ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta dopravní
Ústav letecké dopravy

Návrh kontrolního algoritmu změny letu hybridního UAV

Control algorithm design of hybrid UAV

Diplomová práce

Studijní program: Technika a technologie v dopravě a spojích

Studijní obor: Provoz a řízení letecké dopravy

Vedoucí práce: Ing. Stanislav Kušmírek

Bc. Bohuslav Vladyka

Praha 2023

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta dopravní
děkan
Konviktská 20, 110 00 Praha 1

K621.....Ústav letecké dopravy

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení studenta (včetně titulů):

Bc. Bohuslav Vladyka

Studijní program (obor/specializace) studenta:

navazující magisterský – PL – Provoz a řízení letecké dopravy

Název tématu (česky): **Návrh kontrolního algoritmu změny letu hybridního UA**

Název tématu (anglicky): Design of Hybrid UA Flight Change Control Algorithm

Zásady pro vypracování

Při zpracování diplomové práce se řiďte následujícími pokyny:

- Cílem práce je navrhnout specifickou část PID kontroléru hybridního UA s konstrukčním uspořádáním kvadrokoptéra – letadlo s pevným křídlem umožňující změnu vertikálního letu na horizontální.
- Vypracujte analýzu technické proveditelnosti podle současných aktuálních řešení hybridních bezpilotních letadel s vertikálním vzletem a následným horizontálním pohybem. Najděte obdobné modely ve 3D robotickém simulátoru.
- Vytvořte nebo najděte vhodný existující model hybridního bezpilotního letadla v Gazebo robotickém simulátoru pro aplikování navrženého kontroléru.
- Navrhněte algoritmus kontroléru schopného změny vertikálního letu na horizontální a reverzně z horizontálního letu na vertikální.
- Verifikujte vytvořený algoritmus kontroléru v prostředí Gazebo s kontrolérem hybridního bezpilotního letadla používaný platformou Pixhawk.
- Stanovte limitace navrhovaného kontroléru a formulujte závěry práce.

Rozsah grafických prací: dle pokynů vedoucího diplomové práce

Rozsah průvodní zprávy: minimálně 55 stran textu (včetně obrázků, grafů a tabulek, které jsou součástí průvodní zprávy)

Seznam odborné literatury: Papa, U., Ponte, S. and Del Core, G., Conceptual design of a small hybrid unmanned aircraft system. Journal of Advanced Transportation, 2017.
Meng, W., Hu, Y., ROS+ unity: An efficient high-fidelity 3D multi-UAV navigation and control simulator in GPS-denied environment

Vedoucí diplomové práce: **Ing. Stanislav Kušmírek**

Datum zadání diplomové práce: **15. července 2022**
(datum prvního zadání této práce, které musí být nejpozději 10 měsíců před datem prvního předpokládaného odevzdání této práce vyplývajícího ze standardní doby studia)

Datum odevzdání diplomové práce: **15. května 2023**
a) datum prvního předpokládaného odevzdání práce vyplývající ze standardní doby studia a z doporučeného časového plánu studia
b) v případě odkladu odevzdání práce následující datum odevzdání práce vyplývající z doporučeného časového plánu studia

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE Fakulta dopravní L. S.

doc. Ing. Jakub Kraus, Ph.D.
vedoucí
Ústavu letecké dopravy

prof. Ing. Ondřej Přibyl, Ph.D.
děkan fakulty

Potvrzuji převzetí zadání diplomové práce.

Bc. Bohuslav Vladyka
jméno a podpis studenta

V Praze dne 15. července 2022

Abstrakt

Tato diplomová práce je zaměřena na návrh a optimalizaci specifické části kontrolního řídicího algoritmu pro hybridní bezpilotní letoun s konstrukčním uspořádáním, které kombinuje výhody kvadrokoptéry a samokřídla. To představuje unikátní konstrukci, která umožňuje přechod z vertikálního letu na horizontální a naopak, čímž zvyšuje všestrannost a možnosti využití UAV. Cílem je využít teoretických základů, současných technologií a inovativních nástrojů pro vývoj a testování efektivního kontrolního systému. Práce se významně zaměřuje na analýzu technické proveditelnosti současných řešení hybridních bezpilotních prostředků s vertikálním vzletem a následným horizontálním pohybem, včetně identifikace a studie obdobných modelů ve 3D robotickém simulátoru. Tento proces zahrnuje důkladné prozkoumání různých technologických a konstrukčních aspektů, které ovlivňují letové charakteristiky. Následně pomocí těchto poznatků je vybrán vhodný existující model hybridního letového prostředku v Gazebo robotickém simulátoru. Tento model je pak využit pro aplikaci a testování navrženého kontroléru, což umožňuje provést efektivní a podrobnou analýzu výsledků a výkonu kontroléru.

Klíčová slova: Hybridní UAV, PID kontrolér, Ziegler-Nichols metoda, ROS, Gazebo

Abstract

This thesis focuses on the design and optimization of a specific part of the PID controller for a hybrid UAV with a structural arrangement that combines the advantages of a quadcopter and a self-wing. This represents a unique design that allows the transition from vertical to horizontal flight and vice versa, thus increasing the versatility and potential of the UAV. The aim is to use theoretical foundations, current technologies and innovative tools to develop and test an effective control system. The work significantly focuses on the analysis of the technical feasibility of current hybrid UAV solutions with vertical take-off followed by horizontal movement, including the identification and study of similar models in a 3D robotic simulator. This process involves a thorough investigation of various technological and design aspects that affect flight characteristics. Subsequently, using this knowledge, a suitable existing model of the hybrid flight vehicle is selected in the Gazebo robotic simulator. Then this model is used for the application and testing of the proposed controller, allowing for an efficient and detailed analysis of the controller's results and performance.

Keywords: Hybrid UAV, PID Controller, Ziegler-Nichols method, ROS, Gazebo

Poděkování

V tomto místě bych rád poděkoval vedoucímu své diplomové práce Ing. Stanislavu Kušmírkovi, na kterého jsem se během studia a tvorby diplomové práce mohl kdykoliv obrátit a který se mě vždy snažil nasměrovat správným směrem. Touto cestou bych chtěl poděkovat lidem, kteří utvářeli zpětnou vazbu v rámci prezentování aktuálních výsledků. Mimo tato poděkování bych se chtěl nejvíce zaměřit na poděkování, rodině a kamarádům, kteří mě v průběhu celého studia podporovali a bez kterých by tato práce nemohla vzniknout. Speciální poděkování patří mé přítelkyni, která mi vždy poskytla opěrný bod, při řešení negativních situací, které přišly během zpracovávání této diplomové práce.

Čestné prohlášení

Prohlašuji, že jsem diplomovou práci s názvem „Návrh kontrolního algoritmu změny letu hybridního UAV“ vypracoval samostatně a použil k tomu úplný výčet citací použitých pramenů, které uvádím v seznamu přiloženém k bakalářské/diplomové práci.

Nemám závažný důvod proti užití tohoto školního díla ve smyslu §60 Zákona č.121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

Praze dne 15. května 2023 ……

Obsah

Úvod ... 12
1 Analýza proveditelnosti ... 13
1.1 Generování vztlaku u letadel a bezpilotních prostředků ... 13
1.1.1 Bernoulliho princip ... 14
1.1.2 Úhel náběhu ... 15
1.2 Technologické řešení generování vztlaku u bezpilotních prostředků ... 15
1.2.1 Rotorové mechanismy ... 16
1.2.2 Tiltrotorové mechanismy ... 16
1.2.3 Vektorování tahu ... 17
1.3 Hybridní bezpilotní prostředek ... 18
1.4 Vývojové typy hybridních bezpilotních prostředků ... 20
1.5 Technologické využití hybridních bezpilotních prostředků v praxi ... 22
1.6 Kontrolní řízení hybridních bezpilotních prostředků ... 25
1.6.1 PID kontrolér ... 27
1.6.2 Ladění kontrolního algoritmu ... 28
1.7 Softwarové nástroje pro vytváření a validaci kontrolního algoritmu ... 29
1.8 Validace výkonnosti algoritmu řízení ... 30
1.9 Shrnutí teoretické části práce ... 31
2 Metodologie ... 33
2.1 Gazebo simulátor ... 33
2.1.1 ROS ... 34
2.1.2 QGroundControl ... 36
2.1.3 Ověření konzistence enginu ... 37
2.1.4 Výběr konkrétního 3D modelu ... 37
2.2 Implementace kontrolního algoritmu ... 38
2.2.1 PID kontrolér ... 38
2.2.2 Metody ladění kontrolního algoritmu ... 40
2.3 Metodologické přístupy k vytváření modifikovaného řídicího algoritmu ... 44
2.3.1 Kvalifikace letových intervalů ... 44
2.3.2 Návrh modifikovaného řídicího algoritmu ... 45
2.4 Postup pro spuštění validace a ladění řídicích systémů ... 50
2.5 Experimentální postup pro testování a analýzu řídicího systému ... 51
2.5.1 Porovnání účinnosti pomocí parametrů mezi jednotlivými lety ... 52
3 Prezentace výsledků ... 53

3.1 Ověření konzistence simulovaných letů 53
3.2 Referenční let 54
3.3 Modifikovaný let 55
3.3.1 Modifikace PID kontroléru 55
3.3.2 Simulace modifikovaného letu 56
3.4 Komparace letů 57
3.4.1 Analýza a popis letu v intervalu B 58
3.4.2 Analýza a popis letu v intervalu D 61
3.4.3 Souhrn výsledků z obou intervalů a porovnání řídicích systémů 64
3.4.4 Porovnání vývoje výšky na intervalu C 65
Diskuze 67
Závěr 69

Seznam obrázků

Obrázek 1: Bernoulliho princip na profilu křídla [5] 14
Obrázek 2: Úhel náběhu [6] 15
Obrázek 3: Bell Boeing V-22 Osprey [11] 17
Obrázek 4: F-35B Lightning II [12] 17
Obrázek 5: Bezpilotní prostředek s přestavitelnou konfigurací rotorů [20] 19
Obrázek 6: Bezpilotní prostředek s pevnou konstrukcí rotorů [77] 20
Obrázek 7: Tiltrotorový bezpilotní prostředek [23] 21
Obrázek 8: Novlit 3 tailsitter [24] 21
Obrázek 9: DeltaQuad Pro [25] 22
Obrázek 10: Koncept Zuri 2.0 [29] 23
Obrázek 11: Neoptera Aero eOpter [31] 24
Obrázek 12: ALTI Transition [33] 24
Obrázek 13: Flight Stack 25
Obrázek 14: Chování PID kontroléru v rámci tepelné realizace [45] 27
Obrázek 15: Simulační schéma komunikace [40] 34
Obrázek 16: Systém publish – subscribe, využívající konkrétní topics [52] 35
Obrázek 17: Plánování letové mise v prostředí QGroundControl [57] 36
Obrázek 18: 3D model HIL Standard VTOL [65]: 38
Obrázek 19: Automatické ladění v prostředí QGroundControl [67] 40
Obrázek 20: Postup manuálního ladění při získávání hodnot ZN 42
Obrázek 21: Rozčlenění letu v intervalech 45
Obrázek 22: Implementace třídy PIDController 47
Obrázek 23: Vývojový diagram pro Vzlet - B 49
Obrázek 24: Porovnání konzistence referenčních letů 53
Obrázek 25: Průběh výšky v čase u výchozího letu 54
Obrázek 26: Modifikované simulační schéma komunikace 55
Obrázek 27: Průběh výšky v čase u modifikovaného letu 56
Obrázek 28: Porovnání průběhu celý letů 57
Obrázek 29: Průběh dosahování letové výšky na intervalu B 58
Obrázek 30: Závislost vertikální rychlosti v čase na intervalu B 59
Obrázek 31: Závislost klopení v čase na intervalu B 59
Obrázek 32: Závislost klonění v čase na intervalu B 60
Obrázek 33: Závislost výšky v čase na intervalu D 61
Obrázek 34: Závislost vertikální rychlosti v čase na intervalu D 62

Obrázek 35: Závislost klopení v čase na intervalu D 62
Obrázek 36: Závislost klonění v čase na intervalu D 63
Obrázek 37: Závislost výšky na čase na intervalu C 65

Seznam tabulek

Tabulka 1: Hodnoty PID kontroléru získané automatickým laděním 54
Tabulka 2: Získané hodnoty Ku a Tu 56
Tabulka 3: Hodnoty PID kontroléru získané manuálním laděním a implementací ZN 56
Tabulka 4: Získané hodnoty výkonnostních ukazatelů na intervalu B 60
Tabulka 5: Získané hodnoty výkonnostních ukazatelů na intervalu D 63
Tabulka 6: Porovnání výkonnostních ukazatelů na intervalu B 64
Tabulka 7: Porovnání výkonnostních ukazatelů na intervalu D 65

Seznam tabulek

Příloha 1: Skript řídicího algoritmu 77

Úvod

Rozšiřování leteckého průmyslu sebou přináší řadu inovací a technologií, kde je kladen velký důraz na výkonnost a bezpečnost. Využití bezpilotních prostředků je stále rostoucím trendem, který počítá i s integrací do hustě obydlených oblastí. Průnik bezpilotních prostředků vede do různých oblastí, díky výhodám jako jsou snížení ekonomických nákladů a zmírnění dopadu na životní prostředí. Bez ohledu na konkrétní použití je integrace bezpilotních prostředků do provozu vedena řadou specifických opatření, včetně dodržování zákonů, plánování letových tras a zajištění bezpečnosti.

Jednou z hlavních výzev, při vývoji hybridních bezpilotních prostředků, je složitost hardwarových a softwarových komponent, které jsou využívány pro řízení letu. V případě hardwarových komponent, je třeba využívat takové komponenty, které vedou ke spolehlivé funkcionalitě v každém letovém režimu. Softwarové systémy pro řízení lety jsou rovněž složitými systémy hybridních bezpilotních prostředků. Systémy řízení musí spolehlivě reagovat na velké množství přijímaných informací v reálném čase, a to včetně dat ze senzorů a letových pokynů z řídicí stanice. Kromě toho musí řídicí softwarové systémy obsahovat kontrolní algoritmy pro řízení letu, které umožňují včasně reagovat na letové situace. Vývoj softwarových systémů vyžaduje značné porozumění letové dynamice a výpočetní technice.

V procesu tvorby kontrolních algoritmů pro hybridní bezpilotní prostředky je optimální řízení přechodu mezi vertikálním a horizontálním letem zásadním aspektem. Přechod mezi vertikálním a horizontálním letem představuje účinné a flexibilní chování hybridního bezpilotního. Právě tímto přechodem je vytyčen hlavní přínos diplomové práce, v návrhu a ověření implementace kontrolního algoritmu, který je schopen zvládnout efektivně a bezpečně změnu mezi vertikálním a horizontálním letem.

Optimální výkonnost kontrolních algoritmů během letu a během přechodové fáze, spočívá ve zvolení správné strategie a přístupu úpravy kontrolního algoritmu. Vývojáři mají možnost si vybírat z řady nástrojů a technologií, které pozitivním směrem ovlivňují výstup kontrolního algoritmu, včetně ladění.

Cílem diplomové práce je návrh a implementace kontrolního algoritmu, který využívá přímé komunikace s PX4 autopilotem. K dosažení cíle je zpracována analýza, zaměřena na integraci ladících technologií uvnitř kontrolního algoritmu.

1 Analýza proveditelnosti

Pro implementaci bezpilotních prostředků do běžného provozu je klíčovým krokem řešit minimalizaci rizik, které znamenají pro okolní objekty, jako jsou budovy, vozidla a další letadla, nebezpečí. Následující body jsou důležité k zachování bezpečnosti při integraci bezpilotních prostředků do provozu [1]:

- Dodržování zákonů a platných předpisů
- Plánování letových tras, včetně zohlednění překážek
- Používání technologií, které zajišťují bezpečnost, například systémy detekce překážek
- Spolupráce s ostatními uživateli provozu
- Certifikace a monitoring provozu bezpilotních prostředků
- Stabilní podmínky provozu

Zajištění bezpečnosti dronů během jejich provozu je základním kritériem, které slouží ke zmírnění potenciálních rizik a ochraně přilehlých objektů. Pokud jde o provoz, lze také podrobně prozkoumat faktory, které přispívají k vytváření vertikální vzestupné síly, běžně označované jako "vztlak", a to jak u letadel, tak u bezpilotních prostředků. Aerodynamický princip, na němž je založeno vytváření vztlaku u letadel, spočívá v rychlém pohybu proudícího vzduchu nad povrchem křídla. V souvislosti s bezpilotními letouny s vertikálním vzletem je důležité zohlednit jejich rozměry a specifické letové režimy, které umožňují nastavování rotorů ve směru pohybu. Tyto letové režimy a konstrukční prvky, jako jsou tvar a velikost rotorů či křídel, mají zásadní vliv na schopnost generovat dostatečný vztlak při vertikální či horizontálním letu. [1] [2]

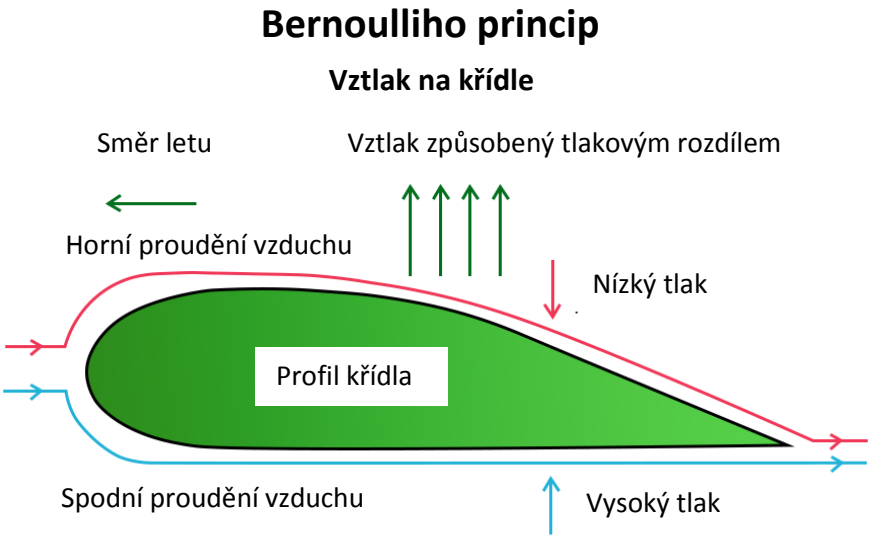
1.1 Generování vztlaku u letadel a bezpilotních prostředků

Generování vztlaku představuje klíčovou složku vzdušného výkonu konvenčních pilotovaných letadel i bezpilotních letounů s možností vertikálního vzletu a přistání. Navzdory skutečnosti, že obě kategorie dopravních prostředků jsou pro dosažení letu závislé na funkci vztlaku, jejich charakteristické mechanismy a odpovídající technologie se mohou výrazně lišit.

Základní principy, na nichž je založena produkce vztlaku, představují základní řešení, které jsou nedílnou součástí pochopení mechanismů, díky nimž letadla i bezpilotní prostředky s vertikálním vzletem a přistáním (VTOL) dosahují manévrovatelnosti ve vzduchu a udržují let. Základní principy důležité pro studium aerodynamiky zahrnují Bernoulliho princip, který vysvětluje vztah mezi tlakem vzduchu a rychlostí, a úhel náběhu, který je reprezentován úhlovou odchylkou mezi linií křídla a relativním větrem. Oba principy hrají zásadní roli při regulaci vztlaku, což v konečném důsledku vede ke stabilnímu a efektivnímu letu běžných letadel i bezpilotních leteckých prostředků se svislým vzletem a přistáním. [3]

1.1.1 Bernoulliho princip

Bernoulliho princip (obr. 1) je základním pojmem v oblasti dynamiky tekutin, kdy zvýšení rychlosti proudění (v tomto konkrétním případě vzduchu) vyvolá výsledný pokles tlaku.

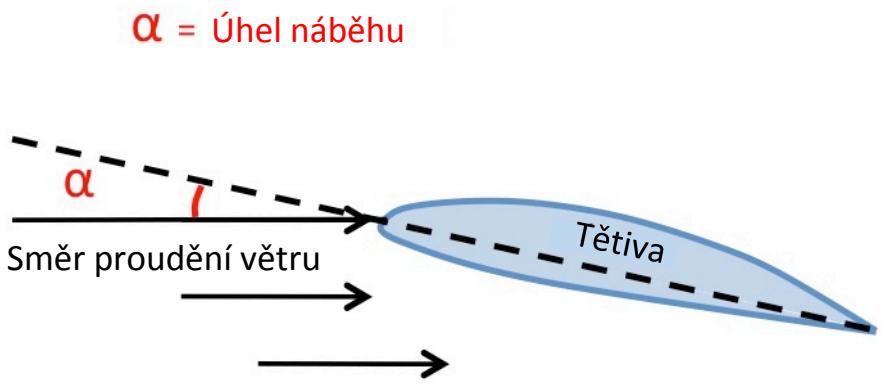


Obrázek 1: Bernoulliho princip na profilu křídla [5]

V běžných letadlech je konfigurace křídla vytvořena tak, aby vznikl rozdílný tlak mezi horní a dolní rovinou křídla. Jak proud vzduchu prochází křídlem, obrys horní plochy generuje zvýšení rychlosti, což vede ke snížení tlaku, a naopak srovnatelně rovinná spodní plocha zůstává na relativně vyšší úrovni tlaku. Rozdíl tlaku, který vzniká mezi horním a dolním povrchem křídla, vyvolává aerodynamickou sílu směřující vzhůru, běžně označovanou jako vztlak, která umožňuje letadlu odlepit se od země a vstoupit do vzdušného prostoru. Podobně i bezpilotní letadla s kolmým vzletem a přistáním vytvářejí vztlak pomocí řízení tlaku vzduchu. Systémy založené na rotorech, konkrétně kvadrokoptéry, využívají rotujících lopatek k vyhánění vzduchu směrem dolů, což vede ke vzniku rozdílu v tlaku vzduchu mezi oblastmi nad a pod rotory. Systémy s vektorováním tahu zahrnují přesný směr tahu generovaného motorem, aby bylo dosaženo vztlaku letadla. Ačkoliv se konkrétní metody generování vztlaku mohou lišit na konstrukci letadla, základní princip vytváření vztlaku vychází na Bernoulliho zákonu, která popisuje vztah mezi rychlostí proudění vzduchu a tlakem. [3] [4]

1.1.2 Úhel náběhu

Úhel náběhu má význam v leteckém inženýrství a avionice a představuje úhel tvořený linií křídla, což je pomyslná čára spojující náběžnou a odtokovou hranu křídla, a relativním větrem, který označuje směr proudění vzduchu vzhledem ke křídlu. Letadla i bezpilotní prostředky využívají k vytvoření vztlaku aerodynamické síly. Tato síla vzniká při působení vzduchu na křídlo nebo právě na rotor bezpilotního prostředku. Křídla jsou navržena tak, aby vytvářela sílu vztlaku, když se letadlo pohybuje rychle dopředu. [4]



Obrázek 2: Úhel náběhu [6]

S rostoucím úhlem náběhu křídla (obr. 2) dochází k současnému nárůstu rozdílu tlaku mezi horní a dolní plochou křídla, což způsobuje následné zesílení generování vztlaku. Pokud však úhel náběhu překročí určitou prahovou hodnotu, proud vzduchu se může odpojit od horního povrchu křídla, čímž dojde k přetažení. Při přetažení se vztlaková síla letadla výrazně sníží, čímž se zvýší pravděpodobnost ztížení řízení a potenciální kolize. U bezpilotních prostředků, jako jsou kvadrokoptéry, je vztlak generován tím, že je vzduch vyháněn pod rotory, což způsobuje tlakový rozdíl mezi oblastmi nad a pod nimi. Vztlak u bezpilotních prostředků závisí na naklopení lopatek rotorů a tahu motoru, které ovlivňují rychlost a směr proudu vzduchu. Změny efektivního úhlu náběhu a generování vztlaku lze dosáhnout změnou směru tahu v systémech s vektorováním tahu. [3]

1.2 Technologické řešení generování vztlaku u bezpilotních prostředků

Generování vztlaku se vztahuje k metodikám a vědeckým pokrokům používaných u bezpilotních letadel se schopností vertikálního vzletu a přistání, kde konečný cíl je zahájení a udržení vzdušného pohybu. Využívají se různé techniky, jako je vektorování tahu nebo rotorové mechanismy, k vytvoření vztlaku prostřednictvím manipulace s tlakem vzduchu za účelem vytvoření síly směřující vzhůru. Účinné vytváření vztlaku a obratné řízení přechodů mezi vertikálním a horizontálním letem jsou zásadními prvky pro provozní účinnost a použití

systémů vertikálního vzletu a přistání bezpilotních letadel v různých oblastech, mimo jiné v oblasti sledování, přepravy nákladu a pátracích a záchranných akcí. [7] [8]

1.2.1 Rotorové mechanismy

Systémy založené na rotorech využívají několika rotorů v tandemu k vytvoření vztlaku, převážně prostřednictvím nasazení sudého počtu protiběžných vrtulí. V kvadrokoptéře, typu bezpilotního letounu, existuje konfigurace se čtyřmi rotory, přičemž dva z rotorů se otáčejí ve směru hodinových ručiček a dva proti směru. Uspořádání rotorů usnadňuje zachování stability a vyrovnávání rotační síly vytvářené každým rotorem. Vzestupná síla se generuje urychlením vzduchu směrem dolů rotory, čímž vzniká tlakový rozdíl mezi množstvím vzduchu umístěného nad a pod rotory, což nakonec vede ke vzniku vzestupné síly. [9]

Konfigurace a konstrukce rotorů hrají klíčovou roli při vytváření vztlaku a celkové výkonnosti systému. Různé faktory, včetně počtů rotorů, množství listů v rámci jednotlivého rotoru, konfigurace tvaru listů a kvality použitých materiálů, mohou ovlivnit účinnost a efektivitu vytvářeného vztlaku. Kromě toho možnost nastavit rychlost jednotlivých rotorů poskytuje přesnou manipulaci se systémem bezpilotního letadla s vertikálním vzletem a přistáním, zahrnující změny výšky, polohy a směru. [9]

1.2.2 Tiltrotorové mechanismy

Technologie spočívá v umožnění změny polohy rotorů vůči danému letovému režimu. Během fáze vzletu a přistání jsou rotory nastavené směrem vzhůru, což umožňuje vzlet svisle jako u helikoptér. Poté, co je dosažena letová výška, jsou rotory přestavěné do horizontální polohy. Toto přestavění vede ke změně letového chování a nyní prostředek oponuje výhodami letadla. [10]

Konkrétním příkladem využití mimo bezpilotní prostředky je Bell Boeing V-22 Osprey (obr. 3). Tento hybridní letoun spojuje vlastnosti technologie vertikálního vzletu a přistání ve vertikální konfiguraci a rychlostí standardního letounu v horizontální konfiguraci letu. Konstrukční vývoj se tak nezaměřuje pouze na samotný vývoj letounu ale zároveň na technologie spojené s rotačním ramenem. Dle letových vlastností letoun vykazuje obratné manévrovací schopnosti v různých režimech letu. [10]

Obrázek 3: Bell Boeing V-22 Osprey [11]

1.2.3 Vektorování tahu

Technogie bezpilotních prostředků se schopností vertikálního vzletu a přistání, je používána s cílem generovat vztlak strategickým usměrňováním síly vytvářené pohonným systémem do určitých vektorů. Tato technika usnadňuje vertikální vzlet a přistání letadla a současně umožňuje pečlivé manévrování za letu v malých rychlostech. Systémy vektorování tahu zahrnují možnost otáčení nebo naklápění motorů nebo trysek, což následně vyvolá změnu směru tahu, která způsobí změnu generovaného vztlaku. Využitím přímého vertikálního vektorování tahu může bezpilotní letoun se schopností vertikálního vzletu a přistání účinně dosáhnout stavu levitace - hovering, zatímco částečné nebo úplné horizontální vektorování tahu usnadňuje dosažení dopředného letu a lepší ovladatelnosti. [7]

Obrázek 4: F-35B Lightning II [12]

S touto technologií se setkáváme i mimo bezpilotní prostředky, a to u bojových stíhacích letounů jako je například F-35B Lightning II (obr. 4). Tento letoun využívá technologií jako je LiftFan nebo pohyblivou trysku, díky které dochází k vertikálnímu vzletu a přistání. Natáčení trysky poskytuje dostatečný vertikální tah potřebný pro stabilizaci a kontrolu výšky.

Kromě natáčení trysky je letoun vybaven malými tryskami v oblasti trupu, které slouží k regulaci klopení. [13]

1.3 Hybridní bezpilotní prostředek

Technologie vertikálního vzletu a přistání umožňuje letadlům a bezpilotním letounům provádět vertikální vzletové a přistávací manévry, čímž zvyšují jejich flexibilitu a schopnost operovat v různorodém terénu. Pro manévrování s vertikálním vzletem a přistáním se používají různé technologie, mezi ně patří systémy založené na rotorových konstrukcích (například vrtulníky a multikoptéry), letadla s křídly a vektorování tahu (jako jsou některé typy stíhaček) a další alternativní řešení. Kromě toho jsou v bezpilotních prostředcích využívány také elektromotory, které umožňují vyšší účinnost a jsou šetrnějším řešením vůči životnímu prostředí. Elektromotory nepotřebují rozsáhlý palivovový systém v křídlech, což vede k rozšíření užitečného zatížení. Současný technologický pokrok usnadnil vývoj různých druhů bezpilotních prostředků pro vertikální vzlet a přistání, příkladem jsou pohonná úhlová křídla, multikoptéry a hybridní letadla spojující výhody konvenčních letadel a vrtulníků. Rozšířené zavádění technologických vymožeností přispělo k tomu, že bezpilotní letouny získaly významné postavení v různých oblastech, mimo jiné v průzkumu, zemědělství a záchranných operacích. [8] [14]

V současné době se stále více výrobců zaměřuje na tvorbu a výrobu multikoptér, které jsou optimalizovány pro různé oblasti použití, zejména pro fotografování a videozáznam, průzkumy, inspekce a monitorování. Tato diskuse se týká standardních bezpilotních prostředků typu X nebo H. Vývoj technologie v této oblasti nabízí značný potenciál pro optimalice trajektorie, kterou se může vývoj ubírat. V současné době se výrobci, kteří stojí v čele výroby kvadrokoptér, aktivně nepodílejí na vývoji hybridních letadel s vertikálním vzletem a přistáním, který spojuje možnosti vertikálního vzletu a přistání vrtulníků s rozšířenou letovou kapacitou letadel s pevnými křídly. Hybridní letouny jsou známé tím, že jsou vhodné zejména pro letecké cesty, které vyžadují cestování na dlouhé vzdálenosti. [14] [15] [16]

Hybridní systém pro vertikální vzlet a přistání představuje kombinaci vlastností letadel s pevnými křídly a multikoptér v bezpilotním prostředku. Tento typ bezpilotního letounu má schopnost provádět vertikální vzletové a přistávací manévry, které připomínají mánevry vrtulníku, a zároveň je schopen během letu plynule přecházet do horizontálního pohybu, podobně jako letadla. Metodika hybridních systémů použitá v tomto kontextu umožňuje vyšší úroveň manévrovatelnosti a letových dovedností při současném zachování schopnosti vertikálního vzletu a přistání, čímž se stává optimálním řešením pro mnohostranné oblasti zahrnující průmysl, zemědělství, dopravu a různé další související sféry. Hybridní bezpilotní prostředky získávají pohonnou sílu z řady technologických prvků konstrukce, inovativních

pohonných jednotech a hybridních systémů, které synergicky spojují více zdrojů energie, aby dosáhly vynikající účinnosti a výkonu. [16] [17]

Přestavitelná konfigurace rotorů

Hybridní letouny s vertilkáním vzletem a přistáním se zásadně liší od tradičních konstrukcí letounů tím, že využívají otočné rotory umístěné na koncích křídel pro stoupavé i klesavé manévry. Při přechodu do horizontálního režimu však hybridní letadlo s vertikálním vzletem a přistáním prochází procesem, při němž se otočné rotory postupně přestaví a nabývají pevné konfigurace, což letadlu poskytuje zvýšenou rychlost. [10] Varianta pozorovaná u hybridních letadel s vertikálním vzletem a přistáním se vyznačuje přítomností rotorů umístěných v konfiguraci s dvojí orientací (obr. 5). Z toho vyplývá schopnost rotace rotorů, jak ve směru, tak proti směru hodinových ručiček. Uvedený atribut umožňuje hybridním letadlům s vertikálním vzletem a přistáním vykazovat zvýšenou obratnost a univerzálnost, pokud jde o jejich směrové změny. [16] [18] [19]

Obrázek 5: Bezpilotní prostředek s přestavitelnou konfigurací rotorů [20]

Pevná konfigurace rotorů

Druhým typem konfigurace jsou takzvaně prostředky s pevnou konstrukcí polohy rotorů (obr. 6), kde je zajištěna snadnější konstrukce. V tomto případě není potřeba řešit mechanizmy pro přestavování rotorů, pouze je omezen výkon vertikálních rotorů. Bezpilotní samokřídla jsou oproti tradičním letadlům schopna operovat s mnohem menšími plochami, a tedy i s nižší hmotností. Dalším benefitem je vysoká rychlost či dolet. Díky své schopnosti vertikálního vzletu a přistání může být nasazen i v oblastech, kde není možné vybudovat letiště. Hybridní bezpilotní prostředek typem samokřídlo s pevnými rotory je konstrukčně složité letadlo, které vyžaduje vysokou úroveň technologie a inženýrských řešení. Vzhledem k vysoké rychlosti a doletu musí být letadlo vybaveno výkonným motorem a dostatečným energetickým zásobníkem. Rotory, které umožňují vertikální vzlet a přistání, musí být schopny generovat dostatečný tah, aby letadlo dokázalo vzlétnout a přistát bez problémů. [16] [21]

Obrázek 6: Bezpilotní prostředek s pevnou konstrukcí rotorů [77]

1.4 Vývojové typy hybridních bezpilotních prostředků

Hybridní bezpilotní prostředky obsahují moderní avioniku, která zajišťuje bezpečný a spolehlivý let. Implementace PID kontrolérů je závislá na konstrukčním typu bezpilotního prostředku. Každý typ přináší výhody a nevýhody, které je potřeba ladit, jak v rámci vývoje konstrukce, tak v rámci následného testování a reálného letu. Z toho vyplývá, že na vývojové typy lze pohlížet z různé perspektivy. V tomto případě je zaměřený pohled z hlediska akčních členů. Akční členy hrají zásadní roli při řízení, včetně jejich schopnosti přecházet mezi režimy letu. [22]

Rozčlenění vývojových typů hybridních bezpilotních prostředků z pohledu akčních členů patří:

- **Bezpilotní prostředek s otočnými rotory (Tiltrotor)** (obr. 7) – hybridní bezpilotní prostředek, u kterého jsou aktuátory, které mají schopnost naklánění rotorů a měnění směru letu. Při vertikální vzletu a přistání jsou rotory nastaveny směrem vzhůru. Poté jsou přestavěny do směru dopředného pohybu. Akční členy zahrnují rotory a také aktuátory změny směru. Vývojové studie samokřídel s otočnými rotory se tak zaměřují na optimalizaci tahu ve vertikálním i horizontálním směru. Kromě optimalizace tahu se optimalizace týká i natočení polohy rotorů. [19] [22]

Obrázek 7: Tiltrotorový bezpilotní prostředek [23]

- **Samokřídlo "ocasní letoun" (Tailsitter)** (obr. 8) – typ hybridního bezpilotního prostředku, který má trup orientován vertikálně. Sada rotorů je v počátku letu směrována vzhůru. Pokračování v horizontálním letu u tailsitteru je pomocí naklápění výškových kormidel na samokřídle. Akční členy zahrnují rotory, které umožňují generovat potřebný tah směrem vzhůru při vzletu a přistání, a aktuátory, které řídí nastavení flaperonů. [22]

Obrázek 8: Novlit 3 tailsitter [24]

- **Standard hybrid VTOL** (obr. 9) – tento typ hybridního bezpilotního prostředku kombinuje vertikální a horizontální rotorové systémy. Během vertikálního směru, tj. přistání a vzlet, využívá sadu rotorů, které jsou směřovány vzhůru. Poté jsou rotory používány pro generování dostatečného vztlaku. Paralelně s rotory pro vertikální pohyb je v provozu také vrtule samokřídla pro horizontální směr letu. V rámci akčních členů se zde využívá sada rotorů ve vertikálním směru, vrtule samokřídla a aktuátory pro ovládání křidélek. Toto řešení vede ke zvýšení manévrovatelnosti, doletu a efektivitě letu. [22]

Obrázek 9: DeltaQuad Pro [25]

1.5 Technologické využití hybridních bezpilotních prostředků v praxi

V současné době je časté používání letounů s vertikálním vzletem a přistáním, které se skládají z různých konfigurací, zejména využití samokřídel a rotorových systémů. Kombinace výhod, které jsou vlastní konvenčním letounům a vrtulníkům, poskytuje zvýšenou schopnost přizpůsobit se různým situacím a zlepšuje výkonnost v různch oblastech. Mezi hybridními letouny lze identifikovat armádní využití, jako je Bell Boeing V-22 Osprey, nasazovaný armádou Spojených států amerických, a F-35B Lightning II. U armádních řešení jsou navrhovány využití pro specifický účel bojových misí. Účely bojových misí sebou nesou i rozdílnou konstrukci letounu včetně řídicích systémů. Z tohoto prostředí lze přenést jednotlivé technologie, které jsou dále využívány v civilním letectví. Výsledkem tohoto propojení využitých technologií v armádě a zapojením autopilota bezpilotních prostředků, jsou získány systémy, které jsou schopny se pohybovat v hustě obydlených částech měst. V městských částech se setkáváme se spoustou nových výzev a aspektů k zajištění bezpečnosti. [10] [13] [14]

Urban Air Mobilty (UAM), který zahrnuje inovativní formy dopravy s využitím bezpilotních prostředků, lehkých letadel či helikoptér pro přepravu osob či zásilek v městských oblastech, se zaměřuje na řešení dopravních problémů v hustě osídlených oblastech, kde stávající dopravní infrastruktura není dostatečná. UAM je vyvíjeno na základě rostoucího zájmu o bezpilotní lety a technologický pokrok v oblasti autonomních letadel. Aktuální cíl UAM spočívá ve vytvoření ekosystému, který bude využívat infrastrukturu obydlených částí v souladu s umožněním bezpečné navigace mezi budovami. Tento aspekt počítá s vývojem kontrolních algoritmů, které se kromě efektivního plánovaní trasy zaměřují i na řízení polohy. [26] [27] Mezi konkrétní plánované příklady použití UAM patří lety na krátké vzdálenosti v městských oblastech nebo mezi městy. Plánovaná realizace těchto letů by měla být zajištěna pomocí hybridních bezpilotních prostředků s vertikální vzletem a přistáním. [27]

Praktická realizace UAM je například koncept hybridní bezpilotní prostředek Zuri 2.0 (obr. 10). Tento bezpilotní letoun s vertikálním vzletem a přistáním vyvíjený českou společností ZURI Aero kombinuje vlastnosti vrtulníku a tradičního letadla. Letoun je navržen pro přepravu osob v městském prostředí. Model 2.0 je vybaven elektromotory, které jsou umístěny na křídlech. Elektromotory zodpovídají za dostatečný tah a správnou funkci rotorů. Mezi akční členy patří rotory, které lze přestavět do jednotlivých letových režimů. Dále zde najdeme akční členy, jako jsou výše zmíněné elektromotory, či právě řídicí systém, který zajišťuje stabilizaci letu, včetně řízení rychlosti a směru. [28]

Obrázek 10: Koncept Zuri 2.0 [29]

Další koncepční model v oblasti UAM je britská Neoptera Aero eOpter (obr. 11). Jedná se o elektrické koncepční řešení pro přepravu osob, které umožňuje vertikální vzlet a přistání. Bezpillotní prostředek typu „tailsitter“ s rotačním trupem je vybaven elektrickými motory, které pohání jednotlivé rotory. Použití elektromotorů zajišťuje nižší hlučnost včetně snížených emisí. Akční členy tohoto konceptu jsou rotory, které generují vztlak a umožňují letadlu vzlet a přistání. Při horizontálním letu jsou rotory využívány jako zdroj pohonu, včetně ovládání směru. Další aktuátory jsou elektromotory a také rotorová ramena, která jsou přestavitelná v různých režimech letu. Výrobce udává i akční členy, které ovládají plochy jako jsou křidélka či směrová kormidla pomocí servomotorů. [30]

Obrázek 11: Neoptera Aero eOpter [31]

Řešení, které není pouze konceptem, ale již se setkáváme s praktickým použitím, se zaměřuje na přepravu zásilek v méně osídlených oblastech. Výrobce ALTI, sídlící v Jihoafrické republice, nabízí řešení hybridních prostředků pro transfer zboží či monitoring. ALTI Transition (obr. 11) kombinuje výhody kvadrokoptér s tradičním letadlem. Model je vybaven elektromotory, které slouží k pohonu rotorů. Kontrukce hybridního prostředku nabízí sadu rotorů směrem vzhůru včetně jedné vrtule, která zajišťuje pohon během horizontalního letu. Sada rotorů, včetně vrtule pro dopředný pohyb jsou akčními členy hybridnního bezpilotního prostředku ALTI Transition. [32]

Obrázek 12: ALTI Transition [33]

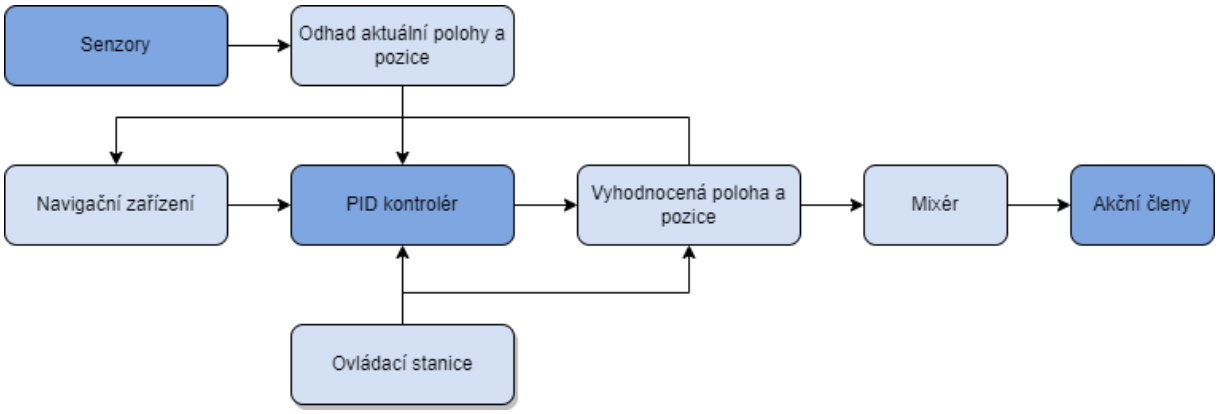
Výše zmíněná řešení jsou ukázkami, že aktuální využití hybridních bezpilotních prostředků je poměrně časté i v oblasti velkých letadel. Koncepční řešení ukazují na fakt, že se s nimi počítá v budoucnosti, a proto je třeba věnovat čas vývoji.

1.6 Kontrolní řízení hybridních bezpilotních prostředků

Řízení hybridních bezpilotních prostředků zahrnuje řízení polohy, výšky a rychlosti v každé fázi letu. Získání letových parametrů je dosaženo integrací různých senzorů, aktuátorů a softwarových algoritmů. Senzorové vybavení zahrnuje akcelerometry, gyroskopy, magnetometry, barometry a prostorovou navigaci. Pomocí těchto zdrojů jsou sbírána měřitelná data, jako je například rychlost, zrychlení, náklony v osách či výška. Pomocí výpočtů jsou získána data o poloze bezpilotního prostředku v prostoru. Data získaná ze senzorů poskytují důležité informace kontroly pohybu. Následné výpočty vedou ke zpracování dat v reálném čase, čímž je umožněno precizní řízení bezpilotního prostředku. Výpočtová řešení jsou klíčová i pro vznikající kolizní situace, kde by měla být včasná reakce. [34]

V tomto kontextu se můžeme bavit o flight stacku (obr. 13), který znázorňuje komunikaci mezi jednotlivými senzory, PID kontrolérem a akčními členy. Jedná se o soubor algoritmů a hardwaru pro řízení a navigaci bezpilotního prostředku. [35]

- **Senzory** – Zařízení, která měří data pro další použití. Příkladem mohou být barometry, gyroskopy či prostý výškoměry.
- **PID kontrolér** – Princip PID kontroléru spočívá v příjmu a zpracování letových informací. Je přijímán meřený stav o poloze spolu s nastavenou hodnotou na vstupu. Výstupní hodnotou je korekce, která spočívá v neustálé úpravě do doby, než dojde ke stabilizaci v nastavené hodnotě.
- **Akční členy** – Funkce akčních členů je vykonávat akční příkazy, které jsou převedené z dat na fyzický úkon z mixéru, jako například “naklonit se dopředu”, zprostředkovává pohyb křidélek.



Obrázek 13: Flight Stack

Aktuátory jsou klíčovým vykonavatelem řídicího signálu zpracovaného systémem řízení letu na jasně hmatatelné fyzické pohyby. U hybridních bezpilotních prostředků jsou součástí akčních členů systémů servomotory pro regulování řídicích ploch, či právě elektromotory, které stojí za ovládáním pohonu rotorů. Kromě toho jsou používány systémy vektorování tahu pro řízení tahu a umožnění plynulého vertikálního vzletu a přistání. [35] [36]

Systémy, které jsou navrženy pro řízení letu, využívají řadu softwarových algoritmů, jejichž cílem je zpracovávat data ze senzorů, optimálně vyhodnocovat a vysílat příkazy aktuátorům. Pro regulaci pohybu a přechodem mezi vertikálním a horizontálním letem se využívá celá řada kontrolních algoritmů. [21] V praxi se tak setkáváme zejména se třemi níže zmíněnými:

- **PID regulátory** – Zajišťují regulaci stabilizace a regulaci polohy.
- **Algoritmy odhadu stavu** – Funkce těchto algoritmů spočívá ve využití dat ze senzorů pro odhad polohy, orientace a rychlosti bezpilotního prostředku.
- **Algoritmy optimální navržení dráhy** – Poskytují určení optimální dráhy za pomoci analytických a systematických strategií k určení nejefektivnější dráhy.

Softwarová řešení jsou navrženy tak, aby zajišťovala optimální výkon složitých systémů, tím že jsou data monitorována v reálném čase a zároveň mohou být data v daný moment nastavována. Pomocí výše zmíněných řešení je minimalizován zásah lidského činitele a snižuje se tak riziko chyb a poruch. Vývoj inovací a zefektivnění softwarových řešení pro řízení vede k usnadnění efektivního provozu u komerčních i průmyslových systémů. [21] [37]

Níže jsou vybrány softwarové prostředky, které jsou vhodné pro řízení systémů hybridních bezpilotních prostředků:

- **PX4** – Open-source platforma autopilota, která lze nasadit na velké množství modelů bezpilotních letadel. Systém PX4 obsahuje značné množství již existujících algoritmů a modulů, které splňuji kritéria na řízení letu, dohad aktuálního stavu a plánování dráhy. [37]
- **ArduPilot** – Open-source platforma, která rozšiřuje možnost podpory o hybridní bezpilotní letouny. [38]
- **Robot Operating System (ROS)** – Open-source systém, který funguje jako middleware, je určený pro vývoj robotických zařízení. Zprostředkovává zajištění flexibility a přizpůsobivosti, což umožňuje nasazení pro vývoj vlastních řídicích systémů. Dále ROS nabízí velké množství knihoven a nástrojů, které usnadňují integraci vlastních algoritmů. [38]

Následná validace bývá proveditelná skrz řadu nástrojů, které poskytují výpočetní sady, často se tak setkáváme s prostředím Matlab. Aplikace určené pro hybridní systémy by měly splňovat řadu předpokladů, které zajišťují dostatečnou úroveň zabezpečení.

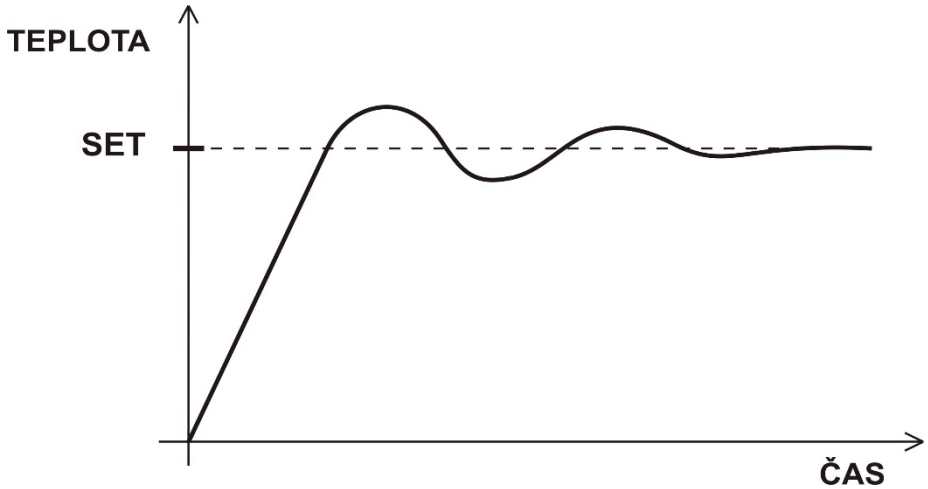
Dostatečná úroveň zabezpečení je zkoumána z více pohledů. Každý pohled vnímá chování systému jiným způsobem, nelze tedy říct, že při splnění pouze některých z kritérií, je systém zcela optimální. [42] Mezi jednotlivé pohledy patří:

- Přesnost
- Bezpečnost
- Ovladatelnost
- Spolehlivost
- Flexibilita
- Kompatibilita

Z celkového hlediska lze říci, že systém řízení hybridních bezpilotních prostředků musí být velmi sofistikovaný, aby mohl řídit složitý typ letadla, včetně ovládání a zpracování dat všech senzorů. Rychlejší odezva systému vede k okamžitému zpracování dat a následné korektuře, což vede k optimální výkonnosti PID kontroléru.

1.6.1 PID kontrolér

PID kontrolér (Proportional, Integral, Derivative) je ovladač, který je používaný v širokém zastoupení zpětnovazebných řídicích algoritmů. Jeho úlohou je v systému regulovat požadovaný výstup systému prostřednictvím minimalizování rozdílu mezi požadovanou hodnotou a skutečným výstupem. Integrace PID kontroléru je rozšířena v celé řadě mechanizmů, do nichž jsou zahrnuty mechatronické systémy, automatizace a dohledová řízení. Postupná úprava rozdílu ve výpočtu zahrnuje vyšší účinnost a odolnost v řídicích



Obrázek 14: Chování PID kontroléru v rámci tepelné realizace [45]

systémech. V mechanických systémech je PID kontrolér používán k udržení polohových, rychlostních či tepelných parametrů (obr. 14), přičemž je zajištěn kontinuální výkon. [43] [44] Kontrolér je sestaven ze třech základních modulů, které jsou prezentovány níže:

- **Proporcionální složka (P)** - Zastává klíčovou roli při generaci výsledné hodnoty, která je v přímé úměře k převládajícímu rozporu. Úkolem proporciální složky je udržet úměrný vztah mezi velikostí chyby a jejím výsledným výstupem. Funkce usnadňuje účinnou reakci systému na případné odchylky zjištěné v rámci chyby a snaží se urychlit celý proces. Citlivost kontroléru je definována proporcionálním přírůstkem K_p. Čím vyšší je hodnota K_p, tím je odezva kontrolního systému rychlejší. Velké zvýšení hodnoty vede k následnému překmitu, oproti tomu nižší hodnota může zpomalit korekci.
- **Integrační složka (I)** – Tato složka se soustřeďuje na akumulaci chyb za určité časové období, čímž vytváří odezvu, která je přímo úměrná k nakumulované chybě. Přítomnost integrační složky pomáhá eliminovat nepřesnost či odchylku v ustáleném stavu, kterou nelze zmírnit proporcionální složkou. Rychlost s jakou integrační složka reaguje na chybu je řízena přírůstkem K_i. Při použití vysoké hodnoty K_i je vyvoláváno oscilační chování nebo dochází k přestřelení hodnoty. Zvolená nízká doba pak značně prodlužuje dobu potřebnou k odstranění chyby v ustáleném stavu.
- **Derivační složka (D)** – Zahrnuje časový gradient chyby a poskytuje zpětnou vazbu, která je úměrná rychlosti změny. Zavedení tlumícího přírůstku je poskytování korekce, díky které dojde k eliminaci překmitu a oscilační pohybu. Koeficient K_d řídí vliv derivační složky na výstupu. Zvýšená stabilita je závislá na zvýšené hodnotě K_d, zatímco snížená hodnota nemusí zajistit dostatečné tlumení a může vést k oscilacím.

PID kontrolér zpracovává pararelně tyto tři složky a moduluje výstup regulace výpočtem součtu jednotlivých členů. Dosažení požadované odezvy systému, která může zahrnovat snížení překmitů oscilací nebo doby ustálení, je podmíněno kalibrací jednotlivých přírůstků (K_p, K_i, K_d). [44]

1.6.2 Ladění kontrolního algoritmu

Ladění kontrolního algoritmu zahrnuje úpravu jednotlivých zisků PID kontroléru. Úprava proporcionálního (P), integračního (I), derivačního (D) zisku vede k dosažení požadovanému chování a optimálnímu výkonu řídicího algoritmu. Optimálního výkonu bývá dosaženo díky stabilitě a minimalizování překmitů, u nesprávně naladěného PID kontroléru je možné získat výstup, který povede k nestabilitě a chaotickému chování, a vede k ohrožení bezpilotního prostředku. K zajištění stability vedou různé metody ladění kontroléru,

zejména zavedení technik, které systém dostanou do ideálního chování. V publikaci „PID Controllers: Theory, Design and Tuning“ od švédských vědců, zabývajících se teorií řízení, se ukazuje, že existuje více technik, které lze využít v rámci ladění. [44] [46]

Rozdělení jednotlivých technik dle výše zmíněné publikace [46]:

- Analytické techniky
- Empirické techniky
- Optimalizační techniky

Rozdíly mezi jednotlivými využitelnými technikami spočívají v přístupu k nalezení optimálních hodnot zisků. Výběr použitelné techniky závisí na konkrétní úloze systému, včetně dostupných informací o systému a požadavcích výkonu bezpilotního prostředku.

1.7 Softwarové nástroje pro vytváření a validaci kontrolního algoritmu

Použití vhodných nástrojů pro ladění a následnou verifikaci hraje klíčovou roli v procesu vývoje autonomních systémů. Tyto nástroje umožňují výzkumným institucím nasimulovat potřebné podmínky, které vedou k ověření a k případnému upravení algoritmů a celé řídicí strategie, bez nákladných a časově náročných experimentů s reálnými bezpilotními prostředky. Dále jsou do simulace implementovány různé scénáře a podmínky, které je možné testovat a poté z toho vyvodit východiska ke zvýšení robustnosti a spolehlivosti systémů. [47]

Softwarové nástroje, které jsou zaměřeny na návrh a validaci kontrolního algoritmu patří:

- Gazebo
- MATLAB - Simulink
- Webots
- V - REP
- Blender

Gazebo

Open-source 3D simulátor umožňující testování robotických systémů v realistické virtuálním prostředí, poskytuje fyzikální engine pro modelování kinematiky a dynamiky. V rámci vývoje jsou v Gazebo simulátoru prováděny testy a ladění řídicích systémů v bezpečném bez rizika poškození skutečného hardwaru. Simulátor využívá řadu fyzikálních enginů, výchozím enginem je ODE, dále pak poskytuje podporu enginům Bullet, DART a Simbody. Každý engine poskytuje různé možnosti pro simulaci fyzikálních vlastností. [47] [55]

MATLAB - Simulink

MATLAB je výpočetní prostředí s vysoce komplexním programovacím jazykem vytvořeným firmou MathWorks, zatímco Simulink je blokově orientované prostředí pro modelování, simulaci a analýzu dynamických systémů. MATLAB – Simulink jsou požívány pro vývoj, testování a analýzu řídicích algoritmů. Kombinace těchto programů poskytuje snadnou integraci s hardwarovými zařízeními a rozsáhlou knihovnu nástrojů a funkcí [59] [60]

Webots

Open-source simulátor poskytující prostředí pro modelování, programování a následnou simulaci, se zaměřuje na výuku a výzkum, kde umožňuje ověřování algoritmů. Webots používá primárně fyzikálního enginu ODE pro simulaci fyzikálních vlastností a poskytuje podporu pro řadu programovacích jazyků jako je C, C++, Python nebo MATLAB. [48]

V-REP

Open-source platforma pro simulaci robotických systémů, která nabízí rozšiřitelné prostředí pro modelování, programování a simulaci robotů. Simulátor nabízí snadnou integraci s podporovanými knihovnami a využívá fyzikálních enginů jako je ODE, Bullet, DART a Vortex. [43]

Blender

Open-source 3D program nabízející možnosti modelování, animací a renderu, je používán k vizualizaci a tvorbě 3D modelů pro robotiku, avšak není primárně určen k simulaci robotických systémů, spíše je určen k vývoji počítačových her. Simulátor využívá fyzikální enginu Bullet, bez obsahu speciálních nástrojů pro robotiku. [78]

1.8 Validace výkonnosti algoritmu řízení

Validace výkonnosti je závěrečným krokem v procesu návrhu a implementace algoritmu řízení. Díky úspěšné validaci je zajištěno, že navržený systém splňuje podmínky stability, rychlosti odezvy a přesnosti. Bez důkladného prověření mohou být systémy náchylné k míře chybovosti a nežádoucím účinkům řídicího systému. Nevyzpytatelné chování může vést k selhání celého systému, v případě bezpilotních prostředků se dostáváme až na hranici incidentu. [61]

Existuje celá řada technik pro validaci. Záleží na konkrétním použití systému. Při validaci výkonnosti algoritmu řízení je třeba se zaměřit na klíčové parametry, jako jsou překmity ve výstupu signálu, doba náběhu, doba ustálení a doba dosažena k maximálnímu vrcholu simulované funkce. Tyto metriky umožňují hodnotit, jak systém rychle a přesně dokáže reagovat na změnu vstupního signálu, včetně adaptace na poruchové

situace. [61] Ke správné interpretaci jednotlivých výkonnostních ukazatelů, je níže krátký popis jednotlivých ukazatelů.

- **Překmit (Overshoot)** – Hodnota překmitu je maximální míra, jakou výstup systému překročí jeho konečnou ustálenou hodnotu. Čím vyšší je míra překmitu, tím vyšší jsou oscilace a nestabilita systému.
- **Doba náběhu (Rise time)** – Hodnota času, která trvá systému, než jeho výstup stoupne z určitého nízkého procenta (většinou 10%) na určité vyšší procento (většinou 90%) konečné ustálené hodnoty. Čím je doba náběhu kratší, tím je rychlejší odezva systému.
- **Doba ustálení (Settling time)** – Čas, který systému trvá, než se jeho výstup ustálí a zůstane uvnitř určitého procentuálního rozmezí (udáváno 2% nebo 5%). Ukazatel představuje, jak dlouho trvá systému dosáhnout stabilního výstupu po přidání vstupního skoku.
- **Doba, k dosažení vrcholu (Peak time)** – Čas, který systému trvá dosáhnout svého maximálního vrcholového výstupu. Tento ukazatel ukazuje představu o tom, jak rychle je dosažena maximální hodnota výstupu. [61]

Kromě výkonnostních ukazatelů lze využívat i kombinaci dalších technik a tím docílit podrobnější validace. Kombinací technik lze vysvětlit, jak moc je systém stabilní a jak dokáže reagovat na změnu. Mimo výkonnostní ukazatele, které jsou výše zmíněné, je možné analyzovat kritické body letu, jako je například spodní kritická hranice výšky. [61]

1.9 Shrnutí teoretické části práce

Technologické zpracování směřuje k populárnímu řešení využívání bezpilotních prostředků s vertikálním vzletem a přistáním. Hybridní bezpilotní prostředky nabízí řešení, které využívá výhod tradičních letadel a zároveň kombinuje výhody vrtulníků. Díky těmto výhodám, se zdá být hybridní prostředek jako možné řešení pro budoucí velmi sofistikovanou infrastrukturu v hustě obydlených oblastech. Vývoj hybridních bezpilotních prostředků však sebou nese výzvy v oblasti použitého hardwaru, ale zejména v oblasti použitého softwaru. [7]

Konstrukce hybridních bezpilotních prostředků je založena na implementaci akční členů řízení. Existují různé konstrukční přístupy, které determinují specifické uspořádání aktivních členů. Za současných podmínek se v praxi setkáváme s pevnou konstrukcí polohy rotorů, ale i s otočným ramenem, na kterém je upevněn rotor. Každé řešení sebou nese řadu výhod a nevýhod. [7] [9] [10] Každý akční člen, který je součástí hybridního bezpilotního prostředku, rozšiřuje komplikovanost konstrukce. Z pohledu spolehlivosti je třeba se zaměřit na konstrukční řešení, které není příliš komplikované. [16] [17]

Hlavním faktorem, o které se opírá zpracování diplomové práce, je vývoj a implementace softwarového řídicího algoritmu. Primární zaměření se věnuje návrhu kontrolního algoritmu hybridního bezpilotního prostředku, včetně validace. Implementace softwarového řešení se liší dle konstrukční složitosti hybridního bezpilotního prostředku. Zpracování zahrnuje řadu proměnných včetně počtu a typu akčních členů, které musí být zohledněny v řídicím algoritmu. Návrh kontrolního algoritmu je složitý proces vyžadující použití sofistikovaných algoritmů a technologií, kde klíčovou součástí je PID kontrolér, který zajišťuje stabilní let, přesné ovládaní a plynulý přechod mezi jednotlivými režimy letu. [43] [44] [46]

Využitím nástrojů pro simulaci, které využívají fyzikálních enginů, je možné navrhnout řídicí kontrolní algoritmus, u kterého proběhne validace ve virtuálním prostředí, avšak chování kontrolního algoritmu bude odpovídat podobnému chování v reálném světě. Během návrhu bude možné využít i technologií, které modifikují jednotlivé složky PID kontroléru tak, že bude zajištěn optimální výkon. K zajištění optimalizace výkonu řídicího algoritmu, je zapotřebí, aby systém splňoval kritéria ukazatelů výkonnosti. Validace tvoří závěrečný krok v procesu návrhu a implementace, při níž je zjišťováno splnění podmínek stability, rychlosti odevzdy a přesnosti. Důkladná validace minimalizuje nežádoucí účinky chování řídicího systému.

2 Metodologie

Zaměření diplomové práce se zaobírá návrhem kontrolního algoritmu, konkrétně proporcionálně-integrační-derivačního (PID) kontroléru, který je aplikovatelný pro řízení bezpilotních prostředků. Adaptabilita a robustnost patří mezi základní prvky, které jsou klíčové pro efektivní využití užitečného výkonu a širokou aplikaci v oblasti autonomních prostředků. PID kontrolér umožňuje ladění jednotlivých parametrů vedoucí k optimalizaci výkonu a splnění konkrétních letových operací. Tato práce je založena na ověřených metodách a algoritmech používaných v oboru autonomního řízení. Díky sofistikovanému řídícímu algoritmu, jako je právě PID kontrolér, je možné dosáhnout stabilních a velmi přesných definovaných letových hodnot. [48]

Při stanovení řídicích cílů je třeba určit strukturu PID kontroléru, která by měla být závislá na cílech a zároveň na dynamice hybridního bezpilotního prostředku. Pro různé úrovně řízení jsou zahrnuty vnořené smyčky. Vnější smyčky, které stojí za kontrolou polohy či vnitřní smyčky, které jsou zaměřeny na řízení klonění. Po stanovení struktury, je nutné zodpovědně vybrat a naladit zisky – Gains, jednotlivých složek. Tyto zisky jsou dále definovány jako K_p, K_i, K_d. Zisky určují časové zpoždění a následnou reakci na chybu mezi požadavkem a skutečným stavem. Ladění parametrů je nezbytná část pro udržení požadovaných řídicích cílů. [48] [64]

Před zavedením PID kontroléru do reálného hardwaru bezpilotního prostředku je zcela žádoucí otestovat jeho výkon v simulačním prostředí. Tento krok poskytuje otestování návrhu PID kontroléru a identifikuje případné problémy, které by se mohly promítnout do reálného provozu. Opětovná simulace vede k získání větší důvěry účinnosti PID kontroléru. Po ověření prostřednictvím simulace zbývá čas na export do reálného modelu, kde je algoritmus podroben experimentálním testům. I během tohoto testování může být PID kontrolér vystaven jemnému doladění jednotlivých parametrů PID. [64]

2.1 Gazebo simulátor

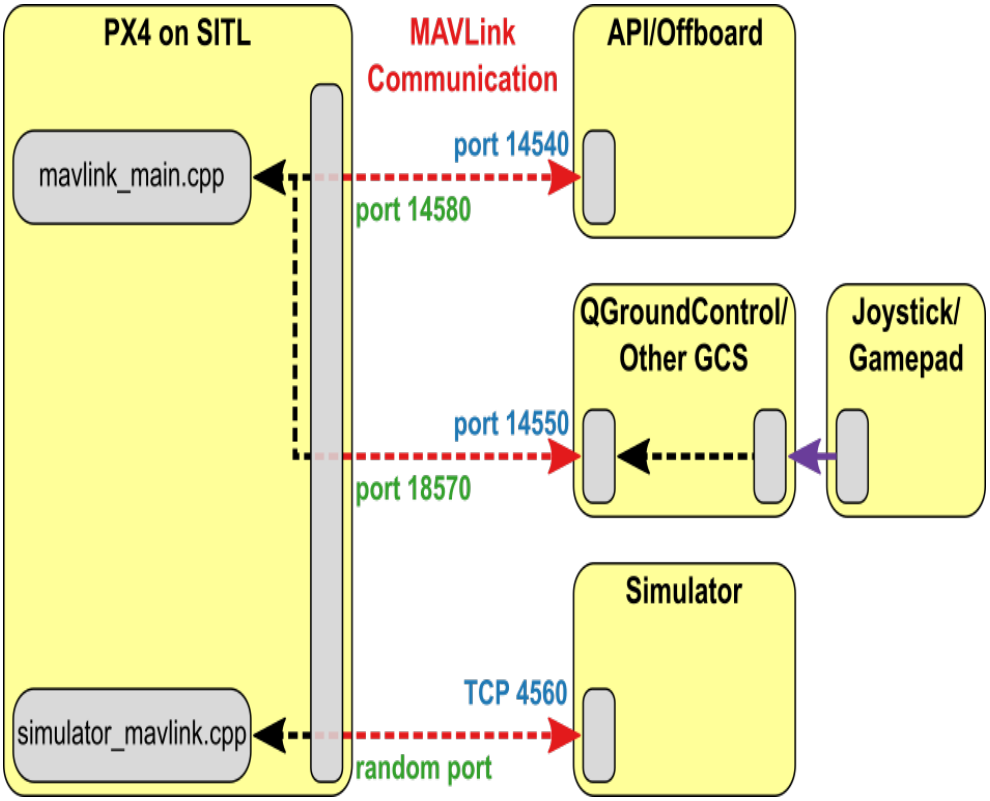
Pro účely diplomové práce byl vybrán simulátor Gazebo. Primárním důvodem výběru simulátoru Gazebo je podpora více fyzikálních enginů, čímž je umožněna přesná simulace fyzikálního světa včetně interakce s jednotlivými objekty. Díky tomu je poskytováno chování, které věrně odpovídá chování systému v reálném světě. Na základě podpory fyzikálních enginů je simulátor schopen pracovat s 3D modely obsahující ovladatelné akční členy, čímž je zajištěna lepší simulace letu. [55] [56]

Podstata sekundárního důvodu tkví, že Gazebo simulátor obsahuje celou řadu 3D modelů, které jsou zároveň součástí PX4 autopilota. Gazebo simulátor nabízí podporované modely, které jsou plně přizpůsobitelné a kompatibilní, lze je upravit tak, aby splňovaly funkcionality

v rámci projektu. Dále jsou tyto modely zároveň kompatibilní s ROS a QGroundControl prostředím, což usnadňuje zprostředkování komunikace mezi simulátorem a řídicím algoritmem, který je nezbytný pro efektivní vývoj a testování bezpilotních prostředků. [48] [55]

Výběr simulátoru Gazebo je podpořen řadou studií, které v rámci svých výzkumů využívají enginu ODE, v rámci Gazebo simulátoru. [48]

Simulační prostředí (obr. 15) tvoří podstatu celého testování a ladění algoritmů řídicích systémů před jejich nasazením do reálného bezpilotního prostředku. Simulační schéma poskytuje přehled o tom, jak je simulační prostředí integrováno v rámci autopilota PX4, ROS a dalších softwarových komponentů. Schéma zobrazuje hlavní části simulačního prostředí a prezentuje jak mezi sebou jednotlivé části komunikují prostřednictvím různých protokolů. [39]



Obrázek 15: Simulační schéma komunikace [40]

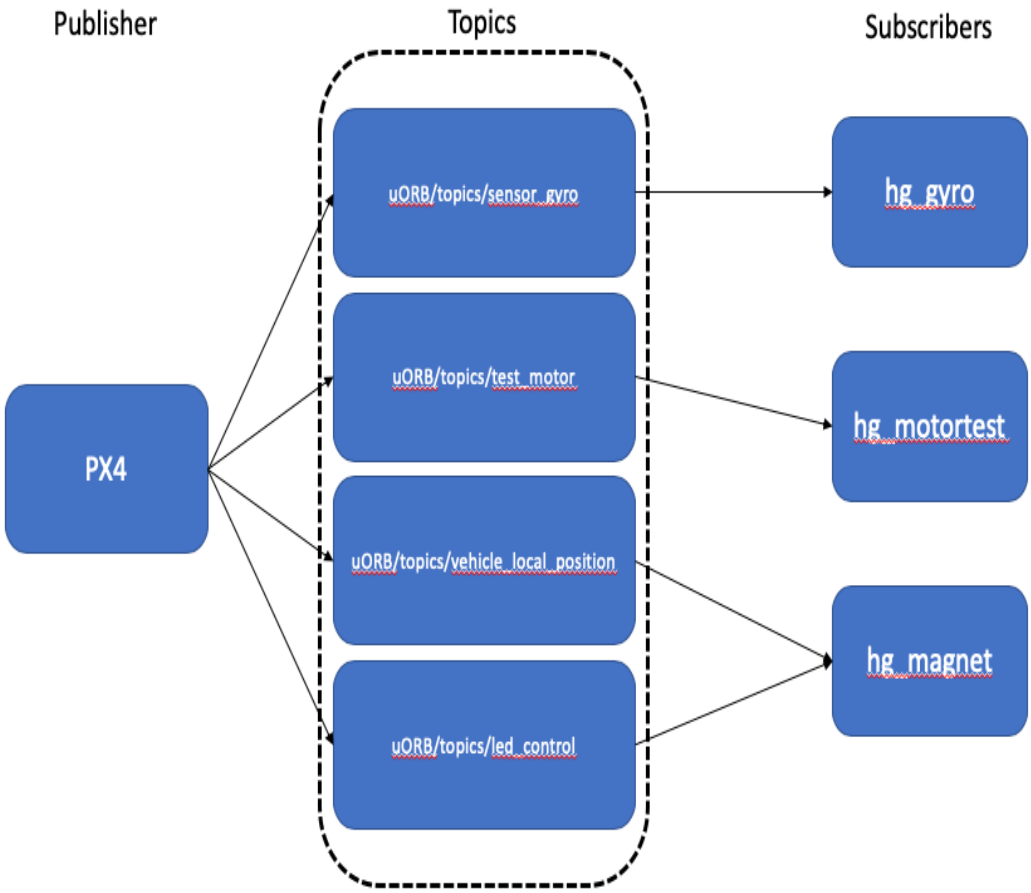
2.1.1 ROS

Open-source framework, který je zaměřen na vývoj robotických aplikací, poskytuje řadu nástrojů a knihoven, které usnadňují tvorbu komplexních robotických systémů napříč různými platformami a hardwarovými konfiguracemi. Díky modulárnímu přístupu a podpoře vývojářů nabízí ROS velké množství projektů pro plánování pohybu, řízení či další projekty spojené s robotikou. [38] [48] [56]

Integrace robotického systému do simulátoru Gazebo poskytuje efektivní kombinaci pro vývoj, testování a ladění řídicích algoritmů bezpilotních prostředků. Díky kombinaci ROS a Gazebo dochází k jednoduché integraci různých senzorů, aktuátorů a algoritmů, což usnadňuje vývoj a testování nových funkcí. Spojení mezi Gazebo a ROS usnadňuje komunikaci mezi jednotlivými komponenty systému, což vede k účinnému využití vývoje a zároveň je umožněna rychlá integrace jednotlivých částí systému. Výsledkem je vytvoření efektivních a flexibilních nástrojů vývoje, testování a ladění autonomních systémů. [47] [56]

MAVROS: ROS rozhraní pro MAVLink

Pro účinnou komunikaci mezi bezpilotním prostředkem a systémem řízení či mezi různými částmi bezpilotního prostředku, je nezbytné mít řádně nastudovaná pravidla o tom, jak funguje sběr a přenos dat. V tomto ohledu hraje klíčovou roli MAVLink zajišťující sdílení informací mezi jednotlivými částmi systému. [49] MAVLink zprávy jsou v rámci celého systému zprostředkovány prostřednictvím uORB (uObject Request Broker), který zajišťuje jednoduchý přenos zpráv mezi jednotlivými komponentami. PX4 autopilot publikuje velké množství *topics*. Na obrázku níže (obr. 16) je prezentován princip publish – subscribe systému,



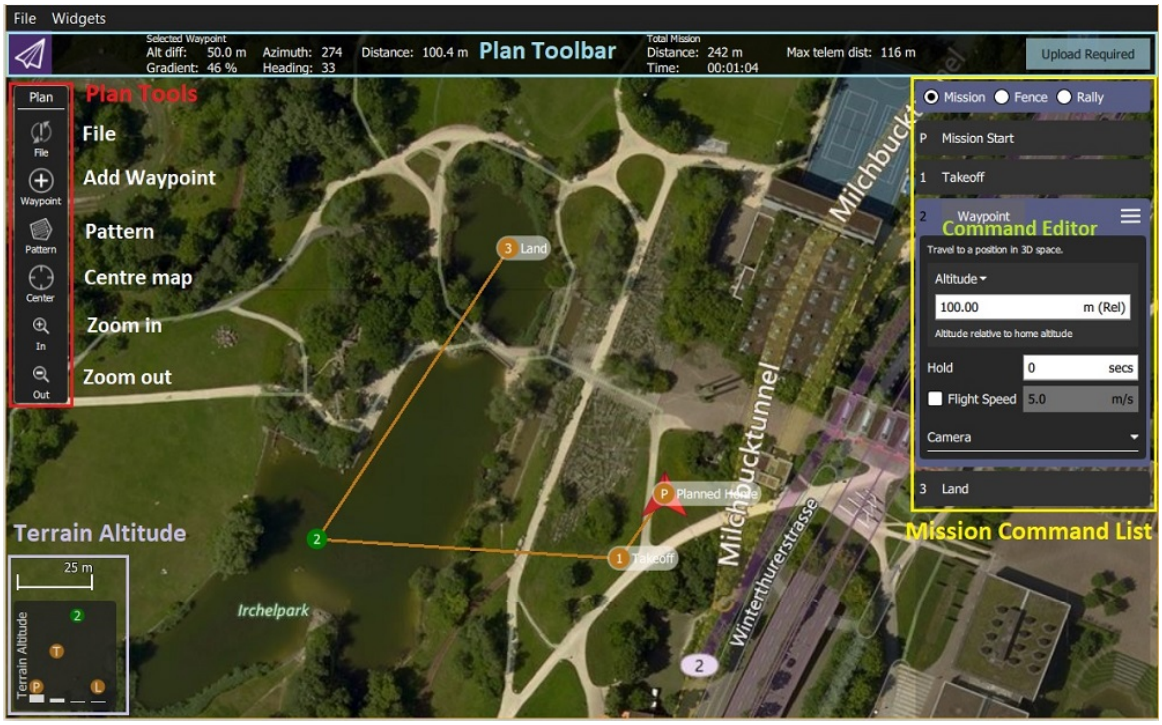
Obrázek 16: Systém publish – subscribe, využívající konkrétní topics [52]

jak PX4 autopilot publikuje mnoho různých témat. Tyto témata obsahují proměnné, které mohou být odesílány od publisherů k subscriberům. Subcriber nemusí nepřetržitě volat funkci pro získávání dat. V celém prostředí autopilota PX4 jsou propojené veškeré publikační a odběratelské moduly pomocí zpráv uORB. [49] [50] [51]

MAVROS, rozhraní ROSu pro komunikační protokol, umožňuje integraci MAVLink komunikace do ROS prostředí, čímž je usnadněn vývoj a ladění řídicích algoritmů. Využití MAVROS společně s Gazebo prostředím poskytuje vývojářům kompletní zajištění při vytváření, testování a ladění autonomních systémů bezpilotních prostředků. Pomocí této kombinace je poskytnuta snadná implementace a optimalizace řídicích algoritmů, zpracování senzorických dat a komunikace mezi jednotlivými komponenty. [50] [53]

2.1.2 QGroundControl

QGroundControl je open-source řídicí stanice, která poskytuje uživatelům možnost konfigurovat a kalibrovat softwarové řešení bezpilotního prostředku. Jedná se o grafické prostředí, které poskytuje plánování letové mise, včetně sledování průběhu letu v reálném čase, a analýzu letového záznamu. Ve vědeckých pracích se setkáváme s aplikací QGroundControl zejména v prostředí optimalizace letových tras či plánování pohybu. Rozšíření aplikace QGroundControl vede k funkcím automatizace plánování letových misí (obr. 17), kde se objevují i definované letové mise typu „vzlet – posun – přistání“. [47]



Obrázek 17: Plánování letové mise v prostředí QGroundControl [57]

Součástí QGroundControl jsou také nástroje pro PID kontrolér, které se netýkají pouze kontroly výšky, ale také klopení, klonění či natočení. Veškeré hodnoty jsou převáděny v reálném čase rovnou do grafu, kde je simulován signál v dané fázi. Hodnoty pak lze převzít a využít například jako výchozí hodnoty pro vytváření vlastní PID kontroléru. [47] [58]

2.1.3 Ověření konzistence enginu

Ověření konzistence letu bylo provedeno na pěti různých referenčních letech, které byly spuštěny v různých časech prostřednictvím aplikace QGroundControl a za dodržení stejných podmínek trasy, včetně požadované výšky v horizontálním letu. Jednotlivé lety zahrnovaly scénář - vzlet do 10 metrů, poté posun po ose x do 50 metrů, po docílení tohoto bodu následovalo přistání. Na základě těchto letů byly provedeny následující analýzy:

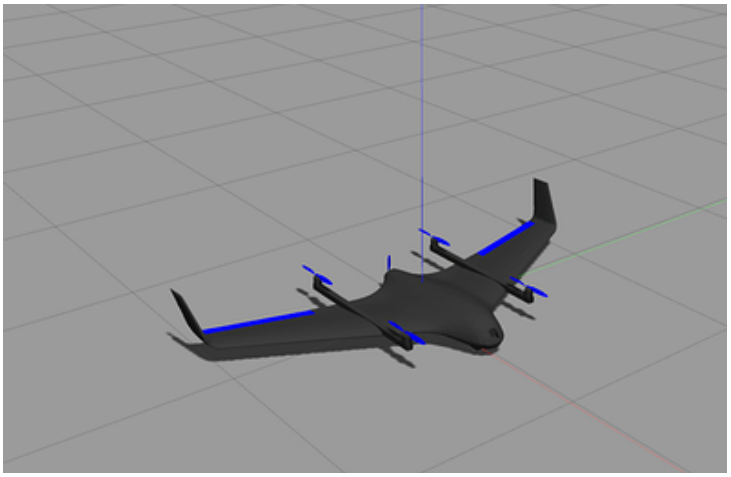
- Okometrická analýza trendu
- Analýza jednotlivých výšek amplitudy

V rámci okometrické analýzy byly sledovány jednotlivé tvary křivky, zda jsou zde nějaké anomálie nebo zda se průběh jednotlivých letů nemění. Ve druhé analýze byly nejprve získany maximální výšky amplitudy jednotlivých signálů, poté tyto hodnoty byly seřazeny od nejvyšší po nejnižší, kde nejvyšší znamenala 100%. Procentuální rozdíly byly porovnány mezi sebou. V případě, že maximální rozdíl byl pod jedním procentem, byl vnímán jako zanedbatelný pro celý výzkum, v tu chvíli byla zamítnuta podmínka pro zprostředkování více letů. Vývoj a implementace byla o to zjednodušena, čímž celý výzkum byl zaměřen na tvorbu návrhu kontrolního algortimu. [56]

2.1.4 Výběr konkrétního 3D modelu

Diplomová práce sebou nese provedení experimentálních letů se specifickým výběrem bezpilotního prostředku. Tento výběr se opírá o využití výhod hybridního bezpilotního prostředku. Zároveň během výběru byl brán potaz na pevnou konstrukci s fixovanou polohou rotorů pro vertikální i horizontální směr letu. Kromě fixovaných rotorů - akčních členů je zde minimalizován počet na další akční členy, pouze na pár křidélek. Z toho plyne, že zvolený 3D model zajišťuje kontrolu pohybu a stability pouze pomocí jednotlivých rotorů. Mimo konstrukční parametry byly dále zohledněny faktory, jako je velikost či dostupnost v případě realizace ve skutečném světě. [48]

V rámci této diplomové práce byl zvolen 3D model HIL Standard VTOL QuadPlane (obr. 18), který je jedním z přednastavených modelů. Model kombinuje výhody samokřídla s možností vertikální vzletu a přistání. Vhodnost tohoto modelu pro diplomovou práci tkví zejména v tom, že je zde možnost testování a optimalizace návrhu kontrolního algoritmu při změně vertikálního na horizontální let. [64]



Obrázek 18: 3D model HIL Standard VTOL [65]:

Jedná se o model se čtyřmi rotory, které umožňují vertikální vzlet, následně tyto rotory slouží jako podpora při generování vztlaku při dopředném pohybu. Dopředný pohyb je tvořen zejména vrtulí v zádní části modelu. Umístění vrtule v zadní části poskytuje vyšší rychlost vůči standardním kvadrokoptérám. Postavení rotorů v horizontálním i vertikálním směru umožňuje stabilnější přechod mezi změnou letu z vertikálního na horizontální a naopak. Stabilizace letu je zajištěna systémem autopilota PX4, který poskytuje autonomní let, stabilizaci či navigaci, což napomáhá k testování kontrolních algoritmů. Dále je tento model rozšířitelný, takže díky znalosti kořenových složek, lze upravit dle konkrétních potřeb. V celkovém ohledu 3D model tvoří dokonalý základ pro tuto diplomovou práci. [64]

2.2 Implementace kontrolního algoritmu

Zaměření této kapitoly klade důraz na klíčové aspekty spojené s vývojem, implementací a ladění řídicího algoritmu pro modifikovaný let hybridního bezpilotního prostředku. Kontrolní algoritmus je základním prvkem, díky kterému je dosahováno požadovaného chování a stabilizace letu. V rámci této části je definována hlubší znalost pro úspěšné navržení a implementaci optimálního kontrolního algoritmu.

2.2.1 PID kontrolér

Je jeden z nejpoužívanější způsobů řízení polohy v praxi, díky své snadné implementaci a jednoduchosti. PID kontrolér se používá k regulaci dynamických systémů, včetně bezpilotních prostředků, kde zajišťuje stabilitu letu a zároveň směřuje k požadované hodnotě.

Zde je podrobnější rozbor, jak každá složka ovlivňuje chování systému v čase a také jak se s úpravou parametrů mění signál PID kontroléru. [48] [61] [62]

- **Proporcionální složka (P)** - Složka je závislá na aktuální chybě *e(t)* mezi požadovanou hodnotou a skutečnou hodnotou měřenou systémem. Poskytnutá korekce je přímo úměrná velikosti chyby. Z toho plyne, že čím větší je chyba, tím větší bude i korekce.

$$\boldsymbol{P = K_p * e(t)} \tag{1}$$

Zvýšením hodnoty proporcionální $\boldsymbol{K_p}$ je docíleno toho, že PID kontrolér bude rychleji reagovat na chybu, avšak to vede k oscilacím v systému. Při snížení hodnoty $\boldsymbol{K_p}$ bude reakční doba větší a systém tak bude způsobovat zpoždění.

- **Integrační složka (I)** – Složka funguje jako akumulátor předchozích chyby a poskytuje korekci přímo úměrnou k celkovému součtu chyb.

$$\boldsymbol{I = K_i * \int \big(e(t)\big)\, dt} \tag{2}$$

Zvýšení integrační hodnoty $\boldsymbol{K_i}$ dochází k eliminaci trvalých chyb, což může vést k překmitávání a tím tak zpomalit systém, který se vrací do požadované hodnoty. Snížení hodnoty $\boldsymbol{K_i}$ vede ke snížení citlivosti na trvalé chyby, zároveň je zlepšena stabilita systému. Odstraňování trvalé chyby může být značně pomalejší.

- **Derivační složka (D)** – Závisí na rychlosti změny chyby a dodává do systému korekci, která je přímo úměrná k rychlosti změny chyby. Derivační složka predikuje budoucí chybu a poskytuje korekci, než se tato chyba dostane na tuto hodnotu.

$$\boldsymbol{D = K_d * \frac{de(t)}{dt}} \tag{3}$$

Zvýšená hodnota derivační složky $\boldsymbol{K_d}$ se lépe adaptuje na předvídanou budoucí chybu a rychleji reaguje na rychlé změny. Tato konfigurace vede k potlačení oscilací a zlepšení stability. Avšak pokud je hodnota příliš vysoká dochází ke zvýšení citlivosti na šum a mohou být vyvolány oscilace. Snížení hodnoty $\boldsymbol{K_d}$ vede k menší citlivosti systému, který bude hůře reagovat na rychlé změny. Což znamená že reakce budou pomalejší a zavedení té hodnoty může vést k nedostatečnému potlačení oscilací. [46]

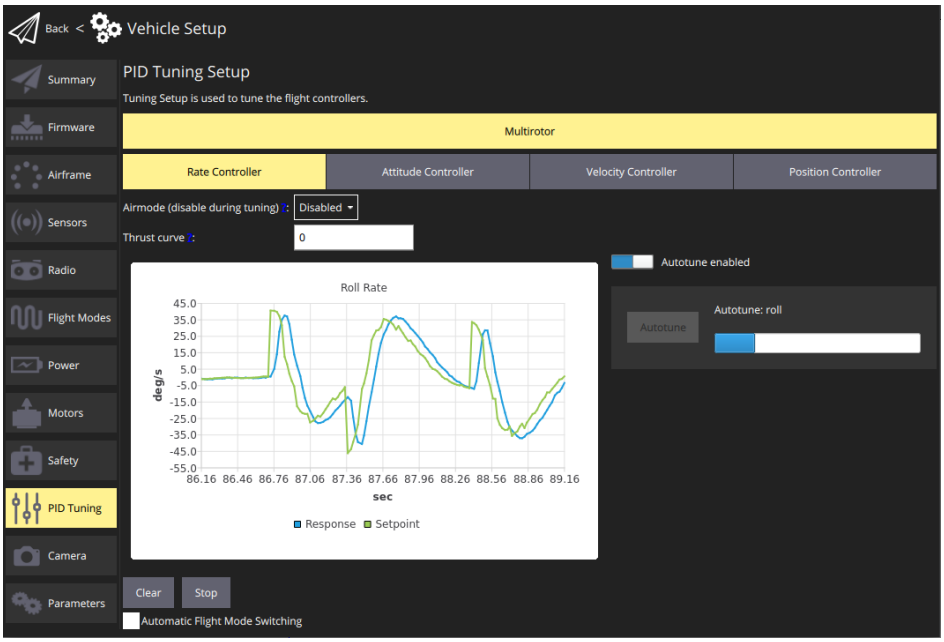
Změna jednotlivých parametrů směřuje k různému chování systému v čase. V manuálním ladění se provádí pouze velmi jemné úpravy pro zajištění optimálního chování bezpilotního prostředku. [62]

2.2.2 Metody ladění kontrolního algoritmu

Použití vhodných nástrojů pro ladění a následnou verifikaci hraje klíčovou roli v procesu vývoje autonomních systémů. Tyto nástroje umožňují výzkumným institucím nasimulovat potřebné podmínky, které vedou k ověření a případně upravení algoritmů a celé řídicí strategie, bez nákladných a časově náročných experimentů se reálnými bezpilotními prostředky. Dále jsou do simulace implementovány různé scénáře a podmínky, které je možné testovat a následně z toho vyvodit východiska ke zvýšení robustnosti a spolehlivosti systémů. [61] [63] [66]

Využití AUTOTUNE pro kalibraci referenčního letu

Během výzkumu je využit dašlí aspekt QGroundControl a tím je automatické ladění PID kontroléru (obr. 19). Nástroj AUTOTUNE seřídí jednotlivé složky PID kontroléru tak, aby byla zajištěna dostatečná optimalizace. Tato funkce umožní velmi rychlé naladění parametrů řídicího systému, což zajistí stabilitu a efektivní výkonnost hybridního bezpilotního prostředku. [58] [63]



Obrázek 19: Automatické ladění v prostředí QGroundControl [67]

Ziegler – Nichols metoda

Mezi manuálními metodami, které jsou implementovány v rámci vývoje bezpilotních prostředků, je Ziegler – Nicholsova metoda. Tato metoda je založena na empirických technikách, kdy jsou složky PID kontroléru přepočteny pomocí empirických vzorců. [63] [66]

Metoda spočívá v určení kritického zisku (gainu) a následně i kritické periody systému. Hledání těchto hodnot začíná tak, že integrační a derivační konstanty jsou nastaveny na nulu.

Poté je zvyšován proporcionální zisk, do fáze, kdy začne systém oscilovat s konstantní amplitudou. V tomto bodě se nachází kritický zisk a kritická perioda. [61] [69]

Metoda ladění využívá získaných hodnot (K_u - kritický zisk, T_u – kritická perioda), které jsou dále upraveny pomocí empirických vzorců.

Empirické vzorce jsou definovány takto:

$$Propocionální\ zisk\ (K_p) = 0{,}6 * K_u \tag{4}$$

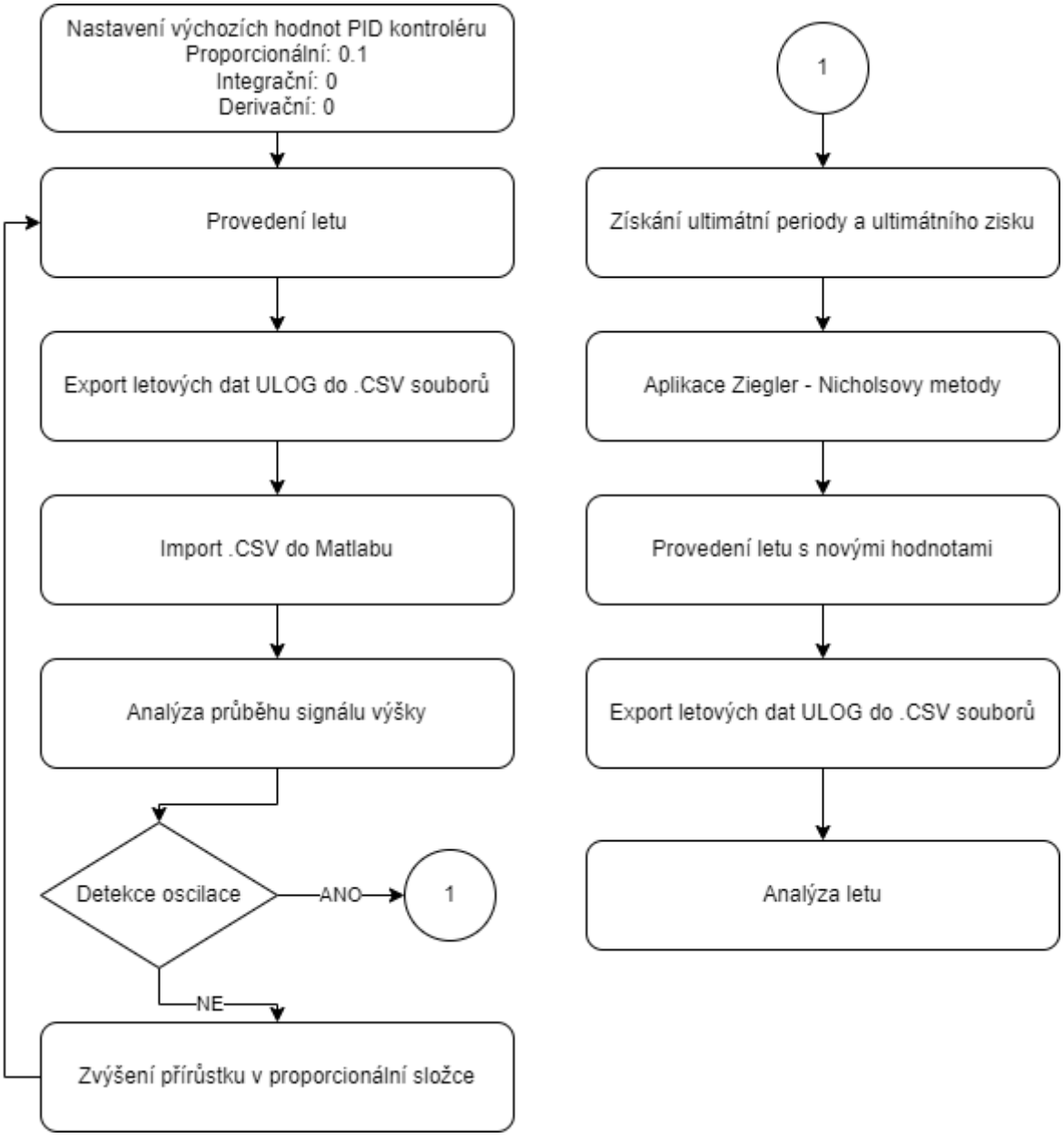
$$Integrační\ zisk\ (K_i) = 2 * \frac{K_p}{T_u} \tag{5}$$

$$Derivační\ zisk\ (K_d) = \frac{K_p * T_u}{8} \tag{6}$$

Použitím těchto vzorců v PID kontroléru je minimalizována chyba řízení. Dochází ke zlepšení stability systému, sníží se doba překmitu a je docílen nižší čas pro dosáhnutí požadované hodnoty. Výsledný signál zahrnuje menší počet oscilací a bude rychleji reagovat k nastavené hodnotě. Kromě optimalizace může docházet k nadměrnému překmitu v případě, že hodnoty nejsou naladěny správně. [61] [69]

Manuální ladění PID kontroléru, které zahrnuje implementaci Ziegler-Nicholsovy metody, je proces, který získává jednotlivé hodnoty PID kontroléru samotným letem, poté následuje offline analýza dat a poté znovu. V této ladící smyčce zustáváme do doby, než jsme spokojeni s chováním bezpilotního prostředku. Pro každý PID kontrolér (altitude, pitch a roll), jsou získávány hodnoty separátně, nejprve jsou provedeny lety pro altitude PID kontrolér, poté pro pitch PID a nakonec pro roll PID kontrolér.

Celý proces začíná nastavením proporcionálního zisku na nízkou hodnotu, která je postupně zvyšována. V ideálním případě začínáme od 0.1 a postupně zvyšujeme do doby než si začneme všímat oscilace signálu. Při detekci oscilace následuje změření ultimátní periody T_u a následně i vyvození ultimátního zisku K_u. Pro přiblížení postupu získávání jednotlivých hodnot je v diagramu (obr. 20) prezentován proces, při kterém jsou získávany hodnoty pro altitude PID kontrolér. [69] [70]



Obrázek 20: Postup manuálního ladění při získávání hodnot ZN

V bodě, kde jsou získány konkrétní hodnoty ultimátní periody a ultimátního zisku, byla použita funkce v MATLABu. V této funkci jsou na počátku definovány vstupní parametry jako je čas a data. Dále se vstupní data převádí na sloupcové vektory, které následně jsou využity při výpočtu první diference (rozdíl mezi sousedními hodnotami), poté jsou nalazeny lokální minima a maxima pomocí funkce *findpeaks(),* která je definována MATLABem. Později je nutné vypočítat amplitudu oscilací mezi jednotlivými extrémy, díky které je spočítána i průměrná amplituda. Následuje výpočet, který vypočítá rozdíl mezi indexy extrémů a také je spočítáná průměrná perioda oscilací. [69] [70]

Získání hodnoty ultimátní periody je udáváno dle Ziegler-Nicholse metody jako průměrná perioda:

$$T_u = průměrná\ perioda \tag{7}$$

Hodnotu ultimátního zisku definuje Ziegler-Nicholsova metoda, jako hodnotu zesílení, při kterém se systém druhého řádu chová jako oscilační zařízení s nekonečně dlouhou dobou trvání kmitů, včetně konstatní amplitudy. V takovém případě je hodnota průměrná amplituda oscilací shodná s maximální amplitudou. Samotný výpočet je tedy podíl průměrné amplitudy a průměrné periody, kde výpočet je doplněn o konstantu $^4/_\pi$, která vychází z aproximace pro systémy druhého řádu. [69] [70]

Výpočet K_u je udán následujícím vztahem:

$$K_u = \frac{4 * průměrná\ amplituda}{\pi * průměrná\ perioda} \tag{8}$$

Získání hodnot pro PID kontroléry klopení a klonění zastává stejného postupu, s rozdílem, že při získávání dat klonění a klopení, je třeba vycházet, že exportovaná data jsou tvořena kvaterniony, které musí být dále přepočítány pro získání hodnot klonění a klopení.

$$Pitch\ (\theta) = asin(2 * (q_0 * q_2 - q_3 * q_1)) \tag{9}$$

$$Roll(\phi) = atan2\big(2 * (q_0 * q_2 - q_3 * q_1)\big), 1 - 2 * ((q_1^2 + q_2^2) \tag{10}$$

Kde matematické funkce *asin(x)* a *atan2(y, x)* slouží pro výpočet trigonometrických operací. Funkce *asin(x)*, je inverzní sinusová funkce, která přijímá argument a vypočítává tak úhel v radiánech. Funkce *atan2(y, x)*, je varianta inverzní tangensové funkce, která přijímá dva argumenty, na místo jednoho. Vypočítává úhel v radiánech mezi kladnou x-osou a souřadnicovým bodem (x,y). [70] [71]

Implementace Ziegler-Nichols metody

Využití této Ziegler-Nicholsovy metody, lze aplikovat více přístupy. Tím prvním přístupem je aplikace Ziegler-Nicholsovy na celý let, tím pádem hledat hodnoty K_u a T_u, které budou po celý let konstantní. V praxi to znamená, že cyklus manuálního ladění bude proveden pouze jednou. V druhém případě je získání hodnot K_u a T_u pro každý interval separátně, z čehož plyne, že se provede manuální ladění nejprve pro vzletovou část, poté pro horizontální let a nakonec pro interval přistání. [69]

Během zpracování diplomové práce bylo zvoleno první řešení, tedy výsledné hodnoty jednotlivých PID kontrolérů využít v rámci celého letu a simulovat standardní let, během kterého byla kalibrace provedena pouze na začátku a poté, je bezpilotní prostředek ovládán bez dalších korektur. Po implementaci jednotlivých softwarových služeb je spuštěn modifikovaný let. [62]

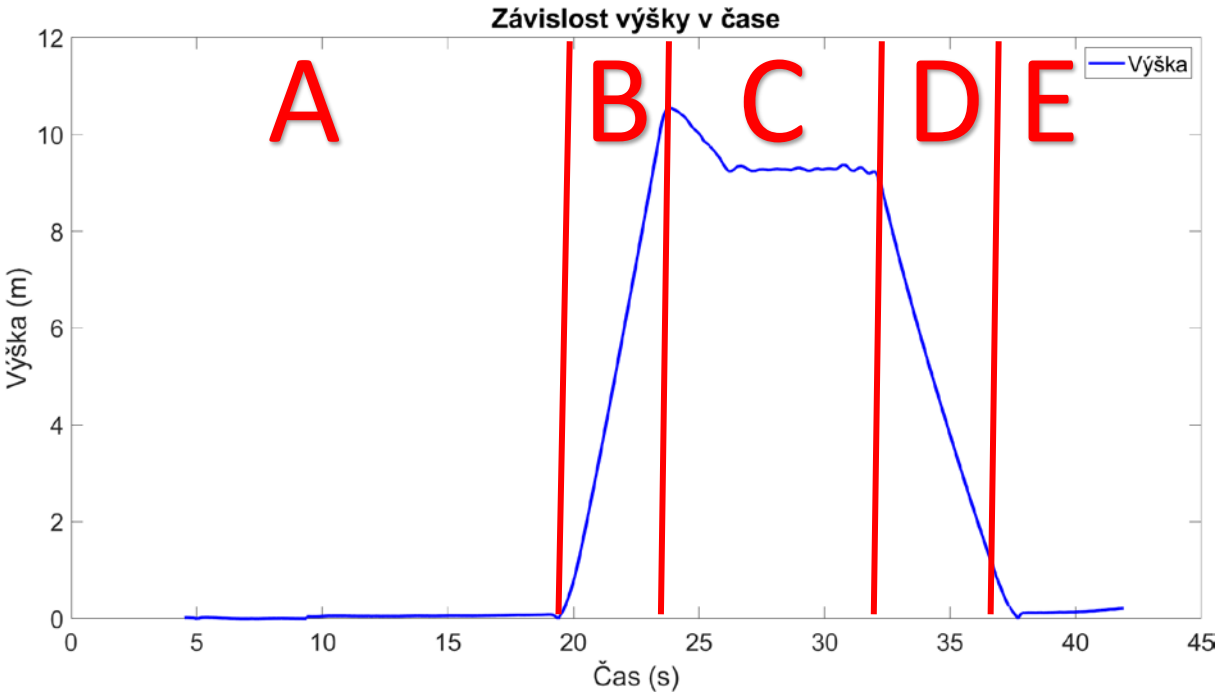
2.3 Metodologické přístupy k vytváření modifikovaného řídicího algoritmu

Na implementaci a vytváření skriptu je pohlíženo z více směrů. Prvním způsobem, jak lze ke skriptu přistupovat, je rozdělení do užitečných částí, díky kterým jsou lépe definovány jednotlivé očekávané letové úkony, například rozdělení na části, jako je vzlet, horizontální let a následné přistání. Při definici rozdělení je nutné vědět, co od tohoto skriptu očekáváme. V případě řešení problémů je možné vzít pouze určitou část skriptu a ten pak optimalizovat. [73]

Druhým způsobem je vytváření samotného skriptu v prostředí Linux. V tomto případě je použito importování podpůrných souborů a složek z webové platformy GITHUB. Souhrn jednotlivých souborů zahrnuje spustitelnou šablonu psanou v kódu C++. Tato šablona obsahuje základní skript, který je dále modifikován. Součástí hlavičkové struktury souboru jsou zahrnuté veškeré knihovny a metody potřebné ke sběru a publikování dat. Dále je zde definována sekvenční logika programu main(). Každá editace znamená novou kompilaci pro spustitelný soubor, který je dále prezentován jako ROS uzel. [72] [73]

2.3.1 Kvalifikace letových intervalů

Následující schéma (obr. 21) ukazuje vertikální pozici bezpilotního prostředku v čase. Detailním popisem lze rozdělit let do několika intervalů – A, B, C, D a E. Zaměření diplomové práce se vztahuje pouze na intervaly B a D. První zkoumaný interval zahrnuje vzlet bezpilotního prostředku až do dosažení požadované výšky, během tohoto intervalu dochází ke změně pohybu z vertikálního na horizontální pohyb. Druhý interval se zaměřuje na klesání bezpilotního prostředku do výšky jednoho metru. Důvodem proč je vydefinován jeden metr a nikoliv nulová výška je skutečnost, že v softwarovém prostředí není země jasně definována. Při přímém poklesu do naprosté nuly, by systém nevyhodnotil tento stav jako přistání. Proto bezpilotní prostředek nejprve sníží výšku na jeden metr a poté je provedena systémová operace dronu „přistávání“.



Obrázek 21: Rozčlenění letu v intervalech

2.3.2 Návrh modifikovaného řídicího algoritmu

Návrh řídicího algoritmu je definován pomocí skriptu pro kontrolní algoritmus bezpilotního prostředku. Algoritmus tvoří zásadní aplikaci pro řízení letu. Vytvoření logicky na sebe navazujících celků má přímý dopad na výkonnost a spolehlivost vytvořeného skriptu. Struktura algoritmu je tvořena sekvenčně, kde v případě splnění podmínek následuje další část. Jednotlivé části skriptu jsou definovány tak, aby splňovaly funkcionalitu a probíhaly v logickém pořadí. Samotný skript tak pro lepší orientaci zastupuje stejné rozdělení, jako tomu bylo u rozdělení letu. Tvorba skriptu začíná již od základní šablony až po finální verzi, která byla použita v experimentu. Finální verze skriptu je v Příloze 1. [74] [75]

Mezi programovací jazyky kontrolních algoritmů patří C++ a Python. Tvorba skriptu kontrolního algoritmu je řešena prostřednictvím programovacího jazyku C++, který je hojně využíván během vývoje robotických řešení, včetně hybridních bezpilotních prostředků. Kromě volby programovacích jazyků, je třeba dbát i následnou validaci systému pro řízení. [41]

Základní šablona, která byla použita, je součástí již existující repozitáře Github *UCM-M143/MavRos-takeoff-n-land*. Šablona poskytuje kompilovaný základ a funkcionalitu, tedy není potřeba vytvářet balíček ROS. Základní struktura se zabývá o řízení vzletu a přistání u bezpilotních prostředků. Řada knihoven a dalších podpůrných souborů zde chybí, proto zvolený skript dokonale vyhovuje zadání i zejména ve výzkumu funkcionality jednotlivých řádků. Tento princip napomohl k pochopení, jak skript pracuje a jak by se dal upravit,

aby splňoval požadavky pro zadání diplomové práce. Šablona přinesla urychlení celého procesu vývoje skriptu a výzkum se tak mohl zaměřit na specifickou adaptaci. [72] [74] [75]

Vysvětlení vlastního ROS skriptu, je kvalifikováno v několika bodech, které pomáhájí lépe pochopit, jak celý skript funguje:

- Zahrnutí potřebných knihoven, definice konstant a globálních proměnných
- Vytvoření třídy PID Controller – implementace PID řízení
- Callback funkce
- Hlavní program – main
    - Inicializace
    - Vytvoření publikovatelů a odběratelů
    - Čekání na připojení k bezpilotnímu prostředku - **A**
    - Aktivace bezpilotního prostředku - **A**
    - Vzlet - **B**
    - Pohyb k bodu - **C**
    - Snížení výšky do jednoho metru – **D**
    - Přistání – **E**
- Ukončení programu

Mimo funkční části skriptu, je skript tvořen i komunikací s uživatelem, kde například jsou vypisovány informace o aktuální výšce či vzdálenosti k bodu. Tyto informace jsou důležité z hlediska sledování výkonu a případné úpravy, tak aby byl zajištěn, co nejlepší výkon. V následující části jsou prezentovány zásadní části skriptu, celý skript je pak součástí přílohy (příloha 1).

Zahrnutí potřebných knihoven, definice konstant a globálních proměnných

Zajištění funkcionality závisí na zahrnutí správných knihoven hned v hlavičkové části, kde jsou dále definovány i jednotlivé kontanty. Mezi stěžejní část patří knihovna *ros/ros.h*, která tvoří základní knihovnu pro práci s ROS, zajišťuje komunikaci mezi jednotlivými uzly a zprostředkovává tok informací mezi odběrateli a publikovateli. [73]

Dále knihovny, které zajišťují systémově funkcionalitu bezpilotního prostředku, jako je například aktivace dronu, změna letového režimu či zprávy obsahující informace o aktuálním stavu bezpilotního prostředku. Poté jsou zde knihovny, které poskytují informace o rychlosti, pozici a rozsahu čidel. Jelikož, žádná z knihoven nedefinuje, jak přesně se je definován klopení bezpilotního prostředku, musí být zahrnuty také knihovny, které obsahují

matematické operace či operace pro práci s kvaterniony, které představují objekty v 3D rotaci. [70] [71]

Kromě zahrnutých knihoven, jsou v hlavičkové části skriptu definovány konstanty a globální proměnné pro řízení bezpilotního prostředku. U výchozí konfigurace před spuštěním samotného skriptu jsou nastaveny zisky nastavené pouze v proporcionálních ziscích výšky, klonění a klopení. Dále pak ještě nastavena hodnota maximálních integračních chyb na 100, díky tomuto nastavení je zamezena akumulace příliš velké integrační chyby.

Vytvoření třídy PID Controller – implementace PID řízení

V této části skriptu je implementována třída *PIDController*, která zajišťuje řízení klonění, klopení a zejména řízení výšky. Třída obsahuje metody a proměnné, díky kterým je možné dále ve skriptu používat pouze odkaz na tuto třídu. Součástí každého úkolu této třídy, jsou definovány jednotlivé zisky včetně maximální integrační chyby. Proces regulace využívá vzorců pro každou složku:

$$\text{Proporcionální složka: } P = K_p * e(t)$$

$$\text{Integrační složka: } I = K_i * \int e(t)dt$$

$$\text{Derivační složka: } D = K_d * \frac{de(t)}{dt}$$

Kde *e(t)* je rozdíl mezi požadovanou hodnotou a měřenou hodnotou. Celkový výstup PID kontroléru nám udává součet jednotlivých složek:

$$\text{Výstup: } Output = P + I + D$$

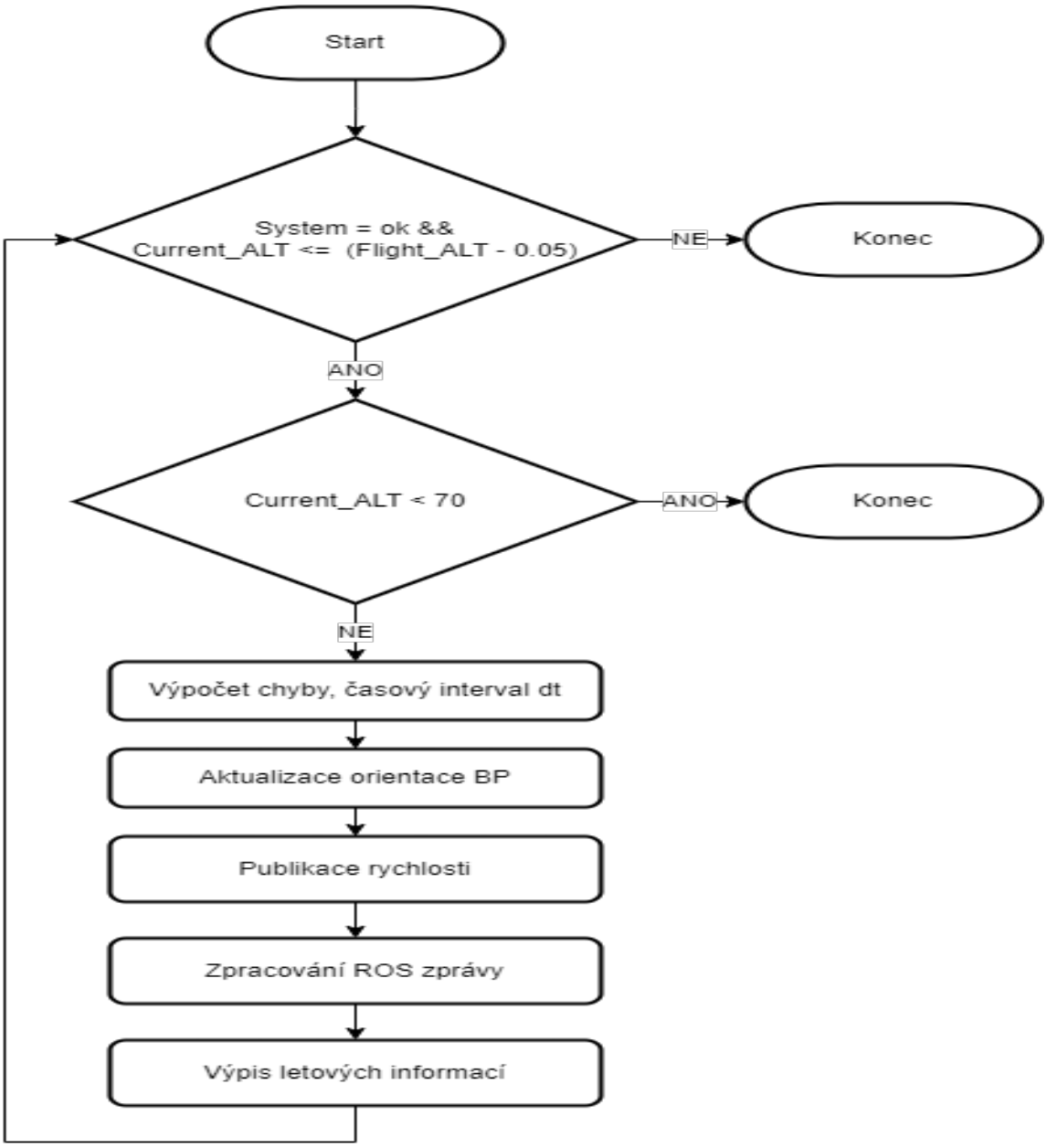
```
PIDController(double Kp, double Ki, double Kd, double max_integral_error)
    : Kp(Kp), Ki(Ki), Kd(Kd), max_integral_error(max_integral_error), prev_error(0.0), prev_error_derivative(0.0) {}

        double update(double error, double dt) {
                error_sum += error * dt;
    error_sum = std::min(std::max(error_sum, -max_integral_error), max_integral_error);
    double error_derivative = (error - prev_error) / dt;


    double alpha = 0.1;
    double error_derivative_filtered = (1.0 - alpha) * prev_error_derivative + alpha * error_derivative;

    double output = Kp * error + Ki * error_sum + Kd * error_derivative_filtered;

    prev_error = error;
    prev_error_derivative = error_derivative_filtered;
    return output;
}
```

Obrázek 22: Implementace třídy PIDController

V případě naší implementace (obr. 22) je výstup PID kontroléru součet jednotlivých složek kde *error* představuje aktuálních chybu. Hodnota *error_sum* je kumulativní součet chyb s omezením na maximální integrační chybu a error_derivative_filtered je chyba derivační složky chyby s alfa koeficientem pro dolní propustný filtr. Tento filtr snižuje vliv šumu a náhodných změn v derivační složce. Koeficient alfa určuje, jak rychle se filtr adaptuje na nové hodnoty chyby. Tato hodnota lze nastavovat mezi 0 a 1 s tím, že čím nižší tato hodnota je, tím je filtr pomalejší a signál je tak více vyhlazen.

Vzlet - B

Část skriptu (Příloha 1), která se zabývá vzletem lze rozdělit na několik částí (obr. 23). V první části, je definována smyčka typu *while*, která se provádí do doby než bezpilotní prostředek nedosáhne letové výšky (10 metrů) nebo v případě špatně nastaveného PID kontroléru do výšky 70 metrů, kde je ukončen program. Uvnitř smyčky pak probíhá výpočet chyb a času, kde je chyba vypočítávána jako rozdíl mezi aktuální výškou a požadovanou výškou, stejně jako časový interval (*dt*) mezi aktuálním a předchozím časem. Aktualizace PID kontroléru se řídí třídou *PIDController*, která je popsána výše. Poté je aktualizována orientace na základě vypočítaných hodnot klonění a klopení a je publikována nová rychlost.



Obrázek 23: Vývojový diagram pro Vzlet - B

V rámci této aktualizace, se aktualizuje i časová chyba. Veškeré informace jsou později vypsány tak, aby bylo možné během letu sledovat letové informace o nastavených hodnotách, chybách a výstupů pro PID kontrolér. V závěru vidíme funkce *ros::spinOnce()* a *rate.sleep()*, které zpracují předchozí zprávy a udrží stabilní frekvenci smyčky. Po dosáhnutí výšky se smyčka ukončí a do konzole je vypsán text „Takeoff done“. Tím je ukončena fáze vzletu. V kódu lze taky využít také variantu s pozicí, která je aktuálně zakomentována.

Snížení výšky do jednoho metru - D

V této fázi bezpilotní prostředek je řízen do výšky 1 metr nad povrchem. Před samotnou smyčkou je nastavení požadovaných hodnot, u výšky je to právě 1 metr a u klonění a klopení jsou hodnoty nastaveny na 0. Dále následuje smyčka, která se provádí dokud nedosáhne bezpilotní prostředek požadované výšky nebo v případě špatně nastaveného PID kontroléru, je zde nastavena kontrolní výška 70 metrů, kdy po dosažení této hodnoty se program ukončí. Výpočet chyby je spočítán jako rozdíl mezi aktuální výškou a požadovanou výškou, stejně jako je to u časvoého intervalu (*dt*) mezi aktuálním časem a předchozím časem. Poté dochází k aktualizaci PID regulátorů. Následuje nastavení aktualizované orientace bezpilotního prostředku pomocí vypočítaných hodnot klonění a klopení. Aktualizují se předchozí hodnoty chyby a času pro výšku. Následně proběhne pomocí funkcí *ros::spinOnce()* a *rate.sleep()* zpracování předchozích zpráv a udržení stabilní frekvence smyčky. V závěru je vypsán do konzole text “Reached 1 meter” a tím je smyčka uzavřena. Vývojový diagram je založen na stejném principu jako je tomu u vzletu.

2.4 Postup pro spuštění validace a ladění řídicích systémů

Jelikož se aplikace pohybuje v prostředí linux, je většina procesů vykonávána pomocí příkazů v terminálu. Než je proveden samotný let, je třeba spustit server Roscore, díky kterému je zprovozněna komunikace mezi jednotlivými uzly v ROS prostředí. Server ROS je spuštěn pomocí následujícího příkazu. [75] [76]

$$roscore$$

Po spuštění hlavního serveru ROS pro koordinaci dat mezi jednotlivými uzly, je spuštěna simulace s autopilotem PX4, prostřednictvím zpráv MAVROS. Komunikace mezi mezi MAVROS a autopilotem PX4 zprostředkovává ovládání bezpilotního prostředku, dále je možné pomocí uzlů číst data ze senzorů a provádět letové úkony. [74] [76]

$$roslaunch\ mavros\ px4.launch\ fcu\_url := "udp://:14550@127.0.0.1:14557"$$

Zprostředkované ovládání však není grafické a nelze simulovat vnější vlivy, které působí na bezpilotní prostředek. Proto je následně spuštěn v dalším okně terminálu příkaz, který otevírá simulátor Gazebo s vybraným modelem a je zahájena SITL simulace PX4. Během simulace lze sledovat chování bezpilotního prostředku číst data ze senzorů a ladit řídicí systém. [76]

$$make\ px4\_sitl\ gazebo - classic - standard - vtol$$

A v této chvíli následují dva možné postupy, které jsou závislé na tom, který typ letu je validován. V případě aplikace referenčního letu je spuštěna aplikace QGroundControl z pracovní plochy. [74] [76]

V druhém případě místo spuštění aplikace QGroundControl je vyžadován ještě jeden příkaz.

$$rosrun \text{ "název složky" "}\textit{název balíčku}\text{"}$$

Poté je proveden let. Po provedení letu, je uložen soubor s koncovkou .ULOG v kořenové složce PX4 autopilotu. V této složce jsou veškeré provedené lety. Pro validaci je potřeba data ze souborů .ULOG konvertovat do :CSV souborů pomocí příkazu.

$$ulog2csv = „\textit{název letu}\text{"}$$

Tento příkaz vytvoří ve stejné složce velké množství souborů, které jsou zpracovány v matematicko – analytickém prostředí MATLAB. Zpracovaná data, se týkají hodnot výšky v čase, vertikální rychlosti v čase, klopení nebo klonění.

2.5 Experimentální postup pro testování a analýzu řídicího systému

Úlohou vyhodnocování experimentální části je určení výkonnostních ukazatelů řídicího systému PID kontroléru. K získání výkonnostních ukazatelů byl proveden referenční a modifikovaný let. Mezi zkoumané výkonnostní ukazatele z hlediska odezvy systému patří:

- Doba náběhu [s]
- Překmit [%]
- Doba ustálení [s]
- Doba, do dosažení maximálního vrcholu [s]

Jednotlivé ukazatele pro každý interval jsou mezi sebou porovnány a vyhodnoceny, který z nich obsahuje menší hodnotu. Kromě výkonnostních ukazatelů odezvy systému jsou analyzovány i kritické body výšky. Kritické body výšky zahrnují zejména spodní body průběhu letu, které představují bezpečnostní riziko bezpilotního prostředku. [76]

Předpokladem experimentálního procesu je určení výkonnosti, že získané výsledky z modifikovaného algoritmu povedou k lepšímu stavu stability a odezvy systému. Měření proběhlo ve vybraných interval B a D. Komparace mezi jednotlivými provedeními proběhla v rámci referenčního i modifikovaného letu. Letecká data z jednotlivých letů jsou zpracovány a analyzovány pomocí softwaru MATLAB. [76]

MATLAB

Využití nástroje MATLAB poskytuje celou škálu podporovaných knihoven, včetně analytických nástrojů pro zpracování telemetrických dat. Po provedení jsou letu jsou letecká data uloženy do formátu ULOG, z tohoto formátu jsou exportovány do .CSV souborů. V rámci .CSV souborů je dále pracováno v prostředí MATLAB, které poskytuje podporu při práci s daty, ale také jsou pomocí tohoto nástroje vytvářeny vizualizace a veškeré matematické operace, které jsou zásadní pro vývoj řídicích systémů.

2.5.1 Porovnání účinnosti pomocí parametrů mezi jednotlivými lety

Dosahování požadované výšky bylo analyzováno pomocí jednotlivých parametrů, které následně byly mezi sebou porovnány. Optimální řešení systému definuje kratší čas vykonání letového úkonu či rychlejší odezvy systému na daný úkon. V našem případě jsou parametry porovnávány tak, že například pro dosažení výšky na intervalu B, jsou zprácovány jednotlivé získané hodnoty a následně porovnány, u kterého provedení letu bylo dosaženo nižší časové hodnoty. Kromě nižší časové hodnoty, je u vyhodnocování řešen překmit, kde je výzkum zaměřen na nižší hodnotu překmitu, ta ukazuje, že v dalším průběhu funkce nemusí být eliminována velmi vysoká hodnota funkce. Lze tedy říct, že čím je hodnota překmitu nižší, nikoliv záporná, tak provedení letu dosahuje pozitivnějších výsledků. V případě záporné hodnoty překmitu dochází k podkmitu. Efektivní provedení a rychlost dosažení letového úkonu jsou základními vlastnostmi, které nám ukazují, které provedení letu je pro nás otpimálním řešením.

V praktickém vyhodnocení řešení byla definována tato podmínka:

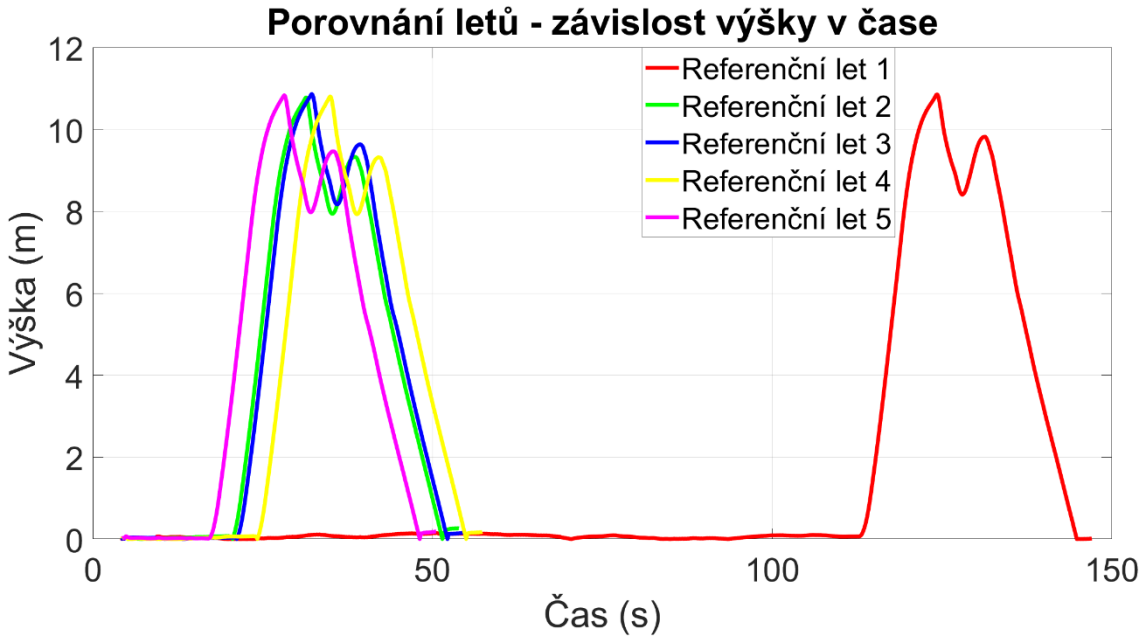
$$Hodnota\ parametru\ referenčního\ letu > Hodnota\ parametru\ modfikovaného\ letu$$

V této podmínce je porovnávána hodnota parametru, splněním této podmínky je získán výsledek „Modifikovaný let“. V případě, že tato podmínka není splněna, následuje text „Referenční let“. Na základě výčtu z předchozích výše zmíněných tabulek, byl zpracován souhrn pro lepší demonstraci jednotlivých porovnání. Tento souhrn je rozčleněn do dvou tabulek a to dle daného intervalu.

3 Prezentace výsledků

3.1 Ověření konzistence simulovaných letů

Na základě metodického přístupu ověřování konzistence simulovaných letů v prostředí Gazebo byl vytvořen obrázek (obr. 24). Na tomto obrázku je zobrazeno 5 referenčních letů, které byly stejného provedení, pouze čas spuštění se lišil. Ačkoliv se jednotlivé lety mohou v konkrétních parametrech mírně lišit, důraz byl kladen na trend funkce. Ten se u jednotlivých letů prakticky shoduje. Pro specifičtější analýzu bylo zpracováno porovnání maximálních výšek amplitudy jednotlivých letů.



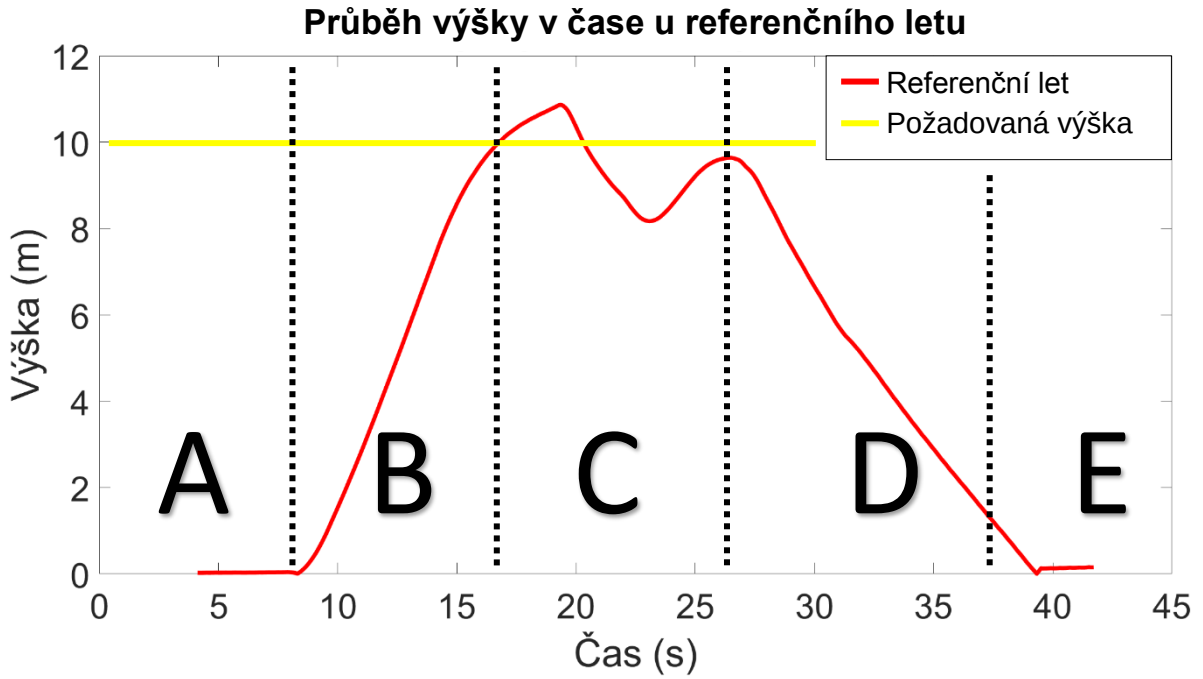
Obrázek 24: Porovnání konzistence referenčních letů

Tyto výšky amplitudy jsou:

- Referenční let 1 - 10.8597 (100%)
- Referenční let 4 - 10.8507 (99,92%)
- Referenční let 2 - 10.8287 (99,72%)
- Referenční let 5 - 10.7995 (99,45%)
- Referenční let 3 - 10.7751 (99,22%)

Maximální rozdíl je 0.78 %, ted méně než 1%, v tomto případě se jedná pouze o časový posun jednotlivých letů.

3.2 Referenční let



Obrázek 25: Průběh výšky v čase u výchozího letu

Referenční let byl proveden prostřednictvím řídicí stanice QGroundControl. V grafu (obr. 25) je znázorněn průběh celého letu. Referenční let byl rozdělen do jednotlivých intervalů (A, B, C, D, E). V intervalu B, ve vzletové fázi, bylo zaznamenáno téměř lineární stoupání do požadované výšky, poté křivka protnula požadovanou výšku a pokračovala v rostoucím trendu na intervalu C. Dále dosáhla křivka svého maxima kolem 11 metrů, odsud následoval strmý pokles hodnoty výšky, který se zastavil až téměř u hodnoty 8 metrů. Z tohoto bodu výška letu opět narůstala, a blížila se k požadované výšce letu, tu však nakonec po druhé neprotla. Těsně pod křivkou byl dosažen vzdálenostní bod a následovalo strmý pokles, který je zřejmý na celím intervalu D. Po dosažení 1 metru, pokračovala křivka k nule, kde bylo provedení letu ukončeno.

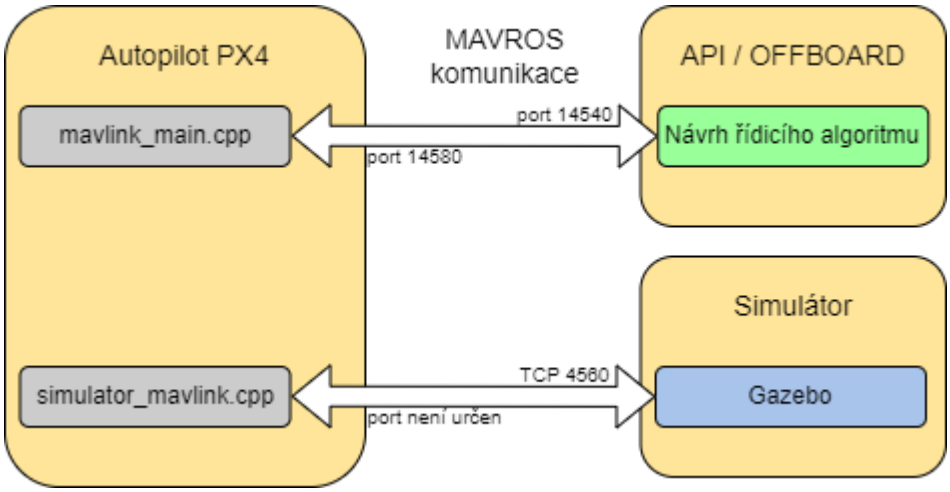
V rámci tohoto letu proběhlo automatické ladění jednotlivých PID kontrolérů. Výsledné hodnoty jednotlivých složek byly zaznamenány do tabulky (tab. 1) níže:

Tabulka 1: Hodnoty PID kontroléru získané automatickým laděním

	P	I	D
Výškový PID kontrolér	12.687	0	0
Pitch PID kontrolér	1	0.285	0.0033
Roll PID kontrolér	1	0.275	0.0043

3.3 Modifikovaný let

Implementace modifikovaného letu byla tvořena řídicím algoritmem, kterým byl skript. Provedení letu pomocí skriptu značně změnilo celé simulační schéma (obr. 26).



Obrázek 26: Modifikované simulační schéma komunikace

Simulační schéma bylo redukováno pouze na 3 bloky. V rámci těchto bloků byl skript (*Návrh řídicího algoritmu*) integrován do bloku API / OFFBOARD, čímž byla zprostředkována přímá komunikace s autopilotem PX4, konkrétně s kořenovým souborem *mavlink_main.cpp*. Touto integrací je eliminována softwarová kontrolní stanice QGroundControl. Eliminací kontrolní stanice došlo ke ztrátě sledování jednotlivých parametrů v reálném čase. Ztráta je následně nahrazena zahrnutím systémových zpráv *ROS_INFO* v skriptu (Příloha 1). Tyto zprávy byly publikovány v rámci TUI terminálu.

3.3.1 Modifikace PID kontroléru

Implementace řídicího systému u modifikovaného letu byla realizována tak, že nejprve byly získány hodnoty K_u a T_u dle metodického postupu Ziegler-Nicholsovy metody ladění. Hodnoty byly získávány pro každý PID kontrolér zvlášť. K dosažení optimálních hodnot, byla zkoumána oscilace během jednotlivých fází letu, tím jsou intervaly vzlet - B a snížení výšky na 1 metr - D, ale také průběh horizontálního letu, tedy na intervalu C. Kde byl sledován parametr udržení výšky v určených mezích, za předpokladu nastavených hodnot pro pitch a roll, které byly stanoveny dle vlastního uvážení na hodnotu 0.05 radiánů, tedy na 2.87 stupňů a každý let, který činil odchylku dosažené výšky mezi počátečním a konečným bodem C více než 2.5 metrů, byl automaticky brán jako nedostačující, jelikož v reálném světě by byla ohrožena bezpečnost. Manuální ladění, včetně implementace Ziegler – Nicholsovy metody, bylo zpracováno u letů, kde byly získávány hodnoty pro PID kontroléry v maximálním rozmezí 1.5 metrů.

Získané hodnoty K_u a T_u jsou prezentovány v následující tabulce (tab. 2):

Tabulka 2: Získané hodnoty Ku a Tu

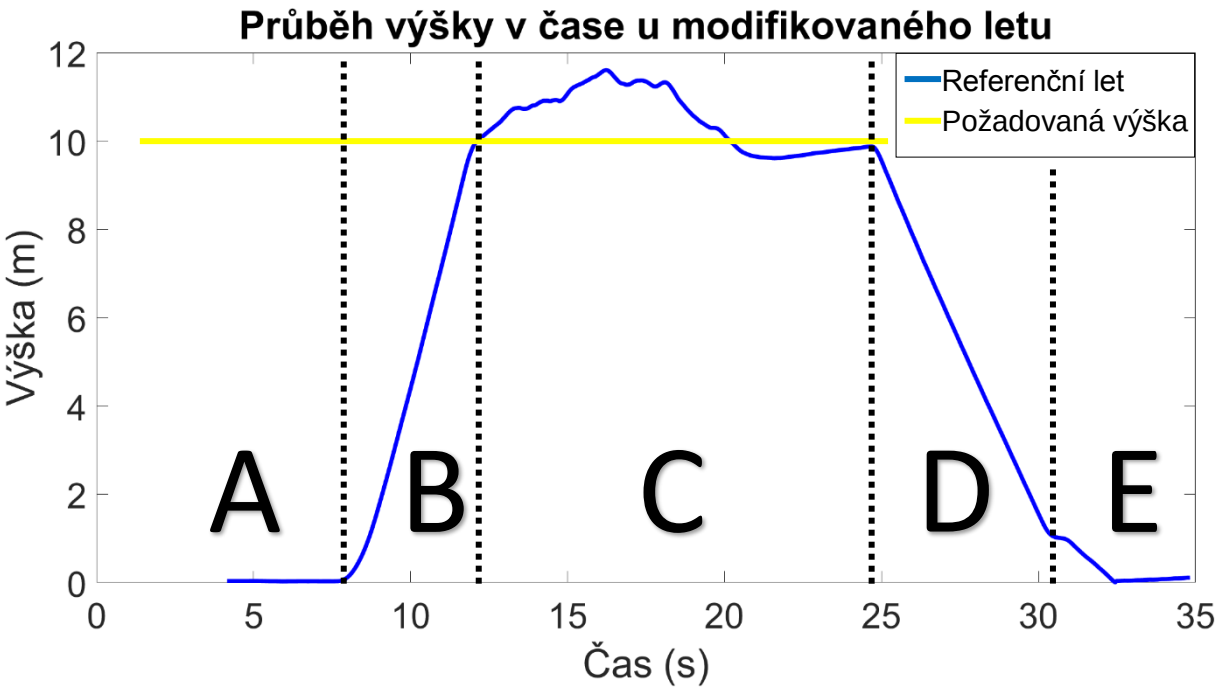
	K_u	T_u
Výškový PID kontrolér	$2.0211 * 10^{-5}$	4.1785
Pitch PID kontrolér	$6.1157 * 10^{-8}$	4.0582
Roll PID kontrolér	$3.6944 * 10^{-7}$	4.2638

Výsledné hodnoty K_u a T_u byly modifikovány pomocí empirických vzorců Ziegler – Nicholsovy metody. V následující tabulce (tab. 3) jsou modifikované hodnoty pro jednotlivé PID kontroléry

Tabulka 3: Hodnoty PID kontroléru získané manuálním laděním a implementací ZN

	P	I	D
Výškový PID kontrolér	$1.213 * 10^{-5}$	$9.674 * 10^{-6}$	$1.056 * 10^{-5}$
Pitch PID kontrolér	$3.669 * 10^{-8}$	$3.014 * 10^{-8}$	$3.102 * 10^{-8}$
Roll PID kontrolér	$2.217 * 10^{-7}$	$1.733 * 10^{-7}$	$1.969 * 10^{-7}$

3.3.2 Simulace modifikovaného letu



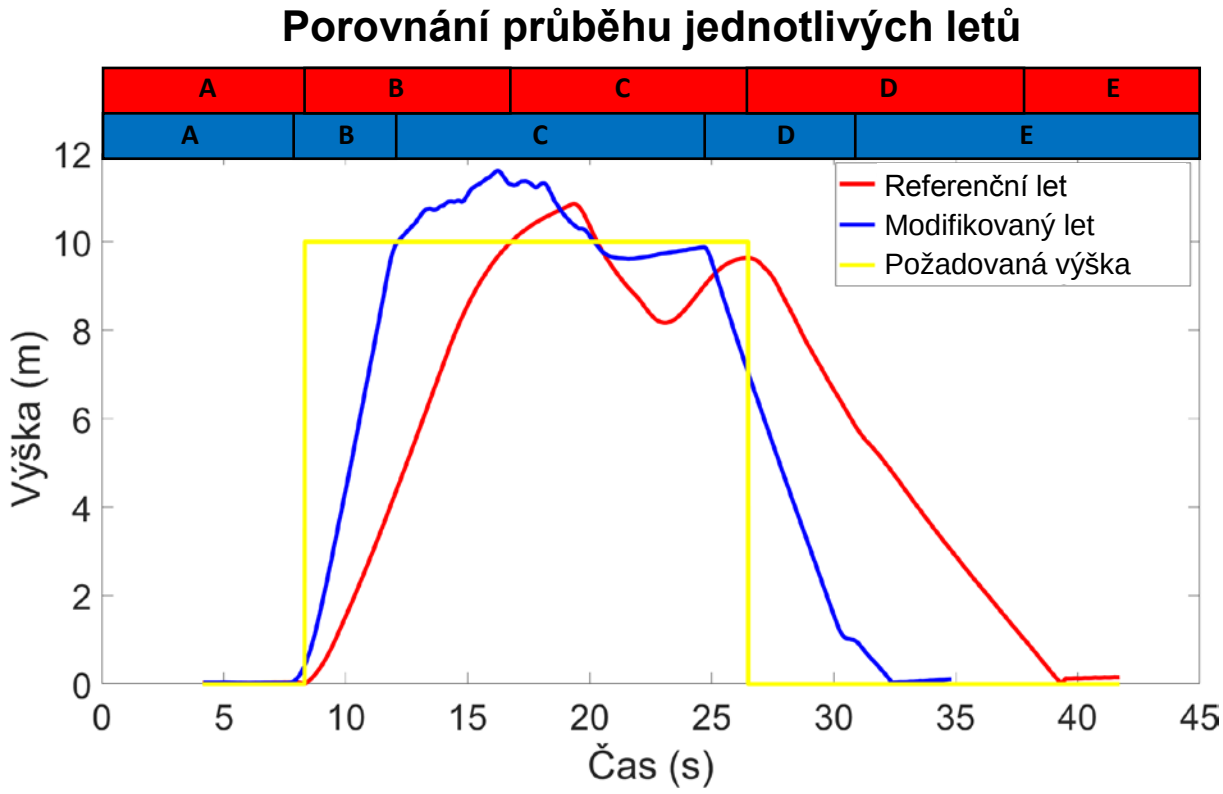
Obrázek 27: Průběh výšky v čase u modifikovaného letu

Modifikovaný let byl proveden integrací skriptu v rámci API / OFFBOARD. V grafu (obr. 27) je znázorněn kompletní let v závislosti na čase. Modifikovaný let byl rozdělen do intervalů (A, B, C, D, E). Průběh křivky po inicializaci systému se dostal do vzletové fáze – B, dále je z grafu zřejmé, že tvar křivky na tomto intervalu je lineární. Po dosažení požadované výšky letu, přešel let do intervalu C kde následovalo zmírnění nárůstu, které mírně oscilovalo. Tento průběh byl zaznamenán až do výšky téměř 2 metrového rozdílu vůči požadované výšce.

Poté došlo k trendu klesání, kde hodnota výšky poklesla až těsně pod požadovanou výšku. Poté následoval velmi mírný nárůst až do bodu, kdy bylo dosaženo vzdáleného bodu. V tomto bodě začal interval D, tedy pokles do výšky jednoho metru. Na tomto intervalu se funkce křivky zdá být téměř lineární až těsně před koncem, kde je prostředek zbržděn. Po zbrždění rychlosti následovalo systémové přistání a ukončení provádění letu.

3.4 Komparace letů

Porovnání refererenčního letu s modifikovaným letem bylo sdruženo do jednoho grafu (obr. 28). Prezentace porovnání byla znázorněna pomocí baru do intervalů ke každému letu. Z toho plyne, že červená linie je určená pro referenční let a modrá linie je pro modifikovaný let. V grafu je dále znázorněna požadovaný výšky, ze které vyplývá zda se provedení jednotlivých letů vyskotovalo pod požadovanou výškou či nad ní.



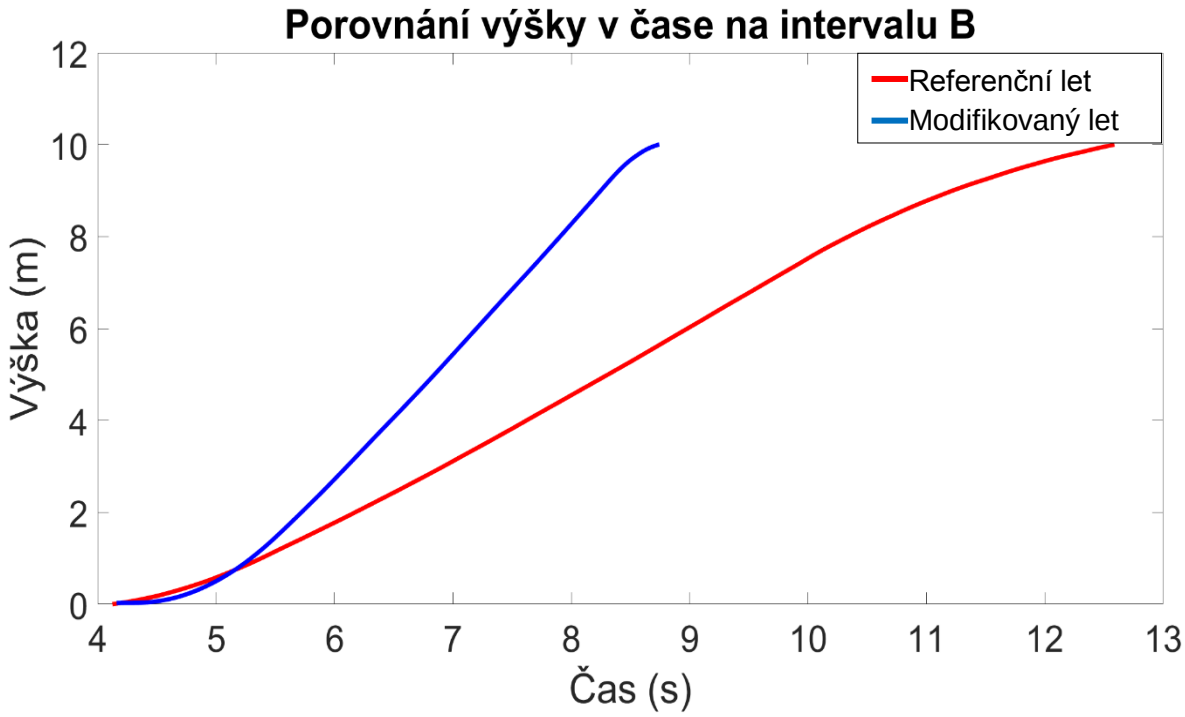
Obrázek 28: Porovnání průběhu celý letů

Podrobnější popis byl rozdělen do šetřených intervalů, kde byla popsána charakteristika jednotlivých letů v rámci daného intervalu. Analýza proběhla na intervalech:

- Interval B – Vzlet
- Interval D – Klesání do jednoho metru

3.4.1 Analýza a popis letu v intervalu B

V grafu (obr. 29) jsou prezentovány jednotlivé lety dle stejné barevné prezentace jako tomu bylo v rámci kompletního porovnání. Podrobnější analýza byla zaměřena v první řadě na čas dosáhnutí požadované výšky. V grafu je zobrazena červená křivka referenčního letu, která vyvozuje mírnější nárůst výšky vůči modifkované modré křivce.

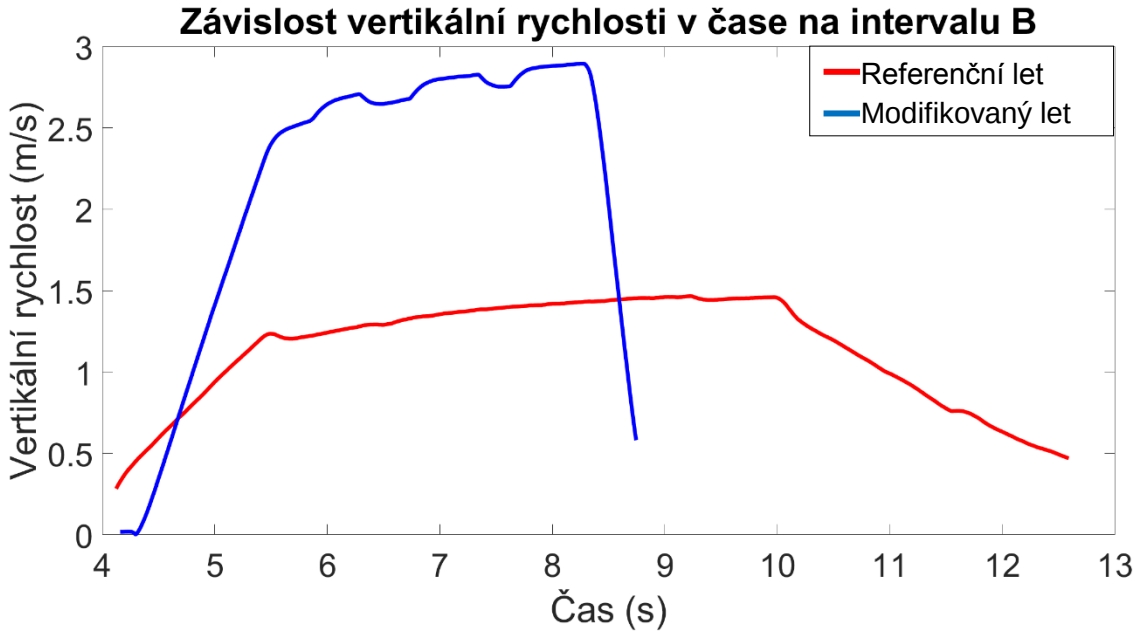


Obrázek 29: Průběh dosahování letové výšky na intervalu B

Časové hodnoty jednotlivých provedení jsou prezentovány níže:

- Referenční let – 8.46s
- Modifikovaný let – 4.58s

Dosažení požadované výšky je závislé na rychlosti, kterou se bezpilotní prostředek pohybuje. Z grafu (obr. 30) je patrné, že jednotlivá provedení přistoupily k dosahování výšky rozdílným způsobem. V rámci modifikovaného letu, byl z počátku zaznamenán prudký trend vzhůru, který se až v hodnotách 2.5 $m * s^{-1}$ začal zpomalovat, kde mírně osciloval poté došlo k přiblížení ke 3 $m * s^{-1}$, kde následoval prudký pokles vertikální rychlosti, čímž došlo k zbrždění vertikální rychlosti těsně před dosažený požadované výšky. Referenční let, operoval v celém průběhu s poněkud klidnějšími hodnotami, které se pohybovaly kolem 1 ~ 1.5 $m * s^{-1}$. Dále v analýze vertikální rychlosti byly změřeny hodnoty průměrné rychlosti.

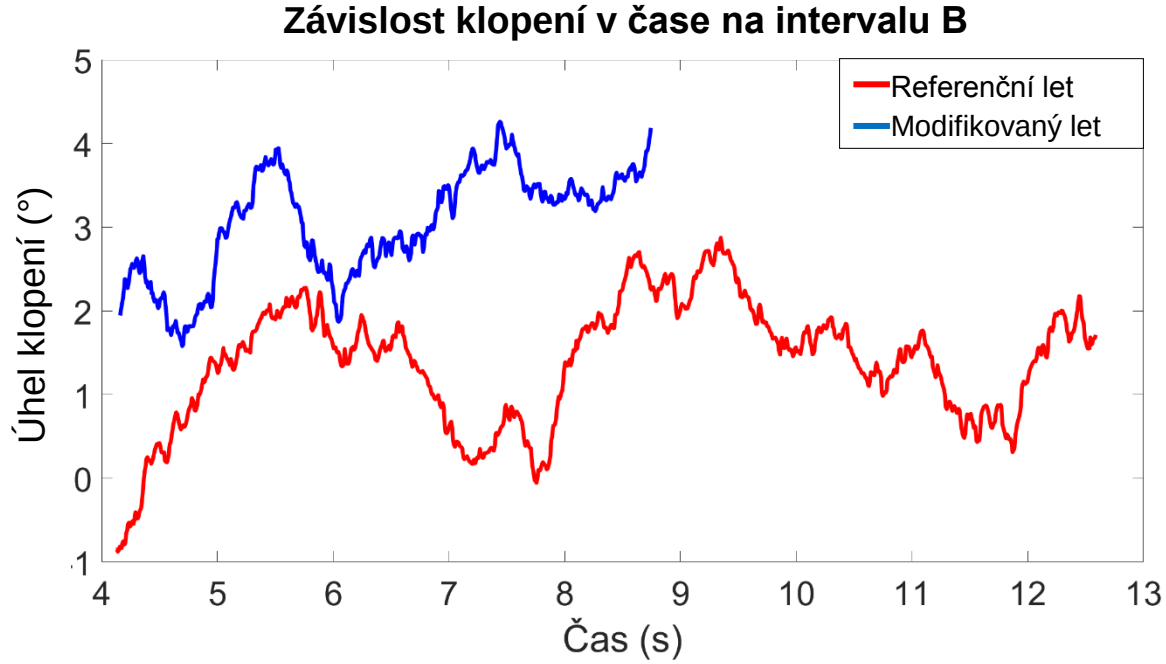


Obrázek 30: Závislost vertikální rychlosti v čase na intervalu B

Změřením průměrných rychlostí byly dosaženy tyto hodnoty:

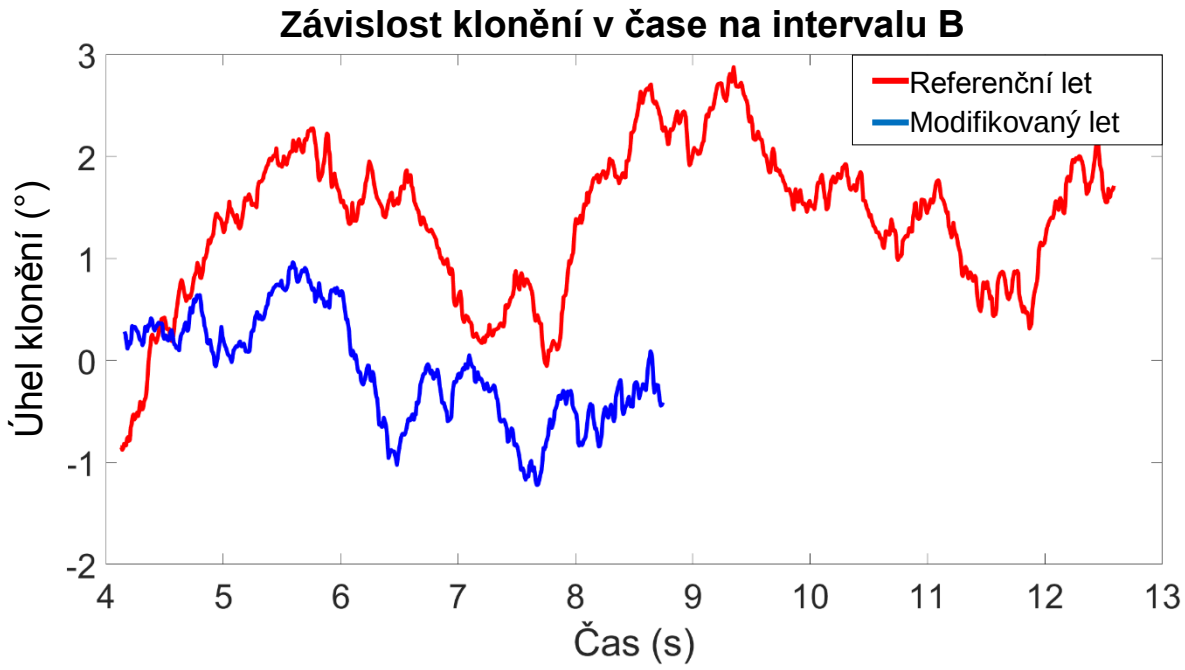
- Referenční let – 1.14 $m * s^{-1}$
- Modifikovaný let – 2.16 $m * s^{-1}$

V grafu (obr. 31) je znázorněn průběh klopení na intervalu B. Průběh je znázorněn ve stupních. Průběh křivky referenčního letu se v počátku pohyboval u záporné u hodnoty 1, poté osciloval mez hodnotami 0 a 2. Oscilace modifikovaného letu byla ve větších úhlových hodnotách a ze začátku oscilace proběhla mezi 2 a 4, poté došlo k nárůstu, který se pohyboval mezi 3 a 4.



Obrázek 31: Závislost klopení v čase na intervalu B

V grafu (obr. 32) je znázorněn průběh klonění na intervalu B. Provedení referenčního letu vycházelo v počátku ze záporné hodnoty, poté došlo k prudkému nárůstu až k hodnotě 2. Poté následoval pokles úhlu až k nule. Dále pokračoval opět nárůst, který dále osciloval v rozmezí 1 – 3 stupně. V referenčním letu vidíme úhel klopení, který začíná v záporné hodnotě 1. Modifikovaný let se osciloval v menším rozmezí a to mezi 1 a -1.



Obrázek 32: Závislost klonění v čase na intervalu B

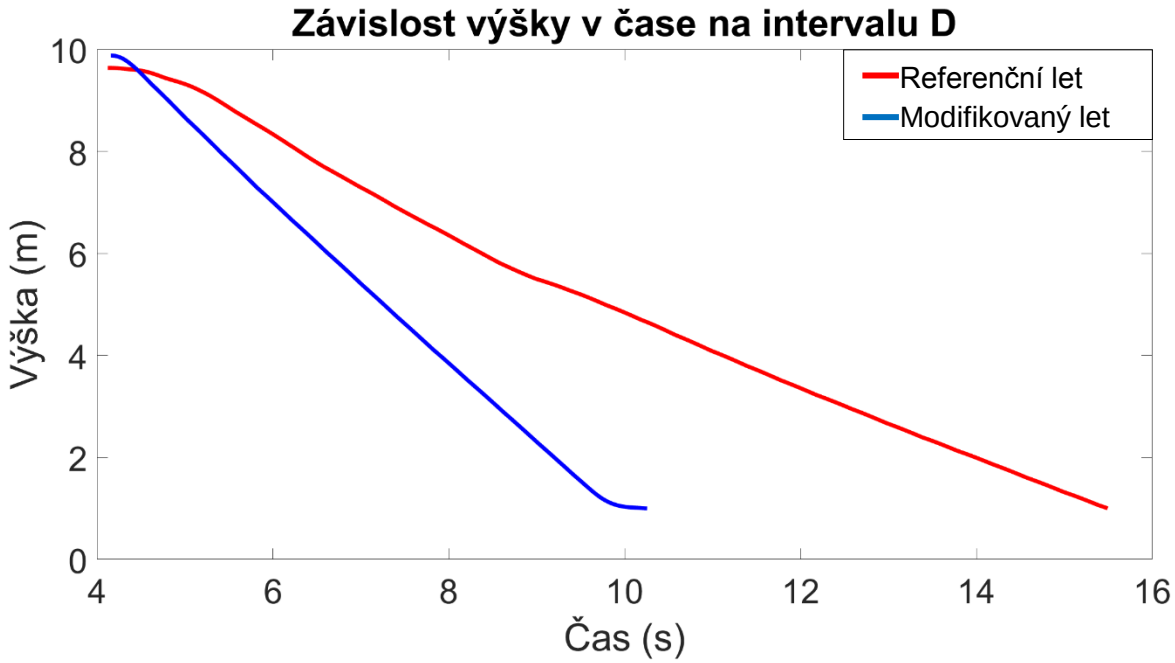
V tabulce (tab. 4) jsou prezentovány jednotlivé parametry, které byly získány během měření a vychází z metodiky, která se zaměřuje na experimentální testování, kde byly definovány výkonnostní parametry pro validaci robustnosti řídicího systému.

Tabulka 4: Získané hodnoty výkonnostních ukazatelů na intervalu B

	ALT 1B	**ALT 2B**	**VEL 1B**	**VEL 2B**
Doba náběhu	5.84	2.96	0.14	0.42
Doba dosažení vrcholu	12.59	8.74	9.23	8.82
Překmit	-	-	212%	399%
Doba ustálení	12.59	8.74	12.59	8.74
	PITCH 1B	**PITCH 2B**	**ROLL 1B**	**ROLL 2B**
Doba náběhu	0.93	1.25	0.93	1.38
Doba dosažení vrcholu	9.35	7.44	9.35	5.92
Překmit	67%	1.73%	68%	-333%
Doba ustálení	12.59	8.74	12.59	-

3.4.2 Analýza a popis letu v intervalu D

V grafu (obr. 33) je prezentován průběh dosahování 1 metru na intervalu D. V grafu je zobrazena červená křivka referenčního letu, která vyvozuje mírnější pokles výšky vůči modifkované modré křivce.

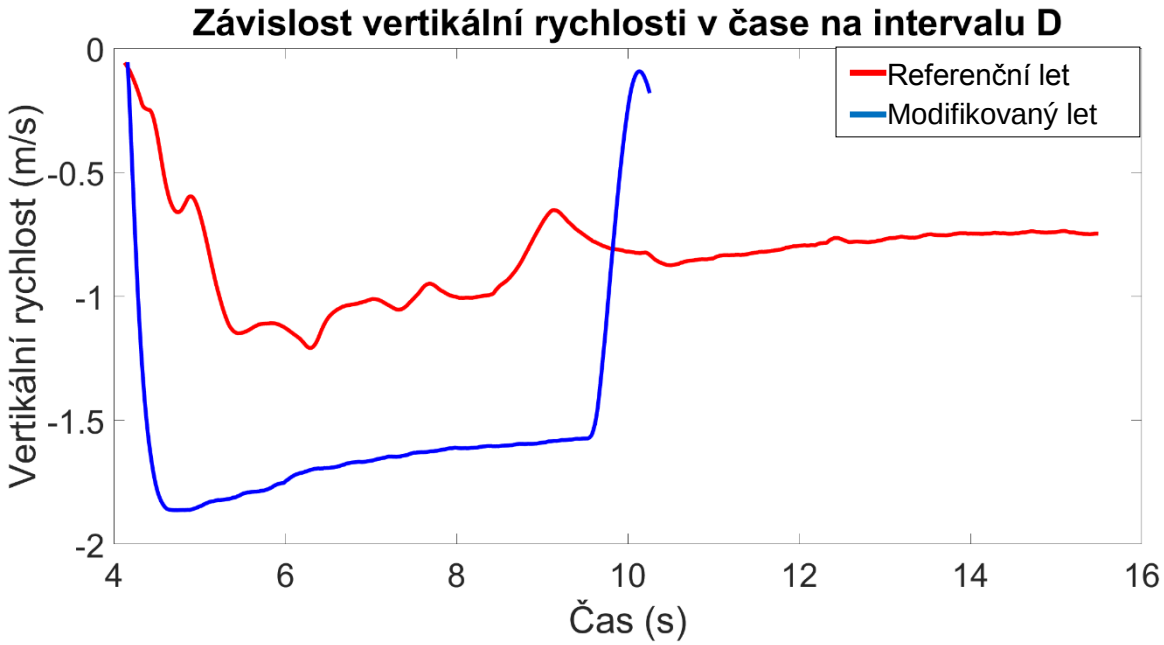


Obrázek 33: Závislost výšky v čase na intervalu D

Časové hodnoty jednotlivých provedení jsou prezentovány níže:

- Referenční let – 11,37s
- Modifikovaný let – 6.10s

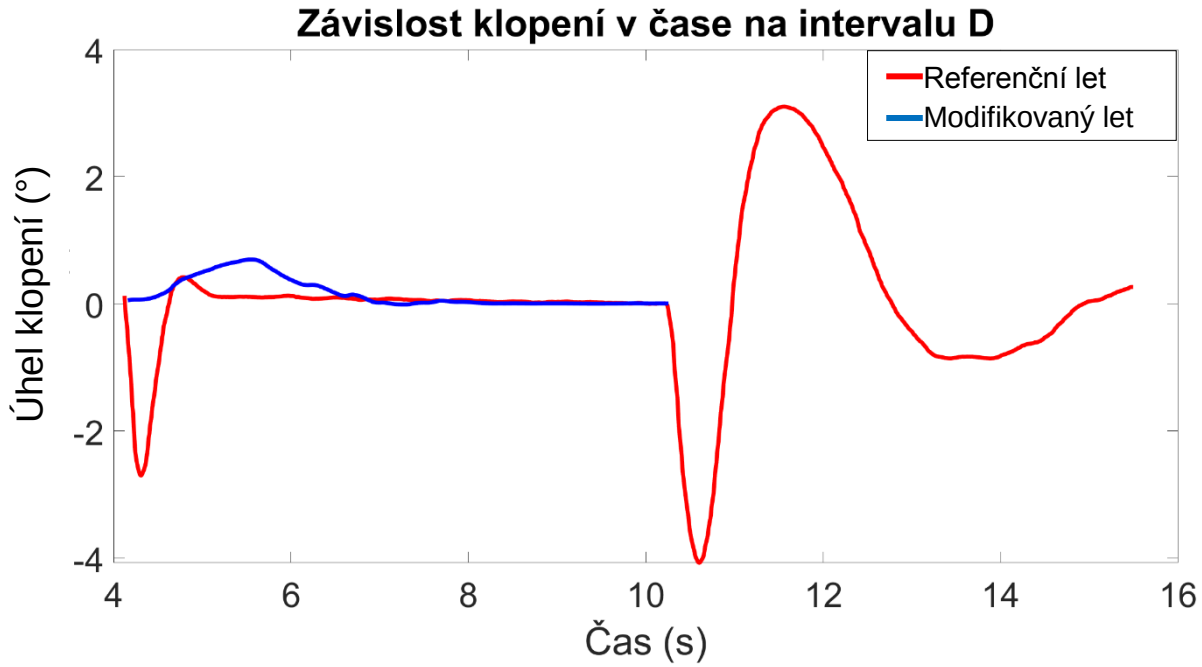
Z grafu (obr. 34) je zřejmé, že přístup jednotlivých trendů křivky byl při dosahování výšky 1 metr odlišný. V rámci modifikovaného letu, byl z počátku zaznamenán prudký klesající trend, který se zpomalil až téměř u -2 $m * s^{-1}$. V tomto bodě následoval velmi mírný nárůst, který setrval až těsně před dosažením 1 m. Před dosažením jednoho metru následoval velmi prudký nárůst vertikální rychlosti, který téměř protnul nulu, poté následovala ještě mírná úprava záporné rychlosti, kde byla splněna podmínka dosažení jednoho metru. U referenčního letu bylo průběh křivky s klesajícím trendem s mírnou oscilací až do záporných hodnot mírně za 1 $m * s^{-1}$. Poté byl křivka začala stoupat a bezpilotní prostředek postupně zpomalovat. V koncovéfázi je zachycena pouze mírná oscilace kolem záporné hodnoty -0.75.



Obrázek 34: Závislost vertikální rychlosti v čase na intervalu D

Změřením průměrných rychlostí byly dosaženy tyto hodnoty:

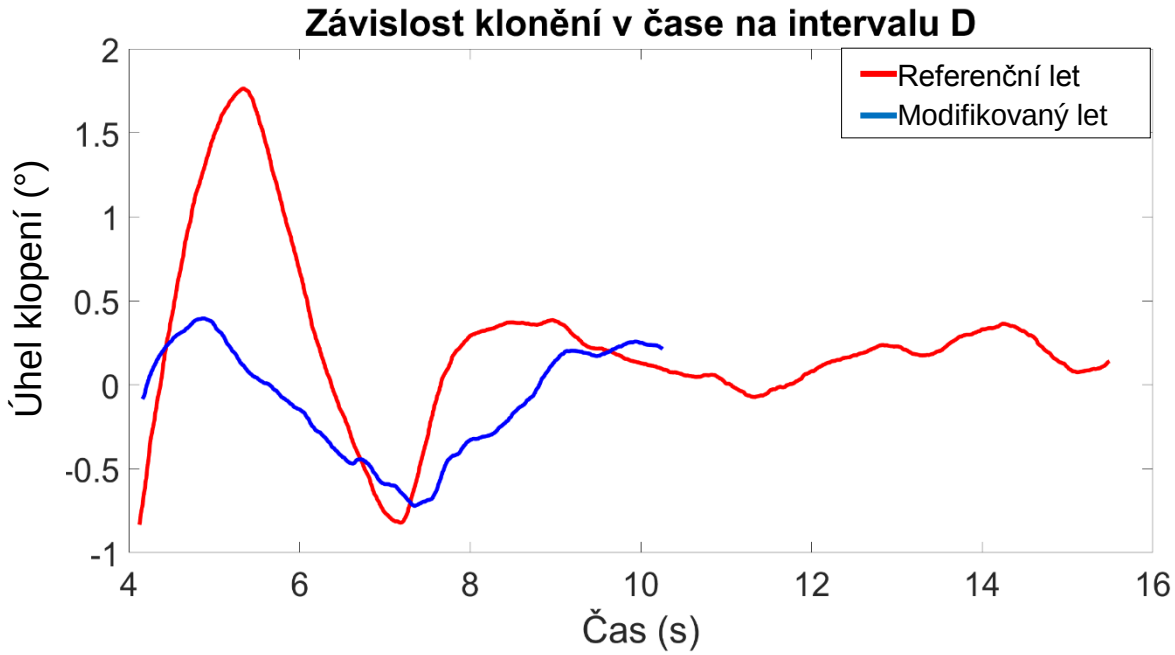
- Referenční let – 0.84 $m * s^{-1}$
- Modifikovaný let – 1.55 $m * s^{-1}$



Obrázek 35: Závislost klopení v čase na intervalu D

V grafu (obr. 35) je zobrazen vývoj klopení jednotlivých letů na intervalu D. Křivka modifikovaného letu měla pouze ze začátku mírný nárůst, který mířil k 1 stupni, poté se však ustálila na nule. U červené křivky referenčního letu nastal oscilace, která se ustálila kolem

hodnoty 0 stupňů. Při poklesu výšku však došlo k anomální oscilaci, která byla způsobena tím, že provedení referenčního letu po dosáhnutí 1 metru pokračovalo až do nulové výšky.



Obrázek 36: Závislost klonění v čase na intervalu D

Graf (obr. 36) poskytuje informace o klonění v intervalu D. V počátku tohoto intervalu byla zaznamenáná velká oscilace u referenčního letu, která se po čase zmírnila do rozmezí 0° – 0.5° klonění. U provedení modifikovaného letu byl zaznamenán průběh v rozmezí 0.5° - -0.5°. Tato oscilace setrvala po celou dobu intervalu D.

Tabulka 5: Získané hodnoty výkonnostních ukazatelů na intervalu D

	ALT 1D	**ALT 2D**	**VEL 1D**	**VEL 2D**
Doba náběhu	0	0	0.89	0.02
Doba dosažení vrcholu	4.13	4.16	6.29	4.74
Překmit	-	-	62%	927%
Doba ustálení	15.49	10.26	15.49	10.26
	PITCH 1D	**PITCH 2D**	**ROLL 1D**	**ROLL 2D**
Doba náběhu	0.55	0.01	0.03	0.15
Doba dosažení vrcholu	11.56	7.26	5.35	4.89
Překmit	-174%	213 %	-121%	85%
Doba ustálení	-	10.26	-	-

V tabulce (tab. 5) jsou prezentovány jednotlivé parametry, které byly získány během měření. V řádku *Doba ustálení* pro klonění a klopení nebyly jednotlivé signály provedení ustáleny, z čehož plyne, že nebyla získána hodnota číselného typu.

3.4.3 Souhrn výsledků z obou intervalů a porovnání řídicích systémů

V intervalu B – vzlet (tab. 6), byly srovnávány jednotlivé hodnoty parametrů pro výšku (ALT B), vertikální rychlost (VEL B), klopení (PITCH B) a klonění (ROLL B). V případě, že jsou alespoň 3 stejné výsledky hodnot, je dosažena kvalifikace letu „Modifikovaný let“ nebo „Referenční let“. Pokud výsledky hodnot vyšly pouze s 50% úspěšností, není kvalifikace určena a v tabulce byla tato informace označena jako „NELZE URČIT“.

Tabulka 6: Porovnání výkonnostních ukazatelů na intervalu B

	ALT B	**VEL B**
	Referenční let - Modifikovaný let	Referenční let - Modifikovaný let
Doba náběhu	Modifikovaný let	Referenční let
Doba dosažení vrcholu	Modifikovaný let	Modifikovaný let
Překmit	-	Referenční let
Doba ustálení	Modifikovaný let	Modifikovaný let
Vyhodnocení	Modifikovaný let	NELZE URČIT
	PITCH B	**ROLL B**
	Referenční let - Modifikovaný let	Referenční let - Modifikovaný let
Doba náběhu	Referenční let	Referenční let
Doba dosažení vrcholu	Modifikovaný let	Modifikovaný let
Překmit	Modifikovaný let	Modifikovaný let
Doba ustálení	Modifikovaný let	Referenční let
Vyhodnocení	Modifikovaný let	NELZE URČIT

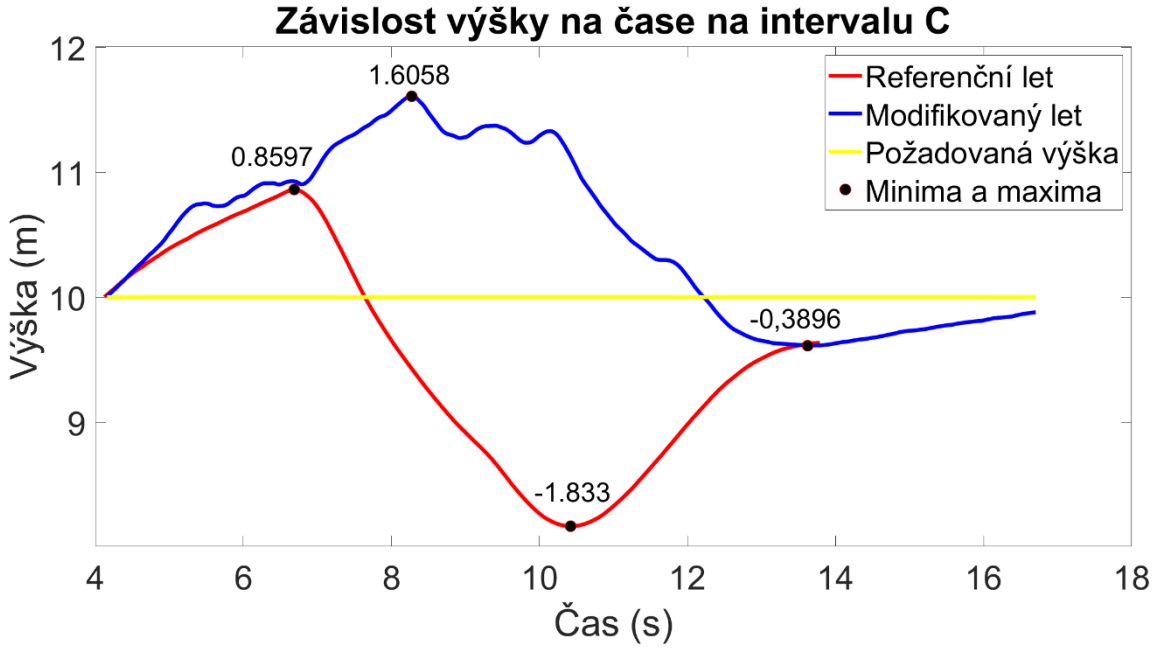
V intervalu D – klesání do jednoho metru v tabulce (tab. 7), jsou znázorněny vyhodnocení výkonnostních ukazatelů jednotlivých parametrů na intervalu D.

Tabulka 7: Porovnání výkonnostních ukazatelů na intervalu D

	ALT D	VEL D
	Referenční let - Modifikovaný let	Referenční let - Modifikovaný let
Doba náběhu	Referenční let	Modifikovaný let
Doba dosažení vrcholu	Referenční let	Modifikovaný let
Překmit	-	Referenční let
Doba ustálení	Modifikovaný let	Modifikovaný let
Vyhodnocení	Referenční let	Modifikovaný let
	PITCH D	**ROLL D**
	Referenční let - Modifikovaný let	Referenční let - Modifikovaný let
Doba náběhu	Modifikovaný let	Referenční let
Doba dosažení vrcholu	Modifikovaný let	Modifikovaný let
Překmit	Referenční let	Modifikovaný let
Doba ustálení	-	-
Vyhodnocení	Modifikovaný let	Modifikovaný let

3.4.4 Porovnání vývoje výšky na intervalu C

Pro komplexnější analýzu letu byl brán ohled i na průzkum mimo výše zmíněné intervaly. V tomto případě na interval horizontálního letu. V následujícím grafu (obr. 37) je prezentován interval C, kde byly porovnávány křivky jednotlivých letů vůči požadované výšce. Z grafu je patrné, že po dosažení požadované výšky, začaly oba lety ještě dále stoupat. Hodnoty stoupání u modifikovaného letu se zastavily až po překročení rozdílu 1.6 metrů mezi výškou modifikovaného letu a požadovanou výškou letu, následovala mírná oscilace a poté strmý pokles výšky až do záporné diference -0.3896 m. Poté ještě následoval mírný nárůst výšky a tím byl ukončeno provedení modifkovaného letu na tomto intervalu. Hodnota stoupání



Obrázek 37: Závislost výšky na čase na intervalu C

referenčního letu byla zastavena na 0.8597 m nad požadovanou výškou. Poté následoval strmý pokles hodnot výšky, až téměř k 2 metrovému rozdílu vůči požadované výšce. Dále křivka letu začala opět narůstat do doby, než byl dosažen vzdálenostní bod. Dosažení tohoto bodu znamenalo ukončení průzkumu na tomto intervalu.

Okometrická analýza definovala minima a maxima, respektive rozdílové hodnoty vůči požadované výšce, pro jednotlivé provedení letu na intervalu C. Tyto maxima a minima jsou:

Referenční let

- Maximum – 0.8597
- Minimum – - 1.833

Modifikovaný let

- Maximum – 1.6058
- Minimum – - 0.3896

Dále na tomto intervalu byla provedena analýza procentuální plochy pod a nad požadovanou výškou.

Hodnoty plochy pod referenční výškou a nad referenční výškou:

Referenční let

- Procentuální podíl pod křivkou – 99.05%
- Procentuální podíl nad křivkou – 0.95%

Modifikovaný let

- Procentuální podíl pod křivkou – 94.7%
- Procentuální podíl nad křivkou – 5.3%

Kromě výše zmíněných metrik byl součástí analýzy proveden také výpočet směrodatné odchylky pro jednotlivé lety. Pomocí směrodatné odchylky je zjištěna variabilita dat. V tomto ohledu bylo díky směrodatné odchylce kvantifikováno kolísání výšky letu kolem průměrné hodnoty výšky.

V rámci referenčního letu byla zaznamená hodnota směrodatné odchylky 0.8654, z čehož plyne, že většina hodnot výšky u referenčního letu se nachází v rozmezí $\pm$ 0.8654 od průměrné výšky letu.

Pro modifikovaný let byla stanovena hodnota směrodatné odchylka 0.6687, z čehož plyne, většina hodnot výšky v rámci modifikovaného letu se nachází v rozmezí $\pm$ 0.6687 metrů od průměrné výšky letu.

Diskuze

Prezentované výsledky z výkonnostních ukazatelů naznačují, že je možné docílit lepší výkonnosti navržením a manuálním laděním PID kontroléru. Obecně bylo dosaženo rychlejší odezvy systému řídicího algoritmu, a to jak u intervalu vzletu – B, tak u intervalu přistání či zklesání do jednoho metru D. V obou intervalech převažuje řídicí systém modifikovaného letu.

Kromě analýzy na intervalech vzletu a přistání byl doplňkově prozkoumán interval C, tedy pohyb od bodu mezi vzletem k bodu přistání. Zde byly zjišťovány hodnoty procentuálních podílů ploch pod a nad požadovanou výškou letu a dále definice kritických bodů. Z výsledných hodnot podílů ploch pod a nad křivkou vychází lépe modifikovaný let z pohledu, že je častěji nad požadovanou výškou a nepohybuje se pouze pod ní. Modifikovaný let poskytuje bezpečnější řešení z hlediska letu ve velmi nízkých výškách. Provedení modifikovaného letu je spíše vedeno nad požadovanou výškou v porovnání s referenčním provedením letu, u kterého je výška snížena až v rozdílu dvou metrů oproti požadované výšce. Klesání do takového bodu, je vnímáno jako kritické, kdy může dojít ke střetu s objekty či přímo se zemí.

Navržení kontrolního algoritmu sebou nese úpravu simulačního schématu autopilota PX4. Úprava simulačního schématu spočívá v eliminaci grafického prostředí virtuální kontrolní stanice, která umožňuje sledování dat či ovládání bezpilotního prostředku. Sledování dat spočívá v definováním, která data je vhodné sledovat během letu, tyto data jsou vypisovány do konzole pomocí informačních zpráv. Díky této úpravě je zprostředkována přímá komunikace mezi řídicím systémem bezpilotního prostředku a autopilotem PX4. Přímá komunikace může vést k rychlejšímu přenosu dat mezi navržením řídicím systémem a autopilotem PX4.

Mimo analyzování výkonnostních ukazatelů, lze si pokládat otázku, zda zvolený 3D model dosahuje optimálních výsledků řídicího algoritmu. V případě změny modelu průběh výšky nebude stejný jelikož hmotnost včetně dalších aerodynamických parametrů nebudou shodné. Při volbě jiného modelu je tedy nutné opakovat manuální ladění včetně implementace vhodné vyhlazovací metody. Což vede k řešení, zda volba Ziegler – Nicholsovy metody je optimálním řešením výběru.

Prostor kam se vývoj řídicího algoritmu může dále ubírat je například ve zkoumání metod pro samotné vyhlazení signálu řídicího systému, Podrobná analýza různých metod může vést k odlišným výsledkům, které nemusí být vždy optimálním řešení pro vývoj v oblasti bezpilotních prostředků.

Samotný vývoj skriptu je další otázkou, která může bést k řešení. Obsahem navrženého skriptu je sekvenční forma, kdy jednotlivé fáze jdou postupně za sebou, je však možné definovat tento přístup pomocí letových režimů a v rámci provedení letu se tak přepínat mezi jednotlivými režimy.

K návrhu se vztahuje ještě jak přistupovat k danému skriptu a zda aplikovat PID kontrolér se stejnými hodnotami nebo pouze v jednotlivých intervalech. V rámci testování a validace proběhl test pouze na celém intervalu, kdy bylo manuální ladění zprostředkováno hned na začátku a poté se již pracovalo s definovanými hodnotami. Pokud však by byl využívány rozdílné hodnoty pro každou fázi letu bylo možné zajistit optimální hodnoty letu v jednotlivých intervalech s negativně ovlivněným časem, kdy v případě každého manuálního ladění na jednotlivých intervalech by musela být pozastavena simulace a poté analyzovány data pouze na tomto intervalu a z těch by byly vypočítány optimální hodnoty a takovýhle postup by byl pro každý interval zvlášť.

Z hlediska metodiky programování vlastního skriptu bylo zjištěno, že na to jak je velká kompatibilita prostředí autopilota PX4 s Gazebo simulátorem, včetně podpůrných nástrojů ROS. Tak se v rámci různých studií, setkáváme pouze s malým množstvím využitelných a spustitelných skriptů, které tvoří základní šablonu pro vytváření vlastních řešení. Vznikající projekty jsou tak odkázány na mnoho strávených hodin výzkumu, jak využívat systému publish – subscribe a jakým způsobem například volat jednotlivé proměnné v rámci informačních zpráv.

Kromě zaměření na návrh skriptu, by navazující výzkum mohl být zaměřen na optimalizaci letové konstrukce, která vychází z vybraného 3D modelu. S cílem dosažení lepší aerodynamiky hybridních bezpilotních prostředků s pevnou konstrukcí rotorů. Studie by mohla zahrnovat různé rozestavění rotorů na křídle, konkrétní tvar křídel či volba využitelného materiálu.

Mezi limitace patří absence vyzkoušení v praxi, kde může bezpilotní prostředek vyvozovat odlišné chování, než tomu bylo ve virtuálních podmínkách. Jako další limitaci této diplomové práce výběr konkrétního bezpilotního prostředku, u kterého je využit aplikovaný skript. Využitím řídicího algoritmu v odlišném modelu může vést k odlišnému chování systému. Což lze manuálním ladění postupem času odbourávat. Poslední limitací je implementace konkrétní metody pro ladění. Ziegler – Nicholsova metoda je hojně využívána k regulaci systémů, avšak není jedinou metodou, která se používá k regulaci systému. Proto je možné, že v případě jiné volby by bylo dosaženo odlišných výsledků.

Závěr

Tato práce se zaměřovala na implementaci vlastního návrhu řídicího algoritmu. Nejprve však bylo nutné vybrat model, který splňoval kritéria počtu akčních členů a zároveň byl plně kompatibilním modelem v prostředí Gazebo. Poté byl vytvořen skript v programovacím jazyce C++, který obsahoval vlastní návrh kontrolního algoritmu.K vytvoření vlastního návrhu byly využity nástroje a techniky, které se opírají o návrh kontrolních algoritmů. Během výzkumu se vycházelo z empirické techniky Ziegler – Nicholsovy metody ladění pro vyhlazení funkce kontrolního algoritmu. Návrhem kontrolního algoritmu prostředníctvím skriptu bylo eliminováno prostředí virtuální kontrolní stanice a díky tomu byla zajištěna komunikace řídicího algoritmu napřímo s autopilotem PX4.

Nejprve bylo provedeno testování konzistence letu prostřednictvím řídicí stanice QGround Control. V rámci tohoto testování bylo prověřeno, zda provedený let neobsahuje žádné anomálie a trend křivky jednotlivých letů je shodný, pouze je posunutý v čase. Výsledkem byl, že rozdíly mezi jednotlivými lety jsou takřka zanedbatelné neboť se lišily pouze v mezích méně než 1%. Po stanovení podmínky konzistence letu, proběhlo testování na dvou letech, které byly zprostředkovány pomocí různých technik, avšak za dodržení stejných podmínek letu, z toho vyplývá, že během obou letů bylo dodrženo pořadí jednotlivých fází, včetně dodržení požadovaných parametrů jako je výška či vzdálenost do které měl hybridní bezpilotní prostředek letět.

Prvním letem byl referenční let, který byl zprostředkován v prostředí QGroundControl, a jednalo se tak o standardizovaný postup provedení letu. Před provedením letu bylo využito nástroje pro automatické ladění, který zkalibroval bezpilotní prostředek tak, aby bylo během letu zajištěno optimálních hodnot. Tento let poskytl základ pro porovnání výsledných hodnot výkonnostních ukazatelů. Druhý let byl zprostředkován pomocí vlastního skriptu, který obsahoval PID kontroléry pro výšku, pitch a roll, včetně vyhlazovací metody signálu Ziegler-Nichols. Díky manuálnímu ladění, které je nutné zprostředkovat v rámci této metody, a následné integraci upravených hodnot pomocí empirických vzorců Ziegler-Nicholsovy metody byl zajištěn příznivější průběh modifikovaného letu v porovnání s referenčním letem.

Porovnání provedení letů bylo rozděleno do jednotlivých intervalů fáze letu. Jednotlivé testování se však zaměřovalo na průběh ve vzletové a přistávací fázi. Na intervalech vzletu a přistání byly porovnávány hodnoty výkonnostních ukazatelů, a kromě hodnot výkonnostních ukazatelů byly porovnány hodnoty času dosažení požadované výšky a vertikální rychlosti. Ve vzletové i přistávací fázi byly kratší časy dosaženy u modifikovaného letu. V celkovém porovnání, které bralo ohled jak na výkonnostní ukazatele, tak na vertikální

rychlost a dosažení požadované výšky, bylo zjištěno, že modifikovaný let vykazuje lepší hodnoty na intervalech vzletu a přistání.

Pro dosažení jasného pohledu na systém, který je robustnějším, bylo třeba pohlédnout i na průběh mimo testované intervaly pro doplnění užitečných informací. Ve fázi horizontálního letu byly měřeny minima a maxima, dosažené výšky jednotlivých provedení, a plochy pod křivkou a na křivkou. V rámci měřením extrémních bodů, byl nalezen bod, takzvaně kritického minima u referenčního letu a to v případě, kdy bezpilotní prostředek začal rapidně zklesávat pod požadovanou výšku s diferencí -1.833 pod požadovanou výšku. Bod kritického minima představuje riziko při nízkých letových výškách, kdy v případě jeho dosažení, může následovat nehoda, která může mít fatální následky.

Návrh řídicího systému společně s metodikou ladění a validace by mohl být dále využit pro výzkum řídicích systému. Vytvořený skript může posloužit jako základ pro vytváření různých aplikací v oblasti bezpilotních prostředků, jelikož je postaven sekvenčně je možné rozšiřovat a modifikovat jednotlivé sekvence.

Přínos této diplomové práci je v tom, že přináší užitečné poznatky pro vývoj hybridních bezpilotních prostředků a jejich řídicích systémů. Dosažené hodnoty modifikovaného PID kontroléru mohou být aplikovány na modely, které mají podobné či shodné vlasnosti s modelem, který je využíván v rámci diplomové práce. Kromě toho je možné tento skript exportovat do reálného modelu a validovat řešení v praxi. Avšak je možné, že v rámci reálného prostředí nemusí být chování hybridního bezpilotního prostředku zcela totožné s chováním ve virtuálním prostředí. Testování v reálném světě může poskytnout přesnější a relevantnější výsledky návrhu řídicího systému.

Seznam použité literatury

[1] CLOTHIER, Reece A., Dominique A. GREER, Duncan G. GREER a Amisha M. MEHTA, 2015. Risk Perception and the Public Acceptance of Drones [online]. 17. únor 2015. B.m.: Wiley. Dostupné z: doi:10.1111/risa.12330

[2] Manual on remotely piloted aircraft systems (RPAS). In: Store.icao.int [online]. Montréal, Québec, Canada: ICAO Store, 2012 [cit. 2023-05-06]. Dostupné z: https://store.icao.int/en/manual-on-remotely-piloted-aircraft-systems-rpas-doc-10019

[3] ANDERSON, John David, 1988. Introduction to flight: Its engineering and history. 3rd ed. New York, NY: McGraw-Hill. ISBN 9780070016415.

[4] ANDERSON, David W. a Scott EBERHARDT, 2009. Understanding flight, second edition. 2nd ed. New York, NY: McGraw-Hill Professional. ISBN 9780071626965.

[5] Bernoulli s Principle Example. In: GeeksforGeeks [online]. Boston, Massachusets, United States of America: Privacy Protect, 2009 [cit. 2023-05-06]. Dostupné z: https://media.geeksforgeeks.org/wp-content/uploads/20220823181307/PrincipleofLiftingofanAircraft.png

[6] Angle of Attack. In: SKYbrary [online]. Belgium: SKYbrary Aviation Safety, 2021 [cit. 2023-05-06]. Dostupné z: https://www.skybrary.aero/sites/default/files/AoA.jpg

[7] BEARD, Randal W. a Timothy W. MCLAIN. Small unmanned aircraft: theory and practice. Princeton: Princeton University Press, c2012. ISBN 978-0-691-14921-9.

[8] FAHLSTROM, Paul and Thomas GLEASON, 2012. Introduction to UAV Systems: Fahlstrom/introduction to UAV systems. 4th ed. Nashville, TN: John Wiley & Sons. ISBN 9781119978664.

[9] POUNDS, P., R. MAHONY a P. CORKE, 2010. Modelling and control of a large quadrotor robot [online]. červenec 2010. B.m.: Elsevier BV. Dostupné z: doi:10.1016/j.conengprac.2010.02.008

[10] LIU, Zhong, Yuqing HE, Liying YANG a Jianda HAN, 2017. Control techniques of tilt rotor unmanned aerial vehicle systems: A review [online]. únor 2017. B.m.: Elsevier BV. Dostupné z: doi:10.1016/j.cja.2016.11.001

[11] V22 Osprey. In: Q4cdn [online]. Unknown: Unknown, 2022 [cit. 2023-05-06]. Dostupné z: https://s1.q4cdn.com/535492436/files/images/2022/IMAGE_1.jpg

[12] F35B. In: Armyweb [online]. Praha: Pivoňka, 2012 [cit. 2023-05-06]. Dostupné z: https://www.armyweb.cz/images/f-35b.jpg

[13] WIEGAND, Chris, 2018. F-35 Air Vehicle Technology Overview [online]. 24. červen 2018. B.m.: American Institute of Aeronautics and Astronautics. Dostupné z: doi:10.2514/6.2018-3368

[14] VALAVANIS, Kimon P. a George J. VACHTSEVANOS, ed., 2015. Handbook of Unmanned Aerial Vehicles [online]. 2015. B.m.: Springer Netherlands. Dostupné z: doi:10.1007/978-90-481-9707-1

[15] BRISTEAU, Pierre-Jean, François CALLOU, David VISSIÈRE a Nicolas PETIT, 2011. The Navigation and Control technology inside the AR.Drone micro UAV [online]. leden 2011. B.m.: Elsevier BV. Dostupné z: doi:10.3182/20110828-6-it-1002.02327

[16] CAI, Guowei, Kai-Yew LUM, Ben M. CHEN a Tong H. LEE, 2010. A brief overview on miniature fixed-wing unmanned aerial vehicles [online]. červen 2010. B.m.: IEEE. Dostupné z: doi:10.1109/icca.2010.5524453

[17] Unmanned Aircraft Systems Integration in the National Airspace System. In: NASA [online]. Washington DC, United States of America: National Aeronautics and Space Administration, 2006 [cit. 2023-05-06]. Dostupné z: https://www.nasa.gov/centers/armstrong/news/FactSheets/FS-075-DFRC.html

[18] ELFEKY, Mahmoud, Moustafa ELSHAFEI, Abdul-Wahid A. SAIF a Mohamed F. AL-MALKI, 2016. Modeling and simulation of quadrotor UAV with tilting rotors [online]. 1. červen 2016. B.m.: Springer Science and Business Media LLC. Dostupné z: doi:10.1007/s12555-015-0064-5

[19] PAPACHRISTOS, Christos, Kostas ALEXIS a Anthony TZES, 2011. Design and experimental attitude control of an unmanned Tilt-Rotor aerial vehicle [online]. červen 2011. B.m.: IEEE. Dostupné z: doi:10.1109/icar.2011.6088631

[20] Model UAV With rotary rotors. In: Ly200-cdn [online]. Peking, China: Alibaba Cloud Computing (Beijing) Co., 2016 [cit. 2023-05-06]. Dostupné z: https://ueeshop.ly200-cdn.com/u_file/UPAK/UPAK499/1910/products/22/101f0a333d.jpg

[21] ORBEA, David, Jessica MOPOSITA, Wilbert G. AGUILAR, Manolo PAREDES, Rolando P. REYES a Luis MONTOYA, 2017. Vertical take off and landing with fixed rotor [online]. říjen 2017. B.m.: IEEE. Dostupné z: doi:10.1109/chilecon.2017.8229691

[22] Anon., 2005. Modelling and Control of Mini-Flying Machines [online]. B.m.: Springer-Verlag. Dostupné z: doi:10.1007/1-84628-179-2

[23] Fixed structure of the unmanned aerial vehicle. In: DefenceTurkey [online]. Çankaya-Ankara, Turkey: Defence Turkey, 2017 [cit. 2023-05-06]. Dostupné z: https://www.defenceturkey.com/files/content/5f402d6386d5a.jpg

[24] Novlit 3 tailsitter. In: ResearchGate [online]. Boston: ResearchGate, 2004 [cit. 2023-05-04]. Dostupné z: https://www.researchgate.net/publication/276379620/figure/fig1/AS:650840016687104@1532183760258/Novlit-3-tail-sitter-VTOL-MAV.png

[25] Fixed wing VTOL. In: PX4 [online]. Zurych, Swtizerland: Lorenz Meier, 2014 [cit. 2023-05-06]. Dostupné z: https://docs.px4.io/main/assets/img/hero_small.d4732a64.png

[26] YANG, Xuxi a Peng WEI, 2021. Autonomous Free Flight Operations in Urban Air Mobility With Computational Guidance and Collision Avoidance [online]. září 2021. B.m.: Institute of Electrical and Electronics Engineers (IEEE). Dostupné z: doi:10.1109/tits.2020.3048360

[27] STRAUBINGER, Anna, Raoul ROTHFELD, Michael SHAMIYEH, Kai-Daniel BÜCHTER, Jochen KAISER a Kay Olaf PLÖTNER, 2020. An overview of current research and developments in urban air mobility – Setting the scene for UAM introduction [online]. srpen 2020. B.m.: Elsevier BV. Dostupné z: doi:10.1016/j.jairtraman.2020.101852

[28] Zuri - technology. In: Zuri [online]. Praha, Czech Republic: GRANSY S.R.O, 2004 [cit. 2023-05-06]. Dostupné z: https://zuri.com/technology

[29] Zuri 2.0 Side. In: Evtol.news [online]. Unknown: GoDaddy.com, 2017 [cit. 2023-05-06]. Dostupné z: https://evtol.news/__media/Aircraft%20Directory%20Images/Zuri%202.0/Zuri-2.0-side-view.jpg

[30] Neoptera Technology. In: Evtol.news [online]. Unknown: GoDaddy.com, 2017 [cit. 2023-05-06]. Dostupné z: https://evtol.news/neoptera/

[31] Neoptera eOpter. In: TransportUP [online]. Unknown: GoDaddy.com, 2017 [cit. 2023-05-06]. Dostupné z: https://transportup.com/wp-content/uploads/2018/06/Neoptera_eOpter.png

[32] ALTI UAS. ALTI UAS [online]. Unknown: TUCOWS, 2016 [cit. 2023-05-06]. Dostupné z: https://www.altiuas.com

[33] ALTI Transition. I2.wp.com [online]. Unknown: MarkMonitor, 1997 [cit. 2023-05-06]. Dostupné z: https://i2.wp.com/www.marquesaviation.com/wp-content/uploads/2016/11/T1c.jpg?resize=1272%2C630&ssl=1

[34] YANGUO, Song a Wang HUANJIN, 2009. Design of Flight Control System for a Small Unmanned Tilt Rotor Aircraft [online]. červen 2009. B.m.: Elsevier BV. Dostupné z: doi:10.1016/s1000-9361(08)60095-3

[35] PX4 Architectural overview. PX4 [online]. Zurych,Switzerland: Lorenz Meier, 2014 [cit. 2023-05-06]. Dostupné z: https://docs.px4.io/main/en/concept/architecture.html

[36] AHMED, Bilal, Hemanshu R. POTA a Matt GARRATT, 2009. Flight control of a rotary wing UAV using backstepping [online]. 12. květen 2009. B.m.: Wiley. Dostupné z: doi:10.1002/rnc.1458

[37] ISLAM, S., M. FARAZ, R. K. ASHOUR, J. DIAS a L. D. SENEVIRATNE, 2015. Robust adaptive control of quadrotor unmanned aerial vehicle with uncertainty [online]. květen 2015. B.m.: IEEE. Dostupné z: doi:10.1109/icra.2015.7139417

[38] STAROLETOV, Sergey, 2021. Work-in-Progress Abstract: Revealing and Analyzing Architectural Models in Open-source ArduPilot [online]. srpen 2021. B.m.: IEEE. Dostupné z: doi:10.1109/rtcsa52859.2021.00034

[39] Description of sitl simulation environment. In: PX4 [online]. Zurych, Switzerland: Lorenz Meier, 2014 [cit. 2023-05-06]. Dostupné z: https://docs.px4.io/main/en/simulation/#sitl-simulation-environment

[40] Description of sitl simulation environment - image. In: PX4 [online]. Zurych, Switzerland: Lorenz Meier, 2014 [cit. 2023-05-06]. Dostupné z: https://docs.px4.io/main/assets/img/px4_sitl_overview.d5d197f2.svg

[41] PATEL, Ujjval, Nathan BAKER a Imraan FARUQUE, 2023. Seascape -- Simple Educational Autopilot System with C++ Architecture & Python Exterior [online]. 19. leden 2023. B.m.: American Institute of Aeronautics and Astronautics. Dostupné z: doi:10.2514/6.2023-1019

[42] JEELANI, Idris a Masoud GHEISARI, 2021. Safety challenges of UAV integration in construction: Conceptual analysis and future research roadmap [online]. prosinec 2021. B.m.: Elsevier BV. Dostupné z: doi:10.1016/j.ssci.2021.105473

[43] SHAKEEL, Tanzeela, Jehangir ARSHAD, Mujtaba Hussain JAFFERY, Ateeq Ur REHMAN, Elsayed Tag ELDIN, Nivin A. GHAMRY a Muhammad SHAFIQ, 2022. A Comparative Study of Control Methods for X3D Quadrotor Feedback Trajectory Control [online]. 15. září 2022. B.m.: MDPI AG. Dostupné z: doi:10.3390/app12189254

[44] RINALDI, Marco, Stefano PRIMATESTA a Giorgio GUGLIERI, 2023. A Comparative Study for Control of Quadrotor UAVs [online]. 8. březen 2023. B.m.: MDPI AG. Dostupné z: doi:10.3390/app13063464

[45] Regulace teploty. Dixell [online]. Unknown: Logitron, 2010 [cit. 2023-05-06]. Dostupné z: https://dixell.cz/wp-content/uploads/GRAF2.jpg

[46] ASTROM, Karl Johan and T. HAGGLUND, 1995. PID controllers: Theory, design and tuning. 2nd ed. Research Triangle Park: ISA. ISBN 9781556175169.

[47] MEYER, Johannes, Alexander SENDOBRY, Stefan KOHLBRECHER, Uwe KLINGAUF a Oskar VON STRYK, 2012. Comprehensive Simulation of Quadrotor UAVs Using ROS and Gazebo [online]. 2012. B.m.: Springer Berlin Heidelberg. Dostupné z: doi:10.1007/978-3-642-34327-8_36

[48] Salih, Atheer & Moghavvemi, Mahmoud & Mohamed, Haider & Gaeid, Khalaf. (2010). Flight PID controller design for a UAV quadrotor.. Scientific Research and Essays. 5. 3660-3667.

[49] CARVALHO, João Pedro, Marco Aurélio JUCÁ, Alexandre MENEZES, Leonardo Rocha OLIVI, André Luis Marques MARCATO a Alexandre Bessa DOS SANTOS, 2016. Autonomous UAV Outdoor Flight Controlled by an Embedded System Using Odroid and ROS [online]. 4. září 2016. B.m.: Springer International Publishing. Dostupné z: doi:10.1007/978-3-319-43671-5_36

[50] PX4 uorb explained. In: PX4 [online]. Zurych, Switzerland: Lorenz Meier, 2014 [cit. 2023-05-06]. Dostupné z: https://px4.io/px4-uorb-explained-part-1/

[51] MAVLink. In: GitBooks.io [online]. Miami, Florida, United States of America: GitBooks, 2013 [cit. 2023-05-06]. Dostupné z: https://bresch.gitbooks.io/devguide/content/en/middleware/mavlink.html

[52] UORBmessages. In: Uav-lab [online]. Massachusets, United States of America: Zihao, 2015 [cit. 2023-05-06]. Dostupné z: https://uav-lab.org/2016/08/02/px4-research-log4-a-first-look-at-px4-architecture/#

[53] Publish - subscribe system. In: Gitbook.io [online]. Unknown: GitBook, 2021 [cit. 2023-05-06]. Dostupné z: https://nxp.gitbook.io/hovergames/developerguide/px4-tutorial-example-code/hg-px4-example-lab-1

[54] IRIS in Gazebo. In: GitBooks.io [online]. Miami, Florida, United States of America: GitBooks, 2013 [cit. 2023-05-06]. Dostupné z: https://dnddnjs.gitbooks.io/drone-autonomous-flight/content/64a9ac3701ac0326821862b638fb8bdf.png

[55] KIM, Yungjun, Seung-yong LEE a Sun LIM, 2020. Implementation of PLC controller connected Gazebo-ROS to support IEC 61131-3 [online]. září 2020. B.m.: IEEE. Dostupné z: doi:10.1109/etfa46521.2020.9212096

[56] TAKAYA, Kenta, Toshinori ASAI, Valeri KROUMOV a Florentin SMARANDACHE, 2016. Simulation environment for mobile robots testing using ROS and Gazebo [online]. říjen 2016. B.m.: IEEE. Dostupné z: doi:10.1109/icstcc.2016.7790647

[57] Mission planning in QGC. In: QGroundcontrol [online]. Zurych, Switzerland: Lorenz Meier, 2012 [cit. 2023-05-06]. Dostupné z: https://docs.qgroundcontrol.com/master/assets/plan/plan_view_overview.jpg

[58] PRASAD, Dandu Reddi Bhargava, Chinmai MADINENI, Inti Tharun KUMAR, Indira BASUMATARY, Kompella Ram Pranav KASYAP a Ashish MOHAN, 2022. Simulation of An Autonomous Website Controlled Payload Delivery UAV for Medical Emergency Response [online]. 26. srpen 2022. B.m.: IEEE. Dostupné z: doi:10.1109/iccubea54992.2022.10011084

[59] HOPF, J., J. DOMMASCHK, N. BLOCK, R. REINFELD, M. KRACHTEN, P. WORRMANN, D. CRACAU a A. KÖTHE, 2020. Unmanned Aircraft Experimental System: The Flying Lab for Applied Flight Control and Flight Mechanics. Deutsche Gesellschaft für Luft- und Raumfahrt - Lilienthal-Oberth e.V. [online]. Dostupné z: doi:10.25967/530237

[60] WHELAN, Jason, Abdulaziz ALMEHMADI a Khalil EL-KHATIB, 2022. Artificial intelligence for intrusion detection systems in Unmanned Aerial Vehicles [online]. duben 2022. B.m.: Elsevier BV. Dostupné z: doi:10.1016/j.compeleceng.2022.107784

[61] OGATA, Katsuhiko. Modern control engineering. 5th ed. Boston: Prentice Hall, c2010. ISBN 978-0-13-615673-4.

[62] PID Control theory. In: CrystalInstruments [online]. Los Angeles, United States of America: Crystal Instruments, 2003 [cit. 2023-05-06]. Dostupné z: https://www.crystalinstruments.com/blog/2020/8/23/pid-control-theory

[63] VURUSKAN, Aslihan, Burak YUKSEK, Ugur OZDEMIR, Adil YUKSELEN a Gokhan INALHAN, 2014. Dynamic modeling of a fixed-wing VTOL UAV [online]. květen 2014. B.m.: IEEE. Dostupné z: doi:10.1109/icuas.2014.6842289

[64] Standard VTOL. In: 404Warehouse [online]. Tempe, Arizona, United States of America: 404Warehouse, 2015 [cit. 2023-05-06]. Dostupné z: https://404warehouse.files.wordpress.com/2016/07/screenshot-from-2016-07-12-120102.png?w=700

[65] Airframe - standard vtol. In: PX4 [online]. Zurych, Switzerland: Lorenz Meier, 2014 [cit. 2023-05-06]. Dostupné z: https://docs.px4.io/v1.12/en/airframes/airframe_reference.html#standard-vtol

[66] BASILIO, J.C. a S.R. MATOS, 2002. Design of PI and PID controllers with transient performance specification [online]. listopad 2002. B.m.: Institute of Electrical and Electronics Engineers (IEEE). Dostupné z: doi:10.1109/te.2002.804399

[67] PX4 Autotune in QGC. In: QGroundcontrol [online]. Zurych, Switzerland: Lorenz Meier, 2012 [cit. 2023-05-06]. Dostupné z: https://docs.qgroundcontrol.com/master/assets/setup/tuning/px4_autotune_hero.png

[68] Anon., 2012. 2012 international conference on advances in engineering, science and management (ICAESM 2012): Nagapattinam, Tamil Nadu, India, 30 - 31 march 2012.

[69] PATEL, Vishakha Vijay, 2020. Ziegler-Nichols Tuning Method [online]. říjen 2020. B.m.: Springer Science and Business Media LLC. Dostupné z: doi:10.1007/s12045-020-1058-z

[70] Understanding quaternions. In: 3DGep [online]. Aukland district, New Zealand: Instra Corporation, 2011 [cit. 2023-05-06]. Dostupné z: https://www.3dgep.com/understanding-quaternions/

[71] Work with quaternions. In: Emis [online]. Eggenstein-Leopoldshafen, Germany: FIZ Karlsruhe Leibniz; Institute for Information Infrastructure, 1995 [cit. 2023-05-06]. Dostupné z: https://www.emis.de/proceedings/Varna/vol1/GEOM09.pdf

[72] MAVROS - Tutorials. In: ROS.org [online]. Unknown: eNom, 1998 [cit. 2023-05-06]. Dostupné z: http://wiki.ros.org/mavros/Tutorials

[73] FAIRCHILD, Carol a Thomas L. HARMAN, 2016. ROS Robotics By Example. Birmingham, England: Packt Publishing. ISBN 9781785286704.

[74] MavRos-takeoff-n-land -TEMPLATE. In: *GITHUB* [online]. California, United States of America: GitHub, 2007 [cit. 2023-05-06]. Dostupné z: https://github.com/UCM-M143/MavRos-takeoff-n-land

[75] FERNANDEZ, Enrique, Luis SANCHEZ CRESPO a Anil MAHTANI, 2015. Learning ROS for robotics programming -. 2nd ed. Birmingham, England: Packt Publishing. ISBN 9781783987597.

[76] ASTROM, Karl Johan a Richard M. MURRAY, 2008. Feedback systems: An introduction for scientists and engineers. Princeton, NJ: Princeton University Press. ISBN 9780691135762.

[77] Bezpilotní prostředek s pevnou konstrukcí rotorů. In: *DroneFromChina* [online]. Beijing, China: DroneFromChina, 2018 [cit. 2023-05-15]. Dostupné z: http://www.dronefromchina.com/Uploads/5df998eba146c.jpg

[78] TAKALA, Tuukka M., Meeri MAKARAINEN a Perttu HAMALAINEN, 2013. Immersive 3D modeling with Blender and off-the-shelf hardware [online]. březen 2013. B.m.: IEEE. Dostupné z: doi:10.1109/3dui.2013.6550243

Seznam příloh

Příloha 1: Skript řídicího algoritmu

```
//Zahrnutí knihoven

#include <ros/ros.h>
#include <geometry_msgs/PoseStamped.h>
#include <mavros_msgs/CommandBool.h>
#include <mavros_msgs/CommandTOL.h>
#include <mavros_msgs/SetMode.h>
#include <mavros_msgs/State.h>
#include <geometry_msgs/TwistStamped.h>
#include <mavros_msgs/PositionTarget.h>
#include <tf2/LinearMath/Quaternion.h>
#include <tf2_geometry_msgs/tf2_geometry_msgs.h>
#include <sensor_msgs/Range.h>
#include <cmath>

//Definice konstanty pro výšku
#define FLIGHT_ALTITUDE 10.0f

//Definice proměnný a nastavení PID kontroléru pro řízení výšky, klonění a
klopení
double roll, pitch, yaw;
double Kp = 2.55, Ki = 0.0, Kd = 0.0;
double setpoint_altitude = FLIGHT_ALTITUDE;
double current_altitude = 0.0;
double error_sum = 0.0;
double prev_error = 0.0;
double max_altitude_integral_error = 100;
ros::Time prev_time;

double Kp_pitch = 1.2, Ki_pitch = 0.0, Kd_pitch = 0.0;
double setpoint_pitch = 0.0;
double error_sum_pitch = 0.0;
double prev_error_pitch = 0.0;
ros::Time prev_time_pitch;
double max_pitch_integral_error = 100;

double Kp_roll = 2.95, Ki_roll = 0.0, Kd_roll = 0.0;
double setpoint_roll = 0.0;
double error_sum_roll = 0.0;
double prev_error_roll = 0.0;
ros::Time prev_time_roll;
double max_roll_integral_error = 100;

double prev_altitude = 0.0;

//Inicializace PID kontrolérů
class PIDController {

public:
      PIDController(double Kp, double Ki, double Kd, double
max_integral_error)
            : Kp(Kp), Ki(Ki), Kd(Kd),
            max_integral_error(max_integral_error), prev_error(0.0),
            prev_error_derivative(0.0) {}

            double update(double error, double dt) {
```

```
            error_sum += error * dt;
            error_sum = std::min(std::max(error_sum, -max_integral_error),
            max_integral_error);
            double error_derivative = (error - prev_error) / dt;

            // Aplikace spodního filtru, která ovlivňuje maximální
            kumulovanou chybu derivační složky
            double alpha = 0.1;
            double error_derivative_filtered = (1.0 - alpha) *
            prev_error_derivative + alpha * error_derivative;

            double output = Kp * error + Ki * error_sum + Kd *
            error_derivative_filtered;

            prev_error = error;
            prev_error_derivative = error_derivative_filtered;
            return output;
      }
      void setKp(double Kp);
      void setKi(double Ki);
      void setKd(double Kd);

private:
      double Kp, Ki, Kd;
      double error_sum = 0.0;

      double max_integral_error;
      double prev_error;
      double prev_error_derivative;

};

//inicializace PID kontrolérů - vytvoření instancí
PIDController altitudeController(Kp, Ki, Kd, max_altitude_integral_error);
PIDController pitchController(Kp_pitch, Ki_pitch, Kd_pitch,
max_pitch_integral_error);
PIDController rollController(Kp_roll, Ki_roll, Kd_roll,
max_roll_integral_error);

//Callbacky pro stav dronu a polohu
mavros_msgs::State current_state;
void state_cb(const mavros_msgs::State::ConstPtr& msg)
{
      current_state = *msg;
}

geometry_msgs::PoseStamped current_posed;
void pose_cb(const geometry_msgs::PoseStamped::ConstPtr& msg)
{
      current_posed = *msg;

      tf2::Quaternion quat;
      tf2::convert(msg->pose.orientation, quat);
      tf2::Matrix3x3(quat).getRPY(roll, pitch, yaw);
}

//Callback pro výšku
void altitudeCallback(const sensor_msgs::Range::ConstPtr& msg)
{
      current_altitude = msg->range;
```

```
	ROS_INFO("Current Altitude: %f", current_altitude);
}

//Funkce distance - výpočet vzdálenost mezi dvěma body
double distance(double x1, double y1, double x2, double y2) {
	return sqrt(pow(x2 - x1, 2) + pow(y2 - y1, 2));

}

//Hlavní funkce - vstupní bod programu
int main(int argc, char **argv)
{
	//Inicializace a příprava ROS, inicializace publisherů a odběratelů,
	nastavení klientů arm, land, set_mode
	ros::init(argc, argv, "offb_node");
	ros::NodeHandle nh;
	ros::Subscriber altitude_sub = nh.subscribe("/range_sensor", 1,
	altitudeCallback);
	ros::Subscriber state_sub =
	nh.subscribe<mavros_msgs::State>("mavros/state", 10, state_cb);
	ros::Publisher local_pos_pub =
	nh.advertise<geometry_msgs::PoseStamped>("mavros/setpoint_position/lo
	cal", 10);
	ros::ServiceClient arming_client =
	nh.serviceClient<mavros_msgs::CommandBool>("mavros/cmd/arming");
	ros::ServiceClient land_client =
	nh.serviceClient<mavros_msgs::CommandTOL>("mavros/cmd/land");
	ros::ServiceClient set_mode_client =
	nh.serviceClient<mavros_msgs::SetMode>("mavros/set_mode");
	ros::Subscriber pose_sub =
	nh.subscribe<geometry_msgs::PoseStamped>("mavros/local_position/pose"
	, 10, pose_cb);
	ros::Publisher local_vel_pub =
	nh.advertise<geometry_msgs::TwistStamped>("mavros/setpoint_velocity/c
	md_vel", 10);

	//Inicializace proměnných a objektů
	double pitch_error_sum = 0.0;
	double max_velocity = 5.0;

	geometry_msgs::TwistStamped velocity;

	ros::Rate rate(50.0);

	tf2::Quaternion quat;

	//Smyčka pro připojení bezpilotního prostředku
	while (ros::ok() && current_state.connected)
	{
		ros::spinOnce();
		rate.sleep();
		ROS_INFO("connecting to FCT...");
	}

	//Inicializace definované polohy
	geometry_msgs::PoseStamped pose;
	pose.pose.position.x = 0;
	pose.pose.position.y = 0;
	pose.pose.position.z = FLIGHT_ALTITUDE;
```

```
for (int i = 100; ros::ok() && i > 0; --i)
{
local_pos_pub.publish(pose);
ros::spinOnce();
rate.sleep();
}

//Inicializace zpráv pro arm, land, set_mode
mavros_msgs::SetMode offb_set_mode;
offb_set_mode.request.custom_mode = "OFFBOARD";

mavros_msgs::CommandBool arm_cmd;
arm_cmd.request.value = true;

mavros_msgs::CommandTOL land_cmd;
land_cmd.request.yaw = 0;
land_cmd.request.latitude = 0;
land_cmd.request.longitude = 0;
land_cmd.request.altitude = 0;

ros::Time last_request = ros::Time::now();
//Smyčka řízení arm a offboard
while (ros::ok() && !current_state.armed)
      {
      if (current_state.mode != "OFFBOARD" && (ros::Time::now() -
      last_request > ros::Duration(5.0)))
      {
      ROS_INFO(current_state.mode.c_str());
      if (set_mode_client.call(offb_set_mode) &&
      offb_set_mode.response.mode_sent)
      {
      ROS_INFO("Offboard enabled");
      }
last_request = ros::Time::now();
}
else
      {
      (!current_state.armed && (ros::Time::now() - last_request >
      ros::Duration(5.0)))
            {
            if (arming_client.call(arm_cmd) &&
            arm_cmd.response.success)
                  {
                  ROS_INFO("Vehicle armed");
                  }
            last_request = ros::Time::now();
            }
      }
local_pos_pub.publish(pose);
ros::spinOnce();
rate.sleep();
}

//Získání informací o současné poloze
double x = current_posed.pose.position.x;
double y = current_posed.pose.position.y;
double z = current_posed.pose.position.z;

//Nastavení PID kontroléru výšky skrz metodu ZN
double Ku_alt = 2.0211e-5;
```

```
double Tu_alt = 4.1785;
Kp = 0.6 * Ku_alt;
Ki = 2 * Kp / Tu_alt;
Kd = Kp * Tu_alt / 8;

//Nastavení PID kontroléru klonění(pitch) skrz metodu ZN
double Ku_pitch = 6.1157e-8;
double Tu_pitch = 4.0582;
Kp_pitch = 0.6 * Ku_pitch;
Ki_pitch = 2 * Kp_pitch / Tu_pitch;
Kd_pitch = Kp_pitch * Tu_pitch / 8;

//Nastavení PID kontroléru klonění(roll) skrz metodu ZN
double Ku_roll = 3.6944e-7;
double Tu_roll = 4.2638;
Kp_roll = 0.6 * Ku_roll;
Ki_roll = 2 * Kp_roll / Tu_roll;
Kd_roll = Kp_roll * Tu_roll / 8;

//Začátek vzletu
ROS_INFO("Takeoff");

//Smyčka vzletu
while (ros::ok() && current_posed.pose.position.z <= (FLIGHT_ALTITUDE
- 0.05)) {

      //Ukončení výšky v případě procesované chyby
            if(current_posed.pose.position.z >= 70.0) {
                  return 0;
            }

      //Výpočet chyb a PID výstupů
      double error = setpoint_altitude -
      current_posed.pose.position.z;
      double dt = (ros::Time::now() - prev_time).toSec();
      double output = altitudeController.update(error, dt);

      double error_pitch = setpoint_pitch - pitch;
      double dt_pitch = (ros::Time::now() - prev_time_pitch).toSec();
      double output_pitch = pitchController.update(error_pitch,
      dt_pitch);

      double error_roll = setpoint_roll - roll;
      ros::Time current_time_roll = ros::Time::now();
      double dt_roll = (current_time_roll - prev_time_roll).toSec();
      double output_roll = rollController.update(error_roll,
      dt_roll);

      //Aktualizace orientace dronu
      tf2::Quaternion quat;
      quat.setRPY(output_roll, output_pitch, 0);
      quat.normalize();
      pose.pose.orientation = tf2::toMsg(quat);

      //Varianta s rychlosti - Aktualizace rychlosti
      velocity.twist.linear.z =  output;
      local_vel_pub.publish(velocity);

      //varianta s pozici
```

```
        /*
        current_posed.pose.position.z = current_posed.pose.position.z +
        output;
        local_pos_pub.publish(pose);
        */

        //Aktualizace předchozích hodnot chyb a času
        prev_error = error;
        prev_time = ros::Time::now();
        prev_error_pitch = error_pitch;
        prev_time_pitch = ros::Time::now();
        prev_error_roll = error_roll;
        prev_time_roll = current_time_roll;

        prev_altitude = current_posed.pose.position.z;

        ros::spinOnce();
        rate.sleep();

        //Výpis informací o řízení bezpilotního prostředku
        ROS_INFO("Kp: %.2f, Error: %.2f, Output: %.2f", Kp_roll,
        error_roll, output_roll);
        ROS_INFO("Altitude: Current: %.2f, Setpoint: %.2f, Error: %.2f,
        Output: %.2f, dt: %.2f | "
        "Pitch: Current: %.2f, Setpoint: %.2f, Error: %.2f, Output:
        %.2f, dt: %.2f | "
        "Pitch: Current: %.2f, Setpoint: %.2f, Error: %.2f, Output:
        %.2f, dt: %.2f | "
        "Roll: Current: %.2f, Setpoint: %.2f, Error: %.2f, Output:
        %.2f, dt: %.2f",
        current_posed.pose.position.z, setpoint_altitude, error,
        output, dt,
        pitch, setpoint_pitch, error_pitch, output_pitch, dt_pitch,
        roll, setpoint_roll, error_roll, output_roll, dt_roll);
        ROS_INFO("VELOCITY: %.8f", velocity.twist.linear.z);

        }
    //Konec vzletové fáze
    ROS_INFO("Takeoff done");


    //Nastavení waypointu a dalších parametrů
    double waypoint_x = 50;
    double waypoint_y = 0;
    pose.pose.position.x = waypoint_x;
    pose.pose.position.y = waypoint_y;
    setpoint_altitude = FLIGHT_ALTITUDE;
    setpoint_pitch = 0.05;
    setpoint_roll = 0.05;

    //Začátek horizontálního letu
    ROS_INFO("Moving to waypoint");
        //Smyčka horizontálního letu
        while (ros::ok() && fabs(current_posed.pose.position.x -
        waypoint_x) > 0.05)
            {
            //Ukončení výšky v případě procesované chyby
            if(current_posed.pose.position.z >= 70.0)
                {return 0; }
            else{
```

```
//Výpočet chyb a PID výstupů
double error = setpoint_altitude -
current_posed.pose.position.z;
double dt = (ros::Time::now() - prev_time).toSec();
double output = altitudeController.update(error, dt);

double error_pitch = setpoint_pitch - pitch;
double dt_pitch = (ros::Time::now() -
prev_time_pitch).toSec();
double output_pitch = pitchController.update(error_pitch,
dt_pitch);

double error_roll = setpoint_roll - roll;
ros::Time current_time_roll = ros::Time::now();
double dt_roll = (current_time_roll -
prev_time_roll).toSec();
double output_roll = rollController.update(error_roll,
dt_roll);

//Aktualizace orientace dronu
tf2::Quaternion quat;
quat.setRPY(output_roll, output_pitch, 0);
quat.normalize();
pose.pose.orientation = tf2::toMsg(quat);

local_pos_pub.publish(pose);

//Aktualizace předchozích hodnot chyb a času
prev_error = error;
prev_time = ros::Time::now();
prev_error_pitch = error_pitch;
prev_time_pitch = ros::Time::now();
prev_error_roll = error_roll;
prev_time_roll = current_time_roll;

//Výpočet rychlosti vůči bodu dosažení
double dist_to_waypoint =
distance(current_posed.pose.position.x,
current_posed.pose.position.y, waypoint_x, waypoint_y);
double vel_command = std::min(max_velocity, Kp *
dist_to_waypoint);

//Aktualizace rychlosti
geometry_msgs::TwistStamped vel_cmd;
vel_cmd.twist.linear.x = vel_command;
vel_cmd.twist.linear.y = 0.0;
vel_cmd.twist.linear.z = 0.0;
vel_cmd.twist.angular.x = 0.0;
vel_cmd.twist.angular.y = 0.0;
vel_cmd.twist.angular.z = 0.0;
local_vel_pub.publish(vel_cmd);

//Výpis informací o řízení bezpilotního prostředku
ROS_INFO("Position: Current: (X: %.2f, Y: %.2f, Z: %.2f),
Waypoint: (X: %.2f, Y: %.2f), "
"Altitude: Setpoint: %.2f, Current: %.2f, Error: %.2f,
Output: %.5f, "
"Pitch: Setpoint: %.5f, Current: %.5f, Error: %.5f,
Output: %.5f, "
```

```
                "Roll: Setpoint: %.5f, Current: %.5f, Error: %.5f,
                Output: %.5f, "
                "Velocity Command: %.5f, Velocity: (X: %.2f, Y: %.2f, Z:
                %.2f, %.2f,%.2f,%.2f)",
                current_posed.pose.position.x,
                current_posed.pose.position.y,
                current_posed.pose.position.z,
                waypoint_x, waypoint_y,
                setpoint_altitude, current_posed.pose.position.z, error,
                output,
                setpoint_pitch, pitch, error_pitch, output_pitch,
                setpoint_roll, roll, error_roll, output_roll,
                vel_command,
                vel_cmd.twist.linear.x, vel_cmd.twist.linear.y,
                vel_cmd.twist.linear.z,Kp,Ki,Kd);
                ROS_INFO("VELOCITY in z: %.8f, VELICOTY in x: %.8f",
                vel_cmd.twist.linear.z, vel_cmd.twist.linear.x);
                ROS_INFO("Kp: %.2f, Error: %.2f, Output: %.2f", Kp_roll,
                error_roll, output_roll);

                ros::spinOnce();
                rate.sleep();
                }
        }
//Konec fáze horizontálního letu
ROS_INFO("Waypoint reached");

//Nastavení výšky, roll a pitch
setpoint_altitude = 1.0;
setpoint_pitch = 0.0;
setpoint_roll = 0.0;

//Inicializace předchozí času
ros::Time prev_time = ros::Time::now();

//Začátek fáze klesání do 1 metru
ROS_INFO("Descending to 1 meter");

//Smyčka klesání do 1 metru
while (ros::ok() && (current_posed.pose.position.z -
setpoint_altitude) > 0.05)
        {
//Ukončení výšky v případě procesované chyby
                if(current_posed.pose.position.z >= 70.0)
                        {return 0; }
                else{

                //Výpočet chyb a PID výstupů
                double error = setpoint_altitude -
                current_posed.pose.position.z;
                double dt = (ros::Time::now() - prev_time).toSec();
                double output = altitudeController.update(error, dt);

                double error_pitch = setpoint_pitch - pitch;
                double dt_pitch = (ros::Time::now() -
                prev_time_pitch).toSec();
                double output_pitch = pitchController.update(error_pitch,
                dt_pitch);

                double error_roll = setpoint_roll - roll;
```

```
                ros::Time current_time_roll = ros::Time::now();
                double dt_roll = (current_time_roll -
                prev_time_roll).toSec();
                double output_roll = rollController.update(error_roll,
                dt_roll);

                //Aktualizace orientace dronu
                tf2::Quaternion quat;
                quat.setRPY(output_roll, output_pitch, 0);
                quat.normalize();
                pose.pose.orientation = tf2::toMsg(quat);

                //Varianta s rychlosti - Aktualizace rychlosti

                velocity.twist.linear.z = output;
                local_vel_pub.publish(velocity);


                //Varianta s polohou
                /*
                pose.pose.position.z = current_posed.pose.position.z +
                output;
                local_pos_pub.publish(pose);
                */


                //Aktualizace předchozích hodnot chyb a času
                prev_error = error;
                prev_time = ros::Time::now();
                prev_error_pitch = error_pitch;
                prev_time_pitch = ros::Time::now();
                prev_error_roll = error_roll;
                prev_time_roll = current_time_roll;

                ros::spinOnce();
                rate.sleep();

                //Výpis informací o řízení bezpilotního prostředku
                ROS_INFO("Altitude: Current: %.2f, Setpoint: %.2f, Error:
                %.8f, Output: %.2f, dt: %.2f, "
                "Pitch: Setpoint: %.5f, Current: %.5f, Error: %.5f,
                Output: %.5f, "
                "Roll: Setpoint: %.5f, Current: %.5f, Error: %.5f,
                Output: %.5f, ",
                current_posed.pose.position.z, setpoint_altitude, error,
                output, dt,
                setpoint_pitch, pitch, error_pitch, output_pitch,
                setpoint_roll, roll, error_roll, output_roll);
                ROS_INFO("VELOCITY: %.8f", velocity.twist.linear.z);*/
                ROS_INFO("Kp: %.2f, Error: %.2f, Output: %.2f", Kp_roll,
                error_roll, output_roll);
                }
        }

    //Konec fáze klesání do 1 metru
    ROS_INFO("Reached 1 meter");

    //Začátek fáze přistání
    ROS_INFO("Landing");
    //Smyčka přistání
```

```
	while (!(land_client.call(land_cmd) && land_cmd.response.success))
		{
		local_pos_pub.publish(pose);
		ros::spinOnce();
		rate.sleep();
		}

	//Konec fáze přistání
	ROS_INFO("Landed");

	//Ukončení funkce
	return 0;
}
```