Haaga-Helia
ammattikorkeakoulu Oy

Radio-olosuhteiden vaikutus jatkuvan integraation testituloksiin

Niko Tiittanen

Opinnäytetyö
Tietojenkäsittelyn
koulutusohjelma
2020

Tiivistelmä

Haaga-Helia ammattikorkeakoulu Oy

Tekijä
Niko Tiittanen

Koulutusohjelma
Tietojenkäsittelyn koulutusohjelma

Raportin/Opinnäytetyön nimi
Radio-olosuhteiden vaikutus jatkuvan integraation testituloksiin

Sivu- ja liitesivumäärä
22 + 16

Opinnäytetyön tarkoituksena oli tehdä Python ohjelma, joka laskisi yksittäiselle testilinjalle puhelimen latausnopeuden rajan jatkuvan integraation 5G-pakettien testiympäristössä. Ohjelmassa laskettiin puhelimelta saatujen arvojen perusteella, verraten laskelmoitua arvoa teoreettiseen maksimiin. Raportissa tarkastellaan ohjelman luomista alusta alkaen ja arvioidaan omaa oppimista. Raportissa myös käydään läpi ohjelmaan liittyviä kaavoja, hieman tietoliikennettä ja 5G-ympäristöä.

Tarve projektille syntyi, kun tarkasteltiin latausnopeustestejä 5G-tukiasemalla, jossa testien raja-arvot tehtiin kyvykkyysmittarista saatujen tulosten perusteella. Kyvykkyysmittaus oli hyvin tehty, mutta itse tulokset eivät aina täsmänneet rajaan, vaikka kyseessä olisi ollut kaksi identtistä ympäristöä. Projektilla haluttiin saada lataustestien raja-arvoista yksi radio-olosuhteista riippuva asia poistettua, jotta itse rajat voisivat olla enemmän ympäristökohtaisia.

Työtä varten täytyi saada puhelimelta erilaisia arvoja, jotka vaikuttivat sen toimintatehoihin latausnopeuden muodossa. Kyseessä olevat arvot olivat puhelimelta saatuja antennien signaali-kohinasuhteen arvoja.

Puhelimelta saatiin arvoja Qualcomm-yrityksen lisensoidulla QXDM-ohjelmalla ja arvot muunnettiin tekstimuotoon saman yrityksen QCAT-ohjelmalla. Raportissa käydään läpi kuinka arvot saatiin, miten tieto jäsenneltiin sekä millä tavoin laskelmat toteutettiin.

Tavoitteena oli saada jäsenneltyä tekstitiedostoista halutut arvot tietorakenteeseen ja laskea niistä matemaattisen kaavan avulla tietoliikennekanavan teoreettinen yläraja, jolla tietoa voidaan kuljettaa luotettavasti. Kyseessä olevat arvot olivat puhelimelta saatuja antennien signaali-kohinasuhteen arvoja. Puhelimelta saatuja arvoja verrattiin 5G-tukiaseman kyvykkyyslaskurin laskemaan teoreettiseen maksimiarvoon.

Asiasanat
5G-tekniikka, Python, analyysi

Sisällysluettelo

Johdanto

Opinnäytetyö tehtiin Nokia Oyj:lle Espoossa. Projekti tehtiin jatkuvan integraation ryhmän sisällä, ja olin projektin aikana palkattuna Nokian työntekijänä kyseisessä ryhmässä. Raportin alussa käydään läpi työn tavoitteita ja tietoperustaa, jonka jälkeen siirrytään ohjelman toiminnallisuuteen. Ohjelman toiminnallisuudesta siirrytään tulosten esittämiseen ja pohditaan niiden merkillisyyttä. Osa tiedosta on peitetty tai esitetty tavalla, jolla ulkopuolisten osapuolien tai heidän ohjelmiensa oikeuksia ei rikota.

1.1 Projektin tarve

Projekti käynnistettiin, koska nykyisillä jatkuvan integraation 5G-pakettien testilinjoilla puhelimien latausarvot saattoivat vaihdella puhelimen asennosta ja ympäröivistä radio-olosuhteista johtuen. Testi koostui 5G-tukiaseman kyvykkyyden testaamisesta latausnopeustestin avulla puhelimella. Testeihin asetetut rajat olivat kiinteät jokaisella testattavalla tukiasemalla. Kiinteistä rajoista haluttiin luopua, joten tarve paremmalle tavalle syntyi. Tällä projektilla pyrittiin korvaamaan nykyiset kiinteät raja-arvot.

Aiemmin raja-arvot olivat asetettu manuaalisesti ajamalla lataustestejä ja tarkastelemalla silmämääräisesti tuloksista paljon rajojen tulisi olla ja olivatko ne kyvykkyysmittarin mukaisia arvoja. Tämän takia päätettiin, että raja-arvot tulisivat perustua johonkin kaavaan, jonka avulla saataisi konkreettinen yksittäiseen testi-ympäristöön perustuva tulos. Kaavan avulla saataisi rajat asetettua tavalla, josta olisi poistettu kokonaan signaalinkohinan suhdearvot, jotta testit olisivat luotettavampia.

1.2 Tavoitteen saavuttaminen

Alun perin projektin ohjelmallista ratkaisua olisi kokeiltu muutaman kerran testiympäristöä vasten ja tarkasteltu siitä saatuja tuloksia. Tavoite jouduttiin lopulta muuttamaan projektin aikana testiympäristöön tapahtuneiden muutosten takia. Uusi tavoite oli tarkastella ohjelman tekoa, omaa oppimista ja projektista saatua palautetta. Tämän lisäksi kuitenkin tarkastellaan ohjelman laskelmallista ratkaisua omassa osiossaan ja tehdään johtopäätöksiä testitiedoston perusteella.

Raportin tavoitteena on kertoa, kuinka ohjelmisto-ongelman kautta rajattiin tavoite tieteellisempien testiraja-arvojen saavuttamiseksi. Tavoitteena oli rakentaa Python-ohjelma, jossa jäsenneltiin halutut arvot tietorakenteeksi ja laskettiin niiden avulla paljon latauksen raja-arvo tulisi olla. Ohjelmaan täytyi rakentaa tietoa jäsentävä osuus, jonka avulla pystyttiin tutkia puhelimelta saatuja arvoja ja laskea niitä halutun matemaattisen kaavan avulla.

Työssä tarkasteltiin yhtä 5G-tukiasemaa ja sen puhelimelta saatuja arvoja. Tukiaseman puhelimelta otettiin yksi mittaus QXDM-ohjelmalla, joka kesti noin kaksi minuuttia. QXDM-ohjelmasta saatu tiedosto muunnettiin QCAT-ohjelmalla tekstitiedostoksi. Mittauksen perusteella rakennettiin ohjelma, joka luki tekstitiedostoa, jäsensi tiedon haluttuun muotoon ja laski saaduista arvoista teoreettisen maksimin, sekä vertasi sitä tukiaseman teoreettiseen maksimiin.

2 Yleistä

Ohjelmisto-ongelmia tyypillisesti voidaan ratkaista erilaisia ohjelmistontuotannon menetelmiä käyttäen, kuten vesiputous tai ketterä kehitys. Menetelmän valinta tulee heijastaa ohjelmiston luonteenpiirteitä, laajuutta ja loppukäyttäjää. Ketterää kehitysmenetelmää voidaan käyttää nopeita muutoksia ja taipuvuutta vaadittavissa ohjelmistonkehityksissä, joissa pienemmät ryhmät tekevät yhteistyötä ja tapaavat usein. Vesiputousmalli voi toimia paremmin isompien ryhmien projekteissa, joissa voidaan odottaa suurempien projektin osien valmistumista ja ylipäätään projektin valmistumisella on suurempi tavoiteaika. (Customer Think 2018.)

Projektissa käytettäväksi kehitysmenetelmäksi valittiin ketterä menetelmä, koska se oli CI-ryhmäni käyttämä menetelmä. Ketterästä menetelmästä oli hyötyä projektin eri osioiden työstämisessä, koska sitä työstettiin melko pienessä CI-ryhmässä, jossa oli useita kehityskohteita auki jatkuvasti. Projektin ketteryyden avulla pystyin tekemään koulutöitä, eri työpaikan projekteja ja työstää opinnäytetyön projektia pienissä paloissa. Projektin jatkokehitys tapahtuu ketterän kehitysmenetelmän mukaisesti CI-ympäristössä.

Projekti perustuu 5G-tukiaseman lataustestien muodostamaan ongelmaan. Ongelma rajattiin 5G-tukiaseman radion ja puhelimen välisten arvojen tarkasteluun Python ohjelman avulla. Python ohjelman tarkoitus oli jäsentää sille annetusta tekstitiedostosta halutut arvot tietorakenteeksi, ja laskea tietorakenteista jokaiselle arvolle matemaattisen kaavan avulla tulos, jota vertaillaan teoreettiseen maksimiarvoon.

Ohjelman rakentamista lähdettiin ratkomaan pala kerrallaan, alkaen tiedonsaannista. Tiedonsaantia varten täytyi rajata testiympäristö ja käytetyt ohjelmat. Käytetyillä ohjelmilla saatiin mittaustieto puhelimelta ja se muunnettiin tekstitiedostoksi. Tekstitiedoston avulla pystyttiin rakentamaan ohjelman tiedonjäsennys-osuus, jossa tekstitiedosto jaettiin tietorakenteiksi. Tietorakenteita varten täytyi määritellä käytetyt kirjastot ja rakenteiden järkevin muoto laskemista varten. Laskeminen toteutettiin etsimällä tieteellistä tapaa laskea tietoliikennekanavan teoreettinen maksimi, johon löydettiin matemaattinen kaava kollegan neuvosta. Teoreettista maksimia lopulta vertailtiin Nokian sisäisen kyvykkyyslaskurin antamaan tulokseen, koska se oli aiemmin käytetty raja-arvon malli.

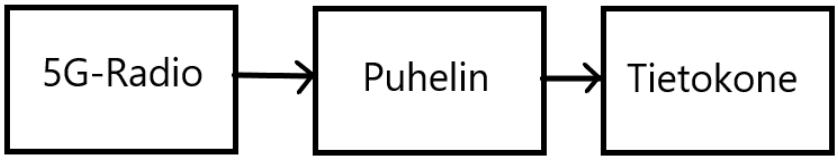
2.1 Ympäristö

Projekti suoritettiin osana CI-ryhmää. Continuous Integration (CI) on kehitysmenetelmä, jossa kehitettyä koodia lisätään yhteiseen jaettuun säilytyspaikkaan usein. CI-ympäristössä voidaan yhdistää tuotettu koodi automaatiotestaukseen, jonka avulla käytetty koodi

on luotettavaa. Automaattinen CI-ympäristön testaus varmistaa koodin olevan aina toimivaa, jolloin virheet huomataan todella nopeasti ja versiota voidaan peruuttaa helposti. Luotettavuus on hyvä asia koodin julkaisussa, koska hyvässä CI-järjestelmässä ei tarvitse pelätä niin paljon kaiken hajoamista. (CodeShip.)

CI-ryhmässä työskentely on jatkuvan integraation lisäksi jatkuvaa oppimista, sillä viikoittain pääsee lisäämään erilaisiin projekteihin omia luomuksia, joita varten täytyy opetella erilaisia teknologioita ja ohjelmointikieliä. CI-ympäristössä olen nähnyt automaation ja optimoinnin tulokset asioissa, kuten koodin testauksessa ja muutoksien julkaisemisen nopeudessa.

Testiympäristö koostui yhdestä 5G-tukiaseman radiosta, testipuhelimesta ja tietokoneesta, jolla kaapattiin haluttuja arvoja puhelimelta USB-yhteyden välityksellä. Testipuhelimessa oli käytössä Qualcomm Snapdragon X55 5G-modeemi. Snapdragon X55 5G-modeemi on yrityksen Qualcomm tuottama järjestelmä, jolla voidaan hyödyntää melkein mitä vain taajuuskaistan tai radiotaajuuden käytännön yhdistelmää, ja sillä voi jopa päästä 7,5 gigabittiä sekunnissa latausnopeuksiin (Qualcomm 2019). Käytössä oleva tietokone oli Windows-pohjainen kannettava tietokone, jossa käytettiin virtualisointina Ubuntu Linux käyttöjärjestelmää (Kuva 1).



Kuva 1. Testiympäristö arvojen tallentamista varten.

Viidennen sukupolven ”New Radio” puhelintiedonvälityksen standardi, jossa on käytössä uusi radiokäyttöliittymä (Radio Interface, RI) ja radioliityntäverkko (Radio Access Network, RAN). Radiotaajuudet menevät uudella sukupolvella kahden ja puolen gigahertsin ja 40 gigahertsin välillä, mutta suurta osaa näistä taajuusväleistä ei olla vielä otettu käyttöön (electronicsnotes). Viidennen sukupolven tarkoituksena on parantaa aiemman sukupolven yhteyksien nopeuksia kymmenkertaisiksi, jonka avulla käyttäjillä voi olla tosiaikaisia yhteyksiä toistensa välillä. Nopeat yhteydet auttavat myös virtuaalisen- ja lisätyn todellisuuden tuomista jokapäiväiseksi työ- ja viihdevälineeksi. Nopeudet mahdollistavat myös älylaitteiden lisääntymisen jokaiseen kotiin ja työpaikkaan. (Thomson.)

2.1.1 Kuinka puhelimelta saatiin tarkasteltavat arvot

Tiedon saanti puhelimelta tapahtui erilaisten lisensoitujen työkalujen avulla. Itse sanomien keräykseen käytettiin QXDM-ohjelmaa. Qualcomm eXtensible Diagnostic Monitor (QXDM) on Qualcommin lisensoima diagnostiikka työkalu, jonka avulla voidaan kaapata ja tarkastella sisäisiä viestejä, kuten diagnostiikka lokeja, viestejä ja tapahtumia ohjelma- ja rautatasolla. Ohjelma toimii USB-yhteydellä tietokoneelta puhelimeen ja siinä on oma käyttöliittymänsä. Ohjelman avulla voidaan luoda tiedosto, joka sisältää kyseiset tallennetut tiedot. (Qualcomm, 2020.)

Näitä sisäisiä sanomia tarvittiin projektia varten luodussa ohjelmassa tiedonjäsennyksen ja laskujen kanssa. Sanomien tarkastelua varten puhelinta kohti lähetettiin iperf3-työkalun avulla User Datagram Protocol (UDP)-dataa. Iperf3 on työkalu, jonka avulla voidaan aktiivisesti testata suurinta saavutettavaa kaistanleveyttä IP-verkossa. Se tukee useita puskureita ja protokollia, kuten TCP, UDP, SCTP ja toimii IPv4 ja IPV6 -osoitteiden kanssa. (iperf.fr 2020.)

Lopulta QCAT-ohjelman avulla muunnettiin QXDM-ohjelmasta saatu mittaustiedosto tekstimuotoon. Qualcomm Commercial Analytics Tool (QCAT) on Qualcomm yrityksen QXDM Professional lisenssiin kuuluva työkalu, jonka käyttöliittymän avulla voidaan tarkastella, suodattaa tai viedä tekstitiedostoksi QXDM-ohjelmasta saatuja isf-tyyppisiä tiedostoja. (Qualcomm 2020.)

3 Python

Ohjelmassa käytettiin ohjelmointikielenä pelkkää Pythonia ja sen omia ja ulkopuolisia moduuleita sekä kirjastoja. Projektin ohjelmointikielenä käytettiin Pythonia, koska CI-ryhmäni muissa työkaluissa käytettiin pääasiassa samaa kieltä. Tämän lisäksi Python oli kokemani sekä kuulemani perusteella tiedonkäsittelyyn erittäin sopiva ja vahva ohjelmointikieli. Projektissa ohjelmasta tehtiin ”perinteinen” Python-ohjelma, jolla on ”.py” -pääte ja se sisältää main() -funktion.

Python on Guido Van Rossumin vuonna 1991 julkaisema tulkattava, interaktiivinen ja olioihin-pohjautuva ohjelmointikieli. Pythonin säätiö on voittoa tavoittelematon organisaatio ja omistavat Python 2.1 ja tulevien versioiden tekijänoikeudet. Python tuo mukanaan moduuleja, poikkeukset, dynaamisen kirjoittamisen, korkean tason dynaamiset tietotyypit ja luokat. Ohjelmointikieli tukee myös olio-ohjelmoinnin ulkopuolisia ajatusmalleja, kuten proseduurillista ja funktionaalista. (Python.)

Python oli vielä vuoden 2020-alussa minulle todella uusi, mutta kiehtova ohjelmointikieli. Opin projektin aikana Pythonista ja tiedonkäsittelystä paljon uutta, vaikka en ollutkaan lopuksi täysin tyytyväinen kirjoittamaani ohjelmaan, koska halusin tehdä kaiken optimaalisemmin aina opittuani lisää. Jatkokehitysideoita tuli useita, joita raportissa mainitsen ohjelman eri toiminnollisuuksia läpikäydessäni.

3.1 Moduulit

Moduuli on tiedosto, joka sisältää Python määritelmiä ja lausekkeita. Useita Python moduuleja sisältäviä tiedostoja kutsutaan ohjelmaksi (script). Moduuleja voidaan kutsua eri tiedostoihin tai tiedostojen main() -funktioon. Moduulitiedoston pääte on ”.py”. (Python 2020, 6.)

3.2 Kirjastot

Python sisältää oman peruskirjastonsa, jotka kattavat laajan alueen erilaisia toiminnollisuuksia. Python on Unix-pohjaisilla tietokoneilla pakettien kautta laajennettavissa, kun taas Windows-pohjaisilla tietokoneilla saattaa asennusohjelman mukana tulla ylimääräisiä kirjastoja. Lisää kirjastoja, Python ohjelmia, moduuleja, paketteja ja rajapintoja voi löytää Pythonin paketti indeksistä. (Python 2020.)

4 Tiedon jäsentäminen

Tiedonjäsentäminen liittyy suuresti datatieteeseen. Datatieteen avulla yritetään ennakoida ja selittää ilmiöitä. Ilmiöiden selittämiseen ja ennakointiin käytetään erilaisia laskentatapoja, kuten algoritmeja (Harju 2017). Projektissa käytetyn Pythonin asema datatieteessä, on hyvä, koska se soveltuu monenlaiseen ongelmanratkaisuun. Projektissa käytetty Pythonille tehty Pandas on yksi laajimmista tiedonkäsittelykirjastoista. Pandas-kirjaston lisäksi Pythonille on lukuisia muita hyviä tiedonkäsittelyn kirjastoja ja suuri yhteisö, jolta voi pyytää apua ongelmiin. (data science graduate programs 2020.) Ohjelmassa tietoa jäsenneltiin tarkastelemalla haluttuja arvoja puhelimelta saatujen mittausten tekstitiedostosta ja muokkaamalla ne lopulliseen tietorakenteeseen.

4.1 Kirjastot

Projektissa käytettiin avoimen lähdekoodin Python kirjastoja NumPy, Pandas ja Matplotlib tiedonjäsentelyyn sekä visualisointiin. Valitsin kirjastot niiden ilmaisten lisenssien lisäksi, koska olin käyttänyt niitä kerran aiemmin työssäni, joten tiesin niiden olevan helppokäyttöisiä ja tehokkaita. Halusin myös oppia kyseisistä ohjelmista lisää. En tutustunut muihin tiedonkäsittely kirjastoihin Pythonille ennen projektia, aikaisempien kokemuksieni vuoksi.

Käytin kirjastojen ohjelmoinnin avoimia rajapintoja projektissa. Application programming interface (API) on avoin rajapinta, joka tarkoittaa sen olevan julkisesti käytettävissä ja ilmainen. Avoimen ohjelmointirajapinnan täytyy olla julkisesti dokumentoitu tarpeeksi tarkasti, jotta se voidaan helposti ottaa käyttöön. Ohjelmointirajapintoja on kahdenlaisia, toiminnallisia- ja datarajapintoja. Toiminnallisesta ohjelmointirajapinnasta voidaan saada kaavoja tai laskenta-algoritmeja, jotka voivat ”muuttaa järjestelmän tietoja rajapinnan kautta.” Datarajapinnan kautta pystytään saamaan tai lukemaan palvelun sisältämää dataa. (avoinrajapinta 2014, versio 1.0.)

4.1.1 NumPy

NumPy, eli Numerical Python on vapaan lähdekoodin Python kirjasto, jolla voidaan tehdä moniulotteisia taulukoita ja matriisi tietorakenteita. NumPy on yksi tehokkaimpia Python kirjastoja ja sillä voidaan tehdä yksiulotteisia tai moniulotteisia taulukkoja. Yksiulotteiset taulut ovat yhden rivin tai yhden sarakkeen kokoisia. Moniulotteiset taulukot voivat olla monirivisiä taulukoita, jotka sisältävät useita sarakkeita. (medium 2020.)

4.1.2 Pandas

Wes Mckinneyn ja pandas-ryhmän (2020, 1.4.2) mukaan pandas on vapaan lähdekoodin data-analyysi kirjasto Pythonille. Pandas-kirjasto on perustettu Numpyn päälle ja se on tarkoitettu helposti lisättäväksi tieteelliseen laskelmointiympäristöön.

Pandas-kirjaston avulla voidaan tuottaa yksiulotteisia sarja-olioita tai kaksiulotteisia datakehyksiksi kutsuttavia olioita. Sarja-olioilla on periaatteessa lista, jossa näkyy indeksinumerot. Datakehys olioilla ovat omat nimet, rivit, sarakkeet ja omat indeksinumeronsa, kuten taulukointiohjelmissa yleensä näkisi (Kuvat 2 ja 3).

```
In [3]: s = pd.Series([1, 3, 5, np.nan, 6, 8])

In [4]: s
Out[4]:
0    1.0
1    3.0
2    5.0
3    NaN
4    6.0
5    8.0
dtype: float64
```

Kuva 2. Yksiulotteisen pandas sarja-olion luonti (pandas, 2020, 2.1.1.)

```
Out[6]:
DatetimeIndex(['2013-01-01', '2013-01-02', '2013-01-03', '2013-01-04',
               '2013-01-05', '2013-01-06'],
              dtype='datetime64[ns]', freq='D')

In [7]: df = pd.DataFrame(np.random.randn(6, 4), index=dates, columns=list('ABCD'))

In [8]: df
Out[8]:
                   A         B         C         D
2013-01-01 -0.015961  0.551206  0.319515 -0.095693
2013-01-02  0.862218  1.644702 -0.316382  0.746015
2013-01-03  1.494845 -1.474497 -1.046234  0.003673
2013-01-04  0.849866  1.383550  0.402318  0.789805
2013-01-05  0.778614 -0.078109 -0.391629 -0.953737
2013-01-06 -0.168529 -0.228812  1.117952 -0.999618
```

Kuva 3. Pandas datakehys-olion luonti (pandas 2020, 2.1.1.)

Pandas soveltuu hyvin monien erilaisten tilastotieteellisten tietolähteiden, kuten SQL ja Excel tapaisten analyysiin, eikä tiedon tarvitse olla järjesteltyä. Tietoa voidaan kirjaston avulla helposti muokata, tallentaa tai lukea. (Bronshtein 2017.)

4.1.3 Matplotlib

Matplotlib on avoimen lähdekoodin Python-kirjasto, jonka tarkoituksena on visualisoida haluttuja tuloksia kaksiulotteisilla- ja hieman kolmiulotteisilla grafiikoilla. Kirjaston alkuperä

on MATLAB-graafisissa komennoissa, mutta se on nykyään erillinen. Matplotlib käyttää Pandas-kirjaston tavoin NumPy-kirjastoa alustana, jotta se toimisi paremmin suurten taulukoiden kanssa työskentelyssä. Kirjasto on tehty yksinkertaisuus mielessä, jotta tuloksia voisi visualisoida vain yhtä tai muutamaa komentoa käyttäen. (Hunter, Dale, Firing & Droettboom 2013.)

Olin kerran aiemmin kokeillut eri työkalua tehdessäni Matplotlib-kirjastoa, mutta se ei ollut silti minulle kovin hyvin muistissa. Kirjastolle löytyi paljon tukimateriaalia ja ohjeita, joten niitä käyttäen sain visualisoitua tietorakenteita melko hyvin. Matplotlibin käyttäminen Pandas-tietorakenteen kanssa oli todella yksinkertaista.

4.2 Tiedonkäsittely ohjelmassa

Ohjelman tiedonkäsittelyssä tarkasteltiin tekstitiedostoa, josta tarkoituksena oli saada haluttuja arvoja tietorakenteisiin. Arvojen ymmärtämiseksi täytyi opetella niihin liittyvät termit, mutta toteutin tiedon jäsentämisen ohjelmallisesti ennen termien ymmärtämistä.

4.2.1 Oleelliset termit

Ohjelman tarkasteltava tekstitiedosto sisälsi tiettyjä arvoja erilaisissa taulukoissa. Arvoja oli SSB- tai TRS lähteestä ja taulukot sisälsivät SNR-arvoja jokaista puhelimen antennia kohden.

Synchronization Signal Block on lohko, joka sisältää synkronointisignaalin sekä PBCH (Physical Broadcast Channel) tiedot, eli käytännössä downlink-signaaleja. Nämä molemmat yhdistyvät yhdeksi lohkoksi, joka liikkuu aina yhdessä. (Sharetechnote.)

TRS, eli Tracking Reference Signal kertoo antennin portin, joka tulisi olla sama kaikkien kyseisen referenssi signaalin resurssin käyttäjien välillä (Sharetechnote).

Tärkeimpänä terminä kuitenkin toimii SNR, sillä se on oleellinen projektin kokonaisuuden ymmärtämiseksi. Signal-to-Noise ratio (SNR), eli signaali-kohinasuhde on matemaattinen lauseke, jossa lasketaan signaalin voima jaettuna kohinan voimalla (Kuva 4).

Kuva 4. Signaali-kohinasuhde (Sharetechnote)

SNR esitetään yleensä desibeleinä ja se voi olla positiivinen tai negatiivinen arvo. Negatiivinen SNR-arvo tarkoittaa signaalin voiman olevan pienempi kuin kohinan voima. SNR on yksi parhaita signaalin voimakkuuden mittareita, ja hyvä signaali kertoo virheiden tapahtumismahdollisuuksista järjestelmässä. (Sharetechnote.)

4.2.2 Tiedon jäsentäminen ja rakenne

Puhelimelta saatiin sen neljän antennin SNR-arvot mittausten aikaväliltä noin 20-40 mikrosekunnin välein. Arvot saatiin QXDM ja QCAT ohjelmien avulla tekstitiedostoksi, josta niitä lähdettiin tutkimaan.

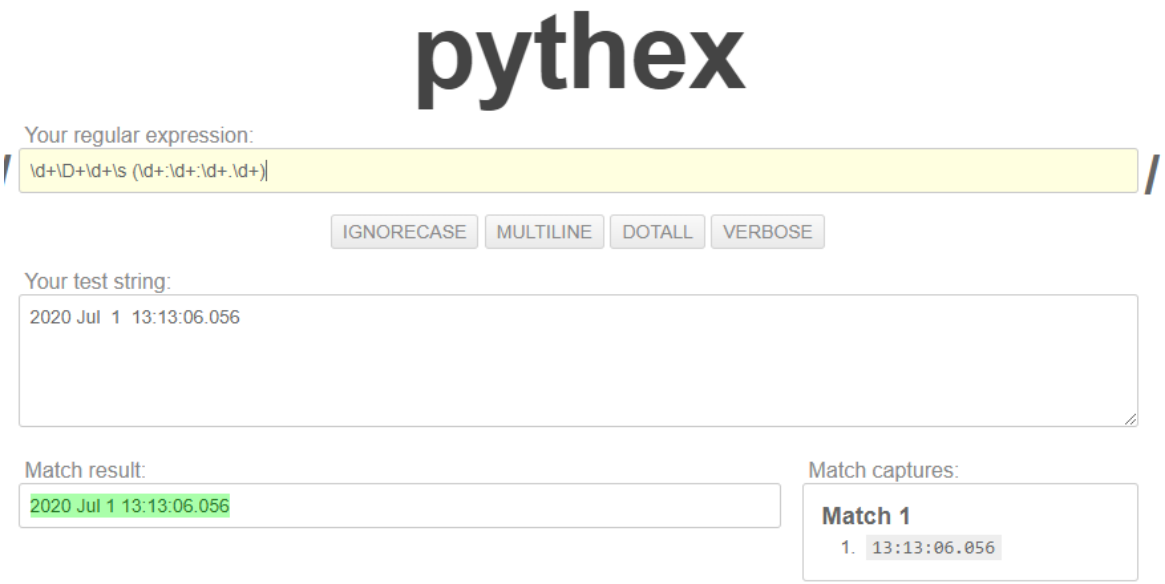
SNR-arvojen lisäksi tekstitiedostosta haluttiin tieto, oliko kyseinen mittaus SSB vai TRS. Tärkeää tämä oli siksi, että SSB-mittauksista saattoi puuttua viimeisten kahden antennin arvot. Tämän lisäksi halusin, että ohjelmassa vertailtaisi sekä SSB- ja TRS arvoja kokonaisuutena, että pelkkinä SSB- tai TRS-arvoina. Muita arvoja, joita tiedostosta haluttiin, olivat aikaleima ja antennin numero. Pelkkä aikaleima ei kuitenkaan riittänyt, sillä samalla aikaleimalla saattoi olla erikseen kaksi mittaustaulukkoa, joten lisättiin tunniste. Tekstitiedoston rivit pysyivät rakenteeltaan samoina koko tiedoston läpi, joten niistä täytyi hakea erikseen säännöllisten lausekkeiden avulla haluttujen rivien arvot.

Regular expression (Regex, RE), eli säännöllinen lauseke on erikoinen koodattu merkkijono, jolla pyritään löytää siihen täsmääviä merkkijonoja. Säännölliset lausekkeet saattavat tuntua aivan uudelta kieleltä aluksi, mutta niiden osaaminen auttaa tiedonjäsentelyssä suuria määriä tietoa tarkasteltaessa. (Computer Hope 2020.) Pythonissa säännölliset lausekkeet toimivat re-moduulin avulla. Pythonissa on oma Perl-ohjelmointikieltä muistuttava syntaksi, jossa voidaan käyttää erikoisia- tai tavallisia merkkejä. Pythonissa on pitkä lista erikoismerkkejä, joilla voidaan määritellä esimerkiksi kuinka muut merkit niiden ympärillä toimivat. Pythonissa yleensä aloitetaan säännöllinen lauseke r-kirjaimella, joka tekee merkkijonosta raakamerkkijonon. Raakamerkkijonoissa ei käsitellä aloittavaa kenoviivaa

erikoistavalla, esimerkiksi r”\n” tulkittaisi kenoviivana ”\” ja erillisenä ”n”-merkkinä, kun pelkkä ”\n”-merkkijono tulkittaisi uutena rivinä (Python 2020).

En ollut ennen projektia kovin monesti käyttänyt säännöllisiä lausekkeita ja olin ennen kysellyt paljon kollegoilta apua niiden muodostamiseen. Tämän projektin aikana en halunnut kysellä apua, vaan halusin opetella itse kaiken tarvittavan ja haastaa itseäni. Muutaman kerran jouduin kuitenkin kysymään apua, vaikka pääsinkin oppimaan paljon.

Muodostin säännölliset lausekkeet etsimällä ensiksi niiden perusteista tietoa, jota löysin paljon Pythonin omilta sivuilta. Tein haut ensiksi kokeilemalla käsin Python-tulkilla, mutta se tuntui todella hitaalta tavalta kokeilla ja oppia. Siirryin seuraavaksi etsimään internetistä hakukoneella erilaisia säännöllisten lausekkeiden kokeiluun tehtyjä työkaluja, joita löytyi paljon. Kovin monta työkalua ei oltu tehty suoraan Pythonia varten, joten sellaisen etsiminen kesti melko kauan. Lopulta löysin yhden Pythonia varten tehdyn Pythex-työkalun, joka tuntui soveltuvan hyvin säännöllisten lausekkeiden kokeiluihin. Pythex-työkalusta näkee myös, kuinka sulkujen sisään laitettu merkkijono ryhmittää sen, jotta hausta voidaan erottaa haluttu osa (Kuva 6.)



Kuva 6. Pythex-työkalun avulla säännöllisen lausekkeen kokeilu. (Pythex)

Taulukko 1. Tekstitiedoston sisältämät rivit ja niistä halutut arvot

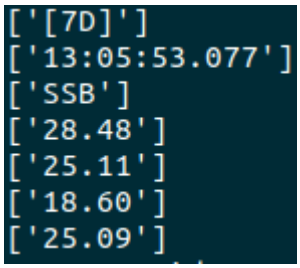
Rivi	**Haku 1.**	**Haku 2.**
2020 Jul 1 13:05:53:077 [AB]	13:05:53:077	[AB]
SSB Or TRS = SSB	SSB	

Antenni 1 SNR = 10.10	10.10	
Antenni 2 SNR = 11.20	11.20	
Antenni 3 SNR = 12.30	12.30	
Antenni 4 SNR = 13.40	13.40	

Esimerkki yhdestä tekstitiedoston tietorakenteesta, jossa näkyy säännöllisiin lausekkeisiin perustuva runko. Esimerkkitaulukkoon ovat merkittyinä suunnilleen, miltä haettavat rivit ja niistä otetut arvot näyttivät. Taulukosta ja lopullisesta lähdekoodin liitteestä on jätetty pois kohdat, jotka saattaisivat loukata QXDM-lisenssiä. Taulukossa ovat merkattuina rivit, joita etsittiin säännöllisten lausekkeiden avulla, ”Haku 1”-alla ovat ensimmäiset riviltä halutut arvot ja ”Haku 2”-alla on toinen arvo, mikäli sellaista etsittiin. Näitä arvoja pystyttiin säännöllisten lausekkeiden avulla täsmäämään ja lisäämään haluttuihin listoihin (Taulukko 1).

4.2.3 Listat, taulukot ja tietorakenteet

QCAT-ohjelmasta saadusta tekstitiedostosta täytyi eritellä tietoa erillisiin listoihin. Tarkasteltavasta tekstitiedostosta eriteltiin omiksi listoikseen mittauksen tunniste, aikaleima, tyyppi (SSB tai TRS), SNR-arvot sekä jokaiselle antennille tyhjät listat. Listoihin saatiin arvot re-moduulin ”match”-toiminnollisuudella, jossa haettiin säännöllisten lausekkeiden hakuihin täsmääviä arvoja. Hauilla saatiin myös SNR-arvot järjestyksessä asetettua jokaisen antennin omaan listaan (Kuva 7).



Kuva 7. Ohjelman yllä mainittujen listojen ensimmäiset arvot

Projektissa ei välttämättä olisi tarvittu NumPy-kirjastoa ollenkaan, mutta en osannut ohjelman teon aikana muuntaa Python-listoja suoraan Pandas-kirjaston tietorakenteeksi, joten NumPy oli taulukointia varten hyvä ja tehokas valinta. Aiemmin mainitut listat koostettiin NumPy-taulukoksi, josta alkoi muodostumaan jo selkeämpi tietorakenne. NumPy-taulukossa näkyi selkeästi jokaisella rivillä kyseessä oleva tunniste, aikaleima, mittauksen tyyppi ja arvot jokaiselle antennille (Kuva 8).

```
[['[7D]' '13:05:53.077' 'SSB' '28.48' '25.11' '18.60' '25.09']
 ['[D9]' '13:05:53.077' 'SSB' '11.42' '12.43' '9.68' '13.31']
 ['[EE]' '13:05:53.097' 'TRS' '32.96' '29.19' '31.66' '33.69']
 ['[46]' '13:05:53.097' 'SSB' '12.44' '13.29' '11.97' '14.08']
 ['[5A]' '13:05:53.137' 'SSB' '13.00' '13.90' 'NA' 'NA']
```

Kuva 8. NumPy-taulukko, jossa näkyy myös puuttuvia SSB-arvoja

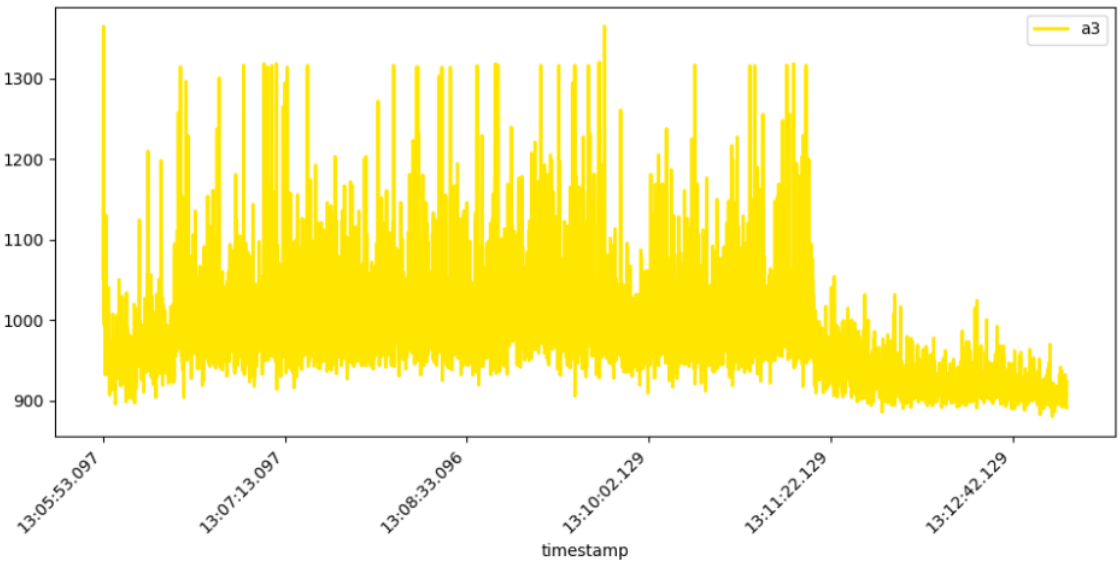
NumPy-taulukko muunnettiin Pandas-kirjaston datakehys-olioksi. Tieto olisi voitu pitää NumPyn taulukkona, mutta datakehys-oliota käytettiin, jotta voitaisi hyödyntää Pandas-kirjaston toiminnollisuuksia eri arvojen laskemiseen, kuten keskiarvo, maksimi ja minimi. Pandasin toiminnollisuuksien avulla pystyttiin myös lisäämään sarakkeille omat tunnisteet ja järjestysnumeron olisi voinut vielä korvata tunnistesarakkeella halutessaan, mutta tietorakenne oli jo selkeä liukuvalla järjestysnumerolla (Kuva 9).

	id	timestamp	type	a1	a2	a3	a4
0	[7D]	13:05:53.077	SSB	28.48	25.11	18.60	25.09
1	[D9]	13:05:53.077	SSB	11.42	12.43	9.68	13.31
2	[EE]	13:05:53.097	TRS	32.96	29.19	31.66	33.69
3	[46]	13:05:53.097	SSB	12.44	13.29	11.97	14.08
4	[5A]	13:05:53.137	SSB	13.00	13.90	NA	NA
...	...	...	...	...	...	...	...
26493	[4E]	13:13:05.996	SSB	29.85	32.30	42.66	33.27
26494	[C7]	13:13:06.036	SSB	29.77	32.25	42.66	35.07
26495	[46]	13:13:06.036	SSB	30.63	31.16	NA	NA
26496	[5E]	13:13:06.056	TRS	26.35	39.60	27.80	39.43
26497	[EA]	13:13:06.056	SSB	29.61	37.94	42.78	33.04

Kuva 9. NumPy taulukosta koostettu Pandas-datakehys

4.2.4 Visualisointi

Visualisointi auttoi hahmottamaan, kuinka arvot jakautuvat eri tietorakenteiden välillä. Visualisointi tuli minulle melkein jälkiajatuksena, kun halusin vain nähdä pystyisikö tietorakenteiden arvoja visualisoida paremmin. Olin positiivisesti yllättynyt Matplotlib-kirjaston helposta käytöstä ja koen oppineeni paljon sen käytöstä. Lopputulos visualisoinnista ei kuitenkaan ollut aivan paras, sillä en käyttänyt siihen mielestäni tarpeeksi aikaa, enkä tutkinut mitkä arvot voisivat olla tärkeimpiä visualisoida. (Kuva 10).



Kuva 10. Matplotlib esimerkki yhden antennin tuloksista.

5 Laskelmat

Tietorakenteesta löytyvien SNR-arvojen muuntaminen teoreettiseksi tiedonsiirron maksimiarvoksi vaati Shannon-Hartley matemaattisen lausekkeen sekä hieman taustatietoja. Itse laskutoimitus tehtiin muokkaamalla kaava Python-funktioksi ja kutsumalla sitä tietorakenteiden sisältämiä SNR-arvoja vasten.

5.1 Shannon-Hartley matemaattinen lauseke

Ted Myersin (2016) mukaan Shannon-Hartley matemaattinen lauseke ilmaisee maksimimäärän virheetöntä digitaalista tietoa, jota voidaan kuljettaa kommunikaatiokanavaa pitkin tietyllä kaistanleveydellä ja kohinan määrällä. Kaavassa C kuvaa kanavan suuruutta bitteinä sekunnissa, B on yhdensuuntainen kaistanleveys hertseinä ja S/N on aiemmin käsitelty SNR, jota ilmaistaan myös hertseinä, eikä desibeleinä. (Kuva 11).

$$Capacity\ \lim it, C = B \log_2\left(1+\frac{S}{N}\right)$$

Kuva 11. Shannon-Hartley matemaattinen lauseke (University of Manchester 2006.)

Pelkkä Shannon-Hartley matemaattinen lauseke ei riitä tähän tarkoitukseen koska SNR-arvot ovat niin suuria. Suuria SNR-arvoja ei voitu käyttää, koska ne eivät tuottaneet kaavan avulla haluttuja tuloksia, vaan tulokset olivat noin kymmenesosan halutusta määrästä. Manchesterin yliopiston (2006, 7.2) mukaan tapauksessa, jossa S/N >> 1, on kanavan arvo ~0.332 kerrottuna kaistanleveydellä ja vielä kerrottuna SNR-arvolla desibeleinä (Kuva 12).

$$C = \frac{1}{\log_{10} 2} B \log_{10}\left(1+\frac{S}{N}\right) \approx 3.32\, B \log_{10}\left(1+\frac{S}{N}\right)$$

Kuva 12. Suurien SNR-arvojen tapauksien kaava (University of Manchester 2006.)

5.2 Ohjelman laskelmat

Sisäisellä kyvykkyyslaskurilla pystyttiin laskemaan teoreettinen kattoraja 5G-tukiaseman latausnopeudelle, jota pystyttiin vertailemaan ohjelman tuloksiin. Kyvykkyyslaskurin tulos testattavalle tukiasemalla oli 1412 megabittiä sekunnissa, ja sitä käytettiin tämän raportin esimerkkituloksissa. Ohjelmassa saatiin minimit, maksimit ja keskiarvot jokaisesta tietorakenteesta Pandas-kirjaston min-, max- ja avg ominaisuuksilla. Laskut tehtiin jokaisessa

funktiossa erikseen, koska aina tietokehyksiä palauttaessa arvoilta katosivat tietotyypit. Tietotyyppien katoamista varten tietorakenteen arvoille asetettiin tietotyypit aina ennen laskutoimituksia.

Koin parhaaksi tavaksi käyttää kaavaa Pythonissa funktiona, joten asetin Shannon-Hartley-kaavan Python-funktioksi. Funktiona kaava oli hyvä, koska se ei tuota erillisenä funktiona mitään sivuvaikutuksia, kun sitä kutsuu haluamassaan kohdassaan se ottaa halutun arvon sisään ja palauttaa sille kaavan avulla lasketun arvon. Funktiona se on myös helppo yksikkö testata, sillä se ottaa aina arvona kokonaisluvun tai desimaalin (Kuva 13).

```
def band_limited_calculation(snr):
    return 0.332 * 100 * snr
```

Kuva 13. Shannon-hartley-kaavan käyttö Python funktiona.

Esimerkkinä ohjelman tuottamista tuloksista, joissa on laskettu Shannon-Hartley -kaavalla kanavan kyvykkyyden maksimiraja. Arvot ovat saatu yhdistämällä jokaisen antennin arvojen minimit, maksimit ja keskiarvot ja jakamalla jokainen niistä antennien määrällä. Taulukossa näkyy kaksi samaa maksimiarvoa, koska kaikkien antennien maksimiarvot tapahtuivat SSB-tietorakenteessa. Maksimiarvot ovat myös hetkellisesti suurempia, kuin kyvykkyyslaskurin teoreettinen maksimi. Kyseessä olevassa testitiedostossa oli noin satatuhatta riviä, joten oletettavaakin olisi, että arvot voisivat hetkellisesti ylittää tasapäistetyn laskennallisen teoreettisen arvon (Taulukko 2).

Taulukko 2. Esimerkki ohjelman testituloksista ilman vertailua teoreettiseen maksimiin

Tietorakenne	**Minimi**	**Maksimi**	**Keskiarvo**
SSB & TRS	388,77	1434,82	1134,44
TRS	913,5	1355,64	1067,11
SSB	388,77	1434,82	1157,44

Lopullinen laskutoimitus tapahtui kaikkien antennien SNR-arvojen kaavan avulla tuottamasta keskiarvosta, joka jaettiin teoreettisella maksimilla, ja siitä saatu prosenttimäärä vielä kerrottiin teoreettisella maksimilla. (Taulukko 3).

Taulukko 3. Esimerkki ohjelman avulla testitiedostosta lasketusta raja-arvosta

Tietorakenne	**Ohjelman suosittelema raja (Mbits/s)**
SSB ja TRS	1129,6
TRS	1073,12
SSB	1157,84

Mielestäni tämä tapa, jossa vertaillaan kaikkien antennien keskiarvoa teoreettiseen maksimiin tuottaa järkevimmän tuloksen. Ohjelmaa pitäisi kokeilla lisää, jotta saisi testattua tuloksien vaihtelun määrän. Tulos on mielestäni realistinen kyseisen tukiaseman lataustestin rajaksi, joten tästä voi päätellä kaavan toimivan halutulla tavalla.

6 Tulokset

Opinnäytetyön alkuperäisessä suunnitelmassani oli tarkoituksena tehdä kolme 30 sekunnin pituista mittausta QXDM-avulla, muuntaa se tekstiksi QCAT-ohjelmalla ja lopuksi jäsentää sekä laskea ohjelmani avulla teoreettinen latausnopeuden maksimi. Tämä suunnitelma muuttui lopulta, koska testiympäristöjen ohjelmiston testaukset vaativat asetusten muuttamista, joka tarkoitti puhelimelta saatavan vain kahden antennin tulokset. Koin kahden antennin olevan liian vähän, joten tulokset perustuvat nyt ohjelman tekoprosessiin, omaan oppimiseeni ja sidosryhmien tyytyväisyyteen.

Taulukko 4. Sidosryhmien tyytyväisyyskysely

Kysymys	**Toimeksiantaja**	**Ohjaaja**
1. Projektin tavoite saavutettiin hyvin	5	4
2. Projekti suoritettiin ajanhallinnallisesti hyvin	5	3
3. Ohjelma on helposti ymmärrettävissä	4	4
4. Ohjelmasta saatu tulos on tavoitteen mukainen	4	5
5. Ohjelmasta on paljon hyötyä	4	4
6. Ohjelmalla on paljon jatkokehitysmahdollisuuksia	5	5

Projektista ja sen aikana luodusta ohjelmasta pyydettiin palautetta kyselylomakkeen muodossa. Kyselystä saadun palautteen keskiarvoksi tuli 4,33, joka pyöristyy alaspäin arvosanaksi neljä. Tämä arvosana kertoo projektin sujuneen todella hyvin, mutta ei täydellisesti. Olen tästä itse samaa mieltä, ettei projektin kaikki osuudet sujuneet aivan täydellisesti, kuten ajanhallinta osioiden valmistumisessa. Minulle jäi projektista positiivinen mielikuva, koska pääsin oppimaan paljon uutta melko haastavasta aiheesta. Arvosana neljä on mielestäni erittäin hyvä arvosana kaikkeen vaivaan nähden (Taulukko 4).

Ohjaajalta saadun kirjallisen palautteen mukaan ohjelma on asiallisen näköinen, mutta säännölliset lausekkeet olisivat vaatineet hieman enemmän ajatusta. Palautteen mukaan säännöllisissä lausekkeissa olisi voitu miettiä haettuja arvoja tarkemmin, eikä hakea koko riviä kerrallaan. Erikoismerkkien käyttöä olisi ohjaajan mukaan voitu miettiä myös säännöllisissä lausekkeissa tarkemmin, jotta säännölliset lausekkeet löytäisivät kirjaimellisesti halutun arvon. Olen ohjelman palautteesta samaa mieltä ja totesin jo aiemmin raportissa olleeni aloittelija säännöllisissä lausekkeissa. Tämän rakentavan palautteen avulla voidaan parantaa ja lyhentää ohjelman säännöllisiä lausekkeita löytämään tarkemmin haluttuja arvoja.

6.1 Hyödynnettävyys ja jatkokehitys

Ohjelmaa voidaan hyödyntää tarkasteltaessa yksittäisten testiympäristöjen lataustestien rajoja, kuten suunnitelmanakin oli. Ohjelma voi myös toimia pohjana uusien matemaattisten työkalujen teossa. Jatkokehitysideoiden avulla, työkalu voidaan myös lisätä CI-ympäristöön parannuksin, ja ideat toimivat myös hyvin muistiinpanoina sen työstämiseksi.

Yleisiä lisäyksiä ohjelmaan voisivat olla argumentit, ohjelman tapahtumien kirjaus ”logger”-avulla, tietokantaan tuloksen tallennus ja web-näkymän kehittäminen. Ohjelman voisi myös pilkkoa kahteen isompaan luokkaan, joista toinen olisi tiedonjäsennystä varten ja toinen laskuja varten. Automaatiomielessä ohjelmaa voisi parantaa hakemalla tiedon automaattisesti QXDM-ohjelmasta ja kääntää siitä saadun tiedoston tekstiksi QCAT-avulla.

Argumentteja lisäämällä ohjelmasta tulisi käyttäjäystävällisempi, eikä jokaisen käyttäjän tarvitsisi muuttaa ohjelman arvoja sitä käyttäessään. Hyviä argumenttilisäyksiä voisivat olla teoreettisen maksimin ja kyseisen radion kaistanleveyden antaminen argumenttina, jotta ohjelmassa ei olisi asetettu staattisia arvoja. Muita lisättäviä argumentteja voisivat olla tiedoston nimen polkuna antaminen ja tapahtumien kirjaamista varten kirjaamistaso, esimerkiksi ”info”, ”debug” ja ”warn”.

6.1.1 Tiedonjäsennys ja visualisointi

Jatkokehityksenä visualisointia voisi parantaa näyttämään tarkemmin eri tietoja, kuten vain maksimit tai keskiarvot. Nykyisessä muodossaan visualisoinnista ei saa mielestäni tarpeeksi tietoa, eikä se ole suorastaan yksiselitteistä, koska antennien väliset erot ovat niin suuret. Tuloksia täytyisi visualisoida enemmän jotta niistä selviäsi tärkeimmät tiedot. Visualisoinnissa voisi myös tarkastella jokaisen sekunnin keskiarvotulosta jokaisen antennin kohdalta ja näyttää myös antennien keskiarvot yhtenä kuvana. Jos tuloksia lisätään tulevaisuudessa web-näkymään, voisi Bokeh olla siinä tapauksessa parempi visualisoinnin suhteen, koska se on tehty web-sivuja varten ja siinä pystyy esimerkiksi tarkentamaan haluttua tulosta. Visualisointiin argumenttina voisi myös lisätä ”visualize”-nimisen argumentin, jonka arvoja voisivat olla ”all”, ”none”, ”ssb” ja ”trs”.

6.1.2 Laskelmat

Laskujen olisi parempi tapahtua yhdessä funktiosta tai luokassa, eikä jokaisessa funktiossa erikseen niin kuin tällä hetkellä. Ohjelma voisi myös tarkastaa SNR-arvojen suuruuden ja käyttää niiden mukaisesti kaavoja, koska SNR-arvot eivät aina ole todella paljon suurempia kuin yksi.

7 Johtopäätökset

Lopputuloksena saavutettiin tavoite luomalla dynaamiset raja-arvot latausnopeustestiä varten 5G-tukiasemalle. Raja-arvot saavutettiin ohjelmallisesti mielestäni hyvin, vaikka keksin paljon kehitysideoita ja parannuksia. Ohjelma toimii suunnitelman mukaisesti ja laskee radion antennien teoreettisen tiedonsiirtonopeuden Shannon-Hartley matemaattisen lausekkeen avulla.

Projekti sujui mielestäni hyvin muutamia vaikeuksia lukuun ottamatta. Projektissa ei ollut alkuun kovin selkeä kuva tavoitteesta, joka piti muokata moneen kertaan uudelleen, jotta päästiin nykyiseen muotoon. Suunnitelmaan olisi voitu käyttää alussa enemmän aikaa, jotta projektissa olisi päästy nopeammin liikkeelle. Projektin aikana myös tapahtui toimistomuutto, joka vaikeutti tuloksien saamista oikealta ympäristöltä. Muuton lisäksi ympäristön testattavan ohjelmiston kanssa tuli vaikeuksia. Tästä huomasin, kuinka herkkiä fyysiset laitteet ovat ja miten ympäristöstä riippuvainen oma ohjelmani on. Ajankohdan olisi voinut muuton takia valita paremmin, mutta ohjelmallinen onnistuminen on mielestäni tärkein osa projektista.

Projektin kesto myös hieman pitkittyi alkuperäisestä suunnitelmasta, mutta se saatiin silti valmiiksi ennen joulua, joka oli lopullinen aikaraja. Aikataulun venyminen johtui aiemmin mainituista vaikeuksista, sekä työn teon ja koulun käynnin välisestä aikatauluttamisesta.

Projekti syvensi osaamistani ohjelmoinnista yleisesti, sekä Pythonista ja sen käytöstä tiedonkäsittelyssä. Opin myös paljon 5G-ympäristöstä ja tietoliikenteestä projektin arvojen etsinnän ja tutkimisen kautta. Opin myös paljon asioiden priorisoinnista työn ja kouluasioiden välisen aikatauluttamisen kautta. Opin keksimään uusia tapoja toimia vaikeuksien ilmetessä, kuten alkuperäisen suunnitelman muuttaminen. Odotan innolla ohjelman jatkokehitystä, sekä integraatiota kunnolla CI-ympäristöön.

Suuri kiitos työpaikalleni mahdollisuudesta ja erityisesti suuri kiitos kollegoilleni kaikesta tuesta opinnäytetyön aikana!

Lähteet

avoinrajapinta 2014. Avoimen rajapinnan määritelmä. Versio 1.0. Luettavissa: http://avoinrajapinta.fi/. Luettu: 4.9.2020.

Bhatia, Vishal. 2018. Tips to Choose the Right Software Development Process for Your Industry. Luettavissa: https://customerthink.com/tips-to-choose-the-right-software-development-process-for-your-industry/ Luettu 16.11.2020

Bronshtein, A. 2017. A Quick Introduction to the “Pandas” Python Library. Luettavissa: https://towardsdatascience.com/a-quick-introduction-to-the-pandas-python-library-f1b678f34673. Luettu: 28.7.2020.

Computer Hope. 2020. Regex. Luettavissa: https://www.computerhope.com/jargon/r/regex.htm Luettu 13.11.2020

Data Science Graduate Programs. 2020. Undestanding How Python is Used in Data Science. Luettavissa: https://www.datasciencegraduateprograms.com/python/ Luettu: 16.11.2020

electronicsnotes 5G NR New Radio: Luettavissa: https://www.electronics-notes.com/articles/connectivity/5g-mobile-wireless-cellular/5g-nr-new-radio.php. Luettu: 7.8.2020.

Farhad, M. 2019. Why Should We Use NumPy? Understanding NumPy Features Before Re-Inventing The Wheel. Luettavissa: https://medium.com/fintechexplained/why-should-we-use-numpy-c14a4fb03ee9 Luettu 13.11.2020

iPerf. iPerf – The ultimate speed test tool for TCP, UDP and SCTP. Luettavissa: https://iperf.fr/ Luettu: 14.10.2020

Harju, J. 2017. Ennakoiva analytiikka ja data science – Mistä on kyse? Luettavissa: https://www.valamis.com/fi/blogi/ennakoiva-analytiikka-ja-data-science-mista-on-kyse Luettu: 16.11.2020

Hunter, J. Dale, D. Firing, F. & Droettboom, M. 2013. Matplotlib. Release 1.3.0.rc1. Luettavissa: https://matplotlib.org/1.3.0/Matplotlib.pdf Luettu 16.11.2020

Mckinney, W. & Pandas Development Team. 2020. pandas: powerful Python data analysis toolkit. Release 1.1.0. Luettavissa: https://pandas.pydata.org/docs/pandas.pdf. Luettu: 30.7.2020.

Myers, T. 2016. Back to basics: The Shannon-Hartley Theorem. Luettavissa: https://www.ingenu.com/2016/07/back-to-basics-the-shannon-hartley-theorem/. Luettu: 4.9.2020

Python 2020. The Python Standard Library. Luettavissa: https://docs.python.org/3/library/. Luettu 7.8.2020.

Python. 2020. re – Regular expression operations. Luettavissa: https://docs.python.org/3/library/re.html Luettu: 13.11.2020

Qualcomm 2019. Snapdragon X55 5G Modem-RF System. Luettavissa: https://www.qualcomm.com/products/snapdragon-x55-5g-modem Luettu: 16.11.2020

Qualcomm 2020. QXDM Professional™ Tool Quick Start. Luettavissa: https://www.qualcomm.com/media/documents/files/qxdm-professional-qualcomm-extensible-diagnostic-monitor.pdf. Luettu: 7.8.2020.

Rodriguez, G. Pythex. Luettavissa: https://pythex.org/ Luettu: 16.11.2020

Sharetechnote. RF – SNR vs SINAD. Luettavissa: https://www.sharetechnote.com/html/RF_Handbook_SNR.html. Luettu: 7.8.2020.

Sharetechnote. 5G/NR – CSI RS. Luettavissa: https://www.sharetechnote.com/html/5G/5G_CSI_RS.html Luettu: 7.11.2020.

Sharetechnote. 5G/NR – SS Block. Luettavissa: https://www.sharetechnote.com/html/5G/5G_SS_Block.html. Luettu: 4.9.2020.

Thomson, C. 5 ways 5G will impact the future of surveying. Luettavissa: https://info.vercator.com/blog/ways-5g-will-impact-the-future-of-surveying. Luettu: 7.8.2020.

University of Manchester. 2006. CS3282: Digital Communications ’05-’06 Section 7: The Shannon-Hartley Theorem. Luettavissa: http://www.cs.man.ac.uk/~barry/mydocs/CS3282/Notes/DC06_7.pdf. Luettu: 5.9.2020

Liitteet

Liite 1. Versiotiedot

```
$ pip freeze
appdirs==1.4.4
attrs==19.3.0
black==19.10b0
click==7.1.2
cycler==0.10.0
kiwisolver==1.2.0
matplotlib==3.3.0
numpy==1.19.0
pandas==1.0.5
pathspec==0.8.0
Pillow==7.2.0
pyparsing==2.4.7
python-dateutil==2.8.1
pytz==2020.1
regex==2020.7.14
six==1.15.0
toml==0.10.1
typed-ast==1.4.1
(venv)
ntiittan at Ubuntu20 in ~/oppari
$ python --version
Python 3.8.5
```

Liite 2. Käytetyt kirjastot ja asetukset

```
"""Parser for QXDM measurements"""
import re
import sys
import argparse

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt


pd.options.mode.chained_assignment = None
np.set_printoptions(edgeitems=10)
```

Liite 3. Tietorakenteiden muodostamisfunktio

```
def form_dataframe():
    """Function for forming dataframe out of available data"""
    id_pat = r"\d+\D+\d+\s (\d+:\d+:\d+.\d+)\s (\[\w+])\s 0x\w+\s [redacted]"

    type_pat = r"(SSB Or TRS)                    = (\w+)"

    snr_pat = r"   [redacted]                      = (\d+\.\d+|NA)"

    id_regex = re.compile(id_pat)
    type_regex = re.compile(type_pat)
    snr_regex = re.compile(snr_pat)

    id_list = []
    timestamp_list = []
    ssb_trs_list = []
    snr_list = []
    antenna1_list = []
    antenna2_list = []
    antenna3_list = []
    antenna4_list = []

    measurements = open("ll1.txt").readlines()

    for line in measurements:
        id_match = id_regex.match(line)
        type_match = type_regex.match(line)
        snr_match = snr_regex.match(line)
        if id_match:
            timestamp_list.append(id_match.group(1))
            id_list.append(id_match.group(2))
        if type_match:
            ssb_trs_list.append(type_match.group(2))
        if snr_match:
            snr_list.append(snr_match.group(2))

    for antenna1, antenna2, antenna3, antenna4 in zip(*[iter(snr_list)] * 4):
        antenna1_list.append(antenna1)
        antenna2_list.append(antenna2)
        antenna3_list.append(antenna3)
        antenna4_list.append(antenna4)

    numpy_array = np.column_stack(
        (
            id_list,
            timestamp_list,
            ssb_trs_list,
            antenna1_list,
            antenna2_list,
            antenna3_list,
            antenna4_list,
        )
    )
    dataframe = pd.DataFrame(
        data=numpy_array,
        columns=[
            "id",
            "timestamp",
            "type",
            "antenna1",
            "antenna2",
            "antenna3",
            "antenna4",
        ],
    )
    return dataframe
```

Liite 4. Shannon-Hartley -kaava funktiona ja sen avulla laskeminen

```
def band_limited_calculation(snr):
    return 0.332 * 100 * snr


def parse_dataframe(dataframe):
    """Calculate channel capacity for given dataframe antennas"""
    id_list = []
    timestamps_list = []
    antenna1_list = []
    antenna2_list = []
    antenna3_list = []
    antenna4_list = []

    dataframe["antenna1"] = pd.to_numeric(
        dataframe["antenna1"], downcast="float"
    )
    dataframe["antenna2"] = pd.to_numeric(
        dataframe["antenna2"], downcast="float"
    )
    dataframe["antenna3"] = pd.to_numeric(
        dataframe["antenna3"], downcast="float", errors="coerce"
    )
    dataframe["antenna4"] = pd.to_numeric(
        dataframe["antenna4"], downcast="float", errors="coerce"
    )

    for index, row in dataframe.iterrows():
        id_value = row["id"]
        id_list.append(id_value)
        timestamp_value = row["timestamp"]
        timestamps_list.append(timestamp_value)
        antenna1_value = band_limited_calculation(row["antenna1"])
        antenna1_list.append(round(antenna1_value, 2))
        antenna2_value = band_limited_calculation(row["antenna2"])
        antenna2_list.append(round(antenna2_value, 2))
        antenna3_value = band_limited_calculation(row["antenna3"])
        antenna3_list.append(round(antenna3_value, 2))
        antenna4_value = band_limited_calculation(row["antenna4"])
        antenna4_list.append(round(antenna4_value, 2))

    np_arr = np.column_stack(
        (
            id_list,
            timestamps_list,
            antenna1_list,
            antenna2_list,
            antenna3_list,
            antenna4_list,
        )
    )
    type_dataframe = pd.DataFrame(
        data=np_arr,
        columns=[
            "id",
            "timestamp",
            "antenna1",
            "antenna2",
            "antenna3",
            "antenna4",
        ],
    )
    return type_dataframe
```

Liite 5. SSB- ja TRS laskufunktio osa 1

```
def calculate_for_all(dataframe):
    """Make calculations for both SSB and TRS"""
    all_dataframe = parse_dataframe(dataframe)
    all_dataframe["antenna1"] = pd.to_numeric(
        all_dataframe["antenna1"], downcast="float"
    )
    all_dataframe["antenna2"] = pd.to_numeric(
        all_dataframe["antenna2"], downcast="float"
    )
    all_dataframe["antenna3"] = pd.to_numeric(
        all_dataframe["antenna3"], downcast="float", errors="coerce"
    )
    all_dataframe["antenna4"] = pd.to_numeric(
        all_dataframe["antenna4"], downcast="float", errors="coerce"
    )

    antenna1_min = all_dataframe["antenna1"].min()
    antenna1_max = all_dataframe["antenna1"].max()
    antenna1_avg = all_dataframe["antenna1"].mean()

    antenna2_min = all_dataframe["antenna2"].min()
    antenna2_max = all_dataframe["antenna2"].max()
    antenna2_avg = all_dataframe["antenna2"].mean()

    antenna3_min = all_dataframe["antenna3"].min()
    antenna3_max = all_dataframe["antenna3"].max()
    antenna3_avg = all_dataframe["antenna3"].mean()

    antenna4_min = all_dataframe["antenna4"].min()
    antenna4_max = all_dataframe["antenna4"].max()
    antenna4_avg = all_dataframe["antenna4"].mean()

    min_avg = (antenna1_min + antenna2_min + antenna3_min + antenna4_min) / 4
    max_avg = (antenna1_max + antenna2_max + antenna3_max + antenna4_max) / 4
    avg_avg = (antenna1_avg + antenna2_avg + antenna3_avg + antenna4_avg) / 4

    theoretical_maximum = 1412.0

    antenna1_min_compare = antenna1_min / theoretical_maximum
    antenna1_min_percent = antenna1_min_compare * 100
    antenna1_max_compare = antenna1_max / theoretical_maximum
    antenna1_max_percent = antenna1_max_compare * 100
    antenna1_avg_compare = antenna1_avg / theoretical_maximum
    antenna1_avg_percent = antenna1_avg_compare * 100

    antenna2_min_compare = antenna2_min / theoretical_maximum
    antenna2_min_percent = antenna2_min_compare * 100
    antenna2_max_compare = antenna2_max / theoretical_maximum
    antenna2_max_percent = antenna2_max_compare * 100
    antenna2_avg_compare = antenna2_avg / theoretical_maximum
    antenna2_avg_percent = antenna2_avg_compare * 100

    antenna3_min_compare = antenna3_min / theoretical_maximum
    antenna3_min_percent = antenna3_min_compare * 100
    antenna3_max_compare = antenna3_max / theoretical_maximum
    antenna3_max_percent = antenna3_max_compare * 100
    antenna3_avg_compare = antenna3_avg / theoretical_maximum
    antenna3_avg_percent = antenna3_avg_compare * 100

    antenna4_min_compare = antenna4_min / theoretical_maximum
    antenna4_min_percent = antenna4_min_compare * 100
    antenna4_max_compare = antenna4_max / theoretical_maximum
    antenna4_max_percent = antenna4_max_compare * 100
    antenna4_avg_compare = antenna4_avg / theoretical_maximum
    antenna4_avg_percent = antenna4_avg_compare * 100

    all_min_compare = min_avg / theoretical_maximum
    all_min_percent = all_min_compare * 100
    all_max_compare = max_avg / theoretical_maximum
    all_max_percent = all_max_compare * 100
    all_avg_compare = avg_avg / theoretical_maximum
    all_avg_percent = all_avg_compare * 100
```

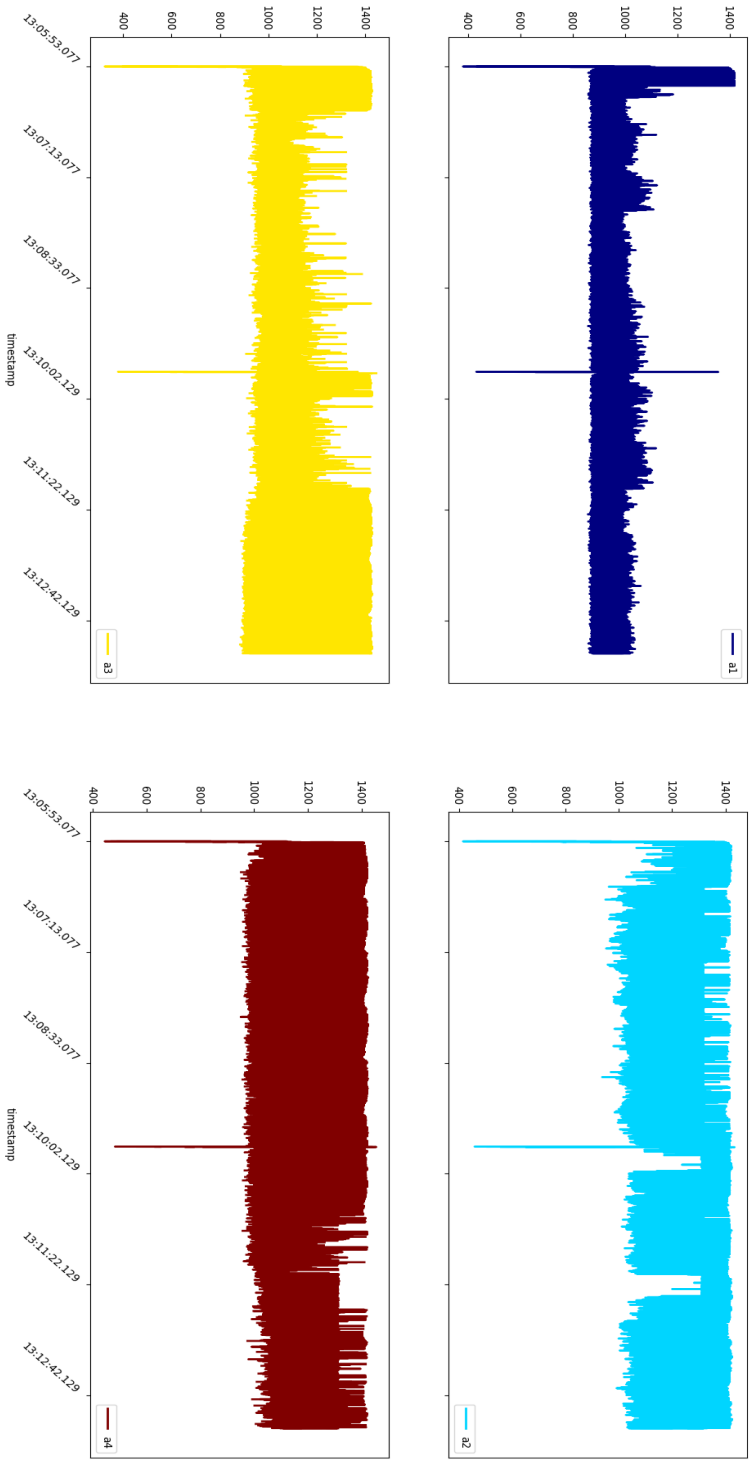
Liite 6. SSB- ja TRS laskufunktio osa 2

```
    compare_limit = theoretical_maximum * round(all_avg_compare, 2)
    print("-----------SSB&TRS----------")
    print("Min, Max & Avg for antenna 1")
    print(antenna1_min, antenna1_max, antenna1_avg)
    print("Min, Max & Avg for antenna 2")
    print(antenna2_min, antenna2_max, antenna2_avg)
    print("Min, Max & Avg for antenna 3")
    print(antenna3_min, antenna3_max, antenna3_avg)
    print("Min, Max & Avg for antenna 4")
    print(antenna4_min, antenna4_max, antenna4_avg)

    print("Average of Min, Max & Avg from all antennas")
    print(round(min_avg, 2), round(max_avg, 2), round(avg_avg, 2))

    print("Antenna 1 values compared to theoretical maximum")
    print(f"Min {round(antenna1_min_percent, 2)}%")
    print(f"Max {round(antenna1_max_percent, 2)}%")
    print(f"Avg {round(antenna1_avg_percent, 2)}%")

    print("Antenna 2 values compared to theoretical maximum")
    print(f"Min is {round(antenna2_min_percent, 2)}%")
    print(f"Max {round(antenna2_max_percent, 2)}%")
    print(f"Avg {round(antenna2_avg_percent, 2)}%")

    print("Antenna 3 values compared to theoretical maximum")
    print(f"Min {round(antenna3_min_percent, 2)}%")
    print(f"Max {round(antenna3_max_percent, 2)}%")
    print(f"Avg {round(antenna3_avg_percent, 2)}%")

    print("Antenna 4 values compared to theoretical maximum")
    print(f"Min {round(antenna4_min_percent, 2)}%")
    print(f"Max {round(antenna4_max_percent, 2)}%")
    print(f"Avg {round(antenna4_avg_percent, 2)}%")

    print("All antenna averages compared to theoretical maximum")
    print(f"Min {round(all_min_percent, 2)}%")
    print(f"Max {round(all_max_percent, 2)}%")
    print(f"Avg {round(all_avg_percent, 2)}%")

    print(f"DL limit according to average from all results, should be {round(compare_limit, 2)} Mbits/s")
    return dataframe
```

Liite 7. Ohjelman SSB- ja TRS tulokset

```
----------SSB&TRS----------
Min, Max & Avg for antenna 1
379.14 1415.98 972.9806
Min, Max & Avg for antenna 2
412.68 1430.92 1218.0106
Min, Max & Avg for antenna 3
321.38 1441.21 1146.5615
Min, Max & Avg for antenna 4
441.89 1451.17 1200.2258
Average of Min, Max & Avg from all antennas
388.77 1434.82 1134.44
Antenna 1 values compared to theoretical maximum
Min 26.85%
Max 100.28%
Avg 68.91%
Antenna 2 values compared to theoretical maximum
Min is 29.23%
Max 101.34%
Avg 86.26%
Antenna 3 values compared to theoretical maximum
Min 22.76%
Max 102.07%
Avg 81.2%
Antenna 4 values compared to theoretical maximum
Min 31.3%
Max 102.77%
Avg 85.0%
All antenna averages compared to theoretical maximum
Min 27.53%
Max 101.62%
Avg 80.34%
DL limit according to average from all results, should be 1129.6 Mbits/s
```

Liite 8. SSB- ja TRS visualisointi



Liite 9. TRS laskufunktio osa 1

```
def calculate_for_trs(dataframe):
    """Make calculations for TRS values"""
    dataframe_type = dataframe.groupby("type")
    trs_dataframe = dataframe_type.get_group("TRS")
    trs_dataframe = parse_dataframe(trs_dataframe)

    trs_dataframe["antenna1"] = pd.to_numeric(
        trs_dataframe["antenna1"], downcast="float"
    )
    trs_dataframe["antenna2"] = pd.to_numeric(
        trs_dataframe["antenna2"], downcast="float"
    )
    trs_dataframe["antenna3"] = pd.to_numeric(
        trs_dataframe["antenna3"], downcast="float", errors="coerce"
    )
    trs_dataframe["antenna4"] = pd.to_numeric(
        trs_dataframe["antenna4"], downcast="float", errors="coerce"
    )
    antenna1_min = trs_dataframe["antenna1"].min()
    antenna1_max = trs_dataframe["antenna1"].max()
    antenna1_avg = trs_dataframe["antenna1"].mean()

    antenna2_min = trs_dataframe["antenna2"].min()
    antenna2_max = trs_dataframe["antenna2"].max()
    antenna2_avg = trs_dataframe["antenna2"].mean()

    antenna3_min = trs_dataframe["antenna3"].min()
    antenna3_max = trs_dataframe["antenna3"].max()
    antenna3_avg = trs_dataframe["antenna3"].mean()

    antenna4_min = trs_dataframe["antenna4"].min()
    antenna4_max = trs_dataframe["antenna4"].max()
    antenna4_avg = trs_dataframe["antenna4"].mean()

    min_avg = (antenna1_min + antenna2_min + antenna3_min + antenna4_min) / 4
    max_avg = (antenna1_max + antenna2_max + antenna3_max + antenna4_max) / 4
    avg_avg = (antenna1_avg + antenna2_avg + antenna3_avg + antenna4_avg) / 4

    theoretical_maximum = 1412.0

    antenna1_min_compare = antenna1_min / theoretical_maximum
    antenna1_min_percent = antenna1_min_compare * 100
    antenna1_max_compare = antenna1_max / theoretical_maximum
    antenna1_max_percent = antenna1_max_compare * 100
    antenna1_avg_compare = antenna1_avg / theoretical_maximum
    antenna1_avg_percent = antenna1_avg_compare * 100

    antenna2_min_compare = antenna2_min / theoretical_maximum
    antenna2_min_percent = antenna2_min_compare * 100
    antenna2_max_compare = antenna2_max / theoretical_maximum
    antenna2_max_percent = antenna2_max_compare * 100
    antenna2_avg_compare = antenna2_avg / theoretical_maximum
    antenna2_avg_percent = antenna2_avg_compare * 100

    antenna3_min_compare = antenna3_min / theoretical_maximum
    antenna3_min_percent = antenna3_min_compare * 100
    antenna3_max_compare = antenna3_max / theoretical_maximum
    antenna3_max_percent = antenna3_max_compare * 100
    antenna3_avg_compare = antenna3_avg / theoretical_maximum
    antenna3_avg_percent = antenna3_avg_compare * 100

    antenna4_min_compare = antenna4_min / theoretical_maximum
    antenna4_min_percent = antenna4_min_compare * 100
    antenna4_max_compare = antenna4_max / theoretical_maximum
    antenna4_max_percent = antenna4_max_compare * 100
    antenna4_avg_compare = antenna4_avg / theoretical_maximum
    antenna4_avg_percent = antenna4_avg_compare * 100

    all_min_compare = min_avg / theoretical_maximum
    all_min_percent = all_min_compare * 100
    all_max_compare = max_avg / theoretical_maximum
    all_max_percent = all_max_compare * 100
    all_avg_compare = avg_avg / theoretical_maximum
    all_avg_percent = all_avg_compare * 100
```

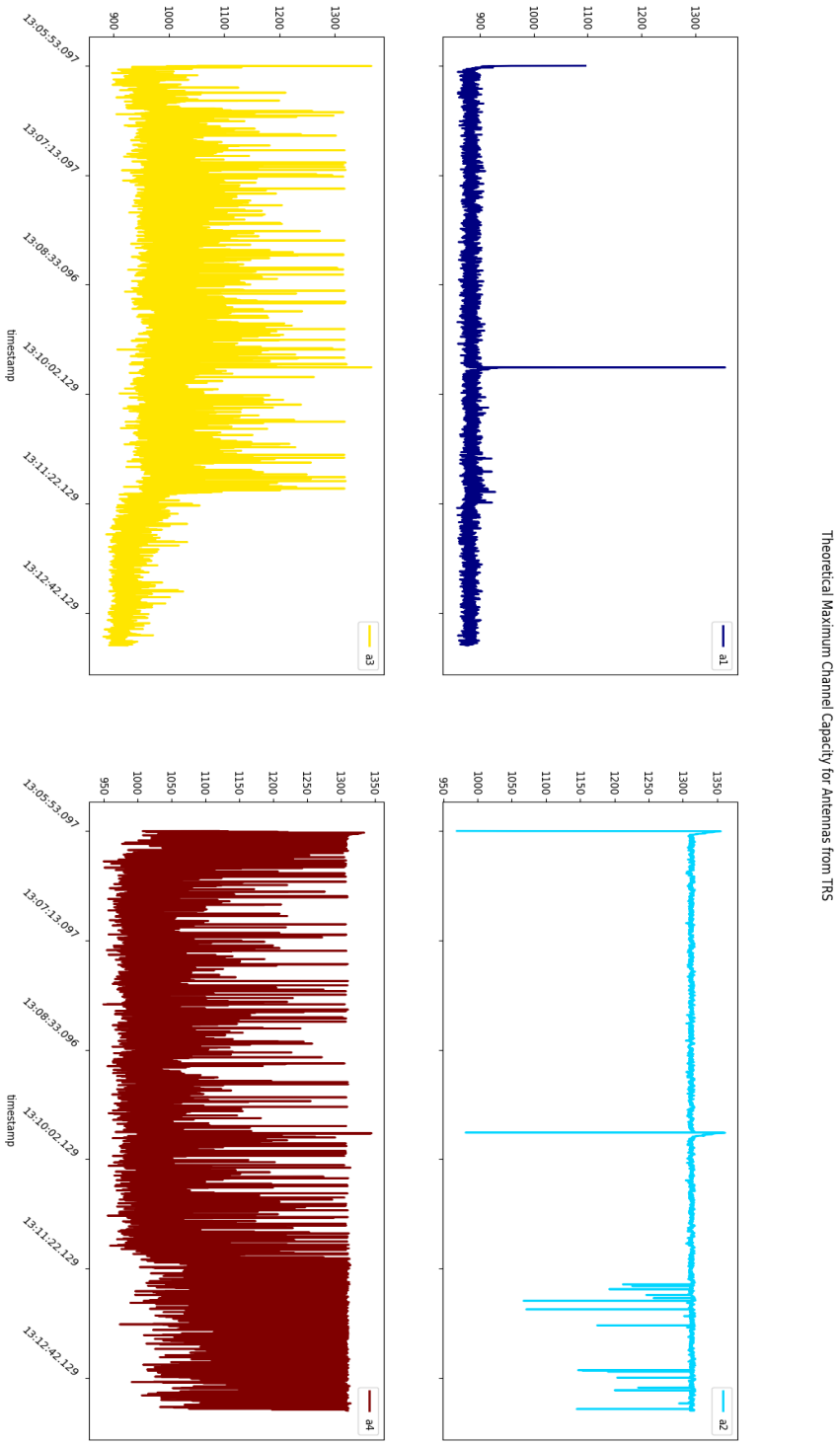
Liite 10. TRS laskufunktio osa 2

```
    compare_limit = theoretical_maximum * round(all_avg_compare, 2)
    print("-----------TRS----------")
    print("Min, Max & Avg for antenna 1")
    print(antenna1_min, antenna1_max, antenna1_avg)
    print("Min, Max & Avg for antenna 2")
    print(antenna2_min, antenna2_max, antenna2_avg)
    print("Min, Max & Avg for antenna 3")
    print(antenna3_min, antenna3_max, antenna3_avg)
    print("Min, Max & Avg for antenna 4")
    print(antenna4_min, antenna4_max, antenna4_avg)

    print("Average of Min, Max & Avg from all antennas")
    print(round(min_avg, 2), round(max_avg, 2), round(avg_avg, 2))

    print("Antenna 1 values compared to theoretical maximum")
    print(f"Min {round(antenna1_min_percent, 2)}%")
    print(f"Max {round(antenna1_max_percent, 2)}%")
    print(f"Avg {round(antenna1_avg_percent, 2)}%")

    print("Antenna 2 values compared to theoretical maximum")
    print(f"Min is {round(antenna2_min_percent, 2)}%")
    print(f"Max {round(antenna2_max_percent, 2)}%")
    print(f"Avg {round(antenna2_avg_percent, 2)}%")

    print("Antenna 3 values compared to theoretical maximum")
    print(f"Min {round(antenna3_min_percent, 2)}%")
    print(f"Max {round(antenna3_max_percent, 2)}%")
    print(f"Avg {round(antenna3_avg_percent, 2)}%")

    print("Antenna 4 values compared to theoretical maximum")
    print(f"Min {round(antenna4_min_percent, 2)}%")
    print(f"Max {round(antenna4_max_percent, 2)}%")
    print(f"Avg {round(antenna4_avg_percent, 2)}%")

    print("All antenna averages compared to theoretical maximum")
    print(f"Min {round(all_min_percent, 2)}%")
    print(f"Max {round(all_max_percent, 2)}%")
    print(f"Avg {round(all_avg_percent, 2)}%")

    print(f"DL limit according to average from TRS, should be {round(compare_limit, 2)} Mbits/s")

    return trs_dataframe
```

Liite 11. Ohjelman TRS-tulokset

```
----------TRS----------
Min, Max & Avg for antenna 1
856.23 1353.56 881.2189
Min, Max & Avg for antenna 2
969.11 1360.54 1311.1647
Min, Max & Avg for antenna 3
880.13 1364.52 986.1332
Min, Max & Avg for antenna 4
948.52 1343.94 1089.9062
Average of Min, Max & Avg from all antennas
913.5 1355.64 1067.11
Antenna 1 values compared to theoretical maximum
Min 60.64%
Max 95.86%
Avg 62.41%
Antenna 2 values compared to theoretical maximum
Min is 68.63%
Max 96.36%
Avg 92.86%
Antenna 3 values compared to theoretical maximum
Min 62.33%
Max 96.64%
Avg 69.84%
Antenna 4 values compared to theoretical maximum
Min 67.18%
Max 95.18%
Avg 77.19%
All antenna averages compared to theoretical maximum
Min 64.7%
Max 96.01%
Avg 75.57%
DL limit according to average from TRS, should be 1073.12 Mbits/s
```

Liite 12. TRS-visualisointi



Liite 13. SSB laskufunktio osa 1

```
def calculate_for_ssb(dataframe):
    """Make calculations for SSB values"""
    dataframe_type = dataframe.groupby("type")
    ssb_dataframe = dataframe_type.get_group("SSB")
    ssb_dataframe = parse_dataframe(ssb_dataframe)

    ssb_dataframe["antenna1"] = pd.to_numeric(
        ssb_dataframe["antenna1"], downcast="float"
    )
    ssb_dataframe["antenna2"] = pd.to_numeric(
        ssb_dataframe["antenna2"], downcast="float"
    )
    ssb_dataframe["antenna3"] = pd.to_numeric(
        ssb_dataframe["antenna3"], downcast="float", errors="coerce"
    )
    ssb_dataframe["antenna4"] = pd.to_numeric(
        ssb_dataframe["antenna4"], downcast="float", errors="coerce"
    )
    antenna1_min = ssb_dataframe["antenna1"].min()
    antenna1_max = ssb_dataframe["antenna1"].max()
    antenna1_avg = ssb_dataframe["antenna1"].mean()

    antenna2_min = ssb_dataframe["antenna2"].min()
    antenna2_max = ssb_dataframe["antenna2"].max()
    antenna2_avg = ssb_dataframe["antenna2"].mean()

    antenna3_min = ssb_dataframe["antenna3"].min()
    antenna3_max = ssb_dataframe["antenna3"].max()
    antenna3_avg = ssb_dataframe["antenna3"].mean()

    antenna4_min = ssb_dataframe["antenna4"].min()
    antenna4_max = ssb_dataframe["antenna4"].max()
    antenna4_avg = ssb_dataframe["antenna4"].mean()

    min_avg = (antenna1_min + antenna2_min + antenna3_min + antenna4_min) / 4
    max_avg = (antenna1_max + antenna2_max + antenna3_max + antenna4_max) / 4
    avg_avg = (antenna1_avg + antenna2_avg + antenna3_avg + antenna4_avg) / 4

    theoretical_maximum = 1412.0

    antenna1_min_compare = antenna1_min / theoretical_maximum
    antenna1_min_percent = antenna1_min_compare * 100
    antenna1_max_compare = antenna1_max / theoretical_maximum
    antenna1_max_percent = antenna1_max_compare * 100
    antenna1_avg_compare = antenna1_avg / theoretical_maximum
    antenna1_avg_percent = antenna1_avg_compare * 100

    antenna2_min_compare = antenna2_min / theoretical_maximum
    antenna2_min_percent = antenna2_min_compare * 100
    antenna2_max_compare = antenna2_max / theoretical_maximum
    antenna2_max_percent = antenna2_max_compare * 100
    antenna2_avg_compare = antenna2_avg / theoretical_maximum
    antenna2_avg_percent = antenna2_avg_compare * 100

    antenna3_min_compare = antenna3_min / theoretical_maximum
    antenna3_min_percent = antenna3_min_compare * 100
    antenna3_max_compare = antenna3_max / theoretical_maximum
    antenna3_max_percent = antenna3_max_compare * 100
    antenna3_avg_compare = antenna3_avg / theoretical_maximum
    antenna3_avg_percent = antenna3_avg_compare * 100

    antenna4_min_compare = antenna4_min / theoretical_maximum
    antenna4_min_percent = antenna4_min_compare * 100
    antenna4_max_compare = antenna4_max / theoretical_maximum
    antenna4_max_percent = antenna4_max_compare * 100
    antenna4_avg_compare = antenna4_avg / theoretical_maximum
    antenna4_avg_percent = antenna4_avg_compare * 100

    all_min_compare = min_avg / theoretical_maximum
    all_min_percent = all_min_compare * 100
    all_max_compare = max_avg / theoretical_maximum
    all_max_percent = all_max_compare * 100
    all_avg_compare = avg_avg / theoretical_maximum
    all_avg_percent = all_avg_compare * 100
```

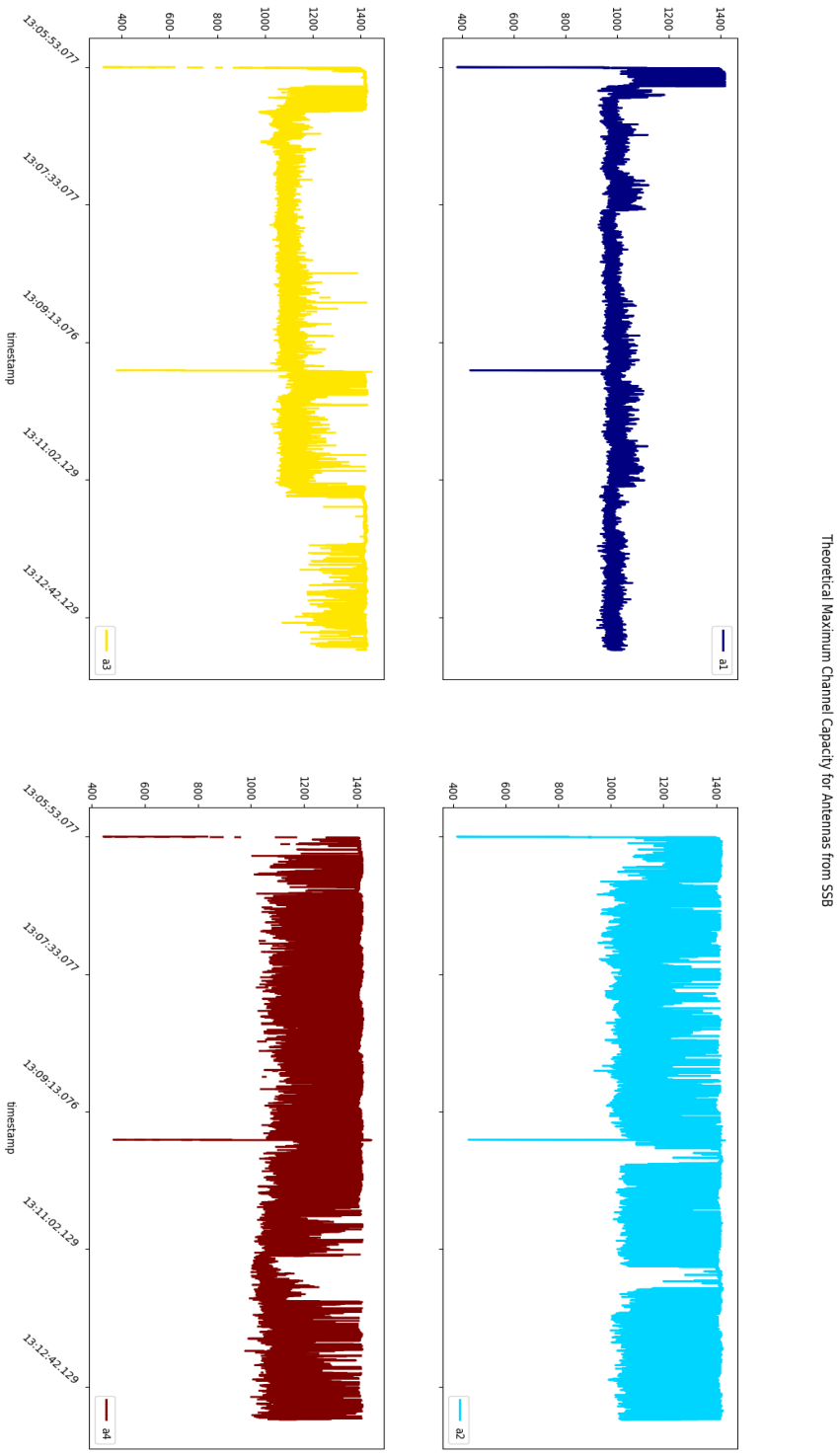
Liite 14. SSB laskufunktio osa 2

```
    compare_limit = theoretical_maximum * round(all_avg_compare, 2)
    print("-----------SSB----------")
    print("Min, Max & Avg for antenna 1")
    print(antenna1_min, antenna1_max, antenna1_avg)
    print("Min, Max & Avg for antenna 2")
    print(antenna2_min, antenna2_max, antenna2_avg)
    print("Min, Max & Avg for antenna 3")
    print(antenna3_min, antenna3_max, antenna3_avg)
    print("Min, Max & Avg for antenna 4")
    print(antenna4_min, antenna4_max, antenna4_avg)

    print("Average of Min, Max & Avg from all antennas")
    print(round(min_avg, 2), round(max_avg, 2), round(avg_avg, 2))

    print("Antenna 1 values compared to theoretical maximum")
    print(f"Min {round(antenna1_min_percent, 2)}%")
    print(f"Max {round(antenna1_max_percent, 2)}%")
    print(f"Avg {round(antenna1_avg_percent, 2)}%")

    print("Antenna 2 values compared to theoretical maximum")
    print(f"Min is {round(antenna2_min_percent, 2)}%")
    print(f"Max {round(antenna2_max_percent, 2)}%")
    print(f"Avg {round(antenna2_avg_percent, 2)}%")

    print("Antenna 3 values compared to theoretical maximum")
    print(f"Min {round(antenna3_min_percent, 2)}%")
    print(f"Max {round(antenna3_max_percent, 2)}%")
    print(f"Avg {round(antenna3_avg_percent, 2)}%")

    print("Antenna 4 values compared to theoretical maximum")
    print(f"Min {round(antenna4_min_percent, 2)}%")
    print(f"Max {round(antenna4_max_percent, 2)}%")
    print(f"Avg {round(antenna4_avg_percent, 2)}%")

    print("All antenna averages compared to theoretical maximum")
    print(f"Min {round(all_min_percent, 2)}%")
    print(f"Max {round(all_max_percent, 2)}%")
    print(f"Avg {round(all_avg_percent, 2)}%")

    print(f"DL limit according to average from SSB, should be {round(compare_limit, 2)} Mbits/s")

    return ssb_dataframe
```

Liite 15. Ohjelman SSB tulokset

```
----------SSB----------
Min, Max & Avg for antenna 1
379.14 1415.98 995.9232
Min, Max & Avg for antenna 2
412.68 1430.92 1194.72
Min, Max & Avg for antenna 3
321.38 1441.21 1201.2589
Min, Max & Avg for antenna 4
441.89 1451.17 1237.8387
Average of Min, Max & Avg from all antennas
388.77 1434.82 1157.44
Antenna 1 values compared to theoretical maximum
Min 26.85%
Max 100.28%
Avg 70.53%
Antenna 2 values compared to theoretical maximum
Min is 29.23%
Max 101.34%
Avg 84.61%
Antenna 3 values compared to theoretical maximum
Min 22.76%
Max 102.07%
Avg 85.07%
Antenna 4 values compared to theoretical maximum
Min 31.3%
Max 102.77%
Avg 87.67%
All antenna averages compared to theoretical maximum
Min 27.53%
Max 101.62%
Avg 81.97%
DL limit according to average from SSB, should be 1157.84 Mbits/s
```

Liite 16. SSB visualisointi



Liite 17. Visualisointi- ja main-funktiot

```
def plot_all(all_dataframe):
    axes = all_dataframe.plot(
        x="timestamp",
        kind="line",
        rot=45,
        lw=2,
        colormap="jet",
        figsize=(60, 15),
        title="Theoretical Maximum Channel Capacity for Antennas from SSB and TRS",
        subplots=True,
        layout=(2, 2),
    )
    plt.show()


def plot_trs(trs_dataframe):
    axes = trs_dataframe.plot(
        x="timestamp",
        kind="line",
        rot=45,
        lw=2,
        colormap="jet",
        figsize=(60, 15),
        title="Theoretical Maximum Channel Capacity for Antennas from TRS",
        subplots=True,
        layout=(2, 2),
    )
    plt.show()


def plot_ssb(ssb_dataframe):
    axes = ssb_dataframe.plot(
        x="timestamp",
        kind="line",
        rot=45,
        lw=2,
        colormap="jet",
        figsize=(60, 15),
        title="Theoretical Maximum Channel Capacity for Antennas from SSB",
        subplots=True,
        layout=(2, 2),
    )
    plt.show()


def main():
    """Main function for running other functions"""
    initial_dataframe = form_dataframe()
    all_dataframe = calculate_for_all(initial_dataframe)
    trs_dataframe = calculate_for_trs(initial_dataframe)
    ssb_dataframe = calculate_for_ssb(initial_dataframe)

    plot_all(all_dataframe)
    plot_trs(trs_dataframe)
    plot_ssb(ssb_dataframe)


if __name__ == "__main__":
    main()
```