VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
PROGRAMŲ SISTEMŲ KATEDRA

Bakalauro baigiamasis darbas

Autonominis ketursraigčio skrydžio valdymas
Autonomus Control of Quadcopter Flight

Atliko: 4 kurso 1 grupės studentas
Rytis Karpuška .....
(parašas)

Darbo vadovas:
Irus Grinis, lekt. .....
(parašas)

Recenzentas:
Vytautas Valaitis.....
(parašas)

Vilnius
2014

Santrauka

Šio darbo metu buvo sukurti ir išbandyti algoritmai ketursraigčio autonominiam skrydžiui palaikyti, bandymus atliekant su realiu ketursraigčiu. Ši užduotis buvo išskaidyta į tris dalis: kampinės pozicijos įvertinimas, kampinės pozicijos valdymas bei abosliučios pozicijos valdymas. Kampinė pozicija yra skaičiuojama pasitelkiant kvaternionus, šitaip efektyviai atliekant skaičiavimus įterptinės sistemos aplinkoje, bei išvengiant užrakinimo (ang.: *gimbal lock*) problemos. Kampinės pozicijos valdymas atliekamas naudojant du PID valdiklius sujungtus nuosekliai. O absoliuti pozicija valdoma atviro-ciklo valdiklio, kuriam yra saugoma tam tikrų kampinių pozicijų lentelė ir jos duomenys yra persiunčiami kampinės pozicijos valdikliams tam tikru laiko momentu.

Summary

During this thesis algorithms for autonomus quatcopter control has been developed and tested in a real environment. Autonomus control of quadcopter has been divided into three parts: Angular position estimation, angular position control and absolute position control. Angular position estimation is being performed with the help of quaternions in order to exrpess 3D rotations efficiently and avoid gimbal lock problem. Angular position control is achieved using two PID controllers connected in series. And absoulte position control is achieved by using open-loop controller where a table of certain angular positions are saved and at apropriate times sent to the angular position controller algorithms.

Turinys

Įvadas

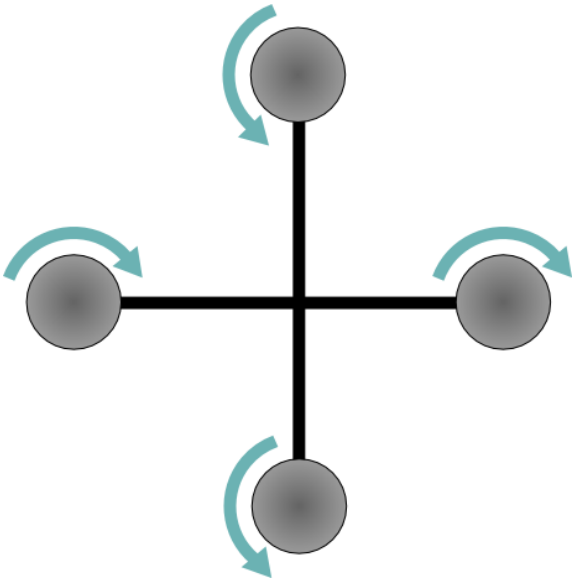
Ateityje ketursraigčiai galės Ketursraigčiai labai greitai populiarieja, taigi jų ir jų valdymo algoritmų poreikis auga kartu.

1. Ketursraigčio techninė įranga

Įprasto ketursraigčio techninė struktūra yra palyginus lengvai suprojektuojama bei pagaminama tačiau to negalima pasakyti apie programinę įrangą, bei algoritmus valdančius skrydį.

1.1. Rėmas, varikliai ir propeleriai

Ketursraigtis susideda iš „X“ formos rėmo, kurio galuose yra po elektrinį bešepetėlinį variklį su propeleriu. Priešingai nei įprastuose sraigtasparniuose, šių propelerių atakos kampas nėra reguliuojamas, o tai leidžia stipriai supaprastinti skraidyklės techninę struktūrą ir atsisakyti sudėtingų mechaninių dalių. Varikliai skirstomi į dvi grupes iš kurių viena sukasi pagal laikrodžio rodyklę, kita – prieš laikrodžio rodyklę. Šių variklių sukimosi greitis yra reguliuojamas siekiant išgauti tinkamą sukamąją bei keliamąją jėgas.



1 pav. Ketursraigčio rėmo ir variklių išdėstymas. Jų sukimosi kryptys.

Šio darbo tikslams pasiekti buvo nupirktas „HobbyKing x525“ 600mm skersmens rėmas, pagamintas iš aliuminio ir stiklo pluošto. Taip pat elektriniai bešepetėliniai 160W galios varikliai „Turnigy D2822“ ir 8 colių ilgio, bei 4 laipsnių atakos kampo propeleriai.

1.2. Valdymo elektronika

Stabilaus skrydžio išlaikymas ketursraigtyje yra per sudėtinga užduotis žmogui (pilotui) todėl pasitelkiama pagalbinė elektronika palengvinanti ketursraigčio valdymą.

Valdymo elektronikai išskiriami tokie uždaviniai:

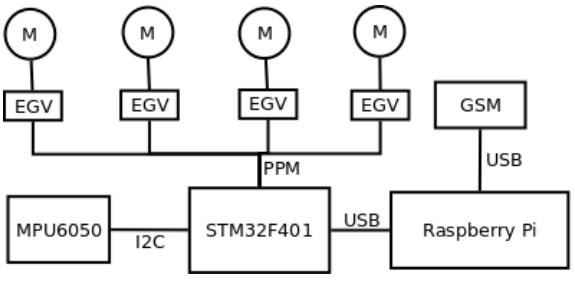
- Skrydžio stabilizavimas bei kontrolė;
- Ryšio su pilotu arba valdančia sistema palaikymas;
- Galios signalų, reikalingų varikliams, generavimas;

Skrydžio stabilizavimas bei kontrolė. Skrydžio stabilizavimui bei kontrolei atlikti naudojami sensoriai pagal kurių duomenis yra paskaičiuojami kokių korekcinių veiksmų reikia imtis norint įgyvendinti piloto ar valdančios sistemos komandas. Išskiriami svarbiausi parametrai yra tikslumas bei greitis.

Atsižvelgiant į rekalavimus šiems parametrams, buvo parinktas kompanijos „STMicroelectronics“ procesorius „STM32F401“ bei kompanijos „InvenSense“ sensorius „MPU6050“

Ryšio su pilotu arba valdančia sistema palaikymas. Ryšio palaikymas parametrizuojamas pagal duomenų persiuntimo greitį ir latenciją, bei veikimo ribas. Ketursaigčio atveju persiunčami duomenys yra tik valdymo signalai iš piloto arba valdančios sistemos, todėl buvo pasirinktas GSM ryšys. Taip pat „Raspberry pi“ kompiuteris su „Linux“ operacine sistema atlikdavo viską kas reikalinga GSM ryšiui palaikyti.

Galios signalų generavimas. Bešepetėliniai varikliai reikalauja trijų galios signalų jų sukimuisi palaikyti. Šiam tiklui buvo nupirkti 18A elektroniniai greičio valdikliai galintys suvaldyti apie 200W galios, tad puikiai tinkantys 160W galios varikliams.

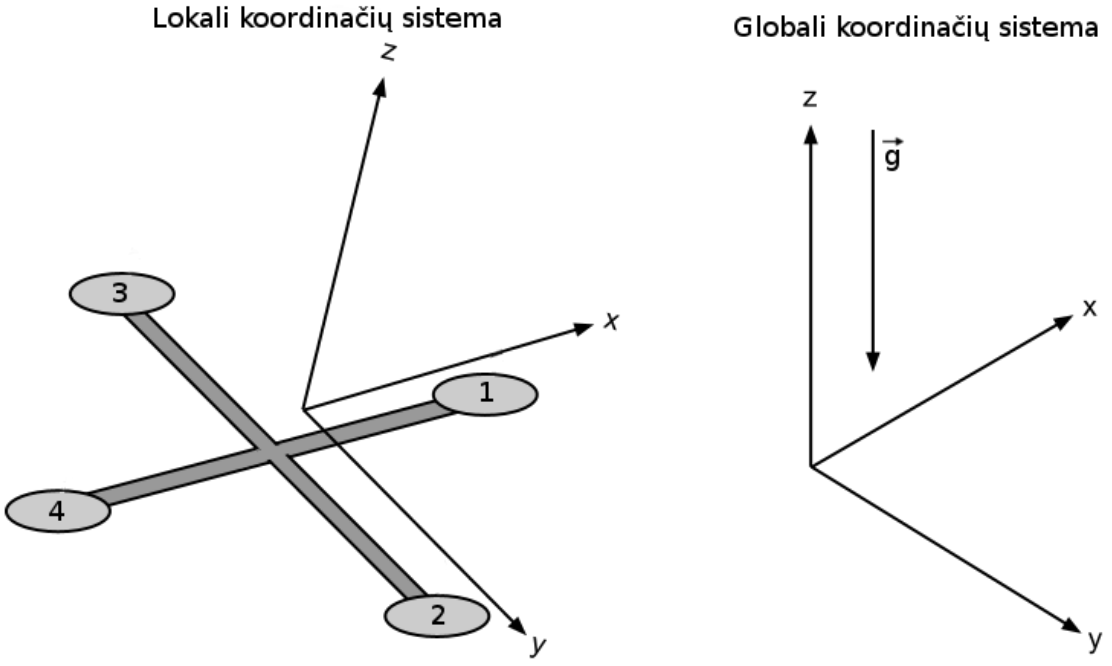


2 pav. Ketursraigčio elektronikos sudedamųjų dalių schema.

2. Matematinis skrydžio modelis

2.1. Lokali ir globali koordinačių sistemos

Pravartu apibrėžti lokalią ir globalią koordinačių sistemas, kuriose bus nagrinėjama ketursraigčio dinamika. Lokalioje koordinačių sistemoje X ir Y ašis yra sulygiuota su ketursraigčio rėmo strypais, kur X rodo pirmojo variklio kryptimi, Y - antrojo. Globali sistema yra susieta su žemės gravitaciniu lauku, ir toje sistemoje gravitacinio lauko vektorius nukreiptas priešinga Z ašiai kryptimi.



3 pav. Lokalios ir Globalios koordinačių sistemos palyginimas.

Konvertavimas tarp šių koordinačių sistemų bus vykdomas kvaternionų pagalba (žr.: TODO_SKYR_NR):

$$q_l = q_p * q_g * q_p^{-1} \tag{1}$$

ir į priešingą pusę:

$$q_g = q_p^{-1} * q_l * q_p \tag{2}$$

Čia q_g – tašką arba vektorių globalioje koordinačių sistemoje atvaizduojantis kvaternionas, q_l – tašką arba vektorių lokalioje koordinačių sistemoje atvaizduojantis

kvaternionas, q_p – kampinės pozicijos kvaternionas (žr.: TODO_SKYR_NR).

2.2. Keliamoji jėga

Ketursaigtis sukuria keliamąją jėgą priversdamas judėti orą žemyn. Kadangi naudojami nekintamo atakos kampo propeleriai, keliamoji jėga reguliuojama valdant propelerių sukimosi greitį. pagal (TODO_REFERENCE) keliamoji jėga gali būti apskaičiuota:

$$T = C * \omega^2 \tag{3}$$

Kur T yra keliamoji jėga niutonais vienam varikliui, C yra konstanta priklausanti tik nuo propelerio nekintamų savybių, o ω yra variklių sukimosi greitis.

Taip pat galime apskaičiuoti bendrąją keliamąją jėga visiems varikliams.

$$T = \begin{bmatrix} C_1 * \omega_1^2 - C_4 * \omega_4^2 \\ C_2 * \omega_2^2 - C_3 * \omega_3^2 \\ \sum_{i=1}^{4} C_i * \omega_i^2 \end{bmatrix} \tag{4}$$

Šiuo atveju T yra keliamosios galios vektorius lokalioje koordinačių sistemoje. Verta pastebėti, kad esant nevienodiems ω_1, ω_4 arba ω_2, ω_3 sukuriama jėga sukanti ketursraigtį apie jo lokalią Y arba X ašį atitinkamai.

O skaičiuojant globalioje koordinačių sistemoje:

$$q_{tl} = \begin{bmatrix} 0 \\ T_x \\ T_y \\ T_z \end{bmatrix} \tag{5}$$

$$q_{tg} = q_p^{-1} * q_{tl} * q_p \tag{6}$$

$$T_g = \begin{bmatrix} q_{tg_x} \\ q_{tg_y} \\ q_{tg_z} \end{bmatrix} \tag{7}$$

Čia q_{tl} yra kvaternionas reprezentuojantis keliamosios jėgos vektorių lokalioje koordinačių sistemoje, q_p kvaternionas reprezentuojantis pasukimą nuo globalios iki lokalios koordinačių sistemos (žr.: TODO_SECT_NR), q_{tg} – keliamosios jėgos vektorių atvaizduojantis kvaternionas globalioje koordinačių sistemoje, T_g – keliamosios jėgos vektorius globalioje koordinačių sistemoje.

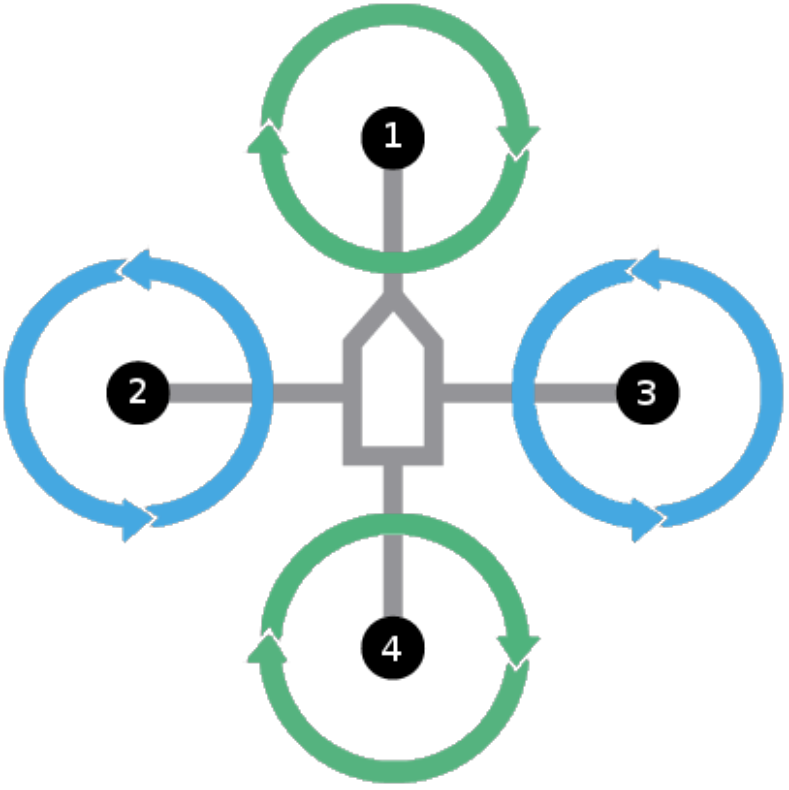
2.3. Sukamoji jėga

Besisukant propeleriams oro trintis į mentes sukelia papildomą jėgą, kuri veikia ketursraigčio rėmą. Kiekvienam iš variklių ši jėga yra nukreipta priešinga kryptimi nei sukasi variklis. Norint išvengti nevaldomo ketursraigčio sukimosi, parenkami du propeleriai skirti suktis pagal laikrodžio rodyklę ir du propeleriai skirti suktis prieš laikrodžio rodyklę, varikliai sujungiami taip, kad jų sukimosi kryptis atitiktų propelerį ir oro srautas būtų nukreiptas žemyn. Konkrečiu atveju varikliai 1 ir 4 sukasi prieš laikrodžio rodyklę, o varikliai 2 ir 3 – pagal. Šitaip vienos krypties jėga kompensuoja kitą ir tampa įmanomas skrydis be sukimosi apie Z ašį lokalioje koordinačių sistemoje.

Šią jėgą modeliuosime tiesiškai priklausančią nuo sukimosi greičio:

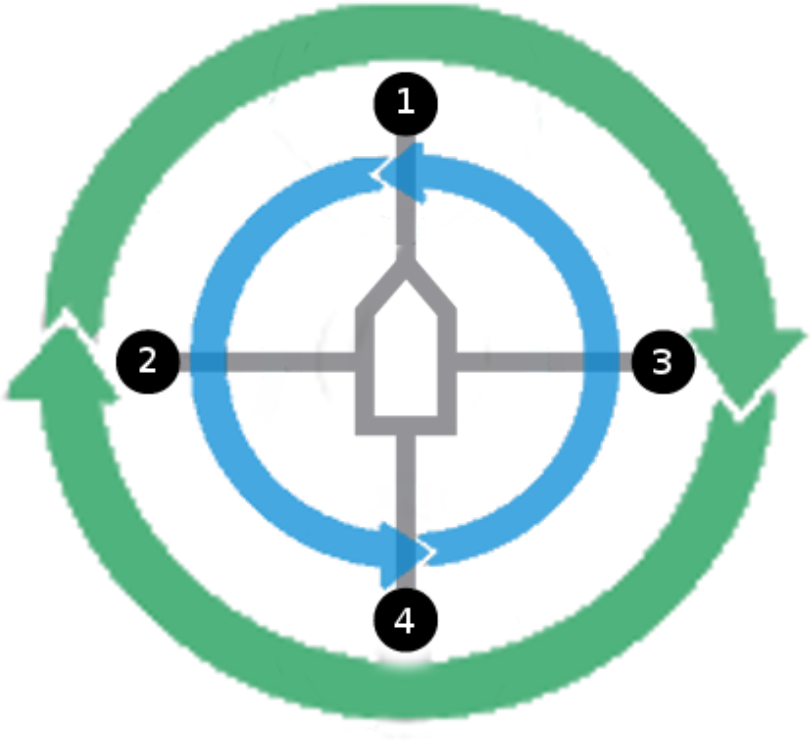
$$F_{tq} = -\omega * C \tag{8}$$

Čia C yra konstata priklausanti tik nuo propelerio savybių. Sukamoji jėga yra sukeliama oro pasipriešinimo, todėl ji yra priešinga judėjimo krypčiai ω.



4 pav. Jėgos sukeliamos dėl oro trinties į propelerius.

Verta pastebėti, kad varikliai skaidomi grupėmis pagal sukimosi kryptį. Tos pačios sukimosi krypties variklis sukuria tos pačios krypties sukamąją jėgą, tik centras kitoje vietoje. Tačiau rėmo atžvilgiu šias grupes galima nagrinėti kaip vieną jėga. Tokiu atveju turime dvi jėgas priešingų krypčių (žr.: TODO_PAV_REFERENCE).



5 pav. Jėgos sukeliamos dėl oro trinties į propelerius.

Šiuo atveju vidinės rodyklės rodo 2 ir 3 variklių sukamąsias jėgas, o išorinės – 1 ir 4 (žr.: TODO_REFERENCE_I_IMAGE). Verta pastebėti, kad rodyklių storis nėra susijęs su jėgos dydžiu.

Sukamosios jėgos neturi vieningos krypties, kuri nepriklausytų nuo stebėjimo pozicijos, todėl ji bus išreikšta ne vektorine forma, taigi nagrinėsime sukamosios jėgos dydį ties varikliais. Taip pat, sakysime, kad jėga yra teigiama kai sukamasi apie lokalią Z ašį pagal laikrodžio rodyklę. Tada gauname, kad sukamoji jėga:

$$F_{tq_{1,4}} = C * (\omega_1 + \omega_4) \tag{9}$$

$$F_{tq_{2,3}} = -C * (\omega_2 + \omega_3) \tag{10}$$

Ir bendra jėga:

$$F_{tq} = F_{tq_{1,4}} - F_{tq_{2,3}} = C * (\omega_1 - \omega_2 - \omega_3 + \omega_4) \tag{11}$$

Šias jėgų kryptis atitinkamai galime pasukti priešingu kvaternionu q_p, ir gausime jėgą globalioje koordinačių sistemoje. Verta pastebėti, kad šių jėgų kryptys visada sutaps su propelerių plokštuma.

2.4. Bendras judėjimo modelis

Ketursraigčio skrydžio judėjimai apibrėžti reikia įsivesti gravitacijos kryptį.

$$g_g = \begin{bmatrix} 0 \\ 0 \\ -9.8 \end{bmatrix} \tag{12}$$

g_g yra apibrėžtas globalioje koordinačių sistemoje.
Tuomet bendra jėga veikianti ketursraigtį:

$$F_g = T_g - m * g_g \tag{13}$$

Čia m yra ketusraigčio bendra masė.
Toliau pagal antrąjį Niutono dėsnį:

$$a_g = \frac{F_g}{m} \tag{14}$$

kur a yra pagreičio vektorius globalioje koordinačių sistemoje.
Toliau randamas greitis bei pozicijos priklausomybė nuo laiko:

$$v_g = a_g * \Delta t \tag{15}$$

$$x_g = a_g * \Delta t^2 \tag{16}$$

Tais pačiais principais randama ir kampinė pozicija.

3. Kampinės padėties skaičiavimas

Ketursraigčio valdymo algoritmai yra tiesiogiai priklausomi nuo kampinės padėties, bet nėra sensorių leidžiančių tai išmatuoti tiesiogiai, todėl šiame skyrelyje apibūdinami sukurti principai pagal kuriuos yra skaičiuojama ketursraigčio kampinė pozicija globalios koordinačių sistemos atžvilgiu.

Ketursraigčių kampinės padėties skaičiavimui buvo parinkta matematinė skaičiavimo sistema vadinama kvaternionais. Šioje sistemoje trimačiai pasukimai gali būti atvaizduoti labai efektyviai ir patogiai.

3.1. Kvaternionai

Kvaternionai matematikoje yra skaičių sistema kurį išplėčia įprastus kompleksinius skaičius. Kvaternionas tai yra vektorius:

$$q = \begin{bmatrix} w \\ i \\ j \\ k \end{bmatrix} \tag{17}$$

kur w yra sveikoji dalis, o i, j, k, yra menamosios.

Menamosios dalys tarpusavyje susiejamos pagal:

$$i^2 = j^2 = k^2 = ijk = -1 \tag{18}$$

o daugindami kiekvieną elementą su kiekvienu gauname tokį sąryšį:

x	**1**	**i**	**j**	**k**
1	1	i	j	k
i	i	-1	k	-j
j	j	-k	-1	i
k	k	j	-i	-1

Įprastas kvaternionas užrašomas kaip šių elementų suma:

$$q = w + xi + yj + zk \tag{19}$$

kur w yra sveikoji dalis, x, y, z yra sveikieji daugikliai, o i, j, k yra menamosios kvaterniono dalys.

Bet šiame darbe kvaternionai bus nagrinėjami vektorine forma:

$$q = \begin{bmatrix} w \\ x * i \\ y * j \\ z * k \end{bmatrix} \tag{20}$$

Taip pat kai kur kvaternionas bus nagrinėjamas jo sudedamosiomis dalimis, todėl q_w atitinka kvaterniono sveikąją dalį, q_x atitinka i sveikajį daugiklį, q_y – j sveikajį daugiklį, q_z – k sveikajį daugiklį.

Daugyba. Kvaternionų daugyba matematiškai išreiškiama kaip jo elementų sudauginimas prastinant pagal distributyvumo taisyklę, ir tai vadinama Hamiltono produktu.

$$q_1 * q_2 = [w_1, x_1 i_1, y_1 j_1, z_1 k_1] * [w_1, x_1 i_1, y_1 j_1, z_1 k_1] \tag{21}$$

Verta pastebėti, kad čia nėra paprasta vektorinė daugyba. Daugyba apibrėžiama taip:

$$q_1 * q_2 = \begin{bmatrix} w_1 w_2 - x_1 x_2 - y_1 y_2 - z_1 z_2 \\ (w_1 x_2 + x_1 w_2 + y_1 z_2 - z_1 y_2) i \\ (w_1 y_1 - x_1 z_2 + y_1 w_1 + z_1 x_1) j \\ (w_1 z_1 + x_1 y_2 - y_1 x_1 + z_1 w_1) k \end{bmatrix} \tag{22}$$

Invertavimas. Invertuotas kvaternionas yra kvaternionas atvaizduojantis pasukimą į priešingą pusę ir žymimas:

$$q^{-1} \tag{23}$$

Invertuoti kvaternioną galima dvejopai, arba invertuojant sveikąją dalį

$$q^{-1} = \begin{bmatrix} -q_w \\ q_x \\ q_y \\ q_z \end{bmatrix} \tag{24}$$

arba invertuojant visus tris menamuosius daugiklius

$$q^{-1} = \begin{bmatrix} q_w \\ -q_x \\ -q_y \\ -q_z \end{bmatrix} \tag{25}$$

Šis metodas ir bus naudojamas toliau.

Taško erdvėje atvaizdavimas kvaternionu. Taškas erdvėje pagal jo X, Y, Z koordinates kvaternionu atvaizduojamas taip:

$$q = \begin{bmatrix} 0 \\ Xi \\ Yj \\ Zk \end{bmatrix} \tag{26}$$

Erdvinio pasukimo atvaizdavimas kvaternionu. Įprastą pasukimą kuomet žinomas kampas ϕ kuriuo pasukama ir ašis X, Y, Z, aplink kurią (prieš laikrodžio rodyklę) yra sukama, atvaizduojamas taip:

$$q = \begin{bmatrix} cos(\frac{\phi}{2}) \\ X * sin(\frac{\phi}{2}) * i \\ Y * sin(\frac{\phi}{2}) * j \\ Z * sin(\frac{\phi}{2}) * k \end{bmatrix} \tag{27}$$

Po suformavimo kvaternionas turi būti normalizuotas.

Tuomet taško erdvėje pozicija po pasukimo kvaternionu q randama taip:

$$q_{tp} = q * q_t * q^{-1} \tag{28}$$

Čia q_{tp} yra tašką atvaizduojantis kvaternionas po pasukimo, q_t yra tašką atvaizduiojantis kvaternions prieš pasukimą.

Lerp. Lerp yra trumpinys tiesinei interpoliacijai (ang.: *Linear Interpolation*). Trimačių pasukimo kontekste, Lerp yra aproksimacija Slerp (ang.: *Spherical Interpolation*).

Darant prielaidą, kad kvaternionas išreiškia pasukimą taip kaip nurodoma pastraipoje „Erdvinio pasukimo atvaizdavimas kvaternionu“ tuomet funkcija Slerp išreiškiama taip:

$$Slerp(q_0, q_1, t) = (q_1 q_0^{-1})^t q_0 \tag{29}$$

Slerp funkcija randa kvaternioną atvaizduojantį pasukimą proporcingą pateiktam skaliarui su ribomis nuo pirmo pateikto kvaterniono iki antro. Pavyzdžiui jeigu turime kvaternioną q_0 sukanti 90°dešinėn, ir kvaternioną q_1 sukantį 90°kairėn, tai gautume tokius rezultatus priklausomai nuo skaliaro

skaliaras	**pasukimo kryptis**	**pasukimo kampas**
0.0	dešinėn	90°
0.25	dešinėn	45°
0.5		0°
0.75	kairėn	45°
1.0	kairėn	90°

Lerp funkcija apibrėžiama taip:

$$lerp(q_0, q_1, t) = \begin{bmatrix} q_{0_w} + (q_{1_w} - q_{0_w}) * t \\ (q_{0_x} + (q_{1_x} - q_{0_x}) * t)i \\ (q_{0_y} + (q_{1_y} - q_{0_y}) * t)j \\ (q_{0_z} + (q_{1_z} - q_{0_Z}) * t)k \end{bmatrix} \tag{30}$$

Slerp funkcija suteikia vienodą judėjimo greitį nuo kvaterniono q_0 prie q_1 priklausomai nuo skaliaro, tačiau reikalauja bent trijų kvaternionų daugybų ir kėlimo laipsniu. Tuo tarpu funkcija lerp, nors ir nesutekia, vienodo judėjimo greičio, bet ji šį judėjimą aproksimuoja pakankamai tiksliai ketursraigčio kampinės pozicijos skaičiavimo tikslams. Taip pat lerp galima apskaičiuoti tik su 4 skaliarinės daugybos operacijom, bei keleta atimčių ir sudėčių.

3.2. Sensoriai

Ketursraigčio kampinė pozicija skaičiuojama pagal du sensorius: giroskopą ir akselerometrą.

Giroskopas. Giroskopas yra sensorius matuojantis keturšraigčio kampinį greitį (visomis trimis ašimis)

$$\omega_{gyro} = \begin{bmatrix} \omega_{gyro_x} \\ \omega_{gyro_y} \\ \omega_{gyro_z} \end{bmatrix} \tag{31}$$

Verta pastebėti, kad visi matavimai atliekami lokalioje koordinačių sistemoje, o mažos paklaidos dėl netikslumų montuojant giroskopą yra ignoruojamos.

Akselerometras. Akselerometras yra sensorius matuojantis akseleracijas visomis trimis ašimis lokalioje koordinačių sistemoje. Verta pastebėti, kad žemės gravitacija sensorių veikia lygiai taip pat, kaip akseleracija priešinga kryptimi. Kadangi žemės graviacijos kryptis yra visada nukreipta žemyn, todėl jos kryptis gali būti naudojama kampinės padėties apskaičiavimui.

$$a_{acc} = \begin{bmatrix} a_{acc_x} + g_x \\ a_{acc_y} + g_y \\ a_{acc_z} + g_z \end{bmatrix} \tag{32}$$

Kaip ir giroskopo atveju, akselerometras atlieka matavimus lokalioje koordinačių sistemoje, o mažos paklaidos dėl netikslumų montuojant akselerometrą yra ignoruojamos. g_x, g_y, g_z, yra gravitacijos vektoriaus komponentės lokalioje koordinačių sistemoje.

3.3. Kampinės padėties skaičiavimas pagal giroskopą

Kaip jau aptarta ankščiau, giroskopas matuoja kampinį greitį, todėl norėdami rasti poziciją turime jį integruoti.

$$x = \int \omega \tag{33}$$

tačiau realus giroskopas atlieka matavimus diskretiškai, todėl daroma prielaida, kad išmatuotas kampinis greitis ω_t yra toks viso laiko momento t metu, tada:

$$x_t = x_{t-1} + \omega_t * \Delta t \tag{34}$$

Kur x_t yra kampinė pozicija laiko momentu t, o ω_t yra kampinis greitis laiko momentu t.

Tą patį išreiškus kvaternionais:

$$q_{pg_t} = q_{p_{t-1}} * q_{g_t} \tag{35}$$

Čia q_{pg_t} kvaternionas laiko momentu t reprezentuojantis pasukimą nuo globalios iki lokalios koordinačių sistemos ir paskaičiuotas pagal giroskopą, q_{g_t} kvaternionas reprezentuojantis pasisukimą laiko momentu t. $q_{p_{t-1}}$ yra galutinės kampinės pozicijos kvaternionas pasukantis globalią koordinačių sistemą iki lokalios laiko momentu $t-1$ (žr.: TODO_REFERENCE).

Kvaternionas q_{g_t} yra sudaromas pagal principus apibūdintus skyrelyje **TODO: reference**, kuomet sukimo kampas ϕ yra giroskopo išmatuoto vektoriaus ilgis:

$$q_{g_t} = \begin{bmatrix} cos(\frac{\sqrt{\omega_x^2+\omega_y^2+\omega_z^2}}{2}) \\ \frac{\omega_x}{\sqrt{\omega_x^2+\omega_y^2+\omega_z^2}} * sin(\frac{\sqrt{\omega_x^2+\omega_y^2+\omega_z^2}}{2}) \\ \frac{\omega_y}{\sqrt{\omega_x^2+\omega_y^2+\omega_z^2}} * sin\frac{\sqrt{\omega_x^2+\omega_y^2+\omega_z^2}}{2}) \\ \frac{\omega_z}{\sqrt{\omega_x^2+\omega_y^2+\omega_z^2}} * sin(\frac{\sqrt{\omega_x^2+\omega_y^2+\omega_z^2}}{2}) \end{bmatrix} \tag{36}$$

Čia ω_x, ω_y, ω_z, yra giroskopo atlikti matavimai. Sudarytas kvaternionas q_{g_t} turi būti normalizuojamas po suformavimo.

3.4. Kampinės padėties skaičiavimas pagal akselerometrą

Akselerometro matavimai, neesant išorinių akseleracijų, leidžia tiesiogiai apskaičiuoti kampinę sensoriaus poziciją, tačiau šis matavimas turi palyginus dideles paklaidas ir triukšmo lygį. Taip pat esant išorinėms akseleracijoms paklaidų pasiskirstimas gali būti necentruotas su gravitacijos vektoriaus reikšme. Todėl filtravimas yra būtinas ir jis įgyveninamas apjungiant kampines pozicijas apskaičiuotas pagal giroskopą ir pagal akselerometrą (žr.: TODO_REFERENCE)

Patogumo dėlei bus skaičiuojama ne absoliuti kampinė pozicija, o pasukimas, kuris pasuka dabartinę poziciją q_p iki akselerometro išmatuotos pozicijos. Tam tikslui pradžioje surandame vektorių lokalioje koordinačių sistemoje, kuris globalioje

koordinačių sistemoje yra nukreiptas tiesiai aukštyn.

$$q_{up_g} = \begin{bmatrix} 0 \\ 0 * i \\ 0 * j \\ 1 * k \end{bmatrix} \tag{37}$$

pasukame:

$$q_{up_l} = q_p * q_{up_g} * q_p^{-1} \tag{38}$$

Išskiriame vektorių viršun

$$v_{up} = \begin{bmatrix} q_{up_{l\,x}} \\ q_{up_{l\,y}} \\ q_{up_{l\,z}} \end{bmatrix} \tag{39}$$

Ir sudarome kvaternioną pavaizduojantį pasukimą nuo dabartinės apskaičiuotos kampinės pozicijos iki akselerometro išmatuotos pozicijos.

$$q_{acc_{diff}} = \begin{bmatrix} \sqrt{||v_{up}||^2 * ||a_{up}||^2} * (a_{up} \cdot v_{up}) \\ a_{up_y} * v_{up_z} - a_{up_z} * v_{up_y} \\ a_{up_z} * v_{up_x} - a_{up_x} * v_{up_z} \\ a_{up_x} * v_{up_y} - a_{up_y} * v_{up_x} \end{bmatrix} \tag{40}$$

Čia $\cdot$ reiškia „dot“ produktą, a_{up} yra akselerometro išmatuotas vektorius. O $||v||$ žymi vektoriaus v ilgį.

3.5. Galutinis kampinės padėties radimas

Pagal giroskopą apskaičiuota pozicija yra palyginus tiksli, bet ilgainiui gali krypti nuo tikrosios pozicijos, t.y. dreifuoti. Tuo tarpu pagal akselerometrą paskaičiuota pozicija yra palyginus triukšminga, bet negali dreifuoti, todėl skaičiuodami galutinę pozicija imame rezultatus, kuriuos duoda giroskopas ir pritaikome papildomąjį (ang.: *complementary*) filtrą.

$$q_{pg_t} = q_{p_{t-1}} * q_{g_t} \tag{41}$$

Tuomet sudaromas kvaternionas reprezentuojantis 0°pasukimą.

$$q_0 = \begin{bmatrix} 1.0 \\ 0.0 \\ 0.0 \\ 0.0 \end{bmatrix} \tag{42}$$

apskaičiuojamas svorio koeficientas pagal akselerometrą apskaičiuotai kampinei pozicijai filtruoti taip, kad šis koeficientas esant $||a|| = 1.0g$ būtų lygus C_{max} ir atitinkamai mažėtų iki C_{cutoff}.

$$W_{acc} = C_{max} - |||a|| - 1.0| * (\frac{C_{max}}{C_{cutoff}^2 - 1}) \tag{43}$$

Šis svoris W_{acc} yra apribojimas žemutine riba 0.

$$W = \begin{cases} W_{acc}, & \text{Jeigu } W_{acc} \geq 0 \\ 0, & \text{Kitu atveju} \end{cases} \tag{44}$$

Toliau yra apskaičiuojamas kvaternionas kuriuo pasukus q_{pg_t} bus atliktas kompensavimas pagal akselerometro duomenis

$$q_{diff} = lerp(q_0, q_{acc_{diff}}, W) \tag{45}$$

Ir pasukame q_{pg_t}

$$q_p = q_{pg_t} * q_{diff} \tag{46}$$

q_p yra galutinis pozicijos kvaternionas naudojamas ketursraigčio skrydžio valdymo algoritmuose.

4. Kampinės padėties valdymo algoritmas

Ketursraigčio kampinės padėties valdymui buvo parinktas PID algoritmas.

4.1. PID valdymo algoritmas

PID valymo algoritmas yra uždaro ciklo valdymo algoritmas. ID operuoja skaičiuodamas skirtumą tarp matuojamo kintamojo ir pateikto tikslo (ang.: *setpoint*). Toliau PID algoritmas reguliuoja proceso kintamąjį „stengdamasis“ sumažinti skirtumą tarp matuojamo kintamojo ir pateikto tikslo. PID yra parametrizuojamas trimis konstantomis:

- P – Proporcinis koeficientas;
- I – Integracinis koeficientas;
- D – Diferencinis koeficientas;

Bendroji valdymo lygtis PID algoritmui:

$$PID(t) = K_p e(t) + K_i \int_0^t e(t)dt + K_d \frac{d}{dt} e(t) \tag{47}$$

Čia $PID(t)$ yra proceso kintamasis, $e(t)$ – matuojamo ir tikslo kintamųjų skirtumas. K_p, K_i, K_d, PID algoritmo parametrai.

Verta pastebėti, kad skaitmeninio procesoriaus aplinkoje PID algoritmas dirba diskretiškai, todėl integracija ir diferenciacija išreiškiama atitinkamai suma bei skirtumu

$$\int_0^t x(t) \widehat{=} x_t + x_{t-1} + x_{t-2} + ... + x_1 + x_0 \tag{48}$$

$$\frac{d}{dt} x(t) \widehat{=} x_t - x_{t-1} \tag{49}$$

4.2. PID pritaikymas ketursraigčio valdymui

PID algorimtai pasiekia geriausių rezultatų dirbdami tiesinėse sistemose, todėl pirma bus išvedama kampinės pozicijos priklausomybė nuo proceso kintamojo, arba konkrečiu atveju – propelerių greičio.

$$F = \omega^2 * C \tag{50}$$

Šiuo atveju ω yra tai, kas yra valdoma PID algoritmo, tiesiškumui pasiekti, ištraukiama šaknis iš PID išeities. Iš to yra formuojamas signalas greičio valdikliams, kurių atsakas yra tiesiškas.

$$u = \sqrt{PID_{speed}(t)} \tag{51}$$

tuomet

$$F = \sqrt{u}^2 * C \tag{52}$$

Kadangi variklio valdiklių valdymo signalai niekada nebūna neigiami:

$$F = u * C \tag{53}$$

Iš antrojo niutono desnio:

$$a = \frac{F}{m} \tag{54}$$

Taigi:

$$a = \frac{u * C}{m} \tag{55}$$

Kadangi masė m yra nekintantis dydis skrydžio metu, konstantą C pervadiname į:

$$C \widehat{=} \frac{C}{m} \tag{56}$$

Čia dešinėje lygties senoji konstanta C, kairėje lygties pusėje - naujoji konstanta C.

Įvedame laiko priklausomybę:

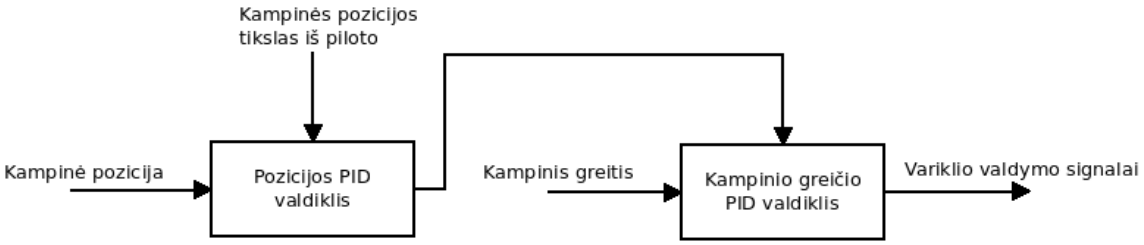
$$\omega = a * t \tag{57}$$

$$\omega = u * t * C \tag{58}$$

Šioje vietoje turimas tiesinis modelis, kuriam taikomas PID algoritmas ketursraigčio kampinio greičio valdymui. Tačiau tikslas yra valdyti kampinę poziciją, o ne greitį, todėl reikia išsivesti greičio priklausomybė nuo laiko:

$$x = \omega * t \tag{59}$$

Pozicija nuo greičio priklauso tiesiškai, todėl čia naudojamas antrasis PID algoritmas nuosekliai jungiamas su kampinį greitį valdančiuoju PID algoritmu (žr.: TODO_REFERENCE_IMAGE).



6 pav. Nuoseklus PID jungimas greičiui ir kampiniai pozicijai valdyti. PID valdikliai vaizduojami su rodykle iš kairės reiškiančią matuojamą proceso kintamąjį, iš viršaus tikslo kintamąjį, dešinėje – PID išeitis.

PID paklaidos skaičiavimas. Ketursraigčio kampinės padėties valdymui naudojami du tokie PID valdikliai. Vienas valdo sukimąsi apie lokalią X ašį, valdydamas apie Y ašį išdėliotus variklius, kitas valdo sukimąsi apie Y ašį valdydamas apie X ašį išdėliotus variklius. Matuojamas proceso kintamasis yra kampas nuo horizontalios padėties. O PID paklaida (ang.: *error*) yra skirtumas tarp matuojamo kintamojo (kampo nuo horizontalios pozicijos) ir piloto (arba autopiloto) nustatytos reikšmės.

PID paklaidą skaičiuojama pagal q_p kvaternioną (žr.: TODO_REFERENCE):

Iš pradžių apskaičiuojamas aukštis, kuriame yra atitinkamas X arba Y variklis. Tam sudaromas kvaternionas žymintis variklio poziciją lokalioje koordinačių sistemoje.

$$q_{eng_x} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \tag{60}$$

Šis kvaternionas pasukamas pozicijos kvaternionu:

$$q_{eng_{x\,h}} = q_p * q_{eng_x} * q_p^{-1} \tag{61}$$

Ir jo pozicija projektuojama į Z ašį.

$$h_x = q_{eng_{x\,h\,z}} \tag{62}$$

Tuomet apskaičiuojamas kampas:

$$\alpha = asin(h_x) \tag{63}$$

α jau gali būti naudojamas kaip matuojamas proceso kintamasis pozicijos PID valdikliui, taip pat $\frac{d}{dt}\alpha$ naudojamas greičio PID valdikliui, tik diskretizuota versija:

$$\frac{d}{dt}\alpha \widehat{=} \alpha_t - \alpha_{t-1} \tag{64}$$

Verta pastebėti, kad šitoks skaičiavimas teisingai paklaidas skaičiuoja tik iki tol, kol jos neviršyja 90°, bet ketursraigtis negali išsilaikyti ore palinkęs tokiais stipriais kampais, todėl atvejai su tokiais kampais nėra nagrinėjami.

PID valdiklio išeitis yra naudojama modifikuoti dabartiniam galios tikslui (ang.: *setpoint*), jeigu tai yra PID valdiklis valdantis pasisukimą pagal X ašį, tuomet antram varikliui galia yra didinama tiek, kokia yra PID išeitis, o trečiam atitinkamai mažinama. Analogiškai veikia ir Y ašis.

4.3. Pasisukimo pagal lokalią Z ašį valdymas

Ketursraigčių pasisukimas apie Z ašį remiasi tuo, kad iš keturių propelerių du sukasi pagal ir du prieš laikrodžio rodyklę. Oro pasipriešinimas sukelia jėgą verčiančią ketursraigtį suktis priešingą kryptimi negu propeleriai. Kol bendras jėgų didumas propeleriams besisukantiems pagal laikrodžio rodyklę yra panašus į jėgų didumą propeleriams besisukandiems prieš laikrodžio rodyklę, tol šios jėgos viena kitą kompensuoja ir ketursraigtis nesisuka apie savo lokalią Z ašį. Pakeitus šį balansą galima priversti ketursraigtį suktis viena ar kita kryptimi, tačiau šia ašimi ketursraigčiai yra kur kas mažiau manevringesni negu X ar Y todėl valdymo algoritmas yra kur kas paprastesnis.

Kaip ir PID kontrolerio atveju, čia skaičiuojama pasisukimo pagal Z ašies paklaida.

Sugeneruojamas kvaternionas atitinkantis X variklio poziciją lokalioje koordinačių sistemoje.

$$q_{eng_x} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \tag{65}$$

Pasukamas pagal esamą poziciją

$$q_{eng_{xl}} = q_p * q_{eng_x} * q_p^{-1} \tag{66}$$

Čia imamas X, Y vektorius ir normalizuojamas.

$$x_{nenorm} = \begin{bmatrix} q_{eng_{xl_x}} \\ q_{eng_{xl_y}} \end{bmatrix} \tag{67}$$

$$x = \frac{x_{nenorm}}{||x_{nenorm}||} \tag{68}$$

Ir skaičiuojamas kampas

$$\alpha_z = \begin{cases} acos(x_x), & \text{Jeigu } x_y \geq 0 \\ -acos(x_x), & \text{Kitu atveju} \end{cases} \tag{69}$$

Valdymo signalai yra gaunami:

$$u_z = \alpha_z * K_{p_z} \tag{70}$$

Šis valdymo signalas yra pridedamas prie variklių galios besisukančių pagal laikrodžio rodyklę, ir atimamas iš variklių, besisukančių prieš laikrodžio rodyklę.

5. Skrydžio autonomiškumas

Skrydžio autonomiškumas pasiekiamas atviro ciklo valdikliu, siunčiant valdymo komandas žemesniame valdymo lygmenyje esantiems PID valdikliams.

5.1. Atviro-ciklo valdymas

Atviro ciklo valdiklis tai yra valdiklis, kuris skaičiuoja valdymo parametrus tik pagal dabartinę būseną ir neatlieka valdomo objekto matavimų. Šio tipo valdiklis ketursraigčiui buvo parinktas dėl fizinių sensorių nebuvimo, kurie galėtų matuoti ketursraigčio poziciją.

5.2. Kampinės pozicijos tikslų lentelė

Ketursraigčio valdymas atliekamas valdant bendrąją galią bei kampą pagal X, Y, Z ašis. Šiuos duomenis priima PID ir P valdikliai, kaip tikslo kintamuosius, ir nustato atitinkamas variklių galias.

Ketursraigčio valdymui sudaroma šių tikslų lentelė ir jie yra siunčiami ketursraigčiui periodiškai.

galia	**kampas X**	**Kampas Y**	**Kampas Z**	**Galiojimo laikas**
20 %	0°	0°	0°	2s
40 %	0°	5°	0°	1s
40 %	0°	-5°	0°	1s
40 %	-5°	0°	0°	1s
40 %	5°	0°	0°	1s
20 %	0°	0°	0°	2s

1 lentelė. Supaprastintas tikslų lentelės pavizdys

Nepriklausomai nuo galiojimo laiko, tikslai yra siunčiami 10Hz dažniu, jeigu galiojimo laikas yra ilgesnis – siunčiamas tas pats tikslas, kol galiojimo laikas pasibaigia. Taip ketursraigtis turi galimybę atpažinti ar nenutrūko ryšys su pilotu ar pilotuojančia programa.

5.3. Atviro-ciklo valdymo trūkumai

Vertinant šį algoritmą pagal pozicijos paklaidas (lyginant du skrydžius vieną su kitu), yra labai didelės paklaidos dėl šio algoritmo nesugebėjimo prisitaikyti prie

nenumatytų išorinių ir vidinių veiksnių (pvz.: Vėjo). Taip pat PID valdikliai nėra tikslūs, o tai dar labiau padidina paklaidas.

TODO_DIAGRAM_OF_ANGLE_ERROR

TODO_DISTURBANCE_EXAMPLE

6. Programinė įranga

6.1. Bendroji architektūra

Ketursraigtis susideda iš trijų esminių fizinių dalių:

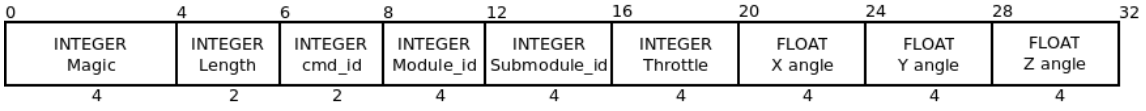
- Kompiuterio – Formuojančio ir siunčiančio valdymo komandas.
- Retransmitoriaus – Persiunčiančio valdymo komandas ketursraigčiui.
- Pagrindinės valdymo elektronikos – Valdančios ketursraigčio skrydį.

Kiekviena iš šių dalių turi savo programinę įrangą.

6.2. Kompiuteriui skirtas klientas

Kompiuteriui skirto kliento buvo paruoštos dvi versijos, viena skirta rankiniam ketursraigčio valdymui vairalazde, kita įgyvendinanti TODO_REFERENCE_OPEN_LOOP aptartą atviro ciklo valdymo algoritmą.

Abiejais atvejais tai yra „python“ kalba parašyta programėlė, 10Hz dažniu nuskaitydama tikslo vertes (ar iš lentelės ar iš vairalazdės) ir suformuodama UDP paketą (žr.: TODO_REFERENCE)



7 pav. Valdymo paketas siunčiamas valdymo elektronikai. Skaičius apačioje rodo lauko ilgį baitais, viršuje lauko poslinkį nuo paketo pradžios

6.3. Retransmitorius

Retransmitorius kartu yra ir fizinis įrenginys. Tai yra raspberry pi kompiuteris su Linux operacine sistema. Šis kompiueris prisijungia prie 3G/4G tinklų ir klausosi UDP paketų siunčiamų jo adresu. Visą gautą informaciją jis tiesiog persiunčia per USB sąsają valdymo elektronikai.

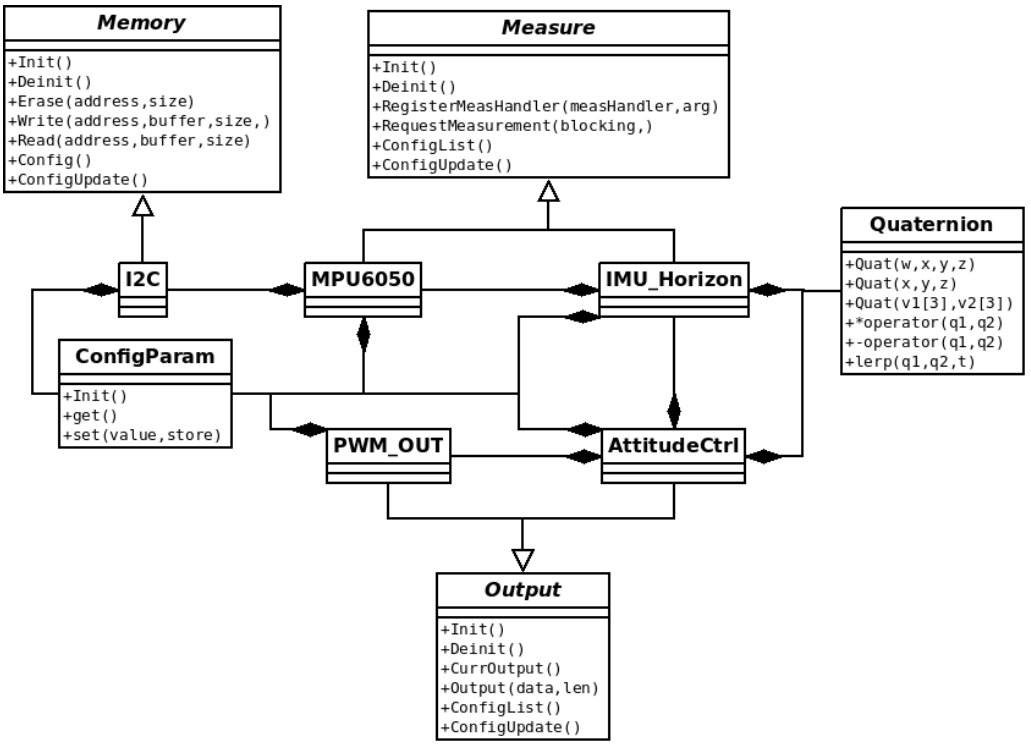
6.4. Ketursraigčio pagrindinis valdiklis

Pagrindinės valdymo elektronika sudaro GY-86 sensorių plokštė, su giroskopu ir akselerometru, keturi vienetai greičio valdiklių, ir STM32F401 procesoriumi. Iš

šitų elementų vienintelis STM procesorius reikalauja programavimo, tad jam buvo parašyta programinė įranga įgyvendinanti visus sukurtus algoritmus ir vykdanti skrydžio kontrolę. STM programinė įranga yra išskaidyta į modulius.

- I2C – modulis skirtas palaikyti komunikacijai su sensoriais.
- MPU6050 – sensorių tvarkyklės modulis.
- Quaternion – kvaternionų algebrą įgyveninantis modulis.
- IMU_Horizon – pozicijos kvaternioną skaičiuojantis modulis.
- AttitudeCtrl – kampinės pozicijos valdymo modulis.
- PWM_OUT – modulis generuojantis signalą variklių valdikliams.
- Connection – USB komandų palaikymo modulis.
- ConfigParam – abstraktus konfiguracijos parametru modulis.

Verta pastebėti, kad kodas rašytas C kalba, bet šiek tiek mėgdžiojami kai kurie OOP principai. Todėl toliau pateikiama klasių diagrama atitinkanti programinės įrangos struktūrą, jei ji būtų rašoma objektine programavimo kalba.



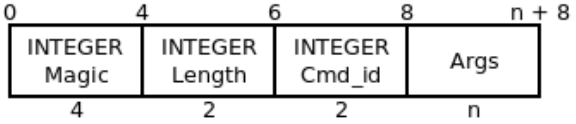
8 pav. Vidinė pagrindinio valdiklio programinė struktūra.

„Connection“ modulis. „Connection“ yra modulis, kuris jungiasi su visomis klasėmis, dėl to atskirai nebuvo pavaizduotas diagramoje. Jis turi tik vieną „Init“ metodą. „Connection“ modulis yra atsakingas už informacijos paėmimą iš USB sąsajos ir jos perdavimą atskiriems moduliams pagal jų ID. Komunikacija per USB sąsaja palaikoma pagal serverio-kliento principus. Kompiuteris būdamas klientu siunčia užklausas ir sulaukia atsakymo kiekvienai užklausai.

"Connection“ modulis palaiko šias komandas:

1. Reset - perkrauna įrenginį;
2. Enable device - aktyvuoja modulį;
3. Disable device - deaktyvuoja modulį;
4. SetConfig - nustato parametro reikšmę;
5. ListActive - pateikia aktyvių modulių sąrašą;
6. ListSupported - pateikia palaikomų modulių sąrašą;
7. ListParam - Pateikia konkretaus akyvaus modulio parametrų sąrašą;
8. Output - Iškviečia modulio, įgyvendinančio „Output“ sąsają metodą „Output“;
9. CurrOutput - Iškviečia modulio, įgyvendinančio „Output“ sąsają metodą „CurrOutput“;
10. ListChecker - pateikia klaidų žurnalą;
11. RequestMeasurement - Iškviečia modulio, įgyvendinančio „Measure“ sąsają metodą „RequestMeasurement“;

Visos komandos paklusta baziniam protokolui, kuriame nurodomas komandos ilgis baitais, komandos identifikatorius be argumentai skirti konkrečiai komandai (žr.: TODO_REFERENCE_IMAGE).



9 pav. Bazinė „Connection"modulio užklausos paketo struktūra. Skaičius viršuje rodo poslinkį nuo paketo pradžios, apačioje - lauko ilgį.

Čia laukas „Magic“ yra paketo pradžios identifikatorius ir visada įgauna tokią pačią reikšmę, kurį yra lygi A5B7C957 šešioliktainėje sistemoje. Laukas „Length“ nurodo viso paketo ilgį (įskaitant ir „Magic“ bei „Length“ laukus). Kiekviena komanda turi savo ID, kuris pateikiamas lauke „Cmd_id“, o lauke „Args“ sudedama jau konkreti kiekvienos komandos reikalaujama informacija.

Išvados

Šiame darbe buvo sukurta fizikiniu modeliu besiremianti ketursraigčio skrydžio valdymo sistema, bei algoritmai kuriais ši sistema vadovaujasi.

Darbe buvo išvestas supaprastintas fizikinis ketursraigčio skrydžio modelis, sukurtas kvaternionais grystas algoritmas leidžiantis rasti ketursraigčio kampinę poziciją pasinaudojant nebrangiais MEMS giroskopo bei akselerometro sensoriais. Toliau pritaikomi uždaro-ciklo PID algoritmai ketursraigčio kampinės pozicijos valdymui, taip pat pateikiamas atviro-ciklo valdymo algoritmas ketursraigčio pozicijos valdymui. Visa tai buvo suprogramuota ir išbandyta realaus ketursraigčio skrydžio metu.

Rezultatai. Šiame darbe buvo sukurti bei išbandyti trys esminiai algoritmai: ketursraigčio kampinės pozicijos skaičiavimo, ketursraigčio kampinės pozicijos valdymo bei ketursaigčio pozicijos valdymo algoritmas. Kampinės pozicijos algoritmas parodė gerus rezultatus, net ir sunkiomis darbo sąlygomis, kuomet sensoriai dirbo stiprios vibracijos sąlygomis. Sensoriui dirbant normaliomis sąlygomis (vibracijų dažnis žemesnis nei ketvirtadalis matavimų dažnio, ir akseleracijos bei sukimo greičiai neviršyja sensoriaus matavimo ribų), buvo pasiekta mažesnė nei 1°paklaida, taip pat nebuvo aptikta jokių užrakinimo (ang.: *gimbal lock*) pozicijų, būdingų panašiems algoritmams besiremiantiems oilerio kampais. Tuo tarpu PID algoritmas parodė šiek tiek prastesnius rezultatus. Esant mažoms kampinės pozicijos paklaidoms buvo pastebimos lengvas ketursraigčio kampinės padėties svyravimas. Taip pat nebuvo rastas tinkamas parametrų rinkinys, kuris tenkintų reakcijos greitį bei sistemos stabilumo reikalavimus. Atviro ciklo pozicijos valdymas turėjo stiprias paklaidas, ir tiksliam autonominiam skrydžiui palaikyti – reikėtų ieškoti kito algoritmo.

Tolimesni darbai, patobulinimai. PID algoritmai nesuteikia optimalaus valdymo, ypač kai jų parametrai yra nustatomi rankiniu būdu ir randamas tik „pakankamai-geras“ parametrų derinys. Tuo tarpų tiesinis-kvadratinis reguliatorius (ang.: *linear-quadratic regulator* arba *LQG*), nors ir lieka šiek tiek rankinio algoritmo reguliavimo, teoriškai galėtų pasiekti geresnius rezultatus. Taip pat specialūs fizika paremti modeliai valdymui galėtų būti sukurti agresyviam ketursraigčio valdymui.

Pozicijos valdymas galėtų būti išreikštas uždaro-ciklo valdymo algoritmu, įtaisius tikslius GPS įtaisus, sukūrus modelį bei algoritmus pozicijos valdymui. Tokie patobulinimai pozicijos valdymo tikslumą galėtų pagerinti dešimtis ar net šimtus kartų. Taip pat oro slėgio sensoriai galėtų būti naudojami automatiniam skrydžio aukščio valdymui. Tam irgi turėtų būti sukurtas modelis, bei valdymo algoritmai.

[AAJ+01] – *Implementing a Sensor Fusion Algorithm for 3D Orientation Detection with Inertial/Magnetic Sensors*, http://franciscoraulortega.com/pubs/Algo3DFusionsMems.pdf

[SSF+11] – *A sensor fusion algorithm for an integrated angular position estimation with inertial measurement units*, http://www.date-conference.com/proceedings/PAPERS/2011/DATE11/PDFFILES/IP1_06.PDF

[MS11] – *Modeling, Design and Experimental Study for a Quadcopter System Construction*, http://brage.bibsys.no/xmlui/bitstream/id/86811/uiareport.pdf

[—-] – *Quadcopter Dynamics, Simulation, and Control* http://andrew.gibiansky.com/downloads/pdf/Quadcopter%20Dynamics,%20Simulation,%20and%20Control.pdf