VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

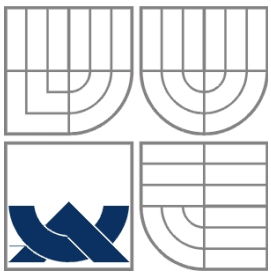
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

ŘÍZENÍ STABILITY KVADROKOPTÉRY

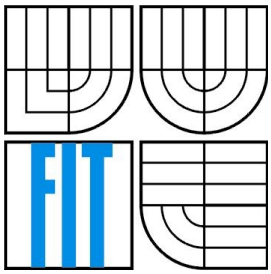
DIPLOMOVÁ PRÁCE
DIPLOMA THESIS

AUTOR PRÁCE BC. JAKUB NEJEDLÝ
AUTHOR

BRNO 2015



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

ŘÍZENÍ STABILITY KVADROKOPTÉRY

STABILITY CONTROL OF QUADCOPTER

DIPLOMOVÁ PRÁCE
DIPLOMA THESIS

AUTOR PRÁCE BC. JAKUB NEJEDLÝ
AUTHOR

VEDOUCÍ PRÁCE ING. JAROSLAV ROZMAN PH.D.
SUPERVISOR

BRNO 2015

Abstrakt

Dílo pojednává o fyzikálních zákonech ovlivňujících chování kvadrokoptéry jako mobilního robota. Popisuje metody řízení pohybů a stability. Výsledkem teoretické analýzy práce je vytvoření simulačního modelu.
Dále zachycuje praktický vývoj software řídící jednotky reálného stroje se samostatným závěrem, srovnání simulačních a praktických experimentů a popis pracovního postupu tvorby fyzikálního systému.

Abstract

This work deals with physical laws affecting behavior of a quadcopter as a mobile robot. It describes methods of controlling movements and stability. The result of the theoretical analysis is creation of simulation model.
Moreover it depicts practical software developement of a real machine controller unit with its own conclusion, comparison between simulation and practical experiments and the workflow of the physical system construction.

Klíčová slova

Multirotor, multikoptéra, kvadrokoptéra, řízení, stabilita, regulace, simulace, modelování, Arduino, implementace řídící jednotky.

Keywords

Multirotor, multicopter, quadcopter, control, stability, simulation, modeling, Arduino, controller unit implementation.

Citace

Jakub Nejedlý: Řízení stability kvadrokoptéry, diplomová práce, Brno, FIT VUT v Brně, 2015

Řízení stability kvadrokoptéry

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Jaroslava Rozmana Ph.D.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jakub Nejedlý
25.05.2015

Poděkování

Rád bych poděkoval svému vedoucímu Ing. Jaroslavu Rozmanovi Ph.D. za přístup k vedení práce, Mgr. Radku Baránkovi za poskytnutí odborných konzultací a mému otci Ing. Jaroslavu Nejedlému za pomoc s tvorbou reálné kvadrokoptéry.

Obsah

1 Úvod

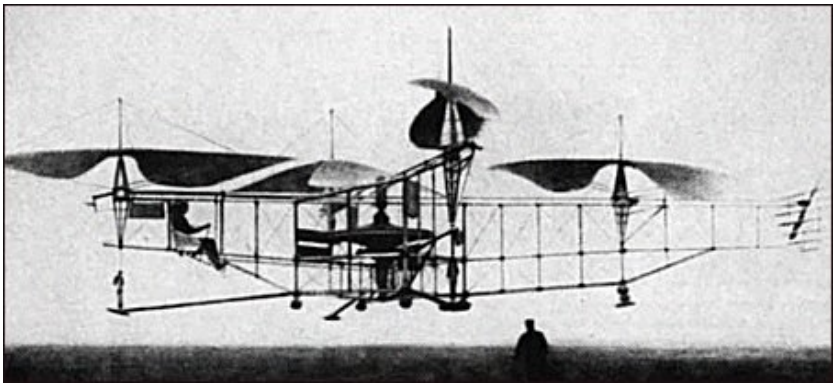
Kvadrokoptéra patří do skupiny létajících robotů zvaných multikoptéry, též multirotory. Jedná se o stroje se třemi a více rotory s vertikálně uloženými osami. Rotory mohou být schopny individuálního náklonu (tilt), což je vlastnost převzatá od vrtulníků, posléze bikoptér (Bell Boeing V-22 Osprey).

Kvadrokoptéru nejspíše poprvé navrhli bratři Breguetové s pomocí profesora Charlese Richeta. Jejich společný výtvor se nazýval Gyroplane No. 1. Trend tehdejší doby byl, na rozdíl od toho dnešního, vytvářet letouny nesoucí lidskou posádku. Proto Gyroplane vážil 500 kg a jeho výška byla 3,7 m. Koncept byl zajímavý v tom, že měl dvouúrovňové rotory – tedy listy jedné vrtule byly ve dvou planparalelních rovinách nad sebou. Tento stroj ovšem nikdy nebyl schopen samostatného letu pro svou neovladatelnost.

Etienne Oemichen, francouzský inženýr, se zabýval různými létajícími stroji s kolmým startem, a tak v roce 1922 dal vzniknout Oemichenu No. 2. Oemichen No. 2 měl ovšem kromě hlavních 4 rotorů používaných na vyvinutí vzletové síly také dalších 8 menších vrtulí pro korekci směru letu. Oemichen No. 2 byl dlouhý více než 10 metrů a vážil 800 kg. Tento kolos a celkem 12 aktuátorů poháněl jediný motor Gnome Rhone o 135 kW. Přes všechny úspěšné lety, z nichž jeden dokonce vyhrál cenu za první 1 km okružní let vrtulového stroje, nebyl koncept rozvíjen.

V roce 1956 zažilo nebe další pokus o prosazení kvadrokoptér v podobě Convertawings Modelu A. Tento multirotor měl dokonce 2 motory o výkonu 90 koňských sil. Sloužil také jako vzor pro další projekty 50. let a mnohé experimenty. Masové rozšíření ovšem svému konceptu, stejně jako jeho předchůdce, nezařídil.

Až nyní, v druhém desetiletí 21. století, zažívají multikoptéry rozmach. Díky dostupné elektronice mohou vznikat relativně malé letouny na dálkové ovládání. Stále se přichází na nová a nová praktická využití. Vzhledem k tomu, že operátor již nebývá fyzicky přítomen na palubě stroje, hovoříme o takvzaných dronech, nebo o UAV (Unmanned Aerial Vehicle).



Obrázek č. 1: Oemichen No. 2

2 Složení

Multikoptéra se typicky skládá z rámu, na který je po obvodu připevněn určitý počet ramen. Tento počet určuje, o kolika-koptéru se jedná. Každé rameno nese aktuátorovou jednotku (rotor) obsahující zpravidla pevně přichycený elektromotor točící vrtulí. Motor je ovládán rychlostním regulátorem na základě signálu z řadiče. Řadič může být plně autonomní nebo napojen na RC přijímač. O energetické potřeby systému se stará akumulátor.

2.1 Vrtule

Svým točivým pohybem vytváří tahovou sílu (propulzi) pro lokomoci. Hlavními parametry vrtule jsou počet listů (Blade count), průměr (Diameter) a náběh (Pitch).

Náběh určuje sklon listu vrtule od základní plochy, která je kolmá k ose otáčení. Sklon se většinou udává se radiánech. Je nutné si povšimnout, že není na celé délce listu stejný, ale kvůli změnšující se obvodové rychlosti se směrem ke středu vrtule zvětšuje. Konstanta vrtule *Kp* se používá v modelářství pro orientační výpočty a zapouzdřuje spoustu vlastností. Mezi ně patří i aerodynamické koeficienty C_L a C_D popisující, jak se na vrtuli projevuje prostupování tekutinou pro daný kolizní úhel. U multikoptér se nejčastěji setkáváme s vrtulemi vyrobenými z umělohmotných či karbonových materiálů.

2.2 Motor

Elektromotor dodává točivý moment pro otáčení vrtule. Pohyb se zakládá na působení magnetických sil pevných a elektrických magnetů. Dle způsobu provedení motory dělíme na střídavé (Alternating Current) a stejnosměrné (Direct Current). DC motory se dají dělit na mechanicky (brushed) či elektricky (brushless) komutované. AC motory jsou elektricky komutované z principu. V tomto textu se dále již budu věnovat pouze AC motorům.

Rychlost otáčení brushed motoru záleží čistě na příkonu a zatížení, kdežto otáčky brushless motorů se dají kontrolovat přímo. U většiny AC motorů určuje konstanta *Kv* o kolik více otáček za minutu vykoná při zvýšení napájecího napětí o 1 volt. Dle zátěže při daných otáčkách se již mění jen výše potřebného elektrického proudu.

Dalším důležitým atributem je například maximální bezpečný proud (popř. špičkový proud). Motory mají svůj vnitřní elektrický odpor R_m. Mimo to se značně projevuje i takzvaný proud na prázdno I_0, tedy velikost proudu, která nepřinese žádný užitek. Jelikož se proud na prázdno mění s napětím, objevuje se ve specifikaci i velikost referenčního napětí U_0, při kterém byl proud na prázdno naměřen. Pomocí motorové konstanty *Rk* se pak dá skutečná hodnota I_0 dále zpřesnit.

2.3 Regulátor rychlosti (otáček)

Regulátor se stará o řízení otáček motoru. Jeho vstupem jsou napětí z baterie a řídící signál.

Nejčastějším typem jsou elektronické regulátory (Electronic Speed Controller), které jsou řízeny pulzně-šířkovou modulací. Běžně používaný protokol vypadá takto: Rámec opakuje každou padesátinu sekundy a obsahuje pulz o šířce 1 až 2 milisekundy. Amplituda musí převyšovat minimální napětí, které se pohybuje lehce nad 3 volty.

V profesionálních systémech se využívají také digitální regulátory, které mají sériové rozhraní a jako signál tak očekávají číselnou hodnotu v binární podobě.

Regulátory mohou mít takzvaný Battery Eliminator Circuit - tedy obvod, který na svém výstupu podává regulované napětí (obvykle 5V), případně proud (obvykle 2A). Využívá se k napájení nevýkonových součástí systému, jako je RC přijímač, řadič, či servomotory.

Někdy také regulátor disponuje signalizací poklesu napájecího napětí. Většina kvalitních typů je také programovatelná, což mimo jiné znamená, že se v případě potřeby regulátor umí přizpůsobit konkrétnímu rozsahu PWM vstupu.

Udávané parametry regulátorů jsou vnitřní elektrický odpor R_r a maximální povolený procházející proud.

2.4 Řadič

Mozkem celé soustavy je řadič. Většinou plní funkci ve dvou úrovních, které nazvu bazální a rozšířená. Toto pořadí odpovídá postupu odspodu Maslowovou pyramidou.

Bazální – v této úrovni bychom se mohli setkat se senzorickou jednotkou, která snímá veličiny jako je výška, zrychlení, natáčení, orientace. Tato data jsou využita k řízení stability a základních pohybových funkcí. Bazální úroveň je hlavním předmětem tohoto textu.

Rozšířená – tato úroveň se již stará o vyšší funkce multikoptéry. Nejčastěji se jedná o předprogramované manévry, orientaci v okolí respektive v poloze. Dle náročnosti této části programového vybavení se řadič implementuje zcela na jednočipu nebo je využito PC jako externí výpočetní jednotka.

2.5 Baterie

Akumulátor je nedílnou součástí každého elektrického mobilního robota.

Nejdůležitějším parametrem je výstupní napětí U_b. V RC robotech bývá násobkem 3,7 voltů, protože se baterie skládá z článků. Například pokud je dodávané napětí 11,1 V, nejspíše se jedná se o takzvané zapojení 3S1P – 3 články v sérii a celkem jedna paralelní větev. Druhým důležitým údajem je kapacita. U baterií pro multikoptéry se pohybuje od desítek miliampér-hodin po desítky ampér-hodin. Pokud se jedná o výkonný multirotor, musíme se zajímat i o maximální vybíjecí proud baterie. Určuje jaký maximální výkon je baterie schopná poskytovat. Ve specifikaci je uváděn jako v jednotce s označením C (capacity). Baterie s ozanečním 30 C je schopná se bezpečně vybít nejrychleji za 1 / 30 hodiny. Výkon je ovšem jako u všech ostatních elektrotechnických součástek negativně ovlivněn jejím vnitřním odporem R_b.

$$Pmax \approx I_{dch} * (U_b - I_{dch} * R_b) \qquad (1)$$

Kde $Pmax$ je maximální výkon baterie, I_{dch} je maximální vybíjecí (discharge) proud, U_b poskytované napětí a R_b vnitřní odpor.

2.6 Přijímač

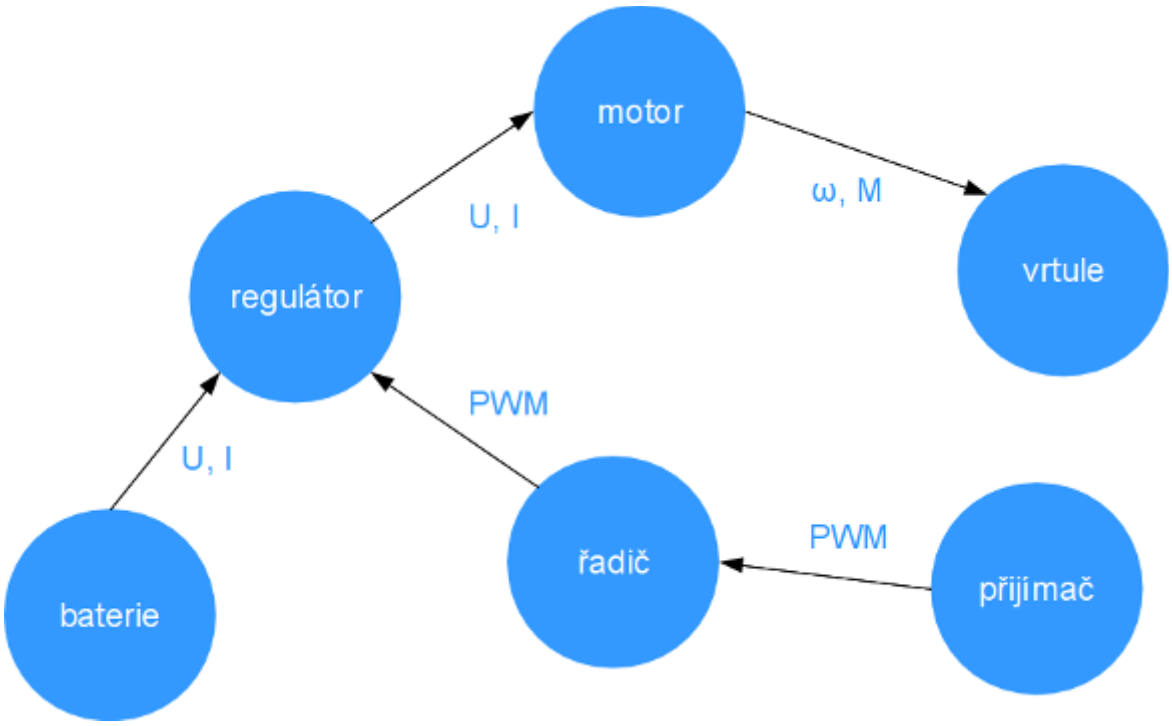
Pokud multikoptéra nemá potřebně rozvinutou plánovací úroveň je zapotřebí ji řídit dálkově. „Radio Control“ přijímač zpracovává elektromagnetický řídící signál (zpravidla o kmitočtu 2,4 GHz) a transformuje ho na signál modulovaný pulzně-šířkově. Tento je pak kompatibilní se vstupním signálem regulátoru.

Počet kanálů určuje počet signálů, který je schopen přijímač zpracovávat v jeden okamžik. Pulzy výstupního PWM signálu každého kanálu mají vzájemně jinou polohu v rámci, aby se daly všechny zpracovat řadičem pomocí jednoho časovače.

Existují i duplexní, takzvané telemetrické, systémy vysílač-přijímač. Kromě údajů ze senzorů pohybu můžou být schopné posílat směrem od robotu k operátorovi třeba i aktuální obraz zachycený videokamerou. Moderním přístupem je pak použití klasického Wi-Fi přijímače, díky kterému pak multikoptéra komunikuje s PC.

3 Fyzikální principy

Nyní popíši některé fyzikální fenomény, které se obecně týkají multirotorů. Na následujícím obrázku číslo 2 je schéma putování signálů respektive energie.



Obrázek č. 2: tok signálů a energie součástmi multirotoru

3.1 Elektrotechnická část

Do této části vstupuje baterie, regulátor a motor. Základní schéma výpočtů je jednoduché:

$$Pm = U_m * I_r \tag{2}$$

Kde Pm je výkon motoru, U_m efektivní napětí motoru a I_r proud v soustavě.

$$Pp = \omega * M \tag{3}$$

Kde Pp je výkon vrtule, ω úhlová rychlost a M točivý moment.

$$\omega = \frac{U * Kv * 2 * \pi}{60} \tag{4}$$

Kde Kv je počet otáček na volt a π Ludolfovo číslo.

Pro výpočet všech neznámých můžeme zvolit následující postup:

Mějme efektivní napětí motoru U_m, které je rovno napětí baterie

1. Vypočtěme úhlovou rychlost ω
2. Vypočtěme točivý moment M^*
3. Vypočtěme výkon vrtule Pp
4. Výkon motoru Pm položme roven výkonu vrtule
5. Vypočtěme proud I_r

* Točivý moment M je funkcí úhlové rychlosti ω a její derivace Δω.

Situace se komplikuje při přihlédnutí k vnitřním (armature) odporům a aplikaci Ohmova zákonu:

$$U_m = U_b - I_r * R_a \tag{5}$$

Kde U_m je napětí motoru, U_b napětí baterie, I_r proud protékající regulátorem a R_a je elektrický odpor soustavy.

$$U_r = U_b - I_r * (R_r + R_b) \tag{6}$$

Kde U_r je napětí na regulátoru, R_r odpor regulátoru a R_b odpor baterie.

$$R_a = R_m - R_r * R_b \tag{7}$$

Kde R_m je odpor motoru.

$$I_m = \frac{Pm}{U_m} \tag{8}$$

Kde I_m je efektivní proud motoru.

$$I_r = I_m + I_0 + \frac{U_0 - U_r}{Rk} \tag{9}$$

Kde I_0 je proud na prázdno, U_0 napětí proudu na prázdno a Rk motorová konstanta.

Bohužel při bližším pohledu zjistíme cyklickou závislost výpočtu neznámých [4]. Pro výpočet napětí na motoru potřebujeme znát proud protékající soustavou. Ten můžeme vypočítat pomocí napětí baterie odporů, ale i sebe samého a efektivního proudu motoru. Nejen proud protékající regulátorem, ale i efektivní proud motoru ještě nemáme k dispozici, protože ten se počítá pomocí napětí na motoru, což je naše výchozí neznámá.

To je důvod, proč se při modelování používá iterativní výpočet s omezenou přesností (*err*). Pseudokód takového výpočtu může vypadat následovně:

```
I_r = 0
do
    U_m = U_b - I_r * R_a
    U_r = U_b - I_r * (R_b + R_r)
    ω = U_m * Kv * (2 * π / 60)
    P = ω * M
    I_m = P / U_m
    I_r2 = I_m + (I_0 + U_0 - U_r / Rk)
    err = abs(I_r - I_r2)
    I_r = I_r2
while err > 0.001
```

Pseudokód č. 1: výpočet proudu pomocí smyčky

Abstrakce, které se při tomto takovémto popisu dopouštíme, postrádá další aspekty úhlové rychlosti, které ovlivňují točivý moment rotoru. Hlavním takovým aspektem je projev tření, tedy přesněji točivého momentu, který musí AC motor neustále vyvíjet, aby tření eliminoval. Při změně úhlové rychlosti se také v reálných systémech projevují důsledky změny setrvační energie, které vyžadují další točivý moment navíc.

3.2 Propulzní část

Tato sekce zahrnuje motor, vrtuli a v neposlední řadě také atmosferické prostředí stroje. Pro začátek uvedu výpočty používané v praxi RC modeláři:

$$Pp = \left(\frac{RPM}{1000}\right)^3 * Kp * D^4 * P \qquad (10)$$

Kde *Pp* je výkon vrtule, *Kp* je vrtulová konstanta, *RPM* počet otáček rotoru za minutu, *D* průměr vrtule [palce] a *P* náběh vrtule [palce].

$$RPM = \frac{\omega * 60}{2 * \pi} \qquad (11)$$

Kde ω je úhlová rychlost rotoru a π Ludolfovo číslo.

Pro orientační výpočty v modelářství jsou rovnice tohoto typu dostačující. Řešitelům ovšem nemusí přijít na chuť použité jednotky, natož zatemnění vlastní fyzikální podstaty. Veškeré rozměry jsou kvůli snaze udržovat vrtulovou konstantu kolem čísla 1 uváděny v palcích. Pokud by se dále pátralo po kompatibilní rovnici například pro výpočet tahové síly (thrust), byla by tato nejspíše uvedena v takzvaných uncích [oz] (1 / 16 libry). Točivý moment (torque) pak vychází v uncích-palcích [oz * in] a podobně.

Jak již bylo zmíněno, náběh vrtule určuje, jak moc jsou listy odkloněny od základní roviny vrtule. Jelikož není distribuce náběhu přes délku listu vrtule rovnoměrná, k výpočtům se používá sekce umístěná ve 3 / 4 od hřídele ke konečku. Počet palců udává zdvih, který by vrtule s náběhem dané hodnoty provedla v pevném tělesu (princip šroubu) při otočení o 360°. Pro přepočet na jednotku radián slouží následující vzorec:

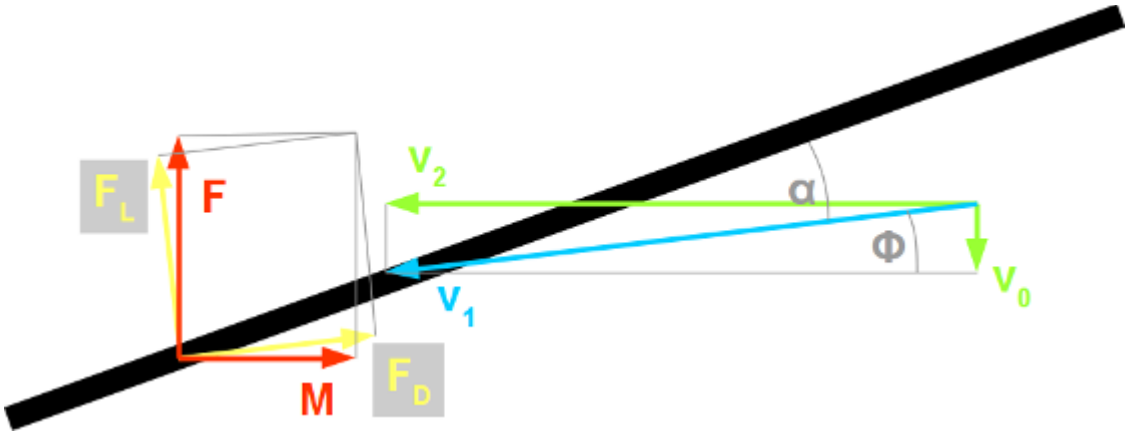
$$P_{[rad]} = \arctan\left(\frac{D * \pi * 3}{P_{[inch]} * 4}\right) \qquad (12)$$

Kde $P_{[rad]}$ je náběh v radiánech a $P_{[inch]}$ náběh v palcích.

3.2.1 Precizní model

Aby byl zachycen měnící se tvar vrtule, popíšu listy jako konečný (blade element theory [5]) počet sekcí o určité délce a šířce ve vzdálenosti od hřídele r s náběhem $P_{[rad]}$.

Pro každou takovouto sekci pak platí následující schéma:



Obrázek č. 3: sekce vrtule z boku – vztah rychlostí, vztlaku a odporu, tahu a momentu

Hlavní funkce vrtule je tvorba tahové síly, která je kolmá na základní rovinu. Lze tedy předpokládat, že v tomto směru bude stroj mít určitou dopřednou rychlost, v případě multikoptér pohyb vzhůru. Vrtule se pohybuje ve vzduchu, tedy tekutině. Tím pádem bude znatelné relativní proudění vzduchu o rychlosti v_0. Vrtule svým otáčením způsobuje, že na každé sekci je mimo v_0 znatelná rychlost v_2. To je rychlost relativního proudění vůči obvodové rychlosti sekce. Složením těchto dvou vektorů získáme výslednici v_1, což je rychlost proudění vzduchu kolem profilu dané sekce.

Z aerodynamických vlastností profilu a kolizního úhlu α (angle of attack) lze vypočítat koeficienty aerodynamického odporu a vztlaku. Způsob, jakým se koeficienty mění s α, je složitý zejména kvůli „stall“ efektu. Odpor F_D (drag) se projevuje jako síla působící proti vektoru rychlosti relativního proudění tekutiny. Vztlak F_L (lift) naopak působí silou ve směru kolmém na směr proudění. Výslednici odporové a vztlakové síle je třeba rozložit do směru základní roviny vrtule, respektive její normály. Takto se dostáváme k požadovanému točivému momentu M a k tahové síle F.

$$\Phi = P_{[rad]} - \alpha \qquad (13)$$

Kde Φ je úhel mezi základní rovinou vrtule a směrem proudění tekutiny (také ho svírá moment s odporovou silou a vztlaková síla se silou tahovou), α kolizní úhel profilu sekce a proudění tekutiny.

Pro tahovou sílu sekce a její točivý moment platí:

$$F_e = \frac{\rho * v_1^2 * (C_L * S_L * \cos(\Phi) - C_D * S_D * \sin(\Phi))}{2} \quad (14)$$

$$M_e = \frac{\rho * v_1^2 * (C_L * S_L * \sin(\Phi) - C_D * S_D * \cos(\Phi)) * r_e}{2} \quad (15)$$

Kde F_e je tahová síla, ρ hustota vzduchu, v_1 relativní rychlost proudící tekutiny, C_L vztlakový koeficient, S_L efektivní plocha pro vztlak, Φ úhel mezi v_1 a základní rovinou vrtule, C_D odporový koeficient, S_D efektivní plocha pro odpor, M_e točivý moment sekce a r_e vzdálenost sekce od hřídele.

Pro výpočet odpovídajících vektorů celé vrtule je třeba sečíst mezivýsledky pro všechny elementy a následně vynásobit počtem listů:

$$F = \sum_{e \in E} F_e * B \quad (16)$$

$$M = \sum_{e \in E} M_e * B \quad (17)$$

Kde F je celková tahová síla vrtule, E množina všech sekcí, M celkový točivý moment vrtule a B počet listů vrtule.

Výpočet celkového momentu vrtule zahrnuje pouze moment generovaný aerodynamickým odporem. Ve skutečnosti motor musí překonávat ještě tření na hřídeli a změnu setrvační energie při případné modifikaci otáček, což se projeví na celkovém reakčním momentu na rotoru.

Výpočet koeficientů se odvíjí nejen od kolizního úhlu, ale i od tvaru profilu sekce. Tomuto fenoménu se říká "aerofoil effect" (airfoil effect) a je důsledkem zákonu zachování energie popsaném Bernoulliho rovnicí.

Na místě je také podotknout, že uvedené vztahy nejsou vyčerpávající a zanedbávají například vliv "skin" efektu a "axial inflow" a "swirl" faktorů. Abstrahována byla také flexibilita vrtule (výpočty se týkají rigidních těles). Koneček vrtule kvadrokoptéry při plných otáčkách ovšem nedosáhne ani poloviny Machova čísla, a proto tyto vlivy nemají podstatnou váhu.

3.3 Mechanická část

V části propulzní jsem představil rozbor vzniku působících sil. Způsob jakým síly vyvinuté na ramena rozpohybují multikoptéru popisuje klasická mechanika. Základem jsou Newtonovy zákony a mechanické energetické ekvilibrium.

Pro vyrovnané síly všech rotorů je rovnice jednoduchá:

$$a=(F_l+F_f+F_r+F_b)*m \tag{18}$$

Kde a je zrychlení kvadrokoptéry v ose z, F_l síla působící na levé rameno, F_f síla působící na přední rameno, F_r síla působící na pravé rameno, F_b síla působící na zadní rameno a m je hmotnost kvadrokoptéry.

Kromě pohybu vzhůru může kvadrokoptéra vykonávat i rotační pohyb. Ten vzniká nerovnováhou sil působících na vzájemně protější ramena. Střed otáčení, tedy těžiště, je v ideálním případě střed kříže rámu.

Pokud je jedna síla z antagonistické dvojice menší a druhá větší, je situace ohledně výpočtu složitější. Musíme se na dvojice protichůdných sil dívat také z pohledu točivého momentu. Rovnice se složením všech sil v tah je jen speciálním případem přerozdělení. Opačným extrémem je, že všechny protichůdné síly budou skutečně působit přesně opačným směrem a stejnou velikostí (skalárně – záporná hodnota se stejnou magnitudou). Výsledkem je nulová tahová síla a samostatná rovnice pro výpočet úhlové rychlosti. Proto přichází na řadu jeden z nejzákladnějších přírodních jevů – rovnováha rozdělení energií. V rovnicích zacházím pro ilustraci s dvojicí levým a pravým ramenem:

$$M_x=F_l*(-r)+F_r*r \tag{19}$$

$$F_{lr}=F_l+F_r \tag{20}$$

$$w_{Flr}*E_{Mx}=w_{Mx}*E_{Flr} \tag{21}$$

Kde M_x je točivý moment kolem osy x, r vzdálenost rotorů od těžiště kvadrokoptéry, F_{lr} tahová síla dvojice, w_{Flr} míra zastoupení tahové síly, E_{Mx} energie setrvačnosti, w_{Mx} míra zastoupení točivého momentu a E_{Flr} kinetická energie.

$$\frac{w_{Flr} * J_x * \omega_x^2}{2} = \frac{w_{Mx} * m * v^2}{2} \tag{22}$$

$$\omega = \frac{M * t}{J} \tag{23}$$

$$v = \frac{F * t}{m} \tag{24}$$

$$\frac{w_{Flr} * M_x^2}{2 * J_x} = \frac{w_{Mx} * F_{lr}^2}{2 * m} \tag{25}$$

$$\frac{w_{Flr} * M_x^2}{J_x} = \frac{(1 - w_{Mx}) * F_{lr}^2}{m}$$

$$\frac{w_{Flr} * M_x^2}{J_x} + \frac{w_{Flr} * F_{lr}^2}{m} = \frac{F_{lr}^2}{m}$$

$$w_{Flr} * \left(\frac{M_x^2}{J_x} + \frac{F_{lr}^2}{m} \right) = \frac{F_{lr}^2}{m}$$

$$w_{Flr} = \frac{\frac{F_{lr}^2}{m}}{\left(\frac{M_x^2}{J_x} + \frac{F_{lr}^2}{m} \right)} \tag{26}$$

$$w_{Mx} = \frac{\frac{M_x^2}{J_x}}{\left(\frac{M_x^2}{J_x} + \frac{F_{lr}^2}{m} \right)} \tag{27}$$

Kde J_x je moment setrvačnosti pro danou osu otáčení, ω_x úhlová rychlost kvadrokoptéry, v rychlost vzhůru a t je čas.

Přerozdělení sil pro tah a natáčení bude w_{Flr} ku w_{Mx}, což odpovídá poměru energetické náročnosti.

Pro získání celkové tahové síly v ose z stačí sečíst váhované tahové síly obou antagonistických dvojic. Váhované točivé momenty jsou složkami x a y celkového vektoru točivého momentu. Poslední jeho složkou je moment otáčení kolem osy z, u něhož výpočty vah odpadají. Z propulzního algoritmu totiž dostáváme rovnou točivé momenty jednotlivých rotorů. Jako takové se stejným způsobem projevují po celé délce ramen a celkové působení se spočítá prostým sečtením s ohledem na znaménko.

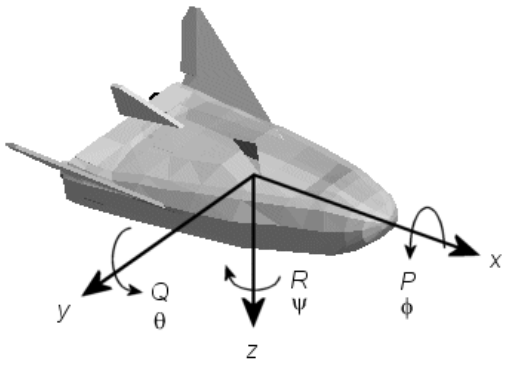
$$M_{xyz} = (w_{Mx} * M_x ; w_{My} * M_y ; M_{zl} + M_{zf} + M_{zr} + M_{zb}) \quad (28)$$

Kde M_{xyz} je celkový vektor točivého momentu a M_{zl} až M_{zb} jsou točivé momenty v ose z vzniklé po řadě na levém, předním, pravým a zadním rotoru.

$$F_{xyz} = (0; 0; F_{lr} + F_{fb}) \quad (29)$$

Kde F_{xyz} je celkový vektor tahové síly.

Souřadný systém používaný pro modelování mobilních robotů, které si sami generují lokomoční podněty, je vztažen relativně k tělu daného stroje. Jde o takzvaný body-fixed coordinate frame (dále BF). To například znamená, že po otočení kvadrokoptéry vzhůru přistávací konstrukcí bude tahová síla působit stále původní orientací po ose z. Souřadný systém pozorovatele může být například flat-earth (dále FE).



Obrázek č. 4: body-fixed souřadný systém [ob1]

Pro výpočet rychlostí jsem již zmínil obecné vzorce (23) a (24). Nyní se ale jedná o vektorový výpočet.

Moment setrvačnosti důležitým parametrem, protože natáčení je základem pohybu multikoptér. Udává v podstatě způsob rozložení hmotnosti otáčeného tělesa kolem dané osy. V trojrozměrném prostoru se pracuje s maticovou reprezentací tenzorů setrvačnosti. Matice 3 x 3 obsahující v diagonále momenty setrvačnosti vzhledem ke všem osám. Vzorce výpočtu nejsou triviální ani pro relativně jednoduché tvary. Pro popis strojů se tedy využívá skládání matic jednodušších komponent. Kompozice probíhá sčítáním matic a translační transformací dle Steinerova teorému paralelní osy takto:

$$J_1 = J_0 + m * d^2 \quad (30)$$

Kde J_1 je transformovaný moment setrvačnosti, J_0 originální moment setrvačnosti, m hmotnost tělesa a d je změna vzdálenosti od dané osy otáčení (zde pouze oddálení).

3.4 Aerodynamická část

Nejsložitější částí popisu fyzikálních jevů působících na multirotory je aerodynamika. Aerodynamický odpor je silové působení na zpravidla tuhé těleso pohybující se ve vzduchu (v tekutém prostředí). Molekuly vzduchu svým relativním kinetickým stavem brání v prostupování letounu. Multikoptéry se zpravidla nepohybují velkými rychlostmi (jen pár kusů na světě zvládá rychlost kolem 100 km / hod), tudíž se neklade důraz na návrh aerodynamické karoserie.

V propulzní části jsem již zmínil výpočet aerodynamického odporu (drag), jeho obecná podoba vypadá takto:

$$F_D = \frac{C_D * \rho * v^2 * S}{2} \qquad (31)$$

Kde F_D je odporová síla působící proti orientaci pohybu tělesa, C_D koeficient odporu, ρ hustota vzduchu, v relativní rychlost proudění a S plocha průmětu tělesa na plochu kolmou ke směru šíření.

Odpor není jediným faktorem ovlivňujícím let. Kromě další, již zmíněné vztlakové, síly vzniká aerodynamickou interakcí i ubočovací síla a točivé momenty kolem všech možných os.

Bohužel u složitějších objektů je výpočet aerodynamických koeficientů velice komplexní, protože se zde nedá uplatnit žádný princip skládání z objektů jednodušších. Koeficienty, jako je C_D, nejsou konstantní za všech okolností. Jsou funkcí takzvaných Reynoldsových čísel, která vyjadřují poměr inerciálních a viskózních sil v tekutině. Ve svém textu předpokládám Reynoldsova čísla pro běžně pozorované jevy, tedy $Re = 10^5$. Jak již bylo zmíněno, aerodynamické koeficienty se také mění s kolizním úhlem.

Toto jsou hlavní důvody, proč je přesný výpočet složitý i pro komplexní simulační nástroje, jako je například Foilsim III [od1].

4 Řízení

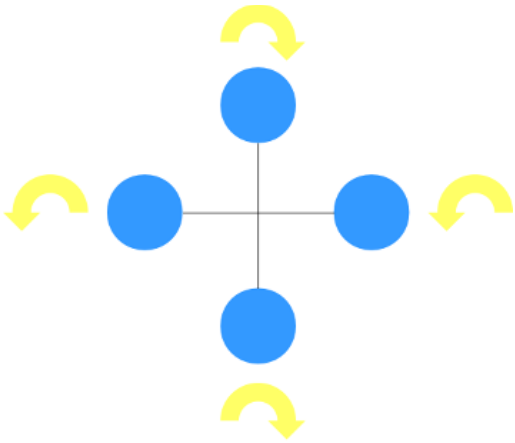
4.1 Bazální pohyby

Pokud multikoptéra nevykonává žádný pohyb, říkáme, že je ve visu (hover).

Následující schémata kvadrokoptéry popisují nejobvyklejší model řízení z pohledu shora. Osa x vede odspodu obrázku nahoru, y zleva doprava a osa z prochází kolmo středem kříže. Nechť je zavedena následující notace: Žlutě zbarvené šipky označují vrtule středně vysokých otáček, červené šipky vrtule nízkých otáček a zelené šipky označují vrtule s relativně vysokými otáčkami.

4.1.1 Pohyb po ose *z*

Jako první pohyb zmíním pohyb po ose z. Na obrázku číslo 5 vidíme v jakých smyslech se vrtule kvadrokotpéry otáčejí. Tahová síla celého stroje se v ose z řídí otáčkami všech motorů. Pokud chceme zvyšovat rychlost vzhůru (k mínus nekonečnu), zvýšíme náležitě otáčky všech motorů stejnou měrou. Pokud chceme rychlost orientovat dolů (k plus nekonečnu), využije se gravitační síly, která převýší nad silou tahovou generovanou náležitě zmírněnými otáčkami motorů.

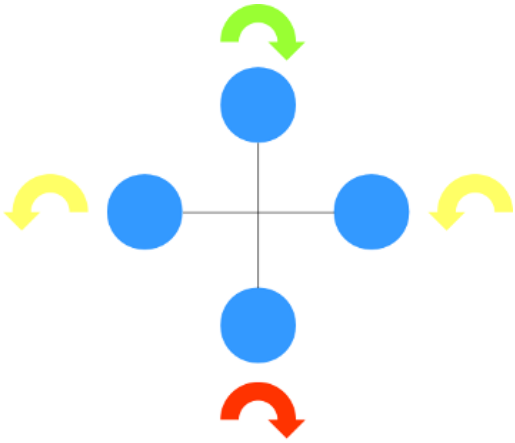


Obrázek č. 5: smysly otáčení vrtulí kvadrokoptéry (stabilní poloha)

4.1.2 Sklon

Na obrázku číslo 6 můžeme vidět, že tahová síla přední vrtule převažuje nad postranními a naopak na zadní vrtuli tahová síla oproti postranním strádá. Při dekompozici na základní řídící povely je obvyklé se snažit, aby úbytek síly na "slabší" vrtuli byl stejný jako nadbytek na "silnější".

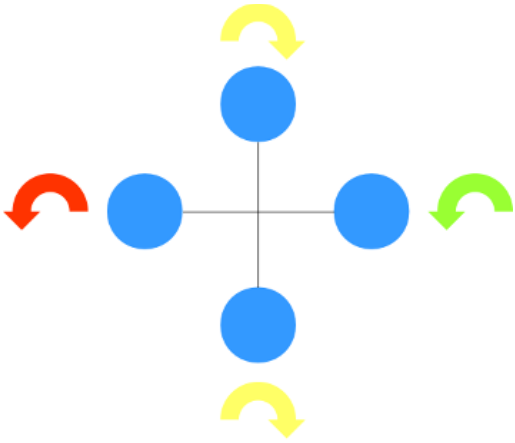
Působením sil vzniká rotační pohyb kolem osy *y* zvaný sklon (pitch). Smysl otáčení je "slabší" vrtule dolů a "silnější" vzhůru po ose *z* (body-fixed souřadnice). Když se celý stroj vychýlí z vodorovné polohy, tak síly, které ve původně působily čistě vzhůru, nyní působí i směrem proti ose *x* (obecné souřadnice). Tímto směrem tedy vzniká adekvátní zrychlení.



Obrázek č. 6: sklon

4.1.3 Náklon

Obrázek číslo 7 popisuje stejný princip, jako je sklon, tentokrát ovšem pro otáčení kolem osy *x*. Tento pohyb se nazývá náklon (roll).



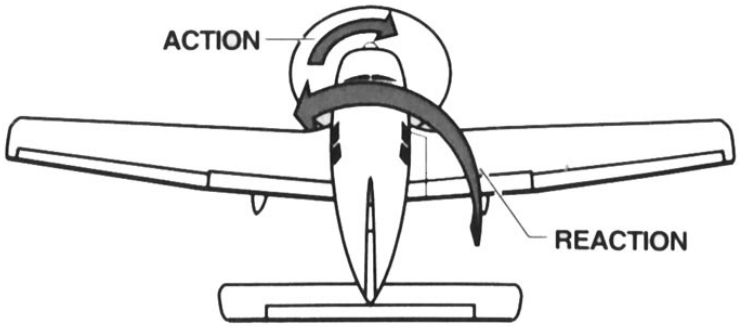
Obrázek č. 7: náklon

Schémata z obrázků 6 a 7 se týkají takzvaného zapojení "+". Mírně složitější je varianta zapojení "X". Zde se u sklonu a náklonu místo jedněch určí dva „silnější“ a dva „slabší“ rotory. Osy otáčení potom neprochází skrze rotory, ale mezi nimi. Modifikace otáček se pak u všech bazálních pohybů týká všech vrtulí.

4.1.4 Torque efekt

Torque efekt je fenomén známý již z problematiky řízení jednomotorových vrtulových letadel. Motor totiž neotáčí pouze vrtulí, ale díky reakčnímu točivému momentu (torque) i letadlem. Rotace celého stroje se díky mnohem větší hmotnosti neprojevuje tak markantně. V době plně manuálního ovládání ovšem musel pilot s tímto jevem počítat. Ze stejného důvodu má helikoptéra v činnosti zadní rotor, i když nevykonává točivý pohyb kolem osy *z*.

Stejně tak se reakční točivý moment projevuje na otáčení kvadrokoptéry. To je důvod, proč se všechny vrtule neotáčí ve stejném smyslu. Přední a zadní vrtule se otáčí po směru hodinových ručiček (Clock Wise) a levá s pravou zase proti směru hodinových ručiček (Counter Clock Wise). Nebo naopak Pokud mají otáčky stejnou velikost, reakční točivé momenty všech vrtulí se vzájemně eliminují a stroj zůstává stabilizovaný.

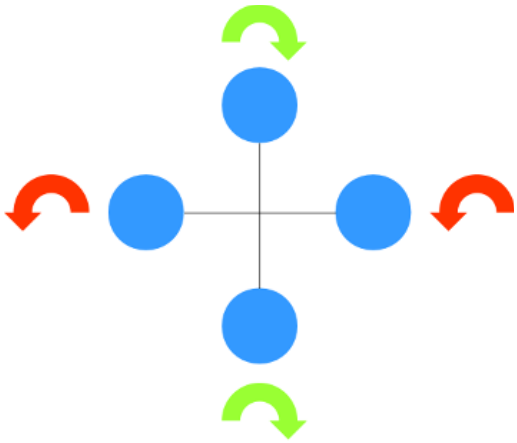


Obrázek č. 8: torque efekt [ob2]

4.1.5 Zatočení

Torque efekt ale není jen na škodu. Díky němu jsme schopni otáčet multikoptéru kolem osy *z*, tedy provádět bazální pohyb zatočení (yaw). Z obrázku číslo 7 lze vyčíst, že postranní motory poskytují méně otáček než přední a zadní. Pro zachování rovnováhy v ostatních smyslech a směrech je zapotřebí, aby úbytky tahových sil na obou "slabších" motorech byly vzájemně stejné a stejně velké jako nadbytky sil "silnějšího" páru. Doposud jsem popisoval vliv tahových sil na naklánění a sklánění. V tomto případě je situace podobná, jelikož velikosti odporových sil generujících reakční točivý moment se stejně jako tahové síly zvyšují s otáčkami. Smysl otáčení stroje je ovšem závislý na tom, v jakém smyslu se otáčí "silnější" polovina motorů.

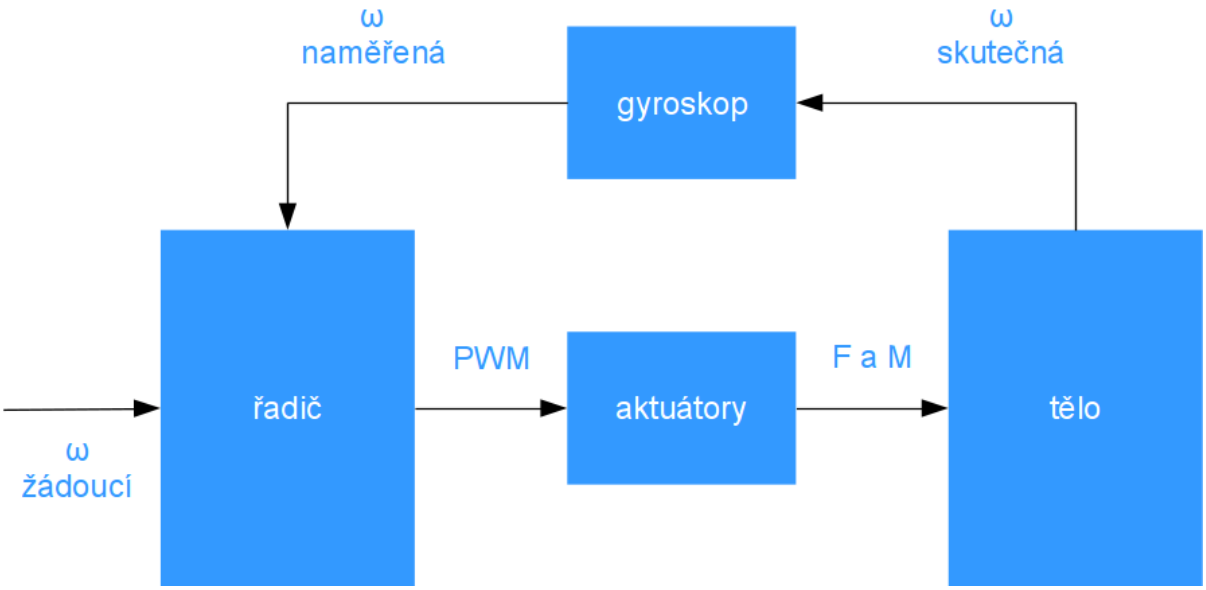
Na obrázku číslo 8 se rychleji pohybují motory po směru hodinových ručiček. Reakční točivý moment bude působit opačně, a tak se kvadrokoptéra bude otáčet proti směru hodinových ručiček.



Obrázek č. 9: zatočení

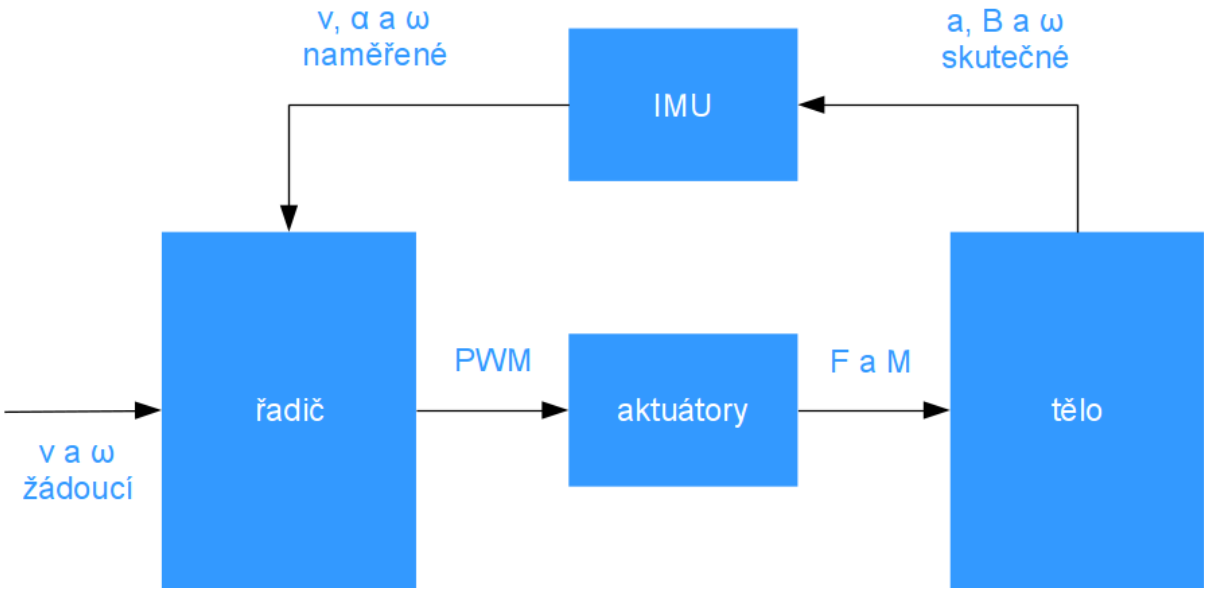
4.2 Řídící smyčka

Po zjištění vztahu mezi působícími silami a základními pohyby kvadrokoptéry je na čase se zaobírat otázkou, jak velké má dané působení být. Tedy například pokud chceme zvýšit rychlost stoupání na 10 m / s, jak velké napětí má regulátor do motoru přivést. Konkrétní odpověď nelze kvůli množství neznámých vlastností prostředí předpovídat analyticky. Mezi hlavní negativní faktory patří výrobní nepřesnosti elektrotechnických součástek, pružnost vrtulí, či nepodchycené atmosferické podmínky. Kvadrokoptéry mají zpravidla lidského operátora, který má relativně vysokou schopnost anticipace a intuice. Děje probíhající v robotickém systému jsou ale příliš rychlé na to, aby je dokázal přesně vyhodnotit. Proto na řadu přichází uzavřená řídící smyčka. Minimální koncept smyčky kvadrokoptéry je popsán následujícím schématem:



Obrázek č. 10: základní řídící smyčka

V tomto pohledu již vnímáme senzorickou jednotku jako samostatnou část, kterou nyní představuje gyroskop. Gyroskop je senzor úhlové rychlosti. Snímá rychlost otáčení multikoptéry. Pokud se stroj naklání, sklání nebo zatáčí příliš rychle nebo příliš pomalu, díky gyroskopu je řadič včas informován. Řadič poté může patřičně upravit signál pro aktuátory a "čekat" na reakci systému, tedy další vstup z gyroskopu.



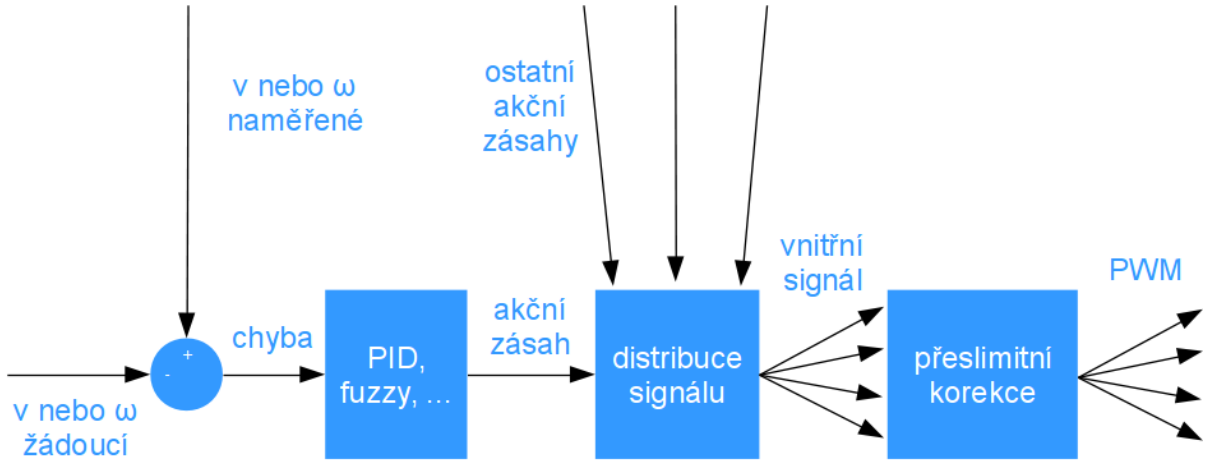
Obrázek č. 11: pokročilá řídící smyčka

Uvedené schéma znázorňuje pokročilejší variantu řízení stabilizace multikoptér. Senzorický blok obsahuje inerciální měřící jednotku (Inertial Measurement Unit), která na rozdíl od gyroskopu poskytuje stavové informace v 6 stupních volnosti (gyroskop poskytuje pouze 3). Jedná se o multisenzor nejčastěji snímající akceleraci, magnetickou indukci a úhlovou rychlost. Proto se tato jednotka často označuje jako devítiosá. Díky predikčním algoritmům a Kalmanově filtru získáváme z IMU navíc informace o rychlosti a aktuálních úhlech natočení multikoptéry.

Hlavním problémem, který IMU pomocí predikce řeší, je přítomnost gravitace v pozemských podmínkách. Přesněji tíhová síla ovlivňuje zrychlení naměřené akcelerometrem a tím pádem i predikovanou rychlost. Díky naměřeným hodnotám z gyroskopu a z magnetometru je ovšem IMU schopná odhadnout, jakým relativním vektorem aktuálně tíhová síla působí a následně jej kompenzovat. Magnetometr (kompas) má podobný problém, protože siločáry magnetické indukce poskytují informaci o natočení pouze ve 2 stupních volnosti. Predikce se tedy aplikuje i opačným směrem ke zpřesnění výstupu magnetometru.

Hlavní rozdíl mezi základní a pokročilou řídící smyčkou je oblast použití. V dopravě či při pořizování videozáznamu uvítáme přesnou kontrolu nad rychlostí. Naopak v akrobatickém letectví je výhodnější mít větší možnosti manévrování bez ohledu na stabilní rychlost.

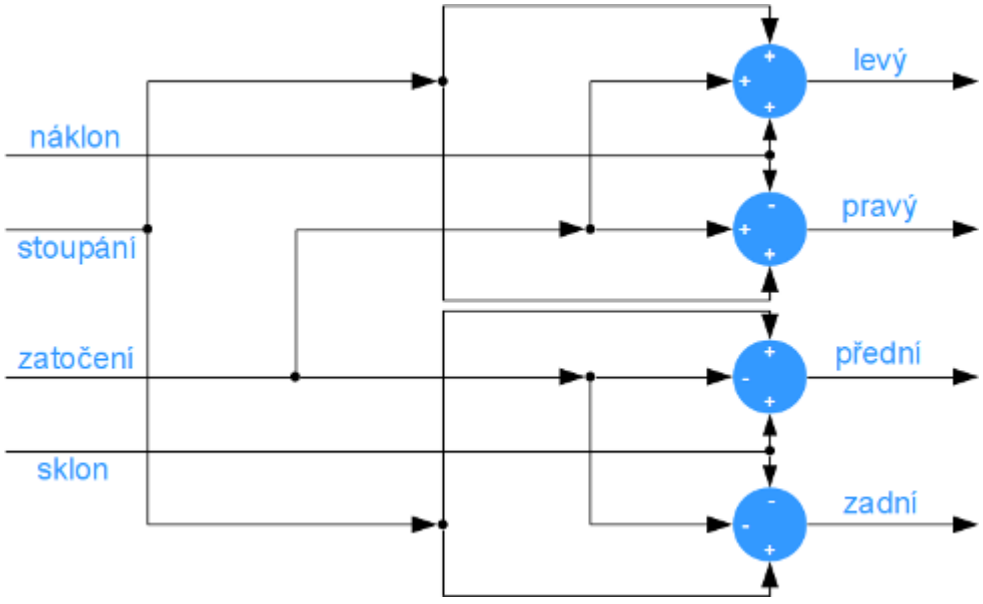
Práci řadiče popisuje následující diagram:



Obrázek č. 12: řadič

Do řadiče vstupuje požadovaná hodnota rychlosti respektive úhlové rychlosti a odhadovaná rychlost respektive úhlová rychlost ze senzorů. Dle velikosti rozdílu žádoucí a naměřené hodnoty se dalšími regulačními metodami získá hodnota akčního zásahu. Nejčastější metodou regulace je použití PID regulátorů. Alternativami jsou LQR, H-∞, LQG (Linear Quadratic Gaussian), SMC (Sliding Mode Controller), fuzzy regulátory či regulátory založené na neuronových sítích.

Akční zásah zde reprezentuje stále jen povel v podobě jedné skalární hodnoty. Takovéto povely se musí nasměrovat k příslušným regulátorům rychlosti motoru pomocí signálové distribuční jednotky.



Obrázek č. 13: signálová distribuční jednotka

Ještě před výstupem z řadiče se distribuované signály musí případně upravit tak, aby nepřekročily meze dané rozsahem vstupního řídícího signálu rychlostního regulátoru motoru. Nejedná se ovšem pouze o takzvané oseknutí. Při distribuci signálu je důležité, aby se zachoval rozdíl mezi akčními povely pro antagonistické aktuátory, protože stabilita otáčení má přednost před rychlostí stoupání.

Příklad řízení:

1. kvadrokoptéra je ve vodorovném visu (hover), který vytíží motory na 100% (pro ukázku)
2. z přijímače se do řadiče dostane povel velkého náklánění doprava → $\Delta\omega = 1$
3. gyroskop zaznamenává nezaznamenává žádnou úhlová rychlost → $\Delta\omega = 0$
4. rozdíl požadované a naměřené hodnoty je 1
5. PID regulátor s koeficienty 0,1; 0,1; 0 vyhodnocuje velikost akčního zásahu na 0,2
6. distribuce signálů zajistí, že se do kanálu pro regulátor levého motoru přičte hodnotu 0,1 a z kanálu pro regulátor pravého motoru 0,1 odečte
7. motory již jsou vytížené na 100% a není tedy možné zvýšit výkon levého motoru. Korekce limitů zajistí, že se od všech kanálů odečte rozdíl hodnoty přesahující řídící mezi a dané meze → v "levém" kanálu je připravena hodnota 1,1, a tak se ze všech kanálů odečte 0,1. Maximální výkon tedy bude podávat jen levý motor a kvadrokoptéra se kromě náklonu doprava pořítí směrem doprava dolů

4.3 Regulace

4.3.1 PID regulátor

PID regulátor je jednoduchá komponenta schopná zachytit průběh regulační odchylky, její kumulovanou hodnotu a změnu v čase. Integrace a derivace si regulátor většinou počítá vnitřně, případně tyto hodnoty obdrží již předpočítané. V praxi jsou pro integraci respektive derivaci přesnější pojmy sumace a diference, protože se hodnoty ze senzorů měří v diskrétním čase.

Velikost odchylky se násobí koeficientem *P* (proporcionální část), kumulovaná hodnota se násobí koeficientem *I* (integrační část) a diference se násobí koeficientem *D* (derivační část). Součtem jednotlivých součinů získáme velikost akčního zásahu:

$$u_t = P * E_t + I * \sum_{i=0}^{t} E_i + D * (E_t - E_{t-1}) \tag{32}$$

Kde *u* je velikost akčního zásahu, *P* proporcionální koeficient, *E* velikost odchylky, *I* integrační koeficient a *D* derivační koeficient.

Derivační složka je velice citlivá na šum, který vzniká ve všech senzorech. Její velikost pak musí být podrobena další filtraci nebo se minimalizuje její vliv.

Výpočet parametrů může být proveden Ziegler-Nicholsovým algoritmem:

1. nastavíme *I = D = 0*
2. postupně od 0 zvyšujeme parametr *P* do doby, kdy odezva systému začne mít podobu netlumených kmitů – periodu pojmenujme *T*
3. nastavíme parametry:
   a) pro PI regulátor: *P = P / 2,2; I = T / 1,2*
   b) pro PID regulátor: *P = P / 1,7; I = T / 2; D = T / 8*

4.3.2 LQR

Lineární kvadratická regulace (Lienar Quadratic Regulation) je způsob nalezení jistého vektoru zisku. Následující vzorec reprezentuje chování regulovaného systému (regulačním cílem je, aby x^{∞} byl nulový vektor):

$$x_{t+1} = A * x_t + B * u_t \tag{33}$$

Kde x_{t+1} je vektor následujícího stavu, *A* matice popisující implicitní změnu stavu, x_t vektor aktuálního stavu, *B* matice popisující vliv akčního zásahu na systém a *u* vektor akčního zásahu.

$$u_t = -K * x_t \tag{34}$$

$$x_{t+1} = A * x_t + B * (-K * x_t) \tag{35}$$

$$x_{t+1} = (A - K * B) * x_t \tag{36}$$

Kde *K* je vektor zesílení.

Takovýto popis je zobecněním PID regulace. Vektor zesílení K může reprezentovat trojici koeficientů PID v případě, že stavový vektor x obsahuje složky: regulační odchylka, kumulovaná regulační odchylka a změna regulační odchylky. Použití LQR ale může být mnohem komplexnější, například stavový vektor x může popisovat vertikální pozici, vertikální rychlost a úhlové rychlosti každého motoru [8].

Princip LQR je založený na výpočtu vektoru K:

$$K = R^{-1} * B^T * P \tag{37}$$

Kde R je penalizační matice akčních zásahů a P pomocná matice, která se spočítá řešením algebraické Riccotovy rovnice:

$$A^T * P + P * A - P * B * R^{-1} * B^T * P + Q = 0 \tag{38}$$

Kde Q je penalizační matice stavových veličin.

Tato rovnice má řešení v podobě P, pokud Q a R jsou pozitivně definitní. Řešení nám poskytuje matici K. Ta jakožto parametr řízení zajišťuje, že regulace bude stabilní a minimální. Minimálnost znamená, že za minimálních příkonů poskytuje minimální regulační chybu vzhledem k daným penalizačním parametrům:

$$J = \sum_{i=0}^{n} \left(x_i^T * Q * x_i + u_i^T * R * u_i \right) \tag{39}$$

Kde J je hodnotící tedy minimalizovaný skalár.

Minimalizace zajistí optimální poměr mezi požadovaným chováním systému a energetickými nároky akčních zásahů. Q a R matice penalizují vektor stavových veličin respektive vektor stavových zásahů. Hodnoty na jejich hlavní diagonále jsou váhy jednotlivých složek penalizovaných vektorů, mimo diagonálu jsou váhy jejich kombinací. Čím méně chceme, aby se stavová veličina či zásah se projevovaly, tím větší váhu jim přiřadíme. Takto se dá ovlivnit například velikost překmitu regulované veličiny, nebo velikost potřebného příkonu řídícího systému.

Nevýhodou LQR metody je fakt, že Riccotova rovnice vychází z popisu lineárního systému. Stejně jako PID regulátor i LQR lze použít na nelineární systémy. Optimalizační výpočet pomocné matice P se ovšem musí provádět nad lineární abstrakcí.

4.3.3 H-∞

H–infinity opět používá k popisu problému lineární formulaci:

$$\begin{bmatrix} z \\ v \end{bmatrix} = \begin{bmatrix} P_{11} P_{12} \\ P_{21} P_{22} \end{bmatrix} * \begin{bmatrix} w \\ u \end{bmatrix} \tag{40}$$

Kde *z* je výstupní vektor systému, *v* výstupní vektor tvořící uzavřenou smyčku, členy P_{xy} jsou matice popisující chování systému, *w* vnější vlivy a *u* akční zásah.

$$u = K * v \tag{41}$$

$$z = F_l(P; K) * w \tag{42}$$

Kde *K* je matice regulátoru a F_l je stabilní přenosová funkce.

$$F_l(P; K) = P_{11} + P_{12} * K * (I - P_{22} * K)^{-1} * P_{21} \tag{43}$$

Kde *I* je jednotková matice.

$$\|F_l(P; K)\|_\infty = \sup_{\omega \in R} \bar{\sigma}(F_l(P; K) * (j * \omega)) \tag{44}$$

Kde $||Fl(P; K)||_\infty$ je norma stabilní přenosové funkce (také H_∞ norma), *sup σ* vrací největší singulární číslo, $j * \omega$ je frekvenční zesílení.

H–infinity je metoda, která již ve své podstatě počítá s neurčitostí existující ve všech reálných systémech. Norma stabilní přenosové funkce vyjadřuje maximální zesílení přes všechny frekvence $\omega \in R$. Člen $j * \omega$ zastupuje právě onu neurčitost. Výpočet H–infinity hledá takové *K*, jehož norma stabilní přenosové funkce je minimální. Numerické řešení využívá vlastnosti singularity matice Hamiltoniánu a metodu půlení intervalu.

4.3.4 Fuzzy regulátor

Fuzzy regulátory jsou založeny na fuzzy množinách. Ty jsou zobecněním běžných množin v tom smyslu, že příslušnost prvku není jen „patří-nepatří“, ale „patří-na-kolik“. Fuzzy množina nad daným univerzem má tedy takzvanou funkci příslušnosti. To je křivka s oborem hodnot 0 až 1 včetně, která k danému číslu univerza určí příslušnost do fuzzy množiny.

V regulaci jsou univerza měřené veličiny a veličiny akčních zásahů. Nad daným univerzem pak existují takzvané lingvistické proměnné, což jsou sady fuzzy množin. Nejčastěji používané fuzzy množiny jedné lingvistické proměnné jsou L (low), M (medium) a H (high), nebo N (negative), Z (zero), P (positive). Jejich funkcí příslušnosti tvary mají často trojúhelníkový nebo lichoběžníkový charakter.

Fuzzy regulace má 3 hlavní fáze:

Fuzzifikace – při fuzzifikaci se vážené regulační odchylce a její vážené změně určí příslušnost do fuzzy množin svých lingvistických proměnných P respektive D. Lingvistická proměnná akčního zásahu nechť je K.

Fuzzy inference – ke každé dvojici fuzzy množin z P x D je přiřazena fuzzy množina z K takzvaným pravidlem. Při použití Mamdaniho principu se síla tohoto pravidla určí jako maximum nebo minimum příslušností vstupů do vstupních fuzzy množin. Síla pravidla určí, jak moc se projeví daná fuzzy množina akčního zásahu ve výsledné agregaci. Agregace všech fuzzy množin z K probíhá operací sjednocení nad fuzzy množinami. Výsledkem této operace je jedna fuzzy množina.

Defuzzifikace – poslední fází je defuzzifikace, tedy převod fuzzy množiny na skalární akční zásah. Nejčastější metody jsou metoda těžiště (COG), kde se vypočtené těžiště plochy pod funkcí příslušnosti promítne na definiční obor – univerzum.

Popsané schéma je obdobou PD regulátoru. Pokud chceme vytvořit obdobu PID, musíme výsledek PD fuzzy regulátoru sečíst s výsledkem fuzzy regulátoru PI. Ten má stejné vstupy jako PD fuzzy regulátor, tedy váženou regulační odchylku a váženou změnu regulační odchylky. Vypočtený akční zásah je ale sumován a až jeho kumulovaná hodnota je jeho výstupem.

Parametry fuzzy regulátoru tedy jsou počet a tvar fuzzy množin, váhy velikosti odchylky a její změny a zesílení výstupního akčního zásahu. Konkrétní nastavení se většinou volí experimentálně, případně se váhy a zesílení dají přepočítat z parametrů již vyladěného PID regulátoru.

4.3.5 Neuronové sítě

Umělé neuronové sítě se obecně používají tam, kde potřebujeme nalézt řešení nelineárního problému, který je příliš komplikovaný na matematický popis. Při použití této metody se nemusíme zajímat o to, jak regulovaný systém popsat, protože se síť sama naučí správné fungování.

Základní jednotkou sítě je neuron s libovolným počtem vstupů a jedním výstupem. Jako zvláštní vstup se používá číslo 1 jako pevný bod pro prahování. Konfigurace neuronu je daná vektorem vah vstupů. Výpočet probíhá zpravidla nad reálnými čísly tak, že se nad vstupním vektorem a vektorem vah provede skalární součin, který vstupuje do takzvané aktivační

funkce. Ta má, co se regulace týče, nejčastěji podobu sigmoidy, popřípadě je lineární pro výstupní neurony. Aktivační funkce vrací již konečný výsledek neuronu.

Neurony se řadí do vrstev, které dělíme na jednu vstupní, jednu výstupní a libovolný počet skrytých. Počet a šířka skrytých vrstev vzrůstá se složitostí řešené úlohy. Vstupní vrstva odpovídá vstupnímu vektoru sítě. Výstupy neuronů ve výstupní vrstvě zase reprezentují vektor výstupní.

Neuronové sítě se učí na dostatečném počtu trénovacích příkladů – dvojic vstup a správný výsledek. Pro učení sítě se používá takzvaný backpropagation algoritmus. Princip spočívá ve zpětném šíření chyby skrz síť z výstupní vrstvy, kde byla výsledná chyba sítě zjištěna a kvantifikována. Při této propagaci chyby se vhodně upravují váhové vektory jednotlivých neuronů.

Pro učení schopnosti regulace je vhodné použít dvě sítě. První síť je hodnotící a predikuje hodnocení daného stavu, jelikož je stavový prostor spojitý. Druhá síť, akční, slouží k výpočtu optimálního akčního zásahu. Neuronů ve skrytých vrstvách jsou zhruba desítky. Akční síť se nejprve naučí simulovat klasický již vyladěný PID regulátor. Další zpřesňování regulačního chování se dá provádět takzvaným TD (temporal difference) učením. Jelikož se jedná o výpočetně náročnou metodu, probíhá algoritmus zpravidla na simulačním modelu (Simulation Based Learning). Krok učení v širším slovasmyslu pak může vypadat takto:

1. akční i hodnotící síť provedou výpočet nad vstupními daty kroku. Vstupy si zapamatujeme jako $input_t$. Výsledek hodnotící sítě nazveme odhad hodnocení aktuálního stavu $V(s_t)$, výsledek akční sítě *action*
2. s určitou pravděpodobností pozměníme *action* o relativně malé číslo, kdyby náhodou existovala poblíž ještě vhodnější hodnota, a získáme tak $action_+$ (fáze objevování)
3. provede se krok simulace a získáme tak vstupní vektor $input_{t+1}$
4. hodnotící funkce vypočte takzvanou odměnu r_{t+1}. Čím lépe přispěl zásah k regulaci, tím větší odměna
5. hodnotící síť odhadne ze vstupů $input_{t+1}$ hodnocení nového stavu $V(s_{t+1})$
6. vypočte se zlepšení: $td = r_{t+1} + \gamma * V(s_{t+1}) - V(s_t)$
7. hodnotící síť se natrénuje nad vstupem $input_t$ a výstupem $r_{t+1} + \gamma * V(s_{t+1})$
8. pokud je změna k lepšímu *td* „dostatečná“, natrénujeme akční síť nad vstupem $input_t$ a výstupem $action_+$

γ je takzvaný „discount factor“, který definuje vztah důležitosti přítomných a budoucích stavů. Pro skutečně kvalitní natrénování musíme neuronové sítě podrobit tisícům až desítkám tisíc stabilizačních průběhů simulace.

4.4 Kalmanův filtr

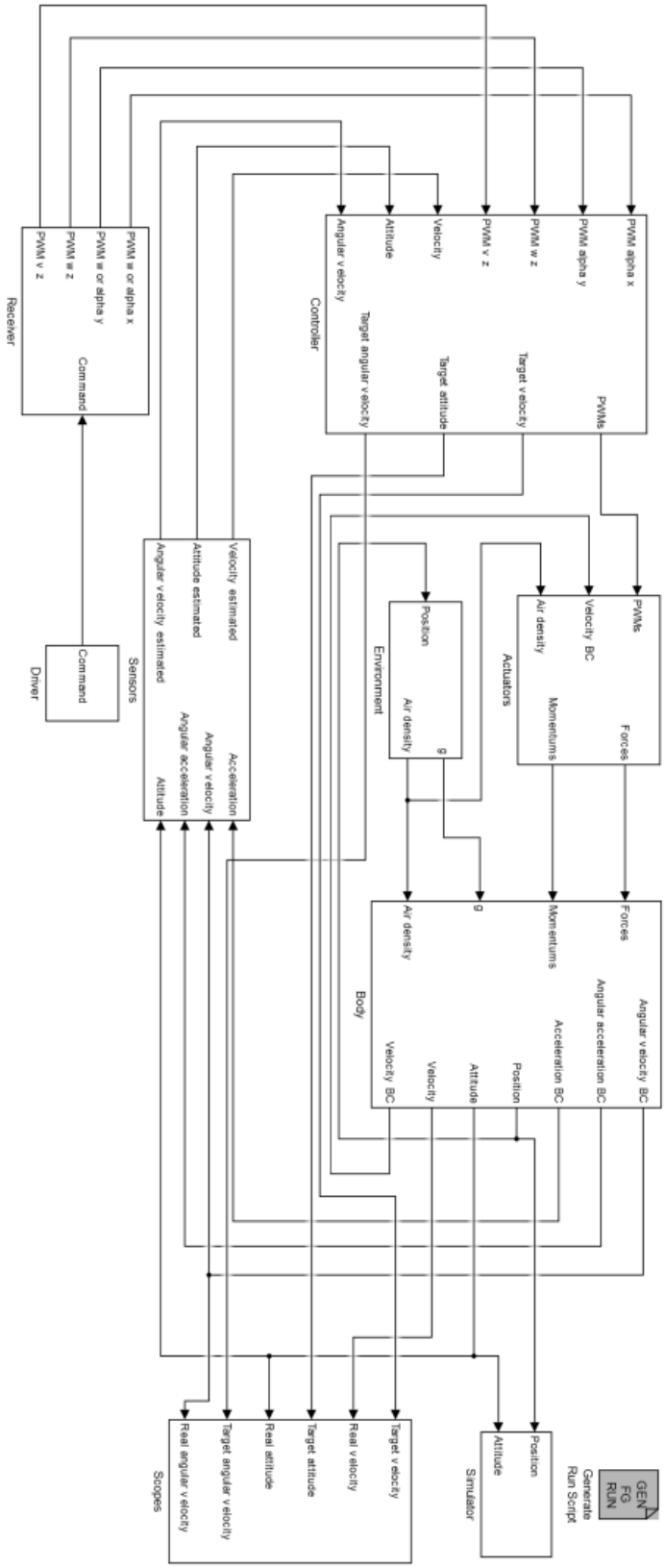
Důležitou součástí senzoru inerciální měřící jednotky je Kalmanův filtr. Je to druh Bayesovského filtru, které slouží ke zpřesnění naměřených hodnot. Data poskytovaná senzory totiž trpí zejména šumem a driftem, z čehož šum se více projevuje u akcelerometru a drift u gyroskopu.

Na rozdíl od komplementárního filtru Kalmanův filtr pracuje nejen s aktuálními stavy, ale i s předchozími. Pomocí vnitřního modelu systému z nich počítá prvotní predikci. Model může mít různé podoby, například výstup z jiného – pomocného senzoru. Jeden krok jeho práce popisuje následující algoritmus:

1. pomocí modelu vypočti a priori predikci stavových veličin
2. z předchozí kovarianční matice chyby a kovarianční matice odhadované chyby procesu vypočti aktuální kovarianční matici chyby
3. vypočti inovační kovarianční matici pomocí kovarianční matice chyby a kovarianční matice odhadované chyby senzoru (šum)
4. z kovarianční matice chyby a inovační kovarianční matice vypočti Kalmanův zisk
5. nový stav se získá úpravou a priori predikce stavových veličin pomocí Kalmanova zisku a hlavní naměřené hodnoty

Kalmanův zisk nám zjednodušeně říká, jakou důvěru máme v naměřenou hodnotu. Pokud je důvěra nízká, výpočet nového stavu se bude spíše orientovat dle odhadu pořízeného modelem. Parametry filtru jsou kromě počátečních hodnot dvě kovarianční matice chyby: modelu a senzoru. Jejich hodnoty jsou inicializovány podle přesnosti jednotlivých komponent.

Takto filtrovaná data mají hladší průběh a dají se jednoduše kombinovat s hodnotami naměřenými „pomocným“ senzorem.



Obrázek č. 14: architektura simulačního modelu

5 Simulace

Pokud chceme věrně modelovat kvadrokoptéru, nejjednodušším způsobem je takzvaný "black box" pohled. Znamená to, že se na činnost stroje budeme dívat jako na systém prvního či druhého řádu, jehož konkrétní parametry se nastaví dle praktických měření na konkrétní reálné sestavě. Já jsem ve své práci ale zvolil jiný způsob, a to takový, že modeluji jednotlivé fyzikální jevy.

Jako nástroj modelování a simulace jsem zvolil Matlab Simulink, který poskytuje grafické rozhraní pro definici modelu pomocí funkčních bloků.

5.1 Ovladač a přijímač

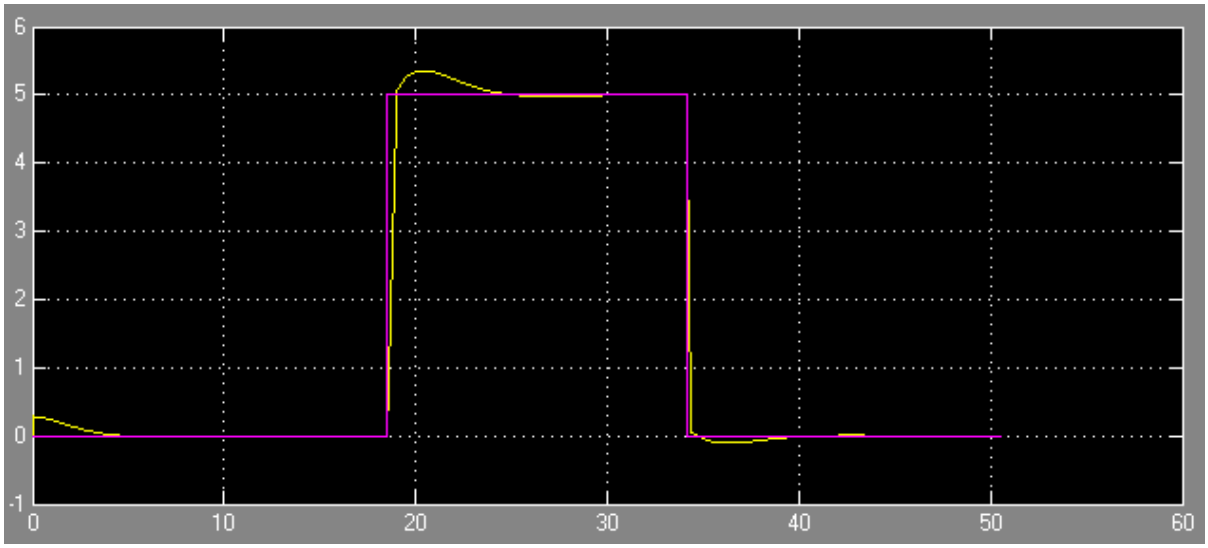
Ovladač obsahuje S-funkci [od2], která umí zachytávat stisknuté klávesy. Pro své potřeby jsem ji upravil tak, aby na výstupu nepřetržitě podávala naposledy zaznamenanou klávesu. Kód klávesy se předá bloku přijímače, který jej převede na příkaz simulující PWM. Zde v modelu se jedná pouze o reálné číslo od 0 do 1. Příkazy představují nastavení extrémní požadované hodnoty dané veličiny v dané orientaci. Ovládání je následující:

- w … maximální úhel sklonu
- s … minimální úhel sklonu
- a … minimální úhel náklonu
- d … maximální úhel náklonu
- q … minimální úhlová rychlost v ose *z*
- e … maximální úhlová rychlost v ose *z*
- y … maximální rychlost stoupání
- c … maximální rychlost klesání
- mezerník … vynulovat příkazy

5.2 Řadič

Model řadiče využívá principy, jaké jsou popsány v kapitole 4.2. Jeho tvorba je ovšem součástí navazující diplomové práce. Bude sloužit také jako předloha skutečného řadiče.

Pro testovací účely je zde řízení implementováno PID regulátory. Nachází se jich zde celkem 6: regulace úhlu náklonu, regulace rychlosti naklánění, regulace úhlu sklonu, regulace rychlosti sklánění, regulace rychlosti stoupání a regulace zatáčení. Výstup regulátoru úhlu náklonu je vstupem do regulátoru rychlosti naklánění, stejně jako výstup regulátoru úhlu sklonu je vstupem do regulátoru rychlosti sklánění. Toto kaskádovité zapojení bohužel zatím není v použitelném stavu. Na následujícím obrázku je zachycena funkce regulátoru rychlosti stoupání. Fialová křivka je požadovaná hodnota, žlutá je skutečná:



Obrázek č. 15: simulace změny požadované rychlosti stoupání
(svislá osa je v jednotkách [m / s] a vodorovná v [s])

5.3 Aktuátory

V bloku aktuátorů jsou 4 podbloky představující rotory. Tyto podbloky jsem kvůli znovupoužitelnosti vytvořil jako samostatné knihovní soubory. Původně jsem chtěl modelovat zvlášť regulátor otáček, motor (elektrotechnické vlastnosti) a vrtuli (propulzní vlastnosti). Díky rychlé smyčce, která se počítá iterativně, jsou všechny tři části úzce spjaty. Modeluje je proto blok funkce s kombinací výpočtů z kapitol 3.1 a 3.2.1.

Parametry tohoto bloku jsou: napětí baterie ($U_b = 11,1\ V$), odpor baterie ($Rb = 0,0035\ \Omega$), odpor regulátoru otáček ($Rr = 0,008\ \Omega$), odpor motoru ($Rm = 0,393\ \Omega$), úhlová rychlost na volt (Kv = 860 / 2 * π / 60 rad / V), proud na prázdno ($I0 = 0,56\ A$), napětí proudu na prázdno ($U_0 = 10\ V$), motorová konstanta ($Rk = 60\ \Omega$), poloměr vrtule ($R = D / 2\ m$) a matice specifikující tvar vrtule. Tato matice obsahuje délky jednotlivých sekcí vůči poloměru a náběh. Hodnoty jsem získal z měření provedených na illinoiské univerzitě [od3]. Konkrétně jsem použil data modelu APC SF 1147 a přepočítal náběh na hodnoty v radiánech:

r [m]	0,15	0,2	0,25	0,3	0,35	0,4	0,45	0,5	0,55
c [m]	0,112	0,137	0,16	0,181	0,198	0,211	0,211	0,227	0,23
P [rad]	0,3428	0,3807	0,3918	0,3819	0,3618	0,3341	0,3019	0,2719	0,2454

r [m]	0,6	0,65	0,7	0,75	0,8	0,85	0,9	0,95	1
c [m]	0,228	0,222	0,213	0,199	0,181	0,158	0,132	0,084	0,035
P [rad]	0,2218	0,2012	0,1827	0,1663	0,1506	0,1346	0,1154	0,0922	0,0686

Tabulka č. 1: parametry vrtule

U knihovního modulu rotoru je kromě výše zmíněných parametrů možné nastavit i smysl otáčení vrtule, který ovlivní znaménko u výsledného točivého momentu.

Na výpočet aerodynamických koeficientů a efektivních ploch jsem aplikoval teorii ideální plochy (ideal flat-plate):

$$C_L = 2 * \pi * \sin(\alpha) \tag{45}$$

$$S_L = c * dr * \cos(\alpha) \tag{46}$$

$$C_D = 2 * \pi * \sin(\alpha) \tag{47}$$

$$S_D = c * dr * \sin(\alpha) \tag{48}$$

Kde C_L je koeficient vztlaku, S_L efektivní plocha vztlaku, C_D koeficient odporu a S_D efektivní plocha odporu.

5.4 Prostředí

V tomto bloku generuji tíhové zrychlení na základě hmotnosti a aktuální polohy kvadrokoptéry dle modelu WGS84. Také se zde nachází blok představující atmosféru, který využívám pro získání hustoty vzduchu závislé na aktuální letové výšce.

5.5 Tělo

Tělo neboli drak kvadrokoptéry je kříž trubek tvořících ramena spojených středovým panelem osazeným elektronikou. Trubky mají čtvercový profil a jejich bod překřížení definuje těžiště stroje. Středový panel je tvořen dvěma rovnoběžnými kruhovými deskami.

Model má svou hmotnost, aerodynamické vlastnosti a tenzor setrvačnosti. Hmotnost je konstantní 1,5 kg.

Model aerodynamiky jsem maximálně zjednodušil. Cílem pouze bylo modelovat mezní rychlost. K tomu posloužil koeficient odporu koule, který je 0,47 pro všechny směry. Plochu jsem taktéž ponechal konstantní pro všechny směry a to 0,03 m^2.

Matice tenzoru setrvačnosti je složena z několika jednodušších. Základními stavebními bloky modelu jsou:

- trubka rovnoběžná s osou *x*: *l = 0,02 m; w = 0,5 m; h = 0,02 m; m = 0,17 kg*
  trubka rovnoběžná s osou *y*: *l = 0,5 m; w = 0,02 m; h = 0,02 m; m = 0,17 kg*
  modely jsou kvádry a vektory posunutí jsou pro obě trubky (0; 0; 0)
- 4x motor: *r = 0,02 m; h = 0,06 m; m = 0,09 kg*
  motory jsou umístěné každý těsně před koncem svého nosného ramena. Zahrnují hmotnost vrtulí. Jsou modelovány válcem a vektory posunutí jsou:
  levý (0; -0,25; -0,042), přední (0,25; 0; -0,042), pravý (0; 0,25; -0,042),
  zadní (-0,25; 0; -0,042),
- horní a spodní disk: *r = 0,08 m; h = 0.002 m; m = 0,1 kg* (spodní)*; m = 0,2 kg* (horní)
  horní disk zahrnuje hmotnost elektroniky a kabeláže. Modelem je válec a vektory posunutí jsou: spodní (0; 0; 0,011), horní (0; 0; -0,011),
- baterie: *l = 0,15 m; w = 0,05 m; h = 0,03 m; m = 0,5 kg*
  baterie je nejhmotnější samostatnou součástkou a je umístěná pod spodní částí středového panelu. Modeluji ji kvádrem s vektorem posunutí (0; 0; 0,027)

V simulačním modelu jsou přítomné matice tenzorů setrvačnosti postaveny nad dalšími znovupoužitelnými bloky. Nižší úroveň zajišťuje modul pro výpočet obecného diagonálního tenzoru, do kterého je možné zadat vektor oddálení od těžiště. Druhou úrovní jsou bloky tenzorů pro kvádr a válec.

$\frac{m*(h^2+l^2)}{12}$	0	0
0	$\frac{m*(h^2+w^2)}{12}$	0
0	0	$\frac{m*(w^2+l^2)}{12}$

Tabulka č. 2: tenzor kvádru [kg / m^2]

$\frac{m*(3*r^2+h^2)}{12}$	0	0
0	$\frac{m*(3*r^2+h^2)}{12}$	0
0	0	$\frac{m*r^2}{2}$

Tabulka č. 3: tenzor válce [kg / m^2]

Po dosazení, aplikaci rovnice (30) a sečtení matic je výsledný tenzor setrvačnosti tento:

0,01641	0	0
0	0,01557	0
0	0	0,0102

Tabulka č. 4: tenzor modelu kvadrokoptéry [kg / m^2]

Tuto matici setrvačnosti předávám na vstup bloku z knihovny Aerospace, který modeluje dynamiku tělesa o 6 stupních volnosti. Mezi jeho další vstupy patří vektor síly (FB) a vektor točivého momentu (FB). Tyto vektory získávám aplikací rovnic (26) a (27) na vektory síly a momentu z aktuátorového bloku. K váhované síle se po té přičtou síly vlivu prostředí. Výstupy bloku dynamiky tělesa jsou rychlost (BF) i (FE), zrychlení (BF), úhlová rychlost (BF) a úhly natočení (FE).

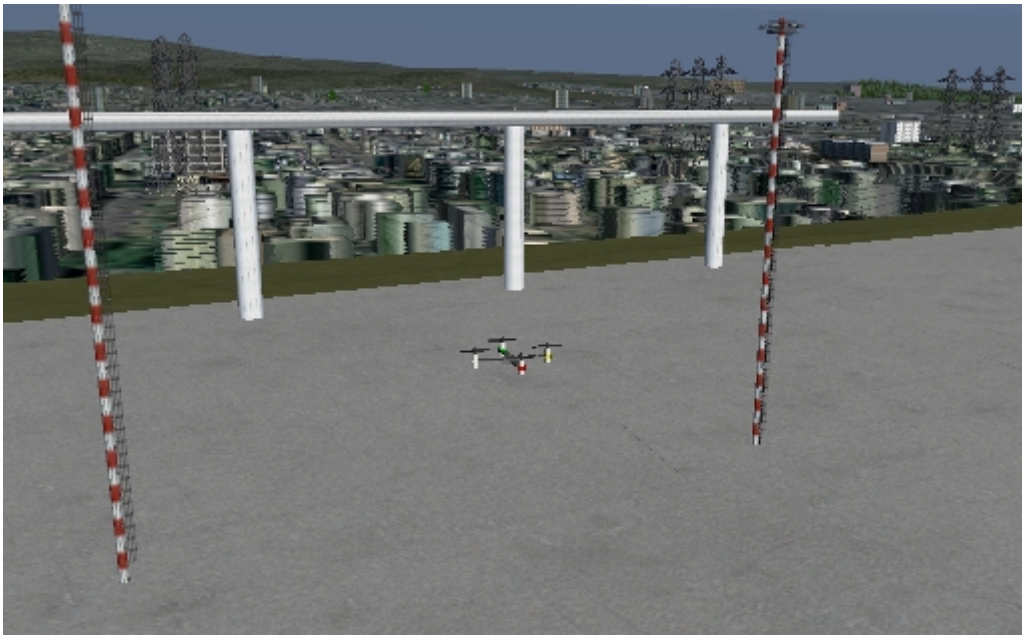
5.6 Senzory

Senzorická jednotka je složena z tříosého akcelerometru a z tříosého gyroskopu. Součástí jsou také integrátory, které poskytují další informace jako rychlost a úhly natočení. Předpokládám, že poskytovaná data již prošla předzpracováním, proto je u akcelerometru nastavena eliminace gravitace a magnetometr zcela chybí. Úroveň šumu je pro zatím nulová.

Za zmínku stojí přítomnost převodu akcelerace v body–fixed systému na flat–earth. Při řízení rychlosti v ose *z* se kvadrokoptéra orientuje podle toho, co vnímá pozorovatel. Takto jednoduchý převod by ovšem neplatil pro rychlosti v osách *x* a *y*. Při zachování směrů požadovaných rychlostí v BF se převod akcelerace na osy *x* a *y* nevztahuje. Další zjednodušení se týká odhadu úhlů natočení, kde k jejich zjištění nestačí použít pouhý integrátor, ale také převod do FE souřadnic. Z uvedeného vyplývá, že model není schopný správně provádět kombinace točivých pohybů.

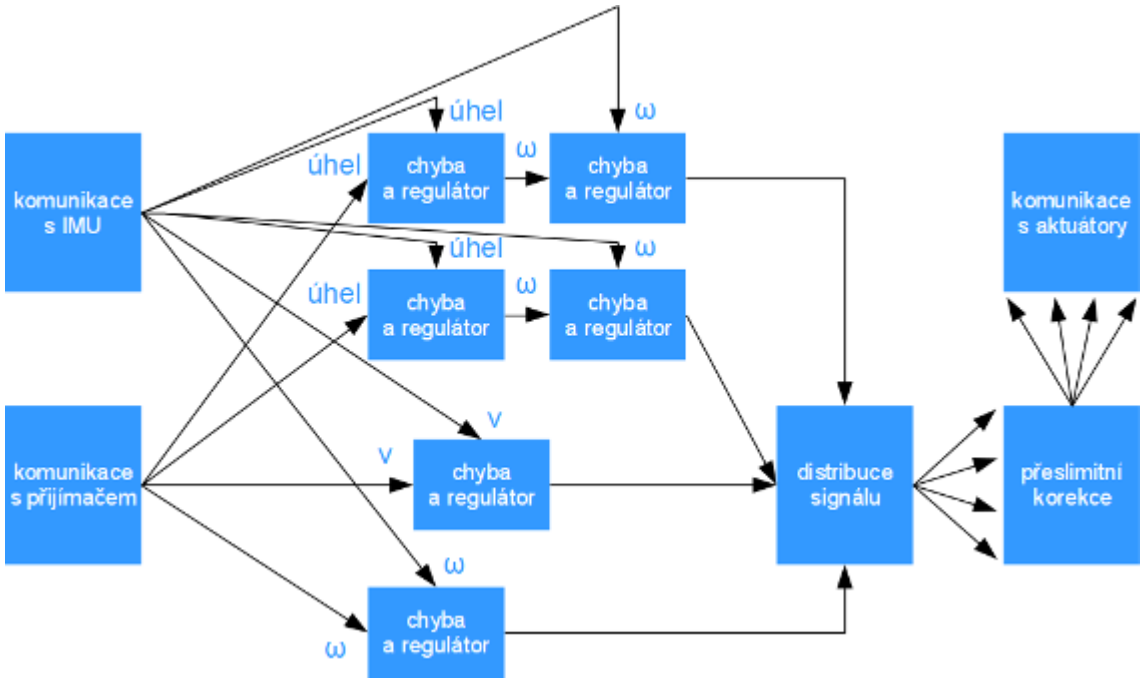
5.7 Vizualizace

Pro vizualizaci využívám externí program Flightgear 3. Komunikační rozhraní mezi Flightgearem a Matlabem poskytuje rozšíření Simulinku Aerospace. Blok vizualizace zajišťuje převod mezi souřadnými systémy flat-earth a Flightgear. Souřadný systém Flightgearu je otočený o -180° kolem osy *y* a pozice se udává v hodnotách GPS. Jako grafický model kvadrokoptéry mi posloužila Quadra [od4].



Obrázek č. 16: ukázka vizualizace

6 Návrh řadiče



Obrázek č. 17: návrh architektury řadiče

Návrh je určen pro firmware mikrokontroléru a vychází z řadiče, který již v modelu existuje. Má ale jisté bloky, které v modelu nemají přímé zastoupení.

- komunikace s IMU – hlavní činností je navození a řízení sériové komunikace se senzorickou jednotkou. Dalším úkolem je případný převod získaných dat do potřebných souřadných systémů a odhad dalších veličin (například integrováním). Zda bude tento úkol provádět řadič záleží na konkrétním typu IMU, proto je zapouzdřen v tomto bloku. Veškeré hodnoty se zde převádí do interní reprezentace
- komunikace s přijímačem – tento blok je zodpovědný za zpracování PWM signálu a jeho převodu do interní reprezentace. Za použití jednoho časovače bude měřit střídu všech kanálů: stoupání, zatáčení, náklonu, sklonu a pomocné kanály (rozšířená funkcionalita)
- komunikace s aktuátory – komunikace směrem k aktuátorům probíhá taktéž prostřednictvím PWM signálu. Jeho generování z hodnot interní reprezentace zajišťuje opět časovač
- blok s označením chyba a regulátor se stará o výpočet regulační odchylky a o vlastní regulaci. Z prezentovaných metod regulace bych zvolil implementaci PID pomocí neuronové sítě. Tato metoda je z principu nelineární a neuronová síť se může v případě potřeby naučit aproximovat i jiné klasické regulátory

7 Implementace

V následujících podkapitolách popíšu princip fungování jednotlivých návrhových částí vzniklého řídícího programu. Kód je určen pro vývojovou desku Arduino Due, jejíž výrobci poskytují pohodlné programátorské rozhraní. Další konkrétní součástkou je senzorická jednotka UM6 od Ch Robotics.

V jednotlivých sekcích bude uveden diagram příslušné třídy. Veškeré vlastnosti a funkce jsem z důvodů ladění vytvořil s veřejnou přístupností.

7.1 IMU paket

ImuPacket
+ ADDR_COMMUNICATION : char + ADDR_CONFIGURATION : char + ADDR_FLASH_COMMIT : char + ADDR_ZERO_GYROS : char + ADDR_SET_ACCEL_REF : char + ADDR_SET_MAG_REF : char + ADDR_RESET_EFK : char + ADDR_GYRO_BIAS_XY : char + ADDR_GYRO_PROC_XY : char + ADDR_ACCEL_PROC_XY : char + ADDR_EULER_PHI_THETA : char + ADDR_QUAT_AB : char + TB_HAS_DATA : char + TB_IS_BATCH: char + TB_BATCH_L3 : char + TB_BATCH_L2 : char + TB_BATCH_L1 : char + TB_BATCH_L0 : char + TB_RESERVED : char + TB_CMD_FAIL : char + TBY_CC : char + start : char [] + pt : char + address : char + data : char [] + checksum1 : char + checksum0 : char
+ ImuPacket () + ~ImuPacket () + reset () : void + setBatchLength (batchLength : int) : void + getBatchLength () : int + getDataLength () : int + computeChecksum () : int + setChecksum (checksum : int) : void + getChecksum () : int + fillVars (vars : float [], varCount : char, offset : char, scale : float) : void

Třída IMU paketu zapouzdřuje základní datovou jednotku IMU UM 6. Poskytuje konstanty adres příkazů a datových registrů IMU a odstiňuje programátora od nepohodlné práce s jednotlivými bity a bajty.

Struktura paketu je dle referenční příručky následující:

's'	'n'	'p'	typ	adresa	data *	KS 1	KS 0

Důvodem, proč pro paket byla vyčleněna vlastní třída, byly funkce poskytující přímé číselné hodnoty z posloupnosti bitů a na druhou stranu přímé číselné nastavování. Patří sem zejména metoda *getBatchLength*, která z bajtu určujícího typ paketu získává popis velikosti případných dat. Datové bajty mohou zcela chybět (nulovaný bit „obsahuje data“ v bajtu „typ“), ale také jich může být větší počet – <délka dávky (batch length)> * 4. IMU paket také poskytuje rozhraní pro manipulaci s kontrolním součtem (KS). Zahrnuje jeho čtení, zápis a vlastní výpočet. Dále metoda *fillVars* převádí surová data na konkrétní číselné hodnoty.

7.2 IMU

Imu
+ S_IDLE : char + S_S : char + S_N : char + S_P : char + S_PT : char + S_ADDR : char + S_CHSM1 : char + ER_PRSS : char + ER_PRSN : char + ER_PRSP : char + ER_CHSM : char + ER_WAIT : char + SC_GYRO : float + SC_ACCEL : float + SC_EULER : float + SC_QUAT : float + ser : HardwareSerial + state : char + error : dataLength : char + requestPacket : ImuPacket + responsePacket : ImuPacket
+ Imu (ser : HardwareSerial, baudRate : int) + setRequest (address : char, batchLength : int, data = NULL : char []) : void + sendPacket (nullPointer = NULL : NULL) : void + receiveByte (b : char) : ImuPacket + receivePacket () : void : ImuPacket + flushReceiver () : void

Manipulaci s IMU pakety zajišťuje třída IMU. Její funkce se dělí na příjem paketů a jejich odesílání. K obojímu jí slouží sériová sběrnice a to konkrétně UART.

Přijímání paketu se děje pomocí stavového automatu metody *receiveByte*, kterou v cyklu volá funkce *receivePacket*. Čtení úvodních bitů ze vstupního bufferu (64 bajtů) do dynamicky alokovaného objektu IMU paketu je triviální činnost. Protože ovšem paket může obsahovat data, počet stavů není pevně daný. Při čtení bajtů za adresovacím bajtem je tedy potřeba stavy popisovat délkou dat a potřebným odsazením. Po úspěšném porovnání vypočteného a příchozího kontrolního součtu z posledních dvou bajtů je paket vrácen do nadřazené funkce. Protože se v prvním stavu jako opatření při nepovedeném čtení provádí mazání dynamicky alokovaného paketu, je do vnitřní proměnné přiřazena hodnota NULL.

NULL se také vrací nadřazené funkci při selhání přijímání, což může zavinit nesouhlas kontrolního součtu, či dokonce špatná startovací sekvence paketu „snp“ (start new packet). Dost často se stává, že paket ještě nedorazil celý. V takovém případě vrací metoda *receivePacket* za zachování současného stavu také NULL, aby nebyly blokovány případné další výpočty.

Odesílání paketu probíhá nad stále stejným paketem, jehož obsah je pokaždé resetován a nastaven pomocí metody *setRequest*. Ta nastaví adresu a délku dávky, o kterou žádáme, či kterou posíláme. To záleží na typu příkazu kódovaném právě adresou registru a tom, jestli využijeme volitelného třetího parametru a přiložíme do paketu data. Při odesílání pomocí metody *sendPacket* se do výstupního bufferu sériového portu sestaví sekvence bajtů paketu s vypočteným kontrolním součtem.

Pokud se posílají data, předpokládá se, že jejich definice může být i statická. Proto se uvnitř metody *setRequest* vytváří dynamická kopie, aby při resetování paketu nenastalo nedefinované chování při aplikaci funkce *free*.

7.3 Senzory

Sensors
+ ROUND_FACTOR : char + S_STANDBY : char + S_CALIBRATION_START : char + S_CALIBRATION_PROGRESS : char + S_CALIBRATION_START2 : char + S_CALIBRATION_PROGRESS2 : char + S_CALIBRATION_START3 : char + S_CALIBRATION_PROGRESS3 : char + S_CALIBRATION_START4 : char + S_CALIBRATION_PROGRESS4 : char + S_CALIBRATION_DONE : char + S_ARMED : char + imu : Imu + angularVelocityOld : float [] + angularVelocity : float [] + acceleration : float [] + eulerAngles : float [] + quaternion : float [] + gravityVector : float [] + velocity : float [] + state : char + clock : int
+ Sensors (ser : HardwareSerial, baudRate int) + sense () : bool + rotateVectorByInversedQuaternion (vector : float [], quaternion : float [], result : float []) : void

Pokud by byla kvadrokoptéra vybavena více senzory, všechny jejich zprostředkující třídy by měly objektové zastoupení v objektu třídy senzory. V současné době je jediným senzorem IMU. Metoda *sense* volá již zmiňovanou metodu *receivePacket*. Pokud je paket přijat, následuje akce dle stavu, ve kterém se objekt senzorů nachází. Stavy se dají rozdělit na dvě hlavní skupiny:

- Příjem dat
- Startovací (kalibrační) sekvence

7.3.1 Příjem dat

Z pohledu třídy se jedná v podstatě o sdružený stav, jelikož, jak název napovídá, je v něm plněna základní funkce senzoru. Příjem dat je kromě ostrého použití (ARMED) aktivní i ve výchozím stavu (STANDBY) a ve stavu pauzy (PAUSED).

V těchto stavech jsou dle adresy přijímány hodnoty z gyroskopu, akcelerometru a jednotky odhadující úhel natočení senzoru. IMU UM6 totiž díky vestavěnému mikroprocesoru zpracovává naměřená data z gyroskopu, magnetometru a akcelerometru mimo jiné tak, že provádí odhad natočení pomocí Kalmanovy filtrace. Tento odhad přijímá třída senzorů ve dvojí podobě: Eulerovy úhly a kvaternion. První z dvojice se stejně jako hodnoty z gyroskopu a akcelerometru uloží pro další použití.

Přijmutím kvaternionu ovšem započne výpočet dalších hodnot, které inerciální měřící jednotka sama o sobě neposkytuje. Naměřená akcelerace je totiž vztažená vzhledem k senzoru – tedy v body-fixed souřadném systému, a tak je třeba ji tranformovat. Kvaterniony mají dvě hlavní výhody oproti Eulerovým úhlům:

1. Provádí se nad nimi efektivní výpočet otáčení vektoru v prostoru.
2. Díky vnitřní reprezentaci pomocí definice obecné osy otáčení nepodléhají fenoménu zvanému „gimbal lock“.

„Gimbal lock“ je jev, který se dá popsat jako splývání os otáčení. Projevuje se tím více, čím více se natočení v některé ose blíží 90°. Také například stojí za nejednoznačností mapování některých skutečných poloh na Eulerovy úhly.

Kvaternion zasílaný IMU popisuje natočení globálního souřadného systému vůči aktuálním souřadnicím senzoru. Výpočet prováděný při jeho přijmutí tedy spočívá v otočení vektoru naměřeného zrychlení. Jelikož požadovaná transformace je otočení z globálního do body-fixed souřadného systému, vektor se musí otáčet rotací body-fixed vůči globálu. Kvaternion ovšem popisuje opačný směr, a proto musí být před použitím invertován. O to a vlastní otáčení se stará metoda *rotateVectorByInversedQuaternion*.

Od akcelerace v obecných souřadnicích je následně odečten referenční vektor gravitace. Takto upravená proměnná již může být integrována, čímž vzniká odhad rychlosti, jakou se stroj pohybuje v prostoru. K měření časového skoku se používá rozdíl návratové hodnoty vestavěné funkce *micros* a jejího předchozího volání na stejném místě.

Všechny proměnné plněné metodou *fillVars* jsou zaokrouhlovány vynásobením zaokrouhlovacím faktorem (ROUND_FACTOR), zaokrouhlením na celá čísla a zpětném podělením faktorem. Úhlová rychlost a odhad natočení jsou ještě korigovány přičtením konstantních hodnot, k jejímž zjištění jsem dospěl metodou popsanou v kapitole 8.2. Úhlová rychlost navíc trpí velkým šumem, proto průměruji dvě poslední příchozí hodnoty.

7.3.2 Startovací (kalibrační) sekvence

Z pohledu senzorů se jedná spíše o kalibrační sekvenci. Po započetí, které je iniciováno explicitní změnou stavu objektu senzorů na CALIBRATION_START, se postupně odesílají pakety s kalibračními příkazy.

Nejdříve se vynuluje gyroskop. Příkaz způsobí, že se do registrů v IMU zapíší hodnoty, které se budou odečítat od měření. Po dobu kalibrace musí být tedy zajištěna nulová rotace kvadrokoptéry. Po přijetí paketu potvrzení dokončení procesu se přechází do dalšího stavu.

Nyní se posílá příkaz pro zapamatování výchozího vektoru akcelerace. Tuto referenci IMU používá s úvahou: Čím bližší je aktuální měřená akcelerace tomuto vektoru, tím blíž je aktuální natočení nulovému ve všech osách. Po potvrzení akce přechází stavový automat do další sekce.

Tato sekce je zodpovědná za zapsání referenčního vektoru magnetického toku. Jedná se o obdobu předchozí kalibrace se stejným cílem ovšem jinou měřenou veličinu.

Jako poslední se resetuje odhadovací algoritmus IMU s veškerými mezivýpočty. Při přijetí potvrzení o úspěchu operace ještě objekt senzorů vyresetuje nakumulovaný odhad rychlosti a zapamatuje si aktuální akceleraci jako vlastní referenční vektor gravitace. Přechází se do stavu CALIBRATION_DONE.

7.3.3 Nastavení IMU

Ve třídě senzorů se také provádí případné nastavení inerciální měřící jednotky UM6. Ta má dva základní nastavovací příkazy (registry):

- Komunikace
- Ostatní konfigurace

Registry prvního příkazu byly nastaveny tak, aby bylo k dispozici: automatické pravidelné odesílání požadovaných hodnot (tzv. broadcast), posílání údajů z gyroskopu a akcelerometru, Eulerových úhlů a kvaternionu. Další dva důležité parametry jsou rychlost přenosu a frekvence automatického odesílání. Rychlost přenosu je nastavena na maximální bezpečnou hodnotu – 57 600 baudů a frekvence broadcastu je zhruba 72 Hz. To je maximální hodnota, při které lze bezpečně přijímat kromě datových paketů také řídící (potvrzení kalibrace apod.). IMU vyhrazuje pro každou veličinu zvláštní paket. Jeden paket tedy obsahuje jen 8 bajtů dat (2 registry = 3 až 4 čísla), což zvyšuje nároky na přenosové pásmo.

Ostatní konfigurace zahrnuje zejména povolenou podporu magnetometru a akcelerometru při odhadu polohy. Automatickou kalibraci gyroskopu jsem zakázal, protože ji provádím až v kalibrační sekvenci. Tehdy je zaručeno, že se se strojem nehne například při zapojování napájecích drátů. V této sadě registrů jsem ještě povolil vlastní výpočet kvaternionu, který se dá touto cestou jakožto náročná operace samostatně deaktivovat.

7.4 Přijímač

Receiver
+ channelCount : char + YAW : char + PITCH : char + THROTTLE : char + ROLL : char + ARMING : char + MODE : char + channelPins : char [] + pulsesMin : int [] + pulsesMid : int [] + pulsesMax : int [] + flags : char + ROUND_FACTOR : char + S_STANDBY : char + S_CALIBRATION_START : char + S_CALIBRATION_PROGRESS : char + S_CALIBRATION_DONE : char + S_ARMED : char + S_PAUSED : char + flagByte : char + clocks : int [] + pulses : int [] + pulsesPersistent : int [] + signals : float [] + state : char
+ Receiver () + receiveSignal () : bool + getPWMChannel0 () : void + getPWMChannel1 () : void + getPWMChannel2 () : void + getPWMChannel3 () : void + getPWMChannel4 () : void + getPWMChannel5 () : void + getPWM (channel : char) : void

Třída přijímače slouží k převodu pulzně-šířkové modulace na vnitřní signál řadiče. Definuje piny, na kterých je připojeno 6 z 8 kanálů fyzického radiopřijímače. Pomocí vestavěných funkcí na ně zaregistruje rutiny obsluhy přerušení (callback), které se volají při každé změně hodnoty pinu. Na každý pin existuje jedna rutina, která volá společnou statickou metodu *getPWM* s pořadovým číslem kanálu coby parametrem.

Tato metoda dle aktuální hodnoty pinu zjistí, zda se jedná o náběžnou nebo sestupnou hranu. Pokud se jedná o náběžnou, aktualizuje časovač přiřazený danému kanálu pomocí funkce *micros*. Pokud se naopak jedná o sestupnou hranu, odečte se hodnota časovače od aktuálního návratové hodnoty *micros*. Rozdíl představuje délku pulzu daného kanálu v mikrosekundách.

Jelikož popisovaný výpočet probíhá v obsluze přerušení, kód musí zajistit určité podmínky:

1. kód funkce *getPWM* musí být rychle proveditelný
2. proměnné se kterými funkce *getPWM* pracuje musí být volatilní – tedy, že s touto proměnnou nesmí kompilátor provádět optimalizace typu „ponechám změněnou hodnotu jen v registru procesoru bez zapsání do paměti“
3. předání volatilních hodnot do proměnných, které neovlivňuje přerušení, musí proběhnout při deaktivovaném přerušení. Toto opatření existuje, protože příkaz přiřazení není atomický a mezi zkopírováním 2 bajtů hodnoty by přerušení mohlo vytvořit zcela nevalidní data

Pokud byl některý z pulzů aktualizován, pak je tento přepočten na odpovídající hodnotu signálu. Každý kanál má 3 význačné hodnoty: minimální (cca 1040), střední (cca 1457) a maximální (cca 1870) hodnotu. Pomocí těchto tří hodnot a aktuální délky pulzu se vypočítá signál v rozsahu od -1 do 1.

Signál ještě prahuji pomocí běžného zaokrouhlení (opět na 1 / ROUND_FACTOR) kvůli redukci šumu.

7.4.1 Startovací (kalibrační) sekvence

Jak jsem již uvedl, třída přijímače definuje 6 kanálů:

- ovládání tahu
- ovládání zatočení
- ovládání náklonu
- ovládání sklonu
- startovací signál
- výběr řídícího módu

Ovládací kanály mají, věřím, zřetelný význam. Startovací signál hraje se svými dvěma polohami hlavní roli ve startovací sekvenci kvadrokoptéry. Výběr módu má také pouze dvě polohy, z čehož každá odpovídá letovému módu blíže popsanému v kapitole 7.6.

Podobně jako u třídy senzorů je jádrem sekvence stavový automat. Z výchozí polohy STANDBY se do prvního stavu kalibrace CALIBRATION_START dostane při přechodu startovacího signálu ze záporných do kladných hodnot. Na tento stav čeká v hlavním souboru objekt senzorů, aby spustil svou část kalibrační posloupnosti. V jejím průběhu znamená přechod startovacího signálu z kladných do záporných hodnot okamžité nastavení výchozího stavu jak v objektu přijímače, tak v objektu senzorů. Tato možnost je zde proto, aby při nahodilém nezachycení některého z řídících paketů mohl být celý proces zopakován. Po dokončení kalibrace senzorů je stav CALIBRATION_DONE opět v hlavním souboru předán do objektu přijímače.

V tuto chvíli nastává kalibrace přijímače. Aktuální délky pulzů ovládání zatočení, náklonu a sklonu jsou ukládány jako příslušné střední hodnoty použité pro výpočet normalizovaného signálu. Děje se tak dokud startovací signál nepřejde z kladných do záporných hodnot, což způsobí přechod do stavu ARMED a následnou aktivaci aktuátorů.

Z důvodu bezpečnosti má startovací signál ještě opačnou, tedy deaktivační úlohu. Ze stavu ARMED se přechodem z jeho záporných hodnot do kladných lze dostat do stavu PAUSED, ve kterém jsou aktuátory opět deaktivovány. Celý koloběh lze uzavřít opětovným přechodem startovacího signálu z kladných do záporných hodnot, což zapříčiní nastavení výchozího stavu STANDBY.

7.5 PID

PID
+ MAX_INTEGRATION : float + P : float + I : float + D : float + proportion Old: float [] + proportion : float + integration : float + derivation : float + maxIntegration : float + clock : int
+ PID (P : float, I : float, D : float) + PID (P : float, I : float, D : float, maxIntegration : float) + regulate (proportion : float) : float

Třída PID představuje stejnojmenný regulátor. Bohužel zvolená platforma Arduino Due nepřekypuje výkonem, a proto kvůli předejití komplikacím byl jako koncept regulace zvolen ten nejjednodušší. Regulace v metodě *regulate* probíhá tak, jak byla popsána v kapitole 4.3.1. Za zmínku stojí snad jen to, že kumulovaná chyba je zde omezena, aby v případě nevhodného nastavení koeficientu I nedocházelo k nevratné ztrátě kontroly nad řízením. Toto omezení nachází opodstatnění i před samotným vzlétnutím, kdy řadič již může cítit mírnou odchylku, ale aktuátory s tím nic nezmůžou.

Použití derivační složky bývá komplikováno šumem. Šum sice nabývá hodnot relativně malé magnitudy, ale jeho změny jsou extrémně rychlé. Proto regulátor obsahuje alespoň jednoduché průměrování hodnot vstupujících do výpočtu derivace.

Změna časového skoku pro výpočet diference a sumace se získává osvědčeným způsobem, a to rozdílem návratových hodnot současného a předchozího volání funkce *micros*. Rozdíl se na sekundy převádí podělením milionem.

7.6 Řadič

Controller
+ X : char + Y : char + Z : char + FRONT : char + LEFT : char + BACK : char + RIGHT : char + MAX_ANGULAR_VELOCITY : float + MAX_ACCELERATION : float + MAX_ATTITUDE : float + MAX_VELOCITY : float + M_ACRO : char + M_TRAN : char + mode : char + angularVelocity : float [] + attitude : float [] + velocity : float [] + targetAngularVelocity : float [] + targetAcceleration : float [] + targetAttitude : float [] + targetVelocity : float [] + actions : float [] + regulatorsAngularVelocityAcro : PID [] + regulatorsAngularVelocityTran : PID [] + regulatorsAttitude : PID [] + regulatorsVelocity : PID []
+ Controller () + control () : void + regulateAcro () : void + regulateTran () : void + control (zAccelerationAction : float, xAngularAccelerationAction : float, yAngularAccelerationAction : float, zAngularAccelerationAction : float) : void + computeNormalizationCoefficient (first : float, second : float) : float + resetRegulators () : void

Veškeré dosavadní třídy byly měly „pouze“ charakter jakéhosi rozhraní. Hlavní logika celého programu je ukryta ve třídě řadiče. Ten má k dispozici již předzpracované signály ze senzorů a přijímače a jeho výstupem jsou akční signály pro jednotlivé motory. Jeho fungování má tyto sekce:

1. regulace
2. saturace
3. distribuce signálu
4. normalizace

7.6.1 Regulace

Regulace se v metodě *control* dělí na 2 typy dle řídícího módu:

1. V módu akrobatickém operátor ovládá a kvadrokoptéra reguluje úhlovou rychlost ve všech osách otáčení a tahovou akceleraci. Hodnota akce tahové akcelerace je očekávána v rozmezí od 0 do 1, proto musí být signál náležitě lineárně transformován. Akce úhlového zrychlení se získají jako návratové hodnoty regulátorů úhlové rychlosti pro akrobatický mód.
2. Dopravní mód:
    1. V dopravním módu je přímo regulována tahová rychlost a úhlová rychlost pouze kolem osy z. Ve zbylých osách otáčení se reguluje úhel natočení stroje. Uspořádání regulátorů v dopravním módu odpovídá zapojení na obrázku 17. Akce tahové akcelerace je získána z regulátoru rychlosti pro osu z (ve výsledném kódu není implementováno – viz kapitola 8.2. Akce cílové úhlové rychlosti náklonu a sklonu jsou vypočteny regulátory natočení pro osy x a y.
    2. Vypočtené akční zásahy jsou reprezentovány vnitřním signálem, pro který platí rozsah hodnot od 0 do 1. V tomto kroku toto omezení řeším oseknutím.
    3. Cílové úhlové rychlosti se ze svých akcí získají vynásobením 2 x konstantou MAX_ANGULAR_VELOCITY. Tato konstanta definuje, jak relativně rychle se obecně kvadrokoptéra otáčí. Pro dopravní mód je ale potřeba rychlejší odezva, proto se násobí ještě dvěma.

7.6.2 Saturace

Protože hodnoty výstupních signálů řadiče musí být v rozmezí od 0 do 1, je zapotřebí akční zásahy vypočtené v předchozím kroku korigovat. Zatím se výpočet nachází ve fázi, kdy jsou akční zásahy k sobě relativní, proto je výhodné je omezit od -0,5 do +0,5.

7.6.3 Distribuce signálu

V této části kódu se akční zásahy převádějí na absolutní signály pro řízení motorů.

1. Nejdříve se do všech signálů přiřadí hodnota akčního zásahu tahové akcelerace.
2. Akční zásah náklonu se přičte k signálu levého motoru a odečte od signálu pravého motoru.
3. Akční zásah sklonu se přičte k signálu předního motoru a odečte od signálu zadního motoru.
4. Akční zásah úhlového zrychlení v ose z se přičte k signálu levého a pravého motoru a odečte od signálu předního a zdaního motoru.

7.6.4 Normalizace

Normalizací jsem zde nazval soubor několika přeslimitních korekcí. Normalizačním koeficientem zde rozumím přesah nad číslo 1 či pod číslo 0. Tento koeficient vypočítává funkce *computeNormalizationCoefficient* vždy z antagonistické dvojice signálů.

1. Zjistím přesah z levopravé dvojice signálů.
2. Zjistím přesah z předozadní dvojicé signálů.
3. Přesah s větší magnitudou odečtu od všech signálů.
4. Vypočítám znovu koeficient, který měl v předchozích krocích menší magnitudu a odečtu ho od signálů příslušné protichůdné dvojice.

Tímto postupem zajistím, že nejvyšší prioritu řízení má regulace náklonu a sklonu. Jelikož ve fázi saturace byly relativní akční zásahy vhodně omezeny, mám také jistotu, že signály pro motory mají vždy hodnoty v rozsahu 0 až 1.

7.7 Aktuátory

Actuators
+ motorCount : char + FRONT : char + LEFT : char + BACK : char + RIGHT : char + PULSE_MIN : int + BLIND_ZONE : int + PULSE_MAX : int + motorPins : int [] + motors : Servo [] + signals : float [] + pulses : int []
+ Actuators () + generatePWM (signals : float [], standby : bool)

Třída aktuátorů ve svém konstruktoru napojí na výstupní piny objekty třídy Servo z knihovny Arduina. V metodě *generatePWM* pomocí těchto objektů generuje dle vstupních signálů na dané piny pulzně-šířkově modulovaný signál pro regulátory rychlosti motorů. Z vnitřního signálu vytvoříme pulzy lineární transformací na hodnoty v rozsahu BLIND_ZONE až PULSE_MAX. BLIND_ZONE by měla být minimální hodnota, která roztočí motor, aby se odstranila skoková změna v přenosové funkci aktuátoru. Objektu typu Servo je předána šířka střídy v podobě počtu mikrosekund.

Metoda generatePWM má kromě vstupních signálů také booleovský parametr příznaku deaktivace. V případě pravdivé hodnoty je na výstupní piny generován minimální pulz zajišťující nečinnost motorů.

7.8 Hlavní soubor

Hlavní soubor propojuje 4 základní stavební bloky: senzory, přijímač, řadič a aktuátory.

Od každého vytvoří jednu instanci. Hlavní soubor má také příznaky aktualizace dat v senzorech či přijímači. Důležitou proměnnou je také stav, který slouží jako spojovací článek mezi stavovým automatem senzorů a přijímače.

V hlavní smyčce jsou nejdříve zavolány metody *sense* a *receiveSignal* pro zpracování případných aktualizovaných dat ze senzorů a přijímače. Příznaky zapamatovány globálními proměnnými. Pokud již byly aktualizovány údaje z obou zařízení, předají se tyto hodnoty řadiči. V případě předávání příkazů v podobě požadovaných zrychlení, rychlostí či natočení,

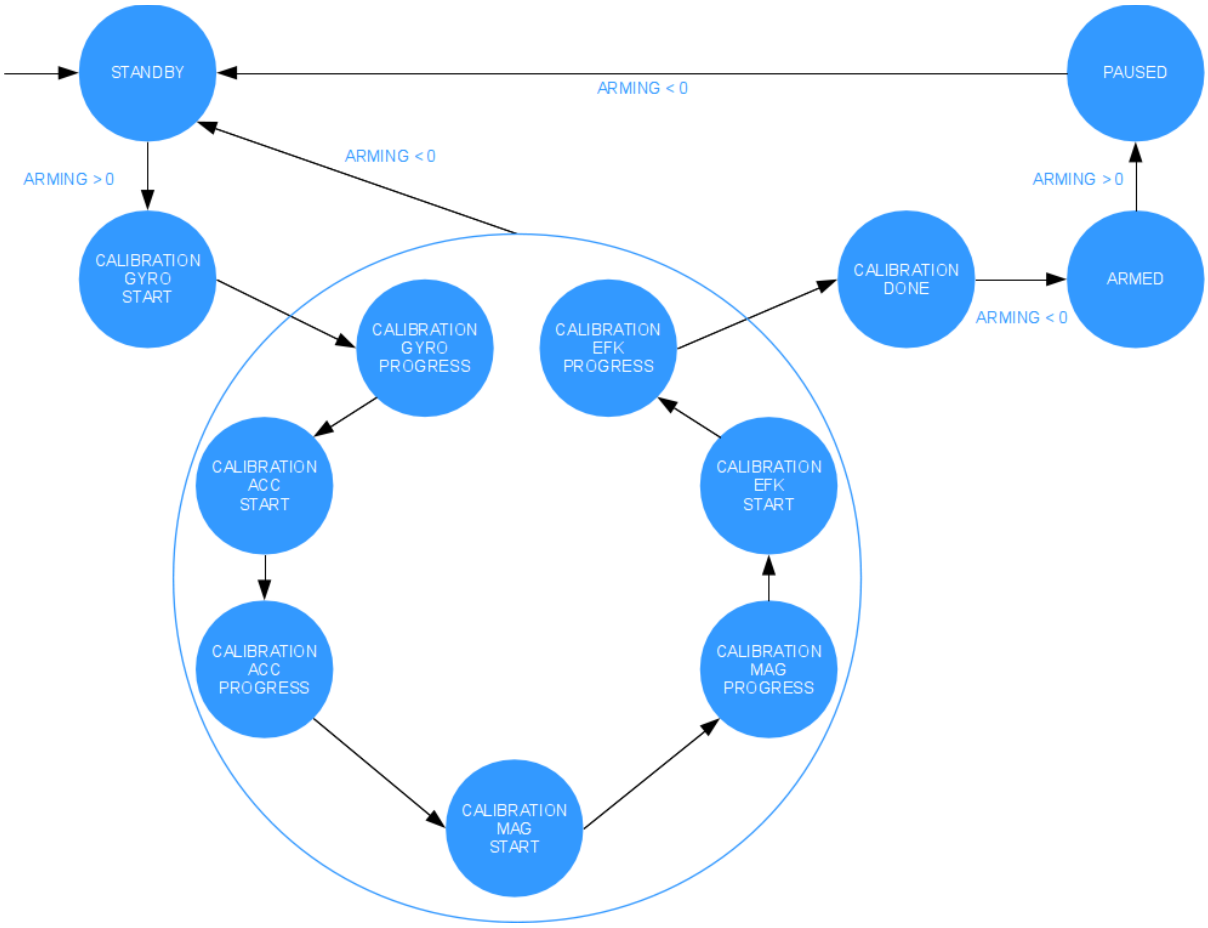
musí se tyto nejdříve vynásobit příslušnými konstantami, protože signály z přijímače jsou taktéž omezeny od 0 do 1.

Když má řadič k dispozici veškeré informace, zavolá se na něm metoda *control*. S nově vypočtenými signály pro motory se zavolá aktuátorová metoda *generatePWM*. Její druhý parametr je nerovnost stavu kvadrokoptéry se stavem ARMED. Příznak aktuality dat senzoru je nastaven na nepravdu, ale pro přijímač se nemění. To je proto, že rychlost a hlavně pravidelnost řídících výpočtů je dána periodou vysílání senzorických dat.

Předávání stavu mezi objekty přijímače a senzorů funguje tak, že se v hlavní smyčce kontroluje, zda jedna z komponent nezměnila svůj stav. Pokud ano, dojde k distribuci tohoto stavu do hlavního souboru a dále do druhého z objektů. Výchozím stavem všech komponent je STANDBY.

- Ve stavu STANDBY se kontroluje změna stavu přijímače. Ten předáním svého stavu CALIBRATION_START spustí kalibrační sekvenci v senzorech.
- Ve stavu CALIBRATION_START se sleduje změna stavu senzorů, protože pokud má tento hodnotu CALIBRATION_PROGRESS, dojde opět k jeho distribuci.
- Ve stavu CALIBRATION_PROGRESS se kontroluje jak stav přijímače tak i senzorů. Pokud kalibrace senzorů proběhla v pořádku, je přijmut stav CALIBRATION_DONE. Naopak přijímač může svým stavem STANDBY ukončit kalibraci předčasně.
- Ve stavu CALIBRATION_DONE je již očekávána pouze změna stavu přijímače na hodnotu ARMED. V tento okamžik se také řadiči nastaví mód řízení dle aktuální hodnoty příslušného kanálu přijímače.
- Ve stavu ARMED může vyvolat změnu stavů přijímač přechodem do stavu PAUSED.
- Ve stavu PAUSED hraje roli opět jen přijímač, který svým stavem STANDBY uvede celý systém do výchozí polohy.

Celý koloběh je zobrazen pomocí upraveného konečného automatu na obrázku 18.



Obrázek č. 18: upravený konečný automat popisující koloběh systému

V hlavním souboru se také nachází funkce *printVars*, která slouží pro pohodlný výpis proměnných při ladění.

Funkce *storeVarsGraph* a *printVarsGraph* slouží k ukládání repektive pozdějšímu zobrazování proměnných. Funkce využívají knihovnu třetí strany DueFlashStorage, která poskytuje rozhraní pro čtení a zápis do flash paměti Arduina, která jako jediná dokáže pojmout požadované množství informací.

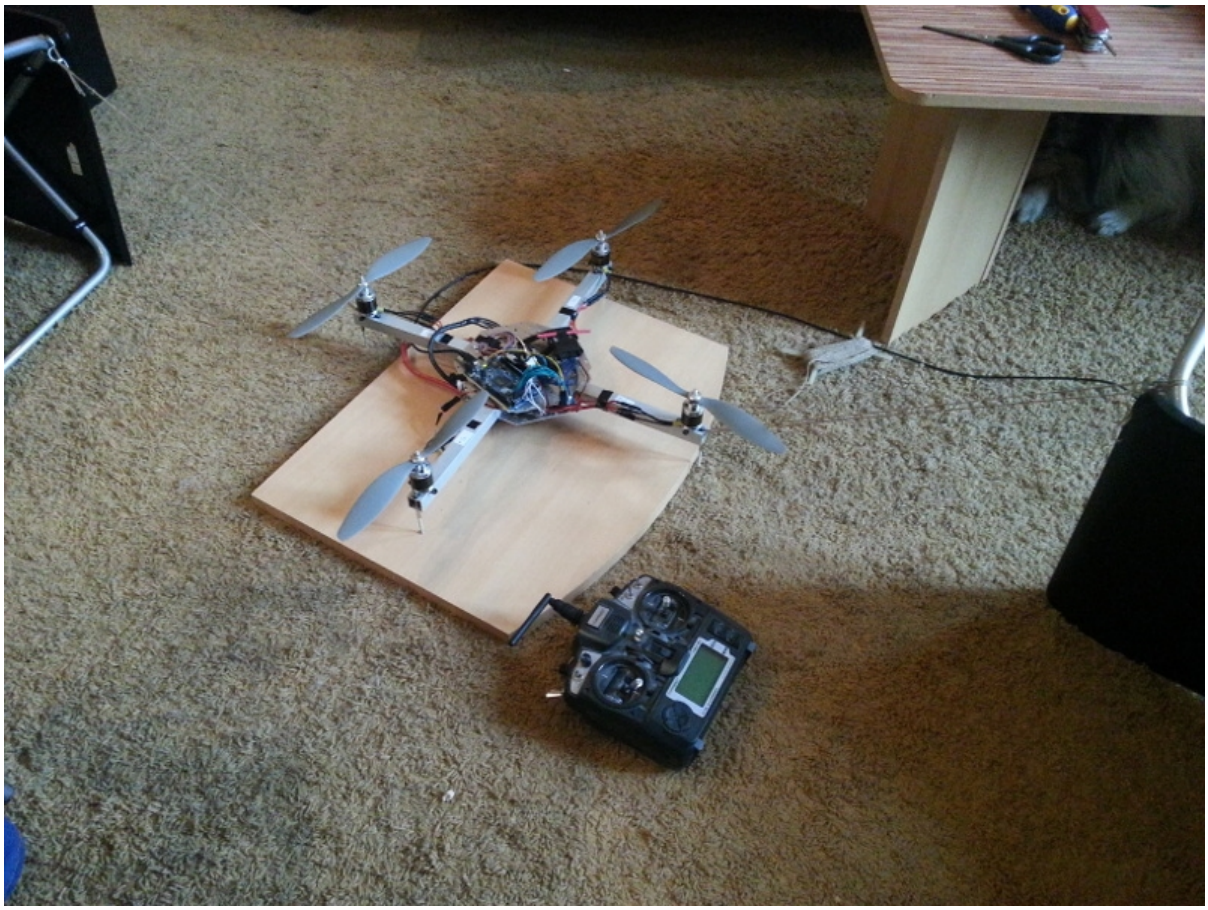
V průběhu konstrukce kvadrokoptéry bylo také potřeba programovat regulátory rychlosti. V hlavním souboru je k těmto účelům zakomentovaný kód, který jednoduše do všech ESC přeposílá signál na kanálu tahu. Více o tomto procesu v příloze číslo 2 na straně 7.

8 Testování reálného systému

8.1 Testovací metodika

K testování bylo zapotřebí vytvořit podmínky pro izolování otáčení na jednotlivé osy. Obrázek 19 zachycuje testovací zařízení pro osu z a obrázek 20 pro osy x nebo y.

Obrázek č. 19: zkušební závěs pro izolaci osy z



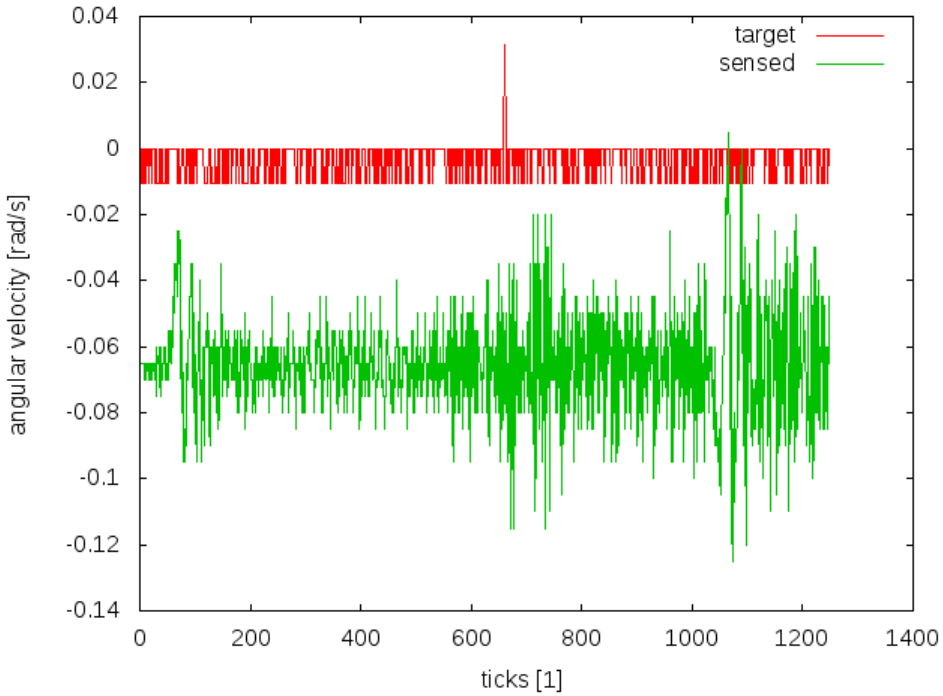
Obrázek č. 20: Zkušební závěs pro izolaci osy y

Po výpočetní stránce jsem musel udělat úpravy popsané v předposledním odstavci kapitoly Hlavní soubor. Data ze senzorů se do paměti zapisují, pokud je systém ve stavu ARMED. Ve stavu STANDBY se naopak uložené hodnoty vypisují na sériový port USB. Tento tok zachycuji programem Screen, který jej dokáže uložit do souboru. Výsledný soubor má v případě mého ladícího výpisu formát CSV. Hodnoty do grafu zanáším pomocí bashového skriptu volajícího program Gnuplot.

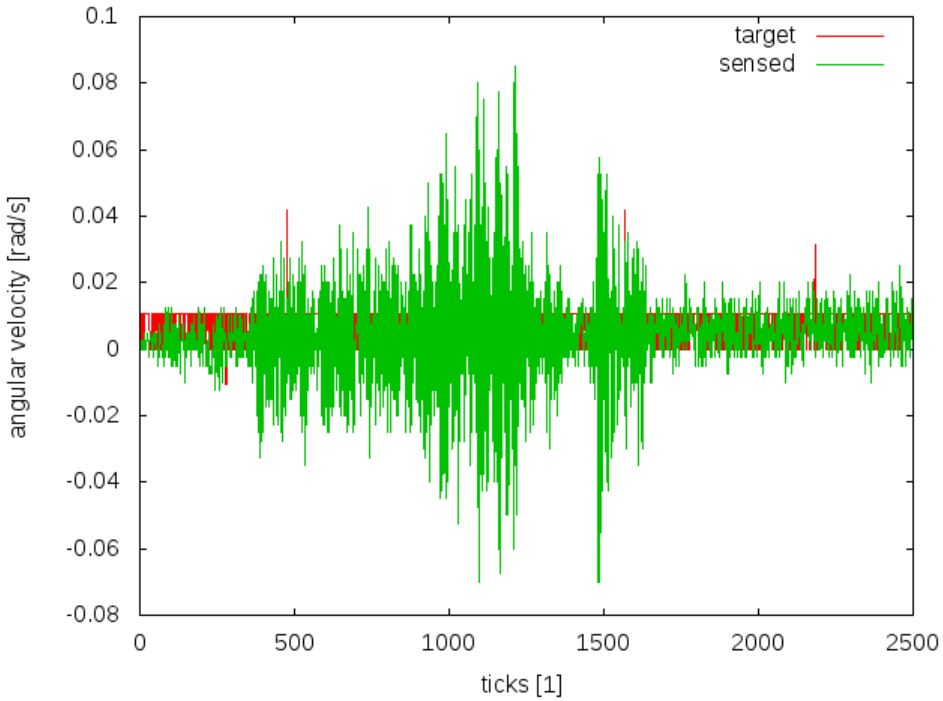
8.2 Test senzorické jednotky

Přestože jsou během startovací sekvence prováděny kalibrace senzorů, při testování jejich přesnosti pro nulový pohyb byly zjištěny výrazné nepřesnosti. Šum se dá vzhledem k obrovským vibracím způsobovaným prací motorů očekávat. Platí čím větší otáčky, tím vyšší magnituda šumu.

Obrázek číslo 21 popisuje odchylku od nulové hodnoty v měření úhlové rychlosti v ose x. Na obrázku 22 již můžeme vidět zlepšení po aplikaci přičítání korekční hodnoty. Stejná úprava se provádí i při měření natočení v ose x.

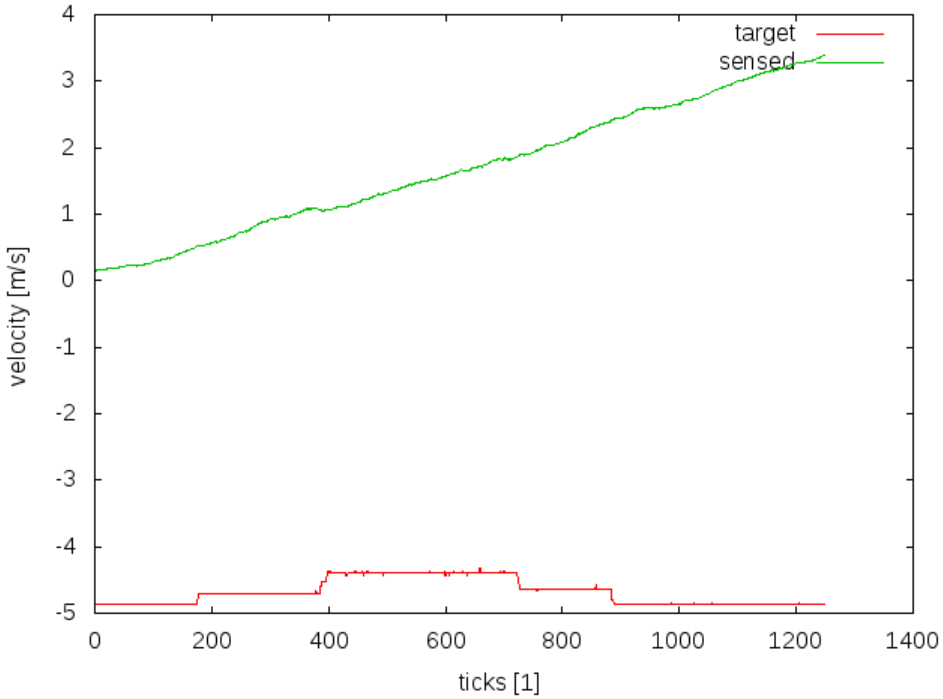


Obrázek č. 21: odchylka naměřené úhlové rychlosti v ose x od nulové hladiny (svislá osa je v tomto typu grafu v jednotkách uvedených v hranatých závorkách a vodorovná udává pořadí obrátky řídící smyčky, červená linka znamená požadovaný průběh (target) a zelená průběh naměřený (sensed))



Obrázek č. 22: měření stejné veličiny jako na obrázku 21 ovšem s aplikací korekce

Horší případ ovšem nastává při odhadování rychlosti. Na stránkách Ch Robotics [od5] je uživatel seznámen se skutečností, že spousta senzorů zahrnující UM 6 není určena k přesnému měření rychlosti. Na obrázku 23 je možno vidět, že chyba je vskutku veliká a rostoucí. Ikdyž je kvadrokoptéra statická, nedá se jednoduše korigovat. Pokročilejší způsoby zpřesňování senzorů bohužel překračují rámec této práce, proto nelze se stabilizací rychlosti stroje počítat.



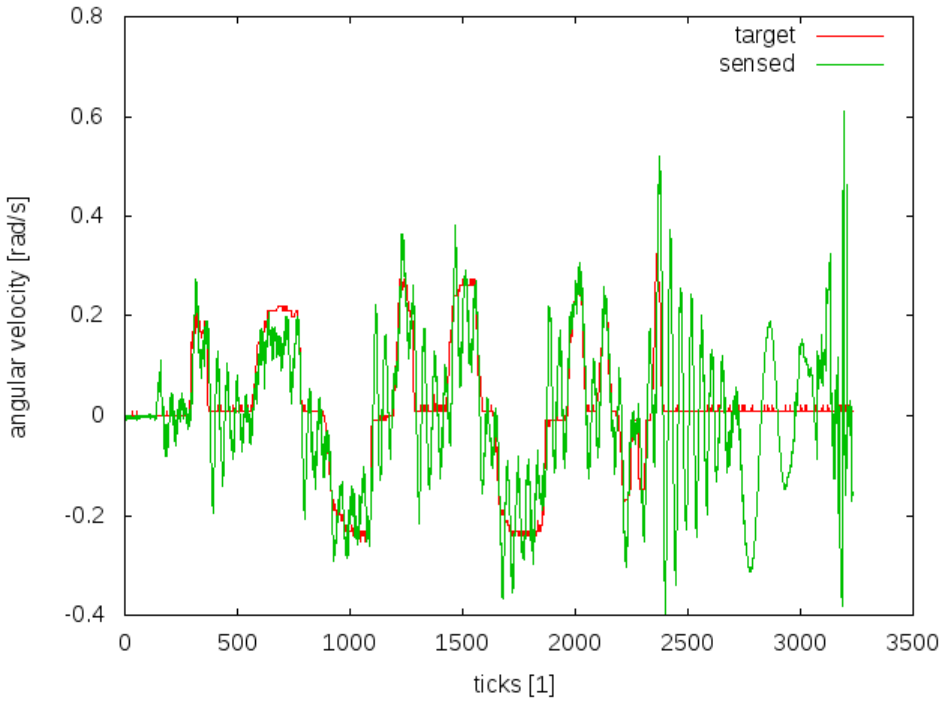
Obrázek č. 23: chyba odhadu rychosti v ose z od nulové hladiny

8.3 Hledání parametrů

Nejprve jsem řešil úlohu pro akrobatický mód. Po několika ukvapených nastavení tipovaných hodnot PID parametrů jsem začal problém řešit následujícím systémem, který zhruba platí pro všechny osy:

1. Z předchozích pokusů jsem znal alespoň řád konstant, tudíž jsem nastavil počáteční hodnoty: P = 0,05; I = 0,05; D = 0
2. Jelikož odezva kvadrokoptéry na pohyb kolíku vysílače byla pomalá, bylo jasné, že je třeba zvyšovat poměr P ku I.

3. Při zvyšování P za snižování I je od určitého momentu cítit, že změna je pro motory již dost rychlá a ty začnou při prudších pohybech kolíku ovladače cukat (ze stejného důvodu jsem vypustil složku D z řízení osy z, protože dle vydávaných zvuků bylo zřetelné, že náhlé změny akčních signálů jsou pro motory příliš velkou zátěží).
4. Složku I jsem po dostatečném snížení zachoval, přestože by složka P měla sama o sobě regulaci zvládnout. Osobně ale preferuji hladší průběhy změn, což právě I zajišťuje.
5. Při ladění os x a y se systém při náročnějších manévrech dostal do rezonance. Pro osu z se tento jev nevyskytoval, proto předpokládám, že na vině je zavěšení nosných lan, které bylo umístěno mimo úroveň těžiště. To způsobilo kromě rotačního pohybu také posun celého stroje kumulující kinetickou energii, která se uvolňovala v pohyb s pravidelnými intervaly ale s opačnou orientací. Na následujícím obrázku je možno vidět rozkmitání snímané úhlové rychlosti kolem cílových hodnot (úsek zhruba od 2 750. kroku se týká snášení stroje z napnutých lan zpět na zem). Nastavení D složky tento problém téměř odstranilo.
6. Výsledné hodnoty pro akrobatický mód jsou:
   a) pro osu x: P = 0,2; I = 0,0; D = 0,008
   b) pro osu y: P = 0,2; I = 0,0; D = 0,008
   c) pro osu z: P = 0,22; I = 0,0; D = 0,0



Obrázek 24: tlumené překmity sklonu při testování řízení úhlové rychlosti na závěsu

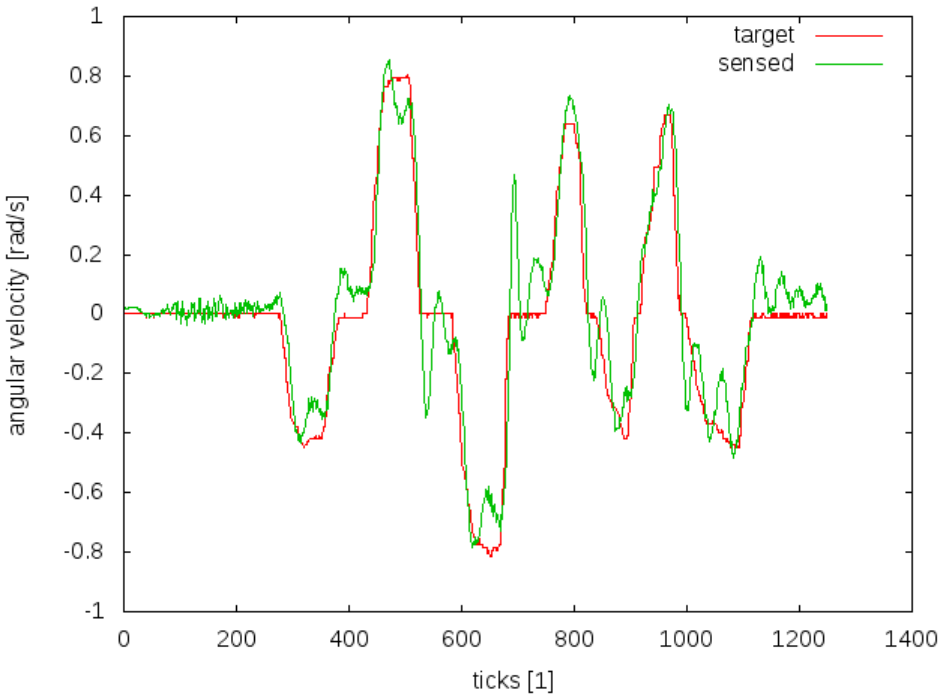
Pro transportní mód jsem na úrovni úhlové rychlosti použil stejné PID konstanty jako pro mód akrobatický. Postup pro získání nastavení regulátorů natočení je takový:

1. Nastavení vrstvy úhlových rychlostí jsem promítl do simulačního modelu.
2. Z praxe je známo, že stačí nastavení parametrů P a I.
3. Díky simulaci jsem získal přehled v jakém řádu mají být hodnoty PI nastaveny. Po nasazení jsem ještě snížil hodnotu složek P v obou regulátorech kaskády, jelikož alespoň na testovacím závěsu docházelo ke zvýraznění kmitání.
4. Výsledné hodnoty pro transportní mód jsou:
   a) regulátor úhlové rychlosti pro osu x: P = 0,2; I = 0,0; D = 0,008
   b) regulátor úhlové rychlosti pro osu y: P = 0,2; I = 0,0; D = 0,008
   c) regulátor úhlové rychlosti pro osu z: P = 0,22; I = 0,0; D = 0,0
   d) regulátor natočení pro osu x: P = 1,0; I = 0,015; D = 0,0
   e) regulátor natočení pro osu y: P = 1,0; I = 0,015; D = 0,0

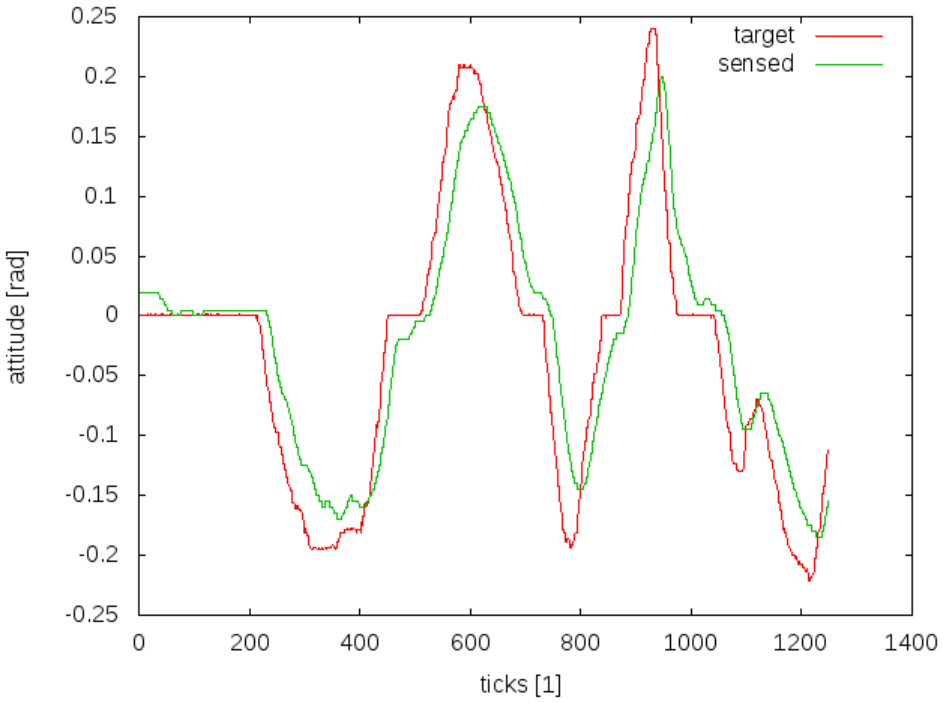
8.4 Záznamy letových dat

Bohužel hned při prvním testovacím letu jsem zazmatkoval a místo ubrání tahu motorů jsem celý systém na dálku deaktivoval. Kvadrokoptéra utrpěla újmu na pravém motoru a dokud nedorazí náhradní součástky, není schopna letu. Poskytnu tedy data naměřená na testovací aparatuře.

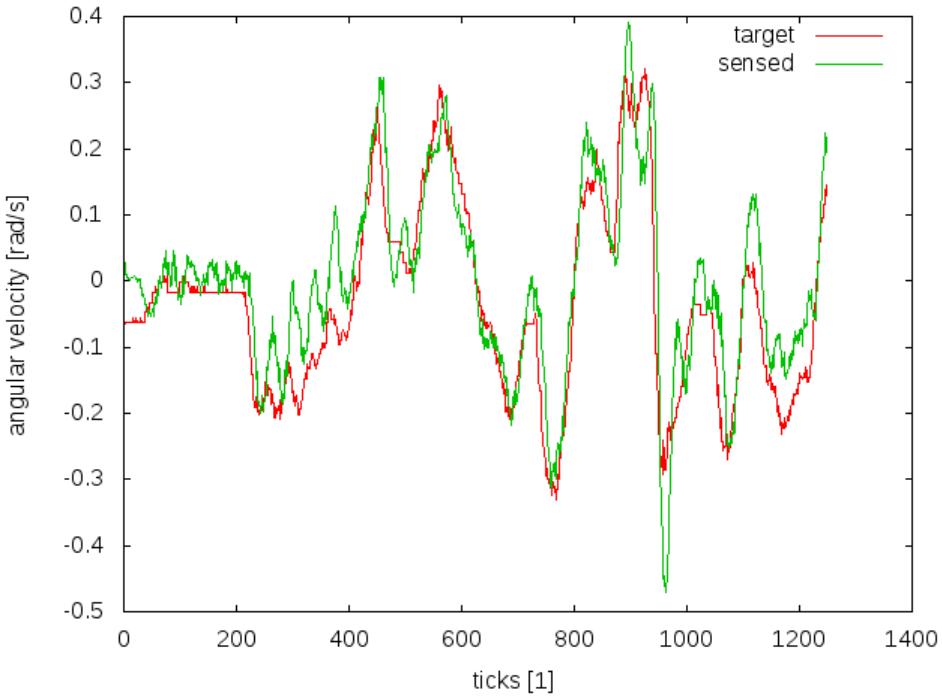
Za povšimnutí stojí zpoždění měřeného vůči požadovanému natočení na obrázku číslo 26. Interval je široký zhruba 20 obrátek řídící smyčky, což odpovídá 0,28 s.



Obrázek č. 25: regulace úhlové rychlosti v ose x



Obrázek č. 26: regulace natočení v ose x



Obrázek č. 27: regulace úhlové rychlosti v ose x při natáčení z předchozího obrázku

9 Testování modelu

Model v Simulinku netrpí nekontrolovatelným rozkmitáním závěsu, a dokonce jsem do něj neamplementoval šum senzorů. Proto model poskytuje výrazně lepší výsledky regulace.

Postup ladění je následující:

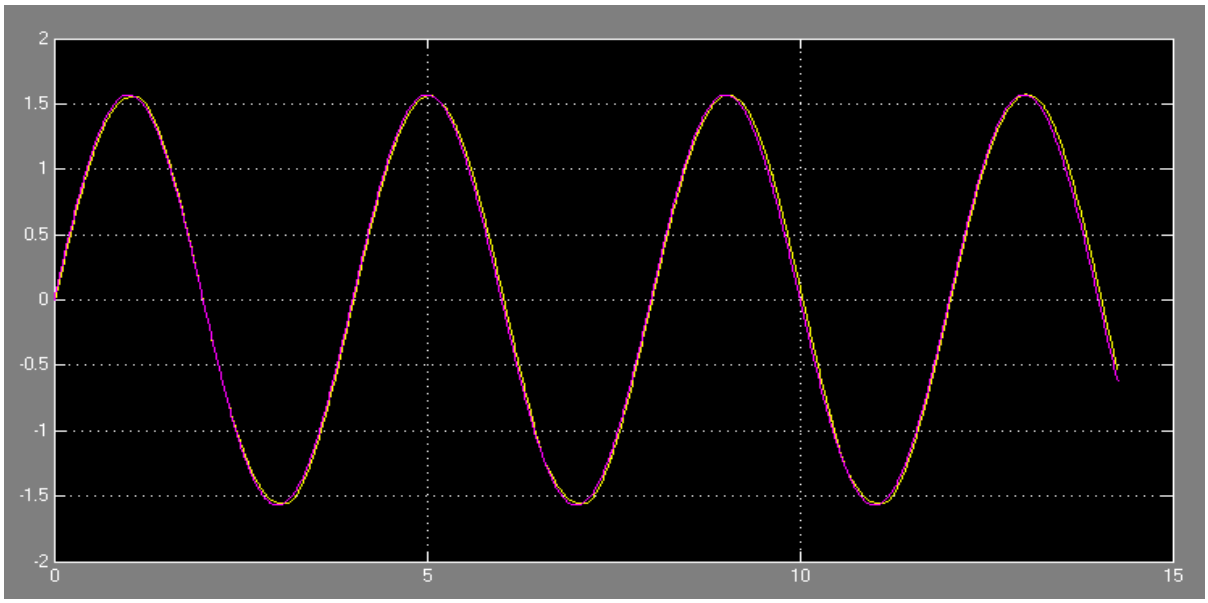
1. Původní nastavení pro akrobatický mód jsem aplikoval jak na regulátory akrobatického módu, tak na regulátory ve vrstvě úhlové rychlosti (dále spodní vrstva) v transportním módu.
2. Při experimentování s modelem jsem hlídal jednak konvergenci natočení a úhlové rychlosti ve všech osách, ale také kontakt napětí motoru s jeho limitem.
3. Nejdříve jsem zvyšoval P složku regulátorů ve vrstvě natočení (dále horní vrstva), dokud nezačalo být vidět, že to již nepřispívá k rychlosti konvergence.
4. Složka I horní vrstvy je nastavena na symbolickou hodnotu, aby v regulaci existoval určitý překmit, který by operátorovi, který chce nejspíše ovládat akceleraci, měl poskytnout jakousi náhradu za pomalejší nárůst natočení. V běžném letu tato složka pomáhá i mírnit projevy šumu senzorických dat a omezit vliv externích sil.
5. Po té jsem zvyšoval složku P ve spodní vrstvě. Teprve od hodnoty 0,6 se začala ustalovat i úhlová rychlost na druhé z vodorovných os. Přidával jsem dále, dokud jsem při nulové vertikální rychlosti nezaznamenal kontakt s limitem vstupního napětí motoru.
6. Konstanty regulátoru vertikální rychlosti jsem ponechal na odhadnutých hodnotách.
7. Výsledné hodnoty pro akrobatický mód:
   a) pro osu x: P = 0,8; I = 0,0; D = 0,0
   b) pro osu y: P = 0,8; I = 0,0; D = 0,0
   c) pro osu z: P = 0,22; I = 0,0; D = 0,0
8. Výsledné hodnoty pro transportní mód:
   a) regulátor úhlové rychlosti pro osu x: P = 0,8; I = 0,0; D = 0,0
   b) regulátor úhlové rychlosti pro osu y: P = 0,8; I = 0,0; D = 0,0
   c) regulátor úhlové rychlosti pro osu z: P = 0,22; I = 0,0; D = 0,0
   d) regulátor natočení pro osu x: P = 1,5; I = 0,015; D = 0,0
   e) regulátor natočení pro osu y: P = 1,5; I = 0,015; D = 0,0

Ke grafům zachycujícím regulace jednotlivých simulací bych rád zmínil jen následující poznatky:

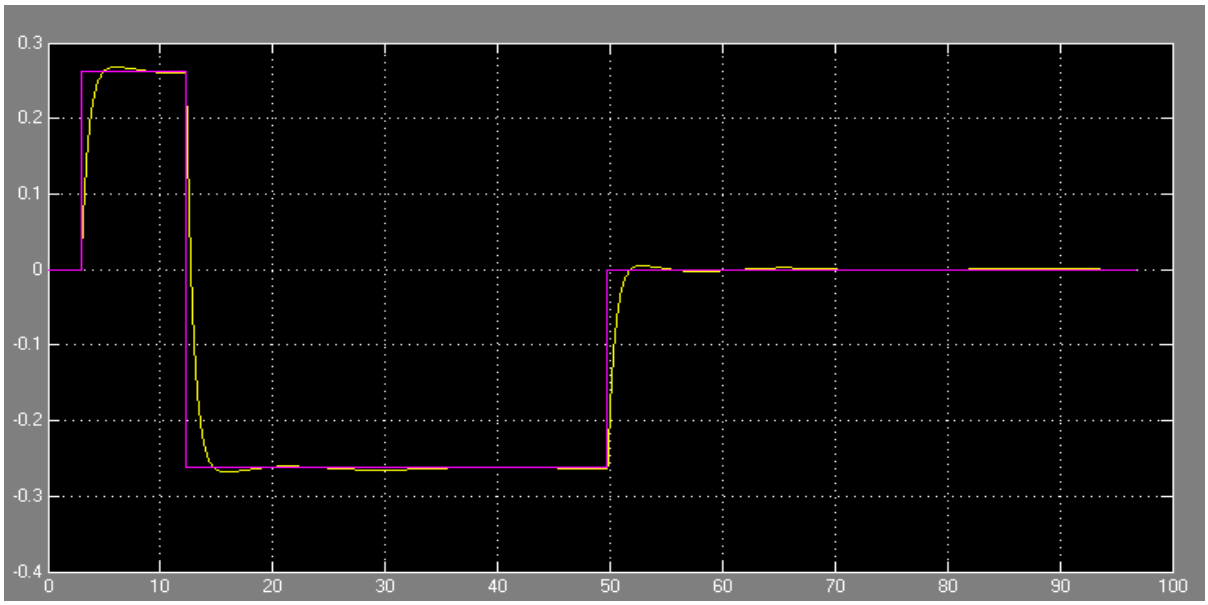
Zaprvé obrázek 31 ukazuje, jak úmyslné změny v jedné ose způsobují neúmyslné změny těch zbylých. Je to způsobeno tím, že odezva sil a momentů motorů na vstupní napětí není lineární. Řízení způsobem přidávání a odebírání voltáže antagonistickým aktuátorům ovšem lineární je.

Ze stejného důvodu jsou na obrázku číslo 33 vidět okamžité zvýšení rychlosti při náhlých změnách v požadované úhlové rychlosti v ose z.

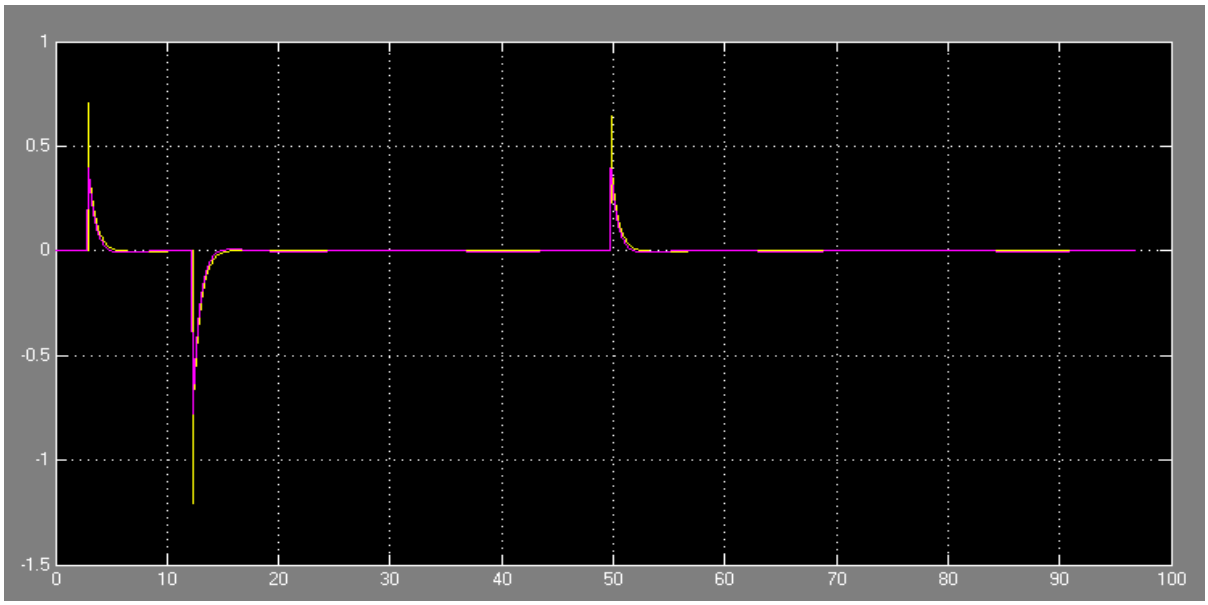
Na obrázku 33 je také viditelné, že určitou dobu zabere vlastní vyrovnání rychlosti po spuštění simulace – přechodu ze stavu bez tíže. Za dobu zhruba 4 sekund stroj klesne o necelý metr.



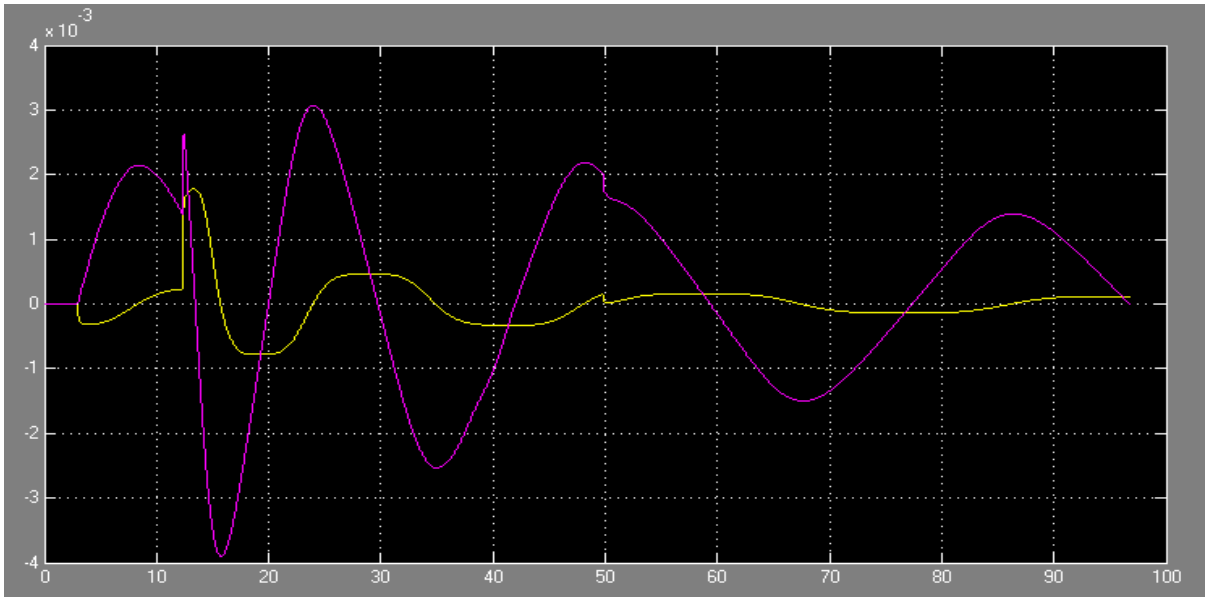
Obrázek č. 28: regulace úhlové rychlosti v ose y za vstupu funkce sinus
(svislá osa je v jednotkách [rad / s] a vodorovná v [s])



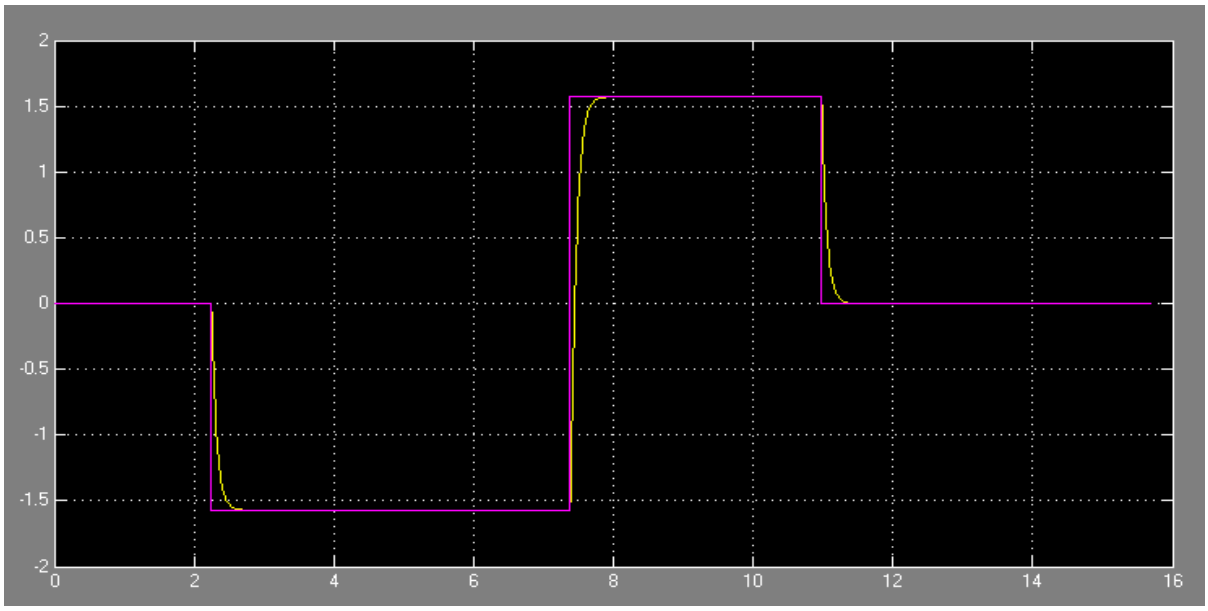
Obrázek č. 29: regulace natočení v ose x za vstupu skokových změn
(svislá osa je v jednotkách [rad] a vodorovná v [s])



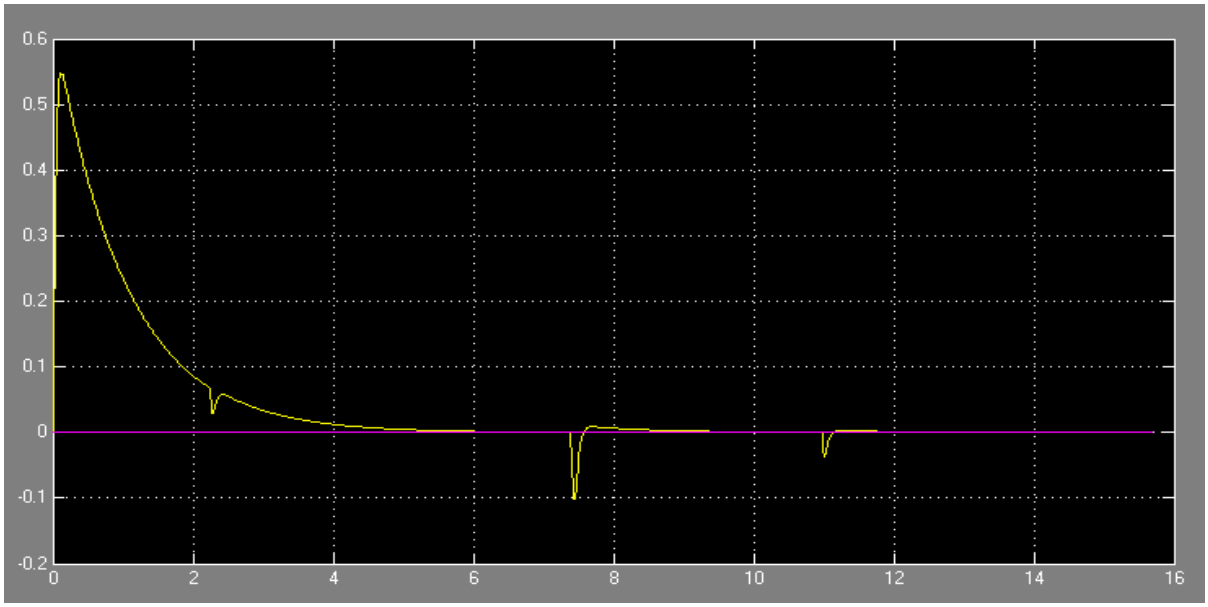
Obrázek č. 30: regulace úhlové rychlosti v ose x při natáčení z předchozího obrázku (svislá osa je v jednotkách [rad / s] a vodorovná v [s])



Obrázek č. 31: regulace úhlové rychlosti v ose y při změnách požadovaného natočení v ose x (svislá osa je v jednotkách [rad / s] a vodorovná v [s])



Obrázek č. 32: regulace úhlové rychlosti v ose z
(svislá osa je v jednotkách [rad / s] a vodorovná v [s])



Obrázek č. 33: regulace rychlosti v ose z při změnách požadované úhlové rychlosti v ose z z předchozího obrázku
(svislá osa je v jednotkách [m / s] (pozor - rychlost roste směrem dolů) a vodorovná v [s])

10 Závěr

Model v Simulinku bude stále představovat idealizované pojetí systému. Proto je srovnávání s reálnou kvadrokoptérou záležitostí úhlu pohledu.

Osobně jsem očekával, že parametry, které budou fungovat na testovacím závěsu budou blíže těm, které používá model. Zatím jsem přesvědčen, že nastavení regulátoru na závěsu spíše neodpovídá tomu pro běžné létání.

Přestože se grafy průběhů regulace pořízené kvadrokoptérou na první pohled dost liší od ideálu, ovládání z pohledu operátora bylo při testování dostačující. Odezva systému by měla být rychlejší, ale to se dá vysvětlit sníženými hodnotami P.

Nejdůležitější pro mne bylo pohodlné ovládání skutečné kvadrokoptéry, které zvládne i uživatel nováček, jako jsem já. Z tohoto pohledu musím projekt ZATÍM zhodnotit jako neúspěšný.

10.1 Možná vylepšení

10.1.1 Model

Simulační model jako takový je možno rozšiřovat a zpřesňovat o další poznatky týkající se fyzikálních zákonů. Zejména by mělo jít o zahrnutí dynamiky motoru související s jeho momentem setrvačnosti. Aerodynamika také není vyčerpávající, ale vzhledem k poměru složitosti problematiky a přínosu se nejedná o velký nedostatek.

10.1.2 Testovací metodika

Za hlavní problém testovací metodiky považuji zavěšení pro izolování os x nebo y. Kromě problému popsaném v kapitole 8.3 zde existuje ještě jeden byť zanedbatelný. Použitý závěs totiž brání v hladkém otáčení, protože se při něm lana objímající úchyt na kvadrokoptéře do sebe zaplétají. Tento odpor je ale pro pozorovatele neznatelný.

Vylepšení by spočívalo v použití pevné konstrukce, na kterou by kvadrokoptéra byla upevněna otočnými čepy s ložisky. Upevnění by mělo být ve výši vrtulí, což ve vertikálu zhruba odpovídá místu, za které visí kvadrokoptéra při běžném letu. Takto by se zcela odstranil vliv rozkmitání celého stroje na zavěšení mimo otáčení kolem aktuálně testované osy.

10.1.3 Reálný stroj

V této kapitole bych rád podotkl, že jsem byl zklamán nepřesností inerciální měřící jednotky UM 6. Do budoucnosti bych zkusil zvolit jiný senzor.

Možnosti filtrování omezuje výkon zvolené vývojové platformy Arduino. Výpočetní smyčka se opakuje s frekvencí 72 Hz, což nedává velký prostor pro nasazení sofistikovanějších filtrů senzorických dat. Původně jsem místo PID nasadit fuzzy regulátory, což by ovšem vedlo k markantnímu snížení frekvence řízení.

Senzor by ale také jistě pracoval lépe, pokud by se snížil vliv vibrací generovaných činností motoru. Jako hlavní opatření navrhuji oddělení motoru od konstrukce gumovou podložkou a vsazení IMU do pěnové stélky.

S přesnějším (mnoho-řádově) odhadem rychlostí by již nebyla otázkou stabilizace, ale preciznost ovládání, jak je dnes již standardem v komerčně užívaných strojích.

Použité součástky jako elektromotory a ESC jsou cenově dostupné a nejsou specializované na výrobu počítačem ovládaných strojů. Je tedy pravděpodobné, že rychlé změny signálů nedokáží adekvátně zpracovat, nebo že tyto změny snižují jejich životnost.

Na grafech průběhu regulace skutečného stroje je zřetelné, že signál z přijímače není stabilní a je také poměrně silně zašuměn. Lépe by zřejmě posloužila komunikace skrze Wi-Fi. Nasazení této technologie ovšem vede k vyšším nárokům na výkon.

Má nejdůležitější zkušenost je z oblasti bezpečnosti stroje. Nyní vím, že se konstruktér při prvních pokusech musí soustředit na to, aby letoun nedokázal ublížit sobě, natož svému uživateli. Proto do budoucna nezůstanu jen u nárazníků na podvozku, ale pasivní bezpečnostní prvky rozšířím pro ochranu vrtulí a svrchní strany kvadrokoptéry.

Literatura

[1] Breguet-Richet Gyroplane No.1. *Aviastar*. [online]. [cit. 2015-01-16]. Dostupné z: *http://www.aviastar.org/helicopters_eng/breguet_gyro.php*

[2] Oemichen. *Aviastar*. [online]. [cit. 2015-01-16]. Dostupné z: *http://www.aviastar.org/helicopters_eng/oemichen.php*

[3] Convertawings Model A. *Aviastar*. [online]. [cit. 2015-01-16]. Dostupné z: *http://www.aviastar.org/helicopters_eng/convertawings.php*

[4] Jim Bourke. Understanding Electric Power Systems part 6. *Control Tower*. [online]. 1.2.1999 [cit. 2015-01-16]. Dostupné z: *http://www.rcgroups.com/forums/showthread.php?t=393254*

[5] Emrah Kulunk (2011). Aerodynamics of Wind Turbines, Fundamental and Advanced Topics in Wind Power, Dr. Rupp Carriveau (Ed.), ISBN: 978-953-307-508-2, InTech, Dostupné z: *http://www.intechopen.com/books/fundamental-and-advanced-topics-in-wind-power/aerodynamics-of-wind-turbines*

[6] Tomas B. Co. Ziegler-Nichols Method. *Chemical Engineering | Michigan Technological University*. [online]. 13.2.2004 [cit. 2015-01-16]. Dostupné z: *http://www.chem.mtu.edu/~tbco/cm416/zn.html*

[7] Jake Abbott. State-Space Control Systems. *Telerobotics Lab.* [online]. 18.4.2013 [cit. 2015-01-16]. Dostupné z: *http://www.telerobotics.utah.edu/index.php/StateSpaceControl*

[8] Lucas M. Argentim, Willian C. Rezende, Paulo E. Santos a Renato A. Aguiar. PID, LQR and LQR-PID on a quadcopter platform. *2013 International Conference on Informatics, Electronics and Vision (ICIEV)*. IEEE, 2013, s. 1-6. DOI: 10.1109/ICIEV.2013.6572698.

[9] Jindřich Chaloupek. Řízení aktivního tlumení pomocí metody H∞. Praha, 2009. Dostupné z: *https://support.dce.felk.cvut.cz/mediawiki/images/d/df/Dp_2009_chaloupek_jindrich.pdf*. Diplomová práce. České vysoké učení technické.

[10] Ben Greer. Quadcopter Stability and Neural Networks. *Lab Notebook.* [online]. 21.5.2014 [cit. 2015-01-16]. Dostupné z: *http://www.gperco.com/2014/05/quadcopter-stability-and-neural-networks.html*

[11] Michael J. Grimble. *Robust industrial control systems: optimal design approach for polynomial systems*. Hoboken, NJ: Wiley, c2006, s. 443-456. ISBN 9780470020739.

[12] Peter Vorobieff. From ideal fluid flow to understanding lift and drag on wings. *Department of Mathematics and Statistics | The University of New Mexico.* [online]. [cit. 2015-01-16]. Dostupné z: *http://www.math.unm.edu/mctp/summer/lecturenotes/fluids/lectures_vorobieff.pdf*

[13] Understanding Euler Angles. [online]. [cit. 2015-04-25] Dostupné z: *http://www.chrobotics.com/library/understanding-euler-angles*

[14] Understanding Quaternions. [online]. [cit. 2015-04-25] Dostupné z: *http://www.chrobotics.com/library/understanding-quaternions*

Obrázky

[ob1] *http://www.mathworks.com/help/aeroblks/bodycs.gif*

[ob2] *http://users.skynet.be/sky92472/Pictures/Q&A/Torque.jpg*

Odkazy

[od1] *http://www.grc.nasa.gov/WWW/k-12/airplane/foil3.html*

[od2] *http://www.mathworks.com/matlabcentral/fileexchange/3630-simulink-keyboard-input/content/sfun_keyboard_input_v1_01.m*

[od3] *http://m-selig.ae.illinois.edu/props/propDB.html*

[od4] *http://www.jentronics.com/fgfs/temp/nmine.zip*

[od5] *http://www.chrobotics.com/library/accel-position-velocity*

[od6] *http://www.mathworks.com/help/aerotbx/ug/quatrotate.html*

Seznam příloh

Příloha 1. DVD: Výsledný simulační model (soubor s hlavním modelem je quadcopter.slx)
Příloha 2. Popis pracovního postupu tvorby fyzikálního systému
Příloha 3. DVD: Archiv .zip s výsledky testování