Kauno technologijos universitetas

Mechanikos inžinerijos ir dizaino fakultetas

Ryšio žemė-BO-BO retransliatoriaus ir jo algoritmo projektavimas ir sukūrimas

Baigiamasis magistro projektas

Edvinas Alenskas
Projekto autorius

Doc. Dr. Saulius Japertas
Vadovas

Kaunas, 2018

Kauno technologijos universitetas
Mechanikos inžinerijos ir dizaino fakultetas

Ryšio žemė-BO-BO retransliatoriaus ir jo algoritmo projektavimas ir sukūrimas

Baigiamasis magistro projektas
Aeronautikos inžinerija (621H40001)

Edvinas Alenskas
Projekto autorius

Doc. Dr. Saulius Japertas
Vadovas

Doc. Dr. Vitas Grimaila
Recenzentas

Kaunas, 2018

Kauno technologijos universitetas
Mechanikos inžinerijos ir dizaino fakultetas
Edvinas Alenskas

Ryšio žemė-BO-BO retransliatoriaus ir jo algoritmo projektavimas ir sukūrimas

Akademinio sąžiningumo deklaracija

Patvirtinu, kad mano, Edvino Alensko, baigiamasis projektas tema „**Ryšio žemė-BO-BO retransliatoriaus ir jo algoritmo projektavimas ir sukūrimas**“ yra parašytas visiškai savarankiškai ir visi pateikti duomenys ar tyrimų rezultatai yra teisingi ir gauti sąžiningai. Šiame darbe nei viena dalis nėra plagijuota nuo jokių spausdintinių ar internetinių šaltinių, visos kitų šaltinių tiesioginės ir netiesioginės citatos nurodytos literatūros nuorodose. Įstatymų nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs.

Aš suprantu, kad išaiškėjus nesąžiningumo faktui, man bus taikomos nuobaudos, remiantis Kauno technologijos universitete galiojančia tvarka.

(vardą ir pavardę įrašyti ranka)	(parašas)

KAUNO TECHNOLOGIJOS UNIVERSITETAS

MECHANIKOS INŽINERIJOS IR DIZAINO FAKULTETAS

Studijų programa .....

MAGISTRANTŪROS STUDIJŲ BAIGIAMOJO PROJEKTO UŽDUOTIS

Studentui(-ei)

1. Baigiamojo projekto tema

   Ryšio žemė-BO-BO retransliatoriaus ir jo algoritmo projektavimas ir sukūrimas

   Designing and creating connection Earth-UAV-UAV repeater and its algorithm

2. Projekto tikslas ir uždaviniai

Tikslas – realizuoti BO ryšio retransliatorių sukuriant autonominio skrydžio algoritmą tarpiniam retransliuojančiam BO bei suprojektuojant bei realizuojant retransliatoriaus įrangą.

Uždaviniai: atlikti literatūros analizę; sukurti autonominio skrydžio algoritmą; suprojektuoti retransliavimo sistemą; realizuoti algoritmą ir retransliatoriaus sistemą.

3. Pradiniai Projekto duomenys: -
4. Pagrindiniai reikalavimai ir sąlygos: -
5. Projekto aprašomosios dalies struktūra

   Bepiločių orlaivių tipai; Retransliacija; Retransliatoriaus parengimas; Algoritmai.

6. Grafinės Projekto dalies sudėtis: -
7. Projekto konsultantai: -

Magistrantas: .....
(vardas, pavardė, parašas, data)

Projekto vadovas.....
(vardas, pavardė, parašas, data)

Krypties studijų programos vadovas.....
(vardas, pavardė, parašas, data)

TURINYS

PAVEIKSLŲ SĄRAŠAS

LENTELIŲ SĄRAŠAS

Alenskas, Edvinas. Ryšio žemė-BO-BO retransliatoriaus ir jo algoritmo projektavimas ir sukūrimas. Magistro baigiamasis projektas / vadovas doc. dr. Saulius Japertas; Kauno technologijos universitetas, Mechanikos inžinerijos ir dizaino fakultetas.

Studijų kryptis ir sritis (studijų krypčių grupė): Aeronautikos inžinerija (E14), Inžinerijos mokslai.

Reikšminiai žodžiai: kvadrakopteris; valdymas; retransliatorius; autonominis; algoritmas.

Kaunas, 2018. 44 p.

SANTRAUKA

Šis darbas skirtas sukurti ryšio bepiločių orlaivių ryšio retransliavimo sistemą. Tokia sistema skirta pagerinti ryšio pasiekiamumą bei padidinti valdymo nuotolį. Šiame darbe aprašomas bepiločio orlaivio, kaip valdymo signalo retransliatoriaus projektavimas ir sukūrimas. Orlaivis suprojektuojamas pasinaudojant skaičiuokle „eCalc“. Jis sukonstruojamas pagal paruoštą projektą. Tokia sistema skirta civiliniams BO perduoti signalą už kliūčių bei pailginti perduodamo signalo atstumą. Darbe taip pat projektuojama retransliavimo įranga. Orlaivis ir įranga ištestuojama realiomis sąlygomis. Išmatuotas retransliatoriaus skrydžio laikas ir atliktas ryšio kokybės nuo atstumo priklausomybės bandymas. Retransliuojančiam BO sukuriamas autonominio skrydžio algoritmas. Ryšio perdavimo įrangai sukuriamas algoritmas skirtas perduoti valdymo signalą neprarandant orlaivio kontrolės. Sukurti algoritmai pateikiami pseudo kodu. Jie taip pat realizuojami ir ištestuojami panaudojant programinės įrangos simuliatorių.

Alenskas, Edvinas. Designing and creating connection Earth-UAV-UAV repeater and its algorithm. Assoc. Prof. Dr, Saulius Japertas; The Faculty of Mechanical Engineering and Design, Kaunas University of Technology.

Study field and area (study field group): Aeronautical Engineering (E14), Engineering Science.

Keywords: quadcopter; control; relay; autonomous; algorithm.

Kaunas, 2018. 44 pages.

SUMMARY

This work is intended to create a communication retransmission system for unmanned aerial vehicle communication. Such a system is designed to improve the accessibility of the radio cotrol and increase its distance. This thesis paper talks about designing and creating a UAV system as a signal relay station for other UAV's. Aerial vehicle is designed using "eCalc" software. It is then constructed by the created design. This system is created for retransmitting control radio signal for civilian UAV's. It will help transmit radio signal over obstacles and help to lengthen the range of operation. This work also talks about designing gear for signal relaying. UAV and radio gear is tested in field. Tests include flight time stress test and radio signal quality over distance test. Autonomous flight algorithm is created for relaying UAV. For radio gear algorithm is created to relay radio control signal in a way that does not allow to lose control of UAV. Algorithms are written in pseudo code. They also are implemented and tested using simulation software.

TERMINŲ IR SANTRUMPŲ ŽODYNAS

BO – bepilotis orlaivis;

LOS – (ang. – Line of sight) tiesioginis matomumas;

NLOS – (ang. – Non line of sight) netiesioginis matomumas;

FHSS – (ang. – Frequency hopping spread spectrum) dažnio šuoliavimo dažnių išplėtimo moduliacija;

FSK – (ang. – Frequency-shift keying) dažnio poslinkio moduliacija;

TDM – (ang. – Time division multiplexing) laiko dalinimo;

UHF – (ang. – Ultra high frequency) ultra aukšto dažnio juosta 300 MHz – 3 GHz;

VHF – (ang. – Very high frequency) labai aukšto dažnio juosta 30 MHz – 300 MHz;

PWM – (ang. – Pulse width modulation) pulso pločio moduliatorius;

PID – (ang. – proportional-integral-derivative controller) proporcijos, integravimo, diferencijavimo valdiklis;

IMU – (ang. – inertial measurement unit) inercinis matuoklis;

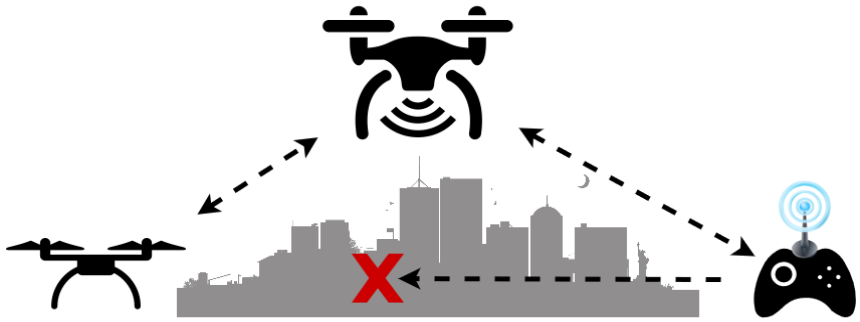
TTL – (ang. – transistor - transistor logic) tranzistoriaus pagrindo loginis keitiklis.

ĮVADAS

Bepiločiai orlaiviai (BO) ir jų panaudojimas per pastaruosius metus labai išplito tiek karinėms reikmėms, tiek civilinėms. Nors šiuo metu daugiausiai bepiločių orlaivių naudojama kariniais tikslais, civilinių bepiločių orlaivių panaudojimas labai sparčiai auga. Tą padeda užtikrinti nebrangi lengvai įsigyjama įranga ir faktas, kad civilinių bepiločių orlaivių pilotavimui nereikalingi specialūs leidimai. Civilinių BO, kuriuos galima įsigyti jau bene visose didesnėse prekybos vietose, paskirtis dažniausiai būna pramoginė. Tokie orlaiviai būna skirti filmuoti ar fotografuoti bei dažnai neturi jokios specialios paskirties. Taip pat bepiločiai orlaiviai pradėti naudoti sportui. Pastaruoju metu didelės kompanijos, tokios kaip „Facebook“, „Amazon“, „Google“ pradėjo domėtis bepiločių orlaivių panaudojimu tiekiant belaidžio interneto bei prekių pristatymo autonominiu BO paslaugas [1]. Tokiems BO valdyti reikalingos netiesioginio matomumo (NLOS) komunikacijos arba ekspromtiniai tinklai, arba pasinaudojus mobiliojo ryšio tinklu. Paprasti civiliniai bepiločiai orlaiviai palaiko tik tiesioginio matomumo (LOS) komunikaciją.

Kariniai bepiločiai orlaiviai dažnai naudoja NLOS komunikaciją, pasitelkdami palydovinį ryšį arba tarpines retransliuojančias stotis, kuriomis gali būti ir kiti BO. LOS komunikacijos didžiausi trūkumai yra iš dalies nedidelis atstumas bei ryšio kokybės ar visiškas ryšio praradimas dėl kliūčių. NLOS komunikacija sprendžia šias bėdas panaudodama tarpinius retransliatorius. Tipinė retransliatoriaus schema (1 pav.) pavaizduoja kaip vyksta komunikacijos, kai komunikuojama su bepiločiu orlaiviu netiesioginio matomumo sąlygomis, naudojant bepilotį orlaivį retransliatorių.

Naudojant aukštesnius dažnius, pavyzdžiui labai paplitusį 2,4 GHz, tokios kliūtys kaip pastatai ar medžiai, sukelia radijo ryšio šešėliavimo efektą. Tiesioginis ryšys su BO negalimas, todėl norint turėti ryšį su BO, reikalinga turėti retransliavimui skirtą įrangą, per kurią siunčiamas signalas išvengtų kliūčių. Tokią įrangą tikslinga iškelti aukštai virš kliūčių, taip kaip pavaizduota 1 pav. Retransliatorius taip pat turi gebėti pasirinkti ir išlaikyti tinkamiausią poziciją užtikrinant ryšio kokybę, išvengiant tiesioginio ryšio praradimą tiek su BO, tiek su operatoriumi.



1 pav. Ryšio retransliavimas už kliūčių

Dažniausiai naudojami bepiločiai orlaiviai komunikacijai perduoda signalą 2,4 GHz dažnių juostoje. Aukšto dažnio komunikacijos turi trūkumą, nes veikia nedideliais atstumais. Panaudojus papildomą BO retransliuoti ryšį tarp operatoriaus ir valdomo BO, galimas atstumo pailginimas ir ryšio kokybės bei patikimumo užtikrinimas skrendant už kliūčių. Pats BO retransliatorius turėtų sugebėti autonomiškai skristi, pasirinkdamas ir palaikydamas optimalų atstumą tarp operatoriaus ir valdomo BO. Taip būtų užtikrinama geriausia ryšio kokybė tarp operatoriaus ir valdomo BO. Visa tai atliktų algoritmas, kuris sugebėtų pagal parametrus išskaičiuoti tinkamiausią poziciją, kurią turi palaikyti retransliuojantis BO. Toks bepilotis retransliatorius išplėstų paprastų civilinių orlaivių panaudojimo galimybes.

Retransliavimui, pasinaudojant BO, yra skirta keletas darbų. Dauguma jų yra kariniai projektai, skirti retransliuoti komunikacijoms, nesinaudojant palydovinio ryšio paslaugomis ar pagerinti ryšį su kariais už kliūčių ar didelių atstumų [2]. Vienas iš tokių darbų tiria beveik tokią pačią sistemą [3], kaip bus tiriama šiame darbe. Darbe [3] realizuota sistema taip pat valdo bepilotį orlaivį retransliuojant komunikacijas. Tačiau vienas iš skirtumų yra tai, jog kontroliuojamas orlaivis nėra tiesiogiai valdomas vartotojo. Bepilotis orlaivis valdomas su užduodamomis komandomis iš valdymo stoties. Skirtingai nei anksčiau paminėtame darbe, šiame darbe valdymas bandomas atlikti tiesiogiai vartotojo su grįžtamu ryšiu (telemetrija). Dar vienas skirtumas yra tas jog minėtame darbe, kitaip nei šiame darbe, autonominis skrydis neatsižvelgia į ryšio kokybę.

Tikslas:

Projekto tikslas – realizuoti BO ryšio retransliatorių sukuriant autonominio skrydžio algoritmą tarpiniam retransliuojančiam BO bei suprojektuojant bei realizuojant retransliatoriaus įrangą.

Uždaviniai:

Tam, kad pasiekti šį tikslą reikalinga atlikti tokius uždavinius:

- atlikti literatūros analizę;
- sukurti autonominio skrydžio algoritmą;
- suprojektuoti retransliavimo sistemą;
- realizuoti algoritmą ir retransliatoriaus sistemą.

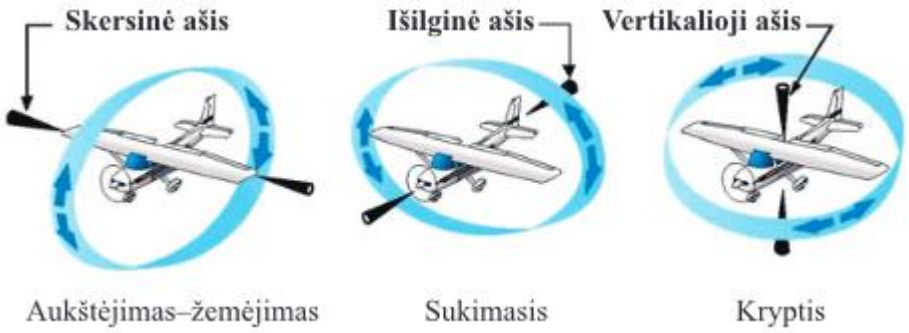
1. BEPILOČIŲ ORLAIVIŲ TIPAI

Renkantis bepiločio orlaivio tipą, atsižvelgiama į tokius aspektus, kaip stabilumas, pozicijos išlaikymo sugebėjimas, energijos sąnaudos, valdymo paprastumas. Multikopteriai atitinka šias savybes geriausiai ir atliekant šį darbą yra pasirenkamas kvadrakopteris.

1.1. Fiksuoto sparno bepiločiai orlaiviai

Pagrindinis tokio tipo orlaivio pranašumas yra tas, jog jį ore išlaikyti reikalinga mažiau energijos, lyginant su žemiau pateiktais orlaivių tipais. Fiksuoto orlaivio keliamąją galią sukelia sparnai. Tačiau orlaivis visada turi judėti kažkokia kryptimi tam, kad išsilaikytų ore. Toks orlaivio nuolatinis judėjimas ir nesugebėjimas išlaikyti pozicijos sukelia sunkumų, norint parinkti tinkamą poziciją kokybiškai ryšio retransliacijai. Kitas trūkumas yra orlaivio valdymas sudėtingomis oro sąlygomis. Pučiant stipriam gūsingam vėjui, orlaivio skrydžio charakteristikos smarkiai keičiasi. Tai pasunkintų saugaus autonominio skrydžio užtikrinimą [4].

Skirtingai nei sraigtasparniai ar multikopteriai, fiksuoto sparno orlaiviu atlikti manevrus galima tik trimis ašimis (1 pav.). Nepaisant mažesnio manevringumo, tokio tipo BO labai paplitę kariniame sektoriuje. Jie pasiekia didesnius greičius ir neretai gali skrisi aukščiau. Kadangi varikliai nenaudojami keliamajai galiai, tokio tipo bepiločiai orlaiviai pasižymi mažesniu triukšmo lygiu.



2 pav. Fiksuoto sparno orlaivio galimi manevrai [4]

1.2. Sraigtasparniai

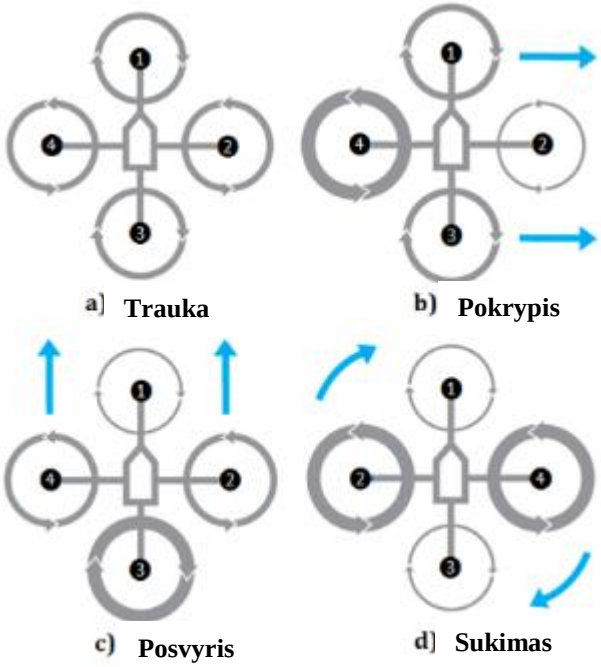
Maži sraigtasparniai turi puikias manevringumo savybes, tačiau jie pasižymi labai sudėtinga konstrukcija bei reikalauja sudėtingo jų valdymo. Nuo multikopterių sraigtasparniai skiriasi tuo, kad keliamoji galia sukeliama ir atliekami manevrai keičiant sraigto menčių kampus. Tam atlikti naudojamas svyruojantis diskas (angl. *swashplate*). Būtent šitoje vietoje ir atsiranda konstrukcijos sudėtingumas. Taip pat sraigtasparniai dažniausiai turi tik vieną pagrindinį sraigtą, todėl norint atstoti sukimo ir giroskopinį momentą, sukeltą besisukančio sraigto, kompensuoti naudojamas papildomas sraigtas ant

uodegos. Jį taip pat naudoja krypties keitimo manevrui atlikti. Kiti manevrai atliekami keičiant svyruojančio disko pakrypimą [5].

1.3. Multikopteriai

Multikopteriai pasižymi dideliu stabilumu bei manevringumu. Svarbu paminėti, jog norint kompensuoti sraigtų sukeltus sukimosi momentus, jų turi būti lyginis skaičius. Pusė sraigtų sukami prieš laikrodžio rodyklę, kita pusė – pagal laikrodžio rodyklę. Išimtis yra multikopteris su trimis sraigtais. Tokios konfigūracijos sukimosi momentai kompensuojami leidžiant vienam iš sraigtų keisti sukimosi ašies posvyrį [6].

Multikopteriai manevrus atlieka keisdami sraigtų apsukas (3 pav.). Kylant ir leidžiantis sraigtai sukami sinchronizuotai lėtinant ar greitinant apsukas visų sraigtų vienu metu. Kryptis keičiama panaudojant sraigtų sukeliamus momentus. Sulėtinami arba pagreitinami visi sraigtai, kurie sukasi į tą pačią pusę. Taip orlaivio kryptis keičiama priešinga greičiau besisukančių sraigtų kryptimi. Likę manevrai atliekami keičiant atskirų sraigtų greičius individualiai. Sraigtai sulėtinami toje pusėje į kuria norima kreipti orlaivį ir pagreitinami priešingoje [7].



15 pav. kvadrakopterio galimi manevrai: a) aukštėjimas žemėjimas; b) svyrimas į šoną; c) svyrimas į prieky/atgal; d) sukimasis į kairę/dešinę [8]

Yra keli aspektai, kurie turi būti sprendžiami projektuojant multikopterį. Multikopterio projektavimas apima visas kitame skyriuje išvardintas sritis. Galimos ir kitos projektavimo aplinkybės bet čia išvardinti svarbiausi ir pagrindiniai projektavimo aspektai kurie lemia skrydžio kokybę, krovinio galimybes bei skrydžio laiką.

1.3.1. Keliamoji galia

Konstruojant kvadrakopterį ar bet kokį kitą multikopterį, svarbu pasirinkti tinkamus propelerius bei variklius. Nuolatinės srovės elektriniai varikliai yra dviejų tipų: senesni ir jau nebenaudojami multikopteriuose šepetėliniai varikliai bei bešepetėliniai. Bešepetėliniai nuolatinės sroves (BNS) varikliai yra dviejų tipų – vidiniai ir išoriniai. Vidiniuose ašis ir šerdis sukasi iš nuolatinių magnetų, kol korpusas išlieka stacionarus. Išoriniuose viskas atvirkščiai – aplink šerdį sukasi korpusas, kurio sienelėse yra nuolatiniai magnetai. Nors vidinius lengviau įdiegti dėl nejudančio korpuso, išoriniai sukuria daug daugiau traukos. Pagrindinis tokių variklių parametras yra greičio konstanta K_v [aps./V], matuojamas apsukom per voltus [9].

Prie variklių turi būti parenkami tinkami propeleriai. Pagrindiniai parametrai, renkantis propelerį, yra jo medžiaga, skersmuo, mentės kampas ir menčių kiekis. Minkštesnių medžiagų propeleriai, esant tam tikroms apsukom, praranda efektyvumą. Didinant menčių kampą ir kiekį, didinama propelerio sukeliama trauka, bet tuo pačiu mažinamas jo efektyvumas bei didinami variklio galios reikalavimai. Didinant propelerio skersmenį, didėja efektyvumas, dėl padidinto oro srauto per laiko vienetą. Tačiau didesnis propeleris reikalauja didesnį sukimo momentą turinčių variklių, be to, didesnis propeleris turės daugiau inercijos ir mažins sistemos reaktyvumą [9].

Kvadrakopterio trauka skaičiuojama pagal formulę:

$$T = \frac{\pi}{4} D^2 \rho v \Delta v, \tag{1}$$

čia T – trauka, D – propelerio skersmuo, ρ – oro tankis. Kadangi

$$v \approx \frac{1}{2} \Delta v,$$

čia v – oro greitis ties propeleriu, Δv – propelerio įgreitintas oro greitis.

Įstačius v į (1) gaunama:

$$T = \frac{\pi}{8} D^2 \rho (\Delta v)^2. \tag{2}$$

Į (3) įstačius Δv iš galios formulės:

$P = \frac{T(\Delta v)}{2}$,

gaunama traukos formulė:

$$T = \frac{\pi}{8} D^2 \rho (\frac{2P}{T})^2,$$

$$T = \left[\frac{\pi}{2} D^2 \rho P^2\right]^{\frac{1}{3}}. \tag{3}$$

Tam, kad išlaikyti kvadrakopterį ore, traukos jėga turi būti didesnė, negu kvadrakoperio svoris:

$$m = \frac{T}{g}, \tag{4}$$

čia g – laisvo kritimo pagreitis.

Propelerio galia apskaičiuojama pagal formulę (5):

$$P = C_p \left(\frac{aps.}{1000}\right)^{f_p}, \quad (5)$$

čia C_p – propelerio konstanta (gaunama iš propelerio specifikacijos), *aps.* – propelerio apsukos, f_p – propelerio galios faktorius (gaunamas iš propelerio specifikacijos). Įstačius (5) į (3), gaunama traukos formulė priklausanti nuo propelerio parametrų ir variklio apsukų.

$$T = \left[\frac{\pi}{2} D^2 \rho \left(C_p \left(\frac{aps.}{1000}\right)^{f_p}\right)^2\right]^{\frac{1}{3}}. \quad (6)$$

Formulė (6) apskaičiuoja trauką vienam propeleriuj. Kvadrakopterio pakeliama masė paskaičiuojam įstačius formulę (6) į (4) ir viską padauginus iš propelerių kiekio (kvadrakopterio atveju 4):

$$m = 4\frac{\left[\frac{\pi}{2} D^2 \rho \left(C_p \left(\frac{aps.}{1000}\right)^{f_p}\right)^2\right]^{\frac{1}{3}}}{g}.$$

Kadangi su šita formule paskaičiuojama kokį svorį kvadrakopteris gali išlaikyti ore atlikti manevrams reikalinga papildomas traukos perteklius. Dauguma šaltinių naudoja trauką dvigubą reikalingai išlaikyti savo masę ore [10].

1.3.2. Energijos sąnaudos

Pačios populiariausios baterijos naudojamos multikopteriuose, yra ličio polimero (LiPo). Jos pasižymi didele energijos koncentracija iki 250 Wh/kg [11]. LiPo baterijos celių konfigūracija ir jos talpa yra svarbi renkantis bateriją konstruojant multikopterį. Įprastinės LiPo celė turi 3,7 V nominalią įtampą. Pilnai įkrautos celės įtampa siekia 4,2 V. LiPo baterijos sudaromos iš keletos celių sujungtų lygiagrečiai (S) ir paraleliai (P). Jungiant lygiagrečiai didinama baterijos įtampa, o jungiant paraleliai didinama talpa. LiPo baterijos taip pat pasižymi didele iškrovos galimybe. Gamintojai baterijas pagal iškrovos galimybę žymi C reitingu. Baterijos maksimali iškrova paskaičiuojama dauginant C reitingą iš baterijos talpos (Ah). Renkant bateriją reikia parinkti tinkamą celių kiekį, C reitingą, bei talpą turinčią bateriją. Nuo celių kiekio priklausys įtampa, nuo kurios priklauso variklio apsukų kiekis. Į C reitingą atsižvelgiama pagal bendrą multikopterio srovės reikalavimą. Talpa parenkama pagal reikalingą skrydžio laiką [10]. Reiktų paminėti, kad LiPo baterijų dėl saugumo ir tarnavimo laiko sutrumpėjimo negalima pilnai iškrauti. Saugi riba yra 80-90% baterijos talpos.

Norint apskaičiuoti apytikslį skrydžio laika naudojama tokia formulė:

$$T = \frac{0.8\, I_c}{I_{vid}},$$

čia I_c – baterijos talpa (Ah), I_{vid} – vidutinė sistemos traukiama srovė (A)

1.3.3. Stabilizacija

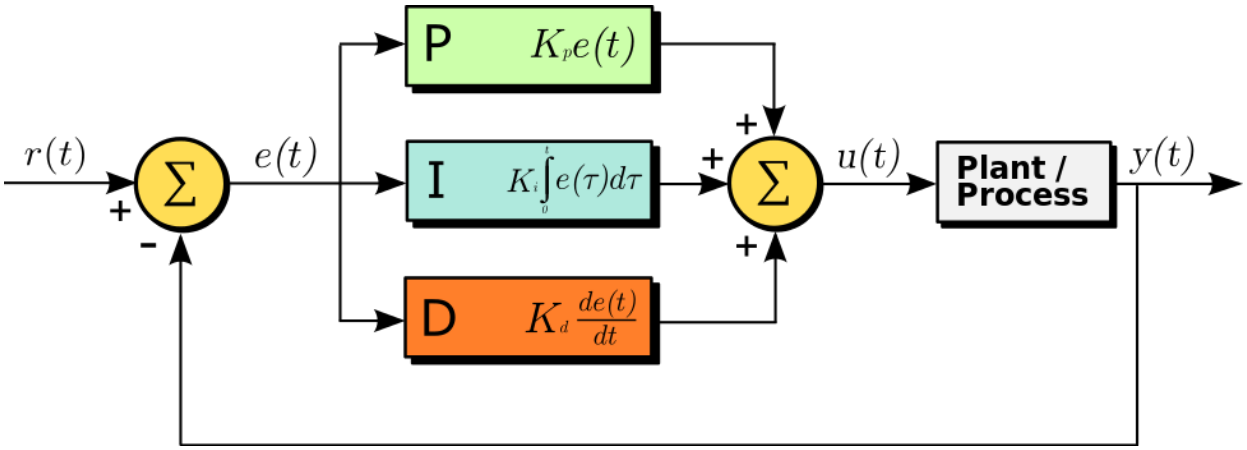
Kadangi multikopteriai turi daugiau nei vieną sraigtą, kuriuos suka atskiri motorai, tokių orlaivių, valdymas be stabilizacijos sistemų būtų tiesiog neįmanomas. Operatorius turėtų visą savo dėmesį sutelkti ties orlaivio stabilizavimu ir iš esmės nebesugebėtų jo tinkamai skraidinti. Būtent dėl to į multikopterius yra sumontuojama įranga kuri stabilizuoja orlaivį ir iš esmės pagerina jo valdymą. Dažniausiai tai yra giroskopų (kartais ir kitų sensorių) ir PID reguliatorių kombinacija. Giroskopas padeda nustatyti orlaivio padėtį erdvėje (trys kampai: polinkio, posvyrio, krypties). Naudojant PID ir giroskopą, multikopterio sraigtai nuolatos keičia greitį bandydami kompensuoti atsiradusias klaidas ir taip išlaiko orlaivį stabilioj pozicijoj.

1.3.4. PID reguliatorius

PID reguliatorius tai grįžtamo ryšio mechanizmas dažnai naudojamas industrinėse kontrolės sistemose. PID reguliatorius nuolatos skaičiuoja klaidą, kuri yra skirtumas tarp esamos (išmatuotos) ir nustatytos pozicijos, ir padaro korekcijas pagal tris parametrus: proporcinį (P), integruojantį (I) ir diferencijuojantį (D). Reguliatorius mėgina sumažinti paklaidą, bėgant laikui keisdamas valdymo kintamąjį, atlikdamas svertinę sudėtį [8]:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt}$$

čia K_p, K_i, K_d yra neneigiami koeficientai, kurie nusako proporcinės, integruojančios bei diferencijuojančios paklaidos svorį. Paprasčiau jie vadinami *P*, *I* ir *D* atitinkamai.



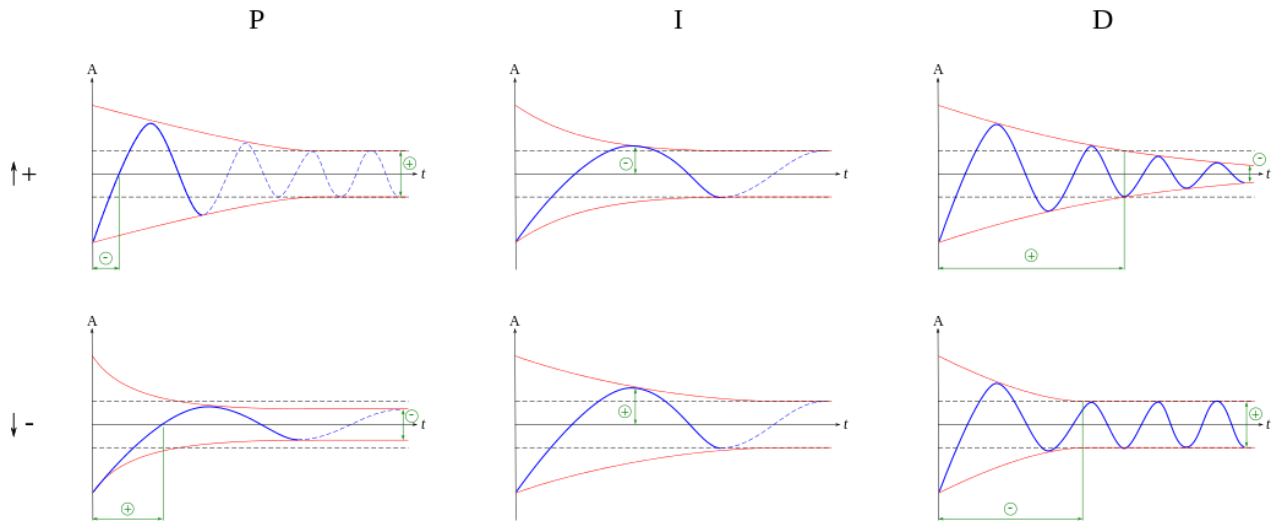
16 pav. Tipinis PID procesas kur r(t) norima pozicija, e(t) paskaičiuota klaida, u(t) kompensacija, y(t) esama pozicija [8]

Keičiant šias tris reikšmes gaunamas skirtingas galutinis rezultatas arba tikslumas pasiekiamas vėliau ar ankščiau su daugiau ar mažiau osciliavimo. Kartais netinkami parametrai gali sukelti vis didėjančius osciliavimus. Žemiau lentelėje pateikti kiekvieno parametro padidinimo efektai. Iš grafiko

(5 pav.) matosi, kaip norimas rezultatas pasiekiamas praėjus laikui, jei PID parametrai didinami ar mažinami [8].

1 lentelė. *P*, *I* ir *D* parametrų įtaka valdymo parametrams

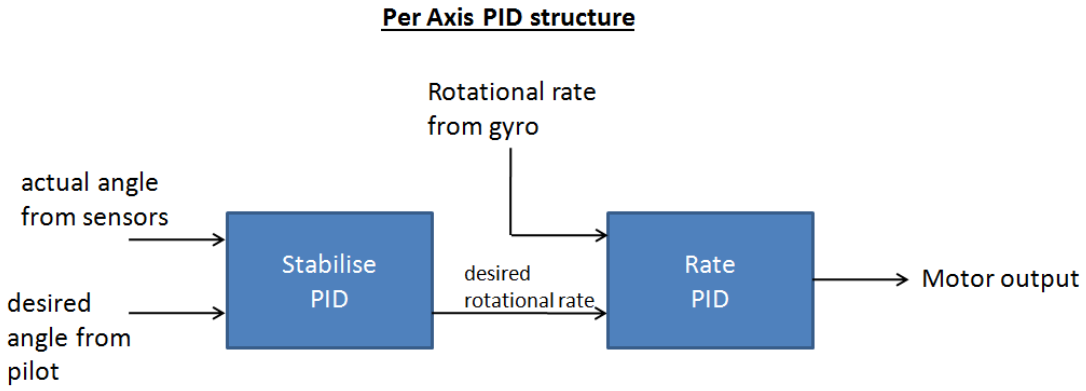
Parametras	Pakylimo laikas	Perkompensavimas	Nusistovėjimo laikas	Nusistovėjimo klaidos	Stabilumas
P	Sumažėja	Padidėja	Mažai pasikeičia	Sumažėja	Suprastėja
I	Sumažėja	Padidėja	Padidėja	Pašalinamos visai	Suprastėja
D	Mažai pasikeičia	Sumažėja	Sumažėja	Teoriškai jokio efekto	Pagerėja jei D mažas



17 pav. PID parametrų įtaka valdymui

1.3.5. PID reguliatoriai multikopteryje

Kaip žinoma, PID reguliatoriai naudojami stabilizuoti multikopterius. Tipinė multikopterio stabilizavimo ties viena ašimi struktūra pateikta 6 pav. Iš šios schemos matoma, kad naudojami du PID reguliatoriai. Vienas, kuris reguliuoja sukimosi (ties viena ašimi) greitį, ir kitas, kuris reguliuoja motoro apsukas. Pirmas PID gauna norima kampą iš piloto bei esamą orlaivio kampą iš sensorių. Antrasis PID turi sukimosi (apie ašį) informaciją iš pirmojo PID bei sukimosi greitį iš giroskopo. Galiausiai galutiniam proceso taške gaunamos reikiamos apsukos motorui(-ams) kuris(-ie) suks orlaivį reikiama kryptimi [8].



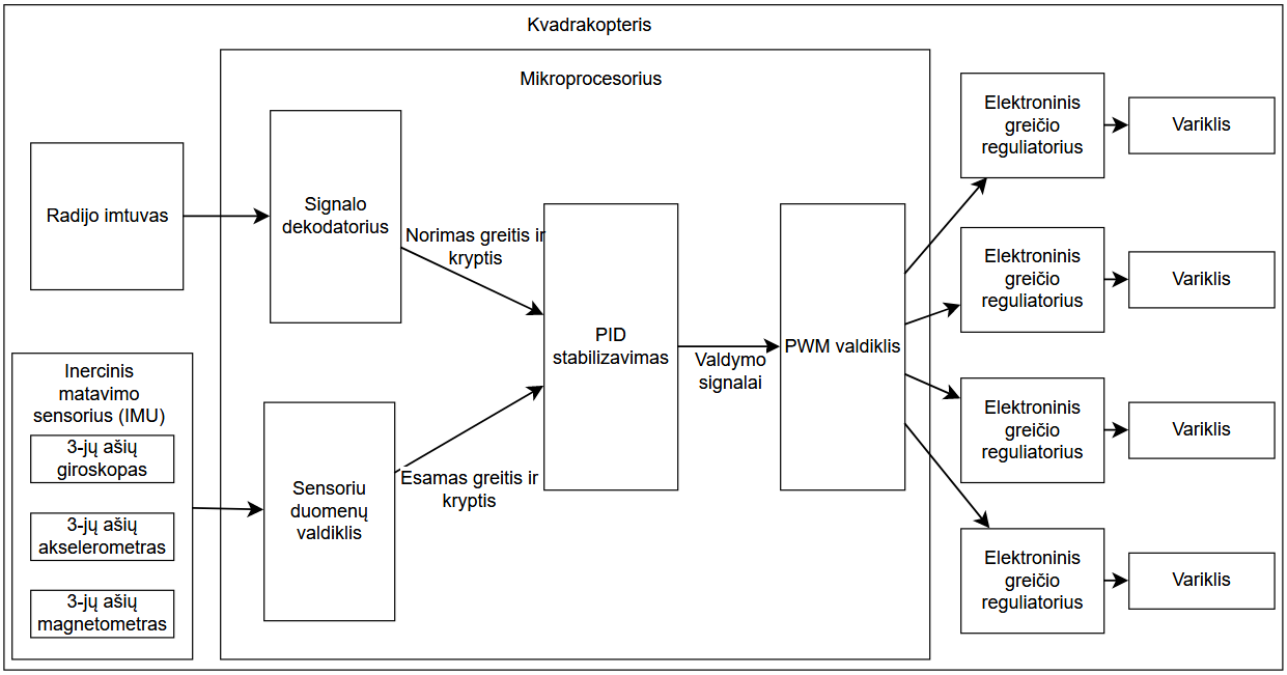
18 pav. Vienos Multikopterio ašies stabilizavimo struktūra [8]

1.3.6. Korpuso aerodinamika

Kalbant apie multikopterių korpuso aerodinamines savybes, yra du pagrindiniai aspektai. Tai korpuso sukeliama pasipriešinimo jėga bei parazitinė keliamoji jėga. Sumažinant laisvai kabančių laidų bei antenų kiekį ar juos tvarkingai išdėliojant, galima pasiekti iki 60% mažesnį pasipriešinimą. Tačiau suteikus aptakų korpusą parazitinė keliamoji jėga padidėja kone dvigubai [12]. Taip pat norint pasiekti didesnį propelerių efektyvumą, juos galima įmontuoti į aerodinaminius vamzdžius. Tokiu būdu panaikinama turbulencija, kuri atsiranda propelerių galuose dėl srauto užlinkimų ties pačiu propelerio galu [13].

1.3.7. Tipinė kvadrakopterio schema

Multikopteriai susideda iš daug skirtingų atskirų dalių ir posistemių, tačiau pagrindinės jų būna visų tipų multikopteriuose. Žemiau pateiktame 7 pav. pavaizduota standartinė minimali kvadrakopterio schema.



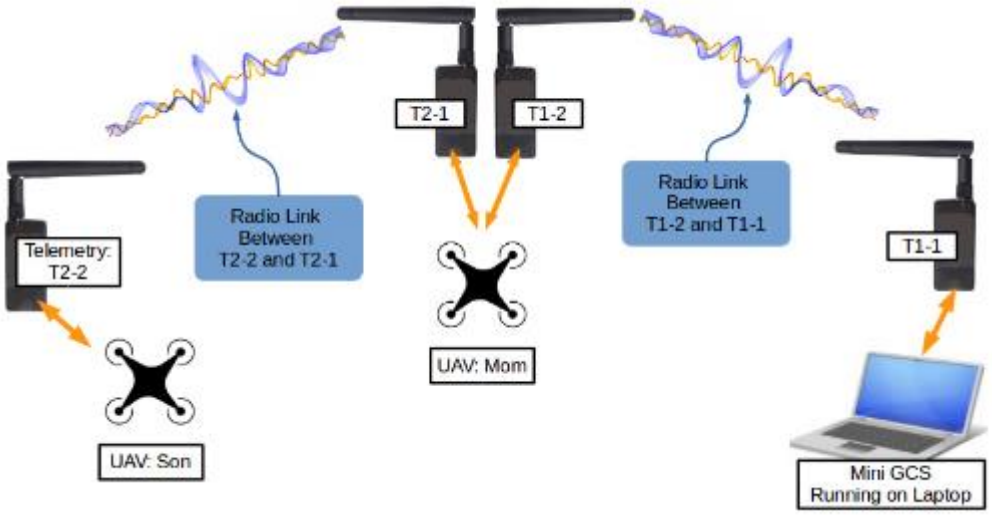
19 pav. Kvadrakopterio standartinė blokinė diagrama

Paprastai multikopteriai turi tokius pagrindinius elementus: inercinį sensorių, radijo ryšio imtuvą, PID (nebūtinai) stabilizavimo valdiklį, pulso pločio moduliavimo (PWM) valdiklį bei variklius, prijungtus su elektroniniais greičio valdikliais. Inercinis sensorius (IMU) matuoja tokius orlaivio judėjimo parametrus visomis trimis ašimis, kaip akceleraciją bei pasisukimo aplink ašis kampus. IMU per sensorių valdiklį perduoda duomenis tiesiai į stabilizavimo valdiklį, dažniausiai PID. Radijo ryšiu gauti valdymo signalai taip pat patenka tiesiai į stabilizavimo valdiklį. PID valdiklis iš sensorių duomenų ir gauto valdymo signalo sugeneruoja stabilizuotą valdymo signalą, kuris perduodamas į PWD valdiklį. Iš ten valdomos variklių apsukos per elektroninius greičio reguliatorius [14], [15].

2. RETRANSLIACIJA

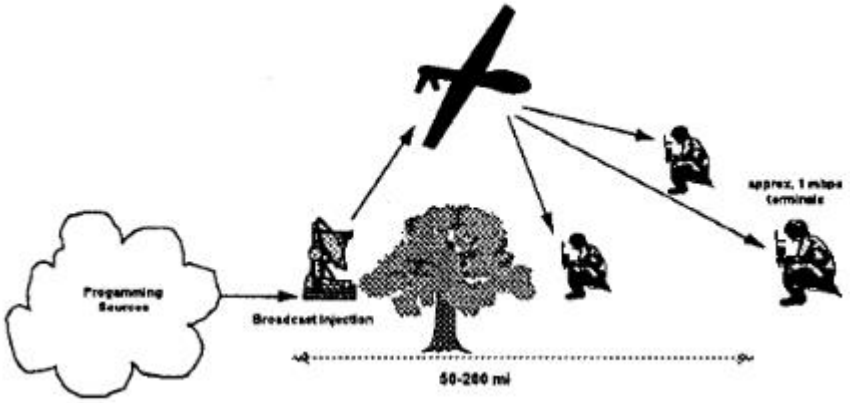
Norint retransliuoti signalą reikia užtikrinti, kad priimtas ir retransliuotas signalai netrukdytų vienas kitam. Tai galima užtikrinti naudojant skirtingus dažnius ir technologijas.

Darbe [16] apie BO retransliatorių naudojama 915 MHz dažnių juosta sudalinta į 50 kanalų. Naudojant FHSS ir TDM technologijas buvo pasiekta duomenų perdavimas skirtingais kanalais juos keičiant kas 100 ms niekada nepasirenkant tų pačių dviem skirtingomis kryptimis. Kai vieną kanalą naudoja valdymo stotis ir retransliatoriaus imtuvas, kitą kanalą naudoja retransliatoriaus siųstuvas ir galinio BO imtuvas. Paleidžiant sistemą, siųstuvai susisinchronizuoja laikrodžius ir pradeda TDM procesą. Šiame darbe TDM užtikrino ryšį į abi puses, kai vieną dalį kadro laiko signalas siunčiamas ir kitą dalį priimamas. Panaudojant FHSS pašalinami trukdžiai tarp skirtingų ryšių šiuo atveju tarp BO ir tarp BO ir valdymo stoties (8 pav.).



20 pav. BO retransliuojančios sistemos vaizdas [16]

Kitam darbe [17], kuris skirtas kariuomenei, aprašyta panaši sistema, kuri labiau skirta žemė–BO–žemė retransliacijai. Darbe minima, kad naudojamos kelios skirtingos technologijos ir daug skirtingų dažnių, kaip X, L juostos bei UHF. Tačiau kaip ir dauguma karinių darbų, šitame nekalbama apie tokias detales, kaip retransliavimo technologiją. Kaip matoma iš 9 pav., šio darbo sistema skirta atstumo padidinimui jau naudojamų ryšio priemonių. Darbe [17] rašoma, kad toks BO turi būti pritaikomas jau egzistuojančioms ir suplanuotoms ryšio sistemoms.



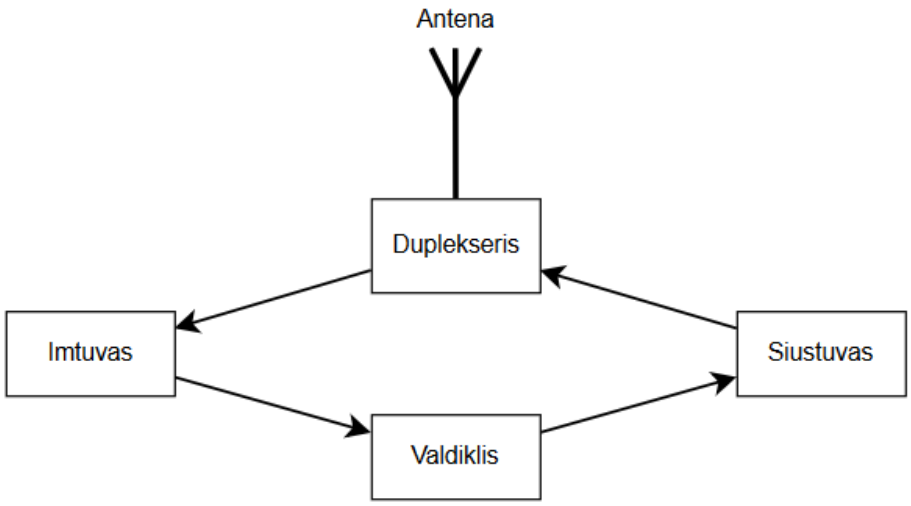
21 pav. Žemė-BO-žemė radijo komunikacijų ryšio retransliavimo schema [17]

Ryšiui tarp palydovų palaikyti taip pat naudojamas retransliavimas. Darbe [18] išvardinti skirtingi dažniai kuriuos naudoja palydovai. Dažniausiai tai arba UHF, arba VHF. Tokie palydovai kaip OSCAR 7 ir 8 taip pat turi kelis atskirus retransliatorius, kuriuos naudoja pakaitomis. Taip pat, kadangi palydovai priima ir siunčia signalus skirtingas dažniais ir skirtingomis antenomis, jie gali priimti ir siųsti duomenis tuo pačiu metu.

Dar viena kryptis, kur plačiai naudojamas retransliavimas, yra tarp automobilinė komunikacija VANET. Tai tinklas, kuris sujungia daug atskirų automobiliu tam, kad galėtų perduoti informaciją apie galimus eismo įvykius [19].

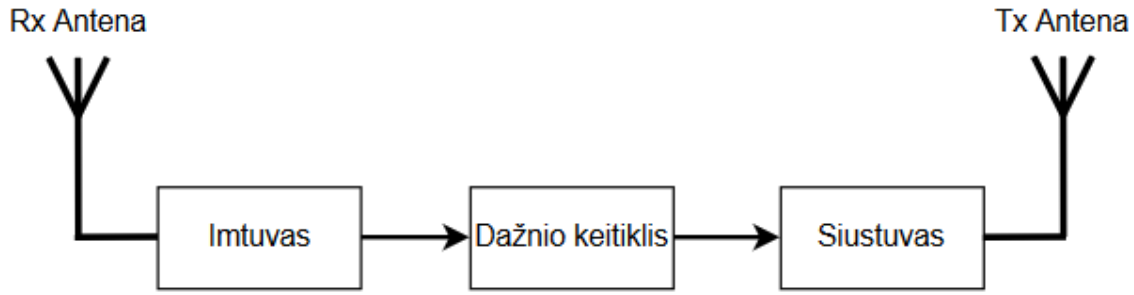
2.1. Tipinės retransliatorių schemos

Retransliatoriai gali naudoti vieną arba dvi antenas. Naudojant vieną anteną yra pasinaudojamas duplekso metodas kai signalas yra nesiunčiamas tol kol yra priimamas ir atvirkščiai. Paveikslėlyje 10 pavaizduota supaprastinta schema naudojant signalo dupleksavimą [20].



22 pav. tipinė supaprastinta retransliatoriaus schema naudojant duplekserį

Naudojant dupleksavimą negalima tuo pačiu metu priimti ir siųsti duomenų, kadangi naudojama tapati antena. Tai išsprendžiama naudojant dvi skirtingas antenas priimti ir siųsti retransliuojamam signalui. Tokia sistema pavaizduota 11 pav.



23 pav. Supaprastinta skirtingų dažnių retransliatoriaus schema

Tam kad, priimtas ir išsiųstas signalai netrukdytų vienas kitam naudojami skirtingi siųstuvo ir imtuvo dažniai [20].

3. RETRANSLIATORIAUS (ORLAIVIO) PARENGIMAS

Retransliatoriui konstruoti svarbiausias parametras yra skrydžio laikas. Jis parinktas atsižvelgiant į dabartinius rinkoje siūlomus mėgėjiškus bei profesionalius multikopterius (neįtraukiami specializuoti, karinės paskirties bei reti savadarbiai multikopteriai). Jų skrydžio laikas svyruoja nuo 5 min iki šiek tiek daugiau nei 30 min. Retransliatoriaus skrydžio laikas parinktas nemažiau 30min. Tokiam laikui pasiekti reikalinga efektyvi sistema. Dėl didesnio skrydžio laiko ir efektyvumo, retransliatoriui parinkti propeleriai, kurie turėtų didelį diametrą. Žinoma, kad didesni propeleriai pasiekia didesnį efektyvumą mažesnėmis apsukomis, todėl pasirinkti 15 colių diametro ir 5,5 colių žingsnio propeleriai. Tai pat žinoma, kad norint įsukti galingesnį motorą, reikės daugiau energijos, todėl pasirinkta 6 celių 6000mAh baterijos sistema, kuri sveria 980 g. Tam taip pat reikalingi mažų apsukų motorai, kurie turi didelį skaičių elektromagnetų ir palaiko 6 celių baterijos įtampą. Tam tikslui parinkti 400 KV motorai su 20 statorių. Rėmo dydis parenkamas atsižvelgiant į propelerių dydį. Rėmo dydis matuojamas jo įstrižainės ilgiu kitaip dar vadinamu važiuoklės baze. Parinkus 15 colių propelerius rėmas parinktas 600 mm.

3.1. Parametrų parinkimas ir skaičiavimai

Tikslūs parametrai parinkti naudojant multikoperiams skirta skaičiavimo programą „eCalc“. Suvedus parinktus parametrus į skaičiuoklę, apskaičiuotas ilgiausias skrydžio laikas siekė 26,9 min. Naudojant skaičiuoklę parametrai buvo pakeisti tam, kad būtų pasiektas norimas rezultatas, tai yra skrydžio laikas didesnis nei 30 min.

Su skaičiuokle parinkti nauji parametrai pateikti žemiau.

Pasirinkto propelerio parametrai:

- Ilgis (diametras): 16 coliai (406,4 mm);
- Žingsnis: 5,5 coliai (139,7 mm);

Pasirinkti motorų parametrai:

- Apsukos per voltą (KV): 380 aps./min/V;
- Magnetų skaičius: 22.

Pasirinkta baterija:

- Talpa: 8000 mAh;
- C reitingas: 15 – 30;
- Masė: 1,110 kg;
- Maksimali įtampa: 25,2 V.

Orlaivio rėmas parinktas taip, kad propeleriai tilptų palikdami kuo mažesnį tarpą, netrukdydami vienas kitam tiek mechaniškai, tiek dėl sukeliamos turbulencijos propelerio galuose. Pasirinkto rėmo važiuoklės bazė yra 650 mm. Visa kita įranga parenkama pagal prieš tai įvardintos įrangos

suderinamumą. Naujų apskaičiuotų parametrų ir pirminių parinktų parametrų palyginimas pateiktas 2 lentelėje.

2 lentelė. Parinktų ir paskaičiuotų retransliatoriaus parametrų palyginimas

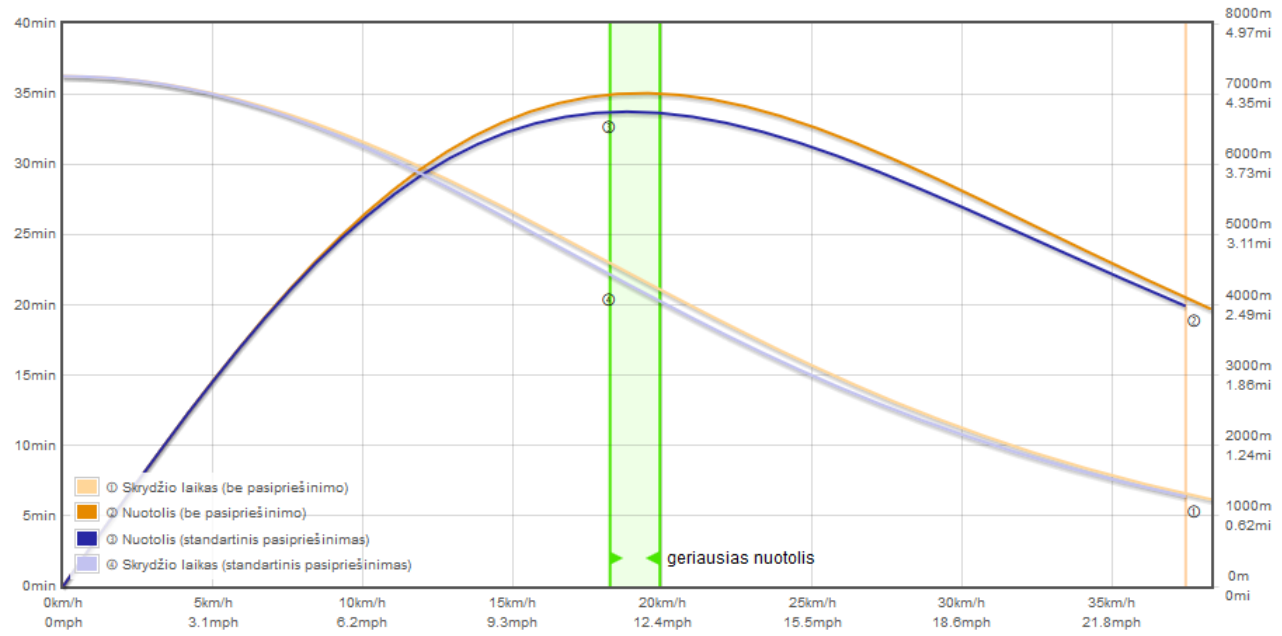
Parametras	Parinktas parametras	Apskaičiuotas parametras
Propelerio ilgis	15 colių	16 colių
Propelerio žingsnis	5,5 coliai	5,5 coliai
Motoro KV	400	380
Motoro magnetų kiekis	20	22
Baterijos talpa	6000	8000
Baterijos C reitingas	20	15
Baterijos celių kiekis	6	6
Rėmo bazė	600 mm	650 mm

Detalūs apskaičiuoti orlaivio parametrai išvardinti žemiau pateiktoje lentelėje.

3 lentelė. Su „eCalc“ apskaičiuoti retransliatoriaus parametrai

parametras	reikšmė
Skrydžio laikas	35,9 min
Traukos ir masė santykis	3,5 : 1
Apytikslis svoris	2,412 kg
Maksimalus paviršiaus greitis	39 km/h
Maksimalus vertikalus greitis	8,2 m/s
Maksimalus krovinio svoris	4,939 kg
Srovė reikalinga kyboti vietoje	11,92 A
Maksimali srovė	90,17 A
Apskaičiuotas sistemos efektyvumas kybant	78 %
Apskaičiuotas efektyvumas maksimaliu pajėgumu	66 %

Žemiau 12 pav. pateikiamas grafikas su nuotolio ir skrydžio trukmės priklausomybė nuo orlaivio greičio.



24 pav. Nuotolio ir skrydžio laiko grafikas

Iš grafiko 12 pav. matomas maksimalus orlaivio skrydžio nuotolis 7000 m yra pasiekiamas skrendant 20 km/h greičiu. Taip pat matosi, kad didžiausia skrydžio trukmė pasiekiama kybant vietoje. Grafike taip pat matomas optimalius greitis norint išlaikyti gerą santykį tarp atstumo ir skrydžio laiko. Tai yra, orlaivis turi skristi apytiksliai 12 km/h greičiu. Skrendant tokiu ar žemesniu greičiu, minimalus skrydžio laikas siekia apie 30 min, o maksimalus nuotolis siekia ~ 6 km.

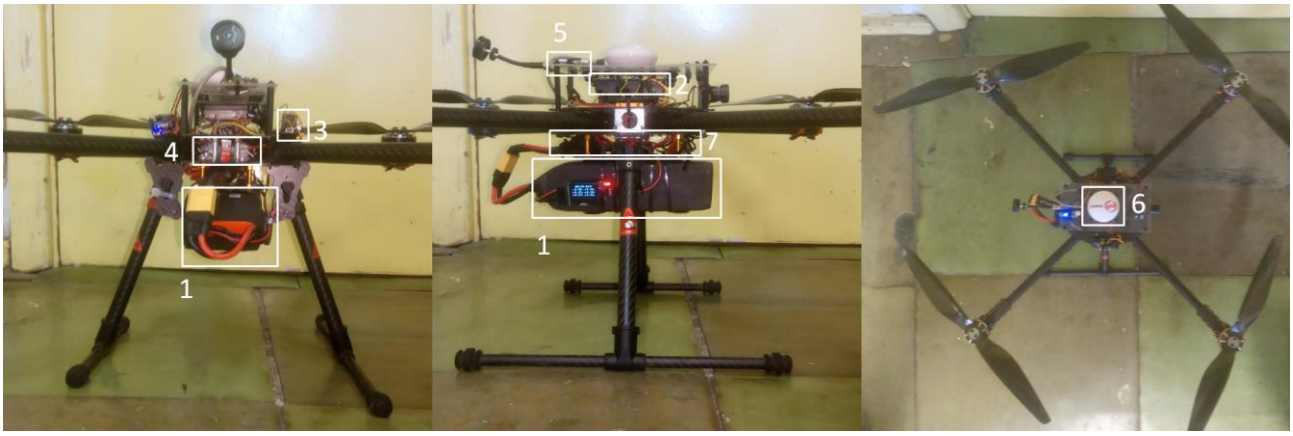
3.2. Retransliatoriaus konstravimas

Konstruojant retransliatorių buvo parinktos dalys, atitinkančios ankščiau paminėtus orlaivio bei retransliavimo įrangos parametrus. Renkant retransliatoriaus prototipą, dėmesys buvo skiriamas parametrų atitikimui ir funkcionalumui, tačiau nebuvo atsižvelgiama į estetiką ir saugumo užtikrinimui. Konstruojant retransliavimo ryšio įrangos prototipą buvo atsižvelgiama į ryšio kokybės ir patikimumo užtikrinimą.

3.2.1. Orlaivis (retransliatorius)

Bepilotis orlaivis retransliatorius surinktas taip, kad atitiktų skrydžio parametrus apskaičiuotus ankstesniame poskyryje. Žemiau pateiktame 13 pav. pavaizduotas surinktas bepilotis orlaivis iš trijų pusių. Paveikslėlyje sužymėti pagrindiniai komponentai. Didesniam stabilumui užtikrinti baterija (1) sukonstruota orlaivio dugne dėl savo didelio svorio (48 % orlaivio masės). Dėl kuo mažesnio atstumo nuo baterijos, maitinimo įranga (7) išdėliota tiesiai virš jos. Greičio reguliatoriai sumontuoti virš maitinimo šaltinio, taip sumažinant stambiu laidų ilgį bei pritraukiant juos arčiau variklių jungčių. Tam, kad sumažinti vibracijų poveikį, skrydžio kompiuteris (2) pakabintas ore ant guminių jungčių. Galiausiai

GPS imtuvas pritvirtintas orlaivio viršuje toli nuo orlaivio trukdžių bei tiesioginio matomumo zonoj su palydovais. Taip pat netoli skrydžio kompiuterio prijungtas ryšio modulis (5) bei valdiklis skirtas apdoroti telemetrijai (3).



25 pav. Surinktas retransliatorius (orlaivis) 1 – baterija, 2 – skrydžio kompiuteris, 3 – telemetrijos (siuntimo, apdorojimo) valdiklis, 4 – greičio reguliatoriai (ESC), 5 – ryšio modulis, 6 – GPS imtuvas

Surinkto orlaivio masė siekia 2,3 kg.

Retransliavimo įranga pakabinama orlaivio dugne po baterija. Kadangi retransliavimo įranga turi atskirą maitinimo šaltinį jai prie orlaivio prijungti reikalingi tik telemetrijos jungtį iš skrydžio kompiuterio.

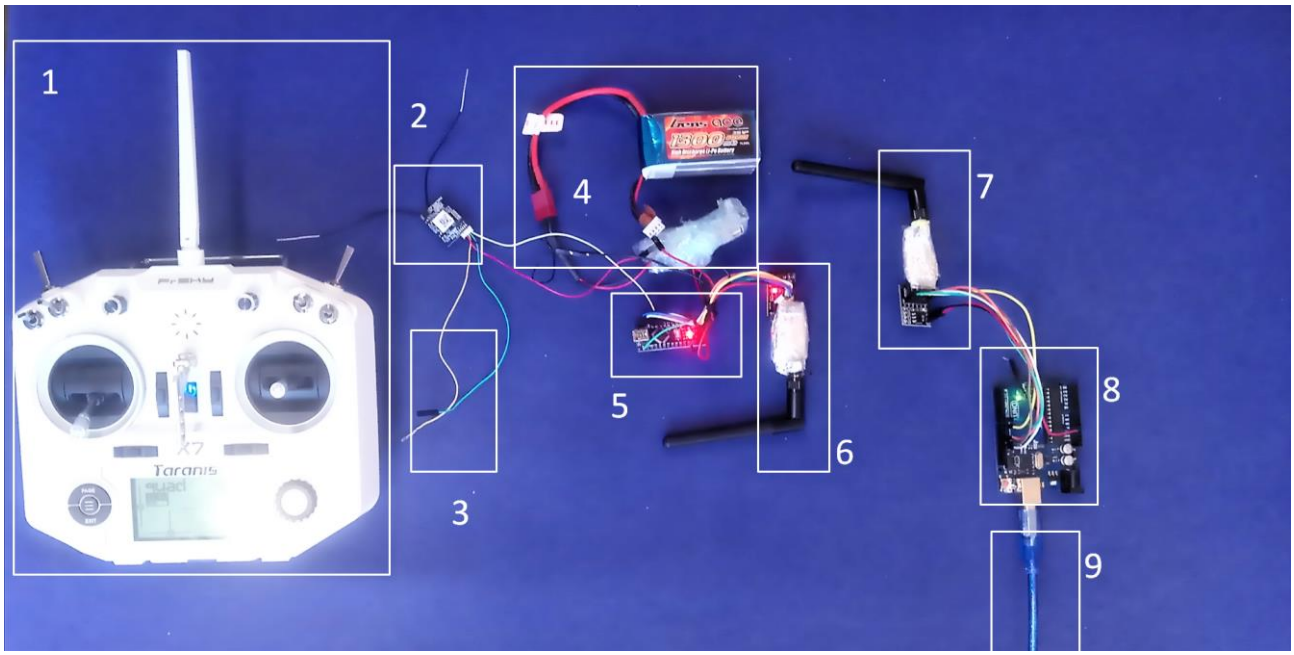
3.2.2. Retransliavimo įranga

Žemiau patiktame 14 pav. matosi viso retransliatoriaus ryšio įrangos komponentai. Ryšio signalo sklidimas tarp šių elementų susideda iš dviejų etapų. Pirmiausia signalas iš siųstuvo (1) perduodamas į imtuvą (2) vėliau signalas apdorotas valdiklio (5) siunčiamas siųstuvu (6) į imtuvą (7) kur vėliau signalas valdikliu (8) perduodamas į valdomą orlaivį (9). Atliekant projektavimą, valdomas orlaivis buvo pakeistas simuliatoriumi. Siunčiant signalą tarp 1 ir 2 bei tarp 6 ir 7 yra naudojami skirtingi protokolai bei dažniai, tačiau abi ryšio grandys yra abipusės ir naudoja duplekso principą. Ryšys tarp siųstuvo ir imtuvo 1 ir 2 naudoja 2,4 GHz dažnių juostą ir dažnio šokinėjimo (FHSS) moduliaciją tuo tarpu ryšys tarp siųstuvo ir imtuvo 6 ir 7 naudoją dažnio slankiojimo (FSK) moduliaciją ir vieną kanalą kurio dažnis yra 2,511 GHz ir plotis siekia 2 MHz.

Siųstuvai/imtuvai 6 ir 7 turi galimybę keisti kanalą, duomenų perdavimo greitį bei siuntimo galią. Šie parametrai buvo parinkti taip:

- Kanalas parinktas taip kad nesikirstu su dažnių juosta kurią naudoja siųstuvas ir imtuvas 1 ir 2 todėl pasirinktas kanalas kurio dažnis yra 2,511 GHz;

- Siuntimo greitį galima nustatyti į 2 Mbps, 1 Mbps ir 250 kbps. Pasirinktas 250 kbps greitis, nes juo pasiekiamas didžiausias veikimo atstumas bei tokio greičio daugiau nei pakanka retransliuojamiems duomenims siųsti;
- Siųstuvo galingumas nustatytas į maksimalų pajėgumą. Šiuo režimu įjungiamas stiprintuvas, kuris duoda 20 mW signalo stiprinimą.



26 pav. Visa retransliatoriaus ryšio įranga 1 – siųstuvas, 2 – imtuvas, 3 – retransliatoriaus skrydžio kompiuterio jungtis, 4 – retransliatoriaus ryšio įrangos atskiras maitinimo šaltinis, 5 – retransliatoriaus siųstuvo valdiklis, 6 – retransliatoriaus siųstuvas, 7 – valdomo orlaivio imtuvas, 8 – valdomo orlaivio imtuvo valdiklis, 9 – jungtis į valdomą orlaivį

Paveiksle 26 taip pat matosi trys sistemos taškai. Operatorius turi tik siųstuvą 1. Retransliatorius su savimi pakelia visas įrangos dalis nuo 2 iki 6. Dalys 7 ir 8 montuojamos ant norimo valdyti orlaivio.

3.3. Parametrų palyginimas su bandymais

Sukonstravus retransliatoriaus orlaivį buvo atlikti keletas bandymų, skirtų patikrinti praeitame poskyryje apskaičiuotus parametrus realiomis sąlygomis.

3.3.1. Skrydžio trukmės bandymas

Šio bandymo tikslas yra išmatuoti orlaivio skrydžio maksimalią trukmę realiomis sąlygomis. Bandymas atliktas taip:

Pilnai pakrautas orlaivis įjungiamas ir iškarto pakeliamas į 50 m aukštį. Pasiekus 50 m aukštį, orlaivis perjungiamas į pozicijos laikymo rėžimą, kur jis paliekamas iki kol įsijungs baterijos išsikrovimo apsauga, kurios metu orlaivis pats nusileidžia saugiu greičiu iki žemės. Skrydžio laikas matuojamas pasinaudojus laikrodį, kuriuo laikas pradėtas matuoti orlaiviui pradėjus kilti ir sustabdytas orlaiviui nusileidus. Taip pat laikas pamatuotas naudojant vidinį orlaivio laikrodį ir automatinius

žurnalus, kuriuose kartu su laiku įrašoma visa informacija apie orlaivio būklę. Vidinis orlaivio laikrodis laiką skaičiuoja milisekundėmis nuo skrydžio pradžios.

Oro sąlygos ir orlaivio pakilimo momentas matosi 13 pav.

27 pav. Retransliatorius kyla į 50m aukštį skrydžio laiko bandymui

Vėjo greitis bandymo metu siekė 5m/s, o vėjo gūsiai siekė 8m/s greitį.

Orlaiviui nusileidus laikrodžiu matuotas laikas rodė 39 min ir 42s. Atsidarius orlaivio žurnalą pastebėta, kad baterijos išsikrovimo apsauga užregistruota 2225346 ms, tai yra 37 min ir 4 s. Ankstesniame skyriuje su programa apskaičiuotas skrydžio laikas yra 35,9 min. Realiomis sąlygomis pamatuotas skrydžio laikas ilgesnis 1 min ir 8 s.

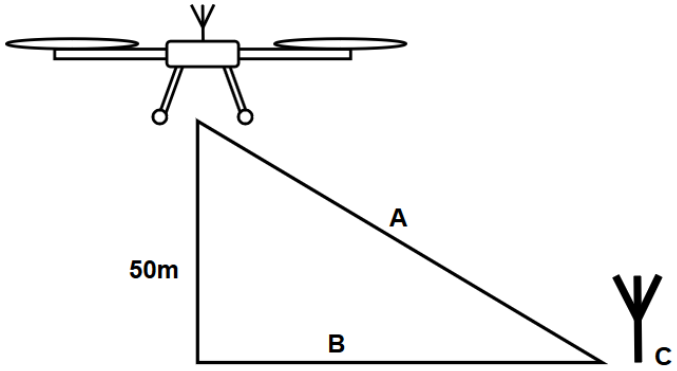
3.4. Ryšio kokybės parametrų parinkimas atliekant bandymą

Norėdami retransliuoti bepiločio orlaivio valdymo signalą turime užtikrinti patikimą radijo ryšio signalą tarp retransliatoriaus ir valdomo bepiločio orlaivio. Tam užtikrinti buvo atliktas ryšio signalo praradimo priklausomybės nuo atstumo bandymas. Bandymui atlikti buvo naudojama įranga pavaizduota 14 pav.

28 pav. Ryšio tyrimo įranga iš kairės į dešinę a) Nrf24l01 ryšio moduliai b) „arduino“ „nano“ mikrovaldiklis c) „Rhode&Schwarz“ FSH8 spektro analizatorius

3.4.1. Bandymo eiga

Ryšio modulis NRF24101 (14 a pav.) prijungtas prie mikrovaldiklis (14 b pav.) ir prijungtas prie orlaivio. Mikrovaldiklis užprogramuotas siųsti pastovų signalą 2,524 GHz dažniu. Mikrovaldiklis ir ryšio modulis prijungti prie orlaivio. Naudojant spektro analizatorių išmatuotas signalo stiprumas už 1 metro. Toliau matavimai atlikti vis didinant atstumą. Bandymo schema pavaizduota 15 pav. parodo kaip buvo matuojamas atstumas. Visų matavimų metu orlaivis su ryšio įranga buvo iškeltas į 50 m aukštį. Atstumas A didinamas 100 metrų žingsniais. Pagal tai panaudojant Pitagoro teoremą paskaičiuoti visų matavimų atstumai B. Naudojant GPS įrenginį visais atstumais B išmatuotas signalo lygis taške C.



29 pav. Bandymo schema

3.4.2. Bandymo rezultatai

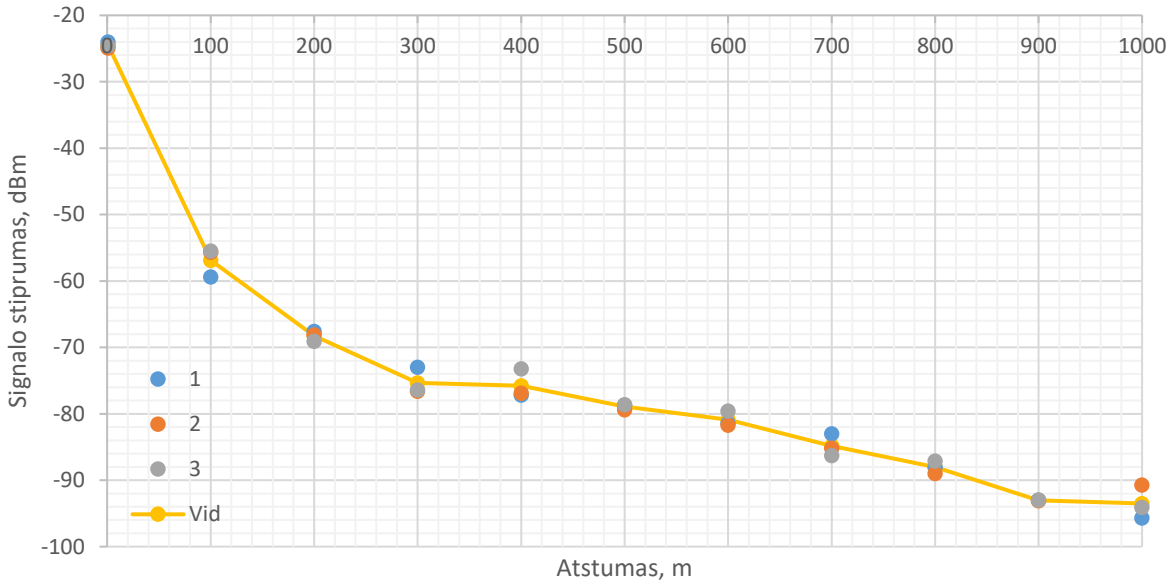
Lentelėje 4 išvardinti bandymai 100-o metrų žingsniais iki 1 km atstumo. Kiekvienam taške pamatuotas signalo lygis 3 kartus.

4 lentelė. Radijo ryšio signalo slopimo eksperimento duomenys

matavimas	realus atstumas, m	orlaivio aukštis, m	paviršiaus atstumas, m	1, dBm	2, dBm	3, dBm	Vidurkis, dBm
0	1	0	1,00	-24,0	-24,9	-24,6	-24,5
1	100	50	86,60	-59,4	-55,7	-55,5	-56,9
2	200	50	193,65	-67,6	-68,1	-69,1	-68,3
3	300	50	295,80	-73,0	-76,6	-76,4	-75,3
4	400	50	396,86	-77,2	-76,9	-73,2	-75,8
5	500	50	497,49	-78,7	-79,4	-78,6	-78,9
6	600	50	597,91	-81,4	-81,7	-79,6	-80,9
7	700	50	698,21	-83,0	-85,2	-86,3	-84,8
8	800	50	798,44	-88,0	-89	-87,1	-88,0
9	900	50	898,61	-93,0	-93,1	-93,0	-93,0

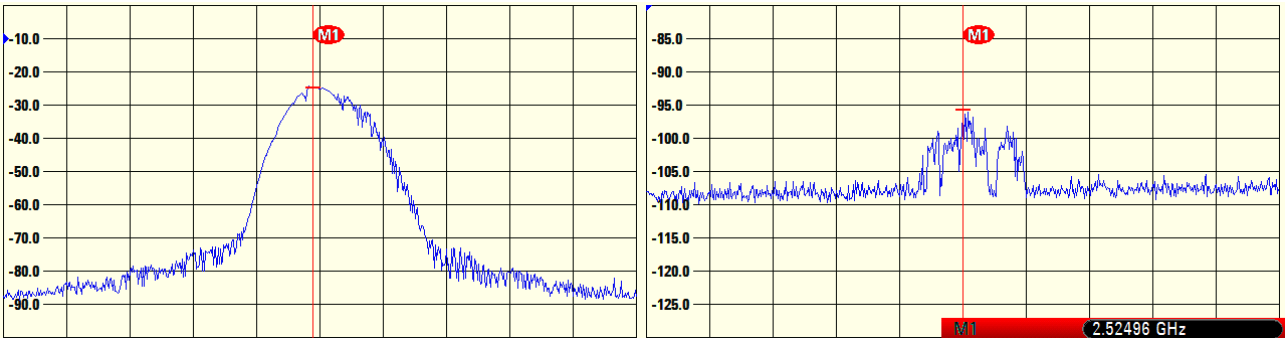
10	1000	50	998,75	-95,7	-90,7	-94,1	-93,5

Žemiau patektame 16 pav. grafiškai pavaizduoti bandymo rezultato duomenys. Iš grafiko matosi kad pirmus 300 metrų signalo lygis sparčiai krenta, tačiau likusius 700 metrų signalo lygis slopsta gerokai mažiau.



30 pav. Ryšio signalo slopimo nuo atstumo grafikas

Taip pat atliekant šį bandymą su spektro analizatoriumi buvo užfiksuotas siųstuvo siunčiamo signalo spektro vaizdas. Paveikslėlyje 19 pavaizduotas signalo spektras už vieno metro nuo siųstuvo ir už vieno kilometro.



31 pav. Siųstuvo siunčiamo signalo radijo spektras už metro nuo siųstuvo (kairėje) ir už kilometro nuo siųstuvo (dešinėje)

4. ALGORITMAI

Pagrindinis pateikto algoritmo tikslas yra palaikyti patikimą ryšį tarp orlaivio ir operatoriaus ir tuo pačiu išlaikyti retransliatorių ore kuo ilgiau. Skyriuje 3 parenkant retransliatoriaus parametrus buvo išsiaiškinta, jog ilgiausiai ore multikopteris išbūna nekeisdamas pozicijos. Algoritmas į tai atsižvelgia ir keičia orlaivio poziciją tik tada, kai ryšio kokybė nukrenta žemiau numatytos reikšmės. Ryšio kokybės parametrą parenkamas atsižvelgiant į atliktą eksperimentą su radijo ryšio sklidimu bei atsižvelgiant į naudojamo radijo imtuvo ryšio signalo slenkstį.

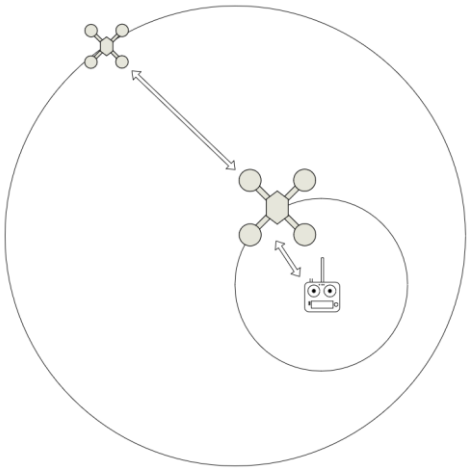
4.1. Pagrindinis algoritmas

21 paveikslėlyje matosi visas pagrindinis algoritmas. Jis susideda iš dalių, kurios pavaizduotos 22, 24 ir 25 paveikslėliuose. Pagrindiniai algoritmo etapai yra 4:

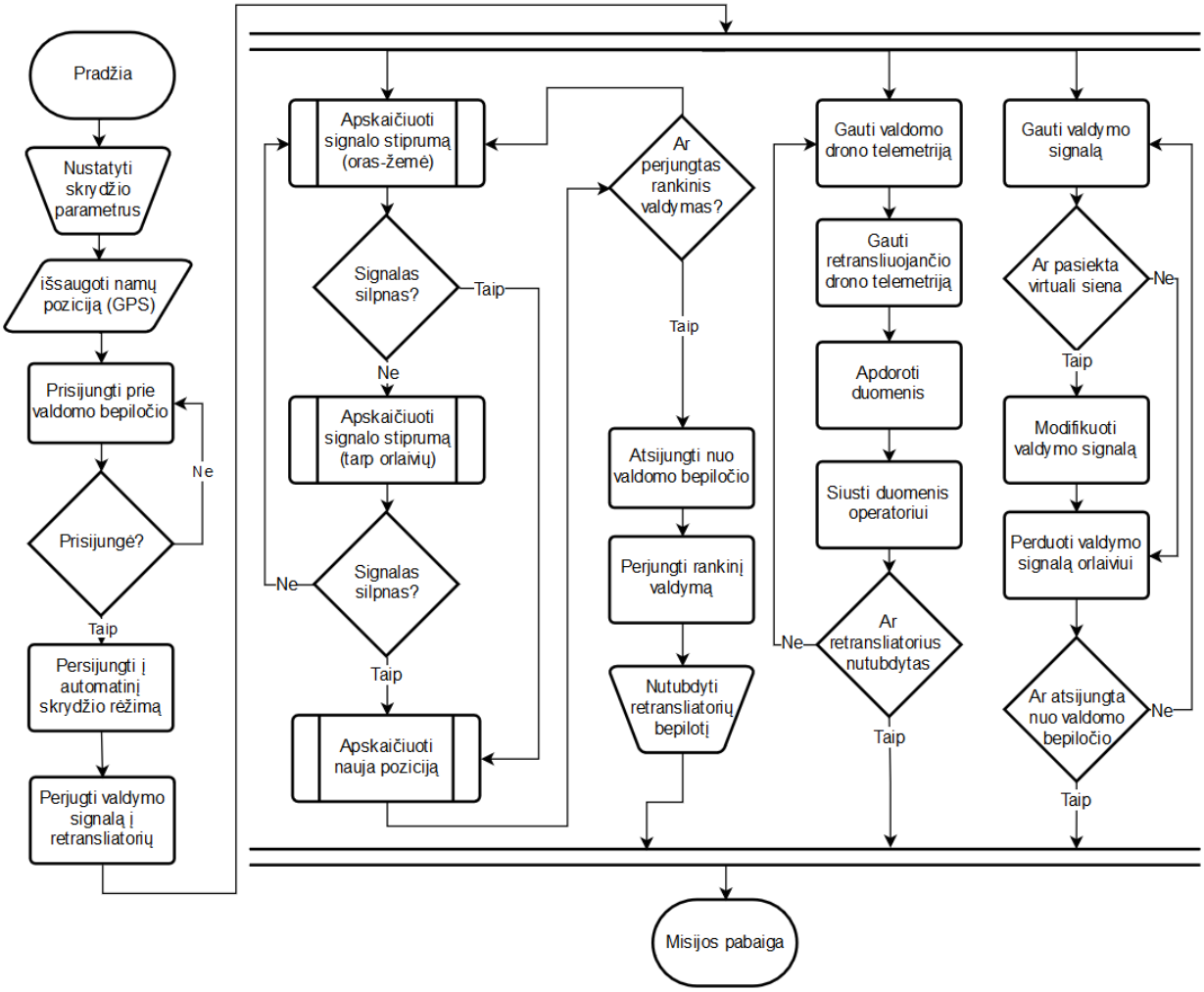
1. Nustatomi pagrindiniai parametrai ir prijungiamas valdomas bepilotis orlaivis;
2. Autonominis retransliatoriaus pozicijos keitimas;
3. Telemetrijos apdorojimas ir pateikimas;
4. Valdymo signalo transformacijos ir retransliacija.

Žingsniai 2 – 4 vykdomi lygiagrečiai po to, kai įvykdomas 1 žingsnis. Misijos pabaiga yra tada, kai atjungiamas valdomas bepilotis orlaivis ir nutupdomas retransliuojantis orlaivis.

Žingsnyje 4 valdymo signalo modifikavimas remiasi virtualios sienos principu (20 pav.). Kai valdomasis orlaivis skrenda nuo retransliatoriaus į zoną, kurioje ryšio kokybė nebūtų užtikrinta, retransliatorius modifikuoja signalą taip, jog valdomas orlaivis negailėtu nuskristi už tos ribos. Kitaip sakant neleidžia operatoriuj nuskraidinti orlaivio į ne ryšio zoną.



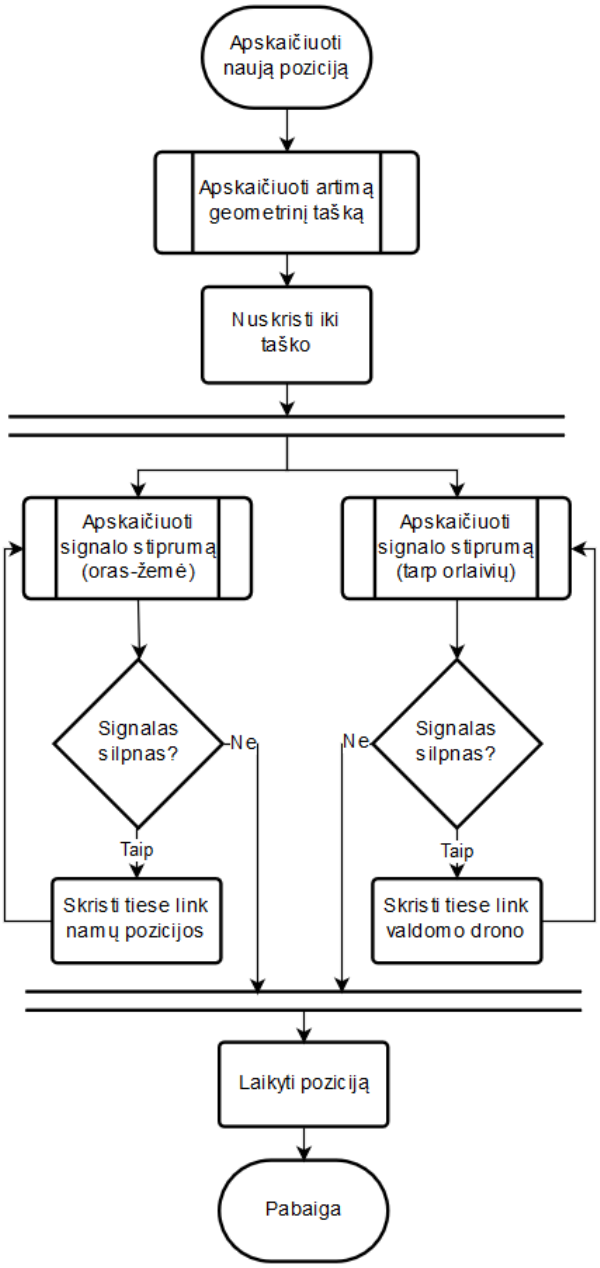
32 pav. Virtualios sienos pavyzdys



33 pav. Pagrindinis bepiločio retransliatoriaus veikimo algoritmas

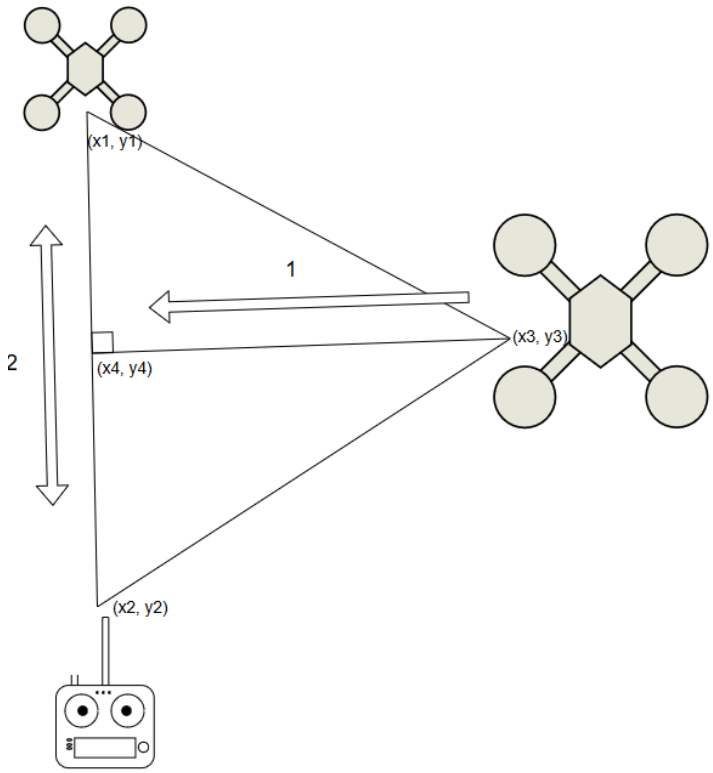
4.2. Autonominis pozicijos parinkimo algoritmas

Autonominiam retransliatoriaus skrydžiui užtikrinti naudojami algoritmai pavaizduoti 22 ir 24 pav. Iš 22 paveikslėlyje pavaizduotos algoritmo dalies matosi jog pozicijos keitimas vyksta dviem etapais. Pirmas etapas yra geometrinis atstumo sumažinimas (24 pav.), antras etapas – atstumo keitimas panaudojant ryšio kokybės matavimus. Antram etape ryšio kokybė tarp abiejų orlaivių ir tarp retransliatoriaus ir operatoriaus skaitoma kaip vienodos svarbos.



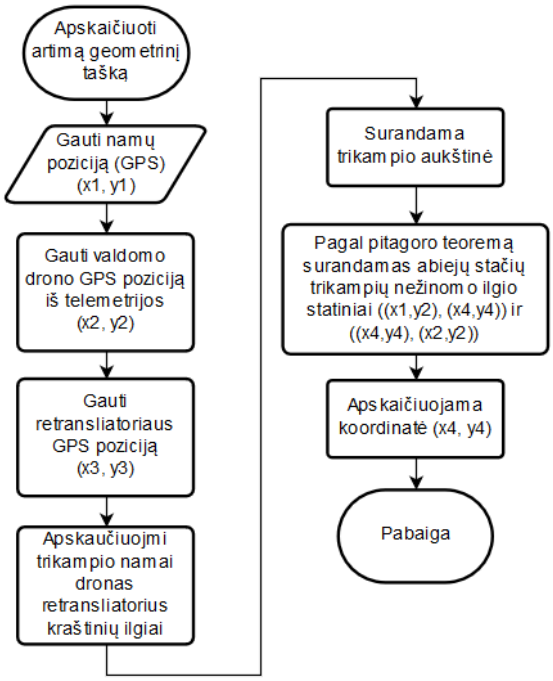
34 pav. Autonominio skrydžio algoritmas

Pirmas žingsnis detaliau pavaizduotas 24 pav. Kadangi atstumai tarp atskirų sistemos taškų nesieks daugiau kilometro, atstumams tarp koordinačių matuoti bus naudojama paprasti trigonometriniai skaičiavimai. Pozicijos nustatymui naudojamos GPS koordinatėmis iš visų trijų sistemos sudedamųjų (operatoriaus, retransliatoriaus, valdomo bepiločio orlaivio). Operatoriaus pozicija nesikeičia ir yra nustatoma operacijos pradžioje naudojant retransliatoriaus GPS imtuvą. 23 paveikslėlyje pavaizduota supaprastinta pozicijos keitimo algoritmo schema.



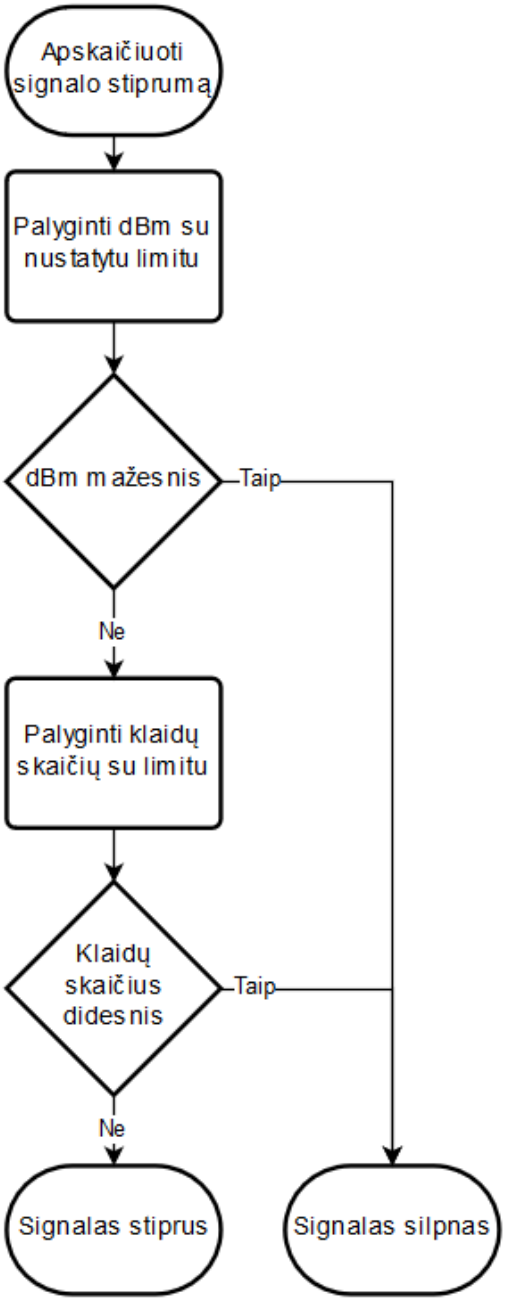
35 pav. Pozicijos keitimo algoritmo žingsniai

Pirmas žingsnis pavaizduotas 23 pav. yra pakeisti retransliatoriaus poziciją naudojant paprastą geometriją. Orlaivis, nekreipdamas dėmesio į aukštį, susiranda tašką tarp operatoriaus ir valdomo bepiločio orlaivio, kuris yra arčiausiai jo pozicijos. Pasiekęs tą tašką, retransliatorius pradeda keisti savo poziciją pagal ryšio kokybę tarp operatoriaus ir valdomo orlaivio. Pasiekus nustatytą „gero“ signalo kokybę orlaivis sustoja vietoj ir laukia kito ryšio sutrikimo.



36 pav. Autonominio skrydžio algoritmo geometrinio pozicionavimo dalis

Paveikslėlyje 25 pavaizduota ryšio kokybės nustatymo algoritmo schema. Iš schemos matosi, kad algoritmas įvertina ryšio kokybe naudodamasis dviem parametrais, tai yra ryšio signalo stiprumas (dBm) ir ryšio signalo klaidų kiekis per laiko vienetą. Jei nors vienas iš parametrų priartėja prie nustatytos ribos signalas skaitomas silpnas kitu atveju stiprus. Riba nustatyta atlikus eksperimentą.



37 pav. Signalo kokybės nustatymo algoritmas

4.3. Pseudokodas

Pagal aukščiau pateiktas diagramas parašytas pseudokodas kiekvienai algoritmo daliai.

4.3.1. Pagrindinis algoritmas

Pagrindinis algoritmas turi du ciklus. Pagrindinis ciklas įsijungia tik tada, kai yra perjungtas retransliavimo rėžimas. Pagrindiniame cikle atliekamas pozicijos keitimas, valdymo signalo retransliavimas bei telemetrijos apdorojimas.

```
BEGIN
    // Įvedami skrydžio parametrai
    // Išsaugoma starto pozicija
    SR = standartinis // skrydžio rėžimas

    WHILE SR != retransliavimas DO
        Skristi(SR); //Skristi standartiniu rėžimu (standartinis – pilnai valdomas operatoriaus)
        IF gaytiNaująRėzimą() == retransliavimas THEN
            SR = retransliavimas;
        ENDIF
    ENDWHILE

    Skristi(SR); // Perjungiamas autonominis skrydžio rėžimas
    Retransliuoti(); // Perjungiamas valdymo retransliavimo rėžimas
    WHILE misijaBaigta() != Taip DO
        TO = gautiTelemetriją(orlaivis); // Gauti telemetriją iš orlaivio
        TR = gautiTelemetriją(retransliatorius); // Gauti telemetriją iš retransliatoriaus
        OrlaivioPozicija = TO.pozicija; //Atnaujinti orlaivio pozicijos duomenis
        siustiOperatoriuj(TO); // Perduoti orlaivio telemetriją operatoriui
        siustiOperatoriuj(TR); // Perduoti retransliatoriaus telemetriją operatoriui

        SKN = signaloKokybė(namai); // Apskaičiuoti signalo kokybę
                                    // tarp retransliatoriaus ir namų pozicijos
        SKO = signaloKokybė(orlaivis); // Apskaičiuoti signalo kokybę
                                       // tarp retransliatoriaus ir valdomo orlaivio
        IF SKN == prasta OR SKO == prasta THEN
            keistiPoziciją();
        ENDIF

        Signalas = gautiValdymoSignalą(); // Gauti retransliuojamą signalą
        IF OrlaivioPozicija – RetransliatoriausPozicija > leistinasAtsumas THEN
            modifikuotiSignalą(Signalas); // Modifikuoti valdymo signalą
        ENDIF
        SiustiSignalą(Signalas); // Perduoti valdymo signalą orlaiviui

        IF gaytiNaująRėzimą() == standartinis THEN
            stabdytiRetransiavimą(); // Sustabdyti valdymo signalo retransliavimą
            baigtiMisiją();
            SR = standartinis; //Perjungti retransliatorių į standartinį rėžimą
        ENDIF
    ENDWHILE
END
```

4.3.2. Pozicijos apskaičiavimas/keitimas

Pozicijos apskaičiavimas vyksta cikle, kuriame yra dvi sąlygos, kurios tikrina signalo kokybę. Ciklas sukasi tol, kol nėra užtikrinta signalo kokybė tarp operatoriaus ir valdomo orlaivio.

```
BEGIN
    PN; // namų pozicija
    ON; // orlaivio pozicija
    P = gautiGeometrinęPoziciją(); // Apskaičiuoti nauja pozicija
    Skristi(P); // Skristi iki pozicijos

    WHILE Skrenda() DO
        Laukti();
    ENDWHILE
```

```
10.
11.     SKN = signaloKokybė(namai); // Apskaičiuoti signalo kokybę
12.                                 // tarp retransliatoriaus ir namų pozicijos
13.     SKO = signaloKokybė(orlaivis); // Apskaičiuoti signalo kokybę
14.                                    // tarp retransliatoriaus ir valdomo orlaivio
15.     WHILE SKN = prasta AND SKO == prasta DO
16.         SKN = signaloKokybė(namai); // Apskaičiuoti signalo kokybę
17.                                     // tarp retransliatoriaus ir namų pozicijos
18.         SKO = signaloKokybė(orlaivis); // Apskaičiuoti signalo kokybę
19.                                        // tarp retransliatoriaus ir valdomo orlaivio
20.
21.         IF SKN == prasta AND SKO == gera THEN
22.             Skristi(PN)
23.         ELSEIF SKO == prasta AND SKN == gera THEN
24.             Skristi(PO)
25.         ELSE
26.             įspėtiOperatorių(); // jeigu kokybė prasta abiejuose galuose
27.                                 // niekur nejudėti toliau ir pranešti operatoriuj
28.                                 // jog reikia sumažinti atstumą;
29.             BREAK
30.         ENDIF
31.     ENDWHILE
32.
33.     Stovėti(); // Laikyti poziciją
34. END
```

4.3.3. Geometrinio taško radimas

Taško radimo algoritmas remiasi paprasta geografinių koordinačių (gautų iš GPS) matematika bei trigonometrija.

```
1.  BEGIN
2.      NP; // išsaugota namų pozicija
3.      RP; // Retransliatoriaus GPS pozicija
4.      OP; // Orlaivio pozicija
5.      R = 6371000 // Žemės spindulys metrais
6.
7.      // apskaičiuojami atstumai metrais tarp taškų
8.      NP_RP = sqrt(sqr((RP.ilguma - NP.ilguma) * cos((RP.platuma + NP.platuma) / 2)) + sqr(RP.platuma
    – NP.platuma)) * R;
9.
10.     RP_OP = sqrt(sqr((OP.ilguma - RP.ilguma) * cos((OP.platuma + RP.platuma) / 2)) + sqr(OP.platuma
    – RP.platuma)) * R;
11.
12.     NP_OP = sqrt(sqr((OP.ilguma - NP.ilguma) * cos((OP.platuma + NP.platuma) / 2)) + sqr(OP.platuma
    – NP.platuma)) * R;
13.
14.     P = (NP+RP + RP_OP + NP_OP) / 2; // trikampio pusperimetris
15.
16.     // atstumas nuo retransliatoriaus iki NP_OP naudojant Herono formulę
17.     H = sqr(P * (p – NP_RP) * (P – RP_OP) * (P – NP_OP)) * 2 / NP_OP;
18.
19.     Kryptis = ATAN2(SIN(OP.ilguma-NP.ilguma)*COS(OP.ilguma), COS(NP.platuma)*SIN(OP.platuma)-
    SIN(NP.platuma)*COS(NP.platuma)*COS(NP.platuma-OP.platuma));
20.
21.     NP_H = sqrt(sqr(NP_RP) – sqr(H));
22.
23.     Platuma = ASIN(SIN(NP.platuma)*COS(NP_H/R) + COS(NP.platuma)*SIN(NP_H/R)*COS(Kryptis));
24.     Ilguma = NP.ilguma + ATAN2(SIN(Kryptis)*SIN(NP_H/R)*COS(NP.platuma),COS(NP_H/R)-
    SIN(NP.platuma)*SIN(Platuma));
25.
26.     RETURN Ilguma, Platuma;
27. END
```

4.3.4. Signalo kokybės apskaičiavimas

Signalo kokybės algoritmas turi dvi sąlygas kurios tikrina signalo stipruma bei klaidų kiekį.

```
BEGIN
    dBm; // signalo stiprumas tarp taškų decibelais
    dBm_min; // minimalus signalo lygis decibelais

    Klaidos; // klaidų kiekis
    klaidos_max; // maksimalus klaidų kiekis

    Kokybė = gera;

    IF dBm < dBm_min THEN
        Kokybė = prasta;
    ENDIF
    IF klaidos > klaidos_max THEN
        Kokybė = prasta;
    ENDIF

    RETURN Kokybė;
END
```

4.3.5. Valdymo signalo modifikavimas

Valdymo signalas keičiamas cikle kuris sukasi tol, kol signalo retransliacija nėra išjungiama. Retransliuojant signalą, taip pat tikrinami orlaivio judėjimo parametrai bei operatoriaus norima valdymo kryptis. Algoritmas užtikrina, kad operatorius negalėtų nuskraidinti orlaivio už ryšio zonos ribų.

```
BEGIN
    R; // geros kokybės zonos spindulys
    SR; // skrydžio rėžimas

    WHILE SR == retransliavimas DO
        Orientacija = valdomo orlaivio orientacija
        Gauti valdymo signalo vektorių
        Gauti orlaivio judėjimo vektorių
        Sudėti valdymo ir judėjimo vektorius
        Apskaičiuoti sekančią orlaivio poziciją vektoriaus kryptimi
        IF sekanti drono pozicija už R spindulio zonos ribos THEN
            Sustabdyti orlaivy vietoje
            Neleisti operatoriaus valdymo praeito judėjimo vektoriaus kryptimi
        ELSE
            Perduoti signalą orlaiviui
        ENDIF
    EDNWHILE
END
```

4.4. Algoritmo realizacija

Kadangi retransliatoriaus realizacijai panaudoti „Arduino“ valdikliai, algoritmas realizuotas C++ programavimo kalba. Retransliatorius suprojektuotas panaudojant keturis valdiklius – skrydžio kompiuteris (valdiklis ATmega2560), signalo apdorojimo valdiklis („Arduino Nano“ valdiklis ATmega328), telemetrijos apdorojimo valdiklis („Arduino Nano“ valdiklis ATmega328) ir valdiklis, skirtas valdomam orlaiviui („Arduino UNO“ valdiklis ATmega328p). Visiems keturiems valdikliams

kodas parašytas atskirai. Taip pat sukurtas abstraktus simuliatorius, atstojantis valdomą orlaivį, su kuriuo buvo ištestuotas algoritmo veikimas. Taip pat testavimui panaudotas „ArduCopter“ simuliatorius. Jis naudojamas simuliuoti retransliatoriaus skrydžio veikimą.

Retransliatoriaus skrydžio kompiuteris naudoja atviro kodo programinę įrangą. Norint pakeisti jo veikimą, tereikėjo pridėti papildomą skrydžio rėžimą, kurį įjungus skrydžio metu pradedamas vykdyti retransliatoriaus skrydžio pagal ryšio kokybę algoritmas.

Programos kodas pateiktas 1 priede.

4.4.1. Telemetrijos valdiklio kodas

Telemetrija iš skrydžio kompiuterio perduodama „MavLink“ protokolu, o siųstuvas telemetriją priima „smart-port“ protokolu. Telemetrijos konvertavimas atliekamas panaudojant „Arduino Nano“ valdiklį. „MavLink“ telemetrijai nuskaityti panaudota atviro kodo biblioteka. Telemetrijos valdiklis taip pat atsakingas už valdomo orlaivio telemetrijos perdavimą retransliatoriui bei operatoriuj. Šią informaciją jis perima iš valdymo signalo apdorojimo valdiklio, kuris prijungtas prie retransliatorius – valdomas orlaivis ryšio siųstuvo/imtuvo.

Programos kodas pateikiamas 2 priede.

4.4.2. Valdymo signalo apdorojimo ir siuntimo valdiklio kodas

Valdymo signalas iš imtuvo perduodamas „S-Bus“ protokolu. Tai protokolas susidedantis iš 25 baitų paketų siunčiamų kas 14 ms. Protokolo greitis yra 100000 bitų per sekundę. Paketas susideda iš tokių komponentų:

- Pradžios baitas (8 bitai);
- 22 duomenų baitai (vienam kanalui po 11bitų iš viso 16 kanalų);
- Informacini baitas (8 bitai kiekvienas iš jų turi atskirą paskirtį);
- Pabaigos baitas (8bitai).

Kadangi kanalų informacija perduodama nestandartiniu 11 bitų dydžiu atsiranda sunkumų nuskaityti informaciją. Vieno kanalo informacija pasiskirsto per du ar tris duomenų baitus. Signalas taip pat yra apverstas. Tai reiškia jog aukšto lygio signalas yra loginis 0, o žemo lygio signalas reiškia loginį 1. Norint nuskaityti tokį signalą panaudotas TTL signalo keitiklis.

Programos kodas pateikiamas 3 priede.

4.4.3. Valdomo orlaivio valdiklio kodas

Valdomo orlaivio valdiklis atlieka kelias funkcijas. Jis priima valdymo signalą iš reransliatoriaus ir perduoda jį orlaiviui (simuliacijai). Taip pat valdiklis atsakingas už telemetrijos nuskaitymą iš orlaivio (simuliacijos) ir jos išsiuntimą retransliatoriuj.

Valdymo signalas iš retransliatoriaus ateina 16 baitų formatu. Vieno kanalo informacija saugoma dviejuose baituose. Nors ir kanalo informacija niekada neišnaudoja visų 16 bitų, toks formatas buvo pasirinktas dėl paprastesnio ir greičiau vykdomo kodo, kadangi informaciją siunčiamą 11 bitų per kanalą formatu tektų dekoduoti, kas jau buvo įvykdyta retransliatoriuje.

Telemetrija iš orlaivio gaunama kas 100 ms. Tokiu pačiu greičiu ji perduodama retransliatoriuj. Telemetrijos paketas susideda iš 11 baitų:

- 4 baitai geografinei platumai;
- 4 baitai geografinei ilgumai;
- 2 baitai krypčiai;
- 1 baitas barometriniam aukščiui.

Programos kodas pateikiamas 4 priede.

IŠVADOS

1. Literatūros analizė parodė, kad Žemė-BO-BO retransliacijos sistemos yra siūlomos tik karinės paskirties projektuose. Taip pat nerasta detalių algoritmų ir pseudo kodų, pritaikytų BO autonominiam valdymui pagal ryšio kokybę;
2. Pasiūlyti skrydžio ir signalo retransliavimo algoritmai. Skrydžio algoritmas atsižvelgia į skrydžio trukmę bei ryšio kokybės užtikrinimą. Retransliavimo algoritmas pagrįstas virtualios sienos funkcija, kuri užtikrina radijo ryšio nepraradimą;
3. Suprojektuota tokia retransliatoriaus sistema, kuri gali retransliuoti ryšio signalą iki 1 km nuo retransliatoriaus. Sistemos darbo laikas yra iki 35 minučių;
4. Sukurti algoritmo pseudo kodai. Jie realizuoti C++ programavimo kalba „Arduino“ mikrovaldiklių platformoje. Kodas parašytas keturioms tarpusavyje komunikuojančioms sistemos dalims;
5. Realizuojant retransliatoriaus sistemą, pagaminta ir išbandyta sistema, kuri atitinka arba viršija suprojektuotos sistemos parametrus. Retransliatoriaus skrydžio laikas 65 s ilgesnis už apskaičiuotą programos „eCalc“ pagalba.
6. Išmatuota, kad retransliatoriaus signalo lygis 1 km atstumu nuo siųstuvo yra -95,7 dBm. Tai užtikrina kokybinį ryšį, kadangi imtuvo slenkstis yra -105 dBm.

LITERATŪROS SĄRAŠAS

[1] K. Nonami, „Prospect and recent research & development for civil use autonomous unmanned aircraft as UAV and MAV,“ 2007.

[2] R. Rangel, K. Kienitz, C. de Oliveira ir M. Brandao, „Development of a Multipurpose Tactical Surveillance System Using UAV'S,“ *Proceedings of the 2014 IEEE Aerospace Conference,* 2014.

[3] B. Li, Y. Jiang, J. Sun, L. Cai ir C.-Y. Wen, „Development and Testing of a Two-UAV Communication Relay System,“ 2016.

[4] B. Merkys, Orlaivių Konstrukcijos, Vilnius, 2012.

[5] W. J. Wagtendonk, Principles of helicopter flight second edition, 2006.

[6] O. A. Somefun, „Simulation and Development of a Tricopter for Parcel Delivery,“ 2015.

[7] L. E. Romero, D. F. Pozo ir J. A. Rosales, „Quadcopter stabilization by using PID controllers“.

[8] L. E. Romero, D. F. Pozo ir J. A. Rosales, „Quadcopter stabilization by using PID controllers“.

[9] D. Bershadsky ir S. Haviland, „Electric Multirotor Propulsion System Sizing for Performance Prediction and Design Optimization“.

[10] D. Bershadsky ir S. Haviland, „Electric Multirotor Propulsion System Sizing for Performance Prediction and Design Optimization“.

[11] J. Tarascon, „Issues and challenges facing rechargeable lithium batteries,“ *Nature Publishing Group,* 2001.

[12] S. Yoon, H. C. Lee ir T. H. Pulliam, „Computational Analysis of Multi-Rotor Flow“.

[13] O. J. Ohanian, „Ducted Fan Aerodynamics and Modeling,“ 2011.

[14] Nic, „Quadcopter Software Design,“ 17 5 2011. [Tinkle]. Available: https://nicisdigital.wordpress.com/2011/05/17/software-design/.

[15]

[16] B. Li, Y. Jiang, J. Sun, L. Cai ir C.-Y. Wen, „Development and Testing of a Two-UAV Communication Relay System,“ 2016.

[17] F. J. Pinkey, D. Hampel ir S. DiPiero, „Unmanned aerial vehicle (UAV) communications relay,“ 2002.

[18] P. Muri ir J. McNair, A Survey of Communication Sub-systems for Intersatellite Linked Systems and CubeSat Missions, 2012.

[19] M. Abuelela ir S. Olariu, „Traffic-adaptive packet relaying in VANET,“ įtraukta *VANET '07 Proceedings of the fourth ACM international workshop on Vehicular ad hoc networks*, Quebec, Canada, 2007.

[20] B. Sutton ir M. Pinfold, „The concept of a Linear Transponder,“ [Tinkle]. Available: http://www.amalgamate2000.com/radio-hobbies/radio/dunedin%20linear.htm.

[21] M. Gatti, „Design and Prototyping High Endurance Multi-Rotor“.

PRIEDAI

1 PRIEDAS. AUTONOMINIO SKRYDŽIO KODAS

```
1.  static bool guided_pilot_yaw_override_yaw = false;
2.  static Vecto3 uav_position;
3.  static bool signal_init(bool ignore_checks) {
4.      if (GPS_ok() || ignore_checks) { // initialise yaw
5.          set_auto_yaw_mode(get_default_auto_yaw_mode(false));
6.          pos_control_start(); // start in position hold mode
7.          poshold_init(true);
8.          mode = HOLD;
9.          return true;
10.     } else {
11.         return false;
12.     }
13. }
14. void pos_control_start() {
15.     mode = TRAVEL;
16.     wp_nav.wp_and_spline_init();
17.     Vector3f stopping_point;
18.     stopping_point.z = inertial_nav.get_altitude();
19.     wp_nav.get_wp_stopping_point_xy(stopping_point);
20.     wp_nav.set_wp_destination(stopping_point);
21.     guided_pilot_yaw_override_yaw = false;
22.     set_auto_yaw_mode(get_default_auto_yaw_mode(false));
23. }
24. static void set_destination(const Vector3f & destination) {
25.     if (mode != TRAVEL) {
26.         guided_pos_control_start();
27.     }
28.     wp_nav.set_wp_destination(destination);
29. }
30. static void signal_run() {
31.     if (!ap.auto_armed) {
32.         attitude_control.relax_bf_rate_controller();
33.         attitude_control.set_yaw_target_to_current_heading();
34.         attitude_control.set_throttle_out(0, false);
35.         return;
36.     }
37.     switch (mode) {
38.         case HOLD:
39.             hold_position_run();
40.             break;
41.         case TRAVEL_SIGNAL:
42.             travel_signal_run();
43.             break;
44.         case TRAVEL:
45.             pos_control_run();
46.             break;
47.     }
48. }
49. static travel_signal_start() {
50.     mode = TRAVEL_SUIGNAL;
51.     vel_control_start();
52. }
53. static travel_signal_run() {
54.     Vector3 heading = inertial_nav.get_heading();
55.     Vector3 desired_heading = get_target_heading();
56.     if (heading > desired_heading || heading < desired_heading) {
57.         inertial_nav.set_heading(desired_heading);
58.     } else {
59.         bool signals[] = bool[2];
60.         check_signal_strength( & signals);
61.         if (!signals[0] && signals[1]) {
62.             set_velocity();
63.         } else if (signals[0] && !signals[1]) {
```

```
64.             set_velocity();
65.         } else if (signals[0] && signals[1]) {
66.             mode = HIOLD;
67.         } else {
68.             mode = HIOLD;
69.             gcs.send_aert_msg("SIGNAL_TOO_LOW");
70.         }
71.     }
72. }
73. static hold_position_run() {
74.     wp_nav.init_loiter_target();
75.     attitude_control.relax_bf_rate_controller();
76.     attitude_control.set_yaw_target_to_current_heading();
77.     poshold_run();
78.     get_positions();
79.     bool signals[] = bool[2];
80.     check_signal_strength( & signals);
81.     if (!signals[0] || !signals[1]) {
82.         guided_set_destination(calculate_point());
83.     }
84. }
85. static void pos_control_run() {
86.     float target_yaw_rate = 0;
87.     if (!failsafe.radio) {
88.         target_yaw_rate = get_pilot_desired_yaw_rate(g.rc_4.control_in);
89.         if (target_yaw_rate != 0) {
90.             set_auto_yaw_mode(AUTO_YAW_HOLD);
91.         }
92.     }
93.     wp_nav.update_wpnav();
94.     pos_control.update_z_controller();
95.     if (auto_yaw_mode == AUTO_YAW_HOLD) {
96.         attitude_control.angle_ef_roll_pitch_rate_ef_yaw(wp_nav.get_roll(), wp_nav.get_pitch(), targ
    et_yaw_rate);
97.     } else {
98.         attitude_control.angle_ef_roll_pitch_yaw(wp_nav.get_roll(), wp_nav.get_pitch(), get_auto_hea
    ding(), true);
99.     }
100.        }
101.        static void get_positions() {
102.            String lat;
103.            String lon;
104.            String alt;
105.            gcs.get_key_value("C_LAT", & lat);
106.            gcs.get_key_value("C_LON", & lon);
107.            gcs.get_key_value("C_ALT", & alt);
108.            uav_position.z = alt.toFloat();
109.            uav_position.x = lat.toFloat();
110.            uav_position.y = lon.toFloat();
111.        }
112.        static Vector3 calculate_point() {
113.            Vector3 point = new Vector3();
114.            Vector3 current_position = inertial_nav.get_position() Vector3 home_position = gps.get_b
    ase_position();
115.            int r = 6371000 // earth radius
116.                float base_current = sqrt(sqr((current_position.x - home_position.x) * cos((current_
    position.y + home_position.y) / 2)) + sqr(current_position.y- home_position.y)) * r;
117.            float current_uav = sqrt(sqr((uav_position.x - current_position.x) * cos((uav_position.y
     + current_position.y) / 2)) + sqr(uav_position.y- current_position.y)) * r;
118.            float base_uav = sqrt(sqr((uav_position.x - home_position.x) * cos((uav_position.y + hom
    e_position.y) / 2)) + sqr(uav_position.y- home_position.y)) * r;
119.            float p = (base_current + current_uav + base_uav) / 2;
120.            float h = sqr(p * (p- base_current) * (p- current_uav) * (p- base_uav)) * 2 / base_uav;

121.            float heading = atan2(sin(uav_position.x - home_position.x) * cos(uav_position.x), cos(h
    ome_position.y) * sin(uav_position.y) - sin(home_position.y) * cos(home_position.y) * cos(home_posit
    ion.y - uav_position.y));
122.            float home_h = sqrt(sqr(base_current)- sqr(h));
```

```
123.            point.y = asin(sin(home_position.y) * cos(home_h / r) + cos(home_position.y) * sin(home_
    h / r) * cos(heading));
124.            point.x = home_position.x + atan2(sin(heading) * sin(home_h / r) * cos(home_position.y),
     cos(home_h / r) - sin(home_position.y) * sin(point.y));
125.            point.z = inertial_nav.get_altitude();
126.            return point;
127.        }
128.        static void check_signal_strength(bool signals[]) {
129.            String signal;
130.            gcs.get_key_value("C_RSSI", & signal);
131.            int signal_level = signal.toInt();
132.            int home_signal = channels.get_rssi();
133.            if (signal_level < -97) {
134.                signals[0] = false;
135.            } else {
136.                signals[0] = true;
137.            }
138.            if (home_signal < -97) {
139.                signals[1] = false;
140.            } else {
141.                signals[1] = true;
142.            }
143.        }
144.        void vel_control_start() {
145.            guided_mode = Guided_Velocity;
146.            pos_control.set_speed_z(-g.pilot_velocity_z_max, g.pilot_velocity_z_max);
147.            pos_control.set_accel_z(g.pilot_accel_z);
148.            pos_control.init_vel_controller_xyz();
149.        }
150.        static void set_velocity(const Vector3f & velocity) {
151.            if (guided_mode != Guided_Velocity) {
152.                vel_control_start();
153.            }
154.            pos_control.set_desired_velocity(velocity);
155.        }
156.        static void vel_control_run() {
157.            float target_yaw_rate = 0;
158.            if (!failsafe.radio) {
159.                target_yaw_rate = get_pilot_desired_yaw_rate(g.rc_4.control_in);
160.                if (target_yaw_rate != 0) {
161.                    set_auto_yaw_mode(AUTO_YAW_HOLD);
162.                }
163.            }
164.            pos_control.update_vel_controller_xyz();
165.            if (auto_yaw_mode == AUTO_YAW_HOLD) {
166.                attitude_control.angle_ef_roll_pitch_rate_ef_yaw(pos_control.get_roll(), pos_control
    .get_pitch(), target_yaw_rate);
167.            } else {
168.                attitude_control.angle_ef_roll_pitch_yaw(pos_control.get_roll(), pos_control.get_pit
    ch(), get_auto_heading(), true);
169.            }
170.        }
```

2 PRIEDAS. TELEMETRIJOS VALDIKLIO KODAS

```
1.  #include "Arduino.h"
2.  #include "FrSkyProcessor.h"
3.  #include "MavlinkProcessor.h"
4.  struct UAVTelemetry {
5.      float latitude;
6.      float longitude;
7.      uint16_t heading;
8.      byte altitude;
9.      Byte powerLevel;
10. }
11. telem;
```

```
String input = "";
uint8_t led_pin = 13;
FrSkyProcessor * frsky_processor;
MavlinkProcessor * mavlink_processor;
Telemetry gathered_telemetry {};
void setup() {
    delay(5000);
    frsky_processor = new FrSkyProcessor(FrSkyProcessor::SOFT_SERIAL_PIN_2, led_pin);
    mavlink_processor = new MavlinkProcessor();
    Serial.begin(115200);
    pinMode(led_pin, OUTPUT);
    analogReference(DEFAULT);
}
void loop() { // Check MavLink communication
    mavlink_processor - > receiveTelemetry(gathered_telemetry);
    if (Serial.available()) {
        byte readValues;
        while (Serial.available()) {
            char inChar = Serial.read();
            if (inChar == ' ') {
                switch (readValues) {
                    case 0:
                        {
                            telem.latitude = input.toFloat();
                            break;
                        }
                    case 1:
                        {
                            telem.longitude = input.toFloat();
                            break;
                        }
                    case 2:
                        {
                            telem.heading = input.toInt();
                            break;
                        }
                    case 3:
                        {
                            telem.altitude = input.toInt();
                            break;
                        }
                    case 4:
                        {
                            telem.powerLevel = input.toInt();
                            break;
                        }
                }
                readValues++;
                input = "";
            } else {
                input += inChar;
            }
        }
        input = "";
    } // send information as custom messages to retranslator
    mavlink_processor - > write("C_LAT", telem.latitude);
    mavlink_processor - > write("C_LON", telem.longitude);
    mavlink_processor - > write("C_HDG", telem.heading);
    mavlink_processor - > write("C_ALT", telem.altitude);
    mavlink_processor - > write("C_RSSI", telem.powerLevel);
    mavlink_processot - > Send(); // append telemetry information for operator
    gathered_telemetry.customInt2 = telem.altitude;
    gathered_telemetry.customFloat1 = telem.latitude;
    gathered_telemetry.customFloat2 = telem.longitude;
    gathered_telemetry.customInt1 = telem.heading;
    frsky_processor - > process(gathered_telemetry, mavlink_processor - > isConnected());
}
```

```
#include "FUTABA_SBUS.h"
#include <SPI.h>
#include "nRF24L01.h"
#include "RF24.h"
#include "printf.h" // Hardware configuration: Set up nRF24L01 radio on SPI bus plus pins 7 & 8
RF24 radio(7, 8);
const uint64_t pipes[2] = {
    0xABCDABCD71 LL, 0x544d52687C LL
}; // Topology
uint16_t channels[8];
FUTABA_SBUS sBus;
struct telemetry {
    float latitude;
    float longitude;
    uint16_t heading;
    byte altitude;
    Byte level;
};
typedef struct telemetry Telemetry;
Telemetry receivedTelemetry;
void setup() {
    sBus.begin();
    Serial.begin(115200);
    printf_begin(); // Setup and configure radio
    radio.begin();
    radio.setAutoAck(1); // Ensure autoACK is enabled
    radio.enableAckPayload(); // Allow optional ack payloads
    radio.enableDynamicPayloads();
    radio.setRetries(15, 15); // Smallest time between retries, max no. of retries
    radio.openWritingPipe(pipes[0]); // Both radios listen on the same pipes by default, and switch
when writing
    radio.openReadingPipe(1, pipes[1]);
    radio.setChannel(0x6f);
    radio.setPALevel(RF24_PA_MAX);
    radio.startListening(); // Start listening
    radio.printDetails(); // Dump the configuration of the rf unit for debugging
}
void loop(void) {
        radio.stopListening();
        sBus.FeedLine();
        if (sBus.toChannels == 1) {
            sBus.UpdateChannels();
            sBus.toChannels = 0;
            for (int i = 9; i <= 12; i++) {
                channels[i - 9] = sBus.Channel(i);
                unsigned long time = micros();
                if (!radio.write( & channels, sizeof(channels))) {
                    printf("failed.\n\r");
                } else {
                    if (!radio.available()) {
                        printf("Blank Payload Received\n\r");
                    } else {
                        while (radio.available()) {
                            unsigned long tim = micros();
                            radio.read( & receivedTelemetry, sizeof(receivedTelemetry));
                        }
                    }
                }
                Serial.print(receivedTelemetry.latitude, 6);
                Serial.print(" ");
                Serial.print(receivedTelemetry.longitude, 6);
                Serial.print(" ");
                Serial.print(receivedTelemetry.heading);
                Serial.print(" ");
                Serial.println(receivedTelemetry.altitude);
                Serial.print(" ");
```

```
                Serial.println(receivedTelemetry.level);
                Serial.print(" ");
            }
        }
```

4 PRIEDAS. VALDOMO ORLAIVIO VALDIKLIO KODAS

```
#include < SPI.h >
#include "nRF24L01.h"
#include "RF24.h"
#include "printf.h" // Hardware configuration: Set up nRF24L01 radio on SPI bus plus pins 9 & 10
RF24 radio(7, 8); // Topology
const uint64_t pipes[2] = {
    0xABCDABCD71 LL, 0x544d52687C LL
};
const byte CHANNEL_COUNT = 8;
uint16_t channels[CHANNEL_COUNT];
int counter = 0;
struct telemetry {
    float latitude;
    float longitude;
    uint16_t heading;
    byte altitude;
    Byte level;
};
typedef struct telemetry Telemetry;
Telemetry telem;
void setup() {
    inputString.reserve(200);
    Serial.begin(115200);
    printf_begin();
    printf("Info: telem size: %d", sizeof(telem));
    delay(2500); // Setup and configure radio
    radio.begin();
    radio.setAutoAck(1); // Ensure autoACK is enabled
    radio.enableAckPayload(); // Allow optional ack payloads
    radio.enableDynamicPayloads();
    radio.setRetries(15, 15); // Smallest time between retries, max no. of retries
    radio.openWritingPipe(pipes[1]); // Both radios listen on the same pipes by default, and switch
when writing
    radio.openReadingPipe(1, pipes[0]);
    radio.setChannel(0x6f);
    radio.setPALevel(RF24_PA_MAX);
    radio.writeAckPayload(1, & telem, sizeof(telem));
    radio.startListening(); // Start listening
    radio.printDetails(); // Dump the configuration of the rf unit for debugging
}
void loop(void) {
    while (radio.available()) { // Read all available payloads
        counter = 0;
        radio.read( & channels, sizeof(channels));
        for (int i = 0; i < sizeof(channels) / 2; i++) {
            if (i > 0) {
                Serial.print("\t");
            }
            Serial.print(channels[i]);
        }
        Serial.println();
        radio.writeAckPayload(1, & telem, sizeof(telem));
    } // LOST CONNECTION
    if (counter > 5000) {
        memset(channels, 0, sizeof(channels));
        for (int i = 0; i < sizeof(channels) / 2; i++) {
            if (i > 0) {
                Serial.print("\t");
            }
```

```
            Serial.print(channels[i]);
        }
        Serial.println();
    } else {
        counter++;
    } // END LOST CONNECTION
    serialStuff();
}
void serialStuff() {
    byte readValues = 0;
    if (Serial.available()) {
        delay(2);
    }
    while (Serial.available()) { // get the new byte:
        char inChar = Serial.read(); // add it to the inputString:
        if (inChar == ' ') { //Serial.println();
            switch (readValues) {
                case 0:
                    {
                        telem.latitude = inputString.toFloat();
                        break;
                    }
                case 1:
                    {
                        telem.longitude = inputString.toFloat();
                        break;
                    }
                case 2:
                    {
                        telem.heading = inputString.toInt();
                        break;
                    }
                case 3:
                    {
                        telem.altitude = inputString.toInt();
                        break;
                    }
                case 4:
                    { // testing signal level from simulator // change to radio.testRDP();
                        telem.level = inputString.toInt();
                        break;
                           }
                   }
                   readValues++;
                   inputString = "";
               } else {
                   inputString += inChar;
               }
           }
           inputString = "";
       }
```