Janne Pakarinen

Lennonautomaatiojärjestelmän kehitysympäristö

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Sähkö- ja automaatiotekniikka

Insinöörityö

15.11.2021

Tiivistelmä

Tekijä:	Janne Pakarinen
Otsikko:	Lennonautomaatiojärjestelmän kehitysympäristö käyttöönotto
Sivumäärä:	29 sivua + 2 liitettä
Aika:	15.11.2021
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Sähkö- ja automaatiotekniikka
Ammatillinen pääaine:	Automaatiotekniikka
Ohjaajat:	Lehtori Timo Kasurinen

Opinnäytetyön päämäärä oli valita joustava kehitysympäristö kokeellisen ilma-aluksen lennonautomaatiojärjestelmälle. Työ oli osa monipuolista ilma-alusjärjestelmää kehittävää isäntäprojektia. Kohdelaitteisto on isäntäprojektissa kehitetty ilma-aluksen prototyyppi. Vaatimuksina kehitysympäristölle ovat mallipohjainen kehitystyö ja nopeatahtinen työnkulku. Työssä vertaillaan ja pohditaan kolmen vaihtoehtoisen alustan soveltuvuutta kehitysalustaksi.

Soveltamalla mallipohjaista suunnittelutapaa, jossa suuri osa testaamisesta toteutetaan simuloimalla fyysisen testaamisen sijaan, säästyy huomattavia määriä aikaa ja resursseja verrattuna perinteiseen suunnittelutapaan. Kehitysympäristöksi valitut kaupalliset Mathworks MATLAB ja Laminar research X-Plane muodostavat tehokkaan ympäristön tarjoten monipuoliset kehitys- ja simulaatiotyökalut.

Ilma-aluksen automatisoitu lento ja vakaan lennon saavuttaminen ovat suunniteltavalle ohjausjärjestelmälle asetetut tavoitteet. Kehitysympäristössä kehitettävän ohjausjärjestelmän tulee säilyttää vakaa ja turvallinen lentotila pitämällä lennon parametrit aerodynaamisen suorituskyvyn rajoissa kaikissa lennon vaiheissa.

Tässä työssä kuvaillaan kehitysympäristön käyttöönottoa ja paneudutaan työnkulkuun. Mallipohjaisen järjestelmäsuunnittelun ja nopeatahtisen työnkulun tuomat edut näkyvät kehitystyön tehokkuudessa.

Työn tuloksena valittiin kehitysympäristö, joka soveltuu mallipohjaiseen kehitystyöhön ja on joustava tulevien sovellusten asettamiin tarpeisiin. Ympäristön työnkulku on nopeaa ja selkeää. Isäntäprojektin etenemisen kannalta lennonautomaatiojärjestelmän kehitystyön sujuvuus on ensiarvoisen tärkeää.

Avainsanat: lento, automaatio, ilma-alus, lennonautomaatiojärjestelmä, mallipohjainen järjestelmäsuunnittelu, kehitysympäristö

Abstract

Author:	Janne Pakarinen
Title:	Flight Automation System Development Environment
Number of Pages:	29 pages + 2 appendices
Date:	15 November 2021
Degree:	Bachelor of Engineering
Degree Programme:	Electrical and Automation Engineering
Professional Major:	Automation Engineering
Supervisors:	Timo Kasurinen, Senior Lecturer

The aim of the thesis work was to select and set up a rapid prototyping development environment for a flight control automation system. Requirements for the environment were model based system engineering methods and a fast-paced workflow.

A comparison between three different suitable environments was conducted by studying and testing the tools and workflow the platforms provide.

By following the philosophy of model-based system engineering, where the majority of physical testing is replaced by simulation, the overall development time and required resources are greatly reduced, when compared to traditional control system development. Commercially available X-Plane by Laminar Research and MATLAB/Simulink by Mathworks form a powerful pair for system design and simulation. These were selected to form the basis for the development platform.

Automated flight of an aerial vehicle and achieving stable flight in a dynamic environment are the primary requirements set for the flight control system. The development environment will be used to produce an automated flight control system, which maintains safe and controllable flight by keeping flight parameters within limits set by the aerodynamic performance of the vehicle during all phases of flight.

The work report describes setting up the environment in detail and provides an insight to the workflow in the development environment.

The benefits of model-based system design and the advantages found through swift workflow boost the speed of development significantly, as noticed when setting up and testing the selected environment.

Keywords: flight, control, aircraft, model-based design, development environment

Sisällys

Lyhenteet ja käsitteet

air data: Ilma-alusta ympäröivän ilman staattisen paineen, dynaamisen paineen, kohtauskulman sekä lämpötilan mitatut arvot.

BET: *Blade Element Theory*. Menetelmä siipiprofiilin liikkeestä johtuvien voimien laskentaan.

IMU: *Inertial Measurement Unit*. Mittalaite, jolla mitataan kulmanopeuksia ja kiihtyvyyksiä.

Lentotila: Lentokoneen asennon ja liikeradan hetkellinen tila

LIDAR: *light detection and ranging*. Optinen tutkan tapaan toimiva laite.

MIL: *Model in the loop*. Simulointi, jossa ajetaan suunniteltavan järjestelmän mallia.

PID: *Proportional-Integral-Deriative*. Säätötekniikassa käytetty säätömenetelmä.

PIL: *Processor in the loop*. Simulointi, jossa ajetaan suunniteltavan järjestelmän mallista generoitua ohjelmakoodia mikrokontrollerilla.

SIL: *Software in the loop*. Simulointi, jossa ajetaan suunniteltavan järjestelmän mallista generoitua ohjelmakoodia.

UDP: *User Datagram Protocol*. Verkkoprotokolla.

VTOL: *Vertical takeoff and landing.* Pystysuoraan tapahtuva lentoonlähtö ja laskeutuminen.

1 Johdanto

Tämän opinnäytetyön päämäärä on kehitysympäristön valinta ja käyttöönotto suunnitteluvaiheessa olevan kokeellisen ilma-aluksen miehittämättömän prototyypin lennonohjausjärjestelmän kehitystyötä varten. Työ on osa laajempaa isäntäprojektia, jonka tavoite on kehittää myöhemmässä vaiheessa monipuolinen ilma-alusjärjestelmä.

Lennonohjausjärjestelmän kehitystyön lähtökohtana on tehokas ja nopeatahtinen mallipohjainen suunnittelutapa. Kohteena on pääosin ohjauselektroniikan ohjelmisto, jolla kokeellista ilma-alusta tullaan ohjaamaan. Vaatimuksia valittavalle kehitysympäristölle ovat sujuva työnkulku, monipuoliset työkalut tiedon käsittelyyn, sekä mikrokontrollerin ohjelmakoodin sujuva päivittäminen kehitystyön aikana. Näiden lisäksi valittavan kehitysympäristön tulee olla monipuolinen ja sen soveltamisen tulee sujua joustavasti, jotta tehtävien suorittamiseen liittyvät toiminnot saadaan halutunlaisiksi. Lennonohjausjärjestelmän tärkeimmät tehtävät ovat: ensisijaisesti ilma-aluksen asennon ja liikeradan hallinta sekä sekundäärisesti suorittaa esimerkiksi kohteen seuraamista videokuvan tai muun anturitiedon perusteella.

Kohdelaitteisto on lennonautomaatiojärjestelmän kehitystä varten suunniteltu miehittämätön prototyyppi. Prototyyppi on rakenteeltaan kääntyväroottorinen monikopteri, jossa on myös kiinteät siivet, hyvän matkalentonopeuden ja toiminta-ajan saavuttamiseksi (kuva 1). Prototyypin tulee kyetä lennonohjausjärjestelmän ohjaamana pystysuoraan lentoonlähtöön ja laskeutumiseen (VTOL) monikopterin tavoin, sekä toteuttamaan kiinteäsiipisessä lentotilassa ennalta määritetty lentosuunnitelma sekä transitio lentotilojen välillä. Prototyypin avulla on tavoitteena suunnitella komission täytäntöönpanoasetuksen (EU) 2019/947:n mukaisen ”erityinen” -kategorian mukainen ilma-alus (1).

Kuva 1. Havainnekuva prototyypistä.

Tässä työssä vertaillaan Matlab-, Scilab- ja ROS-ympäristöjen soveltuvuutta lennonautomaatiojärjestelmän kehitysympäristöksi. Samalla pohditaan mikä ympäristöistä tarjoaa parhaimmat työkalut jatkokehitystä varten. Luvussa 2 käsitellään lennonohjausjärjestelmän periaatteita. Sen jälkeen siirrytään tarkastelemaan kehitystyön luonnetta ja kehitysympäristölle asetettuja vaatimuksia, sekä pohditaan mallipohjaista kehitystyötä, sen olemusta ja kulkua tarkemmin. Luku 4 keskittyy lentotapahtuman perusteisiin ja hallintaan. Viidennessä luvussa tutustutaan vaihtoehtoisiin kehitysympäristöihin ja niiden tarjoamiin mahdollisuuksiin. Lopuksi perehdytään lennonohjausjärjestelmän käyttöönottoon ja kehitystyön aloittamiseen.

2 Lennonohjausjärjestelmän periaatteista

Ilma-aluksen vakaa lento edellyttää tarkkaa asennon ja lentoradan hallintaa. Perinteisen kiinteäsiipisen ilma-aluksen eli lentokoneen asentoa hallitaan aerodynaamisilla ohjainpinnoilla. Näitä ovat yleisesti kallistusta ohjaavat siivekkeet siivissä, sekä pyrstöön sijoitetut korkeusperäsin ja sivuperäsin. Lentäjä ohjaa lentokonetta käyttäen näitä ohjainpintoja suhteessa tekemiinsä havaintoihin ympäristöstä ja lentokoneen lentotilasta. Lentäjän on olennaista säilyttää asentotajunsa, eli lentokoneen asento suhteessa maan pintaan, sekä ymmärrys lentokoneen nopeudesta ja liikeradasta ilmamassassa. (2.)

Miehittämättömän ilma-aluksen hallinnan ja automaattisen toiminnan mahdollistamiseksi ilma-alusta ohjaamaan tarvitaan järjestelmä. Anturitietoa käyttämällä järjestelmä vakauttaa ilma-aluksen asennon ja liikeradan eri lentotiloissa, sekä suorittaa reittipistenavigointia.

Miehittämättömien ilma-alusjärjestelmien kehittyessä on syntynyt uudenlaisia mahdollisuuksia tarjota pienoisilma-aluksilla suoritettavaa lentotyötä. Pienoisilma-alukset ovat kehittyessään mahdollistaneet esimerkiksi kuvaus ja kartoituslentojen toteuttamisen ilman raskasta kalustoa. Anturi- ja mikrokontrolleritekniikan kehityksen myötä on saatavilla fyysisiltä mitoiltaan pieniä ja ennen kaikkea kevyitä, sekä helposti kuljetettavia ratkaisuja, jotka soveltuvat hyvin esimerkiksi maastonmittauksiin ja mallintamiseen. Muun muassa Maanmittauslaitos käyttää kaukokartoitustiedon keräämisessä miehittämättömiä ilma-aluksia (3). Mikrokontrollereiden jatkuvasti kasvava laskentateho mahdollistaa epästabiilin ilma-aluksen ohjaus- ja stabilointijärjestelmän rakentamisen pienellä komponenttimäärällä. Tiedon kerääminen ilmasta käsin erilaisia anturitekniikoita käyttäen on lisääntynyt myös muilla aloilla. Ilmakartoituksia käytetään muun muassa maatalouden satokartoituksissa, rakennusteollisuuden maastomalleissa sekä muiden tietomallien tuottamisessa. Miehittämättömiä ilma-aluksia on alettu kehittää ja käyttää myös monenlaisiin kuljetustarkoituksiin, kuten pienten pakettien kohteeseen toimittamiseen (4).

3 Mallipohjainen kehitystyö

Tässä luvussa pohditaan mallipohjaisen kehitystyön konseptia verrattuna perinteiseen järjestelmäkehitykseen. Toimivia ratkaisuja työn tarpeisiin soveltuviksi kehitysympäristöksi on markkinoilla tarjolla useita. Luvussa käsitellään Matlab-, Scilab- ja ROS-kehitysympäristöjen ominaisuuksia, sekä niiden tarjoamien alustojen toimivuutta tämän työn kannalta.

3.1 Kehitysympäristön koostumus

Yksinkertaisimmassa muodossa perinteinen kehitysympäristö voi koostua mikrokontrollerin ohjelmointityökaluista sekä testauksesta fyysisellä prototyypillä suoritetuilla koelennoilla. Tällöin kehitystyö vaatii raskasta dokumentointia, koodaamista ja tuotetun ohjelmakoodin versionhallintaa (4). Myös runsaiden koelentojen toteuttaminen ja näistä saadun tiedon käsittely ja soveltaminen vievät paljon aikaa ja niistä muodostuu kustannuksia. Toistuviin koelentoihin liittyy myös suurentunut vaurioriski prototyypille ja ympäristölle.

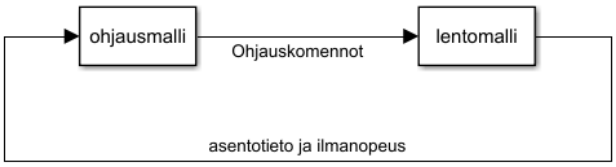
Perinteisen suunnittelufilosofian sijaan järjestelmäsuunnittelussa sovelletaan yhä useammin mallipohjaista lähestymistapaa. Se on käytössä laajasti eri teknologia-aloilla. Mallipohjaisessa kehitystyössä tieto tallennetaan malliin dokumenttien sijaan ja järjestelmää testataan simuloinneilla heti kehitystyön alkuvaiheesta alkaen. (5.)

Mallipohjainen menetelmä on suunnittelufilosofia, jossa tavoiteltu järjestelmä mallinnetaan ja sen toimintaa testataan simuloimalla. Valtaosa kehitystyöstä tehdään simuloidussa ympäristössä ja kehitystyön edetessä siirryttäessä testaamaan fyysisellä laitteistolla järjestelmä on jo pitkälle kehittynyt ja toiminnallinen. Järjestelmäkehitykseen soveltuvissa ympäristöissä testaus suoritetaan pääosin MIL-, SIL- ja PIL-simuloinnilla. Mallipohjainen suunnittelu säästää koelentoihin pohjautuvaan testausmenetelmään nähden huomattavasti aikaa ja vähentää oleellisesti riskejä. Erillistä teknistä dokumentointia ei kehitystyön aikana ole, koska kaikki tieto järjestelmästä on mallissa itsessään. Mallipohjaisesti

toteutetussa suunnitteluprosessissa ratkaisut pohjautuvat toiminnallisiin malleihin järjestelmästä ja sen osista. Mallipohjainen suunnittelu palvelee erinomaisesti tämän työn tarpeita.

3.2 Kehitystyön luonne

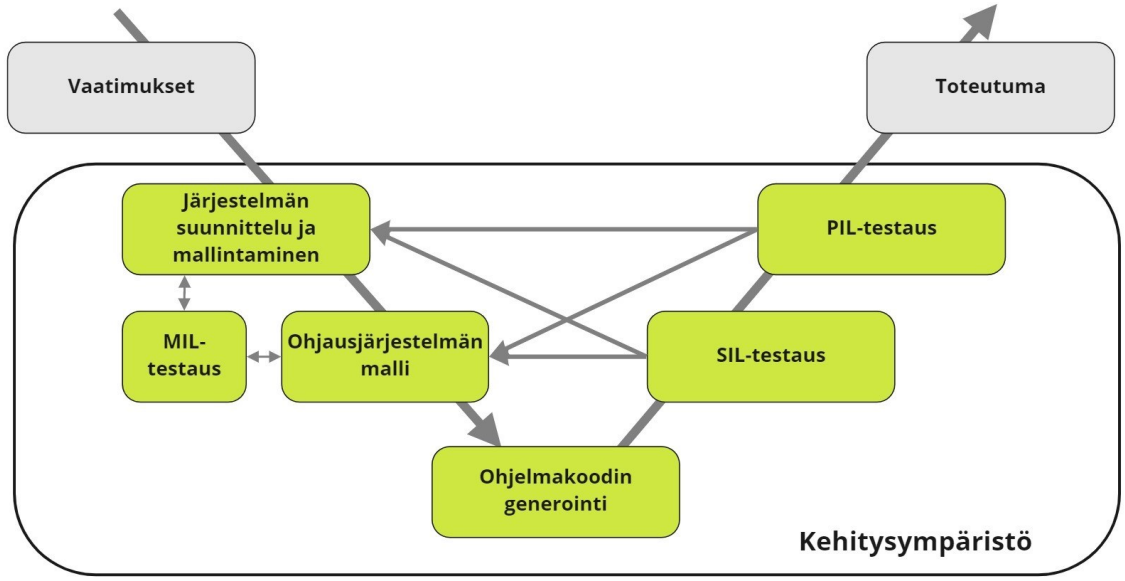
Ohjausjärjestelmän toiminnallisesta mallista generoidaan kehitystyön edetessä ohjelmakoodia suoritettavaksi mikrokontrollerilla. Tässä työssä tavoiteltavassa kehitysympäristössä tehtävä mallipohjainen kehitystyö ei sisällä käytännön tasolla niinkään koodin kirjoittamista, vaan sen sijaan mallintamista ja simulointia. Tätä työtä varten suunniteltiin prototyyppi ja tehtiin sen fyysisiä ominaisuuksia tarkasti vastaava 3d-malli. Voidaan siis todeta, että prototyypin fyysiset ominaisuudet on tallennettu tähän 3d-malliin. Ohjausjärjestelmästä luodaan kehitysympäristössä malli, joka ohjaa mallinnettua prototyyppiä mallinnetussa lentotapahtumassa (kuva 2). Näillä toiminnallisilla malleilla suoritettu simulointi niin ikään mallinnetussa ympäristössä on nopeasti etenevää ja mallien toiminta, sekä mahdolliset virheet ovat näkyvissä jo heti kehitysvaiheessa (6). Kehitystyö sujuu mallien avulla joustavasti, ja muutokset on mahdollista toteuttaa vähäisin kustannuksin.



Kuva 2. Ohjausmallin ja lentomallin kytkentä.

Monimutkaisten järjestelmien kokonaishallinta perinteisiin kehitystapoihin verrattuna on yksinkertaista ja selkeää. Ohjattava ilma-alus mallinnetaan vastaamaan toiminnaltaan ja käytökseltään fyysistä ilma-alusta. Mallinetussa toimintaympäristössä mallia ohjataan ohjausjärjestelmän mallilla (MIL-testaus).

Ohjausjärjestelmä mallinnetaan lohkokaavioilla, joihin perehdytään tarkemmin luvussa 6. Työn edetessä ohjausjärjestelmä kehittyy ja etenee kohti toiminnallisten vaatimusten täyttymistä. Ohjausjärjestelmän mallista generoidaan aina muutoksien jälkeen ohjelmakoodia, jonka toimintaa testataan ohjaamalla koodin avulla ilma-aluksen mallia (SIL-testaus). Kehitystyön edetessä generoitua koodia ajetaan kohdemikrokontrollerilla, jolla ohjataan ilma-aluksen mallia (PIL-testaus) (kuva 3).

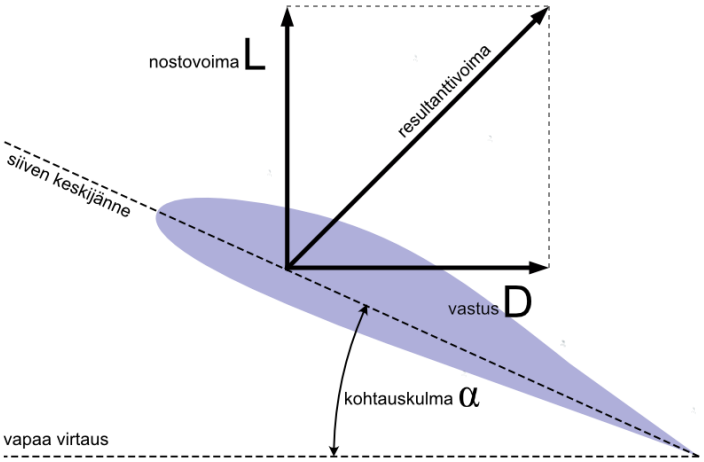


Kuva 3. Kehitystyön periaate kuvattuna V-mallilla.

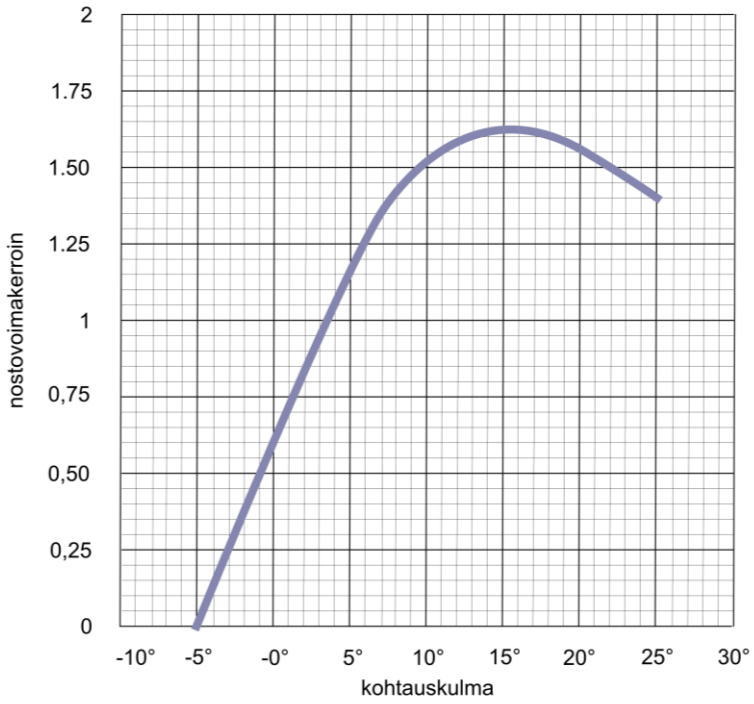
4 Lentotapahtuma ja sen hallinta

4.1 Aerodynamiikan perusteista

Ilmaa raskaamman ilma-aluksen lentämisen mahdollistava nostovoima tuotetaan ilmamassassa liikutettavan siipiprofiilin avulla. Siipi jakaa ilmavirran kulkemaan profiilin ylä- ja alapintoja pitkin ja yhdistää virtaukset jälleen siiven jättöreunassa taittaen virtauksen suuntaa kohdistaen voiman ilmamassaan. Nostovoima L on tämän voiman vastavoima. Siiven jäänteen ja ilmavirtauksen tulosuunnan välistä kulmaa kutsutaan kohtauskulmaksi (kuva 4). Kohtauskulman arvo ja virtauksen nopeus vaikuttavat siiven tuottamaan nostovoimaan (kuva 5). Kohtauskulman kasvaessa nostovoima sekä vastus kasvavat. Virtauksen nopeuden kasvaessa halutun nostovoiman tuottamiseen tarvittavan kohtauskulman arvo pienenee. (7.)

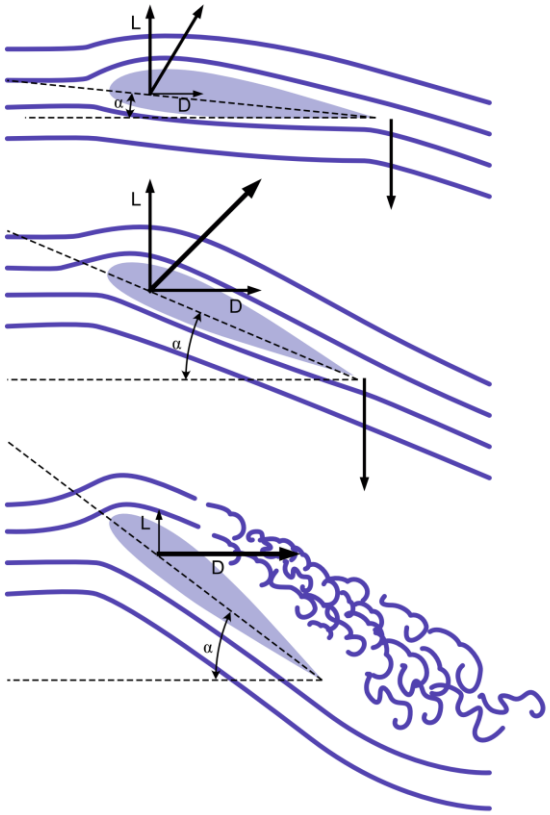


Kuva 4. Kohtauskulma ja nostovoima.



Kuva 5. Kohtauskulman suhde nostovoimaan.

Kohtauskulman kasvaessa virtaus alkaa irrota siiven yläpinnalta – aluksi läheltä jättöreunaa ja lopulta koko yläpinnalta. Tätä tapahtumaa kutsutaan sakkaukseksi ja tällöin nostovoiman tuotto loppuu. Sakkaus johtaa yleensä lentokoneen hallinnan menettämiseen. Sakkaus tapahtuu siipiprofiilikohtaisesti aina samalla kohtauskulman arvolla riippumatta virtauksen nopeudesta (7) (kuva 6).



Kuva 6. Virtauksen irtoaminen siivestä kohtauskulman kasvaessa.

Potkurit ovat myös siipiprofiileja ja ne tuottavat nostovoimaa samalla tavoin kuin pyörimätön kiinteä siipi. Yksinkertaistettuna, potkuria pyörittämällä aikaansaadaan ilmavirtaus potkurin lapojen profiilin yli, jolloin ilmavirtauksen taittumisen vaikutuksesta ilmamassaan kohdistuu voima, jonka vastavoimana saadaan potkurin pyörimisakselin suuntainen työntövoima. Kiinteälapaiset potkurit mitoitetaan käyttötarkoituksensa mukaisesti tietyille etenemisnopeusalueille sekä pyörimisnopeudelle. (7.)

4.2 Ilma-aluksen hallinta

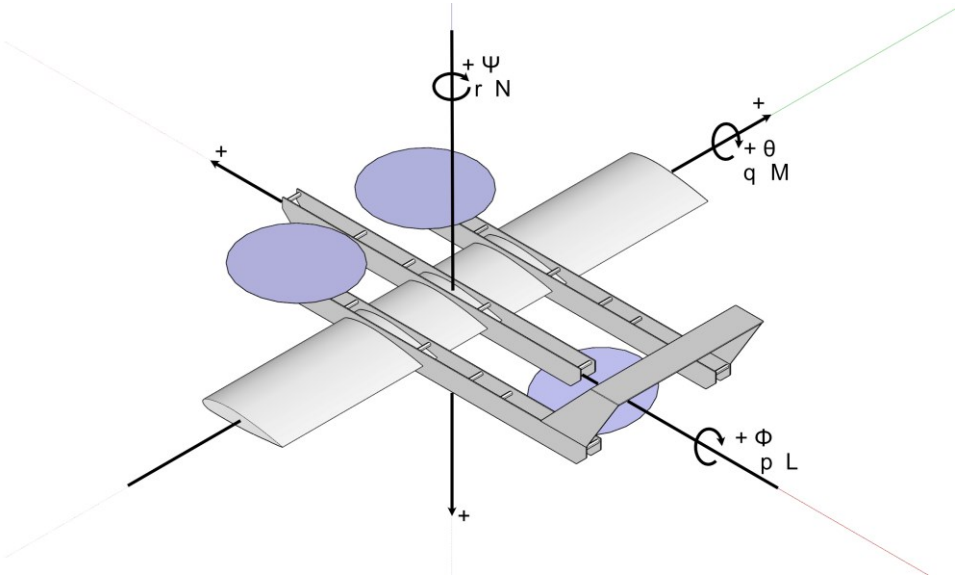
Tässä luvussa käydään läpi työnkulun kannalta tärkeitä seikkoja ilma-aluksen hallintaan vaikuttavista tekijöistä.

Ilma-aluksen hallinnan kannalta on siis oleellista, että siiven kohtauskulman arvot pysyvät lennon aikana sellaisissa rajoissa, että siipi ei sakkaa eikä nostovoiman tuotto lopu. Ilmanopeuden eli sen nopeuden millä siipiprofiili kohtaa vapaan virtauksen hallinta on tärkeää. Erityisesti kiinteäsiipisessä lentotilassa ilmannopeus vaikuttaa suoraan kohtauskulmaan ja ilma-aluksen ohjattavuuteen. Ilma-alusta ohjataan kiinteäsiipisessä lentotilassa muuttamalla nostovoiman symmetriaa tuottaen ohjausmomentit momenttivarren päähän ilma-aluksen painopisteestä sijoitetuilla ohjaussiivekkeillä ja peräsimillä. Pystysuoran laskeutumisen ja lentoonlähdön aikaisessa VTOL-lentotilassa nostovoiman symmetriaa muutetaan potkureiden tuottaman työntövoiman epäsymmetrisellä säädöllä. (8.)

Ilma-aluksen hallintaan tarvittavia mitattavia suureita ovat asentotieto, ilmanopeus, barometrinen korkeus ja kohtauskulma. Asentoa ja sen muutosta mitataan vertaamalla ilma-aluksen runkoon kiinnitetyn koordinaatiston muutosta maan pintaan paikallisesti kiinnitetyn koordinaatiston suhteen (local horizon) (9). Näiden tietojen perusteella lentorataa voidaan ohjata ja samalla huolehtia siitä, että lentotila pysyy ilma-aluksen aerodynaamisen suorituskyvyn asettamissa rajoissa. Tätä työtä varten mallinnetussa prototyypissä mitataan kolmella akselilla kulmanopeus- ja kiihtyvyysarvot. Mittaamiseen käytetään IMU-yksikköä, jonka avulla tarvittavat kulma-arvot saadaan laskettua.

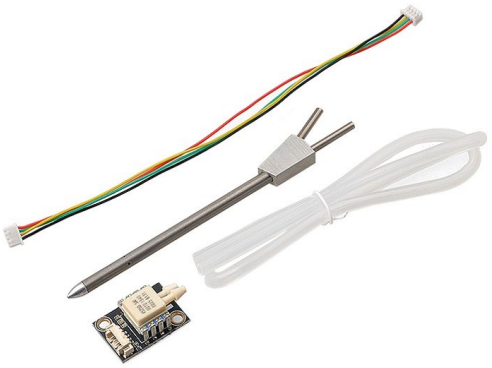
Ilma-aluksen hallinnan mahdollistamiseksi tarkasteltavia suureita ovat seuraavat (kuva 6):

- kulmanopeudet p, q ja r
- kulmat Φ,θ ja Ψ
- momentit L, M ja N.



Kuva 7. Liike koordinaatistossa.

Ilmanopeuden mittaaminen tapahtuu mittaamalla dynaamisen ja staattisen paineen eroa. Mittaamiseen käytetään pitot-putkea ja paine-eroanturia (kuva 8).



Kuva 8. Pitot-putki ja paine-eroanturi (10).

Barometrinen korkeus mitataan käyttäen paineanturia. Kohtauskulman mittaus toteutetaan potentiometrin avulla kiinnittämällä potentiometrin akselille ilmavirrassa kääntyvä evä. Vaihtoehtoisesti barometrisen korkeuden mittaaminen voidaan toteuttaa käyttämällä paine-ero mittausta.

Lennon vaiheet ovat lentoonlähtö (VTOL), matkalento ja laskeutuminen (VTOL). Prototyypin ohjausvoimat tuotetaan lennon VTOL-vaiheissa säätämällä potkureiden työntövoimaa eli kierrosnopeutta, sekä muuttamalla takimmaisen potkurin työntövoiman suuntaa. Potkurin työntövoiman suunnanmuutos toteutetaan kääntämällä moottoritelinettä prototyypin pituusakselin suhteen. Matkalentovaiheessa nostovoima tuotetaan kiinteän siiven avulla. Oleellista lennon toimivuuden kannalta on lennon vaiheiden väliset transitiot eli se, kuinka ilma-aluksen ohjauksessa käytetyt kaksi toisistaan erilaista lento- ja ohjaustapaa vaihtuvat lennon vaiheesta toiseen.

Kiinteäsiipisessä lennonvaiheessa ohjaus tapahtuu siiven ohjaussiivekkeillä (Φ ja p), korkeusperäsimellä (θ ja q) ja sivuperäsimillä (Ψ ja r) (kuva 7). Lentotilojen välisen transition aikana moottoritelineet kääntyvät rungon poikittaisakselin suuntaisesti noin 90 astetta. Ohjauksen kannalta prototyypin teknisen järjestelyn pääpiirteet ovat kaksi yhdellä akselilla kääntyvää moottoritelinettä, yksi kahdella akselilla kääntyvä moottoriteline toimilaitteineen, telineisiin kiinnitetyt moottorit potkureineen ja kiinteän siiven ohjaussiivekkeet sekä korkeusperäsin. Ohjauselektroniikka koostuu STM32F4-mikrokontrollerista, antureista, toimilaitteista ja moottoreiden nopeudensäätimistä.

IMU-tiedon tuottaminen fyysisessä prototyypissä on toteutettu mittaamalla kolmella akselilla kulmanopeuksia ja kiihtyvyyksiä. Mittauksiin käytettiin MPU-9250-laitetta, joka sisältää myös magnetometrin. Mittauksien tuloksista on mahdollista muodostaa asentotietoja. Tämän työn kannalta oleellisimpia ovat asento- ja suuntatieto suhteessa maahan (Φ,θ ja Ψ). Tähän projektiin liittyvien ilmannopeuden, ilman lämpötilan ja barometrisen korkeuden mittaamisen valittiin BMP280- ja MS4525DO-anturit. Sijaintitiedon tuottamiseen käytettiin GPS-yksikköä BN-880. Nämä laitteet valittiin hyvän saatavuuden, pienen fyysisen koon ja painon perusteella.

5 Kehitysympäristön valinta

Tässä luvussa keskitytään kehitysympäristöjen tarjoamiin mahdollisuuksiin. Ympäristöihin tutustuminen toteutettiin tutkimuksella ja kokeiluilla. Ominaisuudet painoarvotettiin ja pisteytettiin vaatimusten mukaisesti tukemaan valintaa tehtyjen havaintojen perusteella. Työn kannalta haluttujen kehitysympäristön vaatimusten määrittämiseksi kartoitettiin halutut ominaisuudet. Vaatimuksiksi määritettiin seuraavat lähtökohdat:

- valmiudet ohjauslogiikan rakentamiselle
- nopea MIL/SIL/PIL- simulointi
- hyvät ohjelmakoodin generointityökalut
- ympäristön monipuolisuus tulevia sovelluksia ajatellen
- monipuoliset tiedon analysointityökalut
- ohjeistuksen saatavuus ja selkeys.

Miehittämättömien pienoisilma-alusten ohjausjärjestelmien ratkaisuiksi soveltuvia alustoja on tarjolla useita. Tässä työssä keskityttiin tarjolla oleviin avoimen lähdekoodin projekteina kehittyviin järjestelmiin. Markkinoilla on pitkälle kehitettyjä valmiita ohjausjärjestelmäratkaisuja, näitä ovat muun muassa ArduPilot, Cleanflight, Betaflight ja iNav. Nämä alustat eivät sovellu lähtökohdaksi tämän työn tarkoituksiin, joten ne on jätetty valintaprosessin ulkopuolelle. (11.)

Kehitettävän lennonautomaatiojärjestelmän päätoiminnot ovat ilma-aluksen lennon aikainen stabilointi, automaattinen ohjaus ja navigointi. Automaattiohjauksen lisäksi, ilma-alusta tulee olla mahdollista ohjata käsin, käyttäen kauko-ohjausta, kaikkien lennon vaiheiden aikana. Käsin ohjatessa VTOL-lentotilassa järjestelmä stabiloi ilma-aluksen leijuntaan ja sallii ohjainkomentojen perusteella liikehdinnän pyöriväsiipisen multikopterin tavoin. Transition aikana VTOL-lentotilasta kiinteäsiipiseen lentotilaan, järjestelmä säilyttää lentokorkeuden moottoritelineiden kääntyessä noin 90 astetta. Samalla ilmannopeus kasvaa tavoitenopeuteen, ja ilma-alus saavuttaa kiinteäsiipisen lentotilan.

Päätoimintojen lisäksi tavoitteena on toteuttaa ilma-aluksen ohjausjärjestelmä, joka kykenee tunnistamaan ja seuraamaan kohteita sekä suorittamaan muita tehtäviä. Kehitysympäristön tulee tarjota joustavia mahdollisuuksia eri periaatteilla toimivien sensoreiden ja näiden tuottaman tiedon käsittelyyn, kuten LIDAR- ja konenäkösovellukset, joita voidaan jatkossa soveltaa niin navigointiin kuin tehtävän suorittamiseen. Kehitysympäristön monipuolisuudella, joustavuudella ja kehitystyötä nopeuttavilla työkaluilla on myöhemmin suuri merkitys isäntäprojektin eli ilma-aluksen tehtävänsuorituskyvyn kehittymisessä.

5.1 Vaihtoehtoiset ympäristöt

Tarjonnan joukosta valikoitui kolme vahvinta alustaa – ROS, Matlab ja Scilab. Valinta tehtiin tutustumalla kehitysympäristöihin, näiden tarjoamiin työnkulkuihin ja ominaisuuksiin. Näistä vaihtoehdoista MATLAB ja Scilab ovat hyvin toisensa kaltaisia ympäristöjä, mutta ROS poikkeaa edellisistä oleellisesti. Ympäristöihin tutustuminen toteutettiin tutkimalla ohjelmistojen ominaisuuksia syventymällä saatavilla oleviin ohjeistuksiin ja suorittamalla näiden pohjalta kokeiluita.

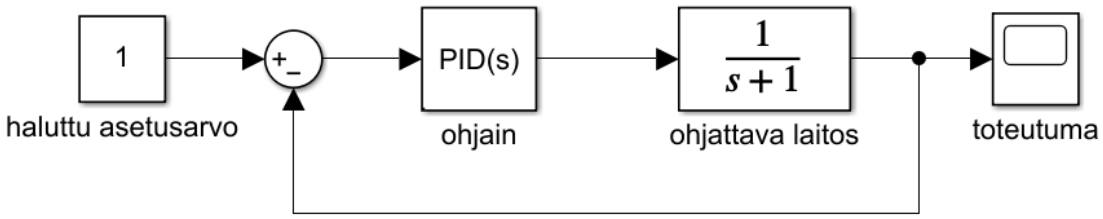
ROS on avoimen lähdekoodin projektina kehittyvä ympäristö, joka tarjoaa monipuoliset työkalut robotiikan sovellusten kehittämiseen ja simulointiin. ROS-ympäristö on kokoelma työkaluja ja sen käyttöliittymä on suurilta osin komentorivipohjainen (12). ROS-ympäristössä visualisoinnin ja simuloinnin toteuttamiseen on tarjolla muun muassa monipuolinen Gazebo-simulaatio. ROS otettiin mukaan vertailuun koska se tarjoaa monipuolisia käyttöskenaarioita ja sillä on laajat simulointiominaisuudet. ROS-ympäristö tarjoaa laajan valikoiman yleisesti robotiikan sovelluksiin päteviä ratkaisuvaihtoehtoja erilaisille toteutuksille. Ohjelmointikokeilu ympäristössä suoritettiin python ohjelmointikielellä. Tässä työssä haettuihin käyttötarkoituksiin ROS soveltui vain rajoitetusti, eikä kohdemikrokontrollerin soveltaminen tarvitussa roolissa ollut järkevää. (13.)

Scilab on avoimen lähdekoodin ohjelmisto, joka on kehitetty MATLAB:n tavoin numeerisen laskennan tarpeisiin. Xcos on työkalu, joka toimii Scilabin rinnalla samaan tapaan kuin Simulink toimii MATLAB:n kanssa (14). Tätä työtä varten

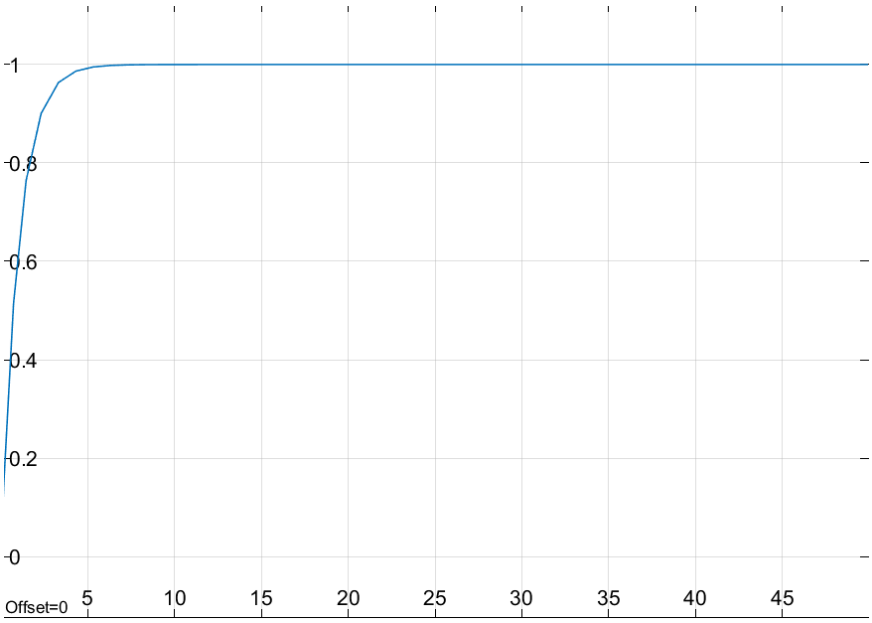
Xcos-ympäristössä suoritettiin logiikan ohjelmointikokeiluja. Lohkokaavioilla mallinnettiin PID-ohjain, jolla suoritettiin askelvastekokeita. Ohjelmakoodin tuottamiseen oli tarjolla useampi ratkaisu. (15.)

Mathworks MATLAB ja sen rinnalla toimiva Simulink on laajasti käytetty ohjelmisto. Se on käytössä monilla teollisuuden aloilla ja sitä käytetään muun muassa ohjausjärjestelmien kehitystyössä ja tutkimuksessa. Ensisijaisesti MATLAB on numeeriseen laskentaan, matemaattiseen mallintamiseen ja tiedon käsittelyyn kehitetty ympäristö. Monipuolisten ominaisuuksiensa ansiosta MATLAB-ympäristön käyttö on laajentunut lukemattomiin käyttötarkoituksiin. MATLAB:n rinnalla toimiva Simulink tarjoaa monipuoliset työkalut dynaamisten järjestelmien mallintamiseen ja niiden simulointiin graafisessa ympäristössä. Ympäristö tarjoaa myös hyviä työkaluja mallipohjaiseen järjestelmäsuunnitteluun. Yhtenä esimerkkinä voidaan mainita System Composer -lisäosa, jonka avulla voidaan suunnitella ja käsitellä järjestelmiä arkkitehtuuritasolla. MATLAB-ympäristöön on saatavilla laaja valikoima lisäosia erinäisiin käyttötarpeisiin, kuten tämän työn kannalta oleelliseen ohjelmakoodin generointiin. Ohjelmakoodin tuottamiseen tarvittavien työkalujen lisäksi tämän työn kannalta hyödyllisiksi osoittautuivat esimerkiksi MATLAB-ympäristön lisäosat: MATLAB Aerospace Toolbox ja Simulink Aerospace blockset. Nämä lisäosat sisältävät kattavat työkalut ilma-aluksen ja ilmakehän mallintamiseen sekä tarvittavien laskutoimituksien suorittamiseen. (16.)

Mallinnettaessa järjestelmää Simulink-ympäristössä järjestelmä kuvataan lohkokaavion muodossa (kuva 9). Käytettävissä on valmiita lohkoja jaettuina käyttötarkoitusten mukaisesti kirjastoihin. Simulinkin graafisessa ympäristössä lohkot voidaan lisätä malliin vetämällä ja pudottamalla kirjastosta malli-ikkunaan. Lohkojen määrittäminen tapahtuu kunkin lohkon parametri-ikkunan avulla. Simulaation ajaminen tapahtuu valitsemalla run-kuvake simulaation välilehdeltä. (17) Simulaation tulos on ajamisen jälkeen nähtävissä esimerkkijärjestelmän ”toteutuma”-scope-lohkon tuottamasta kuvaajasta (kuva 10).



Kuva 9. Esimerkkijärjestelmän säätö Simulinkissä.



Kuva 10. Esimerkkijärjestelmän simulaation tulos.

Ohjelmistojen arvioinnissa hyödynnettiin painoarvokerrointa ja ominaisuudet pisteytettiin tehtyjen havaintojen avulla. MATLAB/Simulink osoittautui erityisesti käytettävyyden ja saatavilla olevan ohjeistuksen osalta vahvimmaksi vaihtoehdoksi (taulukko 1).

Taulukko 1. Pisteytys painotettujen vaatimusten mukaisesti.

		ROS		Matlab		Scilab	
Kehitysympäristön vaatimukset	Painoarvo	Arvosana	Painotettu	Arvosana	Painotettu	Arvosana	Painotettu
valmiudet ohjauslogiikan rakentamiselle	5	4	20	5	25	4	20
nopea MIL/SIL/PIL- simulointi	4	3	12	4	16	5	20
nopea työnkulku	2	2	4	4	8	4	8
ohjelmakoodin generointityökalut	3	4	12	4	12	4	12
monipuolisuus tulevia sovelluksia varten	3	5	15	5	15	3	9
datan analysointityökalut	3	2	6	5	15	4	12
ohjeistuksen saatavuus ja selkeys	2	2	4	5	10	3	6
Tulos			**73**		**101**		**87**

ROS-ympäristö karsiutui pois vaihtoehdoista jo aikaisessa vaiheessa MATLAB- ja Scilab-ympäristöjen työnkulun ollessa selkeämpää ja helpommin lähestyttävää. Kehitysympäristöjen pisteyttäminen osoittautui käyttökelpoiseksi työkaluksi ja sen avulla valinta kallistui selvästi MATLAB-ympäristöön. Kehitysympäristöksi valittiin MATLAB/Simulink.

5.2 Lento- ja ympäristömallin valinta

Edellä kuvattujen ROS-, MATLAB- ja Scilab-ympäristöjen tarjoamilla työkaluilla voidaan simuloida tavoitteen mukaisesti lentoa, lennon ohjausta ja ympäristöä sitä varten luotujen mallien avulla. Lisäksi tarvitaan ympäristöä ja itse lentotapahtumaa simuloiva lentomalli. Tähän tarkoitukseen on tarjolla useita valmiita ja tarkoitukseen erinomaisesti sopivia vaihtoehtoja. Monet kaupalliset lentosimulaattorit tarjoavat hyvin tämän työn tavoitteita palvelevia ratkaisuja. Markkinoilla on myös avoimeen lähdekoodiin perustuvia kehitysympäristöjä. Molempien joukosta löytyy hyviä valmiita lentomalleja sekä todenmukaisia lentosuunnistusympäristöjä. Ulkoista lentomallia käytettäessä, on mahdollista keskittyä tehokkaammin ohjausjärjestelmän suunnitteluun lennon simuloinnin suunnittelun ja ympäristön toteuttamisen sijaan. Tähän työhön sopivina tarkasteltiin FlightGear- ja X-Plane-ohjelmistoja.

FlightGear on ilmailualan suunnittelutyössä laajasti käytetty avoimen lähdekoodin projektina kehittyvä simulaattori. FlightGear-ohjelmiston etuja ovat sen joustavuus ja muokattavuus. X-Plane-simulaattori on Laminar Research -yhtiön

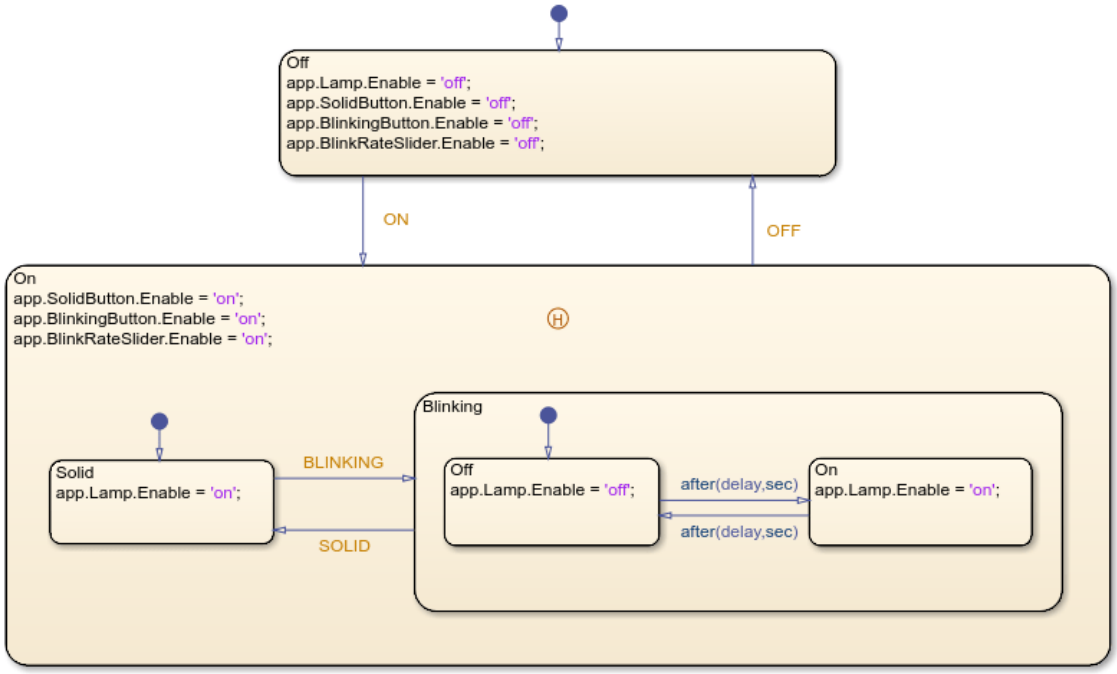
tuottama kaupallinen ohjelmisto. X-Plane-simulaattorissa on mallinnettuna koko planeetta ja sitä ympäröivä ilmakehä. X-Plane-ohjelmistossa on sisäänrakennettuna myös hyvät työkalut ilma-alusten mallintamista varten. Simulaattorin lentomalli perustuu BET-menetelmään, jossa siipiprofiilin ilmamassassa tapahtuvasta liikkeestä johtuvat voimat lasketaan (18). Lentomallin tarkkuus on erittäin hyvä ja tätä tukee myös se että Yhdysvaltojen ilmailuviranomainen FAA:n on hyväksynyt X-Plane-simulaattorin käytön lentokoulutuksessa (19).

Kehitysympäristön lento- ja ympäristömallina päätettiin käyttää X-Plane-simulaattoria vakuuttavan lentomallin ja erinomaisen liitettävyytensä ansiosta.

6 Käyttöönotto

Kehitystyö Simulink-ympäristössä tapahtuu pääosin työskentelemällä PC-työasemalla, jolla kaikkia kehitysympäristön ohjelmistoja ajetaan. Tarkoitukseen valittiin suorituskyvyltään soveltuva PC-tietokone. Valittuun tietokoneeseen asennettiin tarvittavat ohjelmistot ja näiden asetukset konfiguroitiin tarpeiden mukaisesti.

Järjestelmäsuunnittelu Simulink-ohjelmassa tapahtuu lohkokaavioilla graafisesti työskennellen mallintamalla järjestelmäkaavion piirtämisen omaisesti. Logiikan rakentamiseen käytetään Simulink Stateflow -työkalua (kuva 11).



Kuva 11. Stateflow-logiikan ohjelmointiesimerkki (20).

Simulinkin ja lentomallina käytetyn X-Plane-simulaattorin välinen kommunikaatio tuli toteuttaa käyttöönoton alkuvaiheessa, jotta ohjausjärjestelmän mallintaminen voitiin aloittaa.

Generoitavan lähdekoodin kohdelaitteistot ovat mikrokontrollereita, kuten STM32F407. Kohdemikrokontrollereiden ja kehitysympäristön välisen

kommunikaation ohjelmointiin tarvittavat lisäosat ja Simulink-lohkot asennettiin tarvittavien toimintojen saavuttamiseksi.

6.1 Kohdelaitteisto

Ohjelmointiympäristön isäntälaitteistoksi valittiin PC-tietokone Windows-käyttöjärjestelmällä. Kohdemikrokontrollereina toimivat STM32F4DISCOVERY-kehitysalusta, kevyempi kehitysalusta STM32F103C8T6 ”blue pill” sekä sensoridatan käsittelyyn kaavailtu Arduino Nano -kehitysalusta.

6.2 Ohjelmistojen asentaminen ja konfigurointi

Ympäristön käyttöönotto edellytti ohjelmistojen asentamisen PC-tietokoneelle. Ohjelmistojen sekä laitteiston yhteistoimintaa varten tarvittiin myös laiteajurit valituille kehitysalustoille.

Seuraavat ohjelmat asennettiin:

- MATLAB
- Simulink
- X-Plane.

Lisäksi asennettin

- Simulink realtime kernel
- Aerospace toolbox
- Aerospace Blockset
- Control System Blockset
- MATLAB Coder
- Simulink Coder
- Embedded Coder
- STM32Cube MX
- STM32-MAT/TARGET
- MDK-ARM
- Simulink Support Package for Arduino Hardware / Sensors.

Ohjelmistot olivat pääosin käyttövalmiita heti asennuksen jälkeen käyttäen ohjelmistojen asennustyökaluja. Kehitystyön luonteen vuoksi ohjausjärjestelmän

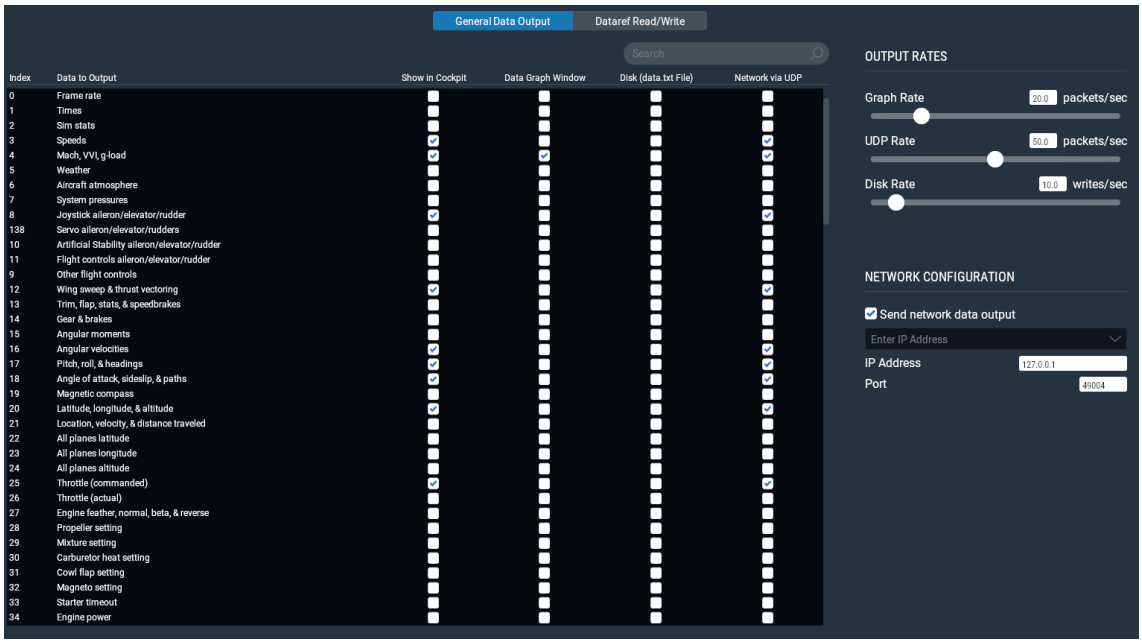
simulointi tuli suorittaa reaaliajassa. Tätä tarkoitusta varten MATLAB realtime kernel asennettiin MATLAB-komennolla ”sldrtkernel -install”.

X-Plane-simulaattorin ympäristö- ja ilma-alusmuuttujat todettiin olevan tehokkaimmin luettavissa, manipuloitavissa ja siirrettävissä simulaattorin ja Simulinkin välillä verkkoliikenteenä UDP-viesteinä ennalta valittuina ryhminä. UDP on minimaalinen protokolla, jossa ei ole yhteydenmuodostusta eikä virheenkorjausta, joten kommunikaatio ei kuormita käytettävän laitteiston resursseja merkittävästi (21). Tämä palvelee hyvin myös PIL-simulaatiotarkoituksia UDP-liikenteen ollessa nopeaa ja helposti toteutettavissa myös mikrokontrollerilla ajettavalla ohjelmakoodilla.

X-plane UDP-datagrammin viisi ensimmäistä tavua sisältää ASCII merkkeinä viestin datan tyypin. Tyyppejä ovat DATA, DREF ja CMND. DATA-tyypin viesti sisältää otsikon jälkeen simulaattorista ennalta valitut dataryhmät (22). Seuraavat neljä tavua määrittävät dataparametriryhmän ja tätä seuraavat 32 tavua sisältävät kahdeksan parametriä 4-bittisinä liukulukuina (kuva 8). Simulaattorin jatkuvasti lähetettävät dataryhmät ovat valittavissa asetuksista (kuva 9).

Otsikko	tunniste 1	32 bittiä dataa, 8 parametriä 4-bittisinä liukulukuina								tunniste 2
68-65-84-65-60	131-0-0-0	166-138-16-71	193-114-2-195	58-143-139-70	224-238-135-59	61-207-225-58	0-192-121-196	0-192-121-196	0-192-121-196	133-0-0-0

Kuva 12. X-Plane-simulaattorin UDP-viestin rakenne.

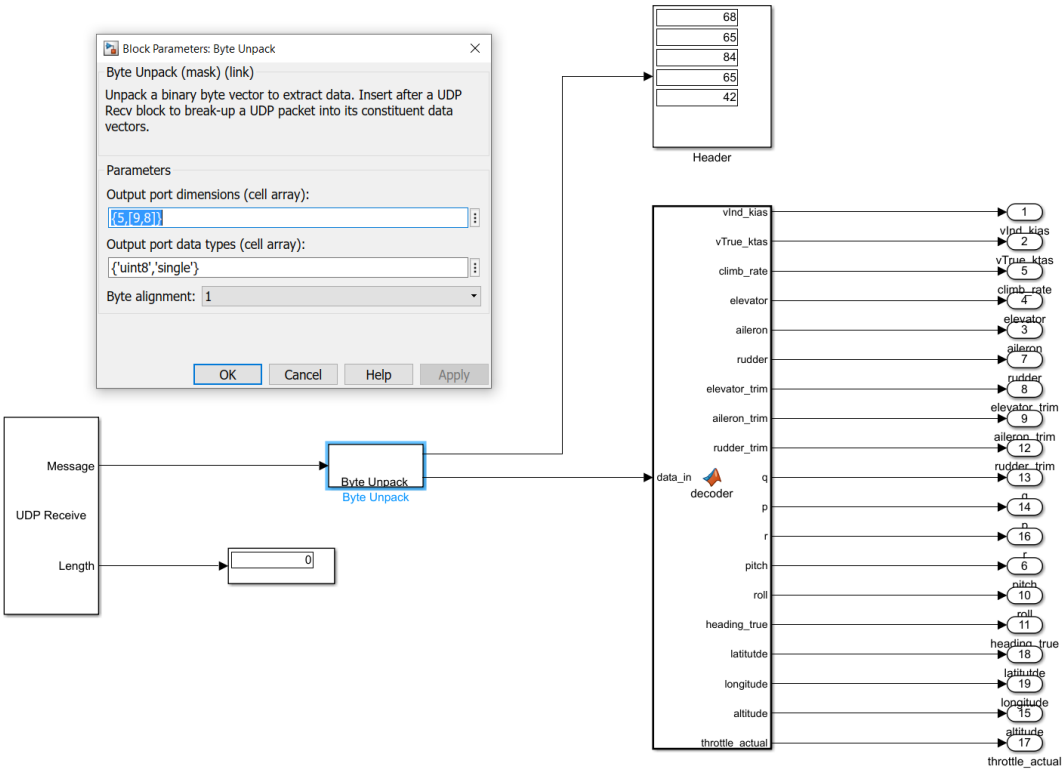


Kuva 13. X-Plane, valitut dataryhmät.

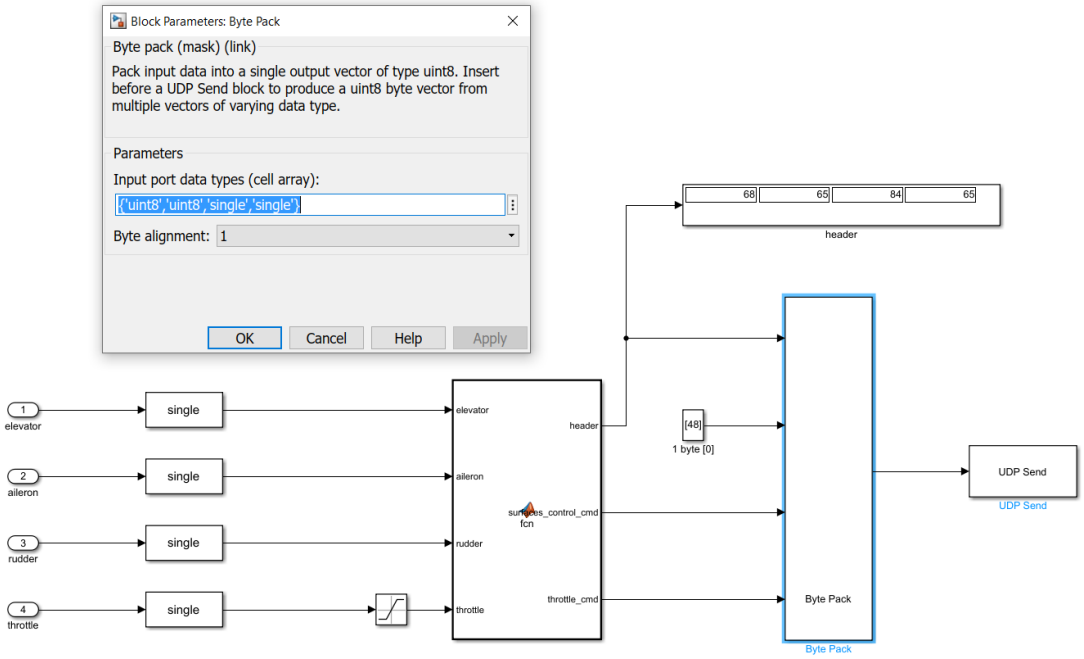
DREF-tyypin viesti sisältää yhden tietyn muuttujan simulaatiossa ja on pituudeltaan 509 tavua. DREF-muuttuja voidaan lukea tai kirjoittaa UDP-viestiä käyttäen. Lähetettävän viestin pituus säädetään 509 tavuun täyttämällä dataryhmien jälkeen puuttuva pituus nollilla. DREF-sanoman toimintaa testattiin tätä varten kirjoitetulla python-ohjelmalla (liite 1). (22.)

Simulinkin UDP Send- ja UDP Receive-lohkot soveltuivat hyvin viestien vastaanottamiseen ja lähettämiseen. Otsakkeen erottamiseen datasta käytettiin Byte Unpack- lohkoa. MATLAB Function -lohkoa käyttäen kirjoitettiin funktio, jolla data sijoitettiin omiin yksittäisiin ulostuloihin sekä sisäänmenoihin (esimerkkikoodi 1). (23.)

X-Plane UDP- kommunikointia varten luotiin alijärjestelmät, joiden lähdöt ja tulot linkittyvät luettaviin ja kirjoitettaviin muuttujiin simulaattorissa. (kuva 13, kuva 14.)



Kuva 14. Simulink-alijärjestelmä UDP-datan vastaanottoon.



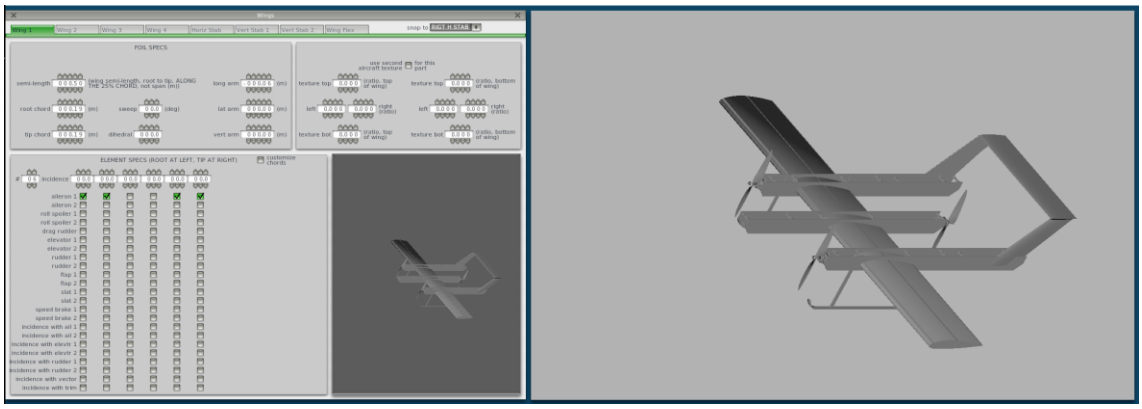
Kuva 15. Simulink-alijärjestelmä UDP-datan lähettämiseen.

```
function [vInd_kias,vTrue_ktas,...
    climb_rate,...
    elevator,aileron,rudder,...
    elevator_trim,aileron_trim,rudder_trim,...
    q,p,r,...
    pitch, roll, heading_true,...
    latitutde, longitude, altitude,...
    throttle_actual] = decoder(data_in)
 vInd_kias = data_in(2,1);
 vTrue_ktas = data_in(4,1);
 climb_rate = data_in(4,2);
 elevator = data_in(2,3);
 aileron = data_in(3,3);
 rudder= data_in(4,3);
 elevator_trim= data_in(2,4);
 aileron_trim= data_in(3,4);
 rudder_trim= data_in(4,4);
 q= data_in(2,5);
 p= data_in(3,5);
 r= data_in(4,5);
 pitch= data_in(2,6);
 roll= data_in(3,6);
 heading_true= data_in(4,6);
 latitutde= data_in(2,7);
 longitude= data_in(3,7);
 altitude= data_in(4,7);
 throttle_actual = data_in(2,8);
```

Esimerkkikoodi 1. MATLAB-funktio, jolla data sijoitetaan ulostuloihin UDP-dataa vastaanottavassa alijärjestelmässä.

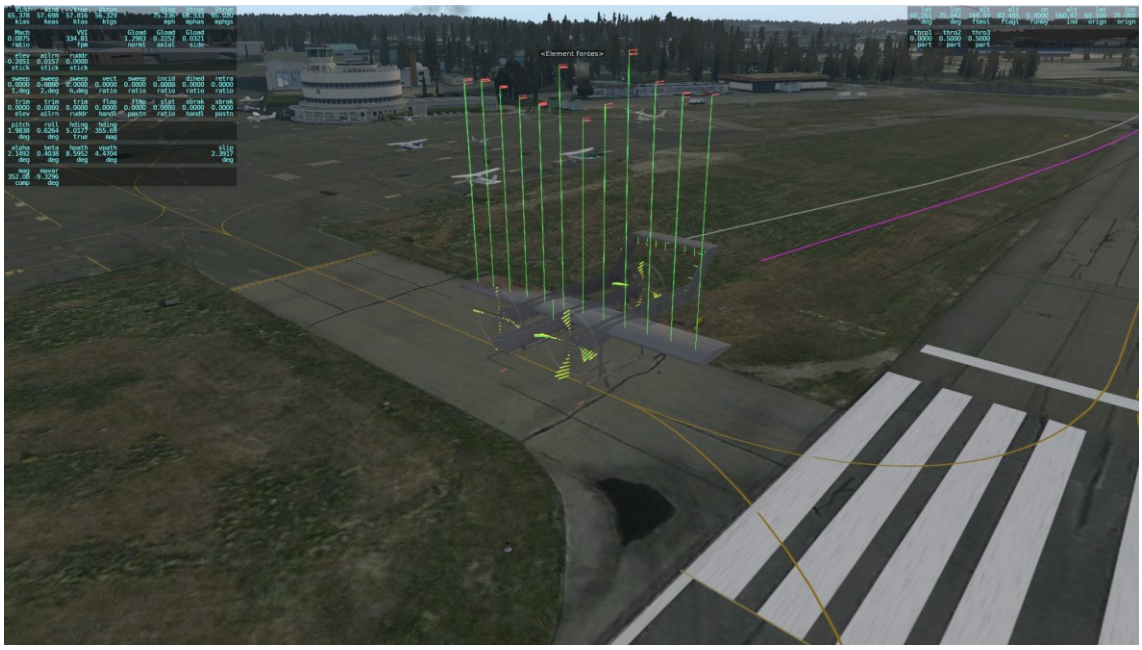
6.3 Kehitystyön aloittaminen

Ilma-aluksen prototyyppi mallinnettiin X-Plane-simulaattoriin käytettäväksi ohjausjärjestelmän kehityksessä. Isäntäprojektissa 3d-mallinnettu prototyyppi tuotiin X-Plane Plane Maker -työkaluun, jossa prototyypin aerodynaamiset ja fyysiset ominaisuudet sekä järjestelmät määriteltiin (kuva 15).



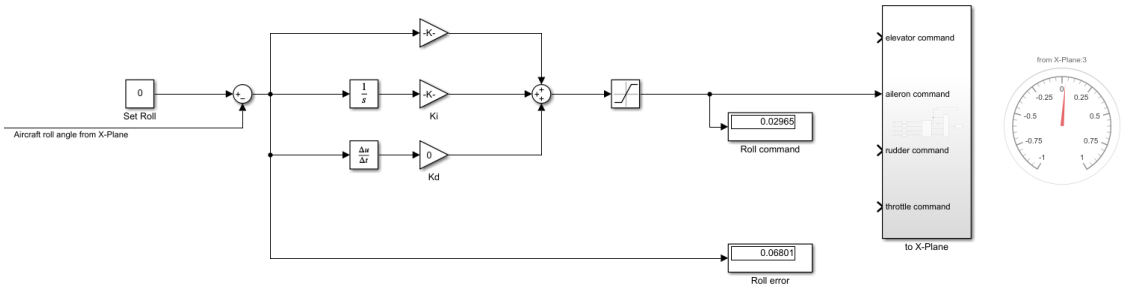
Kuva 16. Prototyypin mallintaminen Plane Maker -työkalussa.

Mallinnetun prototyypin ominaisuudet säädettiin halutulle tasolle suorittamalla tällä kokeita simulaattorissa (kuva 16). Kun prototyyppi oli mallinnettu X-Plane-simulaattorissa ja testattu toimivaksi, aloitettiin lennonohjausjärjestelmän hahmottelu Simulink-ympäristössä.

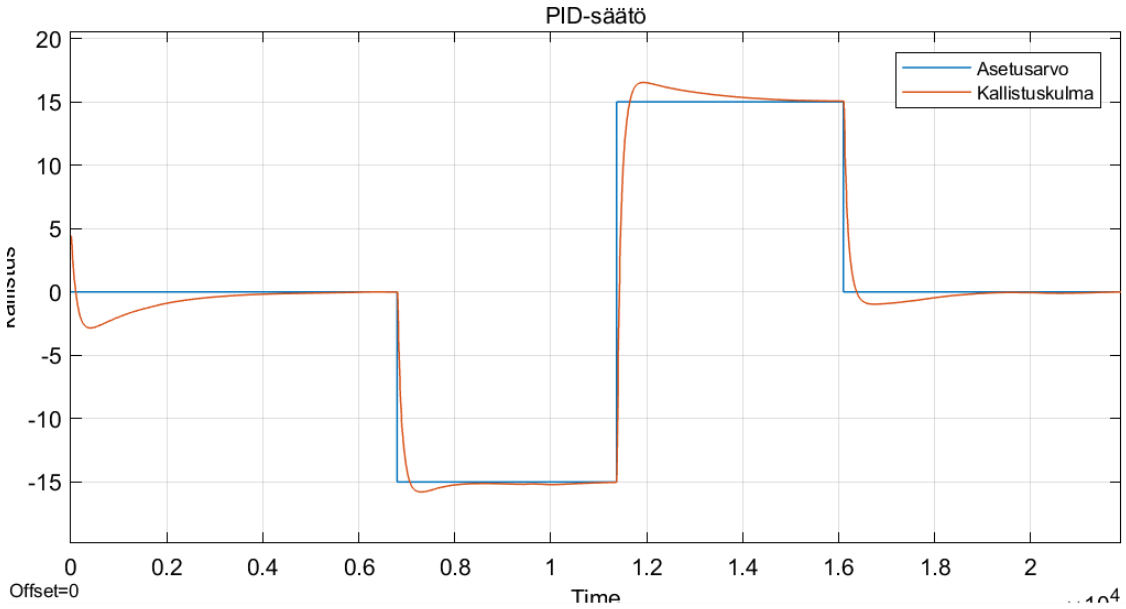


Kuva 17. Mallinnettu prototyyppi X-Plane-simulaattorissa.

Kehitympäristön kokeilemiseksi luotiin säätöjärjestelmä ohjaamaan kallistuskulmaa mallinnetun ilma-aluksen pitkittäisakselilla (23). Tämä toteutettiin luomalla PID-ohjain, joka pitää ilma-aluksen kallistuskulman asetusarvossa käyttäen mitattavana arvona simulaattorista saatua ilma-aluksen kallistuskulmaa ja toimilaitteena simulaattoriin lähetettyä ohjainpintakomentoa. (kuva 17).



Kuva 18. Simulink PID-säätö kallistuskulmalle.



Kuva 19. Askelvaste kallistuskulman PID-säädössä.

Askelvastekokeiden avulla kokeilemalla haettiin sopivat PID-säätimen sopivat vahvistusarvot (kuva 18). Vastaava PID-säädin tehtiin myös ohjaamaan pystynopeutta käyttäen toimilaitteena korkeusperäsimen ohjauskomentoa. Nämä kokeilut todistivat kehitysympäristön toimivuuden ja työnkulun vastaavan tavoiteltua tasoa.

7 Yhteenveto

Tuloksena saatiin nopeatahtiseen työnkulkuun sopiva alusta haastavaan kehitystyöhön. Ympäristön valinnan ja käyttöönoton aikana tehdyt kokeilut osoittivat suunnittelutyökalujen tehokkuuden käytännössä.

Tulevaisuudessa tämän työn pohjalta jatkettavan kehitystyön tavoitteena on vaiheittain rakentaa prototyyppialustalla lennonautomaatiojärjestelmä, joka voi hyödyntää laajaa anturivalikoimaa. Kehitystyön tavoitteena on joustava lentotyöhön soveltuva robotiikkasovellus. Kehitystyö simulaatiopainotteisesti Simulink-ympäristössä sekä pääosin lohkokaavioin ja Stateflow -tilakoneilla mallinnettu logiikka vapauttavat pienen suunnittelutiimin resursseja käytettäväksi koodaamisesta itse suunnittelutyöhön. Valitun kehitysympäristön nopea luonne ja joustavuus antavat hyvän pohjan tuottoisalle kehitystyölle.

Kehitysympäristössä alkavan kehitystyön alkuvaiheessa tullaan rakentamaan ilma-aluksen VTOL-lentotilan leijuntaan ja liikehditään tarvittavat säätöjärjestelmät sekä kiinteäsiipisen lentotilan säätöjärjestelmät. Näistä työ tulee etenemään edellä mainittujen lentotilojen välisen transition säätöjärjestelmään.

Simulink ja saatavilla olevat lisäosat tarjoavat paljon valmiita työkaluja tehtävänaikaisten toimintojen kuten konenäkösovellusten ja pistepilvitiedon keräämisen kehittämiselle (24). Prototyyppialustalla tullaan kokeilemaan laajasti erilaisia anturi- ja tiedonkäsittelyratkaisuja. Ympäristössä tullaan kehittämään myös koko ilma-alusjärjestelmän käyttöliittymä lennon- ja tehtävänhallintaa varten.

Lähteet

1 Euroopan unionin virallinen lehti. 2019. Komission täytäntöönpanoasetus (EU) 2019/947 L 152/49. Bryssel: Euroopan Unioni.

2 Hitchens FE. 2015. The encyclopedia of aerodynamics. 2nd ed. Luton, Bedfordshire: Andrews UK Limited.

3 DroneFinland teki kesällä 60 maatalouden kuvauslentoa. 2016. Maanmittauslaitos. Verkkoaineisto. <https://www.maanmittauslaitos.fi/ajankohtaista/dronefinland-teki-kesalla-60-maatalouden-kuvauslentoa>. 22.9.2016. Luettu 09.09.2021.

4 Fazer lähti ilmojen teille Helsingissä. 2021. Verkkoaineisto. Tekniikka & Talous. <https://www.tekniikkatalous.fi/uutiset/fazer-lahti-ilmojen-teille-helsingissa-drone-kuljettaa-pullat-ja-karkit-kohteeseen-nain-paljon-se-jaksaa-kantaa/d001b262-1ff7-40fd-9e46-5e0724a1224f>. 19.5.2021. Luettu 10.09.2021.

5 Irschik H, Krommer M, Belyaev A. 2012. Advanced dynamics and model-based control of structures and machines. Vienna: Springer.

6 Miksi mallipohjainen suunnittelu? Simulointi ja testaus osaksi vaativaa ohjelmistokehitystä. 2019. Verkkoaineisto. Devetco. <https://devecto.com/miksi-mallipohjainen-suunnittelu/>. Luettu 02.07.2021.

7 Haapanen, Erkki. 1984. Aerodynamiikka. Helsinki: Suomen ilmailuliitto.

8 Cook MV. 2007. Flight dynamics principles. A linear systems approach to aircraft stability and control. 2nd ed. Oxford: Butterworth-Heinemann/Elsevier.

9 Ng TS. 2018. Flight systems and control. A practical approach. Singapore: Springer.

10 Pixhawk Px4 Differential Airspeed Sensor Kit Pitot Tube Airspeedometer. 2021. Aeroflyhobbies.com. Verkkoaineisto. <http://www.aeroflyhobbies.com/flight-controllers/263-pixhawk-px4-differential-airspeed-sensor-kit-pitot-tube-airspeedometer.html>. 12.04.2021. Luettu 12.07.2021.

11 Ebeid E, Skriver M, Terkildsen KH, Jensen K, Schultz UP. 2018. A survey of Open-Source UAV flight controllers and flight simulators. Microprocessors and microsystems 61:11-20.

12 ROS - Robot Operating System. 2021. Verkkoaineisto. ROS.org. <https://ros.org/>. Luettu 12.09.2021.

13 Fairchild C, Harman T. ROS Robotics By Example. 2016. Birmingham: Packt Publishing.

14 About Scilab. 2021. Verkkoaineisto. scilab.org. <https://www.scilab.org/about>. Luettu 12.09.2021.

15 Wouwer AV, Saucez P, Vilas C. 2014. Simulation of ODE/PDE models with MATLAB®, OCTAVE and SCILAB : scientific and engineering applications. Cham: Springer.

16 Simulink. 2021. Verkkoaineisto. Mathworks. <https://se.mathworks.com/products/simulink.html>. Luettu 20.10.2021.

17 Eshkabilov S. 2019. Beginning MATLAB and Simulink: From Novice to Professional. Berkeley: Apress L. P.

18 Design Human-Machine Interface Logic by Using Stateflow Charts. 2021. Verkkoaineisto. Mathworks. <https://se.mathworks.com/help/stateflow/ug/sf4ml-lamp-example.html>. 22.09.2021. Luettu 07.11.2021.

19 How X-Plane Works. 2021. Verkkoaineisto. Laminar Research. <https://www.x-plane.com/desktop/how-x-plane-works/>. Luettu 04.07.2021.

20 FAA-Certified X-Plane. 2021. Verkkoaineisto. Laminar Research. <https://www.x-plane.com/pro/certified/>. Luettu 04.07.2021.

21 User Datagram Protocol (UDP). 2021. Verkkoaineisto. TechTarget. <https://www.techtarget.com/searchnetworking/definition/UDP-User-Datagram-Protocol>. Luettu 02.11.2021.

22 About X-Plane Data Input/Output structure. 2012. Verkkoaineisto. Vlad Sychev. <https://simvim.com/b58/xdata.html> Luettu 07.02.2021.

23 Bittar A, Figuereido HV, Avelar Guimaraes P, Correa Mendes A. 2014. Guidance Software-In-the-Loop simulation using X-Plane and Simulink for UAVs. ICUAS.

24 Kanellakis C, Nikolakopoulos G. 2017. Survey on Computer Vision for UAVs: Current Developments and Trends. Journal of intelligent & robotic systems 87:141-168.

Python-ohjelma, X-Plane UDP- komennon lähettäminen

```
#!/usr/bin/env python3
#JPa
# Send dataref to X-plane 11 via UDP scipt
from struct import *
import socket
from xplane_address import *

type = "DREF0"
value1 = float(input("Syötä arvo: "))
dref1 = "sim/cockpit/autopilot/heading_mag"

# encode to bytes
type_b = type.encode('utf-8')
value1_b = pack('f', value1)
dref_b = dref1.encode('utf-8')
msg_bytes = type_b+value1_b+dref_b
# add padding
msg_padded = type_b+value1_b+pack(str(542-len(msg_bytes))+'s', dref_b)

#debug print message size in bytes
print (len(msg_padded))

# Send message
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM) #INTERNET,
UDP_IP
sock.sendto(msg_padded, (UDP_IP, UDP_PORT))
# Print info
print("UDP target IP:", UDP_IP)
print("UDP target port:", UDP_PORT)
print(dref1)
print(value1)
#print("message:", msg_padded)
```

Kehitysympäristön testaukseen luotu järjestelmä