhojen ominaisuuksien toiminta laiteohjelmistoja kehittäessä. Ohjelmistotestausta suoritetaan, sekä uusille, että vanhoille laitteille uusien ohjelmistoversioiden valmistuessa. (Reiman 2017.)

3.4 Radiomodeemin mekaaninen testaus

Radiomodeemin mekaanisella testauksella varmistetaan laitteiden rakenteiden riittävä kestävyys määrätyissä olosuhteissa. Laitteelle luvatun kestävyysluokituksen vaatimuksien täyttyminen on tuotekehitysvaiheessa varmistettava. (Reiman 2017.)

3.5 Radiomodeemin tyyppihyväksyntätestaus

Radiomodeemin tyyppihyväksyntätestauksella varmistetaan yhteensopivuus eri radioverkoissa. Tämä tehdään, jotta ollaan varmoja siitä, että verkkoon kytketyt laitteet

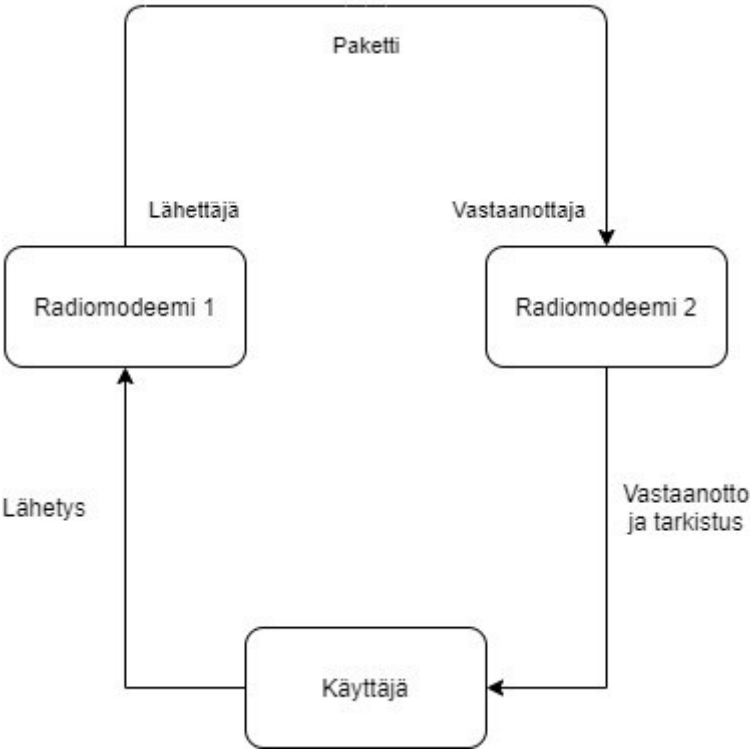
eivät häiritse muuta verkossa tapahtuvaa liikennettä. Tyyppihyväksynnän vaatimukset perustuvat kansainvälisiin standardeihin ja testit suoritetaan erillisissä sertifikaatin saaneissa laboratorioissa. (Reiman 2017.)

3.6 Radiomodeemin tuotantotestaus

Radiomodeemin tuotantotestaus suoritetaan ladonnasta valmistuneille laitteille. Tuotantotestauksella varmistetaan jokaisen yksittäisen sarjatuotetun laitteen toiminta. Laitteille on ennalta määritetty vaatimukset, joiden on testeissä täytyttävä. Mikäli testit eivät onnistu, tuote ei ole myyntikelpoinen ja se on palautettava tuotantoprosessiin korjattavaksi. Tuotantotestauksessa toistetaan paljon samoja testejä, ja se on pääasiallisesti automatisoitu. (Reiman 2017.)

4 OHJELMISTOTESTAUS JA TESTIAUTOMAATIO

Tässä luvussa käsitellään tarkemmin radiomodeemin ohjelmistotestauksen rakennetta, sekä Satelin rakentamaa testiautomaatiota. Radiomodeemin ohjelmistotestaus sisältää useita eri testejä; tiedonsiirtotestejä, komentotestejä ja eri toiminnallisuustestejä. Testikokonaisuus vaihtelee tuotteittain. Radiomodeemin testausta varten on rakennettu testipenkkejä, joissa kaksi laitetta on kytkettynä toisiinsa. Kuvassa 3 on esitettynä testipenkin muodostuminen. Kuvassa käyttäjälle on kytketty koneeseen kiinni kaksi radiomodeemia. Datapaketti lähetetään radiomodeemille 1, ja se vastaanotetaan takaisin radiomodeemilta 2. Vastaanotettu paketti tarkistetaan, ja sen sisällön on vastattava lähetetyn paketin sisältöä. Tämä vastaa yksinkertaista lähetystestiä.



Kuva 3. Testipenkki ja lähetystesti -esimerkki.

Satelin radiomodeemien ohjelmiston toiminta varmistetaan seuraavilla testeillä:

- Quick data transfer test
    - Nopea 100 tavun lähetystesti, toistaa lähetyksen 10 kertaa.
- Fast data transfer test
    - Nopea 100 tavun lähetystesti, toistaa lähetyksen 100 kertaa.

- Basic data transfer test
    - Usealla sarjaväylänopeudella suoritettava lähetystesti, lähettää 10, 100, 300 ja 500 tavun paketteja kutakin 100 kertaa.
    - Käy läpi sarjaväylänopeudet 115200, 57600, 38400 ja 19200.
- Full data transfer test
    - Täydellinen usealla sarjaväylänopeudella suoritettava lähetystesti. Käy läpi 1–1030 tavun pakettikoot ja lähettää kutakin kokoa yhden paketin.
    - Käy läpi sarjaväylänopeudet 115200, 57600, 38400, 19200, 9600, 4800, 2400 ja 1200. Ja niiden eri pariteettibitit, none, even ja odd.
    - Pariteetti bittiä käytetään sarjaliikenteessä varmistamaan lähetyksen onnistuminen. Mikäli vastaanotetun viestin pariteetti eroaa lähetetystä, lähetyksessä on mennyt jotain pieleen. (Computer Hope 2021)
- Add RSSI to data test
    - Testi, jossa lisätään lähetettävään dataan mukaan signaalinvahvuus (RSSI) ja varmistetaan sen vastaanotto.
- CTS handshaking test
    - Testi, jossa varmistetaan laitteiston eri Clear To Send (CTS) moodien toiminta.
    - CTS-tilassa vastaanottava laite ilmoittaa, kun se on valmis tiedonsiirtoon.
- RTS handshaking test
    - Testi, jossa varmistetaan Request To Send (RTS) linjan toiminta.
    - RTS-tilassa lähettävä laite kysyy, onko vastaanottava laite valmis tiedonsiirtoon.
- Encryption feature test
    - Testi, jossa varmistetaan ilmateitse tapahtuvan salauksen toiminta.
- SL commands test
    - Testi, jossa käydään läpi laitteiden eri SL komennot ja varmistetaan niiden toiminta.
    - SL komennot ovat radiomodeemille sarjaväylän kautta lähetettäviä viestejä, joilla on mahdollista asettaa ja varmistaa laitteiston asetuksia.
- NMS commands test
    - Testi, jossa käydään läpi laitteiden eri NMS komennot ja varmistetaan niiden toiminta.
    - NMS komennot on radiomodeemin konfigurointiin käytettäviä komentoja. NMS komennot sisältävät tietyllä laitteistokoodilla olevia asetuksia.

- Sleep mode test
    - Testi, jossa varmistetaan laitteen lepotilan toiminta.
- Power save mode test
    - Testi, jossa varmistetaan laitteen virransäästötilan toiminta.
- Stress test
    - Testi, jossa satunnaisien kokoisia paketteja lähetetään satunnaisella viiveellä.
- SL command mode test
    - Testi, jossa testataan laitteen ominaisuus, joka estää tai sallii SL komentojen lähettämisen laitteelle.
- TX test modes and RSSI
    - Testi, jolla ajetaan lähetyksiä ja tarkistetaan, että signaalinvahvuus vastaa käytettävän tehon suhteen.
    - TX tarkoittaa lähetystä ja RSSI signaalinvahvuutta.
- Region code test
    - Testi, jolla varmistetaan, että laite toimii maakoodilla määritettyjen maiden vaatimusten mukaisesti ja laite estää toiminnan mahdollisilla virheellisillä asetuksilla.
- Channel list test
    - Testi, jolla varmistetaan laitteen kanavalistan oikeanlainen toiminta.
- Error check test
    - Testi, jossa suoritetaan lähetystestejä eri virheenkorjausmoodeilla.
- Radio compatibility test
    - Testi, jossa suoritetaan lähetystestejä eri yhteensopivuustiloissa.

Ennen testiautomaatiojärjestelmän käyttöönottoa kaikki edellä mainitut testit suoritettiin manuaalisesti erillisellä Satelin kehittämällä radiomodeemien testiohjelmistolla.

4.1 Satelin testiautomaatiojärjestelmässä käytettävät teknologiat ja työkalut

Tässä luvussa käydään läpi Satelin testiautomaatiojärjestelmässä käytettäviä teknologioita.

4.1.1 SVN-versionhallinta

Subversion (SVN) on Satelin käyttämä versionhallintaan rakennettu avoimen lähdekoodin järjestelmä. Sitä käytetään eri ohjelmistojen, testien ja tiedostojen säilöntään. Versionhallinnalla mahdollistetaan ohjelmiston jatkuva kehitys ilman vaaraa sen tuhoutumisesta. SVN on julkaistu avoimen lähdekoodin Apache-lisenssillä. (Subversion 2021.)

4.1.2 Jenkins-automaatiopalvelin

Jenkins on avoimen lähdekoodin automaatiopalvelin, joka mahdollistaa ohjelmistotestauksen automatisoinnin. Jenkinsillä käyttäjä voi määritellä erilaisia työrakenteita, sekä määrittää niille perusteet, miten ja koska ne suoritetaan. (Jenkins 2021.)

Jenkinsiä käytetään pääasiallisesti Satelin testikokonaisuuksien ajoittamiseen ja ajamiseen.

4.1.3 Robot Framework-testiautomaatiorunko

Robot Framework on suomalaisen Pekka Klärckin kehittämä geneerinen testiautomaatiorunko, joka perustuu avainsanapohjaiseen rakenteeseen. Se on kehitetty ja kirjoitettu Python-ohjelmointikielellä. Robot Framework käyttää .robot -päätteen tiedostoja. Robot Frameworkkia käytetään yleisesti testiautomaatioon, sekä ohjelmistorobotiikkaan. (Robot Framework 2021.)

Radiomodeemin ohjelmistotestauksen automatisoinnissa käytetään Robot Frameworkillä kehitettyjä testejä yhdessä Python-testikirjaston kanssa.

4.1.4 Python-ohjelmointikieli

Python on Guido van Rossumin kehittämä tulkattava ohjelmointikieli. Python omaa yksinkertaisen ja helposti opittavan syntaksin ja sitä suositellaankin monesti ensimmäiseksi ohjelmointikieleksi. (Python 2021a.)

Satelin testiautomaatiojärjestelmässä käytettävä pääasiallinen kirjasto on kirjoitettu Pythonilla. Se sisältää useita funktioita, joita hyödynnetään Robot Framework -komentosarjoissa.

4.1.5 JSON-tiedostomuoto ja testiprofiilit

JavaScript Object Notation (JSON) on universaaliin tiedonsiirtoon käytettävä tiedostomuoto. JSON on ihmisille helposti luettavaa ja kirjoitettavaa. Tietokoneet pystyvät tuottamaan ja parsimaan JSON-tiedostoja helposti. (JSON 2021.)

Testiprofiilit ovat Satelin testiautomaatiojärjestelmässä käytettäviä JSON-päätteellä varustettuja tiedostoja. Testiprofiilit sisältävät reitittimeen määritettäviä testiasetuksia. Jokaisen testiasetuksen kohdalla JSON-tiedostoon määritellään kohdat

- address, verkon ylitse tarvittaessa käytettävä osoite
- id, laitteessa olevan asetuksen NMS, eli laitekoodi
- meta, tunnistusta helpottavaa meta tietoa
- v, asetukselle määritettävä arvo
- et_ref, viittaus tuotetietokantaan (Ei käytössä).

Kuvassa 4 on esimerkki JSON-testiprofiilitiedoston sisällöstä.

```
{
    "name": "TR4+_0_12.5_kHz_FEC_463.0MHz.json",
    "time": "This file was automatically generated on: 22/02/2021 09:24:53",
    "data": [
        {
            "address": "0.0",
            "id": "1.1996",
            "meta": "Channel Width",
            "v": "12.5 kHz",
            "et_ref": null
        },
        {
            "address": "0.0",
            "id": "1.270",
            "meta": "FEC",
            "v": "On",
            "et_ref": null
        },
        {
            "address": "0.0",
            "id": "1.256",
            "meta": "RX Freq",
            "v": "463000000",
            "et_ref": null
        },
        {
            "address": "0.0",
            "id": "1.257",
            "meta": "TX Freq",
            "v": "463000000",
            "et_ref": null
        }
    ]
}
```


Kuva 4. Esimerkki testiprofiilin sisällöstä.

4.2 Testiautomaation rakenne

Satelin testiautomaatiojärjestelmä on rakennettu käyttäen Jenkins-automaatiopalvelinta ja Robot Framework -nimistä Python-kirjastoa. Testit jakautuvat pääasiassa kolmeen eri testikokonaisuuteen:

- versionhallinnan muutoksien mukana käynnistyvät testit
- joka yö ajettavat testit
- viikonloppuna ajettavat testit.

Ensimmäisenä edellä mainitussa listassa mainitut testit käynnistyvät, kun havaitaan muutoksia versionhallinnassa. Testikokonaisuus on kestoltaan sopivan mittainen. Testikokonaisuudesta saadaan nopea palaute kehittäjille.

Yöllä ajettavat testit ovat hieman laajempi ja pitkäkestoisempi kokonaisuus. Yölliset testit ajetaan yleisesti noin 4 kertaa viikossa ja ne käynnistyvät siten, että voidaan olettaa, että kaikki sen päivän päivitykset ovat tehty.

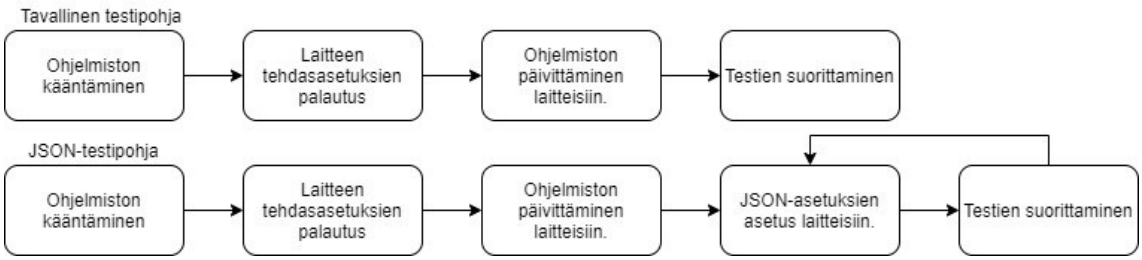
Viikonlopputestit ovat täydellisiä testikokonaisuuksia, jotka sisältävät kaikki testissuunnitelmassa määrätyt testit. Näillä regressiotesteillä varmistetaan ohjelmiston täydellinen toiminta ja päivityksien julkaisukelpoisuus.

Testikokonaisuuksien lisäksi testit ovat jaettu kahteen ajettavaan testipohjaan. Testikokonaisuuksia suoritetaan tavallisella tai JSON-tiedostoja hyödyntävällä pohjalla. Nämä pohjat ovat Jenkinsiin erikseen luotuja työkokonaisuuksia. Pohjien avulla voidaan muuttaa asioita, kuten versionhallinnan osoite, ajettavat testit ja käytettävät testiprofiilit, mutta itse työn eteneminen säilyy samanlaisena.

Jenkinsiin määritettyjen työkokonaisuuksien toiminta etenee seuraavin vaihein:

1. Uusimmat työhön liittyvät tiedostot tuodaan versionhallinnasta.
2. Käännetään uusin ohjelmisto versionhallinan tiedostoista.
3. Varmistetaan, että testattavissa laitteissa on tehdasasetukset sisällä.
4. Päivitetään käännetty ohjelmisto laitteisiin.
5. Ajetaan JSON-asetukset sisälle laitteisiin, mikäli JSON-pohja käytössä. Muutoin siirrytään kohtaan 6.
6. Suoritetaan määrätyt testit.

Kuvassa 5 on esitettynä Jenkins-työn eteneminen testipohjissa. Pohjien ero johtuu siitä, että tavallisessa testipohjassa ei määritetä laitteistoasetuksia ennen testien suorittamista. JSON-testipohjaa käyttäessä laiteasetukset haetaan JSON-tiedostoista niille määritetystä kansiosta.



Kuva 5. Tavallisen ja JSON-testipohjan rakenne.

Jenkinsin työn valmistuttua kustakin työvaiheesta saadaan selville niiden onnistuminen. Joissain tapauksissa, kuten laitteen ohjelmiston päivityksessä, työ pysähtyy, mikäli vaihe on epäonnistunut.

Itse testisuorituksista syntyy HTML-raportti. Raportista on helppo selvittää, mikäli joku testi on epäonnistunut. Raportista on tarkemmin myös nähtävillä, missä vaiheessa testi on epäonnistunut ja miksi. Kuvassa 6 on esimerkki Robot Frameworkin tuottamasta HTML-raportista.

Statistics by Suite	Total	Pass	Fail	Elapsed	Pass / Fail
Execute Tests Weekend TR4	123	115	8	06:25:48	
Execute Tests Weekend TR4 . Add RSSI to Data	3	3	0	00:02:59	
Execute Tests Weekend TR4 . CTS Handshaking	4	4	0	00:06:53	
Execute Tests Weekend TR4 . FastDataTransfer	1	1	0	00:00:30	
Execute Tests Weekend TR4 . FullDataTransfer	12	12	0	01:38:49	
Execute Tests Weekend TR4 . NMS Commands TRx	10	10	0	00:31:09	
Execute Tests Weekend TR4 . Power Save Mode	9	8	1	00:33:05	
Execute Tests Weekend TR4 . RTS Handshaking	5	5	0	00:02:13	
Execute Tests Weekend TR4 . SatelCrypt	3	3	0	00:04:37	
Execute Tests Weekend TR4 . SL Commands	4	3	1	00:42:09	
Execute Tests Weekend TR4 . Sleep Mode	31	30	1	01:16:21	
Execute Tests Weekend TR4 . Stress Test	4	4	0	00:11:13	
Execute Tests Weekend TR4 . TX test modes and RSSI	5	5	0	00:00:50	
Execute Tests Weekend TR4 . Y SL Command Mode	32	27	5	01:14:59	

Kuva 6. Robot Frameworkin tuottama HTML-testiraportti.

5 OHJELMISTON JA KÄYTTÖLIITTYMÄN SUUNNITTELU

Tässä luvussa käsitellään vanhaa asetuksien tuottamiseen käytettyä komentorivipohjaista JavaScript-ohjelmistoa. Samalla käsitellään uuden Python-ohjelmiston käyttöliittymän ja toimintojen suunnittelua.

5.1 Vanha komentorivipohjainen JavaScript-ohjelmisto

Testiprofiilien tuottamiseen käytetty vanha ohjelmisto on komentorivillä ajettava JavaScript-ohjelmointikieltä käyttävä komentosarja. JavaScript on lähinnä web-ympäristöissä, mutta myös muualla käytettävä dynaaminen komentosarjakieli (MDN Web Docs 2021a). Komentosarjan ajaminen suoritetaan Node.js:llä. Node.js on avoimen lähdekoodin omaava järjestelmäriippumaton JavaScript-komentosarjojen suorittamiseen kehitetty ajoympäristö (MDN Web Docs 2021b).

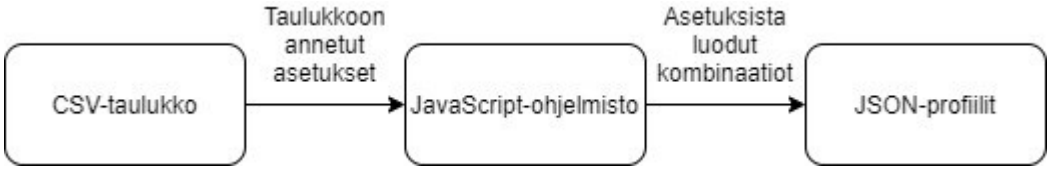
Komentosarjan toiminta perustuu samasta kansiosta luettavaan CSV-tiedostoon. Käyttäjän on avattava CSV-tiedosto esimerkiksi Microsoft Excel -ohjelmistolla ja käsin syötettävä kullekin riville vaadittavat tiedot. Taulukkoon on mahdollista syöttää virheellisiä tietoja ja täten tuottaa profiileja, jotka eivät testivaiheessa toimi.

Taulukkoon annettujen tietojen perusteella tuotetaan profiilit jokaisesta syntyvästä asetuskombinaatiosta. Asetuksille on mahdollista antaa useampi arvo, jolloin kaikki annetut arvot käsitellään erikseen. Taulukossa 1 on esimerkki CSV-tiedoston sisällöstä. Kyseisestä tiedostosta syntyisi 48 eri asetuskombinaatiota, eli testiprofiilia.

Taulukko 1. CSV-tiedoston sisältö.

0,0	1,270	FEC	OFF, ON
0,0	1,256, 1,257	TX Freq, RX Freq	403000000, 438000000, 473000000
0,0	1,1996	Channel Width	12.5 kHz, 25 kHz
0,0	1,344	Radio Compatibility	SATELLINE-3AS, SATEL-8FSK-1, SATEL-8FSK-2, SATEL-16FSK-1
0,0	1,267	TX Power	100 mW

Kuvassa 7 on esitettynä vanhan ohjelmiston toiminta. JavaScript-ohjelmisto käsittelee CSV-taulukon sisällön tuottaen siitä asetuskombinaatiot. Taulukon käsittelyn jälkeen asetuskombinaatioista muodostetaan JSON-päätteiset testiprofiilit.



Kuva 7. Vanhan ohjelmiston toiminta.

5.2 Uusi graafisella käyttöliittymällä oleva Python-ohjelmisto

Vanhan ohjelmiston toimintaa tutkittiin ja sen perusteella suunniteltiin graafinen käyttöliittymä, joka loisi käyttäjäystävällisemmän kokonaisuuden. JavaScript-ohjelmointikielen sijaan uusi ohjelmisto päätettiin kehittää Python-ohjelmointikielellä. Python valittiin ohjelmointikieleksi sen monipuolisuuden ja helppokäyttöisyyden takia. Lisäksi Python on tullut JavaScriptiä paljon tutummaksi muiden työtehtävien lomassa.

Graafisen käyttöliittymän kehittämiseen valittiin työkaluksi PySimpleGUI. PySimpleGUI on Python-kirjasto, jonka käyttö on yksinkertaista ja siitä syntyvä lähdekoodi helposti luettavaa. PySimpleGUI:sta on saatavissa 5 eri versiota, joista tässä työssä päädyttiin käyttämään tkinter -versiota. Tkinter on Pythonin standardi Graphical User Interface (GUI) -työkalu, josta PySimpleGUI tekee huomattavasti helpommin käytettävän yksinkertaisemman syntaksin myötä. (PySimpleGUI 2021.)

Uudessa ohjelmistossa haluttiin säilyttää vanhan ohjelmiston manuaalinen asetusten syöttäminen, mutta samalla tuoda siihen lisää ominaisuuksia. Ohjelmiston suunnittelussa nähtiin parhaaksi tuottaa sovellukseen kaksi erillistä ikkunaa, pääikkuna ja lisäasetusikkuna. Pääikkunaan suunniteltiin helppokäyttöisemmät toiminnot ja lisäasetusikkunaan vanhan ohjelmiston kaltainen toiminta. Kaikkia toimintoja ei haluttu sovittaa yhden ikkunan sisälle ohjelmiston käytön selkeyttämiseksi.

5.2.1 Pääikkunan toimintojen määrittely

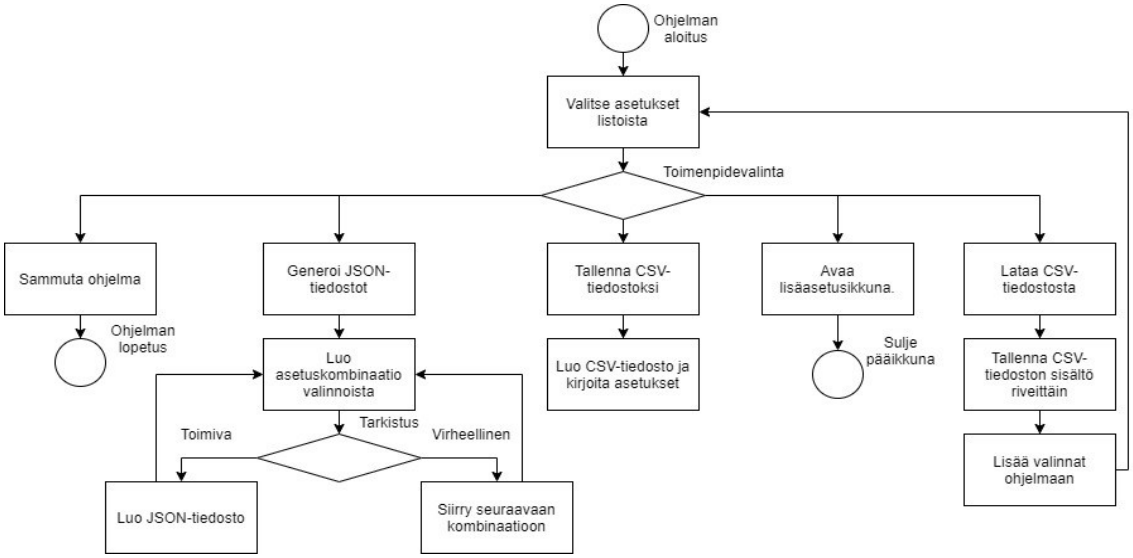
Pääikkuna suunniteltiin toimimaan yksinkertaisena valintaikkunana, mistä käyttäjän olisi helppo valita valmiista listoista haluamansa asetukset. Kun asetuksien valitseminen on tehty, käyttäjä voi yhdellä napinpainalluksella käynnistää asetuskombinaatioiden tuottamisprosessin.

Asetuskombinaatioiden tuottamisen lisäksi ohjelmistoon haluttiin ominaisuudeksi mahdollisuus tallentaa asetuskombinaatiot erilliseen tiedostoon. Tallennuksen avulla asetukset olisivat helposti saatavilla myöhempää käyttöä varten. Tallentamisen lisäksi on luonnollista, että tallennetut asetukset olisi myös mahdollista avata takaisin ohjelmaan.

Pääikkunaan haluttiin lisätä ominaisuudeksi virheellisten asetusyhdistelmien estämiseen suunniteltu funktio, joka tarkastaisi kaikki asetukset ja varmistaisi niiden toimivuuden. Toimivuuden varmistaminen tarkoittaa, että kullakin taajuusalueella on mahdollista asettaa vain sille sopivat asetukset.

Alkuperäisessä JavaScript-ohjelmistossa JSON-tiedostojen nimeäminen ei ollut mahdollista. Jokainen tuotettu testiprofiili nimeytyi ”profile_numero” -tyylillä. Tämä haluttiin muuttaa, jotta nimeäminen saataisiin selkeämmäksi. Uudessa ohjelmassa haluttiin, että tiedostojen nimeäminen on mahdollista. Yksilöivä nimeäminen helpottaa testiprofiilien tunnistamista.

Kuvassa 8 on pääikkunan toimintojen suunnitellussa käytetty vuokaavio.



Kuva 8. Pääikkunan vuokaavio.

5.2.2 Lisäasetusikkunan toimintojen määrittely

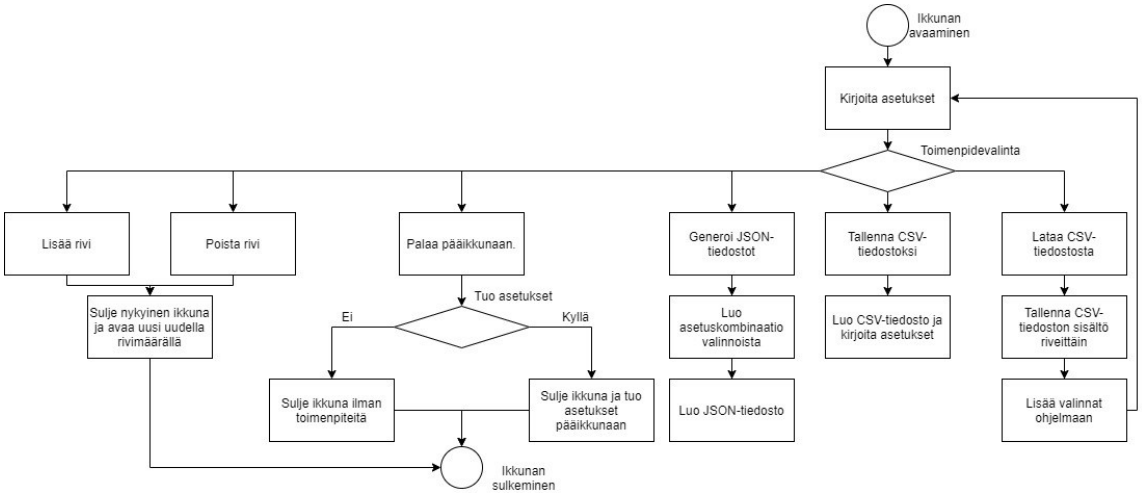
Lisäasetusikkunalla haluttiin säilyttää vanhan ohjelmiston vapaus ja mahdollistaa asetusten kirjoittaminen myös käsin. Tämä edesauttaa tilanteissa, joissa on käytettävä tuotekohtaisempia ja yleisasetuksista poikkeavia asetuksia.

Lisäasetusikkunan ominaisuudeksi haluttiin mahdollisuus lisätä ja poistaa ikkunassa näkyviä rivejä tarvittaessa. Tämän lisäksi kaikki pääikkunan perustoiminnot, eli tiedostojen nimeäminen, tallentaminen ja lataaminen lisättiin mukaan. Pääikkunan suurinta toimintoa, eli väärien asetuskombinaatioiden estämistä ei ollut mahdollista lisätä mukaan. Asetuskombinaatioiden tarkistaminen käsin kirjoittaessa on mahdotonta, koska ei voida ennalta tietää mitä asetuksia käyttäjä syöttää.

Lisäasetusikkunaan lisättiin mahdollisuus siirtää asetukset pääikkunaan. Kaikki asetukset, mitkä ovat lisäasetusikkunasta siirtäessä jo olemassa pääikkunassa, tulevat automaattisesti valituksi pääikkunan listavalinnoista. Asetukset mitä ei ole pääikkunassa olemassa, tallennetaan erilliseen listaan. Kaikki pääikkunan listavalinnat ovat omia listoja, ja lisäasetuksille oli pakko suunnitella erillinen paikka niiden säilyttämiseen.

Kun asetukset ovat tuotu takaisin pääikkunaan, asetuskombinaatioiden tarkistaminen on jälleen mahdollista. Poikkeuksena ovat lisäasetukset, joista ei ole listavalintaa olemassa. Kaikille asetuksille ei ollut mahdollista luoda listavalintoja. Ainoastaan yleisimmät radiomodeemeissa käytettävät asetukset nähtiin tarpeelliseksi lisätä käyttöliittymään.

Kuvassa 9 on esitettynä lisäasetusikkunan vuokaavio.



Kuva 9. Lisäasetusikkunan vuokaavio.

6 OHJELMISTON JA KÄYTTÖLIITTYMÄN TUOTTAMINEN

Tässä luvussa käsitellään itse ohjelmiston toimintojen ja käyttöliittymän tekemistä.

6.1 PySimpleGUI ja tarvittavat toiminnot

Käyttöliittymän luomiseen käytettiin PySimpleGUI:n tarjoamia ominaisuuksia. Niistä käytettiin seuraavia (PySimpleGUI 2021.):

- window, ikkuna, käyttäjälle näytettävä ikkuna
- text, teksti, käyttäjälle näkyvää tekstiä
- listbox, listavalinta, käyttäjälle näkyvä erillinen laatikko, mistä käyttäjän on mahdollista klikata valinta
- input, tekstinsyöttö, ruutu, mihin käyttäjän on mahdollista syöttää tekstiä
- theme, teema, käyttäjälle näkyvä teema, toimii yhdessä window-objektin kanssa
- button, nappi, käyttäjälle näkyvä sen klikkaamisen mahdollistava laatikko
- popup, ponnahdusikkuna, käyttäjälle erikseen ilmestyvä ilmoitusikkuna

6.2 Pääikkuna ja sen toiminnot

Pääikkunaan rakennettiin yhteensä 11 käyttäjän muokattavissa olevaa kohtaa. Kohdat muodostuvat listavalinnoista ja tekstinsyöttöruuduista.

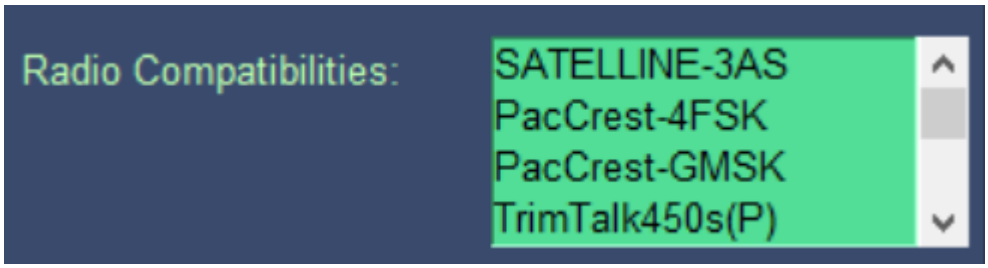
Pääikkunaan sisällytettiin radiomodeemien yleisimmät vaihdettavat asetukset. Alla on pääikkunaan lisätyt asetukset, sekä niiden näyttämiseen valitut objektit:

- taajuusalue, listavalinta
- lähetys ja vastaanotto taajuus, tekstinsyöttöruutu
- yhteensopivuusmoodit, listavalinta
- kanavanleveydet, listavalinta
- salaus, listavalinta
- salaustyyppi, listavalinta
- Forward Error Correction (FEC), listavalinta

- virheenkorjausmoodit, listavalinta

6.2.1 Asetuksien valitseminen

Asetuksien valitsemiseen käyttäjä voi klikata haluamansa asetukset listoista. Listavalinta -kokonaisuus koostuu kahdesta PySimpleGUI-kirjaston tarjoamasta objektista, text ja listbox. Text on pelkkää tekstiä näyttävä osuus. Listbox on valintalaatikko ja siihen on mahdollista määrittää sen sisällä näkyvä lista. Kuvassa 10 on esimerkki listavalinnasta, sekä tekstistä.



Kuva 10. Yhteensopivuusmoodin listavalinta, vasemmalla Text oikealla Listbox -objekti.

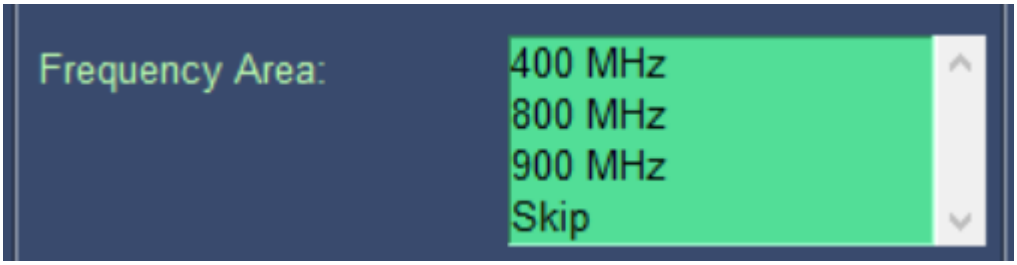
Lähetys- ja vastaanottotaajuudet sekä profiilin nimi kirjotetaan käsin erillisiin tekstinsyöttöruutuihin. Tekstinsyöttö PySimpleGUI-kirjastossa tapahtuu käyttäen Input -nimistä objektia. Kuvassa 11 on esimerkki tekstinsyöttöruudusta.

Kuva 11. Tekstinsyöttöruutu profiilin nimelle, vasemmalla Text, oikealla Input -objekti.

6.2.2 Asetuksien tarkistaminen

Asetusyhdistelmien tarkistaminen perustuu valittuun taajuusalueeseen. Osa asetuksista toimivat vain tietyllä taajuusalueella. Taajuusalueita ohjelmistossa on mahdollista valita kolme. 400, 800 ja 900 MHz. Lisäksi on mahdollista valita asetusvaihtoehto Skip, jolloin taajuusaluetta ei määritetä ollenkaan. Taajuusalueen määrittämättä jättäminen estää myös asetuksien virheellisten kombinaatioiden estämisen toiminnan. Taajuusalue

jätetään määrittämättä, mikäli testattava laite ei tue useampaa taajuusaluetta. Kuvassa 12 on taajuusalueelle luotu listavalinta.



Kuva 12. Taajuusalueelle luotu listavalinta, vasemmalla Text, oikealla Listbox -objekti.

Asetusten tarkistaminen tapahtuu profiilin generointivaiheessa. JSON- ja CSV-tiedostojen luomiselle on molemmille määritetty omat erilliset tarkistusfunktiot. Tarkistusfunktiolle annetaan parametrina sen hetkiset asetukset kustakin yhdistelmästä. Tarkistusfunktion sisälle on määritelty kullekin taajuusalueelle yhteensopivuuslistat. Listoja verrataan annettuihin asetuksiin ja mikäli joku asetus on sellainen, mitä yhteensopivuuslistalta ei löydy funktio palauttaa Falsen. Jos asetukset ovat oikeat, funktio palauttaa Truen. Kuvassa 13 on tarkistusfunktion määrittely.

```
def comp_check(self, freq_area, encryption, encryption_type, channel_width, radio_compatibility, error_check, fec):
    """
    Compatibility check for JSON generation.
    Checks the given setting combination and returns True, if the given settings are found in the compatibility lists.

    :param str freq_area: The given frequency area setting value.
    :param str encryption: The given encryption setting value.
    :param str encryption_type: The given encryption type setting value.
    :param str radio_frequency: The given radio frequency setting value.
    :param str channel_width: The given channel width setting value.
    :param str radio_compatibility: The given radio compatibility setting value.
    :param str error_check: The given error check setting value.
    :param str fec: The given FEC setting value.
    :return: True if compatible, False otherwise.
    """
```

Kuva 13. comp_check -tarkistusfunktion määrittely.

6.2.3 JSON-tiedoston generointi

JSON-tiedostojen generointi tapahtuu useamman vaiheen kautta. Käyttäjä käynnistää prosessin painamalla käyttöliittymässä olevaa ”Generate JSON” -painiketta. Käyttäjän aloitettua generointiprosessin, jokainen annettu listoista syntyvä asetuskombinaatio käydään yksitellen lävitse.

Mikäli taajuusalueeksi ei ole määritelty ”Skip”, jokainen kombinaatio menee tarkistusfunktion läpi. Mikäli tarkistusfunktio palauttaa positiivisen vastauksen, kyseisestä kombinaatiosta luodaan väliaikainen CSV-tiedosto. CSV-tiedosto luodaan käyttäen Pythonin sisäänrakennettua CSV-kirjastoa.

Luomisprosessissa käydään seuraavat asiat läpi:

1. Luodaan profiileille oma kansio ”Profiles/Päivämäärä/Nimi” -tyylillä.
2. Luodaan csv.writer() -olio.
3. Kirjoitetaan asetukset riveittäin CSV-tiedostoon muotoon osoite, id, nimi, arvo.
4. Tallennetaan CSV-tiedosto.
5. Kutsutaan generate_profiles() -funktiota.

Kuvassa 14 on CSV-tiedoston luomiseen käytetyn create_csv -funktion määrittely.

```
def create_csv(self, folder, file_name, encryption, encryption_type, radio_frequency, channel_width,
               radio_compatibility, error_check, frequency, fec, custom, extra_val, profile_num):
    """
    Create a CSV file with the given setting combination.
    This is a helper function for generate_profiles.

    :param str folder: The given folder name.
    :param str file_name: The given file name.
    :param str encryption: The given encryption setting value.
    :param str encryption_type: The given encryption type setting value.
    :param str radio_frequency: The given radio frequency setting value.
    :param str channel_width: The given channel width setting value.
    :param str radio_compatibility: The given radio compatibility setting value.
    :param str error_check: The given error check setting value.
    :param str frequency: The given frequency setting value.
    :param str fec: The given FEC setting value.
    :param boolean custom: Determines if custom frequencies are used or not.
    :param list extra_val: The imported extra values.
    :param int profile_num: Integer value of current profile number.
    """
```

Kuva 14. create_csv -funktion määrittely.

Luomisprosessissa kohdassa 5. mainittu generate_profiles -funktio on itse JSON-tiedoston generointia varten. Generate_profiles -funktio lukee edellisessä kohdassa määritetystä CSV-tiedostosta asetukset riveittäin ja jakaa luetut asetukset sanakirja -muuttujaan. Sanakirja on Pythonissa esiintyvä listan kaltainen tietorakenne, joka koostuu avain:arvo -pareista (Python 2021b). CSV-tiedosto poistetaan, kun JSON-tiedosto on generoitu. Jokainen JSON-tiedosto rakentuu seuraavanlaisesti:

- name, sisältää tiedoston nimen
- time, sisältää ajankohdan tiedoston generoimisesta

- data, sisältää itse asetukset

Kuvassa 4 oli esitetty testiprofiilin rakenteellinen sisältö. Kuvassa 15 on esitettynä generate_profiles -funktion määrittely.

```
def generate_profiles(self, folder, file_name):
    """
    Generate JSON profile from given .CSV file.
    Called from within the create_csv function.

    :param str folder: The given folder name.
    :param str file_name: The given file name.
    """
```

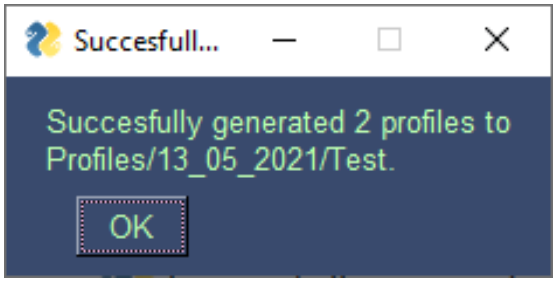
Kuva 15. generate_profiles -funktion määrittely.

Generointiprosessi antaa käyttäjälle jatkuvaa lokitietoa sen edistymisestä komentokehotteessa. Kuvassa 16 on esitetty komentokehotteessa sijaitsevaa lokitietoa.

```
Frequency Compatibility Check:
Encryption OK
Encryption Type OK
Channel Width OK
Error Check OK
FEC OK
Radio Compatibility OK

Settings for current Profile:


Error check: CRC-16 Full
Radio Compatibility: SATELLINE-3AS
FEC: Off
Channel Width: 25 kHz
Radio Frequency Area: 400 MHz
Crypto Mode: Off
Creating profile: Test_400MHz_31

Generation succesfully done.
```

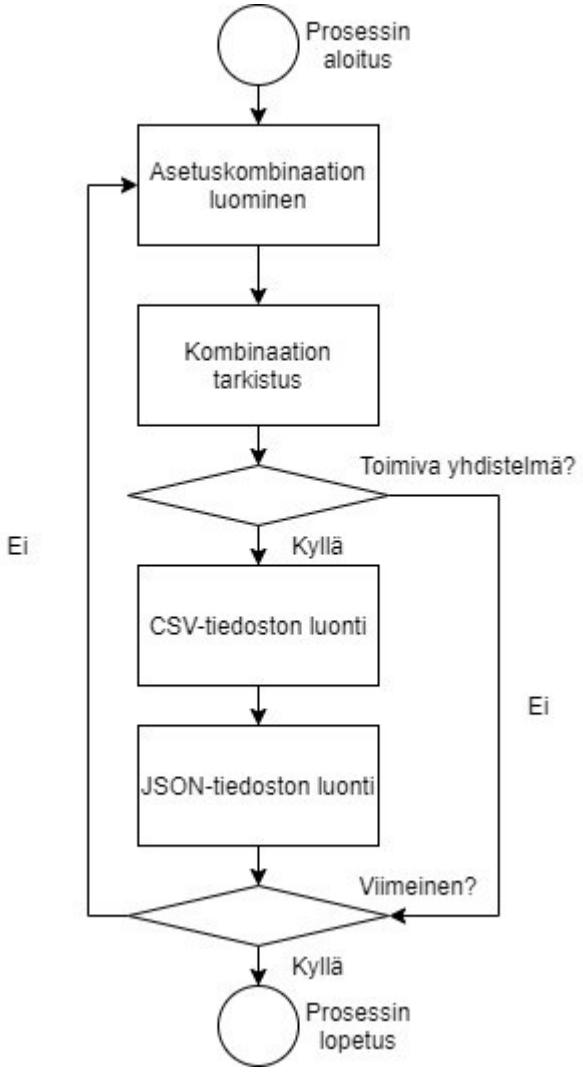
Kuva 16. Lokitietoa komentokehotteesta.

Prosessin valmistuttua käyttäjä saa tiedon onnistumisesta ja generoitujen profiilien määrästä erillisessä ponnahdusikkunassa. Kuvassa 17 on esitetty ponnahdusikkuna.



Kuva 17. Ponnahdusikkuna prosessin valmistuttua.

Kuvassa 18 on esitetty generointiprosessi vuokaaviona.



Kuva 18. JSON-profiilien generoimisen vuokaavio.

6.2.4 Tiedostojen nimeäminen

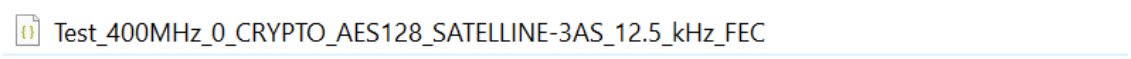
Tiedostojen nimeämiselle kokeiltiin monia eri vaihtoehtoja. Aluksi nimeäminen suoritettiin siten, että käyttöliittymässä annettu profiilin nimi tuli nimen alkuun. Seuraavaksi liitettiin taajuusalue ja lopuksi numerointi. Tämän nimeämisen ansiosta saatiin jokaiselle taajuusalueelle omat profiilit numeroituna ”nimi_taajuusalue_numero” -tyylillä.

Nimeämistyyli oli alustavasti hyvä, mutta Jenkinsissä se ei riittänyt tiedoston sisällön helppoon tulkintaan. Käyttäjän ei ollut mahdollista helposti nähdä, mitä missäkin profiilissa on sisällä. Helpottavaksi toimenpiteeksi lisättiin erillisen tekstilokin generointi. Tekstilokissa oli määritetty yhdessä tiedostossa jokaisen erillisen profiilin sisältö. Kuvassa 19 on esitettynä lokitiedoston sisältöä.

```
0 profile for Test_400MHz.

Error check: Off
Radio Compatibility: SATELLINE-3AS
FEC: On
Channel Width: 12.5 kHz
Radio Frequency Area: 400 MHz
Crypto Mode: On
Crypto Type: AES128
```

Kuva 19. Lokitiedoston sisältöä.

Edellä mainittu lisäys ei kuitenkaan ollut riittävä. Saadakseen näkyville testissä käytetyt asetukset käyttäjän oli avattava erillinen tiedosto. Erillisen tiedoston avaaminen lisäsi työmäärää. Tiedoston nimeämistyyli vaihdettiin uuteen ja lopulliseen muotoon. Nimeämistyylin lopullinen muoto sisällyttää tiedostonnimeen kaikki käytetyt asetukset suoraan. Muutoksen avulla käyttäjä näkee mitkä asetukset on ollut käytössä missäkin testissä, ilman ylimääräisiä tiedostoja. Lokitiedostoa ei ollut kuitenkaan syytä poistaa käytöstä, joten se jätettiin myös toimintaan. Kuvassa 20 esitettynä tiedoston lopullinen nimeämistyyli.



Kuva 20. Tiedoston nimeämistyyli.

6.2.5 Tallentaminen CSV-tiedostoksi

CSV-tiedoston tallentaminen on useamman vaiheen prosessi. Tallentaminen koostuu tarkistuksesta eri taajuusalueilla JSON-tiedoston generoinnin tyyppisesti. CSV-tiedoston luominen poikkeaa JSON-tiedoston generoinnista kuitenkin siten, että kaikki asetukset voidaan kirjoittaa samaan tiedostoon ja niitä ei ole tarvetta käydä läpi erillisinä asetuskombinaatioina.

CSV-tiedostot luodaan taajuusalueen mukaan. Jokaiselle taajuusalueelle luodaan oma tiedosto, joka sisältää vain sen alueella toimivat asetukset. Lisäksi luodaan yksi yhtenäinen _ALL -päätteellä oleva tiedosto, joka sisältää kaikki annetut asetukset. Tämä helpottaa tilanteissa, missä käyttäjä haluaa generoida kaikki profiilit, mahdollistaen samalla vain yksittäisen taajuusalueen profiilien generoimisen. ”Skip” -asetus valittuna käyttöliittymässä taajuusalueeksi, ainoastaan _ALL -päätteen tiedosto luodaan.

CSV-tiedoston tarkistusprosessissa kaikki annetut asetukset välitetään listoina eteenpäin comp_check_csv -funktiolle ja kustakin taajuusalueesta palautetaan listoina ainoastaan yhteensopivat asetukset. Tämän jälkeen annetut asetukset kirjoitetaan CSV-tiedostoon ja tallennetaan ”Saved_CSV/päivämäärä/nimi” -muotoiseen kansioon. Kuvassa 21 on esitettynä comp_check_csv -funktion määrittely. Kuvassa 22 esitettynä save_csv -funktion määrittely.

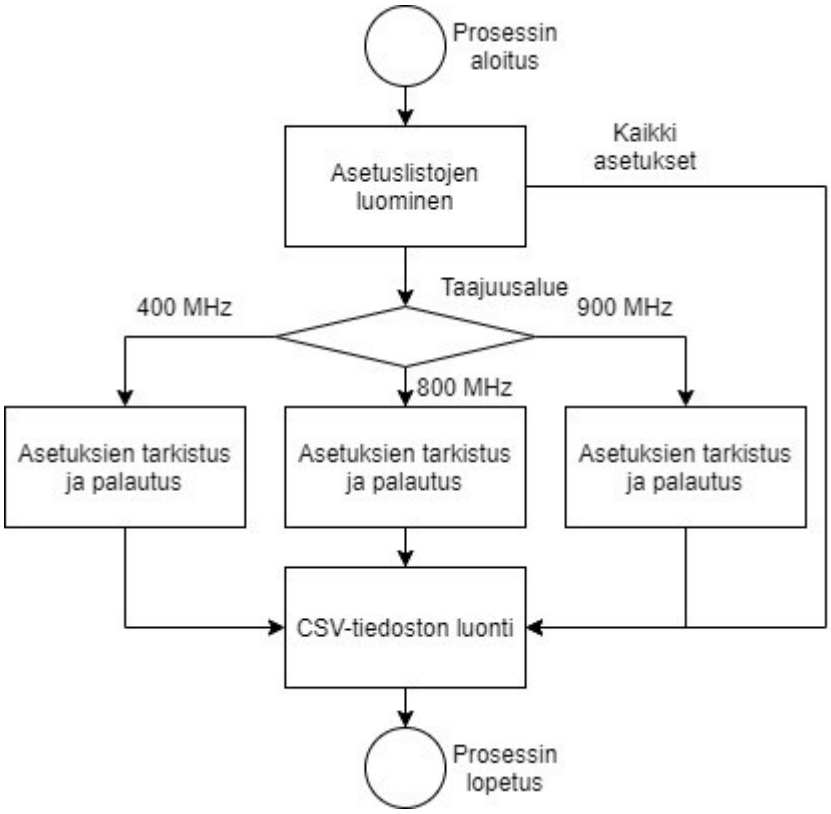
```
def comp_check_csv(self, freq_area, encryption, encryption_type, channel_width, radio_compatibility, error_check, freq, fec_c):
    """
    Compatibility check for CSV file creation.
    All given setting combinations are looped and checked for compatibility.
    If settings are not compatible, they are skipped.

    :param list freq_area: The given frequency area setting values.
    :param list encryption: The given encryption setting values.
    :param list encryption_type: The given encryption type setting values.
    :param list radio_frequency: The given radio frequency setting values.
    :param list channel_width: The given channel width setting values.
    :param list radio_compatibility: The given radio compatibility setting values.
    :param list error_check: The given error check setting values.
    :param list frequ: The given frequency setting values.
    :param list fec_c: The given FEC setting values.
    """
```

Kuva 21. comp_check_csv -funktion määrittely.

```
def save_csv(self, file_name, encryption, encryption_type, radio_frequency, channel_width, radio_compatibility, error_check, frequency, fec, extra):
    """
    Save all the given settings to CSV files.
    Generate one CSV that contains all the settings.
    Generate one CSV for each frequency area used (400, 800, 900)

    :param str file_name: The given file name.
    :param list encryption: The given encryption setting values.
    :param list encryption_type: The given encryption type setting values.
    :param list radio_frequency: The given radio frequency setting values.
    :param list channel_width: The given channel width setting values.
    :param list radio_compatibility: The given radio compatibility setting values.
    :param list error_check: The given error check setting values.
    :param list frequency: The given frequency setting values.
    :param list fec: The given FEC setting values.
    :param list extra: Extra settings from imports.
    """
```

Kuva 22. save_csv -funktion määrittely.

CSV-tallennusprosessin valmistuttua käyttäjä saa tiedon prosessin onnistumisesta ja tallennuskansion sijainnista ponnahdusikkunassa. Kuvassa 23 on tallennusprosessin vuokaavio.

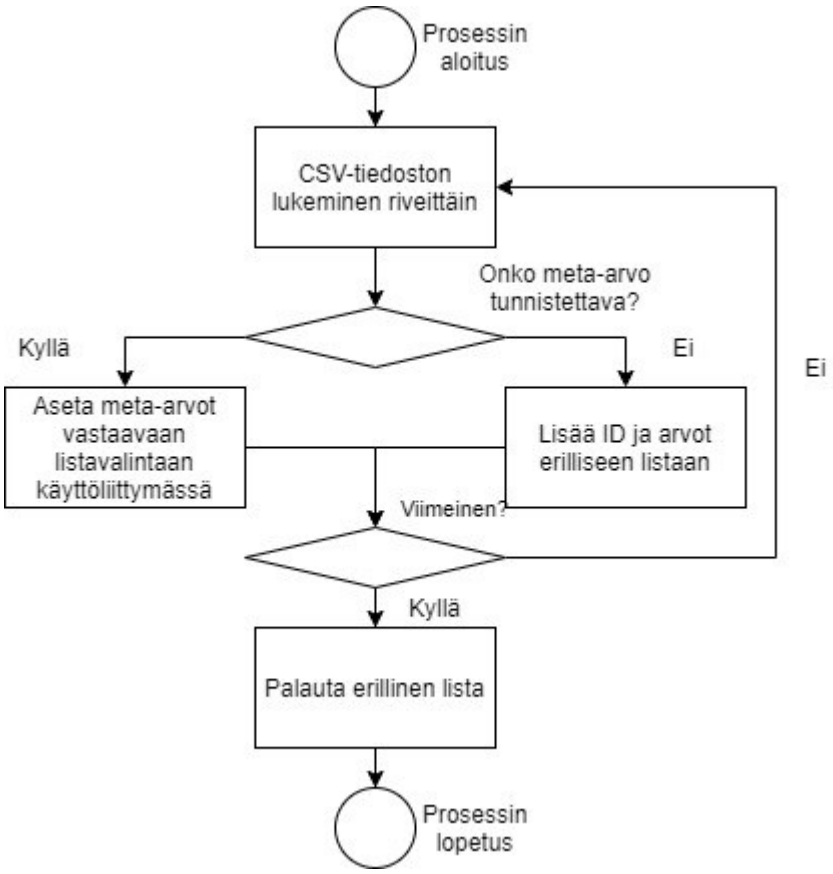


Kuva 23. CSV-tiedoston tallennusprosessin vuokaavio.

6.2.6 CSV-tiedoston lataaminen

CSV-tiedoston latausprosessi on suhteellisen yksinkertainen. Jokainen avattavaan CSV-tiedostoon kirjoitettu rivi luetaan erikseen. Rivien sisältämät arvot paloitellaan sanakirja-muotoon.

Paloiteltu sanakirja sisältää arvot address, id, meta ja v. Meta-arvo kertoo suoraan mihin listavalintaan viitataan käyttöliittymässä. Viitattu lista päivitetään sanakirjan avaimen v:n arvoilla. Address ja id on tässä tapauksessa vain ylimääräistä pois suljettavaa dataa. Mikäli ladattavassa tiedostossa on arvoja tunnistamattomassa muodossa, ne tallennetaan erilliseen lisäasetuksia sisältävään muuttujaan. Tällöin id ja v ovat rivin tärkeintä dataa. Kuvassa 24 on esitettynä CSV-tiedoston latausprosessin vuokaavio.



Kuva 24. CSV-tiedoston latausprosessin vuokaavio.

6.2.7 Lisäasetusikkunan avaaminen ja sulkeminen

Lisäasetusikkuna avataan erillistä ”Custom Value Editor” -nappia painamalla. Tämän kyseisen ikkunan toimintoja käsitellään tarkemmin luvussa 6.3. Lisäasetusikkunan sulkeminen aiheuttaa pääikkunan puolella tarkistustoimenpiteen, mikäli ikkunasta tuodaan asetuksia. Tuodut asetukset käydään läpi ja mikäli pääikkunassa on samalla id:llä asetuksia, nämä asetukset korvataan valitsemalla ne suoraan pääikkunan listavalinnoista. Lisäasetukset tallennetaan CSV-tiedoston lataamisen tyylisesti erilliseen muuttujaan.

6.2.8 Asetuksien nollaaminen

Käyttäjä voi nollata asetukset painamalla käyttöliittymässä sijaitsevaa ”Reset” -nappia. ”Reset” -nappi yksinkertaisesti päivittää listavalinnat ja tekstinsyöttöruudut tyhjiksi. Myös lisäasetukset sisältävä muuttuja tyhjennetään.

6.2.9 Teemavalinta

Käyttöliittymään lisättiin mahdollisuus vaihtaa käytössä oleva teema. Teemat ovat sisäänrakennettuja käytettyyn PySimpleGUI-kirjastoon ja teemaa vaihtaessa koko ikkuna avataan uudelleen uutta teemaa käyttäen. Kun sovellus avataan ensimmäistä kertaa, teema on ennalta määrätty. Kun sovellus suljetaan ensimmäisen kerran se tallentaa käytössä olevan teeman erilliseen bin-tiedostoon. Kun sovellus avataan seuraavan kerran, tämä bin-tiedosto ladataan ja sen sisältämä teema asetetaan käyttöön automaattisesti.

6.3 Lisäasetusikkuna ja sen toiminnot

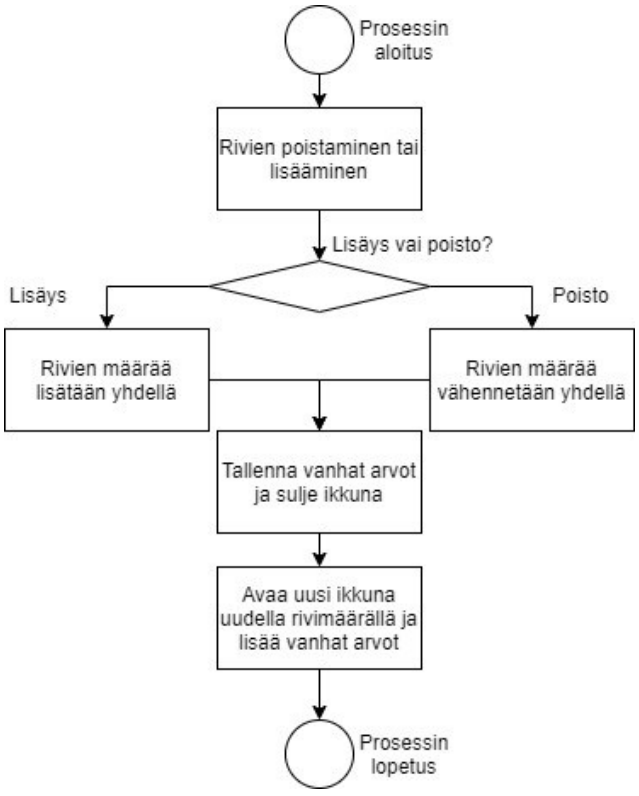
Lisäasetusikkuna mahdollistaa kustomoitavan ympäristön eri asetuksien kirjoittamiselle. Käyttäjä voi lisätä ja poistaa ikkunassa näkyviä rivejä tarpeen mukaan. Jokainen rivi rakentuu neljästä eri tekstinsyöttöä sallivasta laatikosta, joista jokaisella on oma tarkoituksensa. Perusrakenne muotoutuu samantyylisesti kuin CSV-tiedostojen rakenne. Yksi rivi sisältää ”Address”, ”NMS ID”, ”Setting Name” ja ”Setting Value” -laatikot. Pääikkunan tapaan käyttäjän on mahdollista generoida JSON-tiedostoja, tallentaa ja ladata CSV-tiedostoja, sekä antaa nimi luotaville tiedostoille. Pääikkunasta poiketen

lisäasetusikkunassa ei ole virheellisten asetuskombinaatioiden tarkistusta, mikäli sitä käytetään JSON-tiedostojen generoimiseen.

Lisäasetusikkunasta voi halutessaan tuoda asetukset pääikkunaan (import), jolloin asetuskombinaatiot tarkistetaan osittain. Virheiden tarkistus vaatii sen, että annetut asetukset ovat olemassa pääikkunan valintamahdollisuuksissa. Pääikkunasta puuttuvat asetukset tallennetaan erilliseen muuttujaan.

6.3.1 Rivien lisääminen ja poistaminen

Lisäasetusikkunan rivien lisääminen ja poistaminen tapahtuu erillisillä ”Add Rows” ja ”Delete Rows” -napeilla. Rivien lisäyksellä ja poistamisella todellisuudessa luodaan aina uusi ikkuna. Uusi ikkuna sisältää uuden määrän rivejä. Vanha ikkuna suljetaan uuden ikkunan avaamisen yhteydessä. Riveihin jo vanhassa ikkunassa kirjoitetut arvot syötetään takaisin uuden ikkunan riveihin. Kuvassa 25 on rivien käsittelyn vuokaavio.

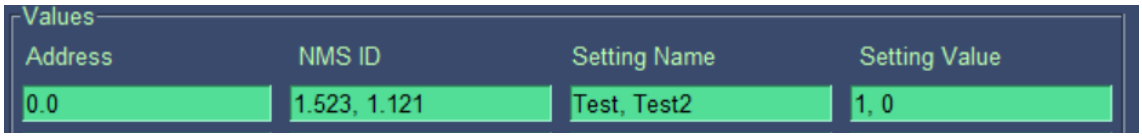


Kuva 25. Lisäasetusikkunan rivien käsittelyn vuokaavio.

6.3.2 JSON-tiedoston generointi

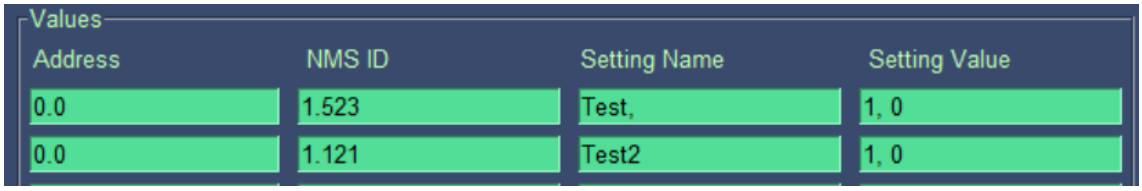
Lisäasetusikkunasta on mahdollista generoida pääikkunan tapaan suoraan JSON-päätteisiä testiprofiileja. Pääikkunasta poiketen prosessi on erilainen. Erillistä asetuskombinaation yhteensopivuutta varmistavaa tarkistusta ei tehdä vaan kaikki asetukset käydään riveittäin läpi ja lisätään profiileihin.

”NMS ID”, ”Setting Name” ja ”Setting Value” -laatikkoihin on mahdollista syöttää useampi arvo pilkulla erotettuna. Tällöin ”NMS ID” ja ”Setting Name” -arvot sisällytetään samaan profiiliin samalla annetulla ”Setting Value” -arvolla. Ominaisuutta käytetään silloin kun halutaan, että kaksi asetusta sisältää saman arvon jokaisessa profiilissa. Esimerkkinä käytöstä on radiomodeemin vastaanotto ja lähetystaajuus, jotka pyritään testauksessa pitämään samoina. Kuvassa 26 esimerkki useamman arvon syöttämisestä.



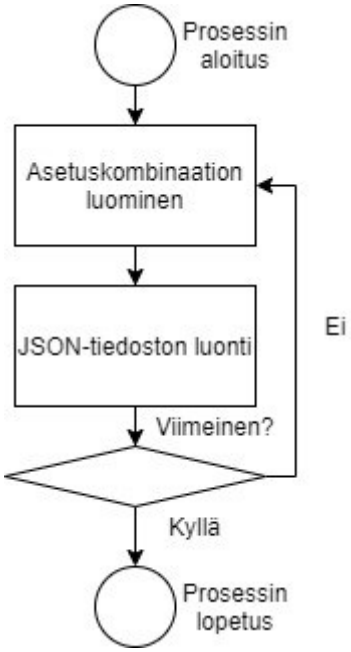
Kuva 26. Esimerkki rivien täyttämisestä lisäasetusikkunassa.

Kuvan 26 tyylisesti syöttäen arvoista muodostuisi kaksi profiilia. ”Setting Value” -kohdan arvot käsitellään molemmat erilliseen profiiliin, koska samassa ei voida asettaa sekä arvoa 1, että 0 annettuihin asetuksiin. ”NMS ID” ja ”Setting Name” -laatikot käsitellään siten, että molemmille asetuksille annetaan arvo 1 toisessa profiilissa ja 0 toisessa. Mikäli arvot olisi kirjotettu molemmat omalle riville, syntyvä profiilien määrä kasvaisi neljään. Tässä yhdistelmässä käytäisiin kaikki eri 0 ja 1-arvojen välillä syntyvät kombinaatiot läpi. Kuvassa 27 esimerkki erilliselle riville syöttämisestä.



Kuva 27. Esimerkki usean rivin täyttämisestä lisäasetusikkunassa.

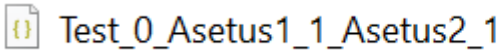
Pääikkunasta poiketen lisäasetusikkunan kautta generoinnissa ei luoda erikseen CSV-tiedostoa jokaisesta kombinaatiosta. Kuvassa 28 esitettynä JSON-tiedoston generoinnin vuokaavio lisäasetusikkunassa.



Kuva 28. Lisäasetusikkunan JSON-tiedoston generoimisen vuokaavio.

6.3.3 Tiedostojen nimeäminen

Tiedostojen nimeämistyyli poikkeaa hieman pääikkunan tyylistä. Lisäasetusikkunasta generoidessa, jokaisen asetuksen nimi ja arvo lisätään alaviivalla erotettuna tiedoston nimeen. Nimeämistyylin muutos helpottaa uusien tuntemattomien asetuksien tunnistamista testivaiheessa. Kuvassa 29 on esimerkki nimeämistyylistä.



Kuva 29. Lisäasetusikkunan nimeämistyyli.

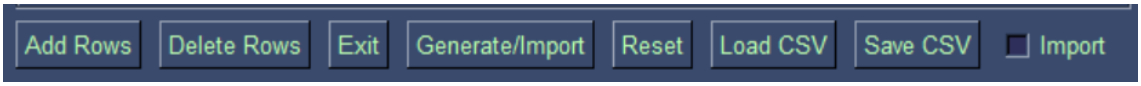
Kuvassa 29 tiedoston nimi jakautuu näin:

- Test on itse profiilille annettu nimi.
- 0 on profiilin numero.
- Asetus1_1 tarkoittaa, että Asetus1:lle on annettu arvo 1.
- Asetus2_1 tarkoittaa, että Asetus2:lle on annettu arvo 1.

6.3.4 Asetusten tuominen pääikkunaan

Asetusten tuominen pääikkunaan on ominaisuus, mikä mahdollistaa asetuksien siirtämisen pääikkunan puolelle. Mikäli käyttäjän syöttämissä asetuksissa on samoja arvoja kuin pääikkunassa, esimerkiksi FEC, tulevat ne automaattisesti valituksi FEC-kohdan listavalintaan. Mikäli asetus on kirjoitettu virheellisesti, se jätetään huomioimatta pääikkunan valinnoissa.

Asetuksen tuominen käynnistyy, kun käyttäjä asettaa Import valintalaatikon käyttöön ja painaa Generate/Import nappia. Kuvassa 30 on esitetty Import valintalaatikko, sekä muut lisäasetusikkunan eri valintamahdollisuudet.

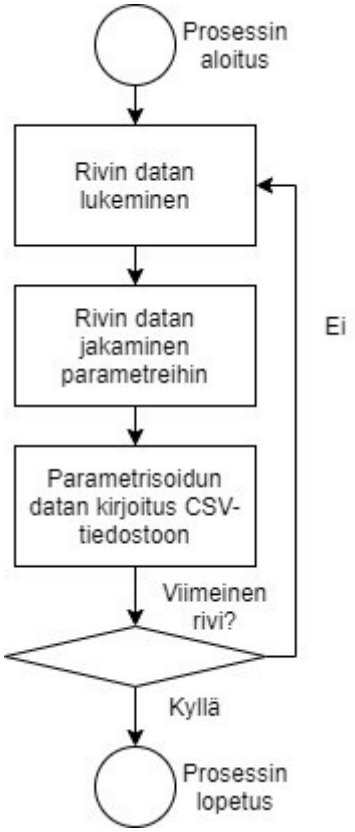


Kuva 30. Lisäasetusikkunan eri valintamahdollisuudet.

6.3.5 Tallentaminen CSV-tiedostoon

Lisäasetusikkunan CSV tallennuksen toiminta vastaa pääikkunan toimintaa. Kuten JSON-tiedostojen generoinnin kanssa, virheellisiä kombinaatioita ei tarkisteta. Datan lukeminen ikkunan riveistä on suoraviivaista ja ne kirjoitetaan suoraan riveittäin CSV-tiedostoon.

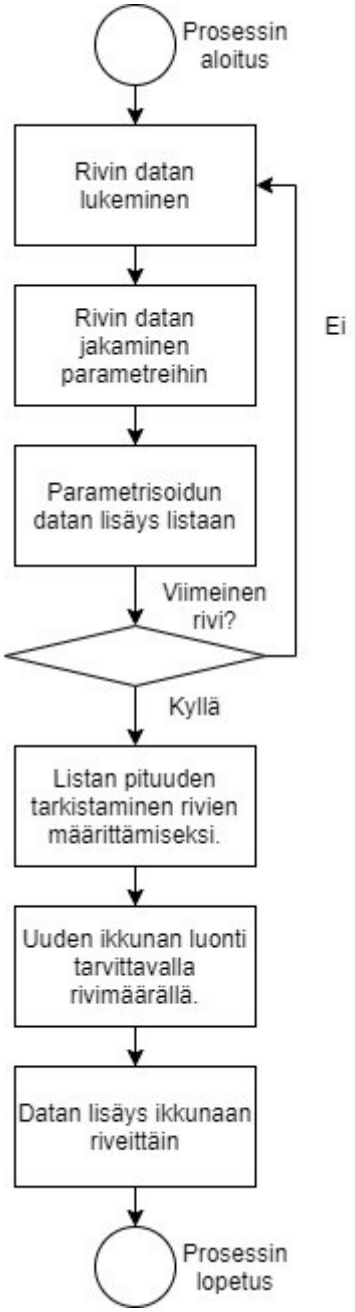
Rivien sisältö lisätään yksitellen listaan ja sen jälkeen sisältö jaetaan riveittäin 4:lle eri muuttujalle. Nämä muuttujat vastaavat CSV-tiedostoon kirjoitettavaa rakennetta ja myös itse ikkunan rivirakennetta. Kuvassa 31 on esitettynä tallentamisen vuokaavio.



Kuva 31. Lisäasetusikkunan tallennusfunktion vuokaavio.

6.3.6 Lataaminen CSV-tiedostosta

CSV-tiedostosta lataaminen vastaa toiminnallisuudeltaan lähes tallentamista. Asiat tehdään vain vastakkaisessa järjestyksessä. Lataustoimintoa käyttäessä tarkistetaan tiedostossa oleva rivimäärä. Tämän perusteella asetetaan uuteen ikkunaan myös vastaava rivimäärä. Tämän jälkeen tiedot luetaan riveittäin jakaen eri pystysarakkeet omaksi rivilaatikkoihin kirjoitettaviksi parametreiksi. Kuvassa 32 on CSV-tiedoston lataamisen vuokaavio.



Kuva 32. CSV-tiedoston lataamisfunktion vuokaavio.

7 OHJELMISTON TESTAUS, KÄYTTÖÖNOTTO JA TULOKSIEN ESITTELY

Tässä luvussa keskitytään lähinnä ohjelmiston toiminnan testaamiseen sen kehityksessä. Samalla esitetään valmiin ohjelman exe-tiedoston tuottamiseen käytetyn PyInstaller-moduulin toiminta.

7.1 Ohjelmiston testaaminen kehityksessä

Ohjelmiston testausta suoritettiin jatkuvasti kehityksen aikana. Jokainen uusi ominaisuus testattiin regressiotestauksen tyylisesti. Regressiotestauksessa varmistetaan, että uudet ominaisuudet eivät riko vanhojen ominaisuuksien toimintaa. (Baresi & Pezzè 2006.)

Sovelluksen kehittämisessä ei ollut mukana kuin yksi ihminen ja ohjelmistotestaus tapahtui kehittäjän toimesta. Testauksessa määritettiin, mitä jokaisen toiminnon tulisi tehdä. Mikäli testattava toiminto teki jotain muuta, tutkittiin miksi ja tehtiin korjaus. Korjauksen jälkeen testattiin toiminto uudestaan. Tätä toistettiin, kunnes toiminto oli korjattu.

Tämän jatkuvan testaamisen lisäksi sovelluksesta lähetettiin työkavereille useita kehityksen alla olevia versioita käytettäväksi ja kokeiltavaksi. Vaikka tämä ei ollut osa varsinaista testausta, siitä saatiin arvokkaita ehdotuksia toimintojen parantamiseksi ja uusien toimintojen luomiseksi.

7.2 Exe-tiedoston luonti PyInstallerin avulla

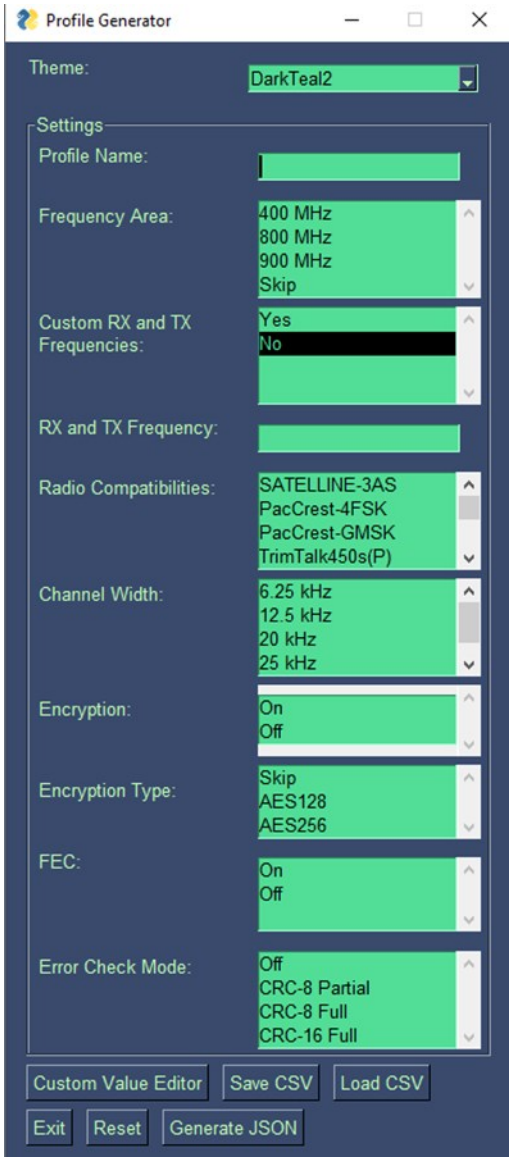
Sovellusta luodessa päätettiin, että on parempi pakata se yhdeksi ajettavaksi exe-tiedostoksi. PyInstaller päätyi olemaan pakkaamiseen sopiva ratkaisu. PyInstaller on Python-ohjelmointikielelle kehitetty sovellus, joka pakkaa kirjoitetut ohjelmistokoodit itsenäisiksi ajettaviksi exe-tiedostoiksi. PyInstaller toimii useassa käyttöjärjestelmässä kuten Windows, Linux, Mac OS X, FreeBSD, Solaris ja AIX. PyInstallerin paketoima exe-tiedosto ei ole riippuvainen ohjelmistokoodin sisältämistä moduuleista ja sen ajamiseen ei tarvita erillistä Python-tulkkia. Tämä mahdollistaa sillä luodun ohjelmiston käyttämisen usealla eri laitteella ilman erillisien ohjelmistojen asentamista. (PyInstaller, 2021)

PyInstallerin käyttö oli helppoa ja suoraviivaista, PyInstaller asennettiin ensin koneelle käyttämällä Python-ohjelmointikielen pip paketinhallintatyökalua. PyInstalleria käytettiin komentoriviltä. Ensin avattiin komentorivillä hakemisto, joka sisälsi paketoitavat tiedostot. Tämän jälkeen ajettiin PyInstallerin käynnistyksen aloittava komento. Kun komento oli suoritettu loppuun, lopputuloksena oli ajettava exe-tiedosto. Exe-tiedoston valmistuttua sen toiminta testattiin ja se todettiin toimivaksi.

7.3 Tuloksien esittely ja pohdinta

Valmis sovellus jakautui kahteen erilliseen ikkunaan, pääikkuna ja lisäasetusikkuna. Pääikkunaan sisällytettiin radiomodeemin yleisessä käytössä olevat asetukset, jotka on mahdollista valita niille luoduista listoista. Toimintoina pääikkunaan kehitettiin testiprofiilien luominen, CSV-tiedoston tallentaminen ja lataaminen, lisäasetusikkunan avaaminen, asetuksien nollaaminen, teeman vaihtaminen, sekä sovelluksen sulkeminen. Edellä mainittujen toimintojen lisäksi käyttäjän on mahdollista nimetä luodut testiprofiilit, sekä tallennetut CSV-tiedostot. Pääikkunassa valituista asetuksista muodostuvat asetuskombinaatiot tarkistetaan. Tarkistuksen aikana löydetyt virheelliset asetuskombinaatiot poistetaan automaattisesti. Kuvassa 33 on esiteltynä sovelluksen pääikkuna kokonaisuudessaan. Kuvassa esiintyvä theme-valintalaatikko on teeman valitsemista varten. Settings-osion sisällä on kaikki pääikkunan kautta laitteelle määriteltävät asetukset. Settings-osion alapuolella olevat napit ovat aiemmin määritettyjä toimintoja varten. Custom Value Editor -nappi avaa lisäasetusikkunan. Save CSV -nappi tallentaa CSV-tiedoston valituilla asetuksilla. Load CSV -nappi lataa asetukset käyttöliittymään, valitusta CSV-tiedostosta. Exit-nappi sulkee sovelluksen. Reset-nappi nollaa

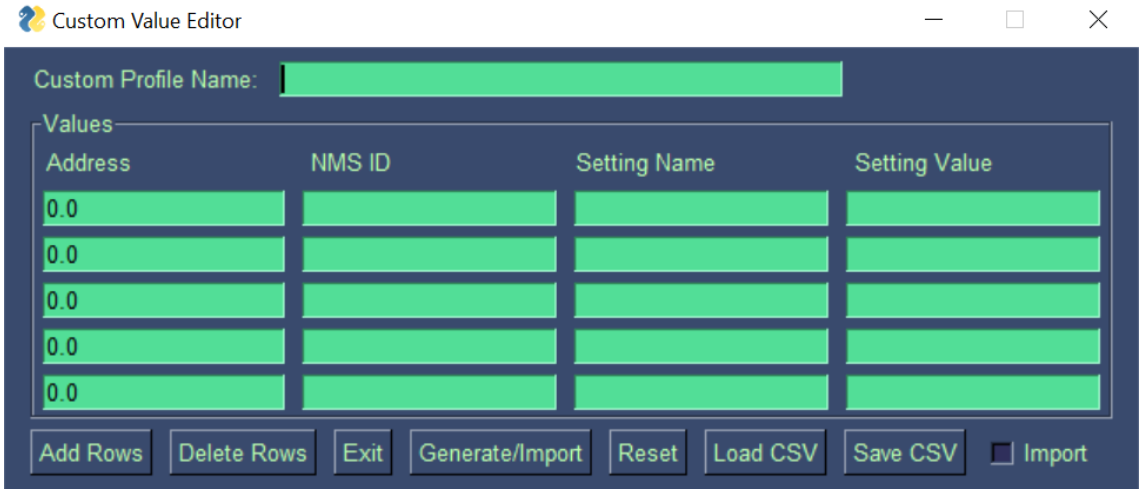
käyttöliittymässä esiintyvät valinnat. Generate JSON -nappi aloittaa testiprofiilien luomisprosessin.



Kuva 33. Sovelluksen pääikkuna

Lisäasetusikkunassa säilytettiin vanhan komentorivipohjaisen JavaScript-ohjelmiston tyyppinen toiminta. Käyttäjän on mahdollista lisätä ja poistaa rivejä tarvittaessa ja muodostaa CSV-tiedoston tyylisesti taulukko, joka sisältää kaikki halutut testiasetukset arvoineen ja laitteistokoodeineen. Lisäasetusikkunalla mahdollistetaan harvinaisempien ja laitteistokohtaisempien asetuksien käyttöönotto. Lisäasetusikkunan käyttö on mahdollista yhdistää pääikkunan kanssa tuomalla kirjatut asetukset pääikkunan puolelle. Kaikki pääikkunan tavalliset toiminnot mahdollistettiin myös lisäasetusikkunassa.

Lisäasetusikkunan käyttäminen asetuksien tuottamisessa vaatii huolellisuutta, sillä kirjattuja asetuksia ei ole mahdollista tarkistaa. Kuvassa 34 on esiteltynä lisäasetusikkuna kokonaisuudessaan. Kuvassa käyttäjä on luonut näkyville 5 riviä. Asetuksille on mahdollista määrittää nimi, mikäli niitä halutaan tuottaa suoraan lisäasetusikkunalla. Tämä nimi kirjataan kuvassa 34 esiintyvään Custom Profile Name laatikkoon. Values-osion sisällä on itse testiasetukset, jotka jaetaan neljään eri lohkoon. Lohkot ovat Address, NMS ID, Setting Name ja Setting Value. Values-osion alapuolella on napit ikkunan eri toimintoja varten. Add Rows -nappi lisää yhden rivin. Delete Rows -nappi poistaa yhden rivin. Exit -nappi sulkee lisäasetusikkunan. Generate/Import -nappi aloittaa testiprofiilien luomisprosessin, mikäli Import -valintalaatikko ei ole aktiivisena. Load CSV -nappi lataa asetukset käyttöliittymään, valitusta CSV-tiedostosta. Save CSV -nappi tallentaa käyttöliittymässä olevat asetukset CSV-tiedostoon. Import -valintalaatikko määrittää Generate/Import -napin toiminnan. Mikäli valintalaatikko on aktiivisena, Generate/Import -nappi tuoo kirjoitetut asetukset pääikkunan puolelle ilman JSON-tiedoston luomisprosessia. Tällöin Generate/Import -nappi sulkee lisäasetusikkunan.



Kuva 34. Sovelluksen lisäasetusikkuna

Kehitetty sovellus helpottaa Satelin testiautomaatiota ja mahdollistaa turvallisemman alustan testiasetuksien luomiselle. Sovelluksen pääikkunan käyttäjän ei tarvitse miettiä valinnoissa syntyviä mahdollisia virheitä, vaan sovellus tekee sen käyttäjän puolesta. Virheiden poissulkemisen johdosta säästetään aikaa testien epäonnistumisien selvittämisessä ja voidaan keskittyä testeissä mahdollisesti löytyviin laite- ja ohjelmistovikoihin. Uuden ohjelmiston käyttö on helppoa ja ei vaadi käyttäjältä suurempaa perehdytystä.

Vanhaa komentorivipohjaista JavaScript-ohjelmistoa ei ollut mahdollista käyttää ilman tarkempaa tietämystä sen toiminnasta.

Lisäasetusikkuna korvaa vanhan JavaScript-ohjelmiston ja sen käyttöön ei ole tarvetta palata. Lisäasetusikkunalla käyttäjän on mahdollista tehdä kaikki vanhan ohjelmiston toiminnot yhdessä ikkunassa ilman ylimääräisiä toimenpiteitä. Lisäasetusikkuna mahdollistaa vanhan ohjelmiston tapaan virheelliset asetukset. Vaikka se ei ole turvallinen suoraan käytettynä, se on alkuperäistä sovellusta huomattavasti nopeampi ja helpompi käyttää. Lisäasetusikkunan asetuksien tuomisen mahdollistava toiminto edesauttaa sovelluksen turvallisempaa käyttöä tarkistamalla asetuksia osittain.

Molemmissa ikkunoissa käytössä oleva CSV-tiedoston lataamistoiminto mahdollistaa vanhojen CSV-tiedostojen hyödyntämisen. CSV-tiedoston tallentaminen taas mahdollistaa asetuksien säilömisen myöhempää käyttöä varten. Käyttäjän ei tarvitse muistaa mitä asetuksia testikokonaisuuksissa on käytetty. Molemmat edellä mainitut toiminnot säästävät testikokonaisuuksien kehittäjiä ylimääräiseltä työltä.

Valmiin ohjelmiston pakkaaminen erilliseksi exe-tiedostoksi mahdollistaa ohjelmiston käyttämisen helposti. Käyttäjä ei tarvitse Python-ohjelmointikieltä tai muita sen käytössä olevia kirjastoja. Käyttäjän ei tarvitse erikseen perehtyä sovelluksen tekemiseen käytettyihin työkaluihin, vaan hän voi keskittyä testiasetuksien luomiseen.

Kokonaisuutena työssä siis päästiin sen asettamaan tavoitteeseen. Sovellus ei kuitenkaan ole täydellinen. Käyttöliittymän eri osien koko on suoraan määritetty, ja se voi aiheuttaa ongelmia pienien resoluutioiden kanssa. Käyttöliittymä ei välttämättä näy kunnolla pienellä resoluutiolla ja sen käyttö vaikeutuu. Käyttöliittymän responssisuuden kehityksen lisäksi uusia asetuksia lisätään mukaan tarvittaessa. Sovellusta kehitetään myös jatkossa työkavereiden ehdotuksien mukaisesti. Myös mahdolliset ohjelmiston pidempiaikaisen käytön jälkeen selvinneet ongelmat korjataan. Uskon työllä olevan positiivinen vaikutus Satelin testiautomaatiojärjestelmän kehittämisessä ja käytössä.

Työympäristön hyötyjen lisäksi sovelluksen kehittämisellä syvennyttiin lisää Python-ohjelmointikieleen ja sen kirjastojen käyttöön. PySimpleGUI-kirjasto oli käyttöliittymän kehityksessä ennestään tuntematon, mutta se osoittautui helppokäyttöiseksi. Python-ohjelmointikielen, sekä PySimpleGUI-kirjaston käyttö varmasti jatkuu myös tulevaisuudessa.

8 YHTEENVETO

Tämän opinnäytetyön tavoitteena oli kehittää testiautomaatiossa käytettävää sovellusta. Kehitettävän sovelluksen päätarkoitus oli luoda testattaville laitteille testiprofiileja, jotka sisältävät testattaviin laitteisiin asetettavia asetuksia. Työssä korvattiin vanha komentorivikäyttöinen JavaScript-sovellus uudella graafista käyttöliittymää hyödyntävällä Python-sovelluksella.

Uudessa sovelluksessa säilytettiin mahdollisuus kirjata asetuksia käsin vanhan ohjelmiston tapaan. Vanhasta sovelluksesta poiketen uuteen sovellukseen lisättiin mahdollisuus valita asetukset myös helppokäyttöisistä listavalinnoista. Pääikkunan valittavat toiminnot olivat testiprofiilien luominen, CSV-tiedoston tallentaminen ja lataaminen, lisäasetusikkunan avaaminen, asetuksien nollaaminen ja teeman vaihtaminen. Vanhassa ohjelmistossa sallittiin kaikkien eri asetuksien kirjaaminen eikä niitä tarkistettu mitenkään. Toteutettuun sovellukseen lisättiin erillinen tarkistusfunktio, jolla pyrittiin estämään virheelliset asetukset. Kaikki pääikkunan toiminnot testattiin ja todettiin toimivaksi ohjelmiston käytön aikana. Pääikkunasta tuli helppokäyttöinen ja selkeä.

Aiemmin mainittu mahdollisuus kirjata asetuksia käsin toteutettiin erikseen avattavalla lisäasetusikkunalla. Tähän ikkunaan sisällytettiin kaikki pääikkunan perustoiminnot tarkistusfunktiota lukuun ottamatta. Lisäasetusikkunalla luotiin mahdollisuus kirjata asetuksia riveittäin käsin, mahdollistaen kustomoitavien asetusprofiilien luomisen. Lisäasetusikkuna käyttää pääikkunassa asetettua teemaa. Lisäasetusikkunasta tuli pääikkunan tapaan helposti luettava ja selkeä. Siinä mahdollistettiin asetusrivien lisääminen ja poistaminen tarvittaessa, jolloin ikkunaan saatiin näkymään käyttäjälle tarvittava määrä rivejä.

Lisäasetusikkunaan luotiin mahdollisuus tuoda asetukset pääikkunaan, jolloin käyttäjä voi ensin valita pääikkunan listoista kaikki perusasetukset ja lisäksi kirjoittaa lisäasetusikkunaan tarvittaessa erikoisemmat asetukset. Lisäasetusikkunan lopullisiksi toiminnoiksi jäivät rivien lisääminen ja poistaminen, profiilien generointi, asetuksien tuominen pääikkunaan, asetuksien nollaaminen, CSV-tiedostojen tallentaminen ja

lataaminen. Kaikki ominaisuudet testattiin ja todettiin toimiviksi ohjelmiston käytön ja kehityksen aikana.

Ohjelmisto jää kehittäjän jatkuvaan kehitykseen myös tämän opinnäytetyön jälkeen. Tulevaisuudessa uusien laitteiden kehittämisen myötä saatetaan tarvita uusia asetuksia, jotka voidaan lisätä mukaan ohjelmistoon. Samalla seurataan myös muiden käyttäjien myötä syntyneitä kehitysehdotuksia ja mahdollisia ohjelmistosta löytyneitä ongelmia. Käyttöliittymä on tällä hetkellä lukittu tiettyyn kokoon ja se aiheuttaa ongelmia, jos käytössä on pieni resoluutio. Tulevaisuudessa tähän ongelmaan voitaisiin puuttua ja varmistaa toiminta kaikilla resoluutioilla.

Kokonaisuutena toteutuksessa päästiin sen asettamaan tavoitteeseen ja saatiin luotua helppokäyttöisempi ja turvallisempi sovellus alkuperäisen tilalle. Valmis sovellus helpottaa ja nopeuttaa testiautomaatiossa käytettävien testiasetustiedostojen luomista ja samalla varmistaa, ettei valituissa testisasetuksissa ole virheitä.

LÄHTEET

Baresi, L. & Pezzè, M. 2006, "An Introduction to Software Testing", Electronic notes in theoretical computer science, vol. 148, no. 1, pp. 89-111.

Brainboxes 2021. What is RTS / CTS Hardware Flow Control? Viitattu 18.5.2021. http://www.brainboxes.com/faq/items/what-is-rts--cts-hardware-flow-control-

Circuit Basics 2021. Basics of uart communication. Viitattu 18.5.2021. https://www.circuitbasics.com/basics-uart-communication/

Computer Hope 2021. Parity Bit. Viitattu 11.5.2021. https://www.computerhope.com/jargon/p/paritybi.htm

EnGenius 2021. What is RSSI and its acceptable signal strength?. Viitattu 6.5.2021. https://helpcenter.engeniustech.com/hc/en-us/articles/234761008-What-is-RSSI-and-its-acceptable-signal-strength-

freeCodeCamp 2021. What is a CSV File and How to Open the CSV File Format. Viitattu 18.5.2021. https://www.freecodecamp.org/news/what-is-a-csv-file-and-how-to-open-the-csv-file-format/

Inductive Automation 2021. What is scada? Viitattu 18.5.2021 https://www.inductiveautomation.com/resources/article/what-is-scada

Jenkins 2021. Jenkins User Documentation. Viitattu 15.3.2021. https://www.jenkins.io/doc

JSON 2021. Introducing JSON. Viitattu 11.5.2021. https://www.json.org/json-en.html

MDN Web Docs 2021a. Java Script Guide. Viitattu 27.4.2021. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Introduction

MDN Web Docs 2021b. Node.js. Viitattu 27.4.2021. https://developer.mozilla.org/en-US/docs/Glossary/Node.js

Novatron 2021. What is machine control? Viitattu 18.5.2021 https://novatron.fi/en/what-is-machine-control/

Oracle 2021. What Is IoT? Viitattu 18.5.2021 https://www.oracle.com/internet-of-things/what-is-iot/

Python 2021a. General Python FAQ. Viitattu 15.3.2021. https://docs.python.org/3/faq/general.html

Python 2021b. Data Structures. Viitattu 6.5.2021. https://docs.python.org/3/tutorial/datastructures.html

PyInstaller 2021. PyInstaller Manual. Viitattu 15.3.2021. https://pyinstaller.readthedocs.io/en/stable

PySimpleGUI 2021. PySimpleGUI User's Manual. Viitattu 15.3.2021 https://pysimplegui.readthedocs.io/en/latest/

Robot Framework 2021. Introduction. Viitattu 15.3.2021. https://robotframework.org/#introduction

Reiman, P. 2017, "Radiomodeemin ohjelmistotestauksen automatisointi". Pro gradu – tutkielma. Turun Yliopisto. Tulevaisuuden Teknologioiden laitos. Viitattu 22.3.2021

Satel 2021a. Satel. Viitattu 13.5.2021. https://www.satel.com/satel/

Satel 2021b. Radio Modems. Viitattu 14.4.2021. https://www.satel.com/products/radio-modems

Satel 2021c. Applications. Viitattu 14.4.2021. https://www.satel.com/applications/

Satel 2021d. Environmental Monitoring. Viitattu 14.4.2021. https://www.satel.com/applications/environmental-monitoring/

Satel 2021e. Industrial Internet. Viitattu 14.4.2021. https://www.satel.com/applications/industrial-internet/

Subversion 2021. Subversion FAQ. Viitattu 14.4.2021. https://subversion.apache.org/faq.html#collab

Symmetry Electronics 2015. What is the Difference Between GNSS and GPS? Viitattu 18.5.2021 https://www.semiconductorstore.com/blog/2015/What-is-the-Difference-Between-GNSS-and-GPS/1550/

Tutorialspoint 2021. Forward Error Correction (FEC). Viitattu 18.5.2021 https://www.tutorialspoint.com/forward-error-correction-fec

W3Schools 2021. Introduction to HTML. Viitattu 18.5.2021. https://www.w3schools.com/html/html_intro.asp

Xilinx 2021. What is an FGPA?. Viitattu 18.5.2021. https://www.xilinx.com/products/silicon-devices/fpga/what-is-an-fpga.html