Oskar Schatz

Droonin kumppanitietokoneen ohjelmiston integrointi ja simulointiympäristön kehitys

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikka

Insinöörityö

30.4.2025

Tiivistelmä

Tekijä:	Oskar Schatz
Otsikko:	Droonin kumppanitietokoneen ohjelmiston integrointi ja simulointiympäristön kehitys
Sivumäärä:	31 sivua
Aika:	30.4.2025
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Tieto- ja viestintätekniikka
Ammatillinen pääaine:	Ohjelmistotuotanto
Ohjaajat:	FM Simo Silander Projektipäällikkö Timo Kasurinen

Tämän insinöörityön tarkoituksena oli Kaakkois-Suomen ammattikorkeakoulun "Droonit ja data-analytiikka sähköverkkojen vianpaikannuksessa ja kunnossapidossa" -hankkeessa käytetyn droonin kumppanitietokoneen integrointi ja käyttöönotto. Insinöörityön toinen tavoite oli kehittää droonia vastaava simulointiympäristö.

Työn aikana alustettiin droonin kumppanitietokone ja siihen asennettiin ja konfiguroitiin tarvittavat ohjelmistot droonissa olevien antureiden käyttöönottoa varten. Työn aikana toteutettiin simulaatioympäristö käyttämällä virtuaalitietokoneita.

Työn tuloksena tehtiin lentoja, joissa droonin anturit olivat käytössä.

Avainsanat: drooni, kumppanitietokone, simulaatio

Abstract

Author:	Oskar Schatz
Title:	Integration of Drone Companion Computer and Developing Simulation Environment
Number of Pages:	31 pages
Date:	30 April 2025
Degree:	Bachelor of Engineering
Degree Programme:	Information and Communications Technologies
Professional Major:	Software Engineering
Supervisors:	Simo Silander, M.Sc. Timo Kasurinen, Project Manager

---

The purpose of the project introduced in this thesis was the integration and deployment of a companion computer used in a drone for the “Drones and data analytics in fault detection and maintenance of power grids” project at the South-Eastern University of Applied Sciences. The second objective was to develop a corresponding simulation environment for the drone.

In the project, the companion computer of the drone was initialized, and the necessary software was installed and configured to enable the use of the sensors on the drone. A simulation environment was also implemented using virtual machines.

Test flights were also conducted with the drone sensors in operation.

Keywords: Drone, Companion computer, Simulation

Sisällys

Lyhenteet

3D: *3-Dimensional*. Kolmiulotteinen.

AMSL: *Above Mean Sea Level*. Paikan korkeus merenpinnasta.

APT: *Advanced Package Tool*. Debian-pohjaisten Linux-käyttöjärjestelmien käyttämä paketinhallintaohjelmisto.

Bash: *Bourne Again Shell*. GNU-projektin komentorivitulkki.

CSV: *Comma-separated values*. Teksitiedosto, jolla kuvataan taulukkotietoa. Taulukon kentät on eroteltu tekstitiedostossa pilkuilla.

DHCP: *Dynamic Host Configuration Protocol*. Uusille lähiverkkoon kytkeytyville laitteille IP-osoitteita antava verkkoprotokolla.

DT: *Devicetree*. Tietorakenne, joka kuvaa tietokoneen laitteistoa.

DTB: *Devicetree blob*. Binäärimuotoinen Devicetree.

DTS: *Devicetree source*. Devicetree selkokielisenä lähdekoodina.

eMMC: *embedded MultiMediaCard*. Sulautettu flash-muisti.

FAST-LIO: *Fast LiDAR-Inertial Odometry*. Algoritmi, jolla voi tuottaa pistepilvitietoa LiDAR-anturin ja IMU-anturin avulla.

FCU: *Flight Control Unit*. Drooneja ohjaava laitteisto, joka voi sisältää autopilottijärjestelmän.

GCS: *Ground Control Station*. Maa-asema, jolla voi ohjata drooneja.

IMU: *Inertial Measurement Unit*. Elektroninen laite, joka mittaa ja ilmoittaa kappaleen ominaiskiihtyvyyden, kulmanopeuden ja joskus myös sen asennon käyttämällä yhdistelmää kiihtyvyysantureista, gyroskoopeista ja joskus magnetometreistä.

LiDAR: *Light detection and ranging*. Valotutka, joka mittaa kohteen etäisyyden lähettämällä laservalopulssin ja mittaamalla ajan heijastuksen palautumiseen.

MAVLINK: *Micro Air Vehicle Link*. Drooneissa käytetty ohjausprotokolla.

MAVROS: *Micro Air Vehicle ROS*. ROS-laajennus, joka mahdollistaa MAVLINK-protokollan käytön.

PCL: *Point Cloud Library*. Avoimen lähdekoodin kirjasto algoritmeista pistepilvitiedon käsittelyä varten.

PCD: *Point Cloud Data*. Tietomuoto pistepilvitiedon tallentamista varten.

ROS: *Robot Operating System*. Robotiikan kehitystä varten luotu avoimen lähdekoodin kokoelma työkaluja, kirjastoja ja käytäntöjä.

SD: *Secure Digital*. SD Associationin kehittämä muistikortti.

SDF: *Simulation Description Format.* XML-muotoinen tietomalli, joka kuvaa esineitä ja ympäristöjä robotiikkasimulaatioissa.

SITL: *Software in the loop*. Ohjelmisto, joka mahdollistaa ArduPilot-ohjelmiston ajamisen ilman erityistä laitteistoa.

UART: *Universal asynchronous receiver-transmitter*. Asynkronista sarjaliikennettä tukeva laite.

USB: *Universal serial bus*. Sarjaväyläarkkitehtuuri oheislaitteiden tietokoneisiin liittämistä varten.

WGS84: *World Geodetic System 1984*. Yhdysvaltain puolustusministeriön määrittelemä ja ylläpitämä tasokoordinaattijärjestelmä.

XML: *Extensible Markup Language*. Merkintäkielen standardi, joka määrittää tietojen merkintämuodon loogisella rakenteella.

YAML: *YAML Ain't Markup Language*. Konfiguraatiotiedostoissa usein käytetty merkintäkieli.

1 Johdanto

Tämä lopputyö on syntynyt Kaakkois-Suomen ammattikorkeakoulun XAMK:in tuottaman hankkeen puitteissa. "Droonit ja data-analytiikka sähköverkkojen vianpaikannuksessa ja kunnossapidossa" -hankkeen yhtenä tavoitteena oli kehittää kustannustehokas mukautettu drooni matala- ja keskisuurijännitteisten voimalinjojen kunnon kartoittamista varten.

Droonien kehitystyössä simuloitujen ympäristöjen käyttö on keskeisen tärkeää. Virtuaalinen testaus parantaa turvallisuutta, koska riskialttiita tehtäviä ja operaatioita voidaan kokeilla ilman fyysistä vaaraa. Samalla kehityskustannukset pienenevät, kun kalliita korjauksia ja laitehankintoja ei tarvita epäonnistuneiden kokeilujen seurauksena.

Simulaatio nopeuttaa kehitystä, sillä uusia algoritmeja ja sensoriasetelmia voi testata ilman fyysisten prototyyppien rakentamista. Lisäksi simulaatio mahdollistaa olosuhteiden tarkkaa toistamista, mikä helpottaa virheiden löytämistä.

Kaiken kaikkiaan simuloidut ympäristöt tarjoavat turvallisen, kustannustehokkaan ja nopean tavan kehittää, testata ja parantaa droonijärjestelmiä ennen niiden käyttöönottoa todellisessa maailmassa.

Tämän lopputyön tavoitteena on integroida hankkeessa käytetyn droonin kumppanitietokoneeseen tarvittavat ohjelmistot, jotta droonissa olevien antureiden havaintoja voi ohjelmallisesti käsitellä ja jotta droonin autopilottia pystyy ohjelmallisesti ohjaamaan kumppanitietokoneen avulla. Droonin laitteisto oli jo tiedossa ennen lopputyön aloittamista.

Työn toinen tavoite on luoda droonin ohjelmistoa vastaava simuloitu ympäristö. Simuloidun ympäristön tulee vastata oikean droonin ohjelmistoarkkitehtuuria niin läheisesti kuin mahdollista, jotta simuloidussa ympäristössä luotu ohjelmisto on mahdollisimman helppo ottaa käyttöön oikeassa laitteistossa.

2 Yleistä

2.1 Droonien historiaa

Miehittämättömiä ilma-aluksia alettiin kehittää pian ensimmäisten miehitettyjen moottoroitujen lentokoneiden jälkeen. Miehittämättömän ilma-aluksen keksijänä pidetään Archibald Montgomery Low'ta (1888-1956), joka rakensi ja testasi ensimmäisen maailmansodan aikana Britanniassa pienen radio-ohjattavan lentokoneen nimeltä Aerial Target [1].

Sotien välisenä aikana kehitys ja testaus jatkui, ja Britanniassa ja USA:ssa valmistettiin useita radio-ohjattavia lentokoneita koulutuskäyttöön maalitauluiksi. Termi drone lienee syntynyt silloin, erään brittimallin inspiroimana, jolle oli annettu nimeksi DH.82B Queen Bee.

Toisen maailmansodan aikana kehittyneimmät miehittämättömät ilma-alukset olivat Natsi-Saksan 1940-luvulla kehittämät V1- ja V2-ohjukset, joilla pommitettiin Lontoota. V-raketit olivat maailman ensimmäiset risteilyohjukset. Niitä ei kauko-ohjattu, vaan niillä oli alkeellinen, epätarkka ja epäluotettava autonominen ohjausjärjestelmä. [2.] Vuonna 1946 amerikkalaiset muunsivat B-17 Flying Fortress -pommikoneita drooneiksi keräämään Bikinin atollilla näytteitä radioaktiivisesta pilvestä [3].

Tiedusteludrooneja otettiin ensimmäisen kerran laajamittaiseen käyttöön Vietnamin sodassa. Drooneille kehitettiin myös monia uusia tehtäviä, kuten toimiminen harhautuksina taistelussa, ohjusten laukaiseminen kiinteisiin kohteisiin ja lentolehtisten pudottaminen psykologisia propagandaoperaatioita varten. [4.]

Droonien korostunut sotilaallinen merkitys on tullut selväksi viimeistään Venäjän aloittamassa sodassa Ukrainaa vastaan, ja tällä hetkellä useat valtiot ovat satsaamassa massiivisesti erilaisiin sotilaallisiin tarkoituksiin kehitettyihin miehittämättömiin ilma-aluksiin [5].

2.2 Droonien hyödyt verrattuna perinteisiin ilma-aluksiin

Miehittämättömiä ilma-aluksia kehitettiin alun perin pääasiassa sotilaskäyttöön, mutta 2000-luvulla teknologian kehittyminen antoi niille huomattavasti laajemman roolin monilla ilmailun osa-alueilla ja myös siviilikäyttössä.

Nykyisin drooneja hyödynnetään etsintä- ja pelastustoiminnassa, poliisivalvonnassa, liikenteen ja sään seurannassa, palontorjunnassa, ilmavalo- ja videokuvauksessa, niin kutsutussa täsmäviljelyssä, arkeologiassa sekä erilaisissa kuljetuspalveluissa. Droonit ovat tärkeitä myös infrastruktuurin tarkastuksessa, rakennusten kuntokartoituksessa, ympäristön tilan ja villieläinten seurannassa sekä hätäavussa ja katastrofien hallinnassa.

Droonit mahdollistivat tehtäviä, joissa miehitettyjen ilma-alusten käyttö olisi mahdotonta tai liian riskialtista, ja niitä voi operoida paikoissa, joihin ihminen ei pääse - jopa avaruudessa. NASAn Ingenuity-helikopteri laskeutui Marsin pinnalle 2021 ja suoritti 72 lentoa planeetan ohuessa ilmakehässä. [6.]

Drooneja voidaan käyttää nopeasti ilman suurta infrastruktuuria kuten lentokenttiä - joitain malleja voi jopa heittää ilmaan käsin tai lähettää liikkuvilta alustoilta. Droonit ovat yleensä halvempia rakentaa, käyttää ja huoltaa kuin perinteiset lentokoneet. Ne eivät tarvitse elossapitojärjestelmiä eikä ohjaamoita, ja niillä on miehitettyihin ilma-aluksiin verrattuna hyvin pitkä lentoaika: 1998 AAI Aerosonde -merkkinen UAV nimeltä Laima ylitti ensimmäisenä miehittämättömänä ilma-aluksena Atlantin valtameren. Lentoon kului 26 tuntia. [7.]

Miehittämättömiin ilma-aluksiin kuuluvat sekä autonomiset (ilman ihmisen ohjausta, eli ohjelmallisesti toimivat) droonit että kauko-ohjattavat lentolaitteet (RPV, remotely piloted vehicle). Alkuvaiheessa miehittämättömät ilma-alukset olivat vielä kauko-ohjattavia, nykyään drooni määritellään yleensä "miehittämättömäksi ilma-alukseksi tai alukseksi, jota ohjataan kauko-ohjauksella tai aluksen omilla tietokoneilla”. [8.]

Droonien tietokoneilla toimiva autopilottiohjelmisto ohjaa droonin liikkeitä automaattisesti ilman jatkuvaa manuaalista ohjausta. Saatavilla on sekä kaupallisia että avoimen lähdekoodin ohjelmistoja. Autopilotti huolehtii muun muassa stabiloinnista ja navigoinnista liikuttaen droonia reitillä GPS:n, kompassin ja korkeussensorien avulla. Lisäksi autopilotti sisältää erilaisia vikaturvatoimintoja (engl. failsafe). Vikaturvatoimintojen ansiosta esimerkiksi ohjausyhteyden katketessa tai akun varauksen laskiessa kriittisen alhaiseksi, drone voi palata automaattisesti lähtöpisteeseensä.

2.3 Droonien käyttö voimalinjojen tarkistuksessa

Droonien käyttö voimalinjojen tarkistuksessa tarjoaa merkittäviä hyötyjä. Droonit voivat lentää lähelle johtoja ja pylväitä ilman vaaraa henkilöstölle ja niihin voidaan asentaa erilaisia antureita, kuten korkearesoluutioisia kameroita, LiDAR (Light detection and ranging) -antureita tai lämpökameroita, joiden avulla voi arvioida voimalinjojen kuntoa tehokkaasti. Droonien käyttö sähköverkkojen tarkistuksessa on kasvanut viime vuosina merkittävästi [9]. Droonit voidaan lisäksi ohjelmoida suorittamaan automaattisia tarkastuslentoja ennalta määritetyn reitin mukaisesti, jolloin tarkistukset ovat toistettavia ja vertailukelpoisia ajallisesti.

Perinteisesti voimalinjojen tarkastuksia on tehty helikoptereilla. Helikoptereilla suoritettava voimalinjojen tarkistus mahdollistaa laajojen alueiden tarkastuksen nopeasti, mutta helikoptereiden käyttöön liittyy merkittäviä kustannuksia, ympäristövaikutuksia ja turvallisuusriskejä. Droonien käyttö helikoptereiden sijasta lisää voimalinjojen tarkastuksessa turvallisuutta, kustannustehokkuutta ja tehokkuutta. [10.]

2.4 ArduPilot

ArduPilot on yksi yleisimmin käytetyistä avoimen lähdekoodin autopilottijärjestelmistä. Sitä käytetään muun muassa miehittämättömien aluksien, kuten

multikoptereiden, kiinteäsiipisten alusten, maa-alusten, veneiden tai sukellusveneiden autononomiseen ohjaukseen.

Vuonna 2009 yksi ArduPilotin lähdekoodin ensimmäisistä versioista julkaistiin Google Code palveluun [11]. Vuosien varrella ArduPilot on kehittynyt laajaksi kokonaisuudeksi ohjelmistoja ja työkaluja, joka pyrkii täyttämään niin harrastelijoiden kuin kaupallistenkin toimijoiden tarpeet kattavilla ominaisuuksillaan [12].

ArduPilot on vain ohjelmisto, jota voidaan ajaa eri laitteistoilla. ArduPilot tukee suurta määrää erilaisia alustoja, joilla sitä voi käyttää, kuten esimerkiksi mikrokontrollereilla tai autopilotiksi suunnitelluilla piirilevyillä. [13.] ArduPilot käyttää MAVLink (Micro Air Vehicle Link) protokollaa muiden GCS:n (Ground Control Station) tai kumppanitietokoneiden kanssa kommunikointia varten [14].

2.5 SITL

SITL (Software in the loop) mahdollistaa ArduPilot ohjelmiston ajamisen tietokoneella ilman erityistä laitteistoa. ArduPilot SITL -ympäristö on kehitetty toimimaan natiivisti sekä Linux- että Windows-käyttöjärjestelmissä. [15.]

2.6 Robot Operating System

ROS (Robot Operating System) on robotiikan kehitystä varten luotu kokoelma työkaluja, kirjastoja ja käytäntöjä. Sen yksi tavoitteista on yksinkertaistaa monimutkaistenkin robotiikan ohjelmistojen kehittämistä. [16.]

ROS käyttää aihepohjaista (engl. topic) viestintämallia, jossa tilaajasolmut (engl. subscriber) voivat tilata aiheita ja julkaisijasolmut (engl. Publisher) voivat julkaista viestejä aiheisiin, joista ne välittyvät tiettyjen aiheiden tilaajille. Tämä mahdollistaa solmujen välisen asynkronisen viestinnän.

Solmut voivat myös tarjota palveluita (engl. service), joita toiset solmut voivat kutsua. Tämä toiminnallisuus mahdollistaa synkronisen viestinnän solmujen

välillä. [17.] Solmut ovat käytännössä käännettyjä ohjelmistoja, joita voidaan ajaa [18].

ROS:n avulla kehitettyjen ohjelmistojen ytimessä on ROS master. Se on keskitetty palvelu, joka pitää kirjaa ohjelmistossa olevien julkaisija- ja tilaajasolmujen yhteyksistä aiheisiin ja hallistee kommunikaation näiden välillä. ROS master seuraa myös solmujen tarjoamia palveluita. [19.] ROS master vastaa myös solmujen rekisteröinnistä ja niiden nimien hallinasta.

2.7 MAVROS

MAVROS (Micro Air Vehicle ROS) on ROS-laajennus, joka kääntää ROS-aiheissa olevan viestinnän MAVLINK-muotoon ja voi välittää sen MAVLINK-yhteensopiville autopiloteille tai GCS:lle, kuten esimerkiksi ArduPilotia käyttäville autopiloteille. MAVROS:n kääntää myös FCU:sta (Flight Control Unit) tulevan MAVLINK-liikenteen viesteiksi ROS-aiheisiin. [20.] Esimerkkikoodi 1 näyttää ROS-aiheita, joihin MAVROS kääntää MAVLINK-liikennettä.

```
/mavros/home_position/home
/mavros/home_position/set
/mavros/imu/data
/mavros/imu/data_raw
/mavros/imu/diff_pressure
/mavros/imu/mag
/mavros/imu/static_pressure
/mavros/imu/temperature_baro
/mavros/imu/temperature_imu
/mavros/local_position/accel
/mavros/local_position/odom
/mavros/local_position/pose
/mavros/local_position/pose_cov
/mavros/local_position/velocity_body
/mavros/local_position/velocity_body_cov
/mavros/local_position/velocity_local
/mavros/manual_control/control
/mavros/manual_control/send
/mavros/mission/reached
/mavros/mission/waypoints
```

Esimerkkikoodi 1. MAVROS ROS -laajennuksen julkaisemia ROS-aiheita.

2.8 Gazebo

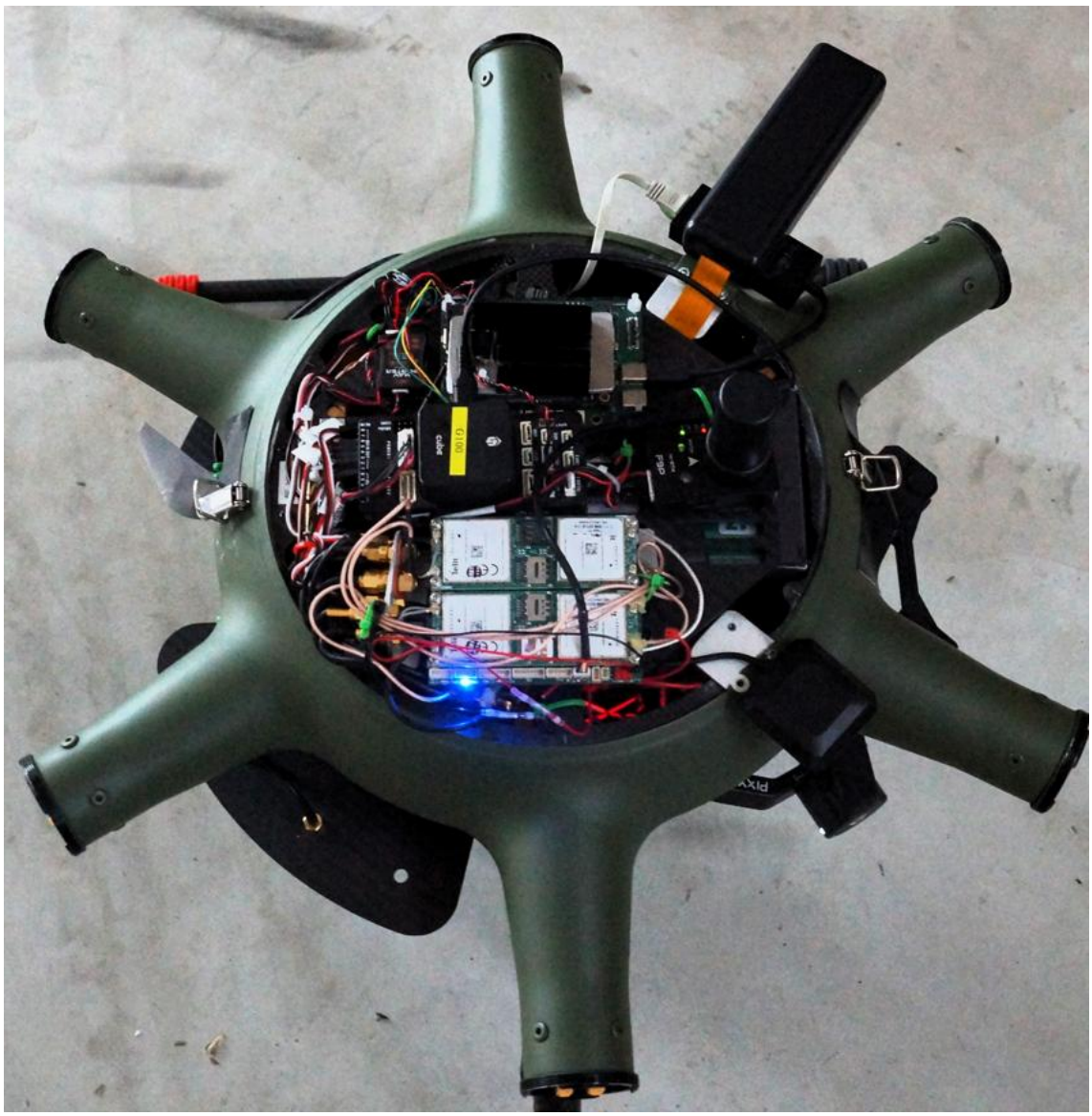
Gazebo on avoimen lähdekoodin 3D (3-Dimensional) -robotiikkasimulaattori, joka mahdollistaa tarkan oikeaa maailmaa mallintavan fysikaalisen mallinnuksen. Mallinnukseen kuuluu muun muassa väännön, kitkan, painovoiman tai minkä vaan muun oikean maailman ilmiön mallintaminen.

Gazebon avulla voi simuloida käytännössä minkä vain tyyppisiä robotteja, kuten maa-aluksia, ilma-aluksia tai veneitä. Gazebo tukee useamman robotin simuloimista samanaikaisesti samassa virtuaaliympäristössä. [21.] Gazebon toiminnallisuus on hyvin integroitu ROS:n kanssa [22].

3 Sähköverkkojen tarkastukseen suunnitellun droonin rakenne

Tehdyssä työssä keskitytään kumppanitietokoneen käyttöönottoon droonissa ja sitä vastaavan simulointiympäristön kehitykseen. Drooni on suunniteltu erityisesti voimalinjojen tarkastuksiin ArduPilot-pohjaisen lennonohjausjärjestelmän ja kumppanitietokoneessa toimivien ohjelmistojen avulla.

Lopputyössä käytössä ollut drooni on rakennettu Foxtechin GAIA -hexakopterin runkoon (Kuva 1). Käytetty drooni varustettiin Sony ILX-LR1 -kameralla, joka oli asennettu Gremsy-gimbaaliin. Kameraa ohjaa Entire-kameraohjain. Kuvauslaitteisto mahdollistaa korkearesoluutioisten kuvien ottamisen ja yksityiskohtaisen infrastruktuurianalyysin.



Kuva 1 Droonin laitteistoa.

Livox Avia LiDAR -anturi oli asennettuna hankkeen puitteessa suunniteltuun ja rakennettuun gimbaaliin, joka perustui 3D-tulostettuihin osiin.

Droonin ja maa-aseman välinen viestintä oli toteutettu HereLink-järjestelmää käyttäen, joka mahdollistaa pienen latenssin videokuvan ja telemetrian tiedonsiirron. HereLinkin tarjoamien yhteyksien lisäksi järjestelmään oli integroitu Elsight Halo -modeemi, joka pystyy ylläpitämään jopa neljää rinnakkaista mobiiliyhteyttä.

Tällä saadaan varmistettua luotettava tiedonsiirto myös syrjäisissä paikoissa, mikä lisää toiminnan joustavuutta ja turvallisuutta, sillä kaksi tiedonsiirtojärjestelmää, jossa toisessa on neljä rinnakkaista mobiiliyhteyttä, mahdollistaa jatkuvan ohjauksen ja valvonnan tilanteessa, jossa jokin useista tiedonsiirtokanavista katkeaa lennon aikana.

3.1 NVIDIA Jetson XAVIER NX -kumppanitietokone

NVIDIA Jetson Xavier NX on tehokas reunalaskentajärjestelmä moduulina (engl. system on a module). Moduulin dimensiot ovat 70 mm kertaa 45 mm. Sen keskeisimpiä käyttötarkoituksia ovat autonomiset järjestelmät (engl. autonomous systems), robotiikka ja tekoälysovellukset. Jetsonin suorittimena toimii kuusiytiminen 64-bittinen NVIDIA Carmek ARM -suoritin ja siinä on 8 gigatavua DDR4-muistia. Mekaanisena liittimenä Jetsonissa toimii 260-pinninen SO-DIMM-liitin, jolla Jetson kytketään kantokorttiin. [23.] Jetson käyttää NVIDIAN kehittämää Linux-versiota, joka pohjautuu Ubuntuun [24]. Lopputyössä käytössä olleessa Jetson-moduulissa on 16 gigatavua eMMC (embedded MultiMediaCard) -tallennustilaa. Lopputyössä käytetyssä droonissa Jetsonin yksi kantokortin UART (Universal Asynchronous Receiver-Transmitter) -liittimistä oli kytketty MavPorterin Port 2 -liittimeen.

3.2 MAVLINK-viestien reititys MavPorter-moduulilla

ArduPilot-ympäristössä telemetria-, komento- ja ohjausviestien siirto perustuu MavLink-protokollaan. MavPorter-moduulilla saadaan reititettyä MavLink-viestit eri laitteiden kuten autopilotin, kameraohjaimen, gimbaalin ja mobiilitietoliikenneyksikön välillä. [25.]

3.3 Cube black -autopilotti

Cube Black -autopilotti on kaupalliseen käyttöön suunniteltu, edistynyt ja modulaarinen autopilotti, joka perustuu Pixhawk-autopilottiin. Varman ja katkeamattoman toiminnallisuuden takaamiseksi siinä on useita redundantteja järjestelmiä, kuten esimerkiksi kolme redundanttia IMU (Inertial Measurement Unit) -

yksikköä. Cube Black tukee ArduPilot-autopilottiohjelmistoa. [26.] Lopputyössä käytetyssä droonissa Cube Black -autopilotti Telem 1 UART -liitäntä oli kytkettynä MavPorter-reitittimen Port 1-liitäntään ja Telem 2 UART -liitäntä oli kytketty Herelink-radioon.

3.4 Livox AVIA LiDAR -laserkeilain

Livox AVIA on luotettava ja kevyt 3D LiDAR -anturi, jonka keskeisimpiä ominaisuuksia ovat sen pitkä havaintoetäisyys ja laaja näkökenttä. Etenkin matalan reflektiviteetin esineiden tunnistus on kehittynyt. Koska AVIA painaa vain 498 grammaa, se on sopiva droonilla suoritettavaan kartoittamiseen. AVIA:n näkökenttä on 77,2 astetta vertikaalisesti ja 70,4 astetta horisontaalisesti. Single return -moodissa anturi tunnistaa 240 000 pistettä sekunnissa. On huomioitavaa, että AVIA ei tunnista esineitä, jotka ovat yhtä metriä lähempänä anturia. Hyvin valaistussa ympäristössä AVIA voi havaita esineitä jopa 320 metrin päästä. [27.] Lopputyössä käytetyssä droonissa Livox AVIA oli kytkettynä USB (Universal Serial Bus) -ethernet-adapterin avulla kumppanitietokoneen USB-C-porttiin.

3.5 Elsight HALO LTE -mobiilitiedonsiirtojärjestelmä

Elsight HALO on etenkin droonien käyttöön suunniteltu mobiiliverkkojen yhteydenhallintajärjestelmä. Droonissa on käytössä LTE-versio kortista, jossa on 4 erillistä SIM-korttipaikkaa. Mahdollisuus käyttää useampaa SIM-korttia samanaikaisesti takaa katkeamattoman yhteyden droonin ja GCS:n välillä. Verkkoyhteys droonin ja GCS:n välillä käyttää VPN-yhteyttä, joka lisää liikenteen turvallisuutta. Elsight HALO tukee MAVLINK-protokollaa. [28.] Elsight HALO oli lopputyössä käytetyssä droonissa kytketty Cube Black -autopilotin USB-liitäntään.

3.6 Sony ILX-LR1

Sony ILX-LR1 on ammattikäyttöön suunniteltu kamera, jossa on 61 megapikselin kenno. Pienen koon ja etäohjattavuusmahdollisuuksien ansiosta se soveltuu hyvin käyttöön droonien kanssa. ILX-LR1 painaa 243 grammaa. [29.] Kameran

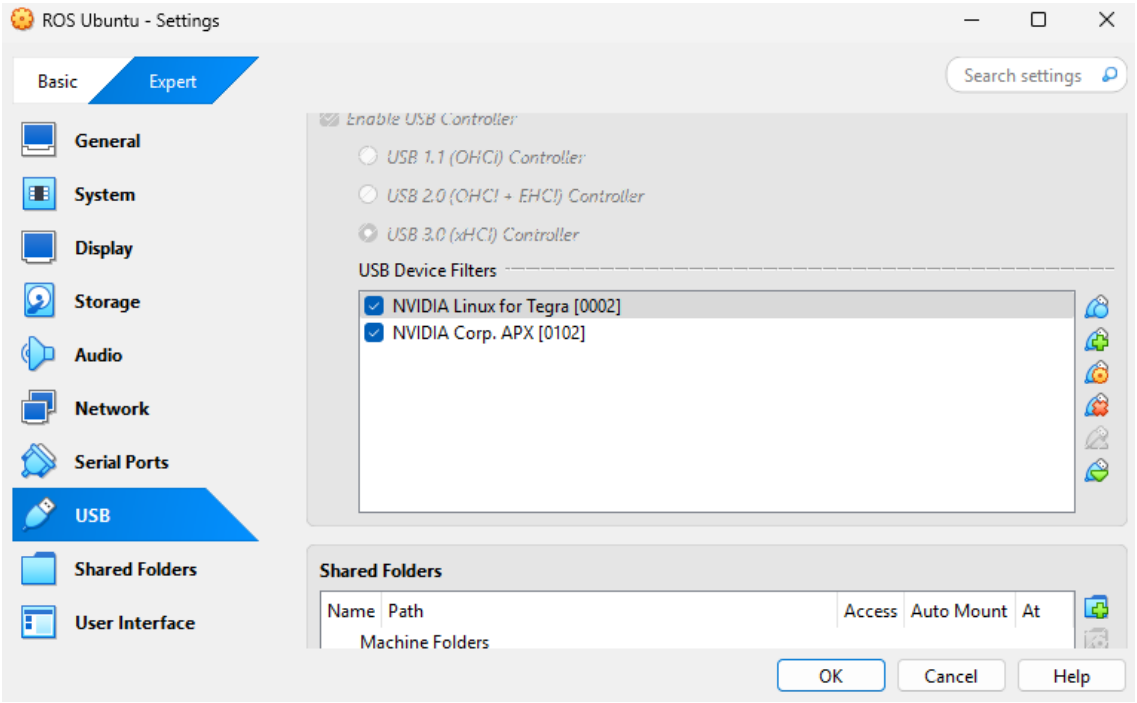
ohjaimena droonissa oli Air Commander ENTIRE R3, joka oli kytketty MavPorter-reitittimen Port 4 -liitäntään.

4 Laitteiston integrointi kumppanitietokoneeseen

Laitteiston integrointi kumppanitietokoneeseen aloitettiin alustamalla NVIDIA:n kehittämä Linux-versio Jetsonille käyttäen NVIDIAN SDK Manager -ohjelmistoa.

Ensimmäistä kertaa alustaessa Jetsonin pitää olla asetettuna pakotettuun palautustilaan (engl. force recovery mode). Pakotettu palautustila saadaan asetettua päälle maadoittamalla FC REC -nimetty pinni. Lopputyössä käytössä olleessa Jetsonin kantokortissa oli asennettuna mikrokytkin, jolla pystyi maadoittamaan edellä mainitun pinnin. [30.]

Vaikka virallisesti NVIDIA SDK Manager ei tue Jetsonin alustamista virtuaalikoneella, suoritettiin alustus kuitenkin virtuaalikonetta käyttäen, jotta ei olisi tarvetta asentaa NVIDIA SDK Manager -ohjelmiston vaatimaa Ubuntu-versiota oikealle laitteistolle. On huomioitavaa, että lopputyön tekemisen aikana NVIDIA on päivittänyt SDK Manager -ohjelmiston siten, että on mahdollista alustaa Jetson myös vanhemmilla JetPack-versioilla, vaikka SDK Manager olisi asennettu uudemmalle Ubuntu-versiolle. Jotta pystyy asentaa vanhempia JetPack-versioita uudella SDK Manager -ohjelmistolla, pitää SDK Manager -ohjelmisto käynnistää parametrillä --archived-versions. Virtuaalikoneella alustamisen onnistumista varten käytettiin Oraclen VirtualBox -ohjelmistoa ja asennettiin isäntätietokoneelle (engl. host) Virtualbox Extension pack. Asentamalla Extension pack -laajennus on mahdollista käyttää USB 3.0 -protokollaa virtuaalitietokoneessa. Virtuaalitietokoneen USB-asetuksissa lisättiin sekä alustetun Jetsonin USB-tunnus että pakotetussa palautustilassa olevan Jetsonin USB-tunnus, jotta Jetson saadaan suoraan liittymään virtuaalikoneen USB:hen (kuva 2).



Kuva 2 Oracle VirtualBox -virtuaalitietokoneen USB-asetukset.

Hankkeessa oli ollut käytössä myös muuta ohjelmistoa, jonka toivottiin toimivan Jetsonilla. Yhteensopivuuden takaamiseksi oli päädytty käyttämään ROS:n melodic-versiota. ROS melodicin yhteensopivuuden takaamiseksi Jetsonin kanssa alustettiin Jetson JetPack 4.6.3 -ohjelmiston versiolla.

Jetson luo virtuaaliverkkoyhteyden (engl. virtual network) micro-USB-kytkentää käyttäen, kun siihen liitetään toinen tietokone. Jetson asettaa itselleen IP-osoitteen 192.168.55.1 ja käynnistää DHCP (Dynamic Host Configuration Protocol) -palvelimen, joka antaa micro-USB:hen kytkettävälle tietokoneelle IP-osoitteen 192.168.55.100 virtuaaliverkkoon. Yhteys Jetsoniin saatiin kytkemällä kantokortin micro-USB-liitäntä tietokoneeseen ja ottamalla SSH-yhteys 192.168.55.1 IP-osoitteeseen (esimerkkikoodi 2).

```
PS C:\Users\User> ssh jetson@192.168.55.1
jetson@192.168.55.1's password:
jetson@192.168.55.1:~$ echo $SSH_CONNECTION
192.168.55.100 58099 192.168.55.1 22
jetson@192.168.55.1:~$
```

Esimerkkikoodi 2. Komentokehoitteet, joilla otetaan SSH-yhteys Jetsoniin.

ROS:n asennus tehtiin lisäämällä ROS:n repositoriot APT (Advanced package Tool) -paketinhallinnan repositoriolistaan ja asennus tehtiin käyttämällä APT-paketinhallinnan install-komentoa.

Ennen kuin on mahdollista aloittaa ROS:n käyttö, pitää ensin asettaa sen kehitysympäristö Linuxissa. Tämä tapahtuu ajamalla ROS:n setup-komentotulkin (engl. shell) skripti source-komennolla. Source-komennon käyttö ajaa skriptin saman komentotulkin kontekstissa kuin mistä se kutsuttiin. Toiminta on suunniteltu tällaiseksi, jotta mahdollistetaan useamman eri ROS-version käyttö samalla käyttöjärjestelmäasennuksella. Koska lopputyössä oli kuitenkin vain yksi tietty ROS-versio käytössä, asetettiin ROS-kehitysympäristön asettava komento .bashrc-tiedostoon, jotta se ajettaisiin joka kerta, kun uusi komentotulkki avataan. Alustamisen jälkeen lisättiin linux-käyttäjä dialout-ryhmään, jotta on mahdollista käyttää sarjaporttia (esimerkkikoodi 3). [31.]

```
sudo adduser $USER dialout
```

Esimerkkikoodi 3. Komentokehote, jolla sisään kirjautunut käyttäjä lisätään dialout-ryhmään.

ROS:n asennuksen jälkeen Jetsonille asennettiin ROS melodic -version kanssa yhteensopiva MAVROS-versio (ros-melodic-mavros). Tämä tehtiin APT-paketinhallintaa käyttäen. MAVROS-asennuksen jälkeen asennettiin vielä Geographiclibin tietoaineistot. MAVROS käyttää Geographiclibiä muuntaakseen korkeuksia AMSL (Above Mean Sea Level) -muodosta WGS84 (World Geodetic System 1984) -muotoon. [20.]

MAVROSin launch tiedostoon asetettiin Cube Black -autopilottiin kytketty UART-laite (esimerkkikoodi 4).

```
<launch>
     <arg name=”fcu_url” default=”/dev/ttyUSB0:115200” />
     <arg name=”gcs_url” default=”” />
     <arg name=”tgt_system” default=”1” />
     <arg name=”tgt_component” default=”1” />
     <arg name=”log_output” default=”screen” />
     <arg name=”fcu_protocol” default=”v2.0” />
     <arg name=”respawn_mavros” default=”true” />

     <include file=”$(find mavros)/launch/node.launch”>
          <arg name=”pluginlists_yaml” value=”$(find mav-
ros)/launch/apm_pluginlists.yaml” />
          <arg name=”config_yaml” value=”$(find mavros)/launch/apm_con-
fig.yaml” />

          <arg name=”fcu_url” value=”$(arg fcu_url)” />
          <arg name=”gcs_url” value=”$(arg gcs_url)” />
          <arg name=”tgt_system” value=”$(arg tgt_system)” />
          <arg name=”tgt_component” value=”$(arg tgt_component)” />
          <arg name=”log_output” value=”$(arg log_output)” />
          <arg name=”fcu_protocol” value=”$(arg fcu_protocol)” />
          <arg name=”respawn_mavros” value=”$(arg respawn_mavros)” />
     </include>
</launch>
```

Esimerkkikoodi 4. apm.launch-tiedosto.

MAVROS asetettiin käynnistymään automaattisesti tekemällä crontab servicellä työ (engl. job), joka kutsuu jokaisen käynnistyksen yhteydessä bash (Bourne Again SHell) -skriptin, joka käynnistää MAVROS:n launch-tiedoston (esimerkkikoodi 5).

```
jetson@ubuntu:~$ crontab -l | grep reboot
@reboot bash /home/jetson/startup.sh
jetson@ubuntu:~$
```

Esimerkkikoodi 5. Komentokehoitteet, joilla näkee crontab-työn, joka kutsuu käynnistysskriptiä.

Käytössä olleessa Jetsonin versiossa oli vain 16 gigatavua tallennustilaa. Tämä on suhteellisen vähän tallenustilaa varsinkin, kun käyttötarkoituksena on suorittaa kartoitusta LiDAR-anturia käyttäen. Lopputyössä käytetyssä Jetsonin kantokortissa on lukija SD (Secure Digital) -kortille, mutta sen käyttöönotto Linuxissa vaati DT (Devicetree) -tiedoston käyttämistä. DT on tietorakenne ja kieli, jolla kuvataan laitteistoa käyttöjärjestelmälle, jotta laitteiston rakennetta ei tarvitse kovakoodata kerneliin. [32.]

Esimerkkikoodi 2, näyttää miten DTB (Devicetree blob) -tiedosto käännettiin DTS (Devicetree source) -tiedostoksi dtc-työkalulla.

```
dtc -I dtb -O dts -o custom.dts /boot/tegra194-p3668-all-p3509-
0000.dtb
```

Esimerkkikoodi 6. Komentorivikehote, jolla käännettiin DTB-tiedosto DTS-muotoon.

DTS-tiedostossa ”sdhci@3400000” -laitteelle asetettiin tila (engl. status) ”okay”. (esimerkkikoodi 7)

```
sdhci@3400000 {
     compatible = "nvidia,tegra194-sdhci";
     reg = <0x0 0x3400000 0x0 0x20000>;
     interrupts = <0x0 0x3e 0x4>;
     iommus = <0x2 0x1a>;
     dma-coherent;
     max-clk-limit = <0xc65d400>;
     bus-width = <0x4>;
     cap-mmc-highspeed;
     cap-sd-highspeed;
     sd-uhs-sdr104;
     sd-uhs-sdr50;
     sd-uhs-sdr25;
     sd-uhs-sdr12;
     mmc-ddr-1_8v;
     mmc-hs200-1_8v;
     nvidia,vqmmc-always-on;
     cd-inverted;
     nvidia,min-tap-delay = <0x60>;
     nvidia,max-tap-delay = <0x8b>;
     pwrdet-support;
     pinctrl-names = "sdmmc_e_33v_enable", "sdmmc_e_33v_disable";
     pinctrl-0 = <0x20>;
     pinctrl-1 = <0x21>;
     ignore-pm-notify;
     resets = <0x5 0x52>;
     reset-names = "sdhci";
     pll_source = "pll_p", "pll_c4_muxed";
     nvidia,set-parent-clk;
     nvidia,parent_clk_list = "pll_p", "pll_p", "pll_p", "pll_p",
"pll_p", "pll_c4_muxed", "pll_c4_muxed", "pll_c4_muxed",
"pll_c4_muxed", "pll_c4_muxed", "NULL";
     clocks = <0x4 0x78 0x4 0x66 0x4 0xf1 0x4 0xdb>;
     clock-names = "sdmmc", "pll_p", "pll_c4_muxed", "sdmmc_legacy_tm";
     uhs-mask = <0x8>;
     nvidia,en-periodic-calib;
     status = "okay";
```

Esimerkkikoodi 7. Osa muutettua DTS-tiedostoa.

DTS-tiedosto käännettiin takaisin DTB-muotoon käyttämällä dtc-komentoa. DTB-tiedosto otettiin käyttöön asettamalla extlinux-bootloaderin asetuksiin kehote sen lataamista varten ja asettamalla DTB-tiedoston nimi (esimerkkikoodi 8).

```
TIMEOUT 30
DEFAULT primary

MENU TITKE L4T boot options

LABEL primary
     MENU LABEL primary kernel
     LINUX /boot/Image
     INITRD /boot/initrd
     FDT /boot/custom.dtb
     APPEND ${cbootargs} quiet root=/dev/mmcblk0p1 rw rootwait root-
fstype=ext4 console=ttyTCU0,115200n8 console=tty0 fbcon=map:0 net.if-
names=0 nv-auto-config
```

Esimerkkikoodi 8. Extlinux.conf-tiedosto.

Jotta Linux-käyttöjärjestelmä pystyy käyttämään SD-kortin sisältöä, pitää se liittää (engl. mount). Lopputyössä päädyttiin liittämään SD-kortti automaattisesti käynnistyksen yhteydessä crontabilla samalla tavalla kuin MAVROS:n käynnistys tehtiin.

Livox AVIA LiDAR tarvitsee toimiakseen Livox-SDK-ohjelmiston [33]. Sen asennus suoritettiin lataamalla Livox-SDK:n lähdekoodin sisältävä git-repositorio ja kääntämällä lähdekoodi binääreiksi cmakella. Jotta LiDAR-anturin tietoa oli mahdollista käyttää ROS:n kanssa, asennettiin myös Livox:n julkaisema ROS-ajuri. Asennus tehtiin lataamalla ROS-työtilan (engl. workspace) sisältävä git-repositorio ja kääntämällä se ROS:n rakennusketjun (engl. buildchain) catkin työkalua käyttäen. Jotta Livox:n ROS-ajuria pystyy käyttämään, pitää sen ROS-paketin alustusskripti ajaa. Paketin alustusskriptin ajo lisättiin myös .bashrc-tiedostoon, jotta sitä ei tarvitse ajaa uudelleenkäynnistyksien jälkeen. Livox ROS -ajuri tukee mm. PCL:n (Point Cloud Library) ja ROS:n tukemia pointcloud1- ja pointcloud2-tietomalleja, mutta Livox on luonut myös oman tietomallin pistepilviviestille. [34.]

Droonissa käytössä olleessa Jetsonin kantokortissa ei ollut ethernet-porttia, minkä takia jouduttiin käyttämään ulkoista USB-ethernet-adapteria liittämään LiDAR-anturi Jetsoniin. LiDAR-anturin yhdistäminen Linuxin kanssa tehtiin vaihtamalla USB-ethernet-adapterin IP-osoite osoitteeksi 192.168.1.50. (esimerkkikoodi 9) LiDAR-anturi ottaa yhteyden edellä mainittuun IP-osoitteeseen lähiverkossa [27].

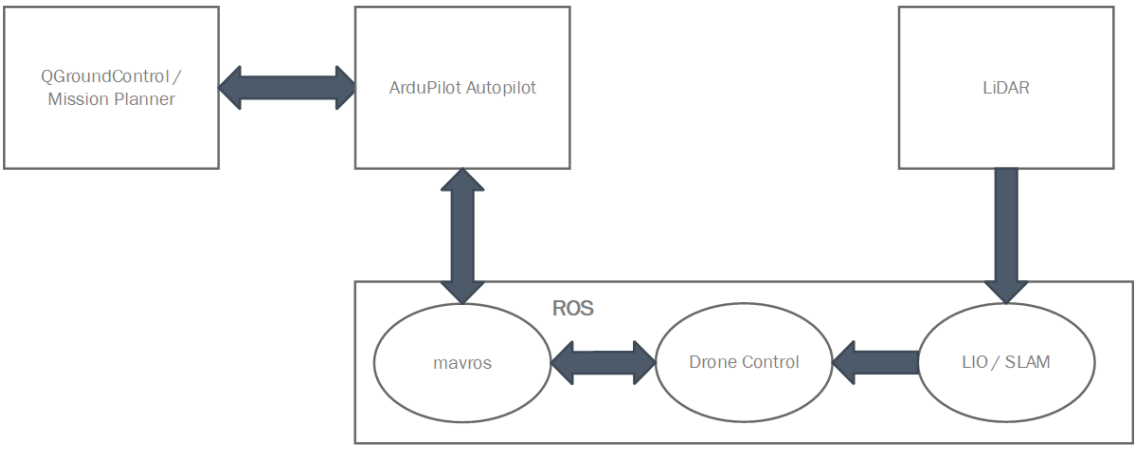
```
ifconfig eth1 192.168.1.50 netmask 255.255.255.0 broadcast
192.168.1.255
```

Esimerkkikoodi 9. Komentorivikehote, jolla vaihdetaan ethernet-adapterin IP-osoite.

Droonissa käytössä olleen Gremsy-kameragimballin ohjausohjelmisto asennettiin lataamalla lähdekoodin sisältävä git-repositorio. Ohjelmisto vaati uudempaa cmake-versiota kuin mikä oli asennettavissa Jetsonissa olleen linux-distribuution repositorioissa. Uudempi cmake asennettiin ajamalla ARM-yhteensopiva asennusskripti ja luomalla symbolinen linkki /usr/bin-kansioon uudelle cmake-binäärille.

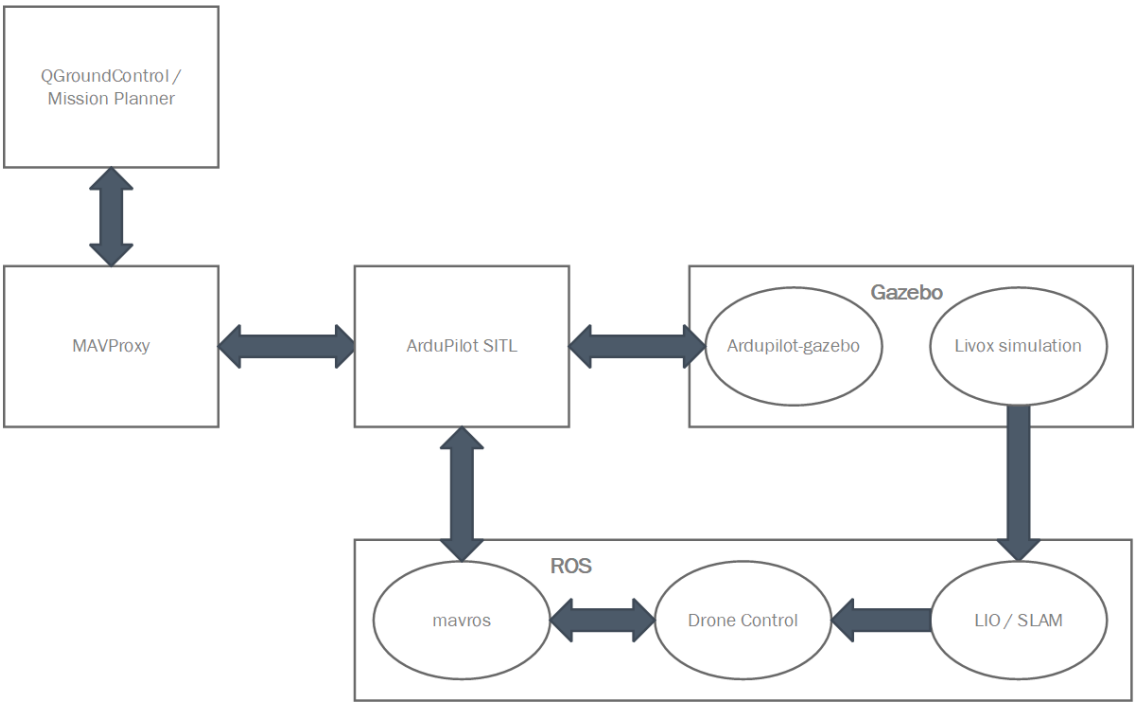
5 Simulointiympäristö

Lopputyön tavoitteena oli kehittää simulaatioympäristö, jonka ohjelmistoarkkitehtuuri vastaisi hankkeessa käytettyä oikeaa droonia. Droonin keskeisimmät osat ohjelmistoarkkitehtuuria ovat ROS ja ArduPilot (kuva 3).



Kuva 3 Droonin ohjelmistoarkkitehtuuri.

Oikeasta droonista poiketen simulaatiossa fyysinen simulaatio tehdään Gazebolla ja LiDAR-anturin toiminta on simuloitu (kuva 4). Lopputyössä käytettiin Livoxin julkaisemaa avoimen lähdekoodin Gazebo-laajennusta, joka pystyy mallintamaan AVIA LiDAR:ia. [35.]

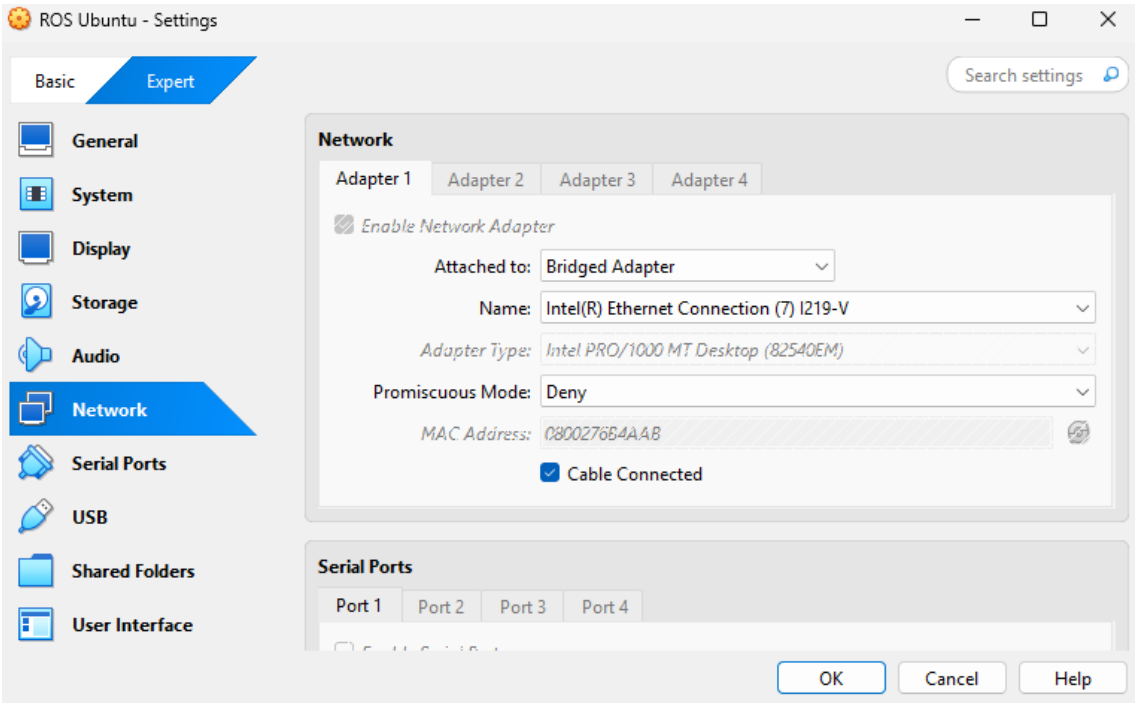


Kuva 4 Simulaattorin ohjelmistoarkkitehtuuri.

Simulaatiossa ei käytetty oikeaa droonia vastaavaa hexakopteria vaan mukautettua IRIS-nelikopterin Gazebo-mallia. Koska oikean droonin kamera ja sen ohjain eivät olleet kytkettynä kumppanitietokoneeseen, ei ole mahdollista ohjata niitä ohjelmallisesti ROS:n avulla. Tämän takia ei ollut syytä lisätä simuloituun drooniin kameraa.

Simulointiympäristö on toteutettu käyttämällä kahta virtuaalitietokonetta. Jotta virtuaalikoneet pystyvät keskustelemaan toistensa kanssa, pitää niiden olla kytkettynä samaan verkkoon. Lopputyössä tämä toteutettiin kytkemällä kummankin virtuaalikoneen verkkoadapterit Bridged-moodiin (kuva 5). Tämä kytkee virtuaalikoneet suoraan isäntätietokoneen (engl. host) verkkoadapteriin. Virtuaalikoneille asetettiin myös manuaalisesti staattiset IP-osoitteet, jotta ne eivät vaihtuisi uudelleenkäynnistyksen yhteydessä. Lopputyössä käytetyssä ympäristössä

ROS:ia ja Gazeboa suorittava virtuaalitietokoneen IP-osoite on 192.168.1.45. ArduPilot SITL:iä suorittava virtuaalitietokoneen IP-osoite on 192.168.1.51.



Kuva 5 Oracle Virtualbox -virtuaalitietokoneen verkkoadapterin asetukset.

5.1 ArduPilot SITL

ArduPilot-autopilotin SITL asennettiin Ubuntu 22-käyttöjärjestelmää käyttävään virtuaalikoneeseen. SITL-ohjelmistoa ajettiin ArduCopter-moodissa. Komentorivikehotteella SITL:iä käynnistäessä annettiin ArduPilot-ohjelmistolle ROS-virtuaalikoneen IP-osoite, jotta SITL löytää ArduPilot Gazebo -laajennuksen ja jotta ArduPilot edelleenlähettää MAVLINK-liikenteen MAVROS:lle. Yleisestä SITL:stä poiketen simulaatio ei tapahdu ArduPilotin ohjelmistossa vaan Gazebon puolella.

Esimerkkikoodi 10:ssa sim-address-parametri kertoo SITL:lle Gazebon osoitteen. -v ArduCopter -parametri kertoo, että käytettävä ajoneuvo on ArduCopter. -f gazebo-iris -parametri kertoo SITL:lle, että simulaatiossa käytettävä droonirunko on gazebo-iris-tyyppiä. —out 192.168.1.45 -parametri komentaa SITL:n edelleenlähettämään MAVLINK liikenteen MAVROS:lle. —out 192.168.1.113 -

parametri komentaa SITL:ä edelleenlähettämään MAVLINK-liikenteen IP-osoitteessa olevalle GCS:lle.

```
sim_vehicle.py --sim-address=192.168.1.45 -v ArduCopter -f gazebo-iris
--console --out 192.168.1.45:14550 --out 192.168.1.113:14550 --map
```

Esimerkkikoodi 10. Komentokehoite, jolla käynnistetään ArduPilot SITL.

5.2 ROS

Koska oikeassa droonissa on käytössä ROS:n melodic-versio, asennettiin virtuaalikoneelle Ubuntu 18 -käyttöjärjestelmä, joka on yhteensopiva käytetyn ROS-version kanssa. MAVROS-asennus suoritettiin samalla tavalla kuin oikeassa droonissa. MAVROS:n konfiguraatio poikkeaa oikean droonin konfiguraatiosta siten, että apm.launch-tiedostossa fcu_url-osoite ei ole autopilottiin kytketty sarjaportti vaan kuuntelevaksi asetettu UDP-portti. Käynnistyessä MAVROS jää kuuntelemaan MAVLINK-liikennettä edellä mainitulla portilla.

5.2.1 ArduPilot Gazebo -laajennus

Pelkän ArduPilot SITL:n käytöstä poiketen lopputyössä kehitetyssä simulaatiossa fyysinen simulaatio tapahtuu Gazebossa käyttäen ArduPilot Gazebo -laajennusta [36]. IRIS-nelikopterin SDF (Simulation Description Format) -malliin lisättiin ArduPilot Gazebo -laajennusta käyttävä osa koodia.

Esimerkkikoodi 11 on osa mukautetun IRIS-mallin SDF-tiedostoa. Esimerkkikoodissa ladataan ArduPilot Gazebo -laajennuksen binääritiedosto Gazeboon. Esimerkkikoodissa näkyy myös määritelmät ArduPilot SITL-virtuaalikoneen IP-osoittelle ja SITL:n käyttämät portit liikennettä varten.

```
<plugin name="arducopter_plugin" filename="libArduPilotPlugin.so">
     <listen_addr>0.0.0.0</listen_addr>
     <fdm_addr>192.168.1.51</fdm_addr>
     <fdm_port_in>9002</fdm_port_in>
     <fdm_port_out>9003</fdm_port_out>
```

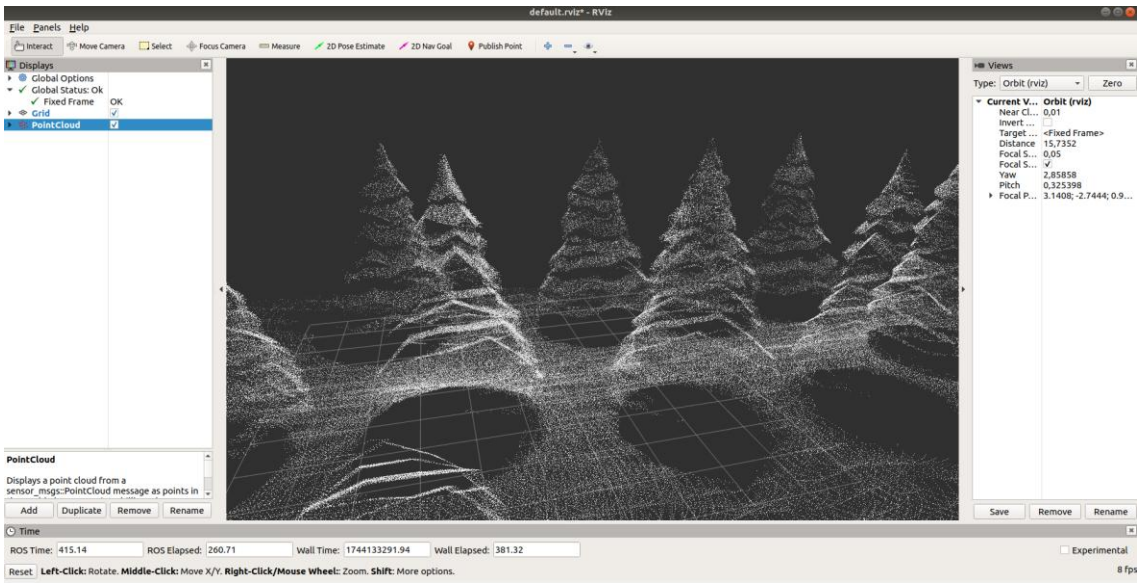
Esimerkkikoodi 11. IRIS-nelikopterin mallin SDF-tiedostoon lisättyä koodia, joka luo yhteyden ArduPilot SITL-virtuaalitietokoneen kanssa.

5.2.2 LiDAR-simulointi

Simuloituun drooniin otettiin käyttöön Livoxin julkaisema Gazebo-laajennus, joka pyrkii mallintamaan Livoxin AVIA LiDAR:n ominaisuuksia mahdollisimman tarkkaan (kuva 6) [35]. Esimerkkikoodi 12 näyttää muokatun IRIS-droonin mallin koodista osan, jossa laajennuksen binääri ladataan Gazeboon. Koodiesimerkissä näkyy myös, kuinka simuloidun LiDAR:n näkökentän ja havaintoetäisyyden ominaisuudet määritellään. Kolmanneksi viimeisellä rivillä esimerkkikoodia määritellään CSV (Comma-seperated values) -tiedosto ladattavaksi Gazebo-laajennukselle, joka määrittelee simuloidun LiDAR-anturin havaintokuvion vastaamaan Livox AVIA:n tuottamaa kuviota repetitiivisessä moodissa. Toiseksi viimeisellä rivillä määritellään pointcloud1-tietomallista pistepilvidataa saavan ROS-aiheen nimeksi /scan.

```
<plugin name="gazebo_ros_laser_controller" filename="liblivox_la-
ser_simulation.so">
     <ray>
          <scan>
               <horizontal>
                    <samples>100</samples>
                    <resolution>1</resolution>
                    <min_angle>${-70.4/360*M_PI}</min_angle>
                    <max_angle>${70.4/360*M_PI}</max_angle>
               </horizontal>
               <vertical>
                    <samples>50</samples>
                    <resolution>1</resolution>
                    <min_angle>${-77.2/360*M_PI}</min_angle>
                    <max_angle>${77.2/360*M_PI}</max_angle>
               </vertical>
          </scan>
          <range>
               <min>0.1</min>
               <max>320</max>
               <resolution>0.002</resolution>
          </range>
          <noise>
               <type>gaussian</type>
               <mean>0.0</mean>
               <stddev>0.01</stddev>
          </noise>
     </ray>
          <visualize>True</visualize>
          <samples>24000</samples>
          <downsample>1</downsample>
          <csv_file_name>/home/user/ardupi-
lot_ws/src/dronedata/scan_mode/avia.csv</csv_file_name>
          <ros_topic>/scan</ros_topic>
</plugin>
```

Esimerkkikoodi 12. IRIS-nelikopterin mallin SDF-tiedostoon lisättyä koodia, joka simuloi Livox AVIA LiDAR -anturin toimintaa.



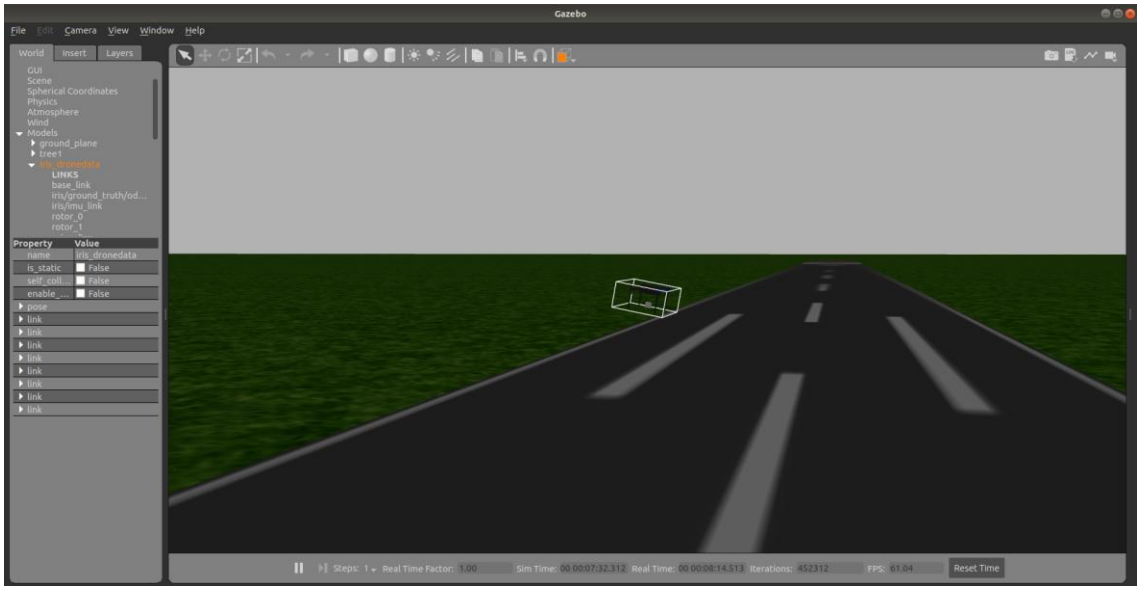
Kuva 6 Simuloidun LiDAR-anturin tuottamaa pistepilvidataa visualisoituna RVIZ-ohjelmalla.

5.3 ROS launch

ROS launch -tiedostot ovat XML (Extensible Markup Language) -tiedostoja, joiden avulla voi käynnistää toisia launch-tiedostoja, useampia ROS-solmuja ja hallita parametrejä automatisoidulla tavalla. Simulointiympäristön käynnistys oli toteutettu tekemällä ROS launch -tiedosto. Esimerkkikoodi 13 on lopputyössä käytetty ROS launch -tiedosto. Esimerkkikoodissa ladataan ensin YAML (YAML Ain’t Markup Language) -muotoinen konfiguraatiotiedosto, minkä jälkeen käynnistetään apm.launch-niminen launch-tiedosto, joka käynnistää MAVROS:n. Seuraavaksi käynnistetään dronedata ROS-paketissa oleva dronecontrol-solmu, joka ohjaa simuloitua droonia (kuva 7). Seuraavaksi käynnistettävässä gazebo_world.launch-tiedostossa käynnistetään Gazebo-simulaattori ja ladataan 3D-maailma, joka sisältää mukautetun IRIS-droonin. Esimerkkikoodin lopussa käynnistetään ROS-aiheiden visualisointia varten käytetty RVIZ-ohjelma.

```
<launch>
     <rosparam file="$(find dronedata)/config/dronedata_cfg.yaml" />

     <include file="$(find dronedata)/launch/apm.launch" />

     <node pkg="dronedata" name="dronecontrol" type="dronecontrol" out-
put="screen" />

     <include file="$(find dronedata)/launch/gazebo_world.launch" />

     <node type="rviz" name="rviz" pkg="rviz" />
</launch>
```

Esimerkkikoodi 13. ROS launch-tiedosto, joka käynnistää Gazebo-simulaattorin, MAVROS:n, droonia ohjaavan ROS-solmun ja RVIZ-ohjelman.



Kuva 7 IRIS-drooni simuloidun LiDAR:n kanssa lennossa Gazebo-simulaattorissa.

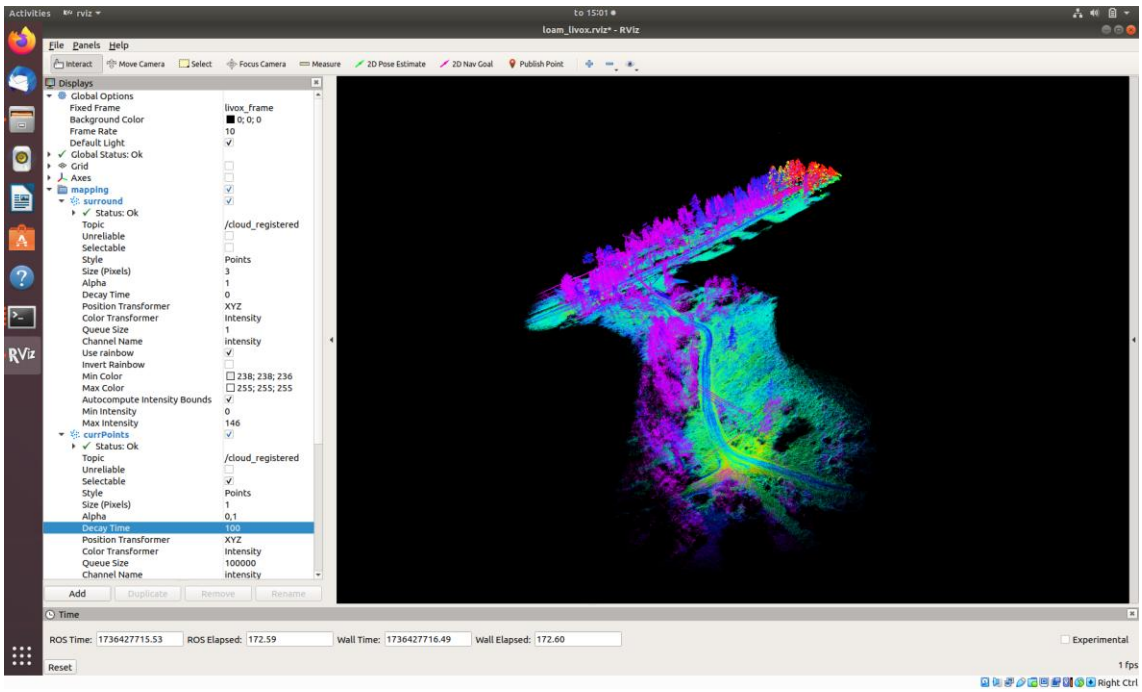
6 Pohdinta

6.1 Tulokset

Ensimmäinen koelento droonilla, jossa kaikki anturit olivat käytössä, suoritettiin 15. tammikuuta 2025 onnistuneesti (kuva 8). Lennon aikana tallennettiin ROSbag. ROSbag-tiedosto sisältää kaiken ROS-aiheissa tapahtuneen viestinnän. ROSbag-tiedostossa olleella pistepilvidatalla tehtiin PCD (Point Cloud Data) -pistepilvitiedosto FAST-LIO (Fast LiDAR-Inertial Odometry) -algoritmia

käyttäen. FAST-LIO käyttää Livox ROS -ajurin tuottamaa pistepilvidataa Livoxin mukauttamana tietomallina ja LiDAR:n IMU-anturin havaintoja (kuva 9). [37.]

Kuva 8 Drooni lähtenyt lentoon.



Kuva 9 FAST-LIO-algoritmin tuottama pistepilvi visualisoituna RVIZ-ohjelmalla.

6.2 Havaitut ongelmat ja kehitysmahdollisuudet

Livox:n LiDAR:ia simuloiva laajennus julkaisee pistepilvidataa pointcloud1-muodossa ROS-aiheeseen. Droonissa käytössä olleesta Livoxin ROS-ajurista poiketen simuloidun LiDAR:n ohjelmisto ei suoraan tue Livoxin mukautetun CustomMsg ROS -viestin käyttöä, joka sisältää myös aikaleiman LiDAR-havainnosta. Esimerkiksi FAST-LIO-algoritmi, jota käytettiin testilennossa hankitun tietoaineiston jatkokäsittelyssä, pystyy korjaamaan vääristymiä pistepilvessä, jos käytössä on Livox:n CustomMsg-tietomalli. Jatkokehityksenä lopputyölle voisi olla Livox-simulaation muokkaaminen siten, että se pystyy julkaisemaan myös Livox:n omaa CustomMsg ROS-viestiä. Vaikka lähdekoodi LiDAR:n simulointia varten ei ole järin pitkä, ovat ainoat kaksi kommenttia lähdekoodissa kiinaksi, minkä takia lähdekoodin muokkaaminen saattaa osoittautua kuitenkin varsin työlääksi [35].

Droonissa olevan Jetsonin asennus oli suoritettava ilman mahdollisuutta käyttää graafista käyttöliittymää, koska Jetsonin kantokortissa ei ollut liittimiä, joihin olisi voinut kytkeä näytön. Täysin grafiikkaliittymätön (engl. headless) asennus tuotti ongelmia esimerkiksi Elsight HALO -mobiilireitittimen ja Jetsonin välisien verkkoyhteysongelmien selvittämisessä. Verkkoyhteysongelmien selvityksessä olisi ollut tarve kokeilla USB:n yli toimivan SSH-yhteyden kytkemistä pois päältä, mutta grafiikkaliittymän puutteen takia silloin ei olisi ollut enää mahdollisuutta ottaa yhteyttä Jetsoniin. Jatkossa vastaavanlaisten ongelmien ehkäisemiseksi olisi suotuisaa, että Jetsonin kehityksessä olisi myös käytettävissä kehittäjäkantokortti (engl. developer kit), jossa on myös liitännät näyttöä varten. Yhteysongelmaa ei saatu korjattua lopputyön tekemisen aikana.

Lopputyössä valittu virtualisointiohjelmisto Oracle VirtualBox tuotti ongelmia. Ohjelmisto lopetti toiminnan muutaman kerran täysin satunnaiselta tuntuvasti, mutta virhetilanteet onnistuttiin aina korjaamaan. Vikatilanteisiin kului kuitenkin turhaa työaikaa. Lopputyön teossa käytettiin myös QEMU- ja virt-manager-virtualisointiohjelmistoja, mutta koska nämä eivät tue Windows-käyttöjärjestelmää, päädyttiin lopputyössä käyttämään ongelmista huolimatta VirtualBox-ohjelmistoa.

MAVROS-dokumentaation mukaan droonin saapuessa tiettyyn reittipisteeseen (engl. waypoint) kääntää MAVROS MISSION_ITEM_REACHED MAVLINK -viestin WaypointReached ROS -viestiksi aiheeseen mission/reached. Simulaatiossa todettiin, että ohjelmisto käyttäytyy dokumentaation mukaisesti ja aiheeseen välittyi ROS-viesti, kun drooni saapui reittipisteeseen. Oikeassa droonissa tosin huomattiin, että Cube Black -autopilotti ei lähetä edellä mainittuja viestejä tai MAVROS ei niitä käännä. Ongelman ilmettyä todettiin, että oikeassa droonissa ei ollut mahdollista pyytää autopilotissa olevaa lentosuunnitelmaa ollenkaan MAVROS:n kautta, toisin kuin dokumentaatiossa kerrottiin. Simulaatiossa lentosuunnitelman pyytäminen onnistui dokumentaation mukaisesti. Lopputyön jälkeen suoritettavassa ohjelmistokehityksessä oli suunniteltu, että droonia ohjataan reittipisteitä muokkaamalla.

USB-ethernet-adapterin käyttö Livox AVIA LiDAR:n yhdistämisessä Jetsoniin tuotti uudelleenkäynnistyksien yhteydessä yhteysongelmia. Jetsonin käyttöjärjestelmä ei uudelleenkäynnistyksen jälkeen aina tunnistanut USB-ethernet-adapteria. Yhteysongelmat saatiin korjattua ennen lentoja irrottamalla USB-ethernet-adapteri ja kytkemällä se takaisin Jetsoniin. Kehitysmahdollisuuksia voisivat olla yrittää saavuttaa uudelleenkytkeminen skriptin avulla ohjelmallisesti tai käyttämällä Jetson-kantokorttia, missä on ethernet-liitäntä. Ongelman juurisyyn selvittämiseen olisi auttanut, että lopputyötä tehdessä olisi ollut myös käytettävissä Jetsonin kehittäjäkantokortti (engl. developer kit), jossa on ethernet-liitäntä.

6.3 Yhteenveto

Lopputyön yhtenä tavoitteena oli integroida "Droonit ja data-analytiikka sähköverkkojen vianpaikannuksessa ja kunnossapidossa" -hankkeessa käytössä olleen droonin anturilaitteisto Jetson-kumppanitietokoneen kanssa. Lopputyön tavoitteessa onnistuttiin ja ensimmäinen koelento, jossa kaikki anturit olivat käytössä, suoritettiin 15. tammikuuta 2025. Lopputyön aikana ilmennyttä yhteysongelmaa Jetson-kumppanitietokoneen ja Elsight HALO -mobiilireitittimen välillä ei saatu ratkaistua.

Lopputyön toinen tavoite oli droonin ohjelmistoarkkitehtuuria vastaavan simulaatioympäristön kehittäminen. Simulaatioympäristö saatiin vastaamaan oikeassa droonissa olevaa ohjelmistoarkkitehtuuria; tosin simuloidun LiDAR-anturin toimintaa ei saatu täysin vastaamaan oikean anturin toimintaa.

Lopputyön aihe oli erittäin kiinnostava ja opin paljon uutta varsinkin drooneihin ja robotiikkaan liittyvästä ohjelmistokehityksestä ja siihen liittyvistä mahdollisuuksista.

Lähteet

1 RAF Aerial Target. Verkkoaineisto. FOP Shvacko V. V. <https://shvachko.net/?p=1378&lang=en>. 29.6.2013. Luettu 4.4.2025.

2 V-2 Jet Vane Motor. Verkkoaineisto. Cape Canaveral Space Force Museum. <https://ccspacemuseum.org/artifacts/v-2-jet-vane-motor/#:~:text=The%20V%2D2%20rocket%20engine,path%20of%20the%20engine%20exhaust>. Luettu 4.4.2025.

3 Hoversten, Paul. 2013. Were drones used in the Bikini bomb tests? Verkkoaineisto. Smithsonian magazine. <https://www.smithsonianmag.com/air-space-magazine/were-drones-used-in-the-bikini-bomb-tests-6578388/>. Luettu 4.4.2025.

4 A Brief History of Drones. Verkkoaineisto. Imperial War Museums. <https://www.iwm.org.uk/history/a-brief-history-of-drones>. Luettu 4.4.2025.

5 Franke, Ulrike. Drones in Ukraine: Four lessons for the West. Verkkoaineisto. 2025. <https://ecfr.eu/article/drones-in-ukraine-four-lessons-for-the-west/>. 10.1.2025 Luettu 4.4.2025.

6 Ingenuity Mars Helicopter. Verkkoaineisto. NASA <https://science.nasa.gov/mission/mars-2020-perseverance/ingenuity-mars-helicopter>. Luettu 4.4.2025.

7 Insitu Aerosonde Laima. Verkkoaineisto. The Museum of Flight. <https://www.museumofflight.org/exhibits-and-events/aircraft/insitu-areosonde-laima>. Luettu 4.4.2025.

8 Drone. Verkkoaineisto. Merriam Webster. <https://www.merriam-webster.com/dictionary/drone>. Luettu 4.4.2025.

9 Sadiku, Matthew N. O.; Adekunte, Paul A.; Sadiku, Janet. O.; 2024. Drones in Power Systems. Verkkoaineisto. International Journal of Trend in Scientific Research and Development <https://www.ijtsrd.com/papers/ijtsrd69436.pdf>. 5.9.2024. Luettu 10.4.2025.

10 Analyzing Safety Risks: Manned Aircraft vs. Drones for Powerline Inspection. 2023. Verkkoaineisto. <https://www.aizytech.com/post/analyzing-safety-risks-manned-aircraft-vs-drones-for-powerline-inspection>. 12.7.2023. Luettu 10.4.2025.

11 History of ArduPilot. Verkkoaineisto. ArduPilot. <https://ardupilot.org/copter/docs/common-history-of-ardupilot.html>. Luettu 5.4.2025.

12 ArduPilot Documentation. Verkkoaineisto. ArduPilot. <https://ardupilot.org/ardupilot/>. Luettu 5.4.2025.

13 Choosing an Autopilot. Verkkoaineisto. ArduPilot. <https://ardupilot.org/copter/docs/common-autopilots.html>. Luettu 5.4.2025.

14 MAVLink Interface. Verkkoaineisto. ArduPilot. <https://ardupilot.org/dev/docs/mavlink-commands.html>. Luettu 5.4.2025.

15 SITL Simulator (Software in the Loop). Verkkoaineisto. ArduPilot. <https://ardupilot.org/dev/docs/sitl-simulator-software-in-the-loop.html>. Luettu 5.4.2025.

16 Sciortino, Claudio; Fagiolini, Adriano; 2018. ROS/Gazebo-based Simulation of Quadcopter Aircrafts. Verkkoaineisto. IEEE. <https://www.researchgate.net/publication/329506092_ROSGazebo-Based_Simulation_of_Quadcopter_Aircrafts>. Luettu 5.4.2025.

17 Services. Verkkoaineisto. ROS. <https://wiki.ros.org/Services>. Luettu 6.4.2025.

18 Understanding ROS Nodes. Verkkoaineisto. ROS. <https://wiki.ros.org/ROS/Tutorials/UnderstandingNodes>. Luettu 6.4.2025.

19 Master. Verkkoaineisto. ROS. <https://wiki.ros.org/Master>. Luettu 6.4.2025.

20 Mavros. Verkkoaineisto. ROS. <https://wiki.ros.org/mavros>. Luettu 6.4.2025.

21 About Gazebo. Verkkoaineisto. Gazebo. <https://gazebosim.org/about>. Luettu 6.4.2025.

22 Tellez, Ricardo. 2019. A History of ROS (Robot Operating System). Verkkoaineisto. The Construct. <https://www.theconstruct.ai/history-ros/>. 7.9.2019. Luettu 6.4.2025.

23 Jetson Xavier NX Series. Verkkoaineisto. NVIDIA. <https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-xavier-nx/>. Luettu 7.4.2025.

24 Jetson Linux. Verkkoaineisto. NVIDIA. <https://developer.nvidia.com/embedded/jetson-linux>. Luettu 6.4.2025.

25 MavPorter. Verkkoaineisto. AirPixel. <https://airpixel.cz/mavporter/>. Luettu 5.4.2025.

26 The Cube Black. Verkkoaineisto. ArduPilot. <https://ardupilot.org/copter/docs/common-thecube-overview.html>. Luettu 3.4.2025.

27 Livox AVIA User Manual. 2020. Verkkoaineisto. DJI. <https://terra-1-g.djicdn.com/65c028cd298f4669a7f0e40e50ba1131/Download/Avia/Livox%20Avia%20User%20Manual%20202204.pdf> Luettu 3.4.2025.

28 Halo For Drone BVLOS Flights & Ground Unmanned Systems. 2024. Verkkoaineisto. Elsight. <https://www.unmannedsystemstechnology.com/wp-content/uploads/2020/10/Elsight-General-Halo-Datasheet-2024.pdf>. Luettu 6.4.2025.

29 ILX-LR1. Verkkoaineisto. Sony. <https://pro.sony/ue_US/products/installable-cameras/ilx-lr1>. Luettu 7.4.2025.

30 How to build NVIDIA a Jetson Xavier NX Kernel. Verkkoaineisto. RidgeRun Developer Wiki. <https://developer.ridgerun.com/wiki/index.php?title=Jetson_Xavier_NX/Development/Building_the_Kernel_from_Source#Jetson_Xavier_NX_Recovery_Mode>. Luettu 5.4.2025.

31 Installing and Configuring Your ROS Environment. Verkkoaineisto. ROS. <https://wiki.ros.org/ROS/Tutorials/InstallingandConfiguringROSEnvironment> Luettu 6.4.2025.

32 Likely, Grant. Linux and the Devicetree. Verkkoaineisto. The Linux Kernel Archives. <https://www.kernel.org/doc/html/latest/devicetree/usage-model.html#>. Luettu 10.4.2025.

33 Livox-SDK. Verkkoaineisto. Github. <https://github.com/Livox-SDK/Livox-SDK>. Luettu 6.4.2025.

34 Livox ROS Driver. Verkkoaineisto. Github. <https://github.com/Livox-SDK/livox_ros_driver>. Luettu 6.4.2025.

35 Livox Laser Simulation. Verkkoaineisto. Github. <https://github.com/Livox-SDK/livox_laser_simulation>. Luettu 6.4.2025.

36 ArduPilot Gazebo Plugin. Verkkoaineisto. Github. <https://github.com/khancyr/ardupilot_gazebo>. Luettu 6.4.2025.

37 FAST-LIO. Verkkoaineisto. Github. <https://github.com/hku-mars/FAST_LIO>. Luettu 6.4.2025