jamk.fi

ArduPilot autopilotin kehitys ja testaus

Reko Meriö

Opinnäytetyö
Maaliskuu 2021
Tekniikan ala
Insinööri (AMK), tieto- ja viestintätekniikka
Ohjelmistotekniikka

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

jamk.fi

Kuvailulehti

Tekijä(t)	Julkaisun laji	Päivämäärä
Meriö, Reko	Opinnäytetyö, AMK	Maaliskuu 2021
	Sivumäärä 64	Julkaisun kieli Suomi
		Verkkojulkaisulupa myönnetty: x

Työn nimi
ArduPilot autopilotin kehitys ja testaus

Tutkinto-ohjelma
Tieto- ja viestintätekniikka

Työn ohjaaja(t)
Jani Immonen, Esa Salmikangas

Toimeksiantaja(t)
Lentola Logistics Oy

Tiivistelmä

Opinnäytetyön tavoitteena oli perehtyä avoimeen lähdekoodiin perustuvan ArduPilot-autopilotin kehitykseen ja testaukseen, sekä tutkia ja pyrkiä ratkaisemaan opinnäytetyön toimeksiantajan siinä havaitsemia ongelmia. Työssä tutkitut ongelmat kohdistuivat erään ArduPilotin tukeman lennokkityypin lentotilan vaihdoksiin. ArduPilot on yksi maailman suosituimmista avoimen lähdekoodiin autopiloteista, mitä käytetään tyypillisesti multikoptereissa ja lennokeissa.

Opinnäytetyön toimeksiantajana toimi hämeenlinnalainen Lentola Logistics Oy, jonka tavoitteisiin kuuluu sähkökäyttöisten ilma-aluksien kehittäminen erilaisiin kuljetustehtäviin sekä kestävien kuljetusmenetelmien käyttöönoton kiihdyttäminen.

Opinnäytetyössä keskityttiin tutkimaan ArduPilotin käyttöä lennokeissa, vaikkakin ArduPilot tukee myös useita muita alustyyppejä. Varsinainen toteutus tehtiin tutkimalla ArduPilotin ominaisuuksia ja dokumentaatiota sekä analysoimalla projektin lähdekoodia, tekemällä siihen muutoksia ja testaamalla muutosten vaikutuksia simulaattorin avulla.

ArduPilotin lähdekoodiin tehdyt muutokset dokumentoitiin osaksi opinnäytetyötä, kuten myös kehitykseen ja testaukseen käytetyt menetelmät.

Opinnäytetyön tavoitteet saavutettiin ja työn lopputuloksena toimeksiantajan havaitsemien ongelmien syyt saatiin selvitettyä ja korjattua tekemällä muutoksia ArduPilotin lähdekoodiin.

Avainsanat (asiasanat)

ArduPilot, autopilotti, avoin lähdekoodi, drone, lennokki, ohjelmistokehitys, simulaattori

Muut tiedot (Salassa pidettävät liitteet)
MALLI: Liitteet 1,4 ja 7 ovat salassa pidettäviä, ja ne on poistettu julkisesta työstä. Salassapidon perusteena on viranomaisten toiminnan julkisuudesta annetun lain (621/1999) 24 §:n kohta 17: yrityksen liike-tai ammattisalaisuus. Salassapitoaika on viisi (5) vuotta. Salassapito päättyy 30.9.2024.

jamk.fi

Description

Author(s)	Type of publication	Date
Meriö, Reko	Bachelor's thesis	March 2021
		Language of publication: Finnish
	Number of pages: 64	Permission for web publication: x

Title of publication

Development and testing of ArduPilot autopilot

Degree programme

Information and communication technology

Supervisor(s)

Jani Immonen, Esa Salmikangas

Assigned by

Lentola Logistics Oy

Abstract

The goal of the thesis was to get familiar with development and testing of ArduPilot – open-source autopilot. Another goal of the thesis was to figure out why the client of the thesis had issues with ArduPilot and how those could be solved. The problems faced by the client affected the aircraft while it was transitioning from one flight mode to another. ArduPilot is one of the world's most popular open-source autopilots, which is mainly used to control multirotors and rc-planes.

Client of the thesis was a private company called Lentola Logistics Oy. Goals of the company are to develop electrical aircraft to different types of logistical purposes and to accelerate usage of sustainable transportation methods.

The study of the thesis was focused towards use of ArduPilot in planes, though ArduPilot also supports multiple other aircraft and vehicle types. The thesis was executed by studying ArduPilot's features from its documentation and by analyzing the source-code of the project; hence modifying the source-code and by verifying the changes in a simulator.

The changes made into ArduPilot's source code were documented as a part of the thesis, as were the methods and strategies used to develop and test the source-code. This was performed in the hopes of new ArduPilot developers finding the documentation useful when starting to develop ArduPilot's features even further.

The goals of the thesis were achieved in a highly satisfactory manner and as the outcome of the thesis causes to the problems noticed by the client were solved and fixed by making modifications to the source-code of ArduPilot.

Keywords/tags (subjects)

ArduPilot, autopilot, drone, rc-plane, open-source, simulator, C++, software development

Miscellaneous (Confidential information)

Sisältö

Kuviot

Taulukot

Sanasto

C++

C++ on Bjarne Stroutstrupin kehittämä monipuolinen ja yleiskäyttöinen ohjelmointikieli.

EEPROM

Electrically Erasable Programmable Read-Only Memory.

GCS

Ground Control Station.

GDB

GDB eli GNU Project Debugger on debuggeri Unix pohjaisille käyttöjärjestelmille.

Git

Git on ilmainen avoimen lähdekoodin versionhallintajärjestelmä.

PWM

Pulse Width Modulation.

QuadPlane

ArduPilotin tukema konetyyppi, joka voi lentää kuten multikopteri tai lentokone.

Sakkaus

Sakkaus on ilmiö, jossa siipi menettää suuren osan nostovoimastaan. Yleinen syy lentokoneen siiven sakkaukselle lennon aikana on liian suuri kohtauskulma

SITL

SITL eli Simulation In The Loop on ohjelma, jolla voidaan suorittaa ArduPilotin lähdekoodia ilman siihen tarkoitettua laitteistoa.

VTOL

VTOL on adjektiivi, jota käytetään kuvailemaan ilma-aluksia, jotka kykenevät laskeutumaan ja lähtemään lentoon paikaltaan. Lyhenne tulee sanoista “vertical takeoff and landing”.

1 Johdanto

Autopilotteja voidaan nykyään havaita useissa erilaisissa aluksissa ja ajoneuvoissa. Perinteisemmin lentokoneissa käytetyt autopilotit ovat tulleet osaksi arkea myös harrastekäyttöön soveltuvissa etäohjatuissa ilma-aluksissa. Autopilotin avulla lentämisestä on tehty entistä helpompaa, automaattisempaa ja kuluttajaläheisempää. Autopilottien yleistymisen ja kehittymisen myötä pienet etäohjattavat ilma-alukset ovat yleistyneet sekä kuluttajien että yritysten käytössä ja uusia käyttökohteita pyritään jatkuvasti kehittämään lisää (Drone technology uses and… 2021).

Opinnäytetyön toimeksiantajana toimi vuonna 2017 perustettu hämeenlinnalainen yritys Lentola Logistics Oy, jonka päämääränä on kehittää sähkökäyttöisiä ilma-aluksia erilaisiin kuljetustehtäviin sekä kiihdyttää kestävien kuljetusmenetelmien kehitystä (Lentola Logistics Oy n.d). Tarve opinnäytetyölle syntyi, kun opinnäytetyön toimeksiantaja oli havainnut kehittämässään lennokissaan ongelmia, joiden uskottiin johtuvan siinä käytetyn autopilotin ohjelmistosta. Autopilotti, jota Lentola Logistics koneessaan käyttää, on avoimeen lähdekoodiin perustuva autopilotti nimeltään ArduPilot. Autopilotissa havaitut ongelmat vaikuttivat tietyn lennokkityypin suorittaessa eräänlaista siirtymää toiseen lentotilaan ja pahimmassa tapauksessa ongelmat saattoivat johtaa koneen maahan törmäykseen. Opinnäytetyön toimeksiantajalla oli täten tarve saada lisää tietoa, kuinka koneessa oleva ArduPilot-autopilotti toimi ja kuinka siinä havaitut ongelmat olisi mahdollista korjata.

Opinnäytetyön tavoitteena täten oli perehtyä avoimen lähdekoodin autopilotin, Ardupilotin kehitykseen ja testaukseen sekä kerätä tietoa ArduPilotista siltä kannalta, että siitä olisi mahdollisimman paljon hyötyä toimeksiantajan ilma-aluksessa havaittujen ongelmien ratkaisemiseksi ja jotta ArduPilotin jatkokehittäminen ja testaaminen olisi tätä opinnäytetyötä lukeneille helpompaa. Toimeksiantajan kannalta hyödyllisimpänä tavoitteena pidettiin kohdattujen ongelmien varsinaista ratkaisemista.

2 Yleistä

2.1 Havaitut ongelmat

Autopilotin kehityksessä ja testauksessa keskityttiin kahteen toimeksiantajan havaitsemaan ongelmaan. Ongelmat tulivat esiin eräässä ArduPilotin tukemassa tailsitter-tyyppisessä koneessa, jossa oli kaksi kääntyvää moottoria (ks. kuvio 1). Kappaleessa 3.3.2 käsitellään erilaisia ArduPilotin tukemia lennokkityyppejä tarkemmin. Lyhyesti mainittakoon, että tailsitter-tyyppiset koneet voivat lentää aivan kuten tavallisetkin lentokoneet, mutta sen lisäksi ne pystyvät myös suorittamaan lentoonlähdön ja laskeutumisen paikallaan eli ne ovat VTOL-kykeneviä ilma-aluksia.

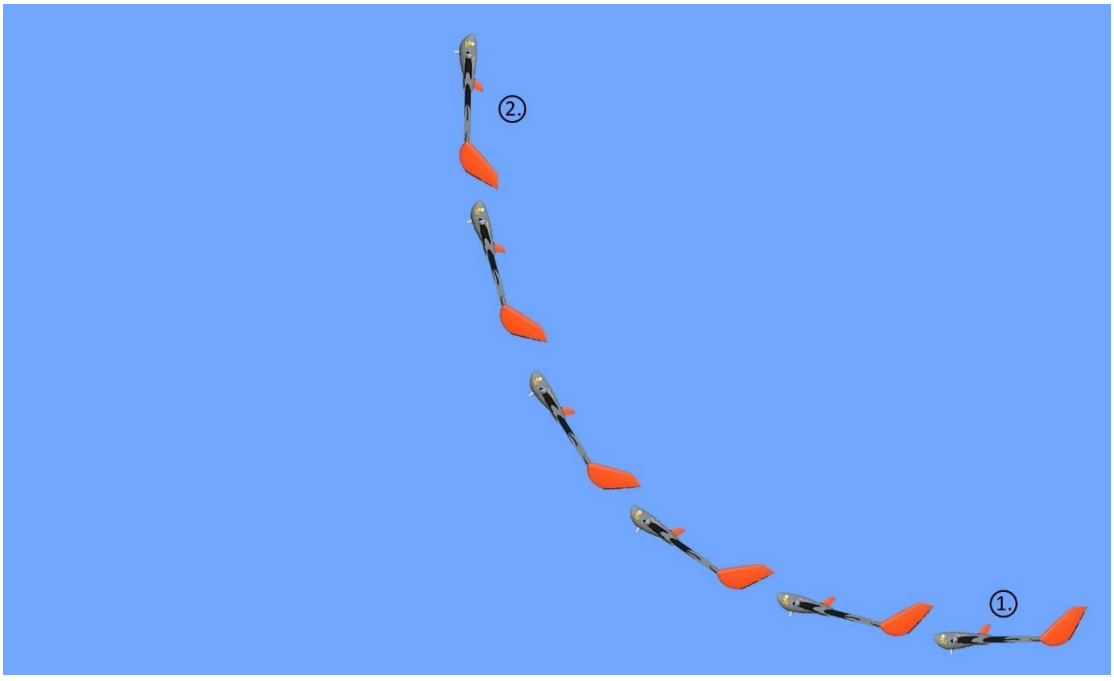
Kuvio 1. Opinnäytetyön toimeksiantajan tailsitter-tyyppiset koneet maassa (Lentola Logistics Oy n.d.)

Taulukossa 1 on nähtävissä yleisiä tietoja kuviossa 1 näkyvistä koneista.

Taulukko 1. Yleisiä tietoja toimeksiantajan koneesta

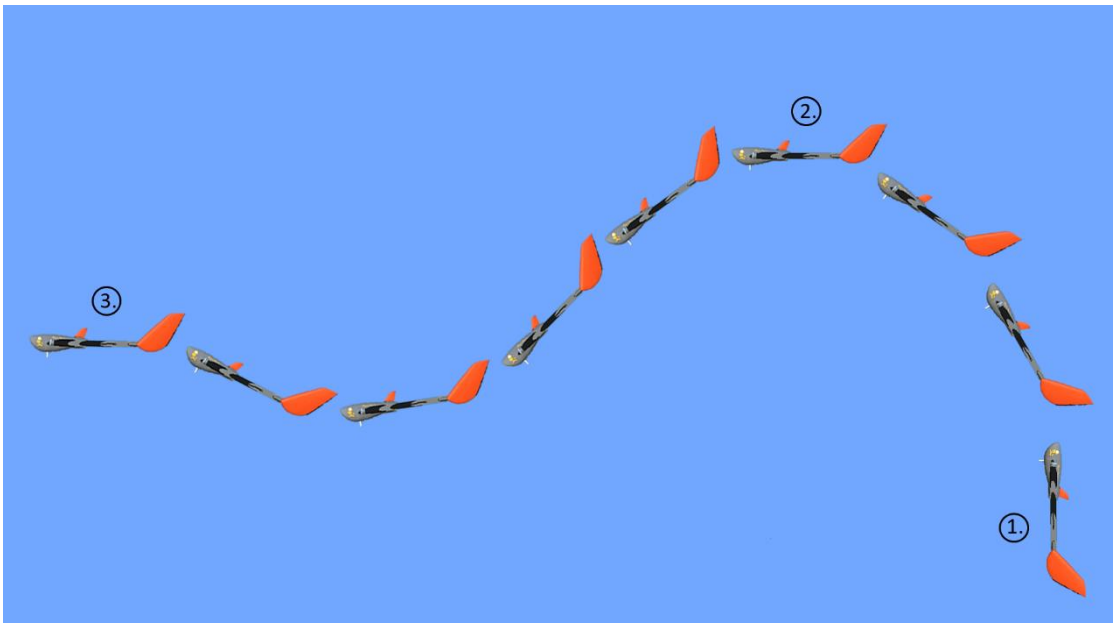
Pituus	Siipien kärkiväli	Paino (lentoonlähtö)
1 metri	2 metriä	12,5 kg

Ensimmäinen ja isoin havaittu ongelma tapahtui koneen siirtyessä tavallisesta lentotilasta VTOL-lentotilaan, jossa kone leijuu paikallaan nokka osoittaen kohti taivasta. Kuviossa 2 koneen ollessa kohdassa 1 siirtymä on alkamassa ja kohdassa 2 kone on siirtynyt VTOL-lentotilaan. Siirtymän päättyessä pystyttiin havaitsemaan, että autopilotti saattoi ohjata konetta liian aggressiivisin liikkein tai vähentää korkeutta liian nopeasti, mikä aiheutti pahimmassa tapauksessa koneen hallinnan menetyksen ja maahan törmäyksen. Tarkkaa syytä ongelmille ei vielä tässä vaiheessa kuitenkaan tiedetty.



Kuvio 2. Tailsitter-koneen siirtymä normaalista lentotilasta VTOL-tilaan

Toinen ongelma tapahtui, kun lentotilaa vaihdettiin VTOL-tilasta tavalliseen lentotilaan (ks. kuvio 3, kohta 1). Kun kone oli suorittanut siirtymän VTOL-tilasta tavalliseen lentotilaan, koneen nokka alkoi laskea alaspäin (ks. kuvio 3, kohta 2) vaikka sen ei odotettu tekevän niin. Hetken aikaa koneen lennettyä loivalla kulmalla kohti maata, koneen nokka alkoi nousta ylöspäin ja koneen päästessä takaisin tavoitekorkeuteensa, normaali lento jatkui (ks. kuvio 3, kohta 3). Kone siis menetti hetken aikaa korkeuttaan siirtymän päätteeksi, vaikka toivottava tilanne olisi se, että kone jatkaa lentoaan samalla korkeudella tai ennemminkin lähtee kasvattamaan sitä.



Kuvio 3. Tailsitter-koneen siirtymä VTOL-tilasta normaaliin lentotilaan

2.2 Miehittämättömät ilma-alukset

Yksinkertaistettuna miehittämätön ilma-alus on ilma-alus, jonka ihmishenkilöstö on korvattu tietokoneilla ja etäyhteyksillä (Austin 2010, 1). Austin (2010) kuitenkin muistuttaa, että miehittämättömiä ilma-aluksia ei tule kuitenkaan sekoittaa yksinkertaisiin lennokeihin tai droneihin. Miehittämättömän ilma-aluksen yksinkertaisista lennokeista ja droneista erottaa, sen ”automaattinen älykkyytensä”. Miehittämätön ilma-alus voi keskustella sen ohjaajan kanssa välittämällä reaaliaikaista ilmakuvaa sekä tiedon aluksen sijainnista, nopeudesta, suunnasta ja korkeudesta. Alus lähettää myös muuta telemetriikkatietoa, kuten polttoaineen määrän ja erilaisten komponenttien lämpötilat. Alus saattaa myös olla suunniteltu siten, että se kykenee tunnistamaan vikatilanteita sen laitteissa ja rakenteissa ja tekemään myös korjaavia liikkeitä vikojen korjaamiseksi ja riskien minimoimiseksi. (Austin 2010, 3.)

Miehittämättömät ilma-alukset ovat saaneet alkunsa sotilaspuolelta kuten monet muutkin innovaatiot, mutta niiden käyttö myös siviilipuolella on alkanut yleistymään viime vuosien aikana (Austin 2010, 3). Miehittämättömiä ilma-aluksia voidaan käyttää useisiin eri tarkoituksiin ja niiden eri käyttökohteet jatkavat kasvuaan uusien teknologioiden kehittyessä. Tyypillisiä käyttötarkoituksia miehittämättömille ilma-aluksille ovat mm. tiedustelu, ilmakuvaus, valvonta ja logistiset tarpeet (Austin 2010, 1-2).

Yksi suuri etu miehittämättömän aluksen käyttöön on sen taloudellisuus. Miehittämättömät ilma-alukset ovat tyypillisesti miehitettyyn ilma-alukseen verrattuna huomattavasti halvempia valmistaa ja operoida. Miehittämättömän ilma-aluksen ohjaus tapahtuu osittain maasta ja osittain aluksen autopilotin toimesta. Aluksen valmistamisessa voidaan säästää tilaa sekä kuluja, sillä jotkin tavallisesti lentäjälle tärkeät laitteet ja osat ilma-aluksessa ovat täysin turhia, jos konetta operoidaan maasta käsin. Esimerkiksi koneen penkit, ohjainlaitteet ja mittaristot voidaan eliminoida kokonaan miehittämättömästä aluksesta ja korvata kameralla sekä etäohjauksella, jonka seurauksena aluksen paino vähenee ja operointikulut laskevat. Miehittämättömät ilma-alukset voivat myös olla kooltaan huomattavasti pienempiä miehitettyihin aluksiin verrattuna. (Austin 2010, 5-8.)

2.3 Multikopterit

Multikoptereissa, josta tavanomaisemmin käytetään termiä drone, ovat vähintään kaksiroottorisia ilma-aluksia (ks. kuvio 4). Termiä ”drone” voidaan myös käyttää muun tyyppisistä miehittämättömistä ilma-aluksista, mutta tyypillisesti puhuttaessa droneista, ne mielletään multikoptereiksi. Yleisimmät multikopterit ovat tyypiltään nelikoptereita, eli niissä on neljä roottoria. Nelikopteri on tyypillisesti yksinkertaisin ja halvin multikopterityyppi valmistaa. Lisäämällä roottoreita, voidaan multikopterin toimintavarmuutta sekä hyötykuorman kantokykyä parantaa. Useampi roottori kuitenkin lisää aluksen painoa ja virrankulutusta, joten suuri määrä roottoreita ei aina ole välttämättä paras vaihtoehto multikopteria hankkiessa. (Multirotor drones n.d.)

Kuvio 4. Kuusiroottorinen multikopteri eli heksakopteri

Multikoptereiden ohjaus tapahtuu pääsääntöisesti radio-ohjaimen välityksellä, mutta niitä voidaan myös ohjata esimerkiksi puhelimen tai tabletin avulla, tai ne voivat lentää jotain ennalta määritettyä reittiä.

Nostovoimaa multikopterit tuottavat pelkästään roottoriensa avulla. Multikopterit voivat leijua paikallaan ilmassa, aivan kuten helikopteritkin, eivätkä ne tarvitse ilmanopeutta pysyäkseen ilmassa kuten lentokoneet. Tuotetun nostovoiman määrää multikopterissa voidaan säädellä säätämällä sen roottoreiden kierrosnopeutta. Roottorien kierrosnopeutta lisäämällä alus tuottaa enemmän nostovoimaa ja pyrkii nousemaan ylöspäin. Multikopteria voidaan liikuttaa kallistamalla sitä haluttuun suuntaan. Lisäämällä roottorien kierrosnopeutta vastakkaisella puolella kuin mihin kopteria halutaan kallistaa ja vähentämällä kierrosnopeutta toisella puolella, multikopteri saadaan kallistumaan haluttuun suuntaan. (Multirotor drones n.d.)

Ilmailun maailmassa multikopterit ovat vielä suhteellisen uusia ilma-aluksia. Multikopterit ovat luonnollisesti erittäin epävakaita lentää ja käytännössä vaativat jonkinnäköisen keskusyksikön, joka pystyy vakauttamaan aluksen lentoa. Valmiit kaupasta saatavat multikopterit ovat kuitenkin pääsääntöisesti helppoja lentää kokemattomillekin lennättäjille. Kaupoista saatavissa multikoptereissa on nykyisin jo kehittyneet keskusyksiköt. Ne pysyvät ilmassa paikallaan tuulisellakin säällä, vaikka lennättäjä ei itse edes koskisikaan sen ohjaimeen. Sen lisäksi niistä löytyy useita käyttäjää avustavia ominaisuuksia, kuten esimerkiksi hyvin yleinen ”palaa kotiin” -tila, jonka aktivoituessa alus lentää täysin itsenäisesti takaisin lähtöpisteeseensä. Joissain malleissa on myös sensoreita, joilla voidaan tunnistaa mahdollisia esteitä ja välttää aluksen törmäys niihin.

Multikoptereiden käyttö useilla eri aloilla on yleistynyt nopeasti viimeisen muutaman vuoden aikana. Niiden edullisuus, saatavuus ja helppo käytettävyys ovat tehneet multikoptereista hyvinkin suosittuja yksityisessä, kaupallisessa sekä sotilaallisessa käytössä ja uusia käyttökohteita kehitetään jatkuvasti lisää. Etenkin multikoptereiden kaupallinen käyttö on kiihdyttämässä tahtiaan uusien käyttökohteiden suhteen. Multikopterimarkkinoiden uskotaan kaupallisessa käytössä kasvavan vuoteen 2025 mennessä 63,6 miljardiin dollariin, verrattuna vuoden 2018 markkinoihin, joka oli arvoltaan 4,4 miljardia dollaria. (Drone technology uses and… 2021.)

Multikopterit kykenevät lähtemään lentoon ja laskeutumaan täysin paikaltaan, joka tarkoittaa, että niillä voidaan operoida huomattavasti laajemmassa ympäristössä

kuin lennokilla, joka vaatii tilaa, sekä tasaisen maaston lentoonlähtöön ja laskeutumiseen. Multikoptereiden kyky leijua paikallaan, tekee siitä ihanteellisen ilmakuvaukseen ja valvontaan, mutta pitkiä lentomatkoja suunnitellessa on paras turvautua lennokin käyttöön. (Multirotor drones n.d.)

2.4 Autopilotti

Ilmailussa autopilotilla tarkoitetaan järjestelmää, minkä tehtävänä on ohjata ilma-alusta lentäjän puolesta. Autopilottia voidaan kuitenkin käyttää myös useissa muissa ajoneuvoissa ja aluksissa.

Autopilotin tehtävänä miehitetyissä ilma-aluksissa on tyypillisesti vähentää lentäjän työmäärää avustamalla lentäjää ilma-aluksen ohjauksessa, jonka seurauksena lentäjän tekemien virheiden määrä lennon aikana vähenee ja lentäjälle jää lisää aikaa muiden tehtävien suorittamiseen etenkin pitkien lentojen aikana. Autopilotin suorittamat toiminnot ovat pitkälti ennalta ohjelmoituja ja lentäjän määrittämiä eivätkä ne vaadi vaikeiden päätösten tekemistä. Ilma-aluksen ohjaukseen autopilotti käyttää apunaan aluksen sensoreilta tulevaa tietoa kuten nopeutta, korkeutta, sijaintia, suuntaa sekä kallistus- ja nousukulmaa. Autopilotille tyypillisiä tehtäviä ovat pitää koneen korkeus, nopeus ja suunta samana tai koneen lentäminen kohti jotain tiettyä pistettä. Edistyneimmät autopilotit voivat mm. myös laskeutua ja lähteä lentoon täysin itsenäisesti. (Yunyi, Longxiang & Xin 2018.)

Miehittämättömissä ilma-aluksissa autopilotti on käytännössä välttämätön. On mahdollista, ettei alukseen saada yhteyttä ja sen tulee kyetä itseohjautumaan. Pääsääntöisesti koneen lentäminen miehittämättömissä ilma-aluksissa tapahtuu autopilotin toimesta. Miehittämättömissä ilma-aluksissa autopilotti kykenee tyypillisesti suorittamaan täysin autonomisia tehtäviä, joihin voi kuulua esimerkiksi pakettien jakaminen, alueiden kartoitus, valvonta. (UAV Autopilot systems n.d.)

2.5 Avoin lähdekoodi

Avoimen lähdekoodin ohjelmisto on ohjelmisto, jonka lähdekoodia kuka tahansa voi tarkastella, muokata ja parannella (What is Open Source n.d).

Lähdekoodi itsessään on ohjelmistoissa yleensä se osa, jota tavalliset käyttäjät eivät koskaan näe. Lähdekoodi on sitä koodia, jota ohjelmoijat käsittelevät ja muuntelevat saadakseen ohjelmiston toimimaan haluamallaan tavallaan. Ohjelmoijat, joilla on pääsy jonkin ohjelmiston lähdekoodiin, voivat parannella ohjelmistoa esimerkiksi lisäämällä siihen uusia ominaisuuksia tai korjaamalla bugeja eli asioita, jotka eivät toimi halutulla tavalla. (What is Open Source n.d.)

Läheskään kaikkien ohjelmistojen lähdekoodi ei ole avointa. Jotkut ohjelmistot on kehitetty siten, että vain sen kehittäjillä tai omistajilla on oikeus käsitellä lähdekoodia. Lähdekoodin hallinnoijat saattavat olla yksittäisiä henkilöitä, tiimejä tai kokonaisia organisaatioita. Tämäntyyppistä lähdekoodia kutsutaan suljetuksi lähdekoodiksi. (What is Open Source n.d.)

Avoimelle sekä suljetulle lähdekoodille yhteistä on se, että niiden käyttäjien tulee hyväksyä lähdekoodiin liitetyn lisenssin määrittämät ehdot käyttääkseen sitä. Lailliset ehdot avoimen ja suljetun lähdekoodin lisensseissä kuitenkin eroavat toisistaan merkittävästi. (What is Open Source n.d.)

Vaikka useat avoimen lähdekoodin lisenssit sallivat niihin liitetyn lähdekoodin täysin vapaan käytön, on kuitenkin mahdollista, että joissain lisensseissä ehdot ovat tiukemmat. On mahdollista, että lisenssi esimerkiksi velvoittaa muokatun lähdekoodin pohjalta rakennetun ohjelmiston lähdekoodin jakamista ohjelmiston yhteydessä; lähdekoodia ei saa siis pitää salassa. (What is Open Source n.d.)

Käsite ”avoin” saattaa antaa käsityksen siitä, että avoimeen lähdekoodiin perustuvat ohjelmistot olisivat aina ilmaisia, mutta näin ei aina ole, vaan myös avoimen lähde-

koodin ohjelmistokehittäjät voivat veloittaa rahaa kehittämistään avoimen lähdekoodin ratkaisuista. Yksi yleinen malli tehdä rahaa avoimella lähdekoodilla, on myydä sen käyttäjille tukea ja palveluita sen käyttöön. (What is Open Source n.d.)

3 ArduPilot

3.1 Yleistä

ArduPilot on avoimen lähdekoodin autopilotti, jota voidaan käyttää useiden eri alustyyppien autonomiseen ohjaukseen. ArduPilot on useiden vuosien kehitystyön tulos, joka jatkuvasti kehittyy eteenpäin useiden eri kehittäjien toimesta. ArduPilotin esittelysivulla projektia kuvaillaan kehittyneimpänä, ominaisuuksiltaan laajimpana ja luotettavimpana avoimen lähdekoodin autopilottina, joka on tällä hetkellä saatavilla. (About ArduPilot n.d.)

ArduPilot on saanut alkunsa vain yhden miehen harrasteprojektina, mutta vuosien mittaan se on kehittynyt useiden eri organisaatioiden, yritysten ja yksityishenkilöiden tukemaksi ja luottamaksi monipuoliseksi autopilotiksi. Ensimmäinen ArduPilotin versio julkaistiin vuonna 2009 Googlen Code palvelussa. (ArduPilot partners n.d.)

ArduPilot on siitä harvinainen autopilotti, että se kykenee ohjaamaan lähes kaikentyyppisiä aluksia. ArduPilotin tukemia alustyyppejä ovat multikopterit, kiinteäsiipiset lentokoneet, helikopterit, maa-ajoneuvot, veneet sekä sukellusveneet. Näiden lisäksi ArduPilotiin kuuluu myös ”Antenna Tracker” niminen ohjelmisto eli suomeksi ”antennin seurain”, jota voidaan käyttää automaattiseen antennin suuntaukseen mahdollisimman hyvän signaalin saamiseksi. Yleisimpinä ArduPilotin käyttökohteina ovat radio-ohjattavat lennokit sekä multikopterit. Oikeissa täysikokoisissa lentokoneissa ArduPilotia tuskin tullaan vielä ihan hetkeen näkemään.

ArduPilot itsessään on vain ohjelmisto, jota jokin laitteisto suorittaa. Jotta ArduPilotia voidaan käyttää autopilottina jossain aluksessa, vaaditaan sen pariksi myös jokin laitteisto. Laitteistoon (engl. hardware) kuuluu keskusyksikkö, joka suorittaa siihen asennettua ohjelmaa eli tässä tapauksessa ArduPilotin koodia. Laitteistoon kuuluu myös sensoreita, jotka välittävät erilaisia tietoja keskusyksikölle sekä laitteita, kuten servoja, jotka liikuttavat lennokin ohjainpintoja tai moottorien nopeudensäätimiä, jotka ohjaavat aluksen sähkömoottoreiden kierrosnopeutta.

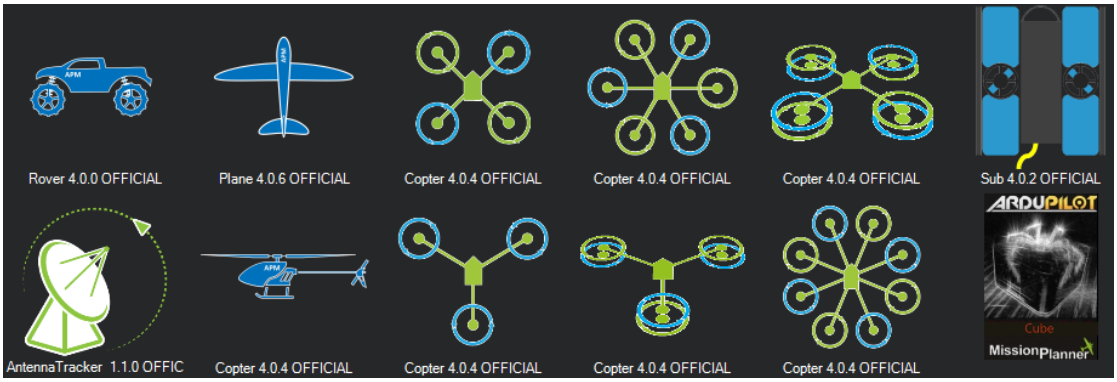
ArduPilotin konfigurointi ja ohjaus tapahtuu pääsääntöisesti käyttämällä jotain maa-asemaa. Mission Planner on Michael Obornen kehittämä avoimen lähdekoodin sovellus, joka toimii ArduPilotin alakategorioihin kuuluvien ArduPlanen, ArduCopterin ja ArduRoverin ns. maa-asemana (Mission Planner n.d). ArduPilotille on saatavilla muitakin maa-asemia, mutta Mission Planner on yksi monipuolisimmista maa-asemista ArduPilotin käyttöön.

Mission Plannerilla voidaan mm. suunnitella ja tehdä tehtäviä aluksille, tarkkailla lukuisia eri suoritusarvoja, säätää autopilotin asetuksia ja analysoida lokitiedostoja (ks. kuvio 5). Sovellus toimii ainoastaan Windows käyttöjärjestelmillä. (Mission Planner n.d.)

Kuvio 5. Yleisnäkymä Mission Planner maa-aseman käyttöliittymästä

3.2 Tuetut alustyypit

ArduPilot tukee useita eri alustyyppejä (ks. kuvio 6) ja kaikkia alustyyppejä tuetaan samantyyppisillä perusominaisuuksilla. Alukset itsessään kuitenkin eriävät toisistaan huomattavasti, joten ArduPilotin ohjelmisto on jaettu useaan eri versioon alusten kesken. Tämän opinnäytetyön sisältö keskittyy pääsääntöisesti ArduPilotin käyttöön lennokeissa.



Kuvio 6. Erilaisia aluskonfiguraatioita havainnollistettu Mission Plannerin käyttöliittymässä

3.3 ArduPlane

ArduPlane on kiinteäsiipisten ilma-alusten ohjaukseen tarkoitettu autopilotti. ArduPlane laiteohjelmisto kykenee täysin autonomiseen kiinteäsiipisen ilma-aluksen ohjaukseen. (ArduPlane n.d.)

ArduPlane autopilotti soveltuu kaikkien kiinteäsiipisten lentokoneiden autonomiseen ohjaukseen. ArduPlane tukee myös kiinteäsiipisiä VTOL-lentokoneita, jotka voivat lentää samalla tapaa kuin tavallinenkin lentokone, mutta ne kykenevät myös laskeutumaan ja lähtemään lentoon täysin paikaltaan.

3.3.1 Yleiset ominaisuudet

Automaattinen lentoonlähtö

ArduPlane tukee useita automaattisia lentoonlähtötapoja kiinteäsiipisille lentokoneille. Automaattista lentoonlähtöä käyttäessä käyttäjän tulee määrittää kulma, jossa lentokone suorittaa lentoonlähdön sekä korkeus, joka koneen tulee saavuttaa ennen kuin kone vaihtaa lentoonlähtötilasta normaaliin lentotilaan. ArduPlanen dokumentaatio suosittelee 10–15 asteen kulmaa lentoonlähtöä varten. (Automatic takeoff n.d.)

Lentoonlähtökorkeus tulee asettaa riittävän korkeaksi, jotta kone ei törmää esimerkiksi puihin sen jälkeen, kun lentoonlähtö on suoritettu. VTOL-lentokoneet suorittavat lentoonlähdön paikaltaan, joten niissä lentoonlähtökulmaa ei määritetä erikseen. ArduPlanen dokumentaatiossa lentoonlähtökorkeudeksi suositellaan 40 metriä, mutta tämä käyttäjän tulee kuitenkin päätellä aina itse lennätyspaikan perusteella. Lentoonlähdössä kone pyrkii pitämään suuntansa koko ajan samana. (Automatic takeoff n.d.)

Perinteisille ei-VTOL-koneille ArduPlane tukee neljää eri tapaa koneen lentoon lähetykselle. Perinteinen tapa kevyiden lennokeiden lentoon lähetykselle on yksinkertaisesti heittää kone vaakatasossa riittävällä nopeudella ilmaan. Konetta heittäessä ArduPilot aloittaa lentoon lähdön, kun koneen kiihtyvyys eteenpäin saavuttaa käyttäjän määrittämän kiihtyvyyden. Käyttäjä voi myös määrittää aikaviiveen moottorin käynnistykselle sen jälkeen, kun riittävän suuri kiihtyvyys on saavutettu. Viiveellä pyritään välttämään pyörivän potkurin osuminen heittäjän käteen. Mikäli GPS-signaali on saatavilla, ArduPlane voi myös käyttää koneen mittaamaa maanopeutta ylimääräisenä turvakeinona määrittääkseen onko kone heitetty ilmaan ja onko moottorin käynnistys turvallista. GPS:ää käytettäessä minimi maanopeudeksi ArduPlanen dokumentaatio suosittelee 4 m/s. (Automatic takeoff n.d.)

Toinen tapa koneen laukaisuun on käyttää katapulttia, jolla kone kiihdytetään ilmaan (ks. kuvio 7). Edellisessä kohdassa mainitut turvamekanismit pätevät myös tähän lä-

hetystapaan. Katapultti on hyvä vaihtoehto koneen laukaisuun, sillä se kykenee kiihdyttämään koneen nopeasti riittävään lentonopeuteen, eikä samaa riskiä potkurin osumiselle ihmisiin ole kuin konetta kädestä heittäessä. Jotkut koneet saattavat myös olla liian isoja heitettäviksi, eikä kiitotien käyttökään ole välttämättä mahdollista, joten katapultilla laukaisu saattaa olla ainut vaihtoehto. Voimanlähteenä katapultit voivat käyttää mm. kaasua, sähkömagneetteja tai jopa ruutia.

Kuvio 7. Suomen maavoimien Orbiter 2b tiedustelulennokki katapultissaan (Minilennokkijärjestelmä n.d.)

Kolmantena vaihtoehtona on käyttää katapulttia yksinkertaisempaa linkoa, jolla lentokone kiihdytetään ilmaan kuminauhan voimalla. Linko toimii hyvin samantapaisesti kuin katapultti.

Neljäntenä vaihtoehtona on käyttää kiitotietä koneen lähetykseen. Kiitotieltä lentoonlähtö on autopilotille kaikista hankalin vaihtoehto, koska koneen ohjaus maassa vaatii paljon tarkkuutta, jotta koneen suunta saadaan pidettyä täysin samana, eikä kone suistu ulos kiitotieltä. Koneen säätäminen kiitotieltä lähtöihin on erittäin tarkkaa ja työlästä. Tämän lisäksi koneeseen tarvitaan elektroninen kompassi, jotta autopilotti pystyy pitämään suunnan täysin samana koko lentoonlähdön ajan. Lentoon-

lähdön haastavuuteen vaikuttaa myös se, onko kone nokkapyörällinen vai kannuspyörällinen. Kannuspyörällinen kone vaatii nokkapyörällistä konetta enemmän säätöä sujuvan lentoonlähdön aikaansaamiseksi. (Automatic takeoff n.d.)

Automaattinen laskeutuminen

ArduPlane voi laskeutua automaattisesti lentokoneella, mikäli laskeutuminen on asetettu osaksi lentotehtävää. Kone suorittaa laskeutumisen siten, että kone osuu maahan laskeutumiseen määritetyssä pisteessä, mutta ei siis täten ole vielä välttämättä täysin pysähtynyt tässä pisteessä.

Laskeutumispiste koostuu neljästä parametristä: pituus- ja leveysasteesta, korkeudesta jolle kone pyrkii laskeutumaan sekä korkeudesta jolle kone kiipeää, jos laskeutuminen keskeytetään. Laskeutumiskorkeus asetetaan nollaksi, mikäli kone laskeutuu lähtöpisteeseen. Tätä pistettä kutsutaan myös kodiksi (engl. home). (Automatic landing n.d.)

Suuntaa laskeutumiselle ei määritetä erikseen, vaan se on vektori, joka alkaa laskeutumispistettä edeltävästä tehtävän pisteestä ja loppuu laskeutumispisteeseen. Lähestymiskulma on riippuvainen näiden kahden pisteen etäisyyksistä. Mikäli pisteiden etäisyys on matala korkeuseroon suhteutettuna, saattaa kone päätyä suorittamaan laskeutumisen liian suurella kulmalla, mikä voi johtaa maahan törmäykseen. Tehtävää suunnitellessa on tärkeää varata riittävä etäisyys ja oikea suunta koneen laskeutumista varten. Laskeutuminen on kuitenkin mahdollista keskeyttää esimerkiksi lähettämällä maa-asemasta laskeutumisen keskeytyskäsky asettamalla radio-ohjaimesta tehovipu yli 90 % teholle tai vaihtamalla koneen lentotilaa. (Automatic landing n.d.)

Lisäksi laskeutumista avustetaan asettamalle koneelle ns. leimahduspiste (engl. flare point), jolle määritetään kaksi parametriä. Ensimmäinen parametri kertoo, kuinka monta sekuntia koneella saa olla aikaa maahan törmäykseen, mikäli se jatkaa korkeuden menettämistä samalla nopeudella. (Automatic landing n.d.)

Esimerkiksi koneen korkeuden muutoksen ollessa -2 m/s ja korkeuden ollessa 10 metriä, koneella on viisi sekuntia aikaa törmäykseen. Jos parametriksi on asetettu esimerkiksi 2 sekuntia, kone saavuttaa leimahduspisteen ollessaan 4 metrin korkeudessa. Toisena parametrinä on minimikorkeus, jonka kone saa saavuttaa ennen leimahduspisteen aktivointia. Koneen saavuttaessa leimahduspisteen eli kumman tahansa raja-arvon, moottorin teho tiputetaan, ja konetta ohjataan ylöspäin, jotta korkeuden muutosnopeus pienenee. (Automatic landing n.d.)

Oletuksena saavuttaessaan leimahduspisteen, autopilotti pyrkii ohjaamaan konetta niin, että korkeuden muutosnopeus tasaantuu 0,25 metriin sekunnissa. Arvoa on myös mahdollista vaihtaa. Koneen suurimmaksi kallistuskulmaksi leimahduspisteen aktivoituessa autopilotti asettaa oletuksena 5 astetta, jotta koneen siiven kärjen eivät osuisi maahan lähestymistä suorittaessaan. Leimahduspisteelle ensimmäiseen parametriin ArduPlanen dokumentaatio suosittelee arvoksi 1,5 sekuntia ja toiseen parametriin 2 metriä. Jokaiselle eri koneelle on toki omat parhaat arvonsa ja muita vaikuttavia tekijöitä ovat esimerkiksi, se minkälaisia antureita koneeseen on asennettu korkeuden määrittämiseksi.

Koneet, jotka käyttävät jonkinlaista sensoria etäisyyden mittaamiseksi (esimerkiksi LIDAR), kykenevät yleensä tarkkoihin ja sitä myötä matalampiin laskeutumisen aloituskorkeuksiin, kuin ne koneet, joissa korkeutta mitataan pelkästään ilmanpaineen perusteella. (Automatic landing n.d.)

Koneen edetessä liian kauas laskeutumispisteestä, voidaan ArduPilotin asetuksiin asettaa parametri, jonka avulla kone saadaan laskeutumaan laskemalla koneen nokkaa entisestään tai sitten pakottamalla kone sakkaamaan, kasvattamalla koneen kohtauskulmaa. Kulmaa kasvatetaan laskeutumispisteeseen relatiivisesti, kunnes se saavuttaa maksimin. Eli mitä pitemmälle kone jatkaa lentoaan laskeutumispisteestään, sitä suuremmaksi kulmaa muutetaan, jotta kone saadaan alas. (Automatic landing n.d.)

Väärinpäin lentäminen

ArduPilot kykenee ohjaamaan lentokonetta myös ylösalaisin, olettaen, että se ylipäätään on koneen rakenteen kannalta mahdollista. QuadPlane-tyyppisiä koneita ei tueta tässä lentotilassa. ArduPlanen dokumentaatiossa ei erikseen mainita mitä käyttötarkoituksia tällä ominaisuudella on, mutta lentotilaa suositellaan vaihdettavaksi takaisin normaaliin ennen laskeutumista, tai muuten autopilotti myös laskeutuu koneella ylösalaisin (Automatic landing n.d).

Sakkauksen esto

Automaattinen sakkauksen esto on tärkeä ArduPlanen ominaisuus. Sakkauksen esto on kuitenkin melko yksinkertainen määrittää. ArduPilotin asetuksiin yksinkertaisesti määritetään pienin ilmanopeus vaakalennossa, jolla kone kykenee lentämään. Sakkauksen esto voidaan myös poistaa käytöstä ArduPilotin asetuksista. Ominaisuus on yleensä hyvä pitää päällä. Kuitenkin jos koneen ilmanopeutta ei ole mahdollista määrittää luotettavasti, saattaa käyttäjän olla parempi jättää ominaisuus pois päältä, jotta kone ei käyttäydy odottamattomasti ilmassa.

Ilmanopeuteen, jolla kone voi lentää ilmassa, vaikuttaa mm. koneen sen hetkinen kallistuskulma. Kallistuskulman kasvaessa, tulee myös lentokoneen ilmanopeuden kasvaa, jotta kone säilyttää saman nostovoiman. Kun sakkauksen esto on päällä, ArduPilot tarkkailee koneen kallistuskulmaa sekä ilmanopeutta. (Stall prevention n.d.)

Sakkauksen eston ollessa käytössä, autopilotti siis tarkkailee kallistuskulmaa sekä ilmanopeutta ja päättelee tämän perusteella, onko kone lähellä sakkaamista. Ilmanopeuden ollessa liian pieni, autopilotti rajoittaa koneen kallistuskulmaa, mutta kuitenkin niin, että 25 asteen kallistuskulma on sallittu millä tahansa nopeudella, jotta kone säilyttää jonkinlaisen liikehtimiskyvyn. Mikäli lentotila on sellainen, jossa autopilotti saa säätää moottorin tehoa itse, autopilotti myös pyrkii joko nostamaan moottoreiden tehoa tai laskemaan koneen nokkaa, jotta kone saavuttaisi sakkausnopeutta suuremman ilmanopeuden. (Stall prevention n.d.)

3.3.2 Eri konetyypit

QuadPlane

Tavallisten kiinteäsiipisten koneiden lisäksi ArduPilot tukee useita erityyppisiä VTOL-lentokoneita, joissa yhdistyvät lentokoneiden ja multikoptereiden hyvät puolet. ArduPlanen dokumentaatiossa kaikki VTOL-kykenevät koneet kuuluvat ”QuadPlane” kategoriaan. Varsinaisia QuadPlane-koneita ovat kuitenkin ne alukset, joissa on 4–8 pystyasennossa olevaa roottoria (ks. kuvio 8), joiden avulla QuadPlane voi lentää kuten multikopteri ja täten suorittaa laskeutumisia ja lentoonlähtöjä pieniltäkin alueilta. QuadPlanen kyky lentää kuten multikopteri tai lentokone tekee siitä erittäin monipuolisen lentoaluksen, joka kykenee huomattavasti pitempiin lentomatkoihin kuin pelkkä tavallinen multikopteri.

Kuvio 8. QuadPlane kahdeksalla pystyasennossa olevalla roottorilla (ArduPlane – QuadPlane overview n.d.)

ArduPlane tukee lukuisia erilaisia QuadPlane-konfiguraatioita, mutta ominaisuuksiltaan ne ovat pitkälti samanlaisia. Monet konfiguraatiot myös muistuttavat toisiaan, eivätkä erot niissä ole niin merkittäviä, että niitä olisi järkevää käydä tässä työssä läpi. Suurimmat erot eri konfiguraatioissa ovat eriävä roottoreiden lukumäärä, moottorien asettelu ja se voiko niiden kulmaa säätää lennon aikana.

Tailsitter

Tailsitter eli vapaasti suomennettuna pyrstöllä istuja, on yksi ArduPlanen tukemista VTOL-konetyypeistä. Tailsittereissä ei ole erikseen korkeusperäsintä ja siivekkeitä, vaan kone on käytännössä pelkkä lentävä siipi, jonka liikehtimistä ohjataan siivekkeiden avulla. Riippuen siitä onko koneessa yksi vai kaksi moottoria, voidaan konetta ohjata muuttamalla myös vasemman ja oikean moottorin tehon suhdetta lennon aikana.

ArduPilot jakaa tailsitter-tyyppiset koneet dokumentaatiossaan kahteen kategoriaan. Ensimmäiseen kategoriaan kuuluvat koneet kykenevät muuttamaan moottoreidensa kulmaa maassa sekä lennon aikana. Näihin konetyyppeihin viitataan ArduPlanen dokumentaatiossa myös nimityksellä ”tilt rotor plane”. Moottoreiden kulmaa säätelemällä lennon aikana autopilotti saa lisää hallintaa koneen ohjaukseen ja kykenee tekemään liikkeitä, joita ei välttämättä pysty tekemään niillä koneilla, joissa moottoreiden kulma on kiinteä. Toiseen kategoriaan putoavatkin ne koneet, joiden moottorit ovat koko ajan samassa kulmassa. Näissä konetyypeissä on tyypillisesti tavallista suuremmat ohjauspinnat riittävän ohjattavuuden aikaansaamiseksi, jotta kone kykenee leijumaan ilmassa hallittavasti. ArduPilot jakaa toiseen kategoriaan kuuluvat konetyypit vielä yhteen alikategoriaan, johon kuuluvat yksi tai kaksimoottoriset koneet sekä ”CopterMotor”-koneet, joissa moottoreita on vähintään kolme. CopterMotor-koneet ovat lentokoneen sekä multikopterin sekoitus, jotka leijuvat paikallaan koneen runko

vaakatasossa, kun taas muut tailsitter-konfiguraatiot leijuvat koneen nokka taivaaseen osoittaen (ks. kuvio 9).

Kuvio 9. Kaksimoottorinen tailsitter-kone leijuntatilassa

Kaikki tailsitter-koneet eivät välttämättä lähde ilmaan pyrstöltään, vaikka nimitys hieman siihen viittaakin. Esimerkiksi ne tailsitter-koneet, joissa moottoreiden kulmaa voidaan säätää, voivat myös lähteä ilmaan mahaltaan (ks. kuvio 10). Kone kääntää lentoonlähdössä moottorit osoittamaan ylöspäin ja nostaa moottoreiden kierrosno-

peutta, joka saa koneen nokan nousemaan kohti taivasta ja siitä ilmaan. Myös CopterMotor konetyypit lähtevät lentoon mahaltaan, mutta ne eivät koskaan nosta koneen nokkaa kohti taivasta ollessaan leijuntatilassa.

Kuvio 10. Tailsitter-kone kääntyvillä moottoreilla lähdössä lentoon mahaltaan

3.3.3 Lentotilat

ArduPlane tukee useita eri lentotiloja. Lentotilat voivat olla täysin manuaalisia, täysin autonomisia tai käyttäjää avustavia. Koneen lentotilaa voidaan vaihtaa joko radio-ohjaimen välityksellä, maa-aseman kautta tai asettamalla lentotilan muutoksen osaksi jotain lentotehtävää.

Kaikki Plane-lentotilat

- **MANUAL**
  Manuaalinen tila. Autopilotti ei avusta koneen ohjauksessa millään tavalla.
- **STABILIZE**
  Konetta voi ohjata lähes manuaalisesti, mutta käyttäjän vapauttaessa ohjauksen, autopilotti pyrkii suoristamaan koneen eli ohjaamalla koneen nousu- ja kallistuskulman nollaan asteeseen.
- **FBWA (Fly By Wire A)**
  Sama kuin STABILIZE, mutta koneen nousu- ja kallistuskulma on rajoitettu käyttäjän asettamiin rajoihin.
- **FBWB (Fly By Wire B)**
  Koneen kääntymistä voi ohjata itse, mutta se on rajoitettu käyttäjän määrittämään kulmaan kuten FBWA tilassa. Autopilotti pyrkii pitämään koneen koko ajan samassa korkeudessa ja ohjauksen vapatuessa kone pyrkii suoristamaan koneen. Koneen ohjaaminen ylös-, tai alaspäin ei tässä tilassa ohjaa korkeusperäsintä, vaan muuttaa koneen tavoitekorkeutta, johon autopilotti koneen ohjaa. Autopilotti säätää myös koneen nopeutta itse, mikäli se on sallittu asetuksissa. Koneelle voidaan määrittää minimi- ja maksiminopeus tätä lentotilaa varten. Kone pyrkii pysymään näiden rajojen sisällä lennättäjän säädellessä moottorien tehoa.
- **CRUISE**
  Sama kuin FBWB, mutta kone pyrkii myös lentämään täysin samaan suuntaan, kuin mihin se on myös ohjattukin. Tässä lentotilassa kone voi myös seurata maastoa ja säädellä minimikorkeutta sen mukaisesti.
- **AUTOTUNE**
  Sama kuin FBWA, mutta autopilotti pyrkii automaattisesti säätämään PID-ohjaimet koneen ohjaukseen.
- **TRAINING**
  Lähes kuin manuaalinen tila, mutta koneen kallistus- ja nousukulma on rajoitettu käyttäjän asettamiin rajoihin. Sakkauksen esto on myös päällä käännöksissä ja se rajoittaa kallistuskulmaa liian pienillä nopeuksilla. Soveltuu hyvin lennokin lennätyksen opetuskäyttöön.
- **ACRO**
  Lähellä manuaalista tilaa, mutta toisin kuin manuaalisessa tilassa, jossa kaikki käskyt menevät läpi raakana koneen servoille ja moottoreille, konetta ohjataan niin, että lennättäjän antama käsky kääntää konetta muunnetaan kulman muutosnopeudeksi, ja konetta kallistetaan sen mukaan. Autopilotti pyrkii myös pitämään koneen kallistus- ja nousukulman täysin samana kuin mihin se ohjattu lennättäjän vapauttaessa koneen ohjauksen.
- **AUTO**

Autonominen tila, jossa kone lentää asetetun tehtävän mukaisesti. Konetta voi myös ohjata tässä tilassa, mikäli se on sallittu koneen asetuksista, mutta kun ohjaus vapautetaan, kone jatkaa seuraavalle tehtäväpisteelle lentämistä.

- **LOITER**
  Autopilotti ohjaa konetta, niin että se lentää ympyrää pisteen ympärillä, jossa lentotila vaihdettiin päälle. Ympyrän säde voidaan määrittää autopilotin asetuksiin.
- **CIRCLE**
  Lähes kuin LOITER, mutta kone lentää ympyrää välittämättä pisteestä missä se asetettiin päälle. Esimerkiksi kovalla tuulella kone saattaa lentää pisteestä poispäin, koska tilassa ei pyritä pysymään saman pisteen ympärillä.
- **GUIDED**
  Kone lentää tässä tilassa maa-asemasta asetettuun pisteeseen ja sen saavuttaessaan kone alkaa lentämään ympyrää tämän pisteen ympärillä.
- **RTL (Return To Launch)**
  Kone palaa lähtöpisteeseen tai lennättäjän erikseen määrittämään pisteeseen ja lentää ympyrää sen ympärillä, niin pitkää kuin lentotila on päällä.
- **LAND**
  Kone suorittaa autonomisen laskeutumisen.
- **TAKEOFF**
  Kone suorittaa autonomisen lentoonlähdön.

QuadPlanen omat lentotilat

Näissä lentotiloissa kone lentää kopterimaisesti. QuadPlane lentotilat on helppo tunnistaa Q:lla alkavasta lentotilan nimestä. Opinnäytetyössä VTOL-lentotiloihin viitatessa tarkoitetaan viittauksella QuadPlane lentotiloja.

- **QACRO**
  Käytännössä manuaalinen tila. Autopilotti ei rajoita kallistuskulmia, tai pyri suoristamaan konetta, kun ohjaus vapautetaan, mutta vakauttaa koneen siihen kulmaan mihin se on myös ohjattu. Esimerkiksi kovan tuulen puhaltaessa konetta toiseen kulmaan, autopilotti pyrkii pitämään kulman samana kuin mihin se on ohjattu.
- **QSTABILIZE**
  Käyttäjä voi ohjata konetta itse, mutta koneen kallistuskulmaa rajoitetaan käyttäjän asettamiin arvoihin. Autopilotti pyrkii suoristamaan koneen käyttäjän vapauttaessa ohjauksen.
- **QHOVER**
  Käytännössä kuin QSTABILIZE, mutta autopilotti ohjaa koneen korkeutta sen perusteella missä asennossa tehovipu on. Vivun ollessa keskiasennossa autopilotti pyrkii pitämään koneen korkeuden samana. Suuremmalla teholla tavoitekorkeus kasvaa ja pienemmällä teholla korkeus laskee.
- **QLOITER**
  Lähes kuin QHOVER, mutta käyttäjän vapauttaessa ohjauksen, autopilotti pyrkii pitämään koneen sijainnin täysin samana sen sijaan että se pyrkisi pitämään koneen suorassa. Korkeuden ohjaus toimii samalla tapaa kuin QHOVER tilassa.
- **QLAND**

Autopilotti pyrkii laskeutumaan koneella pisteeseen, jossa tila asetettiin päälle.

- **QRTL (QuadPlane Return to Launch)**
  Kone palaa takaisin lähtöpisteeseen ja laskeutuu. Asetuksista voidaan erikseen määrittää nopeus ja korkeus, jolla kone lähestyy lähtöpistettä, sekä nopeus sille kuinka nopeasti koneen korkeus saa muuttua suorittaessa laskeutumista.
- **QAUTOTUNE**
  Autopilotti säätää automaattisesti PID-ohjaimet koneen ohjaukseen QuadPlane lentotiloihin tekemällä pieniä liikkeitä ja mittaamalla, kuinka kone vastaa niihin. Tavanomainen aika yhden akselin säätämiselle on noin 3-5 minuuttia.

3.3.4 Lentotilojen muutokset Plane- ja QuadPlane-tilojen välillä

Plane- ja QuadPlane-lentotilojen välisille lentotilan vaihdoksille ArduPlane sisältää kaksi lentotilaa, joita autopilotti käyttää sisäisesti mahdollisimman tasaisen lentotilan vaihdoksen aikaansaamiseksi. Tämä siis tarkoittaa sitä, että Plane-lentotilasta siirtyessä johonkin QuadPlane-tilaan, autopilotti vaihtaa päälle ensin tähän siirtymään kehitetyn lentotilan ja käyttää sitä niin pitkään, kunnes lentotilan vaihdos voidaan katsoa valmiiksi. QuadPlane-tilasta siirryttäessä johonkin Plane-tilaan prosessi on käytännössä sama, mutta lentotilan vaihdokseen käytetään erikseen siihen tarkoitettua lentotilaa. (Flying a QuadPlane n.d.)

Mikäli koneessa on kääntyvät moottorit, tulee autopilotin asetuksiin asettaa erikseen nopeus, jolla moottorit kääntyvät koneen vaihtaessa lentotilaa. Yksikkö nopeudelle on astetta per sekunti. Asetuksella ei määritetä sitä, kuinka nopeasti moottorit oikeasti kääntyvät, vaan kuinka nopeasti niitä halutaan autopilotin kääntävän (Tilt Rotor Planes n.d.) Oletettavasti nopeuden ei kuitenkaan tulisi olla suurempi kuin se mihin moottorit fyysisesti kykenevät.

Tailsitter-koneissa lentotilan vaihdokset eivät toteudu aivan samalla tapaa kuin muissa QuadPlane-koneissa, mutta edellä mainittu asia pätee myös tailsitter tyyppisiin koneisiin.

Lentotilaa vaihtaessa jostain QuadPlane-tilasta johonkin Plane-lentotilaan tailsitter-tyyppisessä koneessa, autopilotti asettaa moottoreiden tehon vähintään leijuntateholle ja alkaa kääntämään koneen nokkaa pystyasennosta kohti horisonttia, kunnes saavuttaa kulman, joka koneen tulee saavuttaa, jotta lentotilan vaihdos voidaan todeta suoritetuksi. Asetuksissa määritetään myös aika, kuinka nopeasti koneen halutaan tekevän tämän muutoksen. ArduPilot jakaa tämän ajan aina kahdella, mutta syytä tälle ei erikseen kerrota ArduPlanen dokumentaatiossa. Jos ajaksi asetetaan esimerkiksi 10 sekuntia, koneen nokka siirtyy pystyasennosta siirtymän tavoitekulmaan 5 sekunnin kuluessa. Oletuksena tavoitekulma siirtymälle on ArduPlanen asetuksissa 45 astetta. Jos siirtymä jostain syystä kestää pitempään kuin mitä ajaksi on asetettu, autopilotti katsoo siirtymän suoritetuksi ja vaihtaa lentotilaa, vaikka tavoitekulmaa ei olisi vielä saavutettukaan. (Tailsitter planes n.d.)

Lentotilaa vaihtaessa jostain Plane-tilasta johonkin QuadPlane-lentotilaan, autopilotti asettaa moottoreiden tehon leijuntateholle ja pyrkii pitämään koneen kallistuskulman nollassa eli siivet vaakatasossa. Autopilotti ohjaa koneen nokkaa ylöspäin kohti taivasta, käyttäen ohjainpintoja liikkeen suorittamiseen. Tilanvaihdos todetaan suoritetuksi, kun koneen nousukulma saavuttaa sille asetetun tavoitekulman tai kun tilanvaihdoksen todetaan kestäneen yli kaksi sekuntia. Kahden sekuntin raja tälle siirtymälle on kovakoodattu ArduPlanen lähdekoodiin eikä sitä voi muuttaa erikseen asetuksista. (Tailsitter planes n.d.)

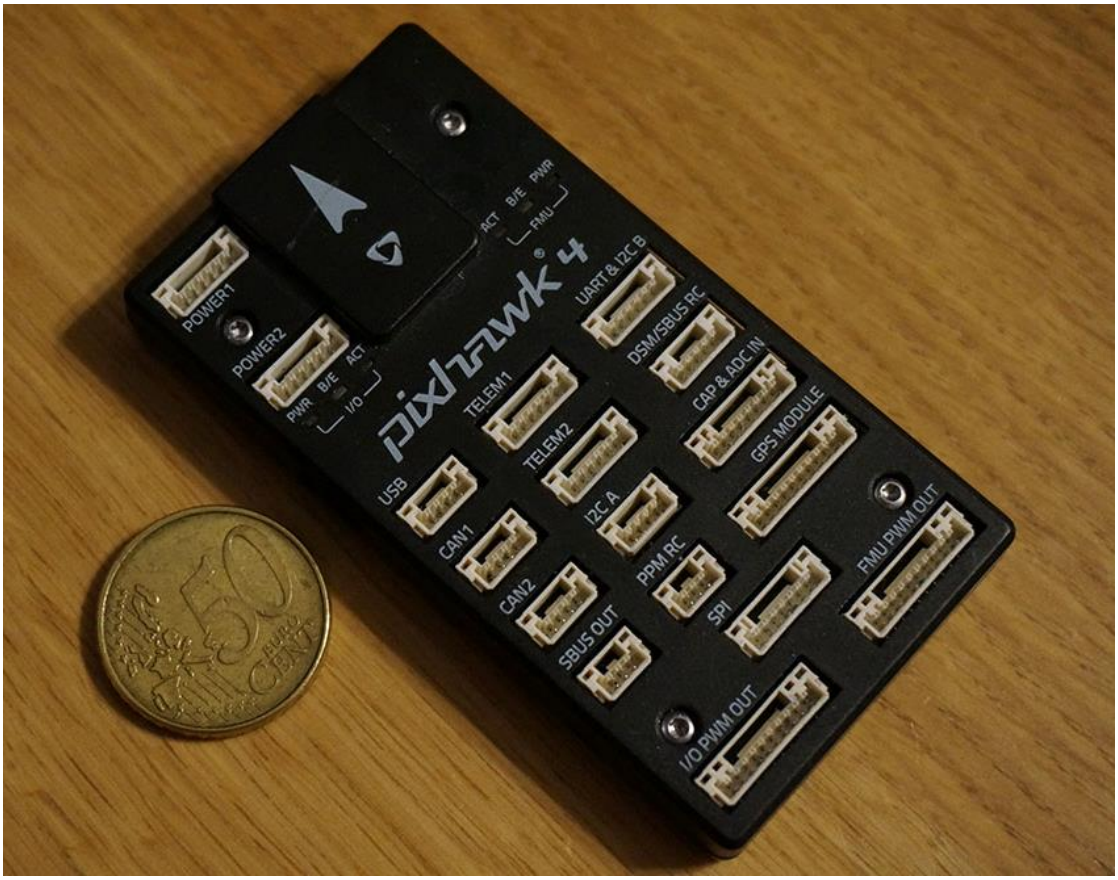
3.4 Käyttöalustat

ArduPilot tukee useita eri laitteistoja sen käyttöön. ArduPilotin tukemiin laitteisiin lukeutuu sekä avoimen että suljetun lähteen laitteistoja.

Eräs suosittu laitteisto ArduPilotin käyttöön on Pixhawk. Pixhawkin valikoimasta löytyy muutama eri vaihtoehto laitteistoksi, mutta kaikista uusin Pixhawk sarjan laitteisto tätä opinnäytetyötä tehdessä on Pixhawk 4 (ks. kuvio 11). Hintaa uudelle Pixhawk 4:lle kertyy noin 200 euroa. Kaikista oleellisimmat anturit, jotka ovat kaksi erillistä kiihtyvyysanturia ja gyroskooppia, sähköinen kompassi sekä korkeusanturi on rakennettu Pixhawk 4:n sisään. Pixhawk 4 tukee myös useita lisäsensoreita, jotka on

helppo kytkeä laitteen portteihin. Esimerkiksi GPS-vastaanotinta ei ole rakennettu laitteen sisään ja se tulee hankkia erikseen, mikäli sille on tarvetta. GPS-vastaanottimen hinta on noin 40 euroa. Pixhawkissa on yhteensä 16 erillistä PWM-ulostuloa, joten siihen pystyy kytkemään useita ulkoisia laitteita. (Pixhawk 4 n.d.)

Pixhawkin ohjelmointi voidaan suorittaa siinä olevan USB portin kautta.



Kuvio 11. Pixhawk 4

4 Kehitys ja testaus

ArduPilot on avoimen lähdekoodin ohjelmisto, joten kuka tahansa voi ladata lähdekoodin internetistä ja tehdä siihen haluamiaan muutoksia. Tarkat lähdekoodin käyttöehdot on määritetty ArduPilotin käyttämässä lisenssissä *”GNU General Public License v3.0”*. ArduPilot autopilotin lähdekoodi on kirjoitettu lähes kokonaan C++ ohjelmointikielellä.

4.1 Käytetyt ohjelmistot

4.1.1 Visual Studio Code

Visual Studio Code (usein käytetty lyhenne VS Code) on Microsoftin kehittämä ilmainen avoimen lähdekoodin tekstieditori, mikä on suunniteltu ohjelmointiin. VS Code julkaistiin keväällä 2015 ja se on sittemmin noussut yhdeksi suosituimmista kehitysalustoista ohjelmoijien keskuudessa. VS Codea voidaan käyttää Windows, Linux ja macOS käyttöjärjestelmillä. VS Code tukee alkuperäisellä konfiguraatiollaan JavaScriptiä, TypeScriptiä ja Node.js:ää, mutta siihen on saatavilla lukuisia laajennuksia muidenkin ohjelmointikielten tukemiseen. VS Codesta löytyy myös useita sisäänrakennettuja ominaisuuksia ja työkaluja, kuten Git versionhallinnan käyttöön. (Visual Studio Code 2020.)

4.1.2 Cygwin

Cygwin on kokoelma suosittuja avoimen lähdekoodin Linux ja Unix käyttöjärjestelmille kehitettyjä työkaluja, joita voidaan Cygwin-ympäristön avulla suorittaa myös Windows käyttöjärjestelmillä. Ydinosuus Cygwinia on kirjasto, joka tarjoaa työkaluille POSIX-järjestelmäkutsut sekä ympäristön, jota työkalujen toimiminen edellyttää. (Cygwin FAQ n.d.)

4.1.3 MAVProxy

MAVProxy on CanberraUAVn kehittämä minimalistinen maa-asema miehittämättömien ilma-alusten ohjaukseen. MAVProxy käyttää MAVLink kommunikaatioprotokollaa aluksen kanssa keskusteluun, jota myös ArduPilot tukee. MAVProxyssa ei ole varsinaista käyttöliittymää, vaan kaikki toiminnot tehdään antamalla komentoja ohjelman terminaaliin. ArduPilotin kanssa MAVProxya käytetäänkin yleisimmin ArduPilotin kehitykseen ja testaukseen. MAVProxyn mukana tulee kuitenkin oletuksena joitain lisäosia, joissa on graafinen käyttöliittymä. MAVProxy voi myös keskustella muiden maa-asemien kanssa UDP-verkkoprotokollaa käyttämällä, jolloin käyttäjä pystyy samaan aikaan käyttämään useaa eri maa-asemaa samalle ilma-alukselle. (MAVProxy n.d.)

4.1.4 SITL

SITL eli Simulation In The Loop on simulaattori, jonka avulla ArduPilotin koodia voidaan suorittaa Linux ja Windows käyttöjärjestelmiä käyttävillä tietokoneilla. Yksinkertaisuudessaan SITL on vain suoritettava ohjelma, joka luodaan ArduPilotin lähdekoodista. Koska tietokoneissa ei ole samanlaisia antureita kuin laitteistoissa, jossa ArduPilottia normaalisti ajetaan, kaikki sensoridata SITLää suorittaessa on myös simuloitua. (SITL n.d.)

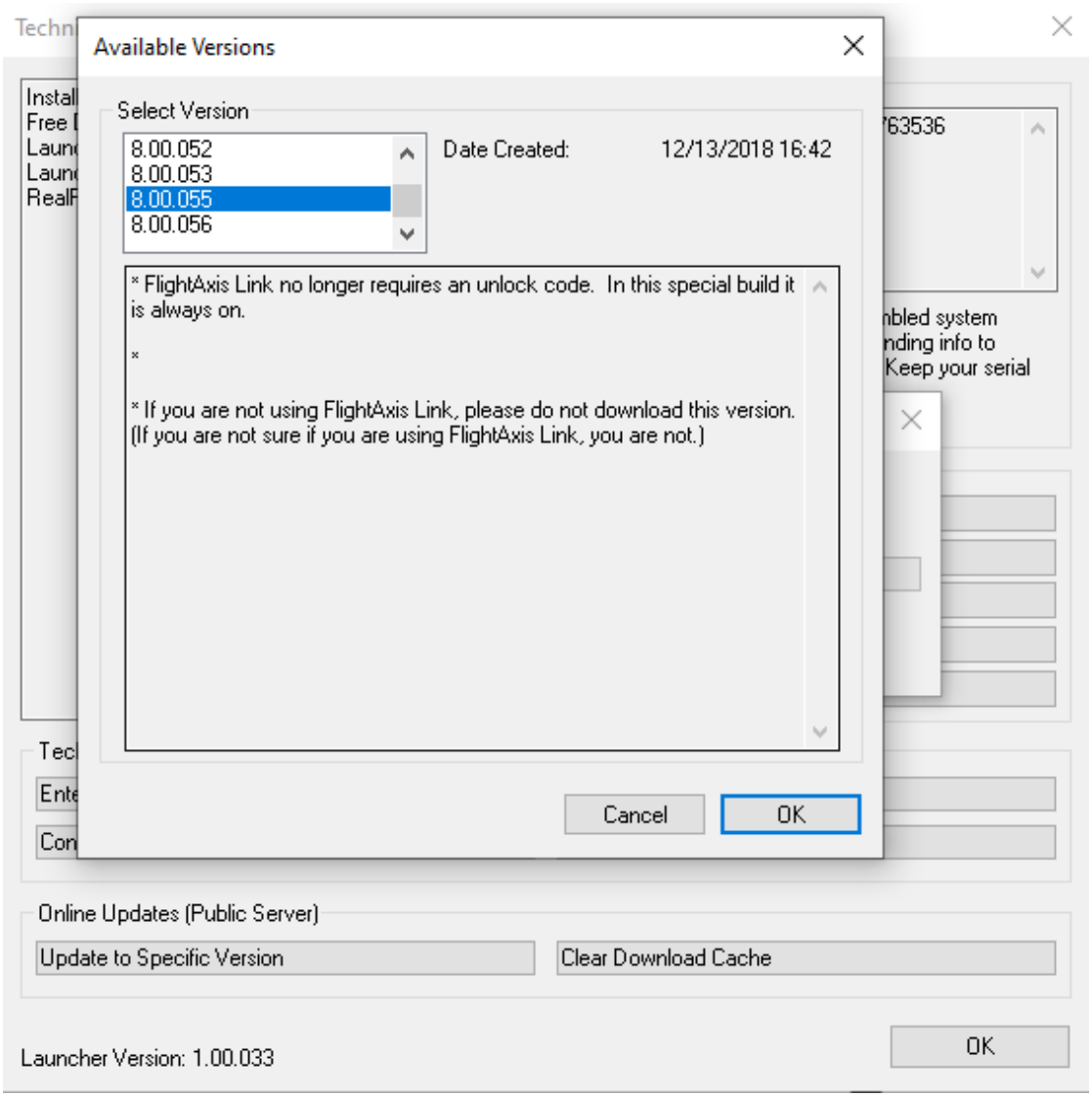
4.1.5 RealFlight 8

RealFlight 8 on Knife Edge Softwaren kehittämä ja Horizon Hobbyn vuonna 2018 julkaisema lennokkisimulaattori. RealFlight toimii ainoastaan Microsoftin Windows käyttöjärjestelmillä. RealFlightia voidaan käyttää yhdessä ArduPilotin SITLn kanssa koneen visualisoimiseen simuloinnin aikana.

4.2 Simulointi ja testaus

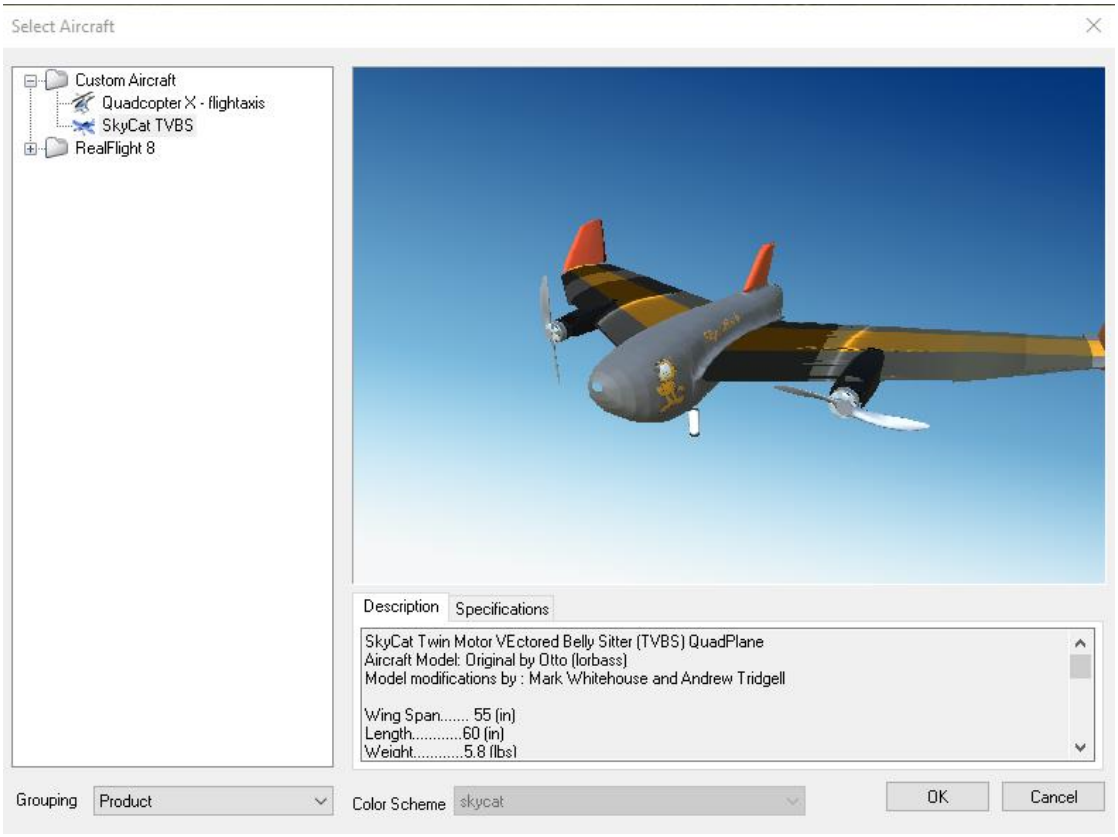
Muutosten tekeminen koodiin on huomattavasti turvallisempaa, halvempaa ja nopeampaa simulaattorin avulla kuin käyttämällä oikeaa ilma-alusta muutosten testaukseen.

Tässä opinnäytetyössä simulointi suoritettiin käyttämällä RealFlight 8 -lennokkisimulaattoria. RealFlight 8 kykenee keskustelemaan SITLn sekä ArduPilot-yhteensopivien maa-asemien kanssa FlightAxis-rajapinnan välityksellä. RealFlight 8:sta on olemassa useita eri versioita, mutta vain yhdessä RealFlight 8:n versiossa on julkinen tuki FlightAxis-rajapinnan käyttöön. Tarkka versio, josta tuki löytyy, on 8.00.055. Vaikka RealFlight 8:sta on olemassa myös versio 8.00.056, johon ohjelma ehdottaa myös päivittämään, niin tällä simulointi ei kuitenkaan ole mahdollista. Aluksi opinnäytetyön tekijä erehtyi päivittämään ohjelmiston viimeisimpään versioon, mutta yhteyttä RealFlightin ja Mission Plannerin välille ei kuitenkaan syntynyt. Ongelmaan kuitenkin löytyi vastaus ArduPilotin foorumilta, jossa käyttäjä ”Eosbandi” kertoi, että RealFlight 8 versionumeron tulee olla täsmälleen versio 8.00.055 toimiakseen ArduPilotin simuloinnissa (Realflight 8 and Mission planner Missing feature 2020). Version sai kuitenkin onneksi alennettua RealFlight 8:n mukana tulevan ”Technical support” työkalun avulla (ks. kuvio 12) ja yhteys Mission Plannerin ja RealFlightin välille saatiin luotua.



Kuvio 12. Version vaihto RealFlight 8 Technical support ohjelmassa

Simulointi tehtiin samantyyppisellä koneella, kuin mikä myös opinnäytetyön toimeksiantajalla oli, eli kaksimoottorisella tailsitterilla, jossa moottoreiden kulmaa voidaan säätää lennon aikana. Kone piti erikseen ladata ArduPilotin GitHub-repositoriosta, sillä RealFlight 8 mukana ei tule valmiina tailsitter tyyppisiä koneita. Kun kone oli ladattu GitHubista, se tuotiin RealFlightiin valitsemalla ylävalikosta **Simulation** > **Import** > **RealFlight Archive (RFX, G3X)**. Tämän jälkeen koneen pystyi valitsemaan valitsemalla ensin ylävalikosta **Aircraft** > **Select Aircraft**, jonka jälkeen uusi ikkuna avautui koneen valitsemiseen. Käyttäjien itse tuomat koneet löytyvät kansiosta ”Custom Aircraft” (ks. kuvio 13).



Kuvio 13. RealFlight 8 lennokin valintaikkuna

ArduPilotin dokumentaatio suosittelee laskemaan RealFlightin grafiikka-asetuksia ilma-aluksia testattaessa, jotta simulaattori jaksaisi pyöriä sulavasti samaan aikaan muiden simulaatioon vaadittavien ohjelmistojen kanssa, mutta opinnäytetyötä tehdyllä tietokoneella tätä ei havaittu tarpeelliseksi. Eräänä toisena huomiona mainittakoon se, että RealFlight 8:aa ei voi käynnistää, mikäli tietokoneeseen ei ole kytketty erillistä ohjainta lennokin ohjaukseen. Ohjaimena voidaan käyttää simulaattorikäyttöön soveltuvaa lennokin radio-ohjainta tai joystickiä. Opinnäytetyötä tehdessä ohjaimena käytettiin ”Mad Catz F.L.Y 5”-joystickiä. Jos konetta lennetään pelkän autopilotin avulla eikä ohjaimelle ole pakottavaa tarvetta, on myös mahdollista asentaa esimerkiksi ”vJoy” niminen ohjelma simuloimaan joystickiä, jonka jälkeen RealFlightin voi käynnistää ilman fyysistä ohjainta. (SITL with RealFlight n.d.)

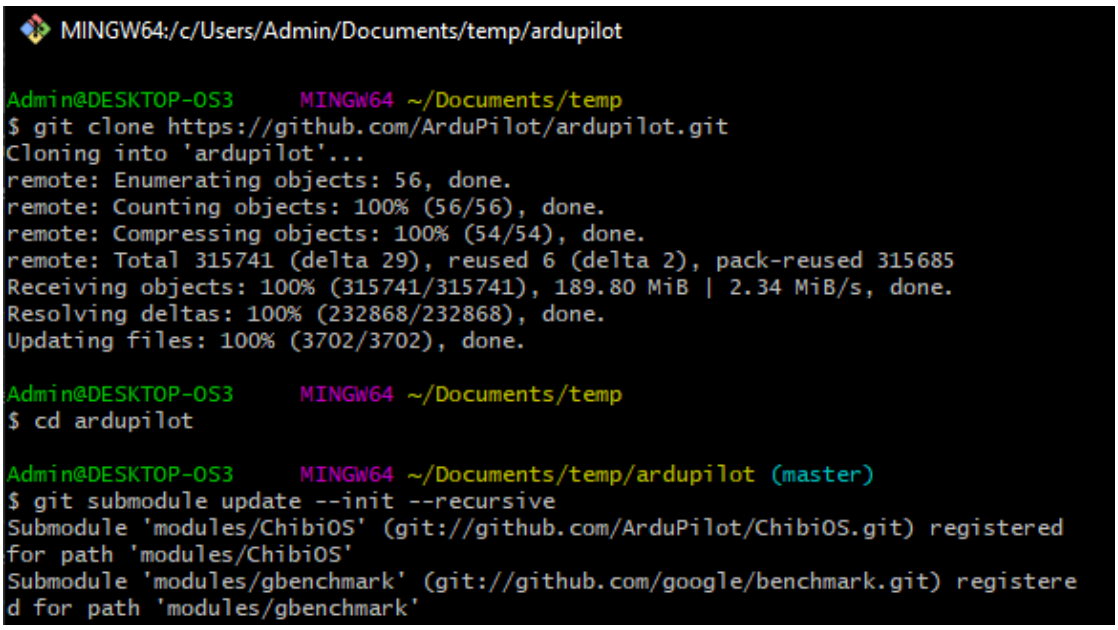
Simuloidessa ArduPilotin virallisia ohjelmistojulkaisuita voidaan simulointiin käyttää Mission Planner ohjelmistoa. Suorittaessa simulaatiota suoraan ArduPilotin lähdekoodista, voidaan simulointiin käyttää MAVProxy ohjelmistoa. Kuviossa 14 nähdään

tailsitter-tyyppinen kone RealFlight 8 -simulaattorissa visualisoituna ja Mission Planerilla ohjattuna.

Kuvio 14. Tailsitter-koneen simulointi RealFlight 8:aa ja Mission Planneria käyttäen

Windows-käyttöjärjestelmillä suorittaessa simulointia suoraan ArduPilotin lähdekoodista, ensimmäinen askel on ladata ArduPilotin lähdekoodi GitHub-palvelusta. Lähdekoodi voidaan ladata kolmella komennolla ArduPilotin GitHub-repositoriosta käyttämällä Git-komentoriviä (ks. kuvio 15):

1. git clone https://github.com/ArduPilot/ardupilot
2. cd ardupilot
3. git submodule update --init –recursive



Kuvio 15. ArduPilotin lähdekoodin lataaminen käyttäen Git-komentoriviä

Kun lähdekoodi on ladattu, on aika käynnistää Cygwin-terminaali, jonka avulla koodi saadaan käännettyä ja käynnistettyä simulaattoria varten. Terminaalissa tulee ensin siirtyä kansioon, jossa ArduPilot sijaitsee, jonka jälkeen simulaattori saadaankin jo melkein käynnistettyä. Käyttäjän tulee kuitenkin itse määrittää mitä alustyyppiä hän haluaa testata. Esimerkiksi lennokkia testatessa tulee terminaalissa siirtyä vielä *”ArduPlane”* kansioon, tai vaihtoehtoisesti antaa komentoriville ylimääräisenä argumenttina: *”-v ArduPlane”*. Opinnäytetyössä esitellyissä komennoissa on käytetty ensimmäistä menetelmää testaukseen. Alla on esimerkki komennosta, jolla kansioon voidaan siirtyä:

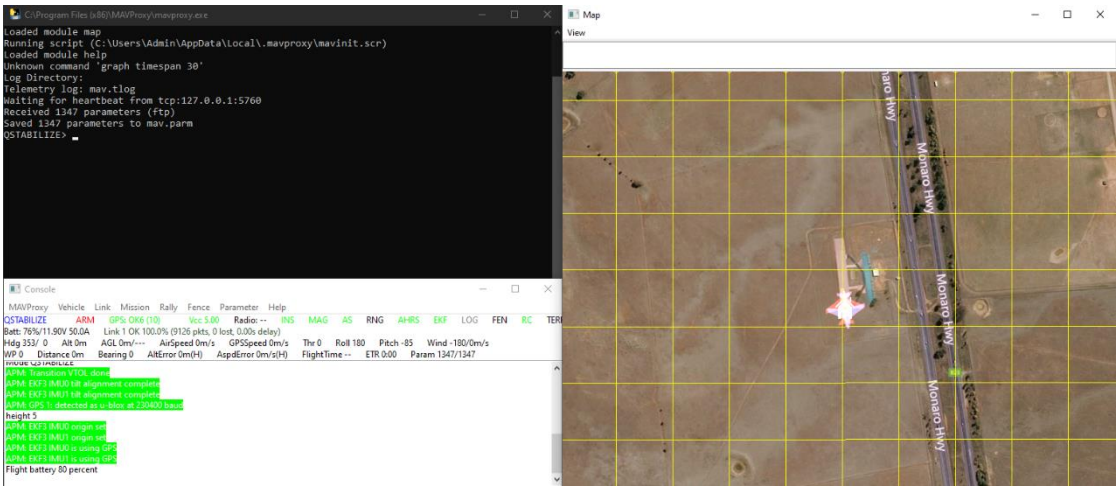
cd c:/Users/Admin/Documents/ardupilot/temp/ArduPlane/

Komento on tietokonekohtainen ja kansion sijainti pitää määrittää komentoon sen mukaan mihin ArduPilotin lähdekoodin on ladannut.

Toisena komentona on itse simulaattorin käynnistys, joka kääntää koodin sekä käynnistää MAVProxyn:

../Tools/autotest/sim_vehicle.py -f plane-tailsitter --model flightaxis --map --console

Kun koodi on käännetty, MAVProxy käynnistyy ja näkyviin tulee kolme eri ikkunaa. Kuviossa 16 vasemmalla ylhäällä nähdään MAVProxyn terminaali, sen alapuolella MAVProxyn konsoli ja oikealla puolella kartta. Kaikkia näitä ikkunoita ei tosin ole pakko avata; esimerkiksi kartan voi jättää pois poistamalla edellä mainitusta komennosta parametrin *"--map"*.



Kuvio 16. MAVProxy käynnistettynä kartan ja konsolin kera

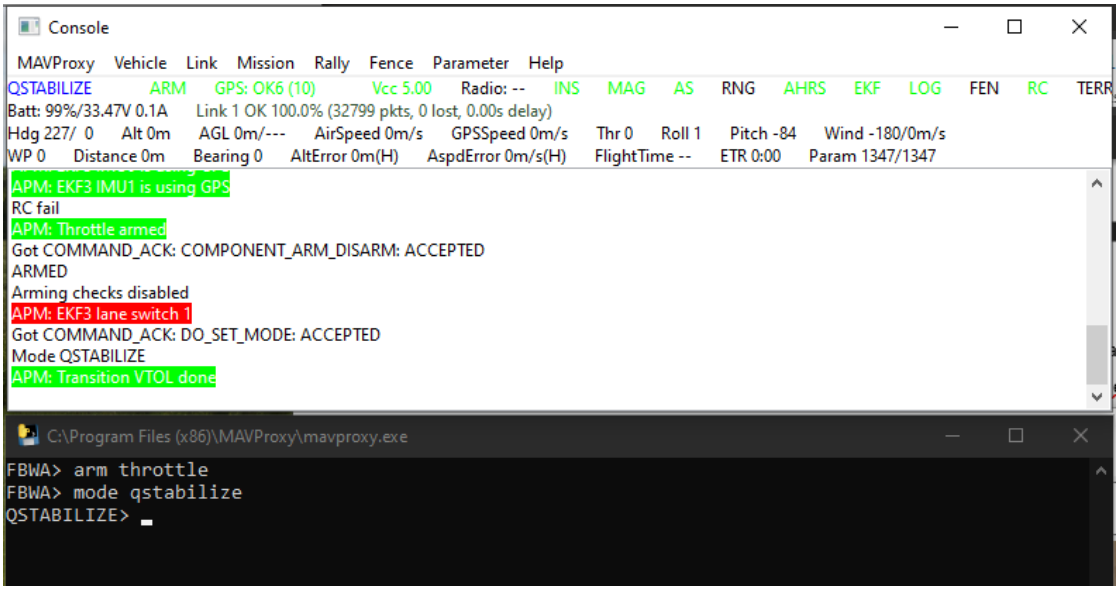
Jos RealFlight on nyt käynnissä, sen vasempaan alareunaan pitäisi tulla näkyviin vihreä teksti *"Flightaxis Controller Device has been activated"*.

Mikäli MAVProxya käytetään ensimmäistä kertaa, simuloitava konetyyppi on muuttunut tai koneen parametrit ovat muuttuneet, tulee käyttäjän ladata koneen parametrit SITLn simuloimaan EEPROM-muistiin. Tämä voidaan tehdä suorittamalla komento: "*param load <tiedoston sijainti>"* MAVProxyn konsolissa. Komento voi näyttää esimerkiksi seuraavanlaiselta:

param load ../../SITL/SITL_Models-master/RealFlight/Released_Models/QuadPlanes/SkyCat_TVBS/SkyCat_TVBS.param

Konetta voi nyt simuloida ja ohjata MAVProxyn kautta ja koneen lentoa voidaan seurata RealFlight-simulaattorista sekä MAVProxyn kartasta. Ennen kuin kone saadaan

ilmaan, pitää sen moottorit ikään kuin vapauttaa varmistuksesta. Moottoreiden varmistaminen on suojamekanismi oikeita ilma-aluksia lennettäessä, jotta aluksen moottorit eivät lähde vahingossa pyörimään aluksen ollessa maassa. MAVProxyn terminaaliin antamalla komennon *”arm throttle”* moottorit vapautetaan tästä varmistuksesta. Konsoliin pitäisi tässä vaiheessa ilmestyä teksti *”ARMED”* (ks. kuvio 17). Konetta voi nyt lentää joko ohjaimen välityksellä tai luomalla tehtävän autopilotille, jota autopilotti lentää autonomisesti. Koneen lentotilaa voidaan vaihtaa komennolla *”mode <tilan nimi>”* (ks. kuvio 17). Kaikki simuloitavalle konetyypille mahdolliset lentotilat voi tulostaa terminaaliin pelkällä komennolla *”mode”*.

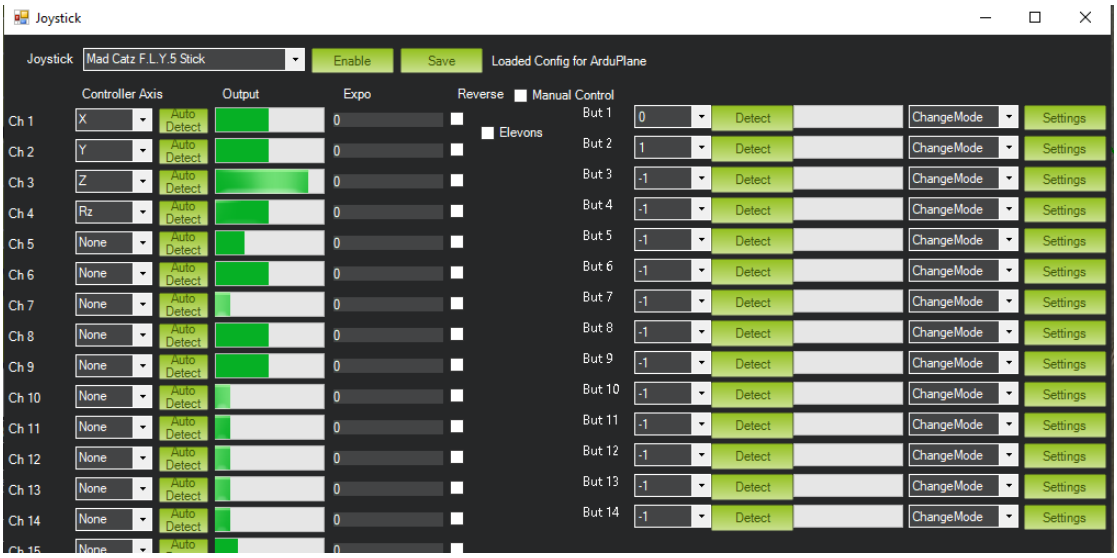


Kuvio 17. MAVProxyn konsoli ja terminaali

Mikäli komentojen antaminen MAVProxyn terminaaliin ei tunnu luontevalta, voidaan apuna käyttää myös Mission Planneria, jonka kautta samat toiminnot voidaan suorittaa käyttämällä graafista käyttöliittymää. MAVProxyn ollessa käynnissä, kun Mission Planner käynnistetään, Mission Plannerin pitäisi automaattisesti luoda yhteys MAVProxyn kanssa. Yhteys luodaan myös, jos ohjelmat käynnistetään käänteisessä järjestyksessä. Mission Plannerin oikeassa yläkulmassa pitäisi lukea ”Disconnect” mikäli yhteys on luotu. MAVProxyn voi tämän jälkeen myös käynnistää uudelleen (jos haluaa esimerkiksi tehdä muutoksen koodiin ja testata sitä) ja yhteys luodaan silti automaattisesti. Kun yhteys on luotu, Mission Planneria voidaan käyttää täysin samalla tapaa kuin sitä käytettäisiin oikeankin lennokin kanssa.

Eräs hyödylliseksi havaittu asia testauksessa oli asettaa Mission Plannerista joystickin napit vaihtamaan koneen lentotilaa. Tämä voidaan tehdä menemällä Mission Plannerin päänäkymässä ”Actions” välilehteen ja klikkaamalla ”Joystick” nappulaa. Tämän jälkeen käyttäjälle avautuu uusi ikkuna, jossa joystick voidaan konfiguroida. Asetukset jäävät muistiin, mutta aina kun Mission Planner käynnistetään uudelleen, on käyttäjän käytävä klikkaamassa samassa ikkunassa olevaa ”Enable” nappia, jotta asetukset tulevat voimaan.

Kuvion 18 oikeassa laidassa nähdään, että joystickin napit 0 ja 1 on asetettu vaihtamaan koneen lentotilaa, kun niitä painetaan. Se mihin lentotilaan napin painallus vaihtaa, määritetään painamalla samalla rivillä olevaa ”Settings” nappia, joka avaa uuden ikkunan, jossa lentotila valitaan.



Kuvio 18. Joystickin konfigurointi Mission Plannerissa

4.3 Debuggaus

Koodin debuggaus voi tarkoittaa montaa asiaa, mutta yleisesti sanottuna se tarkoittaa bugien etsimistä koodista. Debuggaamalla koodia voidaan myös saada parempi käsitys siitä, kuinka koodi toimii. Koodin debuggaukseen on olemassa useita eri menetelmiä. Koodia voidaan debugata esimerkiksi lukemalla lähdekoodia, tai käyttämällä koodin analysoijaa. Koodia voidaan myös debugata käyttämällä jonkinlaista suorituskyvyn profiloijaa tai varsinaista debuggeria. (What is debugging 2020.)

ArduPilotin varsinaiseen debuggaukseen voidaan käyttää kahta lähestymistapaa. Ensimmäinen tapa on perinteinen ns. print debuggaus, jossa koodin upotetaan funktio, joka tulostaa jonkin viestin terminaaliin. Tällä tapaa voidaan tarkastaa esimerkiksi se, että päätyykö ohjelma koskaan ajamaan jotain tiettyä koodinpätkää, tai mitä arvoa jokin muuttuja pitää koodin suoritushetkellä. Toinen ja ehkäpä hieman ammattimaisempi tapa debugata koodia on käyttää varsinaista debuggeria. Debuggeria ei voi kuitenkaan ajaa kuin SITLn kanssa, toisin kuin taas print metodia voidaan käyttää myös oikeiden lentojen aikana. Oikeiden lentojen aikana ArduPilot voi lähettää viestejä maa-asemaan, tai kirjoittaa niitä omaan muistiinsa. Yksi debuggeri, joka toimii SITLn kanssa on ohjelmisto nimeltä GDB eli GNU Debugger. Ohjelmisto tosin toimii vain Linux käyttöjärjestelmillä. GDB:n käyttöön ei tutustuta tarkemmin tässä opinnäytetyössä.

Maa-aseman konsoliin viestien kirjoittaminen onnistuu käyttämällä koodissa makroa:

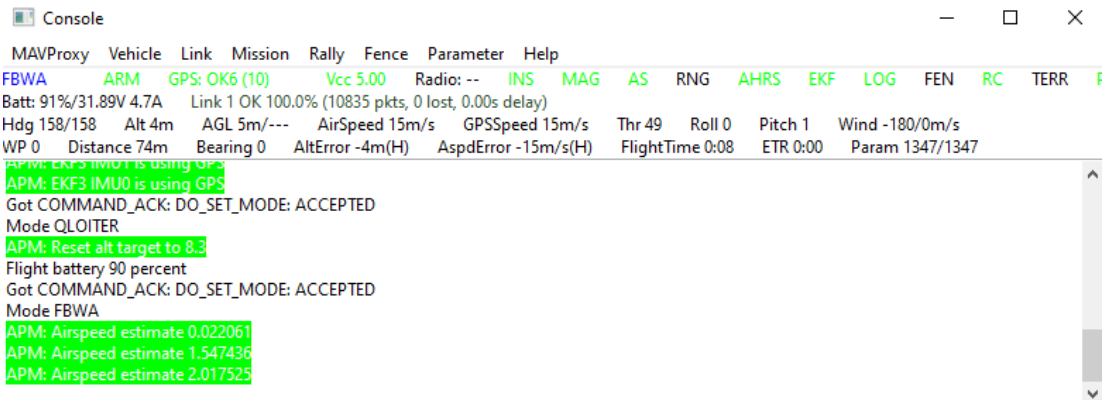
*GCS_SEND_TEXT(MAV_SEVERITY severity, const char *fmt, ...)*

Makro käyttää samaa formaattia tekstin muotoiluun kuin C-ohjelmointikielen printf metodi. Esimerkki makron käytöstä koneen ilmanopeuden arvion tulostamiseen nähdään kuviossa 19.

```
float airspeed_ms;
if (ahrs_view->airspeed_estimate(airspeed_ms)) {
    GCS_SEND_TEXT(MAV_SEVERITY_DEBUG, "Airspeed estimate: %f", airspeed_ms);
}
```

Kuvio 19. Ilmanopeuden arvion kirjoitus maa-aseman konsoliin ArduPilot-koodissa

Kuviossa 20 makro nähdään käytännössä. Kolme ilmanopeuden eri arviota on kirjoitettu MAVProxyn konsolin kolmelle alimmalle riville sen jälkeen, kun lentotila on vaihdettu QLOITER:ista FBWA:ksi.



Kuvio 20. Ilmanopeuden arvio MAVProxyn konsolissa

4.4 Lähdekoodin lataaminen laitteeseen

Kun lähdekoodi on testattu toimivaksi ja se halutaan ottaa käyttöön oikeassa aluksessa, lähdekoodi pitää kääntää laitteelle sopivaan muotoon; toisin sanottuna lähdekoodi pitää kääntää konekieleksi. ArduPilotiin liitetyillä työkaluilla toimenpide onnistuu melko vaivattomasti.

Ensiksi Cygwin-terminaali tulee avata ArduPilotin lähdekoodin sisältävään kansioon. Kansiossa ollessaan, seuraavalla komennolla määritetään laite, johon uusi ohjelma halutaan ladata sisään:

./waf configure –board <laitteen nimi>

Jos varmuutta laitteen nimestä ei ole, voidaan kaikki mahdolliset laitteen nimet tulostaa terminaaliin komennolla:

./waf list_boards

Käytettäessä Pixhawk 4 laitetta, voidaan käyttää seuraavaa komentoa:

./waf configure --board Pixhawk4

Kun laite on määritetty, voidaan lähdekoodi kääntää komennolla:

./waf <alustyyppi>

Käännettäessä lähdekoodia esimerkiksi lennokille, annetaan komentoon alustyypiksi plane:

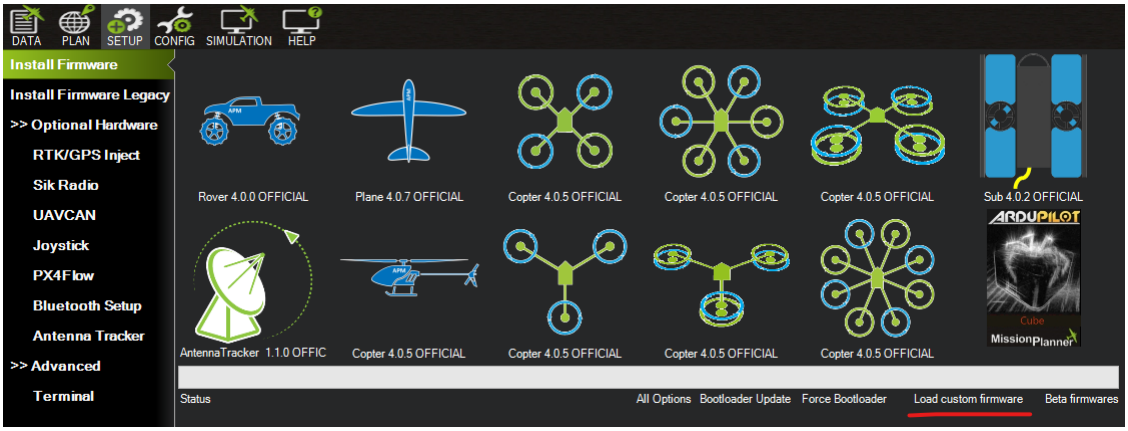
./waf plane

Kun komento on suoritettu onnistuneesti, kansiosta: ”*ardupilot/build/<laitteen nimi>/bin*”, saadaan tarvittavat tiedostot uuden ohjelmiston käyttämiseen. Kansiosta pitäisi komennon suorittamisen jälkeen löytyä mm. ”*apj*” ja ”*bin*” tyyppisiä tiedostoja nimellä ”arduplane”, mikäli alustyypiksi valittiin plane.

Uusi ohjelmisto voidaan ladata laitteeseen samalla, kun koodi on käännetty, lisäämällä komennon loppuun parametri ”*--upload*”. Vain Pixhawk- ja Linux-pohjaiset laitteet tukevat tätä ominaisuutta. Esimerkiksi seuraavaa komentoa käyttäen, koodi käännetään ja ladataan automaattisesti laitteeseen, joka on sillä hetkellä kytketty tietokoneeseen:

./waf plane --upload

Toinen tapa ladata uusi ohjelmisto laitteeseen on käyttää Mission Planneria. Mission Plannerin ”Setup”-välilehdellä vasemmasta laidasta voidaan valita näkymä ”Install Firmware”, jonka oikeasta alareunasta löytyy nappi ”Load custom firmware” (ks. kuvio 21), jota klikkaamalla aukeaa uusi ikkuna, josta voidaan valita ”*apj*”-tyyppinen tiedosto. Tämän lisäksi muuta ei tarvitse tehdä, kunhan laite on kytketty tietokoneeseen.



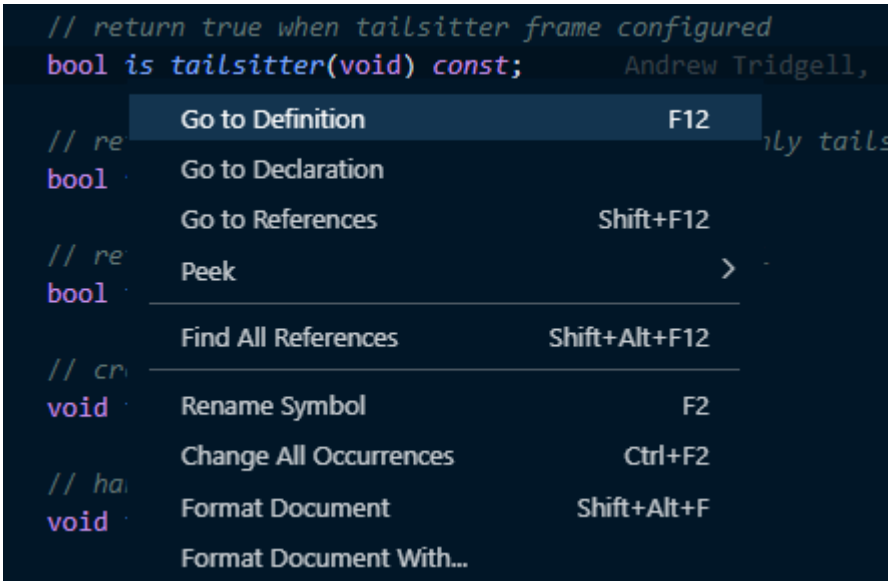
Kuvio 21. Mission Plannerin “Install Firmware” näkymä

4.5 Analysointi

Koko ArduPilot-projekti itsessään sisältää yli 700 000 riviä lähdekoodia, joten lähdekoodin rakenteeseen ja toimintaan perehtyminen vie väkisinkin oman aikansa. Opinnäytetyötä tehdessä koodin analysointia helpotti hyvin nimetyt tiedostot ja kattavasti kommentoitu koodi sekä Visual Studio Codesta löytyvät työkalut. Tässä luvussa käydään läpi yleisiä hyväksi havaittuja työkaluja koodin analysointiin.

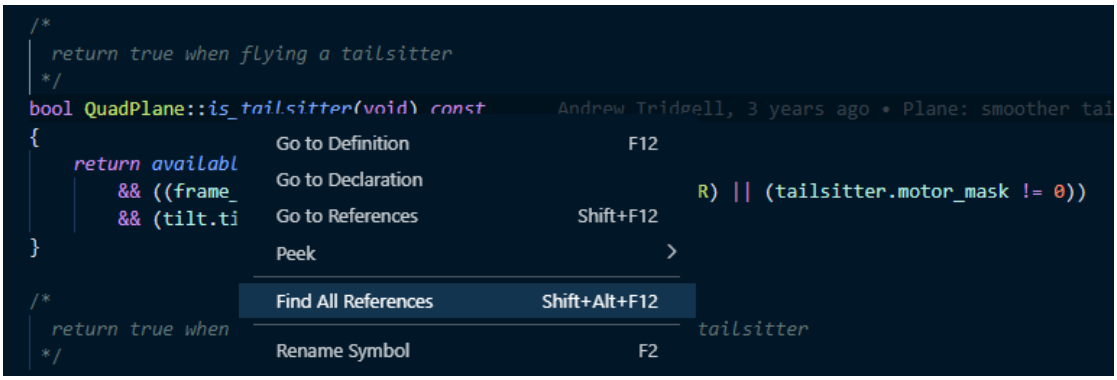
Hyvänä lähtökohtana koodiin perehtymiseen opinnäytetyön tekijä voi suositella ”Plane.h”-tiedostoa, sillä se sisältää kaikki metodit, luokat ja muuttujat, mitä autopilotti lennokkikäytössä käyttää.

Kuviossa 22 nähdään quadplane.h tiedostosta löytyvän ”is_tailsitter” metodin määritelmä. Klikkaamalla metodia hiiren oikealla napilla, VS Codessa aukeaa valikko, josta voidaan suorittaa useita eri toimintoja. Kun jokin mielenkiintoisen niminen muuttuja tai metodi löytyy, voi olla hyödyllistä nähdä kuinka se on implementoitu tai kuinka sitä käytetään. Tällöin voidaan käyttää kuviossa 22 näkyvää ”Go to Definition” toimintoa, joka vie käyttäjän tiedostoon riville, jolla metodi on implementoitu.



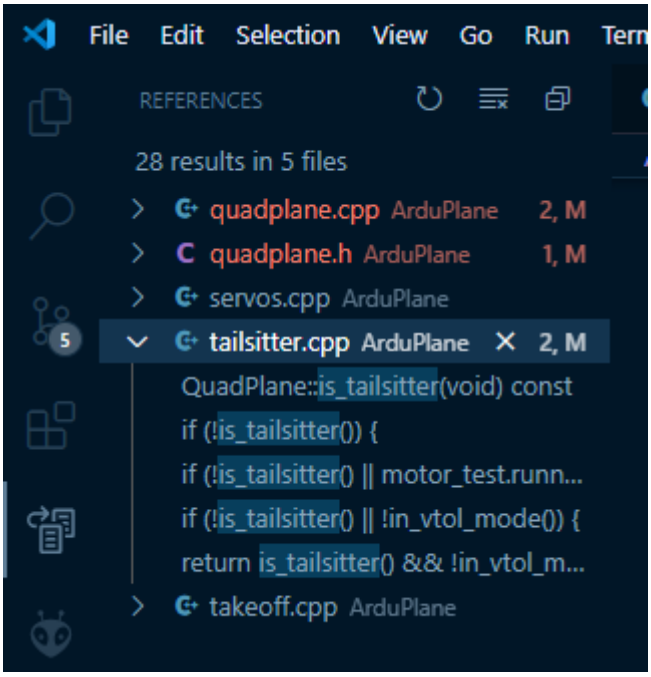
Kuvio 22. Visual Studio Coden “Go to Definition” toiminto

Koodia analysoidessa kiinnostavaa saattaa olla se, missä jotain metodia käytetään. Tällöin voidaan käyttää VS Coden ”Find All References” toimintoa (ks. kuvio 23), joka käy läpi kaikki tiedostot ja kansiot, jotka ovat VS Codessa avattuna ja etsii mahdolliset referenssit kyseiselle metodille tai muuttujalle. Referenssillä tässä tapauksessa tarkoitetaan jotain sijaintia koodissa, jossa jotain metodia tai muuttujaa käytetään.



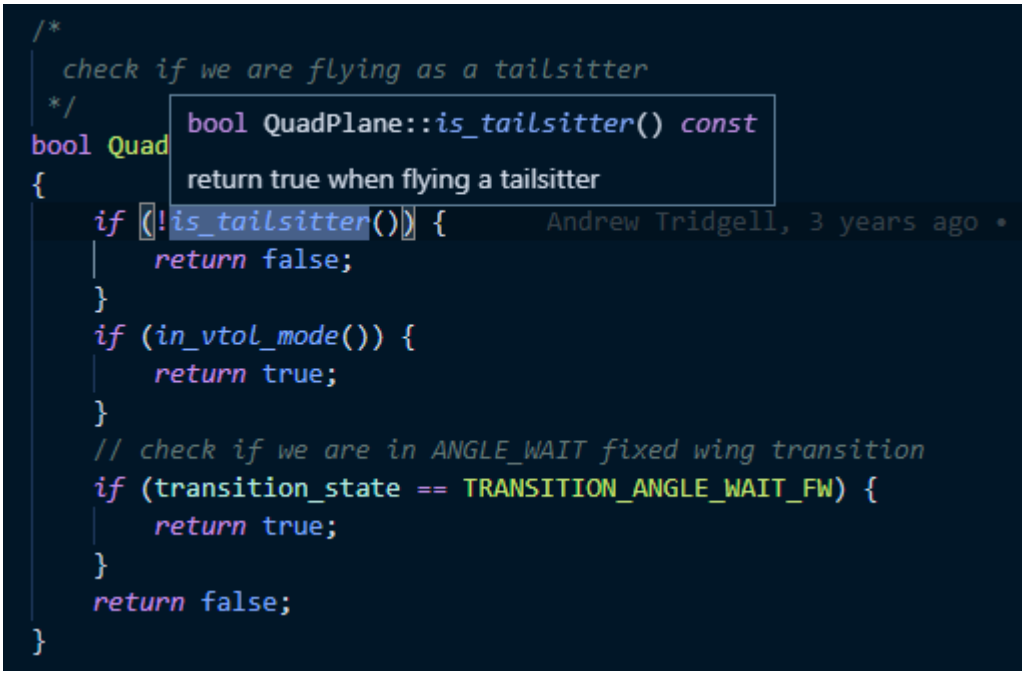
Kuvio 23. Visual Studio Coden “Find All References” toiminto

Kun VS Code on käynyt kaikki tiedostot läpi, mahdolliset referenssit tulevat näkyviin VS Coden vasempaan laitaan (ks. kuvio 24). Referenssejä voi tämän jälkeen klikata ja referenssiä klikatessa, ominaisuus vie käyttäjän tiedostoon riville, jolta referenssi on löydetty.



Kuvio 24. Visual Studio Coden löytämät referenssit QuadPlane luokan ”is_tailsitter” metodille

Jos jokin muuttuja tai metodi on kommentoitu, VS Code näyttää kommentin omassa pienessä ikkunassaan, kun käyttäjä vie hiiren metodin tai muuttujan yläpuolelle (ks. kuvio 25). Ominaisuus saattaa olla erittäin hyödyllinen, jos metodin tai muuttujan tarkkaa käyttökohdetta ei pysty päättelemään pelkästään sen nimen perusteella. Tarkastellessa jotain metodia ominaisuus näyttää myös mahdolliset metodille annettavat parametrit.



Kuvio 25. Visual Studio Coden kommentti-ikkuna

4.6 Ohjelmistoon tehdyt muutokset

4.6.1 Yleiset käytänteet

Muutoksien tekemiseen käytettiin ArduPilotin suosittamia käytänteitä. Avoimen lähdekoodin projekteissa on yleistä käyttää ennalta määritettyjä käytänteitä, jotta koodi pysyy yhtenäisenä useiden eri kehittäjien sitä muokatessa. ArduPilotin käytänteisiin kuuluvat mm. muuttujien ja metodien nimeäminen ohjeiden mukaisesti. Esimerkiksi muuttujat ja metodit tulee aina nimetä käyttämällä vain pieniä kirjaimia ja erottamalla erilliset sanat alaviivaa käyttäen (Style guide n.d).

Käytänteisiin kuuluu mm. myös koodin sisentäminen tietyllä tapaa ja kaikkien julkisten metodien ja tiedostojen kommentointi, niin että niiden käyttötarkoitus olisi selkeä ymmärtää kaikille projektiin uusillekin kehittäjille (Style guide n.d).

4.6.2 Muutokset

Ensimmäinen ongelma, oli koneen suorittaessa siirtymää VTOL-tilasta tavalliseen lentotilaan. Koodi, jolla tätä lentotilan vaihtoa ohjattiin, löytyi tiedostosta quadplane.cpp, josta löytyy ”update_transition” niminen metodi, mikä ohjaa koneen siirtymää VTOL-tilasta tavalliseen lentotilaan. Metodi ohjaa konetta riippuen siitä mikä koneen sen hetkinen siirtymätila on. Mahdollisia siirtymätiloja QuadPlane-tyyppisellä koneella on viisi kappaletta (ks. kuvio 26).

```
enum {
    TRANSITION_AIRSPEED_WAIT,
    TRANSITION_TIMER,
    TRANSITION_ANGLE_WAIT_FW,
    TRANSITION_ANGLE_WAIT_VTOL,
    TRANSITION_DONE
} transition_state;
```

Kuvio 26. QuadPlane-konetyypille mahdolliset siirtymätilat

Huomiona mainittakoon se, että tailsitter-tyyppiset koneet, jota opinnäytetyön toimeksiantajakin käyttää, eivät koskaan päädy ”TRANSITION_AIRSPEED_WAIT” tilaan, vaan ne käyttävät ”TRANSITION_WAIT_FW” tilaa sen sijaan.

Koneen varsinainen ohjaus suoritetaan samasta metodista löytyvästä switch-case rakenteessa eli ns. tilakoneessa. Tailsitter-tyyppisen koneen suorittaessa siirtymää VTOL-tilasta tavalliseen tilaan, on koneen tila aina ”TRANSITION_WAIT_FW”.

Kuviossa 27 nähdään logiikka siirtymälle, jossa konetta käännetään pystyasennosta kohti horisonttia. Ensimmäisenä koneen moottoreille annetaan lupa käyttää tehoa rajattomasti ja ”assisted_flight” muuttuja asetetaan arvoon ”true”, joka opinnäytetyön tekijän parhaan ymmärryksen mukaan tarkoittaa siirtymän aikana sitä, että autopilotti saa luvan ohjata koneen suuntaa vapaasti.

Tämän jälkeen lasketaan kerroin ”transition_rate”, joka kertoo, kuinka nopeasti koneen nokkaa käännetään pystyasennosta kohti horisonttia. Kerroin lasketaan käyttäjän itse määrittämistä parametreistä, jotka ovat aika, jossa siirtymä tapahtuu jaettuna kahdella, sekä kulma joka koneen tulee saavuttaa, jotta siirtymä voidaan todeta suoritetuksi.

Seuraavaksi lasketaan aika millisekunneissa, joka on kulunut siirtymän aloituksesta, ”dt” nimiseen muuttujaan. Kertomalla ”dt” sekä ”transition_rate” saadaan laskettua tavoitekulma koneen nousukulmalle. Kulma rajoitetaan ”constrain_float” metodilla niin, että se ei koskaan ole yli 0 tai alle -85 astetta. Ajan kasvaessa, tavoitekulma laskee sen myötä, niin että koneen nokka laskee pystyasennosta kohti horisonttia.

Seuraavaksi autopilotille kerrotaan koneen haluama kallistus- ja nousukulma, tallentamalla ne ensin omiin muuttujiinsa ja sen jälkeen lähettämällä ne eteenpäin ”attitude_control” luokalle, joka toteuttaa koneen ohjauksen. Kallistuskulmaksi kone tavoittelee aina nollaa siirtymän aikana. Koodissa nähdään myös ”check_attitude_relax” niminen metodi, joka nollaa joitain koneen ohjaukseen käytettäviä PID-ohjaimen arvoja, mikäli ne ovat olleet liian pitkään käyttämättöminä. Esimerkiksi koneen ollessa manuaalitilassa, nämä PID-ohjaimet ovat käyttämättä ja saattavat kerryttää sisäisesti suuren virheen, joka voi johtaa odottamattomaan käytökseen, kun ne taas otetaan käyttöön, jos virhettä ei nollata.

Viimeisenä vaiheena koodissa, asetetaan moottoreiden teho. Tehoksi asetetaan joko ”attitude_control” objektin haluama teho, tai ”hover_throttle”, riippuen siitä kumpi on suurempi.

```
case TRANSITION_ANGLE_WAIT_FW: {
    motors->set_desired_spool_state(AP_Motors::DesiredSpoolState::THROTTLE_UNLIMITED);
    assisted_flight = true;
    // calculate transition rate in degrees per
    // millisecond. Assume we want to get to the transition angle
    // in half the transition time
    float transition_rate = tailsitter.transition_angle / float(transition_time_ms/2);
    uint32_t dt = now - transition_start_ms;
    float pitch_cd;
    pitch_cd = constrain_float((-transition_rate * dt)*100, -8500, 0);
    // if already pitched forward at start of transition, wait until curve catches up
    plane.nav_pitch_cd = (pitch_cd > transition_initial_pitch)? transition_initial_pitch : pitch_cd;
    plane.nav_roll_cd = 0;
    check_attitude_relax();
    attitude_control->input_euler_angle_roll_pitch_euler_rate_yaw(plane.nav_roll_cd,
                                                                  plane.nav_pitch_cd,
                                                                  0);
    // set throttle at either hover throttle or current throttle, whichever is higher, through the transition
    attitude_control->set_throttle_out(MAX(motors->get_throttle_hover(),attitude_control->get_throttle_in()), true, 0);
    break;
}
```

Kuvio 27. Logiikka VTOL-tilasta tavalliseen lentotilaan siirtymiselle

Siirtymä voidaan todeta tailsitter-tyyppistä konetta lentäessä suoritetuksi, kun QuadPlane-luokan ”tailsitter_transition_fw_complete” metodi niin kertoo (ks kuvio 28). Metodi lukee mm. koneen kallistus- ja nousukulman ”ahrs_view” luokalta. Käytännössä siirtymä todetaan suoritetuksi, jos jokin seuraavista ehdoista toteutuu:

- **Kone on lentotilassa, jossa se lentää väärinpäin.**
- **Kone on saavuttanut tavoitekulman siirtymälle.**
- **Koneen kallistuskulma on suurempi kuin tavoitekulma siirtymälle** (tämän ei pitäisi tapahtua normaaleissa olosuhteissa, koska autopilotti pyrkii aina siirtymän aikana pitämään kallistuskulman nollassa, joten kyseessä on todennäköisesti turvamekanismi siltä varalta, että jokin menee pieleen).
- **Koneen siirtymässä on kestänyt pitempään kuin sen on määritetty kestävän** (käyttäjän määrittämä parametri).

```
/*
  return true when we have completed enough of a transition to switch to fixed wing control
 */
bool QuadPlane::tailsitter_transition_fw_complete(void)
{
    if (plane.fly_inverted()) {
        // transition immediately
        return true;
    }
    int32_t roll_cd = labs(ahrs_view->roll_sensor);
    if (roll_cd > 9000) {
        roll_cd = 18000 - roll_cd;
    }
    if (labs(ahrs_view->pitch_sensor) > tailsitter.transition_angle*100 ||
        roll_cd > tailsitter.transition_angle*100 ||
        AP_HAL::millis() - transition_start_ms > uint32_t(transition_time_ms)) {
        return true;
    }
    // still waiting
    return false;
}
```

Kuvio 28. Siirtymän toteaminen suoritetuksi tailsitter-koneissa

Tässä siirtymässä havaittiin yksi pieni ongelma. Koneen ilmanopeus saattaa olla huomattavastikin pienempi kuin koneen sakkausnopeus, kun siirtymä todetaan suoritetuksi. Mikäli ilmanopeus on siirtymän jälkeen liian alhainen, autopilotissa oleva sakkauksenesto alkaa työntämään koneen nokkaa alaspäin, jotta ilmanopeus saadaan mahdollisimman nopeasti riittävän suureksi. Tämä havaittiin myös toimeksiantajan lentolokeista, jossa ilmanopeus oli siirtymävaiheessa alhaisempi kuin koneen sakkausnopeus. Eräs mahdollisuus miksi tätä asiaa ei ole alun perin huomioitu koodissa on se, että pääsääntöisesti koneet, joissa ArduPilotia käytetään, ovat huomattavasti kevyempiä kuin toimeksiantajan kone ja tästä syystä ne saavat kerättyä siirtymän aikana riittävän ilmanopeuden. Jotta tämä ongelma saataisiin korjattua, oli yksi asia melko selkeä. Koneen ilmanopeutta pitäisi saada kasvatettua siirtymän aikana, jotta koneen nokka ei enää tekisi nyökkäystä alaspäin siirtymän päättyessä. Ensimmäinen idea kasvattaa ilmanopeutta oli pakottaa kone kiipeämään suoraan ylöspäin, asetuksiin määritettävän ajan verran siirtymän alussa. Ensiksi aika kovakoodattiin, jotta opinnäytetyön tekijä pystyi testaamaan, että idea oli toimiva. Aika jonka kone kiihdyttäisi suoraan ylöspäin haluttiin kuitenkin helposti säädettäväksi parametriksi, joten seuraava vaihe oli luoda uusi parametri tätä varten.

Uusi parametri QuadPlane-luokalle lisättiin ensin luomalla sille oma muuttuja quadplane.h tiedostoon ja tämän jälkeen käyttämällä ”AP_GROUPINFO” makroa, joka rekisteröi parametrin ArduPilotiin. Makrolle kerrotaan parametrin nimi, eräänlainen indeksi (aina edellisen parametrin indeksiä yhden suurempi), luokan nimi, jonka sisältä parametri löytyy, muuttuja, josta parametrin arvo voidaan lukea sekä oletusarvo. Parametrin nimeksi annettiin ”TAILSIT_VACCT”. Parametriin tulee kuitenkin viitata nimellä ”Q_TAILSIT_VACCT”, sillä ArduPilot lisää ”Q_” merkkijonon automaattisesti parametrin alkuun, kun parametri lisätään QuadPlane-luokalle. Parametri näkyy ”Q_” alkuisella nimellään tavalliselle käyttäjälle esimerkiksi Mission Plannerin parametrien muokkausnäkymässä. Maksimipituus parametrien nimelle on 16 merkkiä. Kuten kuviosta 29 näkyy, parametreille voidaan antaa kommenttien muodossa tietoja, jotka helpottavat niiden käyttöä ja käyttötarkoituksen ymmärtämistä. Joitakin näitä tietoja käytetään myös parametrien alustamiseen, kun ArduPilot käynnistyy.

```
// @Param: TAILSIT_VACCT
// @DisplayName: Vertical acceleration time
// @Description: Time to accelerate vertically in milliseconds when transitioning from vtol to fw
// @Range: 0 10000
// @Units: millisecond
// @Incerement 1
// @User: Standard
AP_GROUPINFO("TAILSIT_VACCT", 21, QuadPlane, tailsitter.vertical_acceleration_time, 2000),
```

Kuvio 29. Itse lisätty parametri quadplane.cpp tiedostossa

Lopullinen ratkaisu ongelmaan oli lopulta melko yksinkertainen. Siirtymän aikana autopilotti asetti moottorien tehoksi leijuntatehon. Koneen nopeutta voitaisiin siis yksinkertaisesti kasvattaa lisäämällä tehoa, jolla siirtymä suoritetaan ja sen lisäksi pidentämällä siirtymän kestoa. Ratkaisuna tähän oli täten lisätä uusi parametri, johon voitiin määrittää teho, jolla siirtymä suoritettaisiin. Siirtymän keston pystyi jo määrittämään asetuksiin, joten suurempia muutoksia ei vaadittu.

Yksi pieni ongelma tosin havaittiin vielä uusien parametrien lisäysten jälkeenkin. Kone suoritti siirtymän edelleen liian nopeasti. Tämä johtui siitä, että koodissa oli tehty toinenkin tarkistus katsomaan siirtymä suoritetuksi. Käytännössä, jos ”tilt rotor”-tyyppisen koneen moottorit olivat suorassa siipeen nähden eikä kone enää odottanut ilmanopeuden kasvua, siirtymä voitiin todeta suoritetuksi (ks. kuvio 30).

```
// if rotors are fully forward then we are not transitioning,
// unless we are waiting for airspeed to increase (in which case
// the tilt will decrease rapidly)
if (tiltrotor_fully_fwd() && transition_state != TRANSITION_AIRSPEED_WAIT) {
    transition_state = TRANSITION_DONE;
    transition_start_ms = 0;
    transition_low_airspeed_ms = 0;
}
```

Kuvio 30. Tiltrotor tyyppisen koneen tarkistus lentotilan vaihdoksen suoritukselle

Tämä on looginen päätelmä niissä QuadPlane-koneissa, joissa moottorit kääntyvät siiven kanssa yhdenmukaisiksi vasta, kun kone vaihtaa lentotilaa VTOL-tilasta tavalliseen, mutta tailsitter-koneissa moottorit ovat lennon aikana pääsääntöisesti aina siiven kanssa yhdensuuntaiset ja täten siirtymä todetaan suoritetuksi liian aikaisin. Tailsitter-koneet eivät myöskään koskaan odota ilmanopeutta siirtymän päättämiseksi, vaan ne odottavat, että koneen nokka saavuttaa tavoitellun kulman siirtymälle. Ratkaisu ongelmaan oli jättää tämä tarkistus tekemättä, mikäli kone on tyypiltään tailsitter.

Näillä muutoksilla kone saatiin keräämään suurempi ilmanopeus siirtymän aikana. Muutokset eivät takaa sitä, että kone saavuttaa siirtymän lopuksi sakkausnopeuden, mutta koneen käyttäjällä on näillä muutoksilla huomattavasti parempi mahdollisuus hallita sitä. Huomioitavaa on myös se, että QuadPlane-tiloissa koneet on suunniteltu lentämään matalammalla nopeudella, joten liian suuren ilmanopeuden tavoittelu koneen ollessa jossain QuadPlane-tilassa ei ole viisasta. Tämä johtuu siitä, että QuadPlane-tiloissa koneen ohjainpintojen liikkeitä ei skaalata samalla tapaa aluksen nopeuden mukaan, kuin tavallisissa lentotiloissa ja suurilla nopeuksilla kone saattaa muuttua epävakaaksi.

Toinen ja kaikista kriittisin ongelma oli koneen suorittaessa siirtymää tavallisesta lentotilasta johonkin QuadPlane-lentotilaan. Simulaattorissa kone ei menettänyt hallintaa siirtymän lopussa toisin kuin toimeksiantajan kone, mutta se muuttui kuitenkin hetkellisesti epävakaaksi. Ongelma tuli esiin vain autopilotin lentäessä tehtäviä ja aluksi syytä epävakaudelle oli vaikeaa käsittää. Kun samaa tehtävää tuli lennettyä simulaattorissa useita kertoja ja tarkastelemalla koneen eri lokitietoja, alkoi syy pikku-

hiljaa selkeytyä. Kone alkoi heti siirtymän jälkeen tekemään käännöstä kohti seuraavaa tehtävän pistettä, mikä saattoi tehdä koneesta epävakaan, mikäli se oli samaan aikaan myös vähentämässä korkeuttaan. Ongelman ratkaisemiseksi oli täten päämääränä saada kone leijumaan hetki paikallaan siirtymän päätteeksi.

Kuviossa 31 nähdään varsinainen logiikka, jolla kone saadaan leijumaan paikallaan lentotilan vaihdoksen päätteeksi. Siirtymä todetaan stabiloiduksi, kun korkeuden muutos on pysynyt sekunnin ajan +- 0.75 m/s sisällä, tai jos siirtymän stabilisointi on kestänyt yli kahdeksan sekuntia siirtymän päättymishetkestä; kahdeksan sekuntin rajalla varmistetaan, että kone ei jää tähän tilaan, jos kone ei jostain syystä tavoita rajaarvoa korkeuden muutosnopeudelle – esimerkiksi jos korkeusanturi sattuu vioittumaan. Konetta leijutetaan paikallaan hyödyntämällä olemassa olevaa QLOITER-lentotilaa. Kutsumalla metodia ”control_loiter” tätä lentotilaa voidaan käyttää koodissa. Metodille annetaan ylimääräisenä parametrina arvo ”true”, mikä kertoo metodille sen, että koneen korkeus halutaan pitää koko ajan samana. Koneen aloittaessa ns. siirtymän stabilisoinnin, kutsutaan ”init_loiter” metodia, joka mm. tallentaa koneen sijainnin ja korkeuden muistiin, jotta autopilotti tietää missä pisteessä sen tulee pyrkiä pysymään.

```
/**
 * hover around for a while after transition from fw to vtol to stabilize the aircraft.
 * @returns true if we are stabilizing the plane, otherwise false.
 */
bool QuadPlane::run_stabilize_transition(void) {      You, seconds ago • Uncommitted changes
    uint32_t now = AP_HAL::millis();

    // Transition must be finished before any stabilization is done
    if (transition_stabilization.is_stabilized || in_tailsitter_vtol_transition(now)) {
        return false;
    }

    if (fabsf(inertial_nav.get_velocity_z()) > 75.0f) {
        transition_stabilization.last_wait_at = now;
    }

    // We have stabilized the aircraft if the z velocity has stayed within limits for a second or we try to stabilize too long
    if (now - vtol_transition_finished_ms > 8000 || now - transition_stabilization.last_wait_at > 1000) {
        transition_stabilization.is_stabilized = true;
        GCS_SEND_TEXT(MAV_SEVERITY_DEBUG, "stabilization took: %d ms", now - vtol_transition_finished_ms);
        return false;
    }

    if (!transition_stabilization.is_initialized) {
        init_loiter();
        // init_loiter sets target altitude to where it was initiated by default, so we override it to keep the plane climbing up
        pos_control->set_target_to_stopping_point_z();
        transition_stabilization.is_initialized = true;
    }

    control_loiter(true);

    return true;
}
```

Kuvio 31. Logiikka siirtymän stabilisointiin

Kuviossa 32 nähdään ”run_stabilize_transition” metodin käyttö koneen lentäessä tehtävää. Siirtymät stabilisoidaan vain, jos kone on tyypiltään tailsitter. Muuta ”waypoint_controller” metodin logiikkaa ei suoriteta, ennen kuin siirtymä on todettu stabilisoiduksi. Tämän tarkoituksena on pitää kone paikallaan siirtymän jälkeen.

```
/*
  run waypoint controller between prev_WP_loc and next_WP_loc
 */
void QuadPlane::waypoint_controller(void)
{
    // if we are tailsitter transitioning from fw to vtol, we must first stabilize the plane
    if (is_tailsitter() && run_stabilize_transition()) {      You, seconds ago • Uncommitted changes
        return;
    }
```

Kuvio 32. Siirtymän stabilisoinnin käyttö lähdekoodissa

Siirtymää suorittaessaan autopilotti pyrkii kääntämään koneen suoraan ja nostamaan nokan kohti taivasta mahdollisimman nopeasti. Jos kone on vielä kaartamassa lentotilaa vaihtaessa, autopilotti alkaa nostaa koneen nokkaa samalla kun se pyrkii kääntämään koneen suoraan. Tilanne, jossa kone aloittaa siirtymän suoraan käännöksestä, ei ole välttämättä kovin yleinen, mutta kuitenkin täysin mahdollinen. Siirtymän aloitukseen tehtiin yksi muutos, jolla pyritään parantamaan siirtymän luotettavuutta vähentämällä koneen sakkausriskiä. Koska koneen kallistuskulma vaikuttaa sen sakkausnopeuteen, on loogista pitää kone suorassa siirtymän aikana. Mitä enemmän kone kallistuu, sitä vähemmän siivet tuottavat nostovoimaa ja sitä todennäköisemmin kone sakkaa. Lähdekoodiin tehdyllä muutoksella autopilotti saa siirtymää aloittaessaan puoli sekuntia aikaa ohjata koneen suoraan, ennen kuin se alkaa ohjaamaan koneen nokkaa kohti taivasta (ks. kuvio 33).

```
uint32_t now = AP_HAL::millis();
if (quadplane.in_tailsitter_vtol_transition(now)) {
    /*
      during transition to vtol in a tailsitter try to raise the
      nose rapidly while keeping the wings level.
      we wait 500 ms before pulling up to make sure wings have time to get level
     */
    if (now - quadplane.transition_start_ms > 500) {
        nav_pitch_cd = constrain_float((quadplane.tailsitter.transition_angle+5) * 100, 5500, 8500);
    }
    else {
        nav_pitch_cd = ahrs.pitch_sensor;
    }

    nav_roll_cd = 0;
}
```

Kuvio 33. Koneen nokan ohjaus kohti taivasta siirryttäessä tavallisesta lentotilasta VTOL-tilaan

Edellä mainituilla muutoksilla koneen siirtymä saatiin melko hyväksi, mutta kone teki vielä siirtymän aikana epämääräisen kiihdytyksen eteenpäin ennen kuin siirtymän stabilisointi alkoi. Kun kone suorittaa siirtymää, sitä ei pysty manuaalisesti ohjaamaan millään tapaa. Syyksi paljastui loppujen lopuksi se, että lentäessään tehtäviä, autopilotti suoritti siirtymää samaan aikaan, kun se ohjasi itseään kohti seuraavaa pistettä. Tämän ei missään tapauksessa tulisi tapahtua, vaan koneen pitäisi suorittaa siirtymä ensin loppuun ja tämän jälkeen antaa kontrolli tehtävien lentämiseen. Kaikissa muissa lentotiloissa koneen ohjaus on estetty siirtymien aikana kaikelta muulta paitsi siirtymää ohjaavalta logiikalta. Autopilotin ”AUTO” tilan ohjaus oli kuitenkin tehty poikkeuksellisesti kaikista muista lentotiloista, eikä siinä oltu huomioitu sitä, että konetta ei pitäisi ohjata siirtymien aikana. Kuviossa 34 nähdään, että koneen ollessa jossain QuadPlane-lentotilassa, sitä ohjataan kutsumalla ”control_run” metodia, mikäli kone ei suorita siirtymää sillä hetkellä. Metodi ohjaa konetta sen hetkisen lentotilan mukaisesti. Koneen ollessa ”AUTO” tilassa metodi ei tee mitään, vaan koneen ohjaus suoritetaan toista kautta.

```
if (control_mode == &mode_training) {
    stabilize_training(speed_scaler);
} else if (control_mode == &mode_acro) {
    stabilize_acro(speed_scaler);
} else if ((control_mode == &mode_qstabilize ||
            control_mode == &mode_qhover ||
            control_mode == &mode_qloiter ||
            control_mode == &mode_qland ||
            control_mode == &mode_qrtl ||
            control_mode == &mode_qacro ||
            control_mode == &mode_qautotune) &&
           !quadplane.in_tailsitter_vtol_transition()) {
    quadplane.control_run();
```

Kuvio 34. QuadPlane-koneen ohjauslogiikkaa

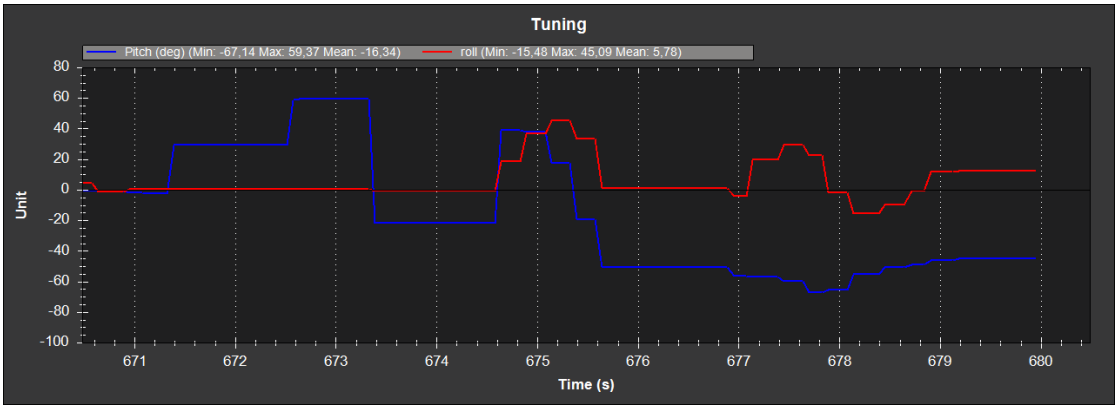
Lisäämällä kuviossa 35 näkyvän tarkastuksen, jossa tarkastetaan koneen ollessa ”AUTO” tilassa, onko tailsitter-tyyppinen kone siirtymässä QuadPlane-lentotilaan, koneen liikkuminen kohti seuraavaa pistettä siirtymän aikana saatiin loppumaan. Käytännössä kaikki ”control_auto” metodin muun logiikan suorittaminen estetään, jos kone on suorittamassa siirtymää. Metodin tehtävänä on QuadPlane-tyyppisten koneiden ohjaus koneen ollessa ”AUTO”-lentotilassa.

```
2643
2644  /*
2645    handle auto-mode when auto_state.vtol_mode is true
2646   */
2647  void QuadPlane::control_auto(void)
2648  {
2649+     if (!setup() || in_tailsitter_vtol_transition(AP_HAL::millis())) {
2650          return;
2651      }
2652
```

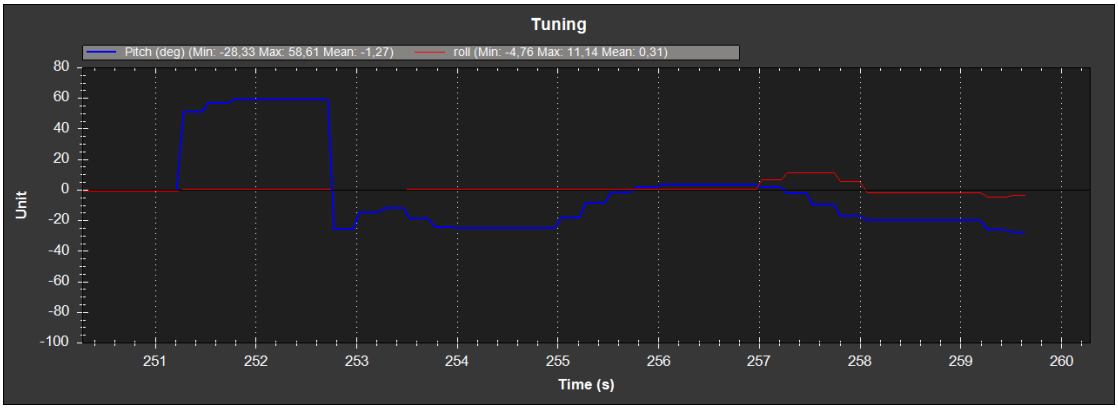
Kuvio 35. Lisätty tarkistus, jolla estetään koneen liikkuminen siirtymän aikana

Mission Plannerilla kaapatuissa kuvioissa 36 ja 37 voidaan havaita muutosten vaikutus siirtymään. Sininen käyrä kuviossa edustaa koneen nousukulmaa ja punainen kallistuskulmaa. Koneen siirtymä alkaa kuvioissa siitä, kun koneen nousukulma kasvaa ensimmäisen kerran. Kuviossa 36 siirtymä alkaa **671** ja **672** sekunnin välillä. Kuviossa 37 siirtymä alkaa **251** ja **252** sekunnin välillä. Siirtymä päättyy, kun koneen nousukulma putoaa negatiiviseksi. Kuviossa 36 siirtymä päättyy **673** ja **674** sekunnin välillä. Kuviossa 37 siirtymä päättyy **252** ja **253** sekunnin välillä. Kuvioita vertaillessa voidaan

havaita, että kone saavuttaa muunnellulla koodilla tavoitekulman nopeammin ja käyttäytyy siirtymän jälkeen rauhallisemmin.



Kuvio 36. Siirtymä VTOL-tilaan alkuperäisellä lähdekoodilla



Kuvio 37. Siirtymä VTOL-tilaan modifioidulla lähdekoodilla

5 Pohdinta

Opinnäytetyön tavoitteena oli perehtyä ArduPilot-autopilotin kehitykseen ja testaukseen sekä toimeksiantajan havaitsemiin ongelmiin autopilotissa. Opinnäytetyön pohjalta syntyi dokumentaatio, joka esittelee yleisimmät ArduPilotin tarjoamat ominaisuudet ja käyttötarkoitukset sekä sen kehitykseen ja testaukseen käytetyt menetelmät. Opinnäytetyössä saatiin selville, miksi lennokin autopilotissa havaittiin ongelmia ja nämä ongelmat saatiin myös korjattua.

Aiheena opinnäytetyö oli haastava, mutta mielekäs. Aiheesta omalla tavalla mielekkään teki pelkästään jo se, että se ei ole kovin tyypillinen opinnäytetyön aihe. Oli mukavaa päästä tutkimaan ja kehittämään jotain uutta. Työn aikana pääsin syventämään omaa osaamistani C++ -ohjelmointikielen parissa, mikä on ollut tavoitteenani jo pitkään. Myös ilmailu on aina ollut yksi itseäni kiinnostava asia. Yksi opinnäytetyön tekemistä edistävä tekijä oli varmastikin oma noin 14 vuoden kokemukseni radio-ohjattavista lennokeista sekä noin 6 vuoden kokemukseni multikoptereista. Ennen opinnäytetyön aloittamista minulla ei ollut omaa kokemusta isoista avoimen lähdekoodin projekteista. Aiempaa kokemusta minulla oli lähinnä vain omien pienempien projektien kehityksestä, jossa kaikki on pääsääntöisesti tehty itse ja lähdekoodi on tuttua. Tähän projektiin lähtiessä uutta itselleni oli siis lähteä muokkaamaan joidenkin täysin tuntemattomien henkilöiden kirjoittamaa lähdekoodia, josta itselläni ei vielä juurikaan ollut mitään käsitystä. Tämä aiheutti alkuun pientä epävarmuutta siitä, kuinka hyvin oppisin tuntemaan lähdekoodin rakenteen – jota ArduPilot-autopilotissa on jopa yli 700 000 riviä. Tiesin kuitenkin, ettei minun tarvitsisi tuntea lähdekoodia läpikotaisin, joten päätin ottaa haasteen vastaan.

Yksi iso kysymys työn alkuvaiheissa oli se, kuinka hyvin koodia pystyttäisiin testaamaan. Jos koodia ei voitaisi testata ilman oikeaa ilma-alusta, olisi se erittäin riskialtista ja hankalaa. Testaus päädyttiin tekemään simulaattorin avulla, mikä tosin vaati jonkin verran työtä sen toimintaan saamiseksi. Simulaattorin toimintaan saaminen toi paljon lisää itseluottamusta työn tekoon.

Ehkä suurin ongelma työn aikana oli se, että ongelmat eivät tulleet samalla tapaa esiin simulaattorissa kuin oikeassa koneessa. Simulaattorissa käytetty kone oli tyypiltään samanlainen kuin oikeakin kone, mutta esimerkiksi painoltaan ja kooltaan se oli erilainen kuin toimeksiantajan kone. Simulaattorissa ongelmat esiintyivät huomattavasti paljon lievemmällä tavalla ja vasta useiden lentojen ja erilaisten lentostrategioiden jälkeen niitä pystyi jollain tapaa havaitsemaan, kun taas toimeksiantajan koneessa ne olivat erittäin selkeästi esillä. Koneen siirtyessä VTOL-tilasta tavalliseen lentotilaan ei simulaattorissa edes havaittu ongelmia lennon aikana. Ongelma ratkaistiin tietämällä, kuinka ArduPilotin jotkin ominaisuudet toimivat. ArduPilotin dokumentaatiota lukeneena tiesin sen, että autopilotin sakkauksenesto työntää koneen nokkaa alaspäin ilmanopeuden ollessa liian matala, minkä avulla pystyimme toimeksiantajan kanssa selvittämään, että myös toimeksiantajan koneen ilmanopeus oli siirtymävaiheessa liian matala ja sitä pitäisi saada suuremmaksi. Vaikka ongelma ei esiintynyt simulaattorissa, koodiin pystyttiin tekemään muutoksia ongelman korjaamiseksi. Muutokset pystyttiin testaamaan simulaattorissa ja varmistamaan, että ne eivät aiheuttaneet odottamatonta käytöstä.

Se miksi työssä esiteltyjä ongelmia ylipäätään kohdataan ArduPilotin kanssa, johtuu hyvin todennäköisesti siitä, että monet ArduPilotia käyttävät koneet ovat kevyempiä kuin toimeksiantajan kone ja ongelmat eivät tule niissä esiin samalla tapaa tai niin helposti.

Jos katsotaan kappaleessa 4.6.2 esitettyjä muutoksia lähdekoodiin, ne eivät välttämättä vaikuta kovin suurilta, mutta näiden muutosten tekemiseen kului kuitenkin hyvä tovi aikaa – enemmän kuin voisi ensi silmäyksellä kuvitella. Ennen kuin ongelmia voidaan alkaa korjaamaan, täytyy niiden syy myös ymmärtää. Opinnäytetyön työmäärää ja haastavuutta ei mielestäni ole helppo tuoda työssä esiin, koska suuri osa kehitystyöstä kului pelkästään koodin analysointiin ja debuggaukseen, mitä on vaikea havainnollistaa. Hyvin pieneltäkin vaikuttavat ongelmat kuitenkin vaativat välillä usean päivän työn. Opinnäytetyö vaati siis paljon pitkäjänteisyyttä – tosin samaa voidaan sanoa ohjelmistokehityksestä yleisestikin.

Lopputuloksena opinnäytetyössä onnistuttiin ehkä jopa paremmin mitä uskalsin edes odottaa. Tiesin aiheen haastavaksi ja työn alkuvaiheilla sen onnistuminen aiheutti joitain epäluuloja. Epäluuloisuutta aiheutti se, kykenisinkö ratkaisemaan toimeksiantajan ongelmia, mutta loppujen lopuksi opinnäytetyön tavoitteet saavutettiin mielestäni kirkkaasti.

Opinnäytetyö oli toimeksiantajan lisäksi hyödyllinen myös minulle, sillä opin työtä tehdessä itsekin paljon uutta. ArduPilotin lähdekoodia lukemalla ja muokkaamalla opin uusia hyviä ohjelmointikäytänteitä ja menetelmiä. Tämä oli myös yksi aiheen valintaan vaikuttanut tekijä, sillä olin kuullut, että avoimen lähdekoodin projektit ovat hyvä lähde ohjelmointitaitojen kehittämiseen ja tämän voin jälkeenpäin todeta myös itse. Etenkin isot avoimen lähdekoodin projektit sisältävät useiden eri henkilöiden kirjoittamaa koodia, joten lähdekoodiin tutustumalla pääsee näkemään monipuolista koodia, josta voi yleensä poimia itselleen uusia tekniikoita ja menetelmiä. Työn aikana sain myös paljon uutta tietoa ArduPilotista ja uskon että tulevaisuudessa se löytää vielä tiensä johonkin omaankin projektiini toimittamaan autopilotin virkaa. Kokonaisuutena olen työhön ja työpanokseeni erittäin tyytyväinen.

Lähteet

About ArduPilot. N.d. Viitattu 2.10.2020. https://ardupilot.org/index.php/about

ArduPilot partners. N.d. ArduPilot. Viitattu 2.10.2020. https://ardupilot.org/about/Partners

Austin, R. 2010. Unmanned aircraft systems: UAVS design, development, and deployment. Chichester: Wiley.

Automatic landing. N.d. ArduPilotin dokumentaatio. Viitattu 6.10.2020. https://ardupilot.org/plane/docs/automatic-landing.html

Automatic takeoff. N.d. ArduPilotin dokumentaatio. Viitattu 3.10.2020. https://ardupilot.org/plane/docs/automatic-takeoff.html

Cygwin FAQ. N.d. Viitattu 25.12.2020. https://cygwin.com/faq.html

Drone technology uses and applications for commercial, industrial and military drones in 2021 and the future. 2021. Viitattu 20.1.2021. https://www.businessinsider.com/drone-technology-uses-applications

Flying a QuadPlane. N.d. ArduPilotin dokumentaatio. Viitattu 18.10.2020. https://ardupilot.org/plane/docs/quadplane-flying.html

Inverted flight. N.d. ArduPilotin dokumentaatio. Viitattu 6.10.2020. https://ardupilot.org/plane/docs/inverted-flight.html

Lentola Logistics Oy. N.d. Toimeksiantajan kotisivut. Viitattu 14.1.2021. https://lentola.com

MAVProxy. N.d. ArduPilotin dokumentaatio. Viitattu 27.10.2020. https://ardupilot.org/mavproxy/

Minilennokkijärjestelmä N.d. Maavoimat. Viitattu 3.10.2020. https://maavoimat.fi/muas

Mission Planner. N.d. ArduPilotin dokumentaatio. Viitattu 10.10.2020. https://ardupilot.org/planner/docs/mission-planner-overview.html

Multirotor drones. N.d. Viitattu 18.1.2021. https://www.unmannedsystemstechnology.com/category/supplier-directory/platforms/multirotor-drones/

Pixhawk 4. N.d. Viitattu 26.12.2020. https://shop.holybro.com/pixhawk-4_p1089.html

Quadplane overview. N.d. ArduPilotin dokumentaatio. Viitattu 12.10.2020. https://ardupilot.org/plane/docs/quadplane-overview.html

Realflight 8 and Mission planner Missing feature. ArduPilotin keskustelufoorumi 18.1.2020. Viitattu 13.10.2020. https://discuss.ardupilot.org/t/realflight-8-and-mission-planner-missing-feature/51279

SITL with RealFlight. N.d. ArduPilotin dokumentaatio. Viitattu 13.10.2020. https://ardupilot.org/dev/docs/sitl-with-realflight.html

SITL. N.d. Viitattu 18.10.2020. ArduPilotin dokumentaatio. https://ardupilot.org/dev/docs/sitl-simulator-software-in-the-loop.html

Stall prevention. N.d. ArduPilotin dokumentaatio. Viitattu 10.10.2020. https://ardupilot.org/plane/docs/stall-prevention.html

Style guide. N.d. ArduPilotin dokumentaatio. Viitattu 12.12.2020. https://ardupilot.org/dev/docs/style-guide.html

Tailsitter Planes. N.d. ArduPilotin dokumentaatio. Viitattu 11.10.2020. https://ardupilot.org/plane/docs/guide-tailsitter.html

Tilt Rotor Planes. N.d. ArduPilotin dokumentaatio. Viitattu 24.10.2020. https://ardupilot.org/plane/docs/guide-tilt-rotor.html

UAV Autopilot systems. N.d. Viitattu 26.12.2020. https://www.unmannedsystemstechnology.com/category/supplier-directory/electronic-systems/autopilot-systems/

Visual Studio Code. 2020. Viitattu 25.12.2020. https://code.visualstudio.com/

What is Open Source. N.d. Viitattu 6.1.2021. https://opensource.com/resources/what-open-source

What is debugging. 2018. Viitattu 18.12.2020. https://docs.microsoft.com/en-us/visualstudio/debugger/what-is-debugging?view=vs-2019

Yunyi, J., Longxiang, G & Xin, W. 2018. Real-time control systems. Transportation Cyber-Physical Systems. 81-113. Viitattu 12.12.2020. https://www.sciencedirect.com/topics/earth-and-planetary-sciences/automatic-pilot