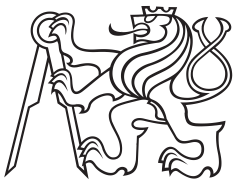
Diplomová práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra kybernetiky

Udržování formace UAV na základě měření vzájemných vzdáleností

Bc. Jan Gärtner

Vedoucí: Ing. Jan Chudoba
Obor: Kybernetika a robotika
Studijní program: Kybernetika a robotika
Leden 2021

Poděkování

V první řadě děkuji vedoucímu práce panu Ing. Janu Chudobovi za poskytnuté konzultace a rady. Dále děkuji rodině a také své Koze za to, že mi umožnili izolaci od okolního světa ve chvílích, kdy to práce vyžadovala.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze 21. května 2021

podpis autora práce

Abstrakt

Cílem diplomové práce bylo nalézt a implementovat metodu vhodnou pro udržování formace laboratorních modelů helikoptér. K řízení jsou použity údaje o vzájemné vzdáleností účastníků.

Byl upraven existující algoritmus sloužící původně k řízení trojúhelníkové formace námořních plavidel. Dále byl implementován a popsán lokalizační algoritmus založený na EKF (*Extended Kalman Filter*). Měření vzdálenosti jsou prováděna senzorem vlastního návrhu postaveným na modulu DecaWave DWM1000.

Jednotlivé kapitoly se zabývají: metodami řízení formací a lokalizace mobilních agentů, moduly DWM1000 a drony Ryze Tello, vývojem firmwaru senzoru vzdálenosti a knihovnou implementující diskutované algoritmy. Poslední kapitola obsahuje výsledky provedených experimentů.

Experimenty prokázaly schopnost zvolené metody udržet pohyblivou formaci s helikoptérami Ryze Tello. Ověřena byla také možnost lokalizace mobilních agentů s přesností až 10 cm.

Klíčová slova: dron, UAV, udržování formace, měření vzdálenosti, DWM1000

Vedoucí: Ing. Jan Chudoba
Inteligentní a mobilní robotika CIIRC,
Jugoslávských partyzánů 1580/3
Praha 6

Abstract

The aim of the thesis was to find and implement a method suitable for maintaining the formation of laboratory models of helicopters. Mutual distance of the participants is used for the control.

An existing algorithm originally used to control the triangular formation of marine vehicles was modified. Furthermore, a localization algorithm based on EKF (*Extended Kalman Filter*) has been implemented and described. Distance measurements are performed by a sensor of our own design based on the DecaWave DWM1000 module.

Individual chapters cover: methods for formation control and localization of mobile agents, DWM1000 modules and Ryze Tello drones, development of the distance sensor firmware, and a library implementing the algorithms discussed. The last chapter contains the results of the experiments performed.

The experiments demonstrated the ability of the chosen method to maintain a moving formation with Ryze Tello helicopters. The ability to locate mobile agents with an accuracy of up to 10 cm was also verified.

Keywords: drone, UAV, formation control, distance measurement, DWM1000

Title translation: UAV formation keeping based on mutual distance measurement

Obsah

Obrázky

Tabulky

Úvod

Výzkum v oblasti udržování formací bezpilotních letounů se dříve soustředil převážně na aplikace ve vojenství, jako ostatně veškerý vývoj bezpilotní techniky. V posledních letech však nabývá na významu i v civilní sféře a do budoucna lze očekávat jeho další rozvoj. Hlavními důvody narůstajícího zájmu o podobné technologie jsou klesající cena potřebného hardwaru a narůstající kvalita a dostupnost senzorů.

Flotily a hejna autonomních helikoptér nabízí široké možnosti při průzkumu prostředí ve venkovních i vnitřních prostorách. Koordinovaný pohyb více sdružených agentů umožňuje navýšit hmotnost dopravovaného nákladu a množství osazených senzorů. Tím lze dosáhnout lepšího pokrytí při průzkumu, vyššího rozlišení při mapování či měření. Koordinovaným pohybem se zároveň snižují náklady na výbavu jednotlivých agentů. Pouze malá část strojů musí být vybavena přesnou lokalizační a navigační technikou, ostatní účastníci mise je dokáží následovat s minimálními požadavky na osazený hardware.

Cílem této práce je poskytnout přehled existujících metod udržování formací a lokalizace UAV (*Unmanned Aerial Vehicle*) neboli bezpilotních letadel či dronů. Po vyhodnocení všech možností navrhneme metodu, která bude využívat měření vzájemných vzdáleností. Vzdálenosti plánujeme získávat pomocí UWB modulů DecaWave DWM100 metodou TOF *(time-of-flight)*. V rámci práce bude potřeba vystavět infrastrukturu pro měření vzdálenosti, moduly DWM1000 touto funkcionalitou přímo nedisponují. Zároveň ověříme jejich použitelnost pro zamýšlený účel.

Nabídneme na problém dva odlišné pohledy. První přístup funguje čistě v lokálním souřadném systému každého agenta. S jeho pomocí lze udržovat pohyblivou formaci tvaru trojúhelníku s minimálními nároky na vzájemnou interakci. Metoda vychází z existujícího návrhu pro řízení námořních plavidel a využívá principu sledování vůdce.

Druhý přístup pracuje s agenty definované polohy, tzv. kotvami. Je implementačně náročnější, má však potenciál širšího uplatnění přesahujícího problematiku formací. K udržování pohyblivé formace může sloužit, budeme-li mít několik letounů s pokročilejší senzorovou výbavou. Ty definují souřadný systém a ostatní agenti se vůči nim lokalizují za pomoci EKF (*Extended Kalman Filter*). Algoritmus však může sloužit k obecné lokalizaci libovolných objektů.

Kapitola 1

Metody řízení formací

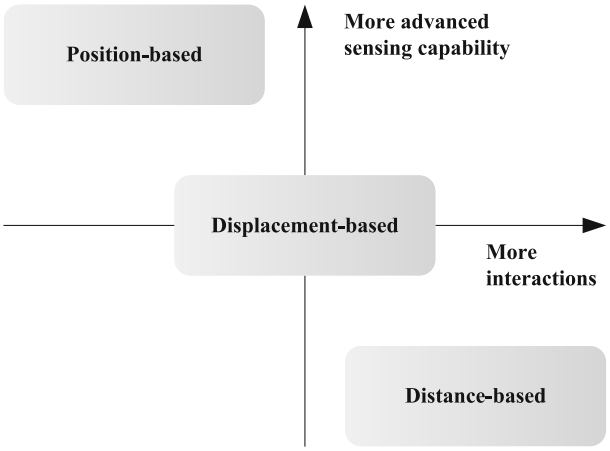
Při studiu odborné literatury jsem se zabýval existujícími metodami pro řízení formací multiagentních systémů s cílem najít metodu přímo aplikovatelnou nebo upravitelnou pro naše potřeby. Přehledně zpracovanou souhrnnou studií o kontrolování formací je např. [OPA15]. Kategorizuje nepřeberné množství existujících přístupů do tří základních skupin:

- *Position-based control* (řízení na základě polohy): agenti jsou schopni měřit svoji polohu v globálním souřadném systému. V něm je také definován tvar formace a agenti svoji polohu aktivně kontrolují.
- *Displacement-based control* (řízení na základě vzájemného posunu): agenti snímají relativní polohu sousedů a na základě toho řídí svůj pohyb. Tvar formace je definován vzájemnou polohou účastníků v globálním souřadném systému. Z toho plyne, že agenti musí mít kromě relativních měření také informaci o orientaci (natočení) globálního souřadného systému. K němu je žádaná formace vztažena.
- *Distance-based control* (řízení na základě vzájemné vzdálenosti): Aktivně regulovanou veličinou jsou vzdálenosti měřené přímo mezi jednotlivými agenty. Dané vzdálenosti definují požadovanou formaci. Agenti jsou schopni měřit relativní polohu svých sousedů vzhledem k vlastnímu souřadnému systému. Souřadné systémy jednotlivých agentů přitom nemusejí být shodně orientovány.

Obr. 1.1 porovnává uvedené přístupy z pohledu potřebné senzorové výbavy a počtu interakcí nutných k procesu řízení. Zatímco *position-based* lokalizace vyžaduje pokročilé senzorické vybavení, proces řízení už je stran počtu akcí „jednodušší“. Na opačné straně leží lokalizace *distance-based* metodami, kde je postačující znalost vzdálenosti a směru několika sousedních agentů, bez přímého vztahu ke globálním souřadnicím.

Považuji za důležité pozastavit se nad uvedeným způsobem dělení do kategorií. *Distance-based* metody mohou vyvolat dojem, že k jejich použití stačí měřit vzájemnou vzdálenost mezi agenty a jsou tak pro nás ideální volbou. Podíváme-li se na definici pozorně, zjistíme, že „agenti jsou schopni měřit relativní polohu svých sousedů vzhledem k vlastnímu souřadnému systému“. Při studiu existujících řešení jsme často narazili na skutečnost, že znalost vzájemných vzdáleností není pro implementaci dostačující. Můžou tedy být vyžadovány vyšší nároky na použité senzory, než by se z prvního přiblížení mohlo jevit, nebo složitější algoritmy, které směr odhadují jinými metodami.

Souhrn [OPA15] námi definovaný problém označuje jako *pure distance-based control* (řízení čistě na základě dat o vzdáleností) a stručně zmiňuje několik studií zabývajících se tímto tématem.



Obrázek 1.1: Kategorizace metod řízení formací multiagentních systémů, převzato z [OPA15]

1.1 Princip *distance-based* metod

Podíváme se nyní podrobněji na metody typu *distance-based control*, jelikož jsou blízké našemu směřování. Popis následujících obecných principů čerpám z [OPA15].

Při použití *distance-based* metod je žádaná formace definována množinou vzájemných vzdáleností mezi agenty. Bez ohledu na to, zda k regulaci stačí znalost těchto vzdáleností, nebo potřebujeme informaci o azimutu, formace musí splňovat jisté předpoklady. Pro jejich vyhodnocení lze využít poznatků z teorie grafů. Agenty (účastníky formace) reprezentují vrcholy grafu. Hrana mezi dvěma vrcholy značí dostupnost informace o vzájemné vzdálenosti daných vrcholů.

Aby množina vzájemných vzdáleností definovala formaci jednoznačně, musí být k dispozici dostatečné množství měřených vzdáleností, tedy graf musí obsahovat dostatečně mnoho hran. Je možné pracovat s úplným grafem (každé dva vrcholy jsou spojené hranou). Tím ale zbytečně navýšíme množství interakcí mezi agenty a zpomalíme proces řízení. Jednou z postačujících podmínek je, aby byl graf *rigidní*. Proto si neformálně zavedeme pojem *rigidita* neboli tuhost grafu. Formální definici pojmu uvádí [OPA15].

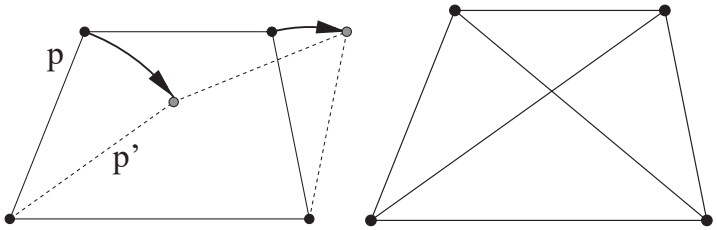
Představme si, že hrany grafu (rovinného, neorientovaného, dvoudimenzionálního) jsou realizovány tuhými tyčemi, které se mohou ve vrcholech volně otáčet (měnit vzájemný úhel). Graf označujeme jako *rigidní*, pokud jej není možno deformovat, tedy otáčet klouby při zachování délek hran. Princip je demonstrován na obr. 1.2. [OPA15] navíc rozlišuje lokální a globální rigiditu, námi formulovaný popis odpovídá lokální variantě, která je vyhovující, pokud jsou počáteční podmínky blízké žádané formaci.

Nyní si stručně představíme základní princip metod pro model jednoduchého integrátoru, plné znění opět v [OPA15]. Uvažujeme

$$\dot{p}_i = u_i,\ i = 1 \dots N, \tag{1.1}$$

kde $\dot{p}_i$ je změna polohy agenta i, N počet agentů a u_i akční zásah. Pohybujeme se ve dvojrozměrném prostoru (s formací definovanou v rovině). Zavedeme potenciálovou funkci

$$\Phi_i = \frac{k_p}{2} \sum_{j \in N_i} \gamma_{ij}(||p_j^i - p_i^i||), \tag{1.2}$$



(a) : Není rigidní **(b) :** Rigidní

Obrázek 1.2: Příklad rigidity grafu, upraveno z [HADB07]

kde $k_p > 0$ je proporcionální koeficient, p_j^i je poloha agenta j v souřadném systému agenta i. Funkce $\gamma_{ij} : \Re \to \bar{\Re}_+$ se vstupním argumentem $||p_j^i - p_i^i||$ je diferencovatelná a obsahuje informaci o požadované formaci. Akční zásah poté můžeme získat gradientní metodou jako

$$u_i^i = -\Delta_{p_i^i}\Phi_i = K_p \sum_{j \in N_i} \frac{\partial \gamma_{ij}(||p_j^i - p_i^i||)}{\partial ||p_j^i - p_i^i||} \frac{p_j^i - p_i^i}{||p_j^i - p_i^i||}, \tag{1.3}$$

kde u_i^i je akční zásah každého agenta v jeho vlastním souřadném systému. Podařilo se nám formulovat rovnice pro akční zásah jednotlivých agentů v jejich vlastních souřadných systémech. Dokážeme tedy zajistit udržení zadaných vzdáleností mezi agenty a v případě rigidního grafu to vede zároveň k udržení požadované formace. Důležitou součástí celého návrhu je samozřejmě funkce $\gamma_{ij}(||p_j^i - p_i^i||)$. Ta ovlivňuje, kde potenciálová funkce Φ_i nabývá minim.

V literatuře se vyskytuje mnoho variant, většina však vychází z podoby

$$\gamma_{ij} = k_p \left(||p_j - p_i||^2 - ||p_j^* - p_i^*||^2 \right)^2. \tag{1.4}$$

Koeficient $k_p > 0$ a p_i^* značí požadovanou polohu agenta i v rámci formace. Jelikož jsou u funkce γ_{ij} zkoumány vlastnosti jako globální stabilita, je zvykem pracovat v tomto případě se souřadnicemi v globálním systému. Za jistých podmínek je dokázána lokální asymptotická stabilita pro řízení gradientní metodou s použitím funkce (1.4). Problematikou stability v tomto kontextu se hlouběji zabývá např. [KBF09]. Existují i další tvary funkce γ_{ij}, které poskytují odlišné podmínky stability nebo např. zabraňují kolizím (přílišnému přiblížení agentů). Podrobnější přehled nabízí [OPA15].

1.2 Existující systémy pro udržování formací

Článek *leader-follower type distance-based formation control of a group of autonomous agents* [OA17] prezentuje metodu udržování formace pomocí principů popsaných v kapitole 1.1. Formace je v tomto případě reprezentována orientovaným acyklickým grafem, autoři tedy berou v potaz jednostranné měření vzdálenosti, kdy pouze jeden z účastníků má k dispozici výsledek operace. Článek obsahuje výsledky simulace dokládající funkčnost metody, použity jsou 3 vedoucí agenti a 7 sledujících v trojúhelníkové formaci. K aplikaci je však potřeba snímat relativní polohy sousedních agentů.

Autoři práce *Distance-based Formation Control with a Single Moving Leader* [KPLA14] popisují další metodu založenou na systému *leader-follower*, ve formaci stačí zvolit jednoho vůdčího agenta. Ten se pohybuje neznámou rychlostí, ostatní agenti odhadují jeho rychlost

Obrázek 1.3: Formace ve tvaru obráceného písmene V, převzato z [KPLA14]

a následují ho. Formace má tvar písmene V směřujícího špičkou vpřed, čímž napodobuje hejna ptáků či uskupení vojenských stíhacích letounů (viz obr. 1.3). Taktéž je potřeba měřit relativní polohu sousedních agentů.

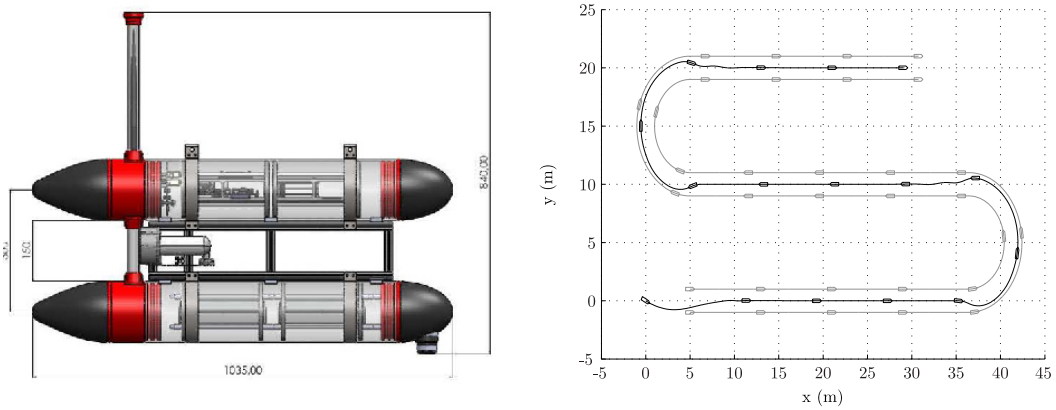
Formation control using range-only measurements [CYA11] zpracovává algoritmus pro udržování formace čistě na základě měření vzdáleností. Pohyb každého agenta je rozdělen na tři fáze:

1. odhad pozice ostatních agentů,
2. řízení vlastní polohy s cílem minimalizovat hodnotu Lyapunovy funkce,
3. setrvání v klidu, čímž umožní ostatním agentům vykonat kroky 1) a 2).

Odhad vzájemné polohy vychází ze změny vzdálenosti, tedy vzájemných rychlostí. Metodu je možné zobecnit i pro pohybující se formace, ovšem celý proces předpokládá nechybová vstupní data. Vzdálenost je snímána v diskrétních, přesně definovaných momentech a údaj není zatížen šumem. Takovým podmínkám se s reálnými drony a použitými senzory nedokážeme přiblížit. Zajištění spolehlivého přechodu mezi fázemi aktivního řízení a vyčkávání „v klidu“ je též realizovatelné spíše s pozemními roboty.

Range-only sensing for formation shape control and easy sensor network localization [AY11] přichází s řešením, jak aplikovat známé metody typu *distance-based control* v případě, že dokážeme snímat pouze vzájemné vzdálenosti. Měřením většího množství vzdáleností uvnitř formace lze dopočítat relativní polohy některých (sousedních) agentů. Autoři zmiňují tři způsoby, kterými lze chybějící informaci získat.

1. Zařazením pevných kotev do systému. Přítomností pevných kotev se otevírají širší možnosti absolutní lokalizace, v další části práce jim bude věnována pozornost. Nesou s sebou ale také řadu omezení a prozatím se jim vyhneme.
2. Měřením azimutů uvnitř formace. Toho nejsme bez přídavného senzorového vybavení schopni.
3. Vzájemným pohybem agentů v rámci formace. Směr pohybu se musí po daném časovém intervalu změnit. To považuji v aplikaci s létajícími agenty za velmi těžko realizovatelné,



(a) : Plavidlo Medusa AMV **(b) :** Výsledky simulace

Obrázek 1.4: Simulace udržování trojúhelníkové formace, převzato z [SAPG12]

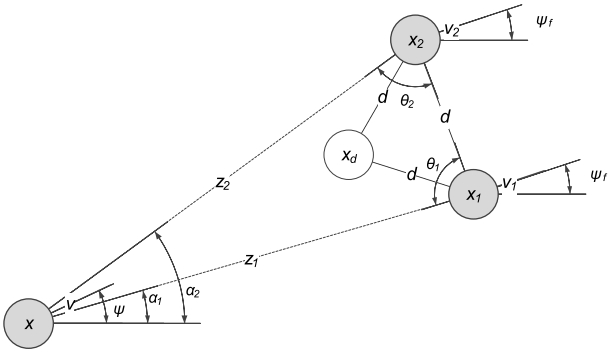
obzvláště pak v případě formace v pohybu. Bude-li se vůdce formace pohybovat, kvůli zpoždění v měření, řízení a pomalé dynamice samotných dronů by se dle mého názoru požadovaný pohyb „ztratil“ v regulačních odchylkách a šumu.

Triangular formation control using range measurements: An application to marine robotic vehicles [SAPG12] je způsobem zpracování mnohem blíže reálnému nasazení. Zatímco doposud citované práce byly spíše teoretického charakteru, formálně dokazovaly funkčnost navrženého algoritmu za podmínek, kterých v našem případě nejsme schopni dosáhnout, [SAPG12] svou metodu prezentuje na simulaci obsahující dynamický model námořního plavidla *Medusa AMV* (viz obr. 1.4a). Koncept počítá se dvěma vůdčími agenty plujícími vedle sebe, kteří jsou následování třetím agentem. Společně tak tvoří formaci tvaru rovnoramenného trojúhelníku. Sledující agent v tomto případě reguluje nezávisle na sobě dvě veličiny: velikost rychlosti a natočení v prostoru (směřování). Blíže se metodou zabývám v kapitole 1.3.

Flexible triangular formation keeping of marine robotic vehicles using range measurements [RSP+14] se zabývá taktéž řízením námořních plavidel a navazuje na článek [SAPG12]. Tvar trojúhelníkové formace je oproti modelu z [SAPG12] upraven. Dva vůdčí agenti se nepohybují vedle sebe, ale plují po stejné trajektorii za sebou. Sledující agent se drží v konstantní vzdálenosti po boku vůdců. Regulována je opět velikost rychlosti a směřování agentů. Tento článek, jako jediný z doposud citovaných, obsahuje nejen výsledky simulace, ale také závěry reálného experimentu. Koncept se ukázal jako funkční. Agenti na ploše cca 120x150 m urazili trajektorii ve tvaru písmene *S*. Chyba řízení se pohybovala v řádu jednotek metrů. Jedinou (avšak zásadní) komplikací tohoto konceptu je, že sledující agenti mají informaci o natočení vůdců v prostoru.

1.3 Udržování trojúhelníkové formace

Podíváme se blíže na metodu prezentovanou v [SAPG12]. Použitý přístup je pro nás výhodný hned z několika důvodů. Od počátku je navržen pro udržování pohyblivé formace v *leader-follower* režimu (sledování vůdce), ostatní agenti odhadují rychlost a směr formace a snaží se držet svou pozici. Funguje i s daty zatíženými šumem a vzájemná vzdálenost agentů je pro tuto metodu postačující vstupní veličinou. Koncept je navíc podložen funkční simulací na dynamických modelech námořních plavidel (obr. 1.4b).



Obrázek 1.5: Udržování trojúhelníkové formace, převzato z [SAPG12]

Formace je definována tvarem rovnoramenného trojúhelníku, viz obr. 1.5. Agenti x_1 a x_2 plní roli vůdců, agent x_3 (v obrázku x) je následuje. Takové uspořádání může na první pohled působit jako omezující. Cílem této práce je však najít metodu vhodnou pro udržení formace (ideálně pohyblivé) s užitím minimální senzorové výbavy agentů. Měření poskytnutá moduly DWM100 jsou zatížena značným šumem, a proto jsme připraveni přijmout nutné omezující podmínky. Tvar formace nepovažujeme za zásadní parametr pro případnou aplikaci.

Metoda stojí na odlišných principech než většina doposud zmíněných. Namísto minimalizace potenciálové funkce gradientní metodou (viz kapitola 1.1) je použit odhad rychlosti formace (absolutní hodnoty) a směřování. Použitelnost metody je dokázána pro případ rovnoměrného přímočarého pohybu vpřed (směrem od následovatele). Pro stabilní aplikaci je tedy nutné, aby změny rychlosti a směru formace probíhaly dostatečně pomalu.

Podívejme se podrobněji na algoritmus definovaný v [SAPG12]. V obr. 1.3 je naznačen význam základních veličin popisujících formaci. z_1 a z_2 určují vzdálenost sledujícího agenta od vůdců. Rychlost formace

$$v_f = \frac{v_1 + v_2}{2} \tag{1.5}$$

je definována vůdci a určuje požadovanou rychlost pohybu. Směr rychlosti formace ψ_f je vždy kolmý ke spojnici dvou vůdčích agentů. Velikost rychlosti sledujícího agenta je označena v a její směr v globálním souřadném systému ψ.

Základem regulace jsou veličiny popisující souhlasnou a rozdílovou složku chyby uspořádání formace. Definujeme souhlasnou složku

$$\epsilon = \frac{z_1 + z_2}{2} - d \tag{1.6}$$

a rozdílovou

$$\delta = z_2 - z_1 \tag{1.7}$$

s hodnotou d definující požadovaný tvar. Jak lze vytušit, souhlasná složka se stane vstupní veličinou pro regulátor rychlosti. Její hodnota stoupá, pokud sledující agent za vůdci „zaostává“. Rozdílová složka mění sovu hodnotu s vychýlením z trajektorie a je použita k regulaci směřování ψ.

Regulace rychlosti probíhá dle výrazu

$$v = K_p^s \epsilon + K_i^s \int_0^t \epsilon \, d\tau, \tag{1.8}$$

K_p^s a K_i^s jsou konstanty proporcionálního a integrálního zesílení. Směřování je regulováno jako

$$\psi = \hat{\psi}_f + \gamma\left(K_p^h \delta\right). \tag{1.9}$$

Funkce $\gamma(x)$ zamezuje otáčení o větší úhel než $\pi/2$ a může jít například o saturační funkci

$$\gamma(x) = \frac{\pi}{2} sat(x). \tag{1.10}$$

Problematickým prvkem rovnice (1.9) je odhad směřování formace $\hat{\psi}_f$. Autoři (jako další citovaní) zmiňují skutečnost, že tato informace bývá v praxi většinou známá. Je to však v rozporu s naším cílem provádět regulaci čistě na základě měření vzdálenosti. Naštěstí uvádí i způsob, jak odhad směřování formace získat ze vzájemné rychlosti agentů $\dot{z}$. Tu lze odhadovat například Kálmánovým filtrem. Následující vztah umožňující počítat orientaci formace:

$$\hat{\psi}_f = arcsin\left(\frac{\xi}{v} sin(\theta_1) - \frac{\dot{\hat{z}}_1}{v}\right) - \theta_1 + \psi. \tag{1.11}$$

Použitý člen ξ bohužel není v článku definován, na základě předchozí kapitoly a kontextu se domnívám, že pravděpodobně platí

$$\dot{\xi} = K_i \epsilon. \tag{1.12}$$

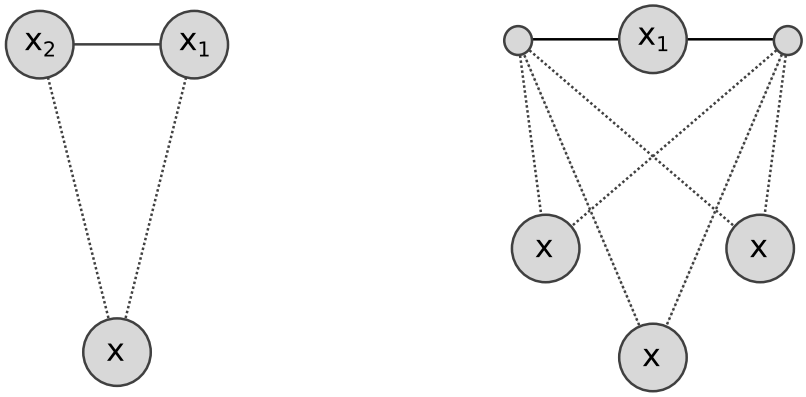
Rovnice (1.11) využívá pouze údaje o vzdálenosti z_1. Autoři uvádí ještě reciproční výraz využívající $\dot{\hat{z}}_2$, θ_2 a následně pracují s kruhovým průměrem obou výrazů, čímž zužitkují maximum dostupných informací a zlepší přesnost odhadu. Pro pochopení principu si však vystačíme s uvedeným odhadem (1.11).

Autoři nám tedy poskytli metodu řízení formace pracující bez nutnosti komunikace mezi vůdci a sledujícím agentem. (Vůdci samotní bez dalších úprav nějaké spojení potřebují pro zajištění synchronizovaného pohybu, viz dále.) Metoda funguje čistě na základě měření vzájemných vzdáleností. Důkazy stability regulátorů a konvergence regulační odchylky k nule jsou uvedeny v původním článku (a jsou platné pro rovnoměrný přímočarý pohyb). Ze všech studovaných metod tato nejlépe naplňuje naše představy o řešení problému, navíc je podložena výsledky simulace. Při hlubším vhledu do problému přesto přicházíme s jistými zjednodušeními a modifikacemi.

1.3.1 Navržené modifikace

Za nepříznivý považuji poměr počtu vůdčích a sledujících agentů. V původním provedení jsou využiti dva vůdčí agenti, které je schopen následovat pouze jeden sledující agent. To je v případech nutnosti přesunu většího množství letounů velmi omezující. Pokusíme se proto systém modifikovat tak, aby dva vůdce dokázalo sledovat větší množství autonomních plavidel či helikoptér.

Zaprvé je možné umístit několik sledujících agentů za sebe, aby každý udržoval vlastní rovnoramenný trojúhelník s odlišnou délkou stěn. A zobecníme-li navíc algoritmus tak, aby dokázal pracovat s formací ve tvaru obecného trojúhelníku (nikoli pouze rovnoramenného), získáme ještě vyšší variabilitu v umístění následovatelů za vůdce.



(a) : Původní dle [SAPG12] **(b) :** Upravená formace

Obrázek 1.6: Úprava trojúhelníkové formace s vůdci x_1, x_2 a sledujícími agenty x.

Původní návrh definuje vzdálenost d mezi sledujícím agentem a oběma vůdci. V obrázku 1.5 je dokonce současně použita i pro vzdálenost mezi vůdci. My zavedeme veličiny d_1, d_2 a umožníme, aby vzdálenost od každého z vůdců byla různá. Šířku vůdců označíme symbolem w. Rovnice (1.6) a (1.7) sloužící k výpočtu rozdílové a souhlasné složky chyby nahradíme výrazy reflektujícími možnou odlišnost obou vzdáleností. Souhlasná složka bude mít tvar

$$\epsilon_i = (z_{i1} + z_{i2}) - (d_{i1} + d_{i2}) \tag{1.13}$$

a rozdílová složka

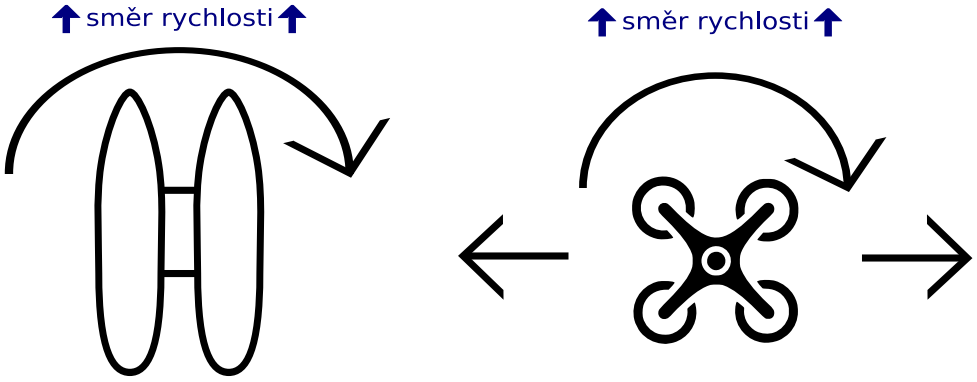
$$\delta_i = (z_{i1} - d_{i1}) - (z_{i2} - d_{i2}). \tag{1.14}$$

Index i značí jednoho z několika sledujících agentů, z_{i1} je vzdálenost agenta i od prvního vůdce, d_{i1} a její referenční hodnota.

Problematická je dle originálního návrhu rovněž nutnost užití dvou vůdčích agentů, kteří se musí pohybovat synchronizovaně. Toho nejsme bez užití dalšího senzorového vybavení schopni dosáhnout. V praxi by bylo nutné zajistit pohyb vůdců jiným způsobem a popisovanou metodu použít pouze pro sledující agenty.

Jednou z možností, jak omezení obejít, je umístění dvou senzorů vzdálenosti (rádiových modulů) na jednoho agenta. Oba vůdci se z podstaty mají pohybovat po přímé trajektorii s konstantní vzájemnou vzdáleností. Umístíme-li na vůdčího agenta dva moduly DWM1000 v dostatečném rozpětí, symetricky od středu a zároveň tak, aby spojnice modulů byla kolmá ke směru pohybu agenta, docílíme totožného efektu, jako kdyby v čele formace stáli dva vůdci. Tím se zcela dokážeme vyhnout nutnosti užití dvou vůdců se vzájemnou synchronizací pohybů.

Srovnání původního návrhu trojúhelníkové formace a modifikovaného návrhu přináší obr. 1.6. Verze 1.6b oproti 1.6a poskytuje možnost „zavěsit“ více sledujících agentů za jednoho vůdce. Přestává však platit část teorie popsané v původním článku a funkčnost algoritmu musíme ověřit experimentálně. Potíže očekávám zejména v situaci, kdy se sledující agent dostane se svou referenční polohou mimo prostor vytyčený rozpětím antén vůdce. V tu chvíli dochází s vychýlením do strany pouze k malým změnám rozdílové složky δ, zároveň výrazněji roste i



(a) : Námořní plavidlo **(b) :** Helikoptéra

Obrázek 1.7: Možnosti pohybu s cílem regulovat vychýlení do strany v trojúhelníkové formaci

hodnota souhlasné složky ϵ. V původním návrhu se sledující agent pohybuje uprostřed mezi vůdci a změna ϵ s vychýlením do stran je minimální.

Další modifikace se nabízí, uvědomíme-li si, že připravujeme systém pro udržování formace dronů. Článek, který nám je vzorem, pracuje s námořními plavidly. Autoři článku navrhli regulátor rychlosti a směřování plavidla, jelikož plavidlo Medusa s největší pravděpodobností neumožňuje řízení jiných veličin. Oproti tomu helikoptéra disponuje dalšími stupni volnosti, které lze řídit, jak naznačuje obr. 1.7. Dojde-li k vychýlení agenta do strany v průběhu pohybu formace, námořní plavidlo upraví své směřování a ke zmenšení výchylky dojde až s následným pohybem vpřed. Ovšem dron dokáže výchylku do strany regulovat přímo.

Další změny v implementaci

Při pokusech simulovat průběh algoritmu v kapitole 4.1 se nám nepodařilo realizovat odhad směřování formace podle původní rovnice (1.11) pracující se vzájemnou rychlostí agentů. Regulátor ve tvaru (1.9) nekonvergoval k požadované hodnotě s žádnou variantou testovaných konstant. Přistupme proto pro účely simulace k úpravě regulátoru do tvaru

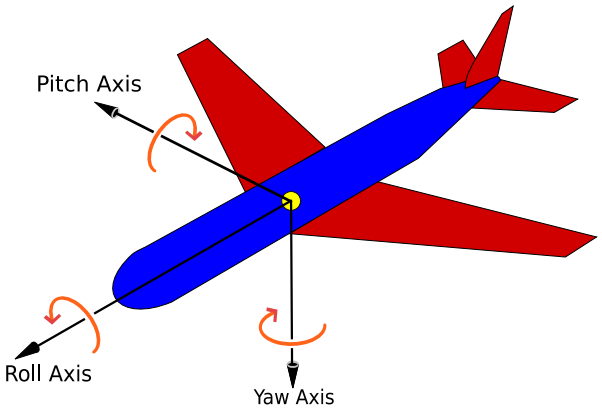
$$\psi = \gamma \left(K_p^h \delta \right) + K_i^h \int_0^t \delta \, d\tau. \tag{1.15}$$

Odhad směřování formace $\hat{\psi}$ byl zcela vypuštěn, v každém iteračním kroku pouze zjišťujeme aktuální výchylku δ. Přesouvá-li se formace kontinuálně jiným směrem než původně předpokládaným, na změnu zareaguje integrální složka regulátoru a dojde ke změně úhlu sledujícího agenta ψ_i.

Nyní opusťme otázku simulace. Při řízení modelu helikoptéry už vůbec směr rychlosti agenta vyjádřený formou úhlu nepotřebujeme. Nejsme ho schopni měřit ani regulovat. Úprava směřování je realizována postupným otáčením letounu (změnou úhlu ψ) s narůstající chybou δ. Otáčení se v čase integruje a dojde ke změně orientace dronu v prostoru. Efekt tak odpovídá rovnici (1.15), ovšem hodnotu úhlu ψ ve skutečnosti neznáme. Implementace algoritmu v knihovně pro řídicí PC obsahuje PD regulátor ve tvaru

$$\dot{\psi} = K_p^h \delta + K_d^h \dot{\delta}, \tag{1.16}$$

kde K_d^h je derivační konstanta. Ta zajistila při experimentech lepší odezvu letounu.



Obrázek 1.8: Standardní značení os letounů, převzato z [Wik21]

Společně s natočení dronu je vykonáván i pohyb do stran Ψ dle obr. 1.7b. V knihovně pro PC je řízen taktéž PD regulátorem

$$\Psi = K_p^r \delta + K_d^r \dot{\delta}, \tag{1.17}$$

kde K_p^r a K_d^r jsou proporcionální a derivační konstanty pro pohyb do stran.

Derivační složka byla doplněna i do regulátoru rychlosti (1.8). Dynamika modelu helikoptér se ukázala být velmi „pomalá“ a PI regulátor pro udržení vzdálenosti nebyl vyhovující. V knihovně pracujeme s verzí

$$v = K_p^s \epsilon + K_i^s \int_0^t \epsilon \, d\tau + K_d^s \dot{\epsilon}, \tag{1.18}$$

přičemž derivační konstanta je dominantní.

Ovládáním rychlosti v a pohybu do stran Ψ se rozumí řízení náklonu letounu v osách *pitch* a *roll* dle obr. 1.8. Změna náklonu vyvolá pohyb požadovaným směrem, ovšem se zpožděním a náběhem daným dynamikou systému. Proto jsou v regulátorech (1.18) a (1.17) zapotřebí derivační složky. Změnu úhlu $\dot{\psi}$ vykonáváme rotací v ose *yaw*.

V knihovně pro PC je navíc implementován klouzavý průměr délky 8 pro filtrování naměřených vzdáleností před vstupem do regulátorů.

Dalším drobným vylepšením je blokace otáčení v ose *yaw*, pokud rychlost nepřesahuje stanovený limit. Díky tomu je možno s formací zastavit a uspořádání zůstane stabilní. Drobné výchylky diferenciální složky chyby δ regulujeme pohybem do stran v ose *roll*. S algoritmem dle původního návrhu by zastavení letounu nebylo možné. Helikoptéra na rozdíl od plavidla přetrvává v otáčení i bez pohybu vpřed a to není žádoucím chováním.

Kapitola 2

Metody lokalizace s pevnými kotvami

V kapitole 1 jsme se zabývali možnostmi udržování formace bez znalosti polohy agentů v globálním souřadném systému. Formace byla regulována minimalizací potenciálové funkce či korekcí vzájemných vzdáleností mezi agenty (*distance-based* metody). Cílem bylo umožnit přesun agentů na větší vzdálenosti a udržet tvar formace i při pohybu. Nyní se podíváme na odlišný přístup využívající systému pevných kotev.

Článek [AY11] zmiňoval umístění kotev do systému jako jednu z možností pro získání odhadu relativní polohy sousedních agentů. Následně pracoval s klasickým schématem *distance-based* metod. To jsem vyhodnotil jako nevyužití potenciálu kotev. Přijmeme-li omezení s kotvami spjatá, zvolíme raději úplně jiný náhled na problém.

Kromě možnosti udržování formací získáváme schopnost snímat a zaznamenávat polohu letounů při ručním řízení nebo lokalizovat libovolný jiný objekt v globálním souřadném systému. Znalost globální polohy lze následně využít pro udržování formace, v úvodu kapitoly 1 jsme zmiňovali metody *position-based* řízení. Možné aplikace ale tuto oblast dalece přesahují.

Hlavním omezením je nutnost pohybovat se v blízkosti pevných kotev, složitější jsou přesuny formace na větší vzdálenosti. Poté na „kotvy se známou polohou“ musíme nahlížet jako na letouny vybavené pokročilejší lokalizační technikou, které ostatním agentům definují souřadný systém. Celá formace se lokalizuje vůči tomuto systému a můžeme klidně provozovat přesuny v globálním souřadném systému.

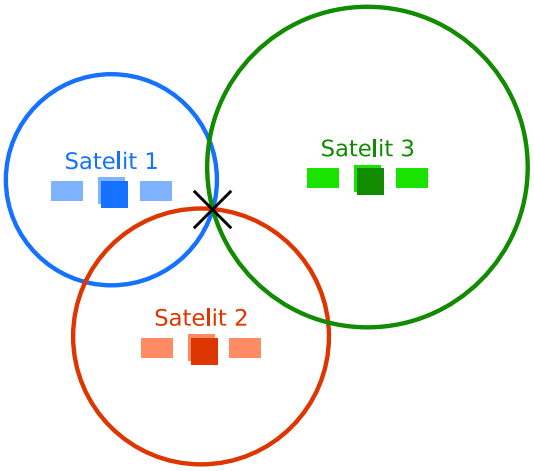
Cílem této kapitoly bude nalézt způsob lokalizace na základě znalosti vzdálenosti k pevným kotvám.

2.1 Multilaterace

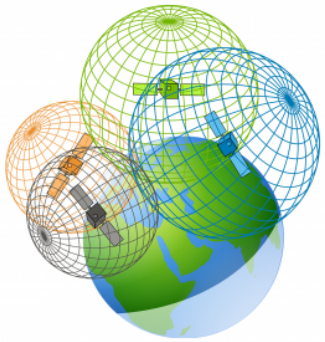
První možností, která se nabízí pro řešení našeho problému, je multilaterace. Jedná se v principu o hledání bodu v prostoru, jehož vzdálenost k bodům známé polohy (kotvám) měříme. Multilateraci využívá mj. globální lokalizační systém GPS.

Speciálním případem multilaterace je obecně známější trilaterace. V jejím případě jsou použity 3 body definovaných souřadnic. Hledaný bod odpovídá průsečíku kružnic se středem v oněch bodech a poloměrem odpovídajícím naměřeným vzdálenostem (obr. 2.1a). Jsou-li měření přesná, kružnice se protnou přímo v hledaném bodě.

V případě bodu v prostoru $p \in \Re^3$ je minimální počet kotev (známých bodů) pro jednoznačnou lokalizaci 4, viz obr. 2.1b. Kružnice jsou nahrazeny kulovými plochami. V praxi navíc nelze



(a) : Trilaterace ve 2D



(b) : Multilaterace, převzato z [GIS21]

Obrázek 2.1: Princip multilaterace

dosáhnout přesných měření, chybu potlačujeme užitím ještě většího množství kotev. Kulové plochy poté nemají jednoznačný průsečík a jeho aproximace je nalezena optimalizačními metodami. Použít lze např. metodu nejmenších čtverců. Úloha je definována jako hledání přibližného řešení přeurčené soustavy ve tvaru

$$\begin{aligned}(x-x_1)^2+(y-y_1)^2+(z-z_1)^2-d_1^2&=0\\(x-x_2)^2+(y-y_2)^2+(z-z_2)^2-d_2^2&=0\\(x-x_i)^2+(y-y_i)^2+(z-z_i)^2-d_i^2&=0,\end{aligned} \tag{2.1}$$

kde x, y, z jsou souřadnice hledaného bodu p a x_i, y_i, z_i souřadnice pevných kotev. Naměřené vzdálenosti k příslušným kotvám značíme d_i.

2.2 Kálmánův filtr

Na výsledek multilaterace se přenáší šum z měřených vzdáleností d_i. Hojně užívanou metodou, jak šum potlačit, je Kálmánův filtr, jehož princip vychází z článku R. E. Kálmána [Kal60]. Filtr je podrobně a srozumitelně popsán také v [RSH17], odkud přejímám značení pro účely této kapitoly. Článek vysvětluje principy a postup návrhu filtru, funkčnost je demonstrována na přiloženém kódu v Matlabu.

Základní funkcí filtru je odhadovat stav lineárního systému popsaného stanovými rovnicemi. Pro odhad postačuje znalost předchozích odhadů, jejichž aktualizaci provádíme na základě provedených měření. Měření i samotné hodnoty stavů systému jsou zatíženy nejistotou (šumem). Jsme schopni odhadovat i stavy, které nejsou přímo měřitelné, nutnou podmínkou je jejich pozorovatelnost (z pohledu teorie řízení).

Zkoumaný lineární systém v diskrétním čase popíšeme rovnicemi

$$x_k = F_{k-1}x_{k-1} + G_{k-1}u_{k-1} + w_{k-1} \tag{2.2}$$

$$y_k = H_k x_k + v_k, \tag{2.3}$$

kde x je stavový vektor systému, u vstupní a y výstupní vektor. Symbol w značí procesní šum jednotlivých stavů, v šum měření výstupu. F, G a H jsou matice popisující dynamiku

systému (stavová, vstupní a výstupní). Samotná filtrace probíhá ve dvou krocích: predikčním a korekčním. Nejprve se pokoušíme odhadnout hodnotu stavu na základě dynamiky systému a jeho předpokládané hodnoty v minulosti. Následně srovnáváme odhadnutou hodnotu s nově získanými daty z měření a odhad aktualizujeme. Zmíněné provádíme s ohledem na předpokládanou úroveň šumů v a w.

Predikční krok implementujeme jako

$$\hat{x}_{k|k-1} = F_{k-1}\hat{x}_{k-1} + G_{k-1}u_{k-1}. \tag{2.4}$$

$\hat{x}_{k|k-1}$ značí odhad stavu x v kroku k na základě dat z kroku $k-1$. Tedy bez použití aktuálních naměřených dat. Následně dojde k odhadu kovarianční matice P chybového vektoru stavů. Ten získáme jako

$$P_{k|k-1} = F_{k-1}P_{k-1}F_{k-1}^T + Q_{k-1}, \tag{2.5}$$

kde Q je kovarianční matice procesního šumu w. (Před zahájením filtrace je potřeba inicializovat hodnotu P_0.)

Hodnotu $P_{k|k-1}$ využijeme k určení Kálmánova koeficientu K. Ten stanovuje, jak silná bude korekce odhadu $\hat{x}_{k|k-1}$ v dalším kroku filtrace. Píšeme

$$K_k = P_{k|k-1}H_k^T \left(H_k P_{k|k-1} H_k^T + R_k\right)^{-1}, \tag{2.6}$$

kde R je kovarianční matice (výstupního) šumu měření v. Se znalostí K provedeme korekční krok filtrace. Na základě rozdílu mezi odhadovaným výstupem $\hat{y}_{k|k-1} = H_k\hat{x}_{k|k-1}$ a získaným měřením z_k aktualizujeme odhad stavu

$$\hat{x}_k = \hat{x}_{k|k-1} + K_k\left(z_k - H_k\hat{x}_{k|k-1}\right). \tag{2.7}$$

Poslední operací filtrační smyčky je aktualizace kovarianční chybové matice P dle výrazu

$$P_k = (I - K_k H_k)\, P_{k|k-1} \tag{2.8}$$

se symbole I vyjadřujícím jednotkovou matici.

Kromě zajištění popisu systému vhodným modelem a maticemi F, G, H musíme při konstrukci filtru stanovit také odhad kovariančních matic P_0, Q a R. Matice P_0 vypovídá o kvalitě odhadu počátečního stavu a ovlivňuje převážně náběh filtru. Důležitější je vhodně nastavit koeficienty matic Q a R. Ty se uplatňují po celou dobu filtrace a mají přímý vliv na výpočet Kálmánova koeficientu K.

Na základě rovnice (2.5) narůstá hodnota odhadu $P_{k|k-1}$ s nárůstem Q. Tedy čím větší předpokládáme šum stavových proměnných, tím větší je i hodnota Kálmánova koeficientu dle (2.6). Z toho plyne výraznější úprava odhadu $\hat{x}_k$ v korekčním kroku. Šum stavových proměnných reprezentuje nepřesnosti modelu systému, např. vstupy, které nejsme schopni měřit, a které způsobují změnu skutečného stavu x v rozporu s modelem. Rovnice (2.6) naopak říká, že koeficient K s narůstající hodnotou R klesá. Tedy s nižší vypovídající hodnotou výstupu systému se spoléháme více na predikční část algoritmu. Zjednodušeně řečeno, poměr hodnot koeficientů matic Q a R určuje, jak silně bude filtr „filtrovat“.

Koeficienty kovariančních matic určíme na základě odhadovaných rozptylů σ^2. Pro vektorovou veličinu $y \in \Re^2$ a její kovarianční matici R bychom např. psali

$$R = \begin{bmatrix} \sigma^2_{x_1} & \sigma^2_{x_1x_2} \\ \sigma^2_{x_1x_2} & \sigma^2_{x_2} \end{bmatrix}, \tag{2.9}$$

přičemž koeficienty mimo hlavní diagonálu jsou nulové, jsou-li komponenty vektoru y navzájem nezávislé. Předpokládáme, že se jedná o gaussovský šum.

Pozn.: Popsali jsme si fungování diskrétního provedení Kálmánova filtru, vhodného pro digitální zpracování signálů. Existuje samozřejmě i verze pro systémy ve spojitém čase.

2.2.1 Model mobilního agenta

Při pohledu na problém globální lokalizace jako na výpočet polohy objektu multilaterací a následné filtrace výsledků můžeme systém popsat stavovým vektorem $x \in \Re^2$ jako

$$x(k) = \begin{bmatrix} x_1 & x_2 \end{bmatrix}, \tag{2.10}$$

kde x_1, x_2 jsou souřadnice značící polohu agenta v rovině v čase k. Ve článku [BKAM13], který se zabývá podobným problémem, jsou uvedeny navíc ještě stavy pro složky rychlosti. Vzhledem k tomu, že v konfiguraci s pevnými kotvami není umožněn přesun agentů na větší vzdálenosti a cílem je v našem případě sledování polohy objektů na omezeném prostoru, nepředpokládá se soustavný pohyb konstantní rychlostí. Stavy popisující rychlost proto vypustíme a vystačíme si se dvěma stavy.

Oba stavy (souřadnice bodu) dokážeme prostřednictvím multilaterace přímo měřit. Zároveň u obou stavů předpokládáme konstantní hodnotu, jelikož o pohybu objektu nemáme bližší informace. Ke změně polohy z pohledu popisu systému dochází pouze vlivem šumu w_k. O vstupu $u \in \Re$ také nemáme informace. Ve vztahu k rovnicím (2.2) a (2.3) můžeme tedy psát

$$F = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, G = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, H = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{2.11}$$

Uvedený model může být použit pro filtrování výsledků multilaterace a získání polohy sledovaného objektu v rovině.

2.3 Další aplikace a formy Kálmánova filtru

V kapitole 2.2 jsme si nastínili princip fungování Kálmánova filtru s cílem využít ho ke zpracování výsledků multilaterace. Existují však i odlišné přístupy k lokalizaci mobilních agentů. Několik dalších přístupů uvádí článek *Divided Difference Kalman Filter for Indoor Mobile Localization* [BKAM13].

Autoři se zabývají problémem lokalizace na základě měření vzdálenosti. K určování vzdálenosti přitom využívají metodu RSSI, kterou popisuji v kapitole 3.1.3. V rámci článku jsou představeny 4 různé přístupy využívající odlišné typy Kálmánova filtru. Jedním z nich je koncept založený na multilateraci, popsaný výše (kapitola 2.2.1). Výsledky multilaterace jsou za pomoci klasického Kálmánova filtru (alespoň částečně) zbaveny šumu. Dále článek popisuje aplikaci tzv. *Divided Difference* Kálmánova filtru, *Unscented* Kálmánova filtru a *Extended* Kálmánova filtru. Všechny uvedené přístupy jsou testovány v simulaci a v závěru porovnány. My si nyní stručně vysvětlíme jejich princip.

2.3.1 EKF

Nejprve se zaměříme na *Extended Kalman Filter* (EKF). Podrobný a srozumitelný popis metody opět nabízí článek [RSH17]. Jedná se v zásadě o rozšíření Kálmánova filtru umožňující

nasazení na nelineární systémy. Klasický Kálmánův filtr, jak jsme ho definovali v kapitole 2.2, vyžaduje systém popsaný lineárními stavovými rovnicemi. Ty jsou potřebné pro sestavení matic F, G a H, viz rovnice (2.2) a (2.3). Můžeme se však setkat se systémy, kde samotná dynamika stavů nebo způsob jejich měření (výstupní rovnice) obsahují nelineární prvky. Takový systém popisujeme obecnými rovnicemi

$$x_k = f\left(x_{k-1}, u_{k-1}\right) + w_{k-1} \tag{2.12}$$

$$y_k = h\left(x_k\right) + v_k, \tag{2.13}$$

kde f je funkce určující dynamiku stavů a h výstupní funkce. V obou případech se jedná o funkce vektorové (nelineární).

Abychom mohli aplikovat postup filtrace z kapitoly 2.2, spočítáme Jacobiho matice funkcí f a h. Pro časový moment k píšeme

$$F_k = \left.\frac{\partial f}{\partial x}\right|_{\hat{x}_{k-1}} \tag{2.14}$$

$$H_k = \left.\frac{\partial h}{\partial x}\right|_{\hat{x}_{k|k-1}} \tag{2.15}$$

a matice F_k, H_k použijeme pro popis linearizovaného systému v daném bodě. Proces filtrace vychází z lineárního případu.

Pro predikční krok použijeme plný (nelineární) tvar funkcí f, h namísto rovnice (2.4), tedy

$$\hat{x}_{k|k-1} = f\left(\hat{x}_{k-1}, u_{k-1}\right). \tag{2.16}$$

Následuje výpočet Jacobiho matice dle (2.14) a (2.15), čímž získáme F_k, H_k. Odhad

$$P_{k|k-1} = F_{k-1} P_{k-1} F_{k-1}^T + Q_{k-1} \tag{2.17}$$

je shodný s lineární verzí filtru v rovnici (2.5), stejně jako výpočet koeficientu

$$K_k = P_{k|k-1} H_k^T \left(H_k P_{k|k-1} H_k^T + R_k\right)^{-1}. \tag{2.18}$$

Korekční krok

$$\hat{x}_k = \hat{x}_{k|k-1} + K_k \left[z_k - h\left(\hat{x}_{k|k-1}\right)\right] \tag{2.19}$$

využívá nelineárního tvaru funkce h a výraz

$$P_k = \left(I - K_k H_k\right) P_{k|k-1} \tag{2.20}$$

je beze změny, kromě toho, že matice H_k nepopisuje systém přesně, ale je získána linearizací.

V případě modelu reprezentovaného maticemi (2.11) jsme si vystačili se základní variantou Kálmánova filtru. Po zavedení EKF máme možnost rozšířit popis systému o nelineární prvky a zcela vypustit krok výpočtu multilaterace. Zůstaneme u předpokladu, že stav $x \in \Re^2$ reprezentující polohu objektu v rovině se mezi časovými kroky mění jen o úroveň procesního šumu w_k. Systémová matice tak zůstane ve tvaru

$$F = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{2.21}$$

Nezmění se ani náš pohled na vstupy, o nichž nemáme informaci, platí nadále

$$G = \begin{bmatrix} 0 \\ 0 \end{bmatrix}. \tag{2.22}$$

Rozdíl nastává u výstupní matice H_k. Zahrneme do ní přímo informaci o tom, že neměříme polohu, nýbrž vzdálenost agenta od kotev se známými souřadnicemi. Funkce $h(x_k)$ definovaná v rovnici (2.13) nabude tvaru

$$h(x_k) = \begin{bmatrix} \sqrt{(x_1^2 - p_{11}^2) + (x_2^2 - p_{12}^2)} \\ \sqrt{(x_1^2 - p_{21}^2) + (x_2^2 - p_{22}^2)} \\ \dots \\ \sqrt{(x_1^2 - p_{i1}^2) + (x_2^2 - p_{i2}^2)} \end{bmatrix}, \tag{2.23}$$

kde x_1, x_2 značí složky aktuální polohy agenta ve 2D v momentě k a p_{i1}, p_{i2} značí souřadnice pevné polohy kotvy i. Počet použitých kotev definuje dimenzi výstupního vektoru y, jelikož z každé kotvy přichází samostatný údaj o vzdálenosti k agentu.

Matici H_k zbývající k dokončení linearizovaného modelu systému získáme v souladu s rovnicí (2.15) jako

$$H_k = \begin{bmatrix} \frac{x_1 - p_{11}}{d_1^2} & \frac{x_2 - p_{12}}{d_1^2} \\ \dots & \dots \\ \frac{x_1 - p_{i1}}{d_i^2} & \frac{x_2 - p_{i2}}{d_i^2} \end{bmatrix}. \tag{2.24}$$

Členy d_i značí vzdálenost agenta od kotvy i ve tvaru

$$d_i = \sqrt{(x_1 - p_{i1})^2 + (x_2 - p_{i2})^2}. \tag{2.25}$$

Kovarianční matice výstupu R bude mít diagonální tvar

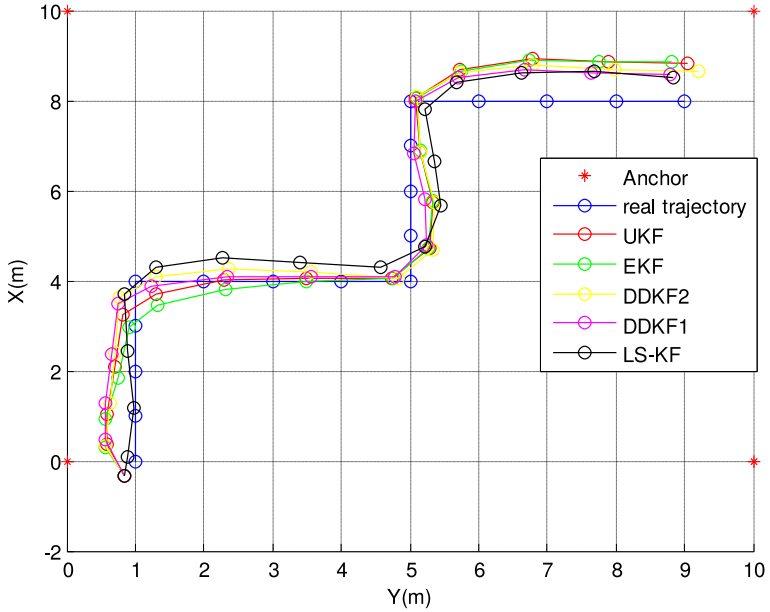
$$R = \begin{bmatrix} \sigma_1^2 & 0 & \dots \\ 0 & \sigma_2^2 & \dots \\ \dots & \dots & \sigma_i^2 \end{bmatrix}, \tag{2.26}$$

přičemž σ_i je rozptyl šumu měření vzdálenosti k pevnému bodu i. Dá se předpokládat, že použité kotvy osazené stejným senzorem budou dosahovat stejných šumových parametrů, takže hodnota rozptylu bude stejná pro všechna i.

2.3.2 UKF, DDKF

Kromě přístupu využívajícího multilateraci s běžným Kálmánovým filtrem a lokalizace založené na EKF prezentuje článek [BKAM13] také implementaci tzv. *Unscented Kalman Filter*, zkráceně UKF. EKF založený na linearizaci systému v daném bodě funguje dobře pro systémy, jejichž nelinearita není příliš vysoká. Pro silně nelineární systémy přestává být dle autorů [BKAM13] EKF efektivní.

Jednou z možných alternativ EKF je právě UKF. Ten pomocí *unscented* transformace odhaduje statistické parametry proměnných na základě vzorku *sigma* bodů. Druhou alternativou, kterou autoři zpracovali, je *Divided Difference Kalman Filter* (DDKF). Testována byla implementaci prvního i druhého řádu. Základem procesu je v obou případech *Stirlingova interpolace*. Přestože UKF i DDKF jsou hojně využívanými instrumenty, v této práci jim nebude věnována větší pozornost, neboť nebyly pro řešení našeho úkolu zvoleny.



Obrázek 2.2: Odhad trajektorie agenta s užitím pevných kotev v simulaci. Srovnání použitých metod, převzato z [BKAM13].

2.4 Srovnání přístupů

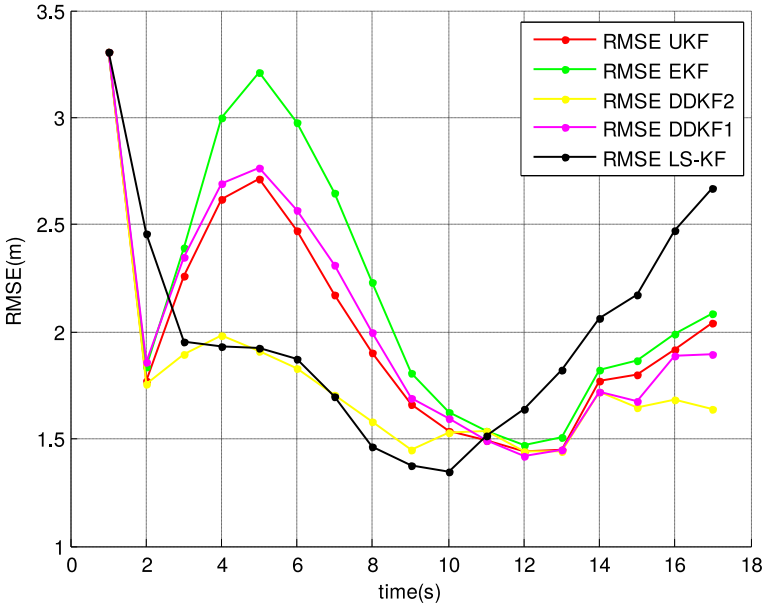
Ukázali jsme si několik typů Kálmánova filtru a jejich využití pro lokalizaci mobilního agenta. Filtr byl nasazen buď pro zlepšení výsledků multilaterace, nebo jako instrument pro přímý odhad polohy z naměřených vzdáleností. Každá ze zmíněných variant má výhody i nevýhody. Volba vhodného nástroje záleží vždy na preferencích uživatele a zamýšlené aplikaci. Autoři článku [BKAM13] srovnávali mnoho aspektů zkoumaných filtrů v úkolu lokalizace za použití pevných kotev.

V rámci simulací provedených v [BKAM13] se agent pohyboval po trajektorii vyznačené na obr. 2.2 modrou barvou. Ostatními barvami jsou vyneseny odhady polohy s užitím jednotlivých metod. Obr. 2.3 poté zaznamenává chybu lokalizace v čase. Z grafů je patrné, že nejmenší chyby dosáhli autoři s metodou DDKF2, tedy *Divided Difference Kalman Filter* 2. řádu. Jako nejpřesnější tuto metodu vyhodnotili i v závěru článku. Rozdíly však nejsou diametrální. Medián chyby pro nejpřesnější metodu (DDKF2) dosahoval $1,7\,\mathrm{m}$, pro nejméně přesnou metodu (EKF) $1,9\,\mathrm{m}$. Maximální chyba pro zaznamenaná EKF byla $3,2\,\mathrm{m}$, pro DDKF2 pouze $2\,\mathrm{m}$. (Hodnoty přibližně odečteny z grafů.)

Podstatný vliv na výkon filtrů má volba počátečních parametrů. Je možné, že s jiným nastavením konstant by bylo pořadí dosažené přesnosti odlišné. Autoři uvádí, že nejcitlivější na počáteční volbu konstant je UKF. V textu bylo vyhodnoceno také několik různých rozložení kotev v rámci sledovaného prostoru. Jako nejlepší se obecně jeví umístění pevných bodů po obvodu (v případě obdélníkového tvaru do jeho rohů).

Důležité je poznamenat, že citlivost algoritmů na změnu uspořádání kotev se velmi lišila. DDKF2 s jiným uspořádáním kotev dosahoval horších výsledků než EKF, na něhož měla změna vliv minimální. Výpočetní náročnost přístupů je řádově srovnatelná, pohybuje se od 766 do 2341 operací na jeden cyklus filtrace, přičemž nejnáročnější je UKF a DDKF2.

Autoři vyhodnotili jako nejvhodnější pro danou aplikaci metodu DDKF2 (jak napovídá už



Obrázek 2.3: Chyba lokalizace agentů s užitím pevných kotev v simulaci. Srovnání použitých metod, převzato z [BKAM13].

název článku). Já jsem však došel k závěru, že výkon metod je srovnatelný. Změna počátečních konstant a zejména jiné rozmístění kotev by vedly k odlišnému výsledku. Kromě toho jsem přihlédl i k implementační náročnosti testovaných přístupů a rozhodl jsem se pro použití EKF.

2.4.1 Změny v implementaci

Při implementaci algoritmu popsaného v kapitole 2.3.1 došlo k drobným úpravám.

Nedokonalost původního návrhu spočívá ve skutečnosti, že k provedení kroku filtrace potřebujeme znát najednou hodnoty všech výstupů systému. Je nutné změřit vzdálenost ke všem zařazeným kotvám a poté provést odhad změny polohy. To se ukázalo jako problematické. Odměr vzdálenosti je poměrně pomalý a než provedeme měření se čtvrtou kotvou (při použití čtyř kotev), vzdálenost od první již nemusí být aktuální. Navíc výsledek odhadu získáme se čtyřikrát menší frekvencí, než by bylo možné.

Popis systému stavovými rovnicemi jsem proto při implementaci do knihovny pro PC upravil. Namísto reprezentace každé kotvy samostatným výstupem dle rovnice (2.23) zavádím pouze jeden výstup systému. Dimenzionalita matice H (2.24) se tak zmenší na $1 \times n$ (n je počet souřadnic polohy, tedy 2 nebo 3). A platí

$$H_k = \begin{bmatrix} \frac{x_1 - p_1}{d_1^2} & \frac{x_2 - p_2}{d_1^2} \end{bmatrix} \tag{2.27}$$

pro lokalizaci v $\Re^2$. (Knihovna umožňuje i lokalizaci v $\Re^3$.)

Průchod Kálmánovým filtrem tak můžeme provádět po každém odměru vzdálenosti od mobilního agenta k libovolné kotvě. Polohu kotvy p_1, p_2 v matici H vždy změníme dle toho, jaké výsledky momentálně získáme. Tento přístup nevyžaduje provádět měření vzdáleností ke kotvám v pravidelném pořadí. Selže-li jeden z pokusů o odměr, můžeme bez problémů pokračovat další kotvou a nezpomalovat odezvu systému.

Kapitola 3

Použité prostředky

3.1 Moduly DWM1000

Cílem práce je navrhnout algoritmus pro udržování formace s využitím modulů DWM1000 (obr. 3.1). Jedná se o moduly založené na čipu DW1000 sloužícím k UWB (ultra-wide-band) komunikaci dle standardu IEEE802.15.4-2011. Na stejném standardu je vystavěna např. technologie ZigBee [Zig21].

Modul DWM1000 integruje zmíněný čip DW1000, napájecí obvody, anténu a časovače. Níže uvádím základní technickou specifikaci zařízení dle datasheetů DW1000 [Dec15] a DWM1000 [Dec16].

- Podpora frekvenčních pásem od 3,5 GHz do 6,5 GHz
- Nízká spotřeba energie
- Nastavitelný výkon antény
- Přenosová rychlost od 110 kbps do 6,8 Mbps
- Dosah až 290 m
- Odolnost k vícecestnému šíření signálu
- Podpora lokalizace s přesností až 10 cm
- Podpora TDOA (*time difference of arrival*) a *2-way ranging*

3.1.1 Dosah komunikace

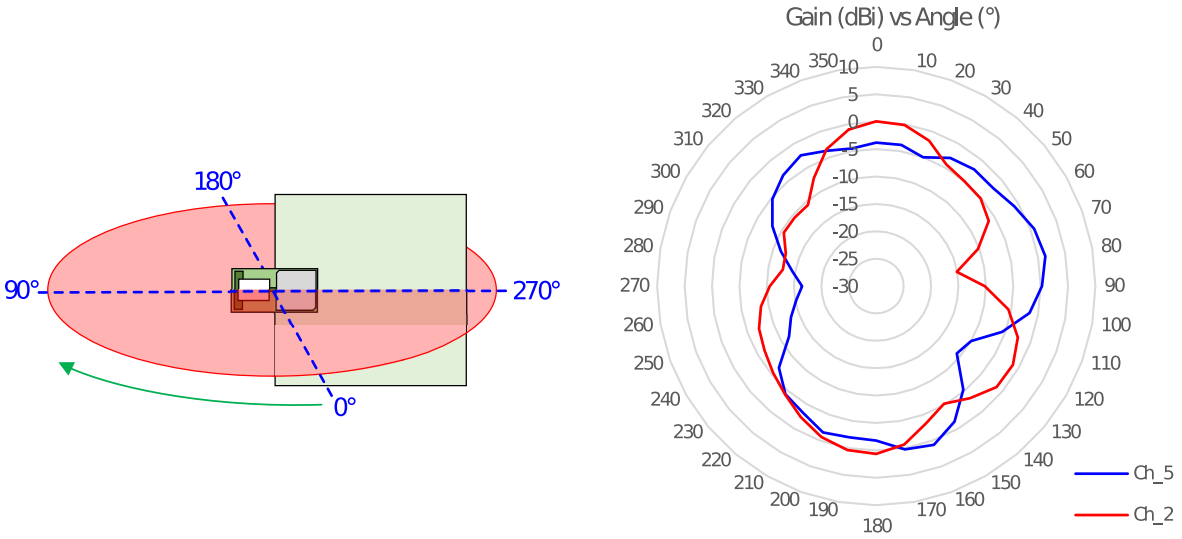
Za důležité parametry pro reálné nasazení jsme považovali dosah komunikace a přenosovou rychlost. V souvislosti s tím je třeba uvědomit si skutečnost, že ne všech parametrů uvedených v datasheetu je možno dosáhnout současně. Je-li pro nás prioritní dosah komunikace, musíme se smířit s nižší přenosovou rychlostí (110 kbps), a tedy se zvýšenou latencí celého systému. Naopak při vyšších rychlostech se budeme potýkat s výrazně nižším dosahem.

Problematický je i samotný údaj maximálního dosahu. Udávaná hodnota může definovat oblast, ve které lze provozovat spolehlivou komunikaci. Zároveň však může jít o nejvyšší naměřenou vzdálenost, při které došlo v rámci experimentu k úspěšnému přenosu dat. V nejméně příznivém případě jde o čistě teoretický údaj.

V praxi může nastat situace (vlivem odrazu signálu od povrchu země), kdy dokážeme přenášet data např. na vzdálenost 150 m mezi moduly, ale s přiblížením na 100 m dojde ke ztrátě signálu. Údaj maximálního dosahu 150 m potom není pro naši aplikaci vypovídající, jelikož by komunikace v některých částech území nebyla spolehlivá.

Při posuzování dosahu komunikace musíme brát v potaz také směrovou charakteristiku antény. Z obr. 3.2 vyplývá, že zisk antény závisí na směru a na použitém komunikačním

Obrázek 3.1: Modul DWM1000, převzato z [Dec21]



Obrázek 3.2: Směrová charakteristika antény v jedné rovině [Dec16]

kanálu. Při natočení antény o 90° dojde k poklesu zisku přibližně o 15 dBi. Dosah komunikace ve směru s vyšším útlumem je potom na zlomku maximální hodnoty.

Problematika dosahu komunikace v bezdrátových sítích je velmi komplikovaná a údaj z datasheetu musíme vnímat jako čistě orientační. Detailní informace o dosahu a jeho optimalizaci v sítích postavených na DW1000 lze čerpat např. z [Dec14b].

3.1.2 Vliv oddrazů signálu

Nesmíme zapomenout také na to, že odražený signál sice může poskytnout dostatečně robustní komunikační kanál. Ale informace potřebná k provedení lokalizace (typicky doba šíření signálu) je odrazem často znehodnocena, jelikož s odrazem nedochází k šíření po nejkratší možné dráze.

S tím je spojen problém vícecestného šíření signálu. Bezdrátové technologie jsou často označovány jako „imunní vůči vícecestnému šíření“. Také datasheet k čipu DW1000 [Dec15] obsahuje zmínku o *high multipath fading immunity*. V praxi to znamená, že zprávy lze doručovat i v prostředí, kde se signál šíří paralelně po několika trajektoriích a do přijímače přichází více kopií radiového signálu posunutého v čase. Přijímač dokáže izolovat původní informaci a dekódovat validní data. Není však zaručeno, že doručená zpráva přišla „nejkratší možnou cestou“.

K vícecestnému šíření signálu dochází nejčastěji v uzavřených prostorách, někdy i v případě přímé viditelnosti mezi agenty (např. v blízkosti stěn). V konfiguraci bez přímé viditelnosti je problém ještě výraznější. Pokud není možná komunikace po přímé spojnici v prostoru (signál je blokován překážkou), je lokalizace velmi ztížena. Bez znalosti prostředí a definovaného rozmístění překážek nejsme schopni vzdálenost mezi agenty měřit.

3.1.3 Měření vzdálenosti pomocí DWM1000

S pomocí modulů jsme schopni měřit vzájemnou vzdálenost mezi agenty, tato funkce však není v hardwaru přímo implementována. Moduly poskytují informaci o síle přijímaného signálu a časových značkách přijatých a odeslaných zpráv.

Aplikační poznámky k modulům DW1000 [Dec14a] uvádí několik přístupů, jak z měřených veličin získat vzájemnou vzdálenost.

- Metody RSSI *(Received Signal Strength Indication)* jsou založené na poklesu výkonu přijímaného signálu s narůstající vzdáleností. Se znalostí vysílacího výkonu lze na straně přijímače odhadovat vzdálenost od zdroje vysílání. Přístup vyžaduje znalost prostředí, kterým se signál šíří (pro určení vztahu útlumu na vzdálenosti).
- Metody založené na měření času
  - TOF *(Time of Flight)* měří přímo dobu šíření signálu prostředím.
  - TDoA *(Time Difference of Arrival)* využívá k určení vzdálenosti rozdílný čas doručení jedné zprávy mezi agenty rozmístěnými v prostoru.
  - PDoA *(Phase Difference of Arrival)* signál je přijímán maticí antén, směr zdroje signálu určuje fázový posun signálu mezi anténami. Vzdálenost poté získáme ze znalosti směru od několika agentů.

Pro námi zamýšlenou aplikaci, tedy měření vzdálenosti s cílem lokalizace agentů v reálném čase, se jeví jako nejvhodnější použít metody TOF. Oproti RSSI dosahují vyšší přesnosti a k implementaci není potřeba tolik antén jako v případě metod PDoA.

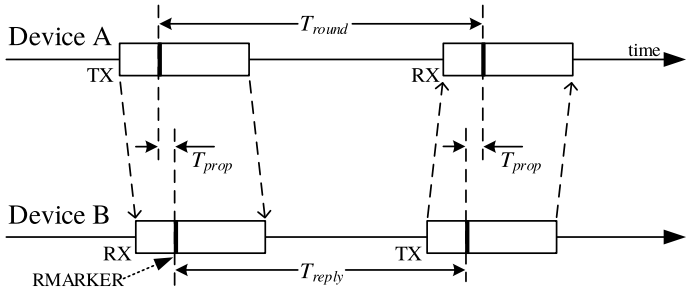
Metody TOF

Základní schéma TOF metody počítá s možností synchronizace hodin mezi agenty. První agent (iniciátor) vyšle požadavek k měření v čase t_1. Zpráva obsahuje časovou značku odeslání zprávy. Druhý agent (respondent) zprávu obdrží, zaznamená čas přijetí t_2 a údaj porovná s časem odeslání obsaženým v těle zprávy. Vzdálenost d mezi iniciátorem a respondentem poté odpovídá součinu

$$d = (t_2 - t_1) \cdot c, \tag{3.1}$$

kde c je rychlost šíření signálu prostředím. Ve volném prostoru se blíží rychlosti šíření světla ve vakuu, tedy $c \approx 3 \cdot 10^8\, m/s$.

Pro představu, s oddálením agentů o $1\,cm$ se prodlouží doba šíření signálu o pouhých $3,3 \cdot 10^{-11}$ s. Máme-li ambice měřit vzdálenost s podobným rozlišením, potřebujeme použít časovače s frekvencí v řádu desítek GHz. Moduly DWM1000 takového rozlišením dosahují, LSB časových značek má hodnotu přibližně $15,7 \cdot 10^{-12}$ s [Dec17]. Synchronizace hodin mezi agenty s dostatečnou přesností však není realizovatelná. Z výše uvedených důvodů hodiny synchronizované nejsou a musíme využít systém dvoucestného měření vzdálenosti.



Obrázek 3.3: Diagram jednostranného dvoucestného měření vzdálenosti (*Single-sided Two-way ranging*), převzato z [Dec17]

Dvoucestné měření vzdálenosti (jednostranné)

Dvoucestné měření vzdálenosti přináší způsob, jak u metod TOF odstranit potřebu synchronizace hodin [Dec17]. (Časový diagram můžeme vidět na obr. 3.3.) Podstatou je měření intervalů T_{round} a T_{reply}. Zařízení A odešle požadavek na měření k zařízení B. Doba od odeslání požadavku do doručení odpovědi je označena jako T_{round}. Čas, který potřebuje zařízení B na zpracování požadavku, značíme T_{reply}. Rozdíl těchto hodnot vypovídá o době šíření signálu prostředím T_{prop} tam a zpět.

Odhad vzdálenosti potom provádíme jako

$$\hat{d} = \hat{T}_{prop} \cdot c = \frac{1}{2} \left(T_{round} - T_{reply} \right) \cdot c. \tag{3.2}$$

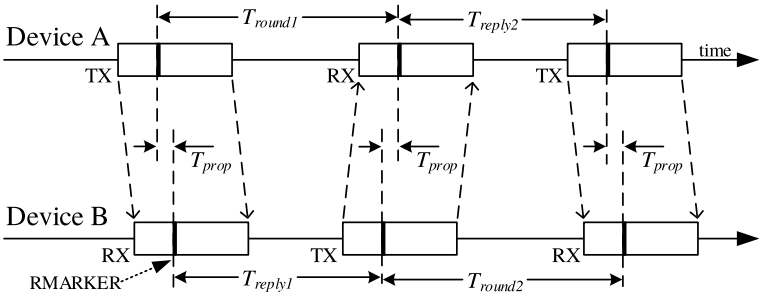
Ve vzorci je záměrně uveden odhad vzdálenosti $\hat{d}$ a odhad intervalu $\hat{T}_{prop}$. Rovnost s reálnými hodnotami zcela neplatí. Musíme si uvědomit, že časové značky o odeslání a přijetí zpráv byly vytvořeny na dvou zařízeních pomocí dvou odlišných hodin. Při prostém odečtení hodnot T_{round} a T_{reply} zanedbáváme chybu vzniklou rozdílným taktem zapojených časovačů.

Výhodou uvedené metody je její jednoduchost a minimální počet zpráv (konkrétně 2) nutných přenést za účelem provedení měření. To umožňuje dosahovat krátké odezvy, která může být v jistých aplikacích kritickým parametrem. Potřeba synchronizace hodin je odstraněna. S narůstající délkou periody T_{reply} a horší jakostí hodin však chyba měření prudce stoupá. Metoda je vhodná spíše pro měření vyšších vzdáleností s menšími nároky na přesnost.

Manuál [Dec17] uvádí typické hodnoty chyby způsobené nepřesností hodin při použití čipu DW1000. Hodnoty závisí mj. na délce vysílané zprávy, protože T_{reply} zahrnuje jak čas potřebný na zpracování požadavku na straně zařízení B, tak také trvání samotného vysílání. V kombinaci s informací, že přesnost hodin u modulů DWM1000 dosahuje (v závislosti na podmínkách) jednotek ppm [Dec16], můžeme předpokládat, že nepřesnost odhadu vzdálenosti způsobená odlišným taktem hodin s moduly DWM1000 bude dosahovat jednotek nebo nižších desítek cm. Samozřejmě se jedná pouze o jeden ze zdrojů chyb, které společně definují přesnost modulů.

Dvoustranné dvoucestné měření vzdálenosti

Chceme-li minimalizovat chybu způsobenou různým taktem hodin mezi zařízeními A a B, lze využít metod dvoustranného dvoucestného měření. V principu se jedná o jednostranné měření, iniciované nejprve zařízením A a zodopvězené zařízením B, a následně provedené v opačném směru. Manuál [Dec17] uvádí variantu se třemi nebo čtyřmi přenášenými zprávami.



Obrázek 3.4: Diagram dvoustranného dvoucestného měření vzdálenosti (*Double-sided Two-way ranging*), převzato z [Dec17]

Podíváme se blíže pouze na variantu se třemi zprávami, jelikož je výhodnější po stránce úspory času i energie. Diagram je znázorněn na obr. 3.4. Posloupnost je zahájena požadavkem o měření ze strany zařízení A. Zařízení B odpoví stejně jako v případě jednostranné metody. Tím zároveň iniciuje druhé měření a čeká na odpověď zařízení A. Oproti dvojici jednostranných měření provedených separátně dojde k optimalizaci, vystačíme si s výměnou tří zpráv namísto čtyř. Ve srovnání s jednostranným měřením jde o nárůst, ovšem potlačili jsme chybu způsobenou rozdílným taktem hodin.

Odhad času šíření signálu při dvoucestném dvoustranném měření provádíme dle výrazu

$$\hat{T}_{prop} = \frac{T_{round1} \cdot T_{round2} - T_{reply1} \cdot T_{reply2}}{T_{round1} + T_{round2} + T_{reply1} + T_{reply2}}. \tag{3.3}$$

Význam použitých časových úseků je zřejmý z obr. 3.4.

V rámci práce jsme se k měření rozhodl implementovat tuto metodu. Vývoj senzoru vzdálenosti popisuji blíže v kapitole 5.2.

3.2 Drony Tello

Pro účely testování navržených algoritmů máme k dispozici drony Ryze TELLO. Produkt byl vyvinutý ve spolupráci společností Ryze Tech a DJI. Jedná se o jednoduché kvadroptéry malých rozměrů (98 x 92,5 x 41 mm) a hmotnosti (80 g) [Ryz20]. Díky těmto vlastnostem jsou vhodné pro provoz v místnosti s minimálním rizikem poškození zdraví nebo majetku.

Helikoptéra je řízena čtrnáctijádrovým procesorem Intel, disponuje kamerou se schopností přenášet obraz v reálném čase. Za letu je automaticky stabilizována, díky osazení optickými senzory, akcelerometry a barometrem. Dokáže udržet svoji výšku a orientaci v rovině, stabilizovat náklon a fixovat polohu. Je tedy vhodná pro užití širokou veřejností bez technického povědomí.

Produkt s sebou přináší i řadu nevýhod. Výrobce nezveřejňuje detailnější specifikace produktu, dostupných funkcí či osazených senzorů. Většina informací o produktu je sdělována formou marketingových bannerů a líbivých animací bez možnost nalézt ucelený přehled vlastností produktu.

K dronům není dodáván ovladač, řízení je řešeno aplikací v mobilním telefonu, který se ke stroji připojuje přes WiFi. Zároveň existují způsoby, jak dron ovládat programově z počítače.

Drony akceptují jednoduché příkazy zaslané po síti v ASCII formě v rámci oficiálního SDK vytvořeného výrobcem. SDK je však velmi špatně dokumentované a nabízí omezené množství příkazů. Druhou cestou je ovládání pomocí neoficiálních příkazů, zveřejněných na fórech uživateli.

3.2.1 Oficiální SDK

Přes internetový vyhledávač je k dohledání PDF soubor s informacemi o SDK k dronům Tello [Ryz]. Obsahuje návod, jak dron ovládat přes programové rozhraní. Je nazván *Tello SDK 1.0.0.0*, postrádá uvedení data zveřejnění či verze dokumentu. Zpracování celého souboru působí značně neprofesionálně. Kromě toho je k nalezení též PDF soubor *SDK 2.0 User Guide* [Ryz18]. Oba dokumenty jsou sice uloženy na doméně *ryzerobotics.com*, ovšem ani na jeden ze souborů se mi nepodařilo najít odkaz z webových stránek. Jedná se tedy o PDF soubory nalezené vyhledávačem bez možnosti zjistit, zda se jedná o platné či aktuální informace.

Kromě příkazů pro vzlet, přistání, zjištění stavu baterie či aktuální rychlosti je možné v obou verzích SDK řídit polohu dronu. Příkazy *up x*, *down x*, *left x*, *right x*, *backward x* a *forward x*, kde x je hodnota v cm, zahájí posun letounu o definovanou vzdálenost. Sada v dokumentu [Ryz18] je však oproti [Ryz] bohatší. Namísto nastavení rychlosti pohybu vpřed příkazem *forward x* umožňuje let jakýmkoli směrem (*go x y z speed*). Zásadní je také příkaz *rc a b c d*, nedostupný v původní verzi SDK, který umožňuje implementovat joystick, ovládat výšku letounu a chování v jednotlivých osách (viz obr. 1.8).

Provedl jsem aktualizaci firmwaru dronu na nejnovější verzi pomocí oficiální aplikace pro mobilní telefon a pokusil se zprovoznit ovládání z počítače přes WiFi. Napsal jsem jednoduchý program v jazyce Python, který dronu odesílá příkazy dle návodu v dokumentu. Dospěl jsem k závěru, že část příkazů z novějšího SDK 2.0 není s mým dronem funkční (helikoptéra na ně nereaguje). Žádný z dokumentů bohužel neobsahuje zmínku o tom, k jaké verzi firmwaru se vztahuje. Příkazy ze starší verze SDK 1.0.0.0 odpovídaly popisu z [Ryz].

Zdrojem problémů je s největší pravděpodobností odlišnost SDK použitého v helikoptérách Tello a příbuzném modelu Tello EDU. Dle informací z internetového fóra společnosti DJI [DJI19] je jedním z mála rozdílů mezi modely právě novější verze SDK. Bohužel dokument [Ryz18] neobsahuje žádnou informaci, která by naznačovala, že se vztahuje pouze k modelu Tello EDU.

3.2.2 Neoficiální knihovny

Další možností jak ovládat drony Tello z PC je použití komunitních knihoven. Ty implementují nízkoúrovňové příkazy, které nejsou výrobcem publikované. Uživatelská komunita je postupně dekódovala ze záznamů síťové komunikace při ovládání dronu oficiální aplikací pro mobilní telefony. Soupis dekódovaných příkazů můžeme nalézt např. zde [Tel20].

Jednou z knihoven implementujících zmíněné nízkoúrovňové příkazy je PyTello [Pin]. Knihovna umožňuje simulovat joystick a řídit pohyb v jednotlivých osách z obr. 1.8 podobně jako SDK pro Tello EDU.

Cílem práce není implementovat udržování formací pouze s drony Tello. Snažíme se vyvinout co nejobecněji aplikovatelnou metodu a její funkci případně ověřit na těchto helikoptérách. Chceme proto mít možnost řídit náklon helikoptér v jednotlivých osách, jelikož se jedná o standardní způsob ovládání multikoptér. V tomto ohledu oficiální SDK nenaplňuje naše

potřeby a byla zvolena cesta využití komunitních knihoven. Většina softwaru vyvíjeného v rámci práce je psána v jazyku C++. Užívám proto knihovnu v tomto jazyce odvozenou od knihovny PyTello [Pin]. Knihovna byla poskytnuta vedoucím práce a následně upravena.

3.2.3 Připojení k síti

Při práci s drony Tello se můžeme setkat s řadou komplikací. První nastává v momentě potřeby komunikovat s více helikoptérami současně. Připojení k dronům probíhá prostřednictvím WiFi sítě, kterou vytvoří po zapnutí řídicí jednotka dronu. K síti se uživatel připojí telefonem s mobilní aplikací (případně vlastní aplikací v PC). SDK 2.0 pro Tello EDU umožňuje připojení helikoptéry do existující WiFi sítě. S dronem Tello se mi toho nepodařilo dosáhnout ani za pomoci komunitních knihoven.

Je tedy třeba pro připojení ke každému dronu použít zvláštní síťovou kartu. Při odesílání paketu nadále specifikovat, která karta má odchozí paket zpracovávat, aby byl doručen ke správnému letounu. Všechny drony používají stejnou IP adresu i port.

3.2.4 Přehřívání helikoptéry

Neprodleně po zapnutí dronu začne docházet k zahřívání procesoru. Na spodní straně helikoptéry jsou průduchy, které zajišťují chlazení. Konstrukce je však navržena tak, že k účinnému chlazení dochází pouze za letu, se zvýšeným prouděním okolního vzduchu. Necháme-li dron zapnutý ležet na zemi, dojde po jisté době k jeho přehřátí a automatickému vypnutí. Interval se při delším používání (a s tím souvisejícím zahříváním baterie) zkracuje na méně než jednu minutu.

Při snaze ovládat více dronů současně je toto chování velmi nepraktické. V průběhu testování a vývoje dojde k přehřátí a vypnutí některých helikoptér. Po jejich opětovném zapnutí je nutné vyčkat spuštění firmwaru a obnovit připojení k WiFi síti. Zde vyzývám k obezřetnosti, může dojít k samovolnému zapnutí dronu při nabíjení. Je nutné ohlídat, zda se WiFi karta připojila ke správnému letounu, jinak může dojít k poškození vrtulí či napájecího kabelu.

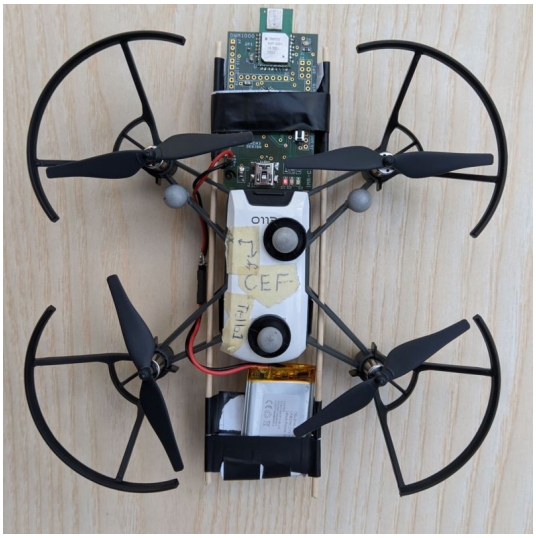
V průběhu připojování k dronu se zpravidla přehřejí a vypnou zbylé helikoptéry. Na provedení sekvence potřebných úkonů má uživatel jen pár pokusů, než dojde k vybití baterie některého z dronů. Zdržení spojené s výměnou baterie samozřejmě zapříčiní přehřátí všech zúčastněných helikoptér. Už práce se dvěma drony Tello současně vyžaduje velkou dávku trpělivosti ze strany uživatele.

3.2.5 Upevnění dodatečných senzorů

Z podstaty této práce bylo potřeba na helikoptéry umístit senzor pro měření vzájemných vzdáleností. Vzhledem k nízké hmotnosti dronů Tello se sebemenší zatížení projevuje na stabilitě a výdrži baterie. Vhodné umístění senzoru a LiPol akumulátoru zajišťujícího jeho napájení je klíčové pro provedení experimentů.

Posun těžiště by měl negativní vliv na stabilitu letounu. Při upevňování senzoru k tělu helikoptéry je proto žádoucí dosáhnout symetrického zatížení vůči těžišti. Zároveň je snahou minimalizovat vzdálenost od těžiště a nezvyšovat moment setrvačnosti letounu, který nepříznivě ovlivňuje dynamiku řízení.

Významný vliv na výkon, ovladatelnost, stabilitu i spotřebu energie má také vytvoření



Obrázek 3.5: Umístění senzoru vzdálenosti a LiPol článku na helikoptéru Tello

překážek proudícímu vzduchu. Senzor ani akumulátor by neměl výrazně zasahovat do oblasti pod (ani nad) vrtulemi. Nevhodné je též zakrytí chladicích průduchů na spodní straně dronu. Navíc musíme zachovat funkci optických senzorů sloužících ke stabilizaci letu a měření výšky umístěných vedle průduchů.

Po vyhodnocení všech požadavků a možností byl zvolen způsob ukotvení zaznamenaný na obr. 3.5. Podařilo se tak dosáhnout návrhu použitelného pro testování navržených algoritmů. Na přední části letounu je zavěšen LiPol článek určený k napájení senzoru v přední části helikoptéry. I při tomto způsobu ukotvení je však vliv na stabilitu helikoptér znatelný.

Zásadní je poznatek, že schopnost řídicí jednotky udržovat v klidovém stavu konstantní polohu, výšku a náklon helikoptéry se lišila mezi jednotlivými kusy stejného modelu dronu. Problém nevyřešilo ani provedení kalibrace akcelerometrů dle návodu výrobce. Důvodem mohly být nepatrné odlišnosti v konstrukci pro uchycení senzorů, které byly vyráběny ručně. Každopádně jen část z dostupných helikoptér zůstala použitelná pro let s osazeným senzorem vzdálenosti. U zbývající části jsou po vzletu a pokusu zůstat na místě (bez zasílání jakýchkoli řídicích příkazů) zaznamenány náhlé změny výšky i polohy, případně samovolné přistání letounu.

3.3 Lokalizační systém Vicon

K získání referenčních dat o poloze zkoumaných objektů při provedených testech byl použit lokalizační systém Vicon. Jedná se o sadu IR kamer umístěných v laboratoři Inteligentní a mobilní robotiky ČVUT CIIRC.

Sledovaný prostor je nasvícen zdrojem IR záření. Na pohybujících se objektech jsou umístěny kuličky s vysoce odrazivým povrchem v (IR spektru). Kamery pokrývají celou sledovanou oblast a zachycují záření odražené od kuliček. Při viditelnosti jednoho bodu v prostoru několika kamerami z různých směrů je systém schopen dopočítat polohu objektu v prostoru.

Některé helikoptéry byly (kromě vlastích senzorů pro měření vzájemné vzdálenosti s modulem DWM1000) osazeny také kuličkami (*markery*) systému Vicon, viz obr. 3.6. To

Obrázek 3.6: Umístění markerů (reflexních kuliček) systému Vicon na helikoptéru Tello

umožňuje lépe vyhodnotit výsledky některých experimentů nebo použít referenční měření pro účely kalibrace námi navrženého systému.

Kapitola 4

Provedené simulace

Před samotnou implementací jsem provedl v jazyku Python simulaci metod, se kterými jsme se seznámili v teoretické části práce. Konkrétně jde o metodu pro udržování trojúhelníkové formace (popsanou v kapitole 1.3) a o systém lokalizace s použitím pevných kotev a EKF (popsanou v kapitole 2.3.1).

Simulace druhé z metod nepřinesla překvapivá zjištění a vyústila v reálnou implementaci, jejíž popis i výsledky si uvedeme později. Simulací samotnou se zde proto již nezabývám. V případě trojúhelníkové formace však muselo dojít k dalším úpravám, jelikož se nedařilo ověřit funkčnost odhadu směřování formace dle původního návrhu.

4.1 Udržování trojúhelníkové formace

Nyní se podíváme na konkrétní způsob implementace metody pro udržování trojúhelníkové formace založené na článku [SAPG12]. Princip metody je blíže popsán v kapitole 1.3.

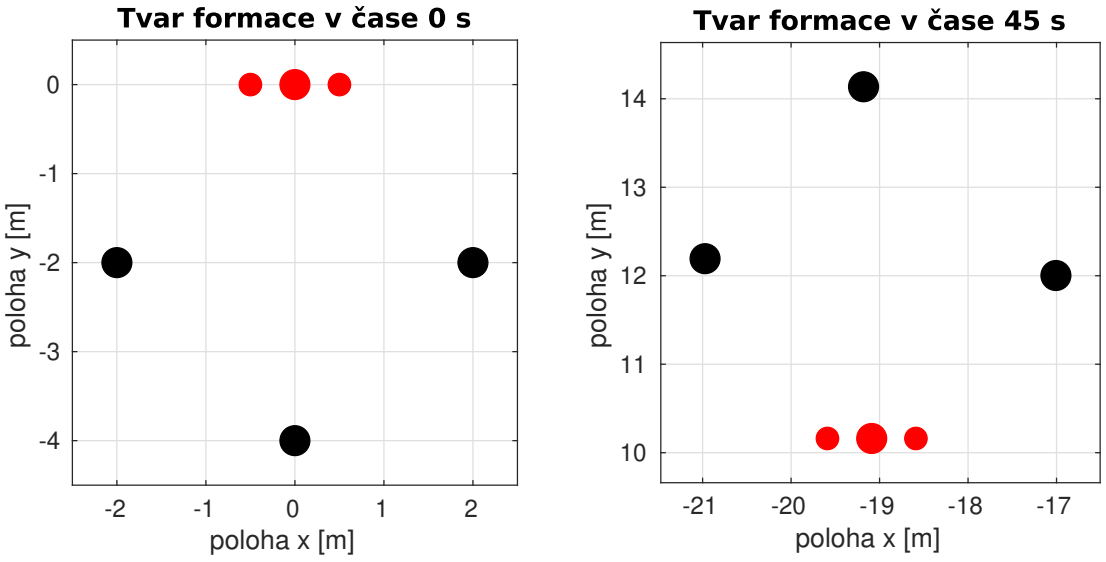
Funkčnost metody jsme se pokusili ověřit v simulaci s užitím programovacího jazyka Python. Jednotliví agenti jsou pro jednoduchost reprezentováni pouze nehmotnými body v rovině. Dynamický model helikoptér, na kterých plánujeme metodu vyzkoušet, nemáme k dispozici. Podaří-li se ověřit stabilitu řídicího algoritmu ve fázi simulací, bude algoritmus následně testován na reálných strojích.

Do simulace byly zařazeny některé modifikace navrhované v kapitole 1.3. Byl použit pouze jeden vůdčí agent se dvěma simulovanými rádiovými moduly a rovnou tři agenti sledující, jak naznačuje obr. 1.6b. Každý agent i je objektem s definovanou polohou $x_i \in \Re^2$, skalární rychlostí v_i a orientací v rovině ψ_i. Vůdčí agenti se pohybují po předem definované trajektorii, sledující agenti aktualizují svou rychlost a orientaci na základě naměřené vzdálenosti k vůdcům. Pohyb letounu do stran (řízením osy *roll* dle obr. 1.8) do simulace zahrnut nebyl.

Vůdčí agent simuluje osazení dvou rádiových modulů s definovaným rozpětím (vzájemnou vzdáleností) w. Spojnice modulů je vždy kolmá ke směru pohybu ψ_i, se středem v lokaci vůdce x_i. Polohu rádiových modulů p_1, p_2 v čase t získáme jako

$$\begin{aligned} p_1(t) &= x_i(t) + \frac{w}{2} \cdot \left[\cos\left(\psi_i(t) - \frac{\pi}{2}\right) \quad \sin\left(\psi_i(t) - \frac{\pi}{2}\right)\right] \\ p_2(t) &= x_i(t) + \frac{w}{2} \cdot \left[\cos\left(\psi_i(t) + \frac{\pi}{2}\right) \quad \sin\left(\psi_i(t) + \frac{\pi}{2}\right)\right]. \end{aligned} \tag{4.1}$$

Tvar formace je definován referenčními vzdálenostmi d_{i1} a d_{i2}, trojúhelník nemusí být rovnoramenný.



(a) : Požadovaný (výchozí) tvar formace **(b) :** Tvar formace v průběhu pohybu

Obrázek 4.1: Tvar široké trojúhelníkové formace při simulaci

Sledující agenti ve smyčce proměřují vzdálenosti z_{i1}, z_{i2} mezi vlastní polohou x_i a dvojicí rádiových modulů p_1, p_2. Platí tedy

$$\begin{aligned} z_{i1} &= ||x_i - p_1||\,, \\ z_{i2} &= ||x_i - p_2||\,. \end{aligned} \tag{4.2}$$

Na základě znalosti z_{i1}, z_{i2} potom řídí svoje stavové veličiny v_i a ψ_i tak, aby se přiblížili ideálnímu stavu, kdy

$$\begin{aligned} z_{i1} &= d_{i1}, \\ z_{i2} &= d_{i2}. \end{aligned} \tag{4.3}$$

Nejprve je spočtena souhlasná složka chyby

$$\epsilon_i = (z_{i1} + z_{i2}) - (d_{i1} + d_{i2}) \tag{4.4}$$

a rozdílová složka chyby

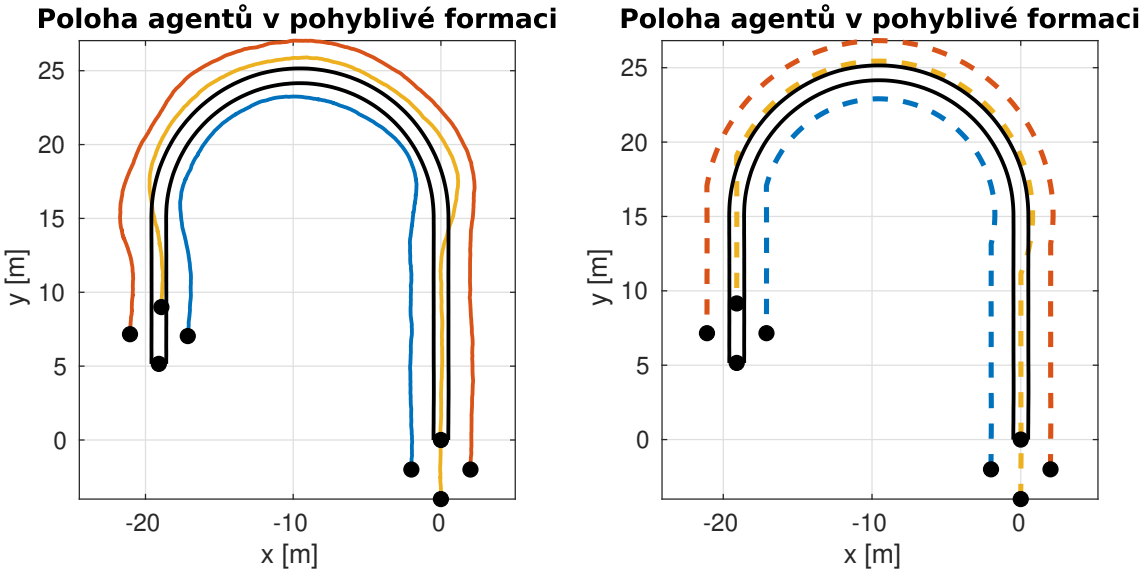
$$\delta_i = (z_{i1} - d_{i1}) - (z_{i2} - d_{i2}). \tag{4.5}$$

Postup regulace rychlosti v_i a směřování ψ_i se řídí modifikovanými rovnicemi (1.15) a (1.18). Pro každého agenta jsou obě veličiny regulovány zvlášť.

Připomínám, že koncepčně je systém navržen k přesunu formace rovnoměrným přímočarým pohybem a změny směru i rychlosti tedy musejí probíhat dostatečně pomalu.

4.1.1 Simulace pohybu široké formace

Provedl jsem testy s formací vycházející z obr. 1.6b. Jeden vůdce je následován třemi agenty. Rozpětí rádiových modulů na vůdci je nastaveno na hodnotu $w = 1\,\mathrm{m}$ a v čase $t = 0\,\mathrm{s}$ se vůdce nachází v počátku souřadného systému. Přesné rozmístění agentů na počátku simulace znázorňuje obr. 4.1a (červeně znázorněn vůdce se dvěma rádiovými moduly). Záměrně byla



(a) : Výsledky simulace **(b) :** Referenční hodnoty

Obrázek 4.2: Trajektorie široké trojúhelníkové formace při simulaci

zvolena poměrně velká šířka formace (4 m) vůči rozpětí rádiových modulů na vůdci (1 m), abychom ověřili stabilitu algoritmu za těchto podmínek.

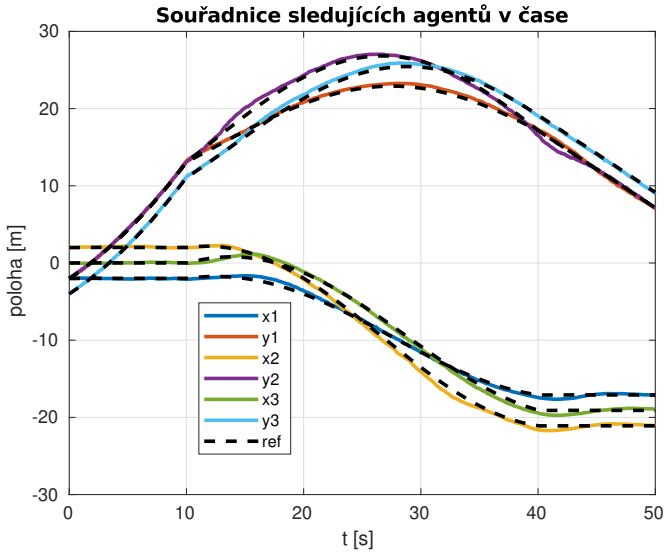
Vzdálenost mezi krajními sledujícími agenty je větší než hodnota w. Při jejich vychýlení do strany dochází pouze k malé změně rozdílové složky chyby δ. V ideálním případě (je-li sledující agent umístěn mezi vůdci) by měla vzdálenost od jednoho senzoru vůdce klesat a od druhého stoupat. Například při vychýlení levého sledujícího agenta ještě více doleva jeho vzdálenost od obou vůdčích senzorů naroste, což je nepříznivé. Naštěstí roste rychleji vzdálenost od agenta pravého a parametr δ nabývá správného znaménka potřebného k regulaci, pouze s menší hodnotou. Tento fakt by šel dokázat derivací funkce $\epsilon(x, y)$ ve směru vychýlení, ale z obrázku 4.1a je to patrné.

Simulace zahrnuje uměle přidaný šum normálního rozdělení o střední hodnotě $\mu = 0$. Agenti (včetně vůdců) provádí každou změnu polohy s aditivní náhodnou složkou o rozptylu $\sigma^2 = 1\,\text{cm}$. Měření vzdálenosti (simulace rádiových modulů) je zatíženo šumem o rozptylu $\sigma^2 = 5\,\text{cm}$. Vzorkovací perioda simulace je $T_s = 0,1\,\text{s}$. V průběhu jedné simulační smyčky dojde k

- pohybu vůdce předem definovanou rychlostí v_i ve směru ψ_i,
- měření vzdálenosti od každého sledujícího agenta ke dvojici rádiových modulů vůdce,
- provedení jednoho kroku regulace v_i a ψ_i sledujících agentů,
- pohybu sledujících agentů nově spočtenou rychlostí v_i ve směru ψ_i.

Formace se pohybovala po trajektorii písmene U, podobně jako v simulaci zveřejněné ve článku [SAPG12]. Dráha uražená agenty je zaznamenána na obr. 4.2a (černě vyznačena dráha dvou rádiových modulů vůdce). Na první pohled nás může zarazit, že žlutá linie jednoho ze sledujících agentů kříží černou linii vůdce. Referenční obrázek 4.2b však potvrzuje, že toto

chování je validní. Sledující agent na zatočení vůdce doleva reaguje prvotně otočením doprava, jelikož musí zajistit otočení celé formace v rovině se zachováním původního tvaru.



Obrázek 4.3: Souřadnice jednotlivých agentů v čase při simulaci široké trojúhelníkové formace

Obr. 4.3 obsahuje průběh souřadnic jednotlivých agentů v čase. Černou čarou je vyznačena referenční hodnota. Na obr. 4.1b následně vidíme, jak se daří držet tvar formace při pohybu vůdců.

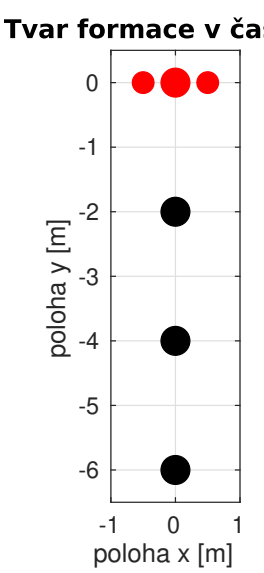
4.1.2 Simulace pohybu dlouhé formace

V dalším kroku byla simulace zopakována s formací ve tvaru dle obr. 4.4a. Tvar byl zvolen jako protipól předchozího rozložení. Sledující agenti se v tomto případě nedostávají mimo prostor vymezený rozpětím vůdce. To zlepšuje konvergenci regulace. Na druhou stranu, vyšší vzdálenost posledního agenta od vůdce vyžaduje při změně směru opis oblouku s větším poloměrem, v důsledku toho i dosažení vyšší rychlost pohybu. Z obrázků 4.6 a 4.4b je patrné, že agent x_3, vzdálený nejvíce od vůdce, dosahuje nejvyšší odchylky od požadované polohy. Obr. 4.5 obsahuje záznam referenční a simulované polohy agentů v rovině.

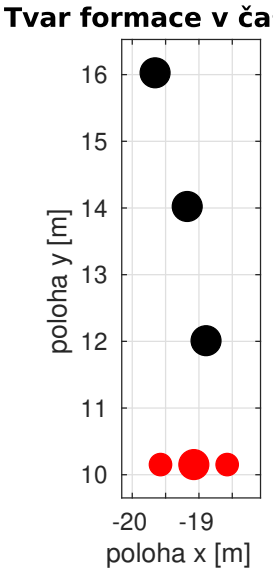
4.1.3 Výsledek simulací

Byla provedena řada dalších simulací, v práci uvádím jen zajímavé extrémní příklady. Celkově se však potvrzují naše očekávání, že nejstabilnější formace je taková, kdy vůdce dosahuje dostatečného rozpětí, sledující agenti od něj nejsou příliš vzdáleni a jejich požadovaná poloha není vychýlena od středu.

Ukázali jsme však, že i sporná uspořádání (široká formace a dlouhá formace) jsou v simulaci schopna provozu. V praxi budeme tvar formace navrhovat tak, abychom se přiblížili ideálnímu formátu. Při rozmísťování agentů nesmíme zapomenout na zajištění dostatečné vzájemné vzdálenosti, regulační algoritmus žádným způsobem nezabraňuje možným kolizím.

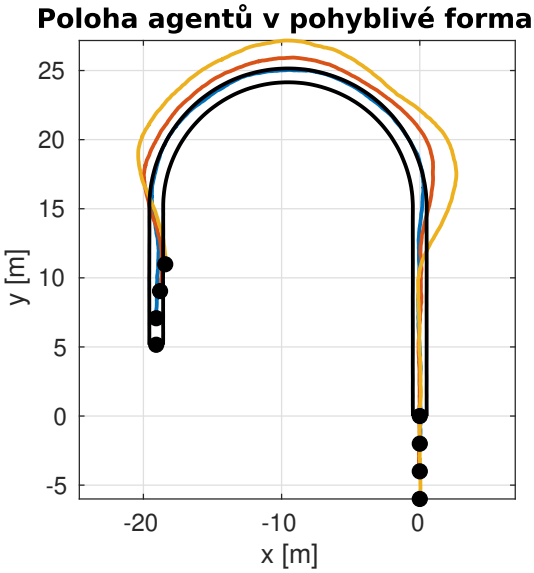


(a) : Požadovaný (výchozí) tvar formace

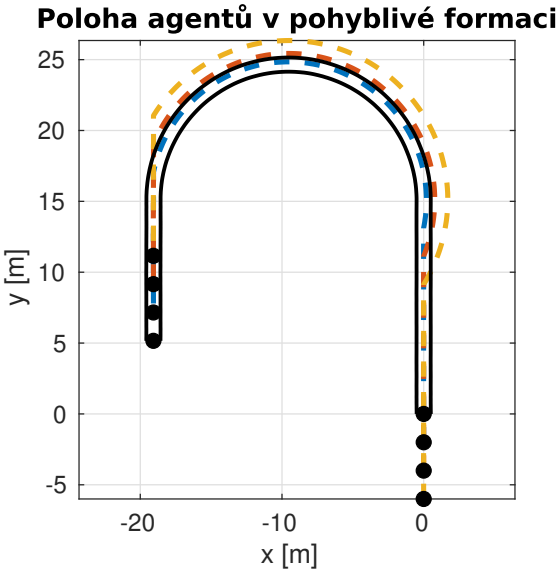


(b) : Tvar v průběhu simulace

Obrázek 4.4: Tvar dlouhé trojúhelníkové formace při simulaci

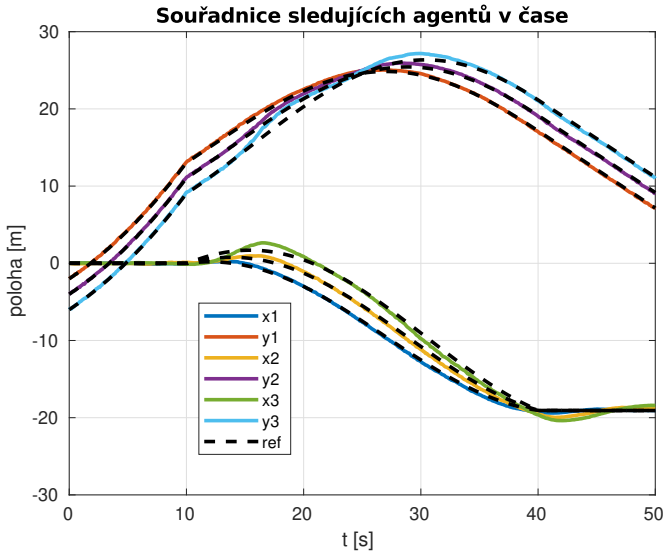


(a) : Výsledky simulace



(b) : Referenční hodnoty

Obrázek 4.5: Trajektorie dlouhé trojúhelníkové formace při simulaci



Obrázek 4.6: Souřadnice jednotlivých agentů v čase při simulaci dlouhé trojúhelníkové formace

Kapitola 5

Implementace

5.1 Architektura systému

V rámci plnění úkolu pracujeme s několika funkčními celky. Metody pro řízení formací a lokalizaci agentů popsané v teoretické části chceme otestovat na reálných modelech helikoptér. Je nutné navrhnout infrastrukturu, která zajistí propojení všech součástí systému.

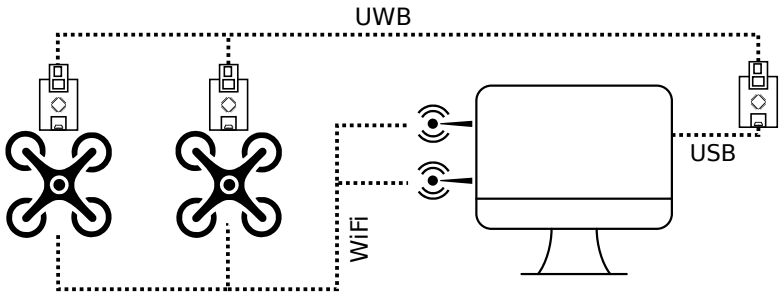
K měření vzdálenosti používáme moduly DWM1000. Ty však samy o sobě touto funkcí nedisponují. Senzor vzdálenosti bylo potřeba navrhnout a naprogramovat, výsledky měření zprostředkovat dalším částem systému.

Drony Ryze Tello nejsou osazeny programovatelnou výpočetní jednotkou. Pro jejich řízení a regulaci je třeba provádět výpočty na centrálním stolním počítači, simulovat dálkové ovládání a dronům zasílat pouze základní letové instrukce. Centrální počítač tedy musí komunikovat se všemi senzory vzdálenosti a získávat od nich výsledky měření.

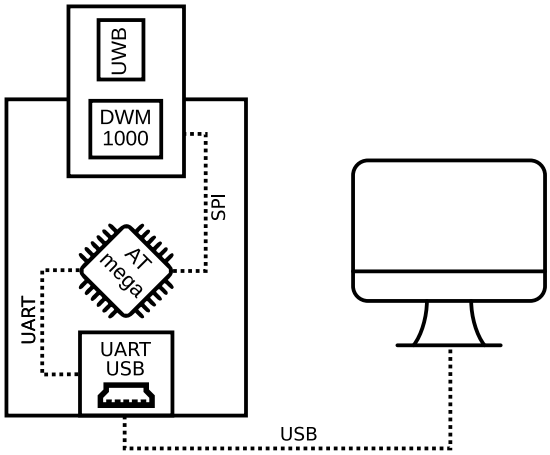
Schéma popisující architekturu systému můžeme vidět na obr. 5.1. Centrální řídicí PC je přes USB připojeno k jednomu ze senzorů vzdálenosti. Ten může dle potřeby přímo plnit funkci senzoru, nebo pouze sloužit ke komunikaci s ostatními senzory přes protokol UWB. Modul vysílá do bezdrátové senzorové sítě požadavky na měření a výsledky předává zpět centrální řídicí jednotce.

K řídicímu PC jsou nadále připojeny síťové WiFi karty sloužící ke komunikaci s drony. Vzhledem ke způsobu navazování spojení mezi počítačem a drony je třeba zvláštní síťové karty pro každý letoun.

Každý z dronů je osazen senzorem vzdálenosti s modulem DWM1000. Nabízí se otázka, proč drony nevyužívají ke vzájemné komunikaci protokol UWB. Hlavním důvodem je nemožnost připojit senzor k řídicí jednotce dronu a zpracovávat data přímo na palubě. K řízení helikoptér je vždy zapotřebí centrální počítač a většina datové komunikace proto proudí přes něj. Síť UWB slouží čistě k měření vzdáleností a zajišťuje režii pro nastavení senzorů a předání výsledku do PC.



Obrázek 5.1: Architektura systému propojující drony Ryze Tello, centrální řídicí PC a senzory vzdálenosti s DWM1000.



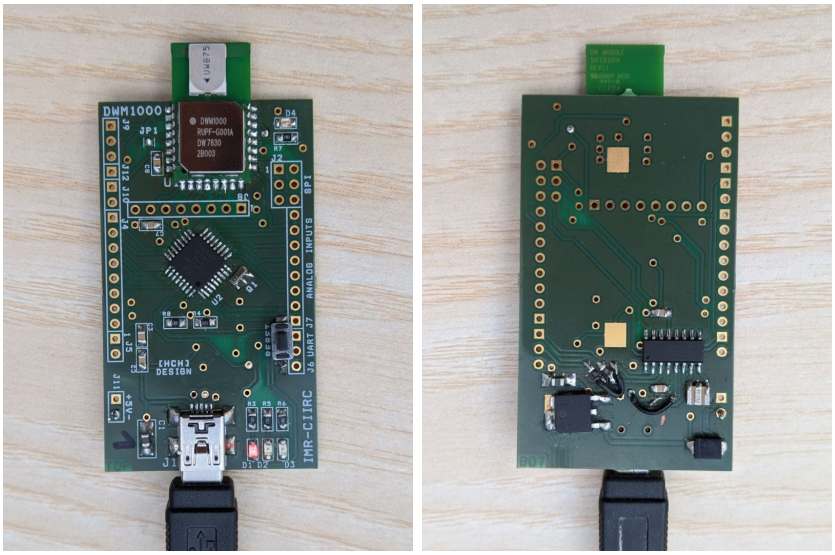
Obrázek 5.2: Schéma senzoru vzdálenosti s modulem DWM1000

5.2 Senzor vzdálenosti

Jak již bylo zmíněno mnohokrát, senzor vzdálenosti je postaven na modulech DWM1000. Bezdrátový protokol UWB splňuje standard IEEE802.15.4-2011. Modul je schopen vytvářet přesné časové značky v momentě odeslání a přijetí zprávy, čehož využíváme k měření doby šíření signálu prostředím.

Na obr. 5.2 je blokové schéma použitého senzoru. (Deska plošných spojů mi byla poskytnuta vedoucím práce, mým úkolem je tvorba firmwaru.) Skládá se ze samotného modulu DWM1000 a mikroprocesoru ATmega. Mikroprocesor komunikuje s modulem přes rozhraní SPI, připravuje obsah zpráv odesílaných přes UWB a zpracovává přijaté zprávy z jiných modulů. Zároveň prostřednictvím převodníku UART/USB na desce senzoru zprostředkovává komunikaci s PC. Přijímá z něj pokyny a odesílá zpět výsledky měření. Fyzické vyhotovení senzoru si můžeme prohlédnout na obr. 5.3.

Komunikační protokol modulu DWM1000 přes rozhraní SPI je popsán v dokumentu [Dec17]. Každá zpráva obsahuje adresu registru, ve kterém chceme provést operaci čtení nebo zápisu. Veškerá konfigurace modulu pak probíhá nastavováním příslušných registrů, obsah přijatých zpráv se získává čtením registrů. Pro příklad, při odeslání zprávy je nejprve nutné zapsat data do *Transmit Data Buffer* registru a nastavit parametry přenosu (jako data rate či délku preambule) v *Transmit Frame Control* registru. Vysílání zahájíme nastavením bitu *TXSTRT* v *System Control* registru. Modul automaticky odešle všechny části datového rámce a dokončení zahlásí změnou hodnoty *TXFRS* bitu v *System Event Status* registru.



(a) : Přední strana **(b) :** Zadní strana

Obrázek 5.3: Realizace senzoru vzdálenosti s modulem DWM1000

Většina základních operací s modulem DWM1000 je implementována v Arduino knihovně DW1000-ng [F-A21]. Jelikož naše deska je postavena na stejném mikroprocesoru jako vývojový kit Arduino nano, jsme schopni knihovnu nahrát na náš senzor a využít. Rutinní činnosti jako odeslání zprávy a vyčítání časových značek tak můžeme provádět pohodlně z vyšší vrstvy programu.

Knihovnu je možno nahrát na mikrokontrolér pomocí Arduino IDE. Celý firmware senzoru proto vyvíjím v tomto prostředí. Nutné je na procesor ATmega nejprve flashnout Arduino bootloader, aby dokázal s IDE komunikovat přes USB. Flashování bootloaderu lze provést pomocí specializovaného programátoru. Obejít se lze i bez něj, programátor dokážeme nasimulovat jakýmkoli jiným Arduinem. Pomůže k tomu skript Atmega bootloader programmer [Nic12]. Arduino v roli programátoru propojíme přes piny SPI s piny ISP (*In-system programming*) na desce senzoru. Je však nutné použít převodníku napěťových úrovní (nebo alespoň odporového děliče). Zatímco standardní Arduino desky běží na napájení 5 V, senzor s DWM1000 je připojen na regulátor 3,3 V. V případě připojení Arduina k ATmega procesoru s nižším napájecím napětím může dojít k jeho trvalému poškození.

Firmware senzoru vzdálenosti běží na osazeném procesu ATmega. Zajišťuje komunikaci mezi PC a modulem DWM1000 a reaguje na příchozí zprávy vzdušnou i metalickou cestou. K jeho naprogramování jsem využil knihovnu DW1000-ng [F-A21].

V rámci knihovny DW1000-ng jsou implementována dvě hw přerušení (*interrupty*) reagující na události dokončení bezdrátového přenosu (vysílání a příjmu). V rámci firmwaru poté stačí kontrolovat hodnoty příslušných proměnných. Tím si udržíme přehled o stavu bezdrátové komunikace bez nutnosti periodicky odesílat dotazy na modul DWM1000 přes SPI.

5.2.1 Základní smyčka programu

Základní programová smyčka je navržena tak, aby perioda jejího vykonání byla co nejkratší. Pomalý běh firmwaru by měl za následek opožděné reakce na požadavky o měření vzdálenosti a brzdil by všechny navazující procesy. Smyčka sestává jen ze dvou základních kroků. Kontroly

přijetí bezdrátových zpráv přes UWB a kontroly dostupnosti příkazů z PC přes UART (USB). Pokud dojde k detekci jedné z událostí, jsou volány funkce k dekódování přijatých zpráv.

Senzor se může nacházet v jednom ze tří módů. Po spuštění je automaticky zahájen provoz v módu RESPONDER, který umožňuje reagovat na požadavky k měření (průběh hlavní programové smyčky odpovídá popisu výše). Kromě toho existuje mód NOTASSIGNED. V něm se senzor chová stejně, ale na požadavky k měření nereaguje. A také mód INITIATOR. Nachází-li se senzor v tomto módu, vykonává na začátku hlavní smyčky navíc jeden krok. Rozešle postupně požadavky k měření (příkazy POLL) na adresy všech senzorů v režimu RESPONDER. Přepínání mezi módy se provádí zasláním příkazu přes USB nebo UWB. Seznam adres ostatních modulů je též třeba nastavit příkazem.

Po spuštění senzoru (připojení napájení nebo restartu) je vykonána funkce setup(). Jejím úkolem je uvést modul DWM100 do základní konfigurace a připravit ho k bezdrátovému přenosu. Dále jsou iniciovány hw interrupty pro dokončení přenosu, senzor je uveden do režimu RESPONDER a dojde k vyčtení adresy uložené v EEPROM paměti každého senzoru. Adresa slouží k jednoznačné identifikaci konkrétního senzoru v rámci sítě a umožňuje zasílání adresovaných zpráv.

5.2.2 Komunikační protokol

Senzor udržuje dva komunikační kanály. Čeká na příkazy zaslané přes USB, pokud je připojen k řídicímu PC. V lokalizační síti zpravidla stačí mít k PC připojen jeden senzor. Druhým kanálem je bezdrátová UWB komunikace, díky které jsme ve spojení s ostatními senzory. Definoval jsem jednoduchý komunikační protokol použitelný pro drátovou i bezdrátovou komunikaci. Standardní délka příkazu je 6 bytů. Každá zpráva začíná bytem kódujícím typ příkazu (ID). Dle přijatého ID je zpracován zbytek datového rámce.

Nejprve se podíváme na kategorii příkazů využívaných čistě v bezdrátové UWB komunikaci mezi senzory. Jejich přehled nabízí tabulka 5.1. Sloupce reprezentují datové byty zprávy, ID příkazu na pozici 0 je ve skutečnosti kódováno číslem.

byte ID příkazu	**0**	**1**	**2**	**3**	**4**	**5**	**6**
POLL		-	-	-	-	-	-
POLL_ACK		-	-	-	-	-	-
RANGE		-	-	-	-	-	-
RANGE_FAILED		-	-	-	-	-	-
WIRELES_CMD		počet	-	-	-	-	-
WIRELES_CMD_ACK		ID	-	-	-	-	-

Tabulka 5.1: Protokol bezdrátové komunikace mezi senzory

POLL. Je jedním z příkazů, na který reaguje senzor v režimu RESPONDER přímo ze základní programové smyčky. Slouží k zahájení procesu měření (o něm více v kapitole 5.2.3). Většina ostatních typů zpráv je ze základní smyčky ignorována a senzor je přijme jen ve stavu, kdy jsou očekávány.

POLL_ACK, RANGE, RANGE_FAILED. Tyto příkazy jsou vyměňovány mezi senzory v rámci procesu měření (o něm více v kapitole 5.2.3). Slouží k tvorbě časových značek a reportování chyb procesu.

WIRELESS_CMD. Je druhým příkazem, který senzor ve výchozím stavu přijímá. Jeho hlavním účelem je umožnit konfiguraci senzorů, které nejsou připojeny k řídicímu PC. Potřebujeme-li např. nastavit na všech senzorech adresu pro reportování výsledků měření, rozešleme ze senzoru připojeného k PC příkaz WIRELESS_CMD do ostatních senzorů. Za ním, v rámci jedné zprávy, následuje jeden či více příkazů sloužících primárně k ovládání senzoru z PC. Jejich přehled nalezneme v tabulce 5.2.

WIRELESS_CMD_ACK. Potvrzuje přijetí příkazu bezdrátovým kanálem od jiného senzoru. Ten na základě potvrzení ukončí proces opakovaného zasílání. Zpráva obsahuje na druhé pozici hodnotu ID. Jedná se o číslo přijaté zprávy, nikoli o značení typu příkazu.

Dále jsou definovány příkazy sloužící k interakci senzoru a PC. V případě potřeby je lze odeslat přes UWB za pomoci příkazu WIRELESS_CMD. Jejich přehled najdeme v tabulce 5.2.

byte ID příkazu	**0**	**1**	**2**	**3**	**4**	**5**	**6**
SET_ROLE		nová role	-	-	-	-	-
RESPONDERS_LIST		počet	adr 1.	adr 2.	...	...	...
RANGING_PERIOD		perioda (uint16)		-	-	-	-
SEND_WIRELESS_CMD		adresa	počet	pokusy	-	-	-
RANGE_REPORT_ADR		adresa	-	-	-	-	-
PERFORM_RANGING		adresa	-	-	-	-	-
MAKE_REPORT		adr. 1	adr. 2	vzdálenost (float)			

Tabulka 5.2: Protokol komunikace senzorů vzdálenosti s PC

SET_ROLE. Slouží ke změně role senzoru. Přepínat lze mezi hodnotami NOTASSIGNED, INITIATOR a RESPONDER. INITIATOR v pravidelných intervalech provádí měření vzdálenosti k senzorům v režimu RESPONDER. V systému nemusí být žádný senzor v režimu INITIATOR, měření můžeme kontrolovat a vyvolat v daný okamžik pomocí příkazu z PC.

RESPONDERS_LIST. Definuje seznam senzorů, které má INITIATOR v pravidelných intervalech proměřovat. Zpráva obsahuje počet adres, následuje jejich výčet.

RANGING_PERIOD. Stanovuje periodu v ms, s jakou má probíhat proměřování senzorem v režimu INITIATOR.

SEND_WIRELESS_CMD. Příkaz je posílán z PC přes USB, pokud potřebujeme zaslat libovolný jiný příkaz na modul nepřipojený k PC. Obsahuje adresu senzoru, který chceme kontaktovat, počet příkazů a počet pokusů. Za příkazem pak musí následovat konkrétní příkazy, které chceme přidat. Senzor připojený k PC na základě tohoto pokynu vyšle WIRELESS_CMD na danou adresu. Opakuje vysílání dle hodnoty *pokusy*, pokud není doručeno potvrzení o přijetí.

RANGE_REPORT_ADR. Definuje adresu senzoru, který sbírá výsledky všech měření v síti. Zpravidla se jedná o senzor připojený k PC, který komunikuje s lokalizačním softwarem.

PERFORM_RANGING. Příkaz zajistí vyslání požadavku POLL na zadanou adresu a tím zahájí proces měření mezi dvěma senzory. Je zasílán přes USB. Chceme-li z PC zahájit měření mezi jinými dvěma senzory, je třeba odeslat tento příkaz prostřednictvím WIRELESS_CMD jednomu z účastníků měření.

MAKE_REPORT. Příkaz obsahuje výsledek měření vzdálenosti mezi dvěma senzory. Je zasílán zpravidla senzoru připojenému k PC, který po přijetí předá data do řídicího počítače přes USB.

5.2.3 Proces měření vzdálenosti

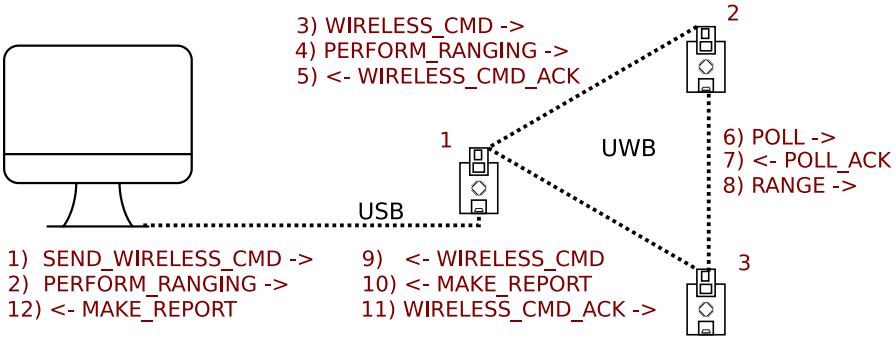
Proces měření vzdálenosti může být vyvolán senzorem v režimu INITIATOR, který v pravidelných intervalech měří vzdálenost k definovanému seznamu dalších senzorů. Zahájit ho lze také jednorázově zasláním příkazu PERFORM_RANGING. Oba zúčastněné senzory vstupují do procesu měření a ve firmwaru je spuštěna příslušná funkce request_ranging() či answear_ranging(). Od této chvíle až do ukončení měření nemůže senzor zpracovávat jiné požadavky. Nereaguje na příkazy zaslané přes USB ani UWB, vyhodnotí je až po dokončení procesu. Měření spočívá v odeslání a přijetí několika zpráv v definovaném pořadí se zaznamenáváním příslušných časových značek. Končí odesláním výsledku, nebo vypršením časového limitu v případě, že druhá strana přestane reagovat.

Jeden ze senzorů přejde do funkce request_ranging() a na adresu druhého senzoru vyšle příkaz POLL. Následně je spuštěn while cyklus, který po celou dobu měření čeká na dokončení operací odesílání a příjmu zpráv, případně vypršení časového limitu. V úspěšném procesu měření je očekávána následující sekvence událostí: Dojde k potvrzení odeslání příkazu POLL, aktivaci antény a přijetí zprávy POLL_ACK. Senzor zaznamená časové značky obou událostí a nastaví zpožděné odeslání poslední (třetí) zprávy RANGE. Před tím, než data opustí anténu modulu DWM1000, uloží do datového rámce oba zaznamenané časy, a navíc i čas budoucího odeslání této zprávy (typ RANGE).

Druhý senzor po přijetí zprávy POLL aktivuje funkci answear_ranging(). Stejným způsobem jako první senzor kontroluje průběh měření a zaznamenává časové značky událostí. Odešle zprávu POLL_ACK a vyčká doručení zprávy RANGE. Po jejím dekódování má k dispozici časy odeslání a doručení všech tří zpráv, celkem tedy šest časových značek. Sekvence událostí odpovídá průběhu časového diagramu na obr. 3.4. Vzdálenost mezi senzory je odhadnuta výpočtem dle rovnice (3.3).

Senzor odpovídající na měření předá výsledek na adresu nastavenou příkazem RANGE_REPORT_ADR. Je-li shodná s jeho vlastní adresou načtenou z EEPROM, rovnou odešle údaj přes USB do připojeného PC. V opačném případě pomocí zprávy WIRELESS_COMMAND odešle příkaz MAKE_REPORT na danou adresu. Senzor připojený k PC ji vyhodnotí a výsledky měření předá.

Na obr. 5.4 je vyobrazena posloupnost příkazů putujících mezi účastníky při jednoduchém požadavku na měření. Řídicí PC požaduje změření vzdálenosti od senzoru 2 k senzoru 3. K počítači je přes USB připojen senzor 1, který figuruje v roli prostředníka mezi senzorovou sítí a počítačem. Čísla před každým příkazem určují pořadí jejich odeslání. Ukazuje se, že



Obrázek 5.4: Posloupnost příkazů při měření vzdálenosti mezi dvěma senzory

naprosto základní operace v senzorové síti pod sebou skrývá poměrně komplikovanou operaci. Dojde při ní mezi aktéry k výměně 12 příkazů zaslaných v o něco menším počtu zpráv (WIRELESS_CMD agreguje více příkazu do jednoho datového rámce).

V kapitole 5.2.6 se dozvíme, že odeslání zprávy přes UWB není instantní záležitostí. Řádově můžeme mluvit o čase 1 ms potřebném k tomu, aby jeden datový rámec opustil anténu modulu. Nyní si můžeme udělat lepší představu o možnostech, které nám UWB v lokalizaci nabízí. Dostáváme se k periodě řádově 10 ms pro zjištění jedné vzdálenosti v systému. Pro určení polohy agenta je samozřejmě vyžadováno nejedno měření popsané v obr. 5.4. Z toho plyne, že požadavek na rozšíření systému o každého jednoho dalšího agenta bude mít významný vliv na odezvu měření i řízení formace.

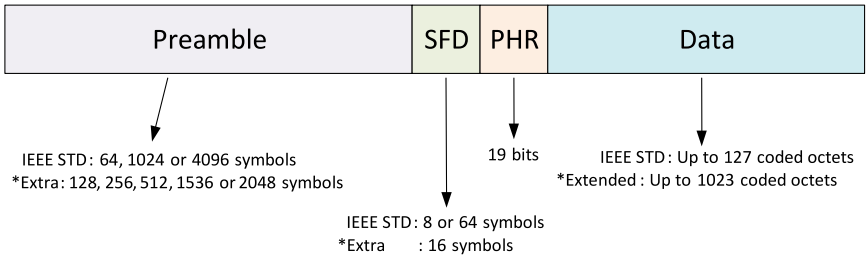
5.2.4 Adresování zpráv

Moduly DWM1000 nabízí možnost *MAC Frame Filtering*, tedy filtrování přijatých zpráv na základě typu zprávy a adresáta. Modul v principu zachytí všechny zprávy vysílané v dosahu jeho antény. Chceme-li, aby zpracovával jen vysílání určená jemu, *MAC Frame Filtering* nám to umožňuje. Povolení této funkcionality je komplexní operací, manuál [Dec17] jej popisuje blíže. Dále je třeba nastavit modulu jeho adresu, po restartu zařízení je na všech modulech výchozí hodnota. Po spuštění senzoru proto v rámci funkce setup() nastavuji hodnotu SHORT_ADDR v registru *PAN Identifier and Short Address*. Způsobů adresování je několik, já jsem zvolil právě porovnávání adresy v hlavičce přijatého rámce s hodnotou nastavenou v 16bitovém poli SHORT_ADDR.

Knihovna DW1000-ng obsahuje některé funkce pro nastavení *MAC Frame Filtering*, ovšem při odesílání zprávy neukládá do hlavičky adresu příjemce. Implementoval jsem vlastní funkci *writeFrameHeader()*, která nastaví potřebné bity v datovém rámci před odesláním zprávy. Za tímto účelem v kódu figuruje jedno datové pole bytů pro odesílané zprávy a jedno pro přijímané. Do každého pole směřují dva pointery, jeden na jeho začátek (počátek celého rámce, oblast hlavičky) a jeden na počátek oblasti pro ukládání obsahu zprávy. Setkáme se tedy celkem se 4 proměnnými typu byte* (*frame_in*, *frame_out*, *data_in*, *data_out*). Fyzicky alokována jsou však jen dvě pole. Funkce připravující obsah k odeslání (a vyčtení po přijetí) poté pracují s příslušným ukazatelem na hlavičku nebo samotný obsah zprávy.

5.2.5 Potvrzování přijetí

Při testování jsem se potýkal se značnou nespolehlivostí datového přenosu mezi moduly DWM1000. Významná část zpráv nebyla po odeslání doručena a bylo nutné zavést nějakou



Obrázek 5.5: Struktura datového rámce modulu DWM1000, převzato z [Dec17]

formu potvrzování přijetí zpráv. Moduly v rámci *MAC Frame Filtering* nabízí funkci *Automatic Acknowledgement*. Tu jsem pro její komplexnost nepoužil a implementoval jsem vlastní systém potvrzování na úrovni firmwaru.

Každé odeslané zprávě ukládám do hlavičky rámce identifikační číslo. U příkazů, které to vyžadují, odešle modul po přijetí zprávy nazpět zprávu typu WIRELESS_CMD_ACK. Ta informuje odesílatele o tom, že zpráva s daným ID byla úspěšně doručena. Vyžaduje-li odesílatel toto potvrzení a nedojde k jeho přijetí v časovém limitu, reaguje opakovaným odesláním zprávy. Jedním z parametrů funkce *sendWirelessCommand()* je počet pokusů, který má modul vykonat při odesílání zprávy. Vysoký počet opakování je nastaven u konfiguračních zpráv, jejichž nedoručení by např. mohlo ohrozit funkčnost lokalizace.

5.2.6 Tvorba časových značek

Modul DWM1000 je vhodný pro implementaci senzoru vzdálenosti právě díky schopnosti přesně zaznamenávat časové okamžiky přijetí a odeslání zpráv. Vyslání jedné zprávy (datového rámce) však není instantní operací. Na obr. 5.5 vidíme strukturu datového rámce. Délka jednoho rámce dosahuje řádově 1 000 symbolů (záleží na konfiguraci), přičemž každý symbol prodlužuje dobu vysílání dle [Dec17] asi o $1\mu s$. Než dojde k odeslání datového rámce do prostředí, signál z počátku vysílání stihne teoreticky urazit vzdálenost několika kilometrů.

Je nutné definovat, který okamžik považujeme za moment odeslání zprávy, protože doba vysílání je řádově delší než intervaly, které chceme měřit. Dle standardu IEEE 802.15.4 je okamžikem odeslání moment, kdy anténu opustí první symbol PHR (část rámce těsně před vlastními daty, viz obr. 5.5).

Obr. 5.5 napovídá, že nejdelší vysílací čas zabírá část zvaná preambule, která slouží k signalizaci budoucího vysílání. Moduly ve stavu příjmu skenují prostor a snaží se detekovat sekvenci odpovídající preambuli. Její délka je nastavitelná, kratší preambule dovolují rychlejší komunikaci, klesá však spolehlivost přenosu. Zejména při komunikaci na delší vzdálenosti nebo v prostředí se silnějším rušením, protože přijímací modul nemusí vysílání zachytit. Ve firmwaru senzoru používáme preambuli délky 1024 symbolů. Kratší preambule způsobovala problémy s kvalitou spojení i v rámci jedné místnosti. Pro porovnání, přenášená data tvoří v našem případě pouhých 24 bytů.

Důležitou funkcí, kterou moduly DWM1000 nabízí, je tzv. opožděné odeslání. Umožňuje nám dopředu naplánovat moment odeslání zprávy ještě před tím, než k němu dojde. Čas odeslání nastavíme zápisem do registru *Delayed Send or Receive Time*. Poté máme ještě možnost upravit obsah zprávy a zahájit proces odesílání aktivací příslušných bitů v *System Control Register*. Modul předpočítá, kdy je třeba zahájit napájení antény a začít vysílat

preambuli. Přejde do neaktivního stavu a po probuzení systémovým časovačem zahájí přenos tak, aby první symbol PHR sekvence opustil anténu v nastavený moment. Díky tomu může odeslaná zpráva obsahovat informaci o čase svého odeslání. Toho využíváme při dvoustranném dvoucestném měření vzdálenosti. Čas odeslání lze nastavit s rozlišením 8 ns [Dec17].

5.2.7 Zpoždění antény

Při opožděném odesílání modul předpočítává, v jakém čase má zahájit vysílání, aby první symbol PHR sekvence opustil anténu v definovaném čase. Mezi informací, kterou modul odhadne, a časem, kdy signál skutečně opouští anténu, existuje zpoždění. Hodnota zpoždění je variabilní napříč zařízeními a pro zvýšení přesnosti měření je s ní třeba počítat. Moduly umožňují zápis zjištěných hodnot do registru *Transmitter Antenna Delay*. Poté zpoždění automaticky zahrnují do interních výpočtů a funkce související se získáváním časových značek pracují precizněji.

Dle instrukcí v manuálu [Dec17] jsem nastavit vhodné parametry vysílání pro účely kalibrace, vyčetl optimální vzdálenost odpovídající použitému kanálu a zvolil doporučený vysílací výkon. Bohužel se zvolenými parametry se mi nepodařilo navázat mezi moduly spojení. Vysílací výkon nebyl pro uvedenou vzdálenost dostačující a musel jsem ho korigovat tak, aby komunikace na vzdálenosti 8 m fungovala spolehlivě. Na dvojici modulů jsem nastavil hodnotu zpoždění antény na 0 a zahájil dvoucestné dvoustranné měření. Implementoval jsem skript v jazyku Python, který provádí opakované měření vzdálenosti, průměruje výsledky daného množství odměrů a mění hodnotu zpoždění antény, dokud se výsledky nepřiblíží referenční hodnotě změřené laserovým dálkoměrem.

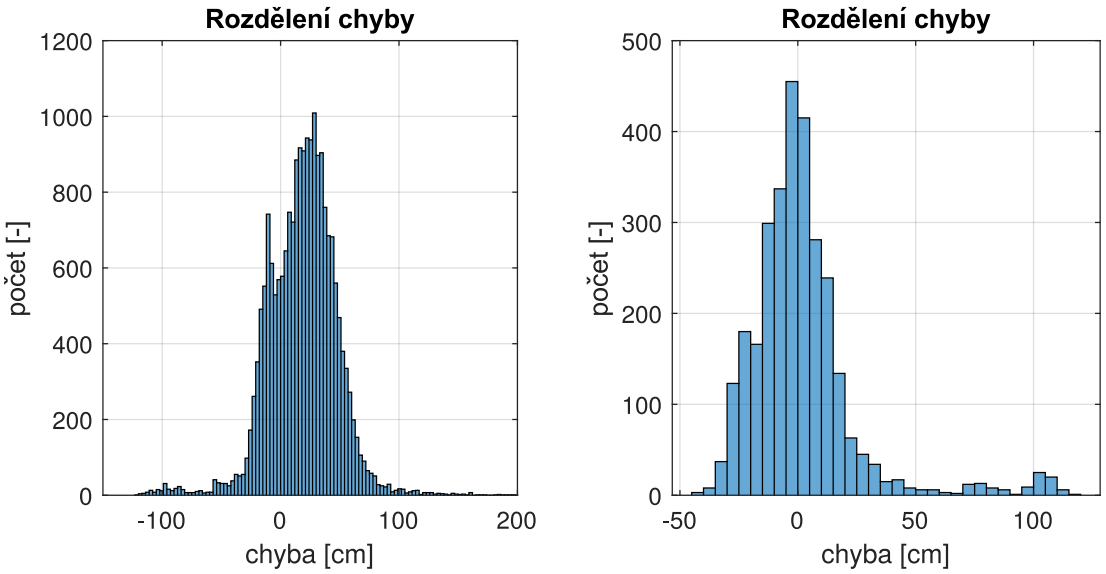
Popsaný postup jsem aplikoval pro různé dvojice modulů, aby každý modul figuroval alespoň ve třech měřeních. Výsledky z jednotlivých experimentů jsem zaznamenával, získané zpoždění antény bylo vždy součtem zpoždění obou zúčastněných modulů. Se získanými daty jsem metodou nejmenších čtverců nalezl přibližné řešení soustavy a stanovil zpoždění jednotlivých antén. Hodnoty jsou uloženy ve firmwaru senzorů a po zapnutí jsou automaticky nahrány do registrů modulu. Zjistil jsem rozdíly mezi anténami okolo 20 LSB, přičemž 1 LSB odpovídá dle manuálu 15,65 ps.

Při opakované kalibraci provedené s odstupem několika dní jsem zaznamenal odlišné výsledky. Odhad vzdálenosti se lišil od referenčních hodnot přibližně o 10 cm na vzdálenosti 8 m. Zpoždění antén a pravděpodobně i další parametry zanášející do procesu chybu se v čase mění. Více v kapitole 5.2.8.

5.2.8 Převodní charakteristika

Po zkalibrováni antén jsem se rozhodl ověřit přesnost měření pomocí systému Vicon. Do prostoru jsem umístil jeden senzor s modulem DWM1000 a do jeho těsné blízkosti položil marker systému Vicon. S druhým senzorem (taktéž s markerem) jsem se pohyboval po monitorovaném prostoru a kontinuálně měřil vzdálenost. Experiment jsem provedl pro několik dvojic senzorů. Výsledky jsem zaznamenával a v Matlabu poté sestavil histogram chyby měření, tedy rozdílu vzdálenosti změřené Viconem a naším senzorem. Histogram je k nalezení na obr. 5.6a a na první pohled je z něj patrné, že senzor systematicky vrací posunuté výsledky.

Zpracoval jsem proto i převodní charakteristiku senzoru, viz obr. 5.7. Abychom zjistili, zda se nějaký ze senzorů nechová odlišně oproti ostatním, na obr. 5.7a je barevně vyveden



(a) : Vstupní data před kompenzací

(b) : Kontrolní měření po kompenzaci

Obrázek 5.6: Histogram chyby měření senzoru vzdálenosti

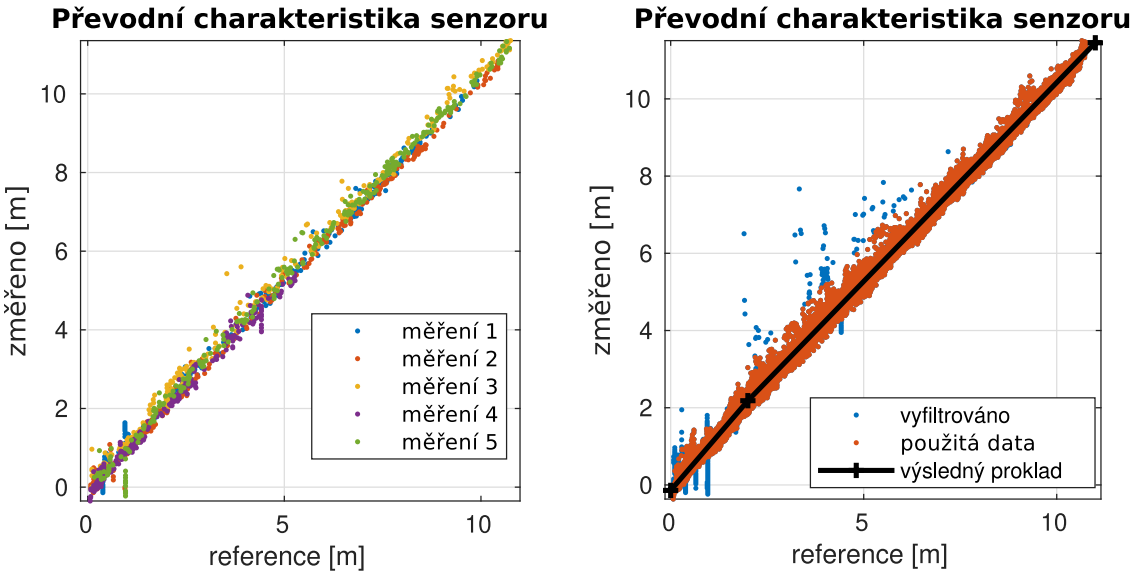
výběr z měřených bodů v rámci jednotlivých experimentů. Výsledky se zdají konzistentní a výraznější rozdíly mezi senzory nejsou. Sloučil jsem tedy data ze všech experimentů do jednoho velkého vzorku a provedl proklad přímkou s užitím metody nejmenších čtverců. Výsledek nebyl optimální, opticky se mi zdálo, že data blíže nule (do vzdálenosti asi dvou metrů) jsou více zkreslená. Přistoupil jsem nakonec k prokladu lomenou čárou namísto přímky. Na obr. 5.7b můžeme vidět černě výsledek prokladu. Modře jsou vykresleny body výrazně vzdálené od ideálu, které jsem z prokladu vyloučil.

V převodních charakteristikách si můžeme všimnout úseků, kde pro totožnou referenční hodnotu existuje řada různých měření. Místa se projevují v grafu jako svislé čáry. Ty vznikají v momentě výpadku detekce systémem Vicon, proto se po nějakou dobu nemění referenční vzdálenost, a nejedná se o chybu měření.

Po nalezení vhodného prokladu jsem implementoval kompenzaci měřených výsledků do knihovny zpracovávající výsledky na PC a provedl kontrolní měření. Výslednou převodní charakteristiku nalezneme na obr. 5.8a. Modrou barvou jsou vynesena původní data použitá k prokladu. Obr. 5.8b přibližuje detail převodní charakteristiky po kompenzaci. Ten je zajímavý hysterezí v oblasti okolo 1,7 m. Nejspíše reprezentuje chůzi směrem od senzoru a zpět při kontrolním měření. Vyskytovat by se však neměla, jelikož v průběhu experimentu byla po celou dobu zajištěna přímá viditelnost mezi senzory (nebyl důvod k šíření po delší trajektorii) a pohyboval jsem se velmi pomalu (vyloučen je vliv zpoždění senzoru). Nejedná se ani o náhodný artefakt, jelikož podobných míst je na převodní charakteristice více. Jediný důvod, který mě napadl, je možný vliv orientace antény.

Dalším výstupem kontrolního měření je histogram chyby na obr. 5.6b. Ten jasně ukazuje, že kompenzace zlepšila přesnost senzoru.

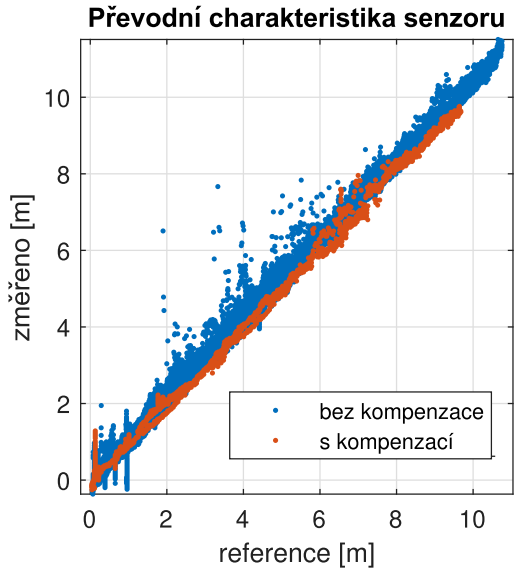
Stejné kontrolní měření převodní charakteristiky jsem opakoval po týdnu od kalibrace (nalezení prokladu), abych zjistil, zda se přesnost senzoru mění s časem. Kompenzaci výsledků jsem provedl na základě původních parametrů. Jak můžeme vidět na obr. 5.9, převodní



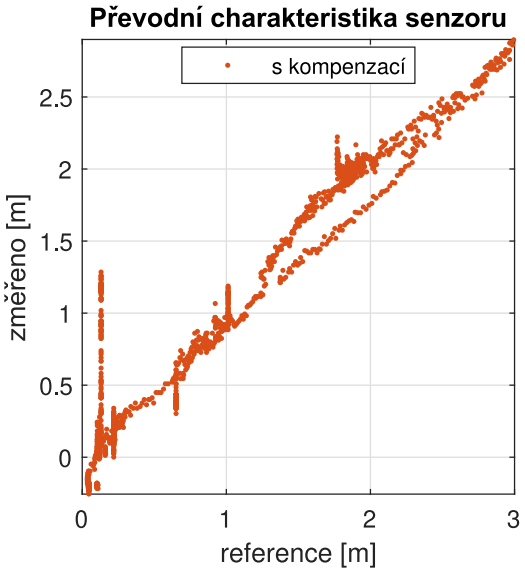
(a) : Před kompenzací **(b) :** Po kompenzaci

Obrázek 5.7: Proklad převodní charakteristiky senzoru

charakteristika je posunuta o 10 cm výše. Ke změně tedy dojít mohlo, vzhledem k rozptylu předchozích měření to však považuji za změnu spíše nevýznamnou.

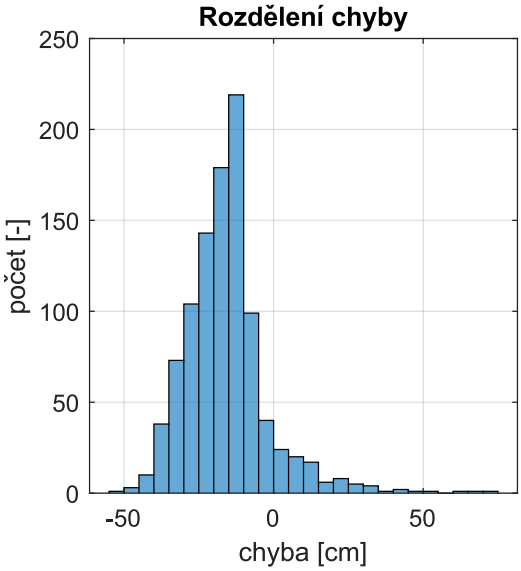


(a) : Histogram chyby měření

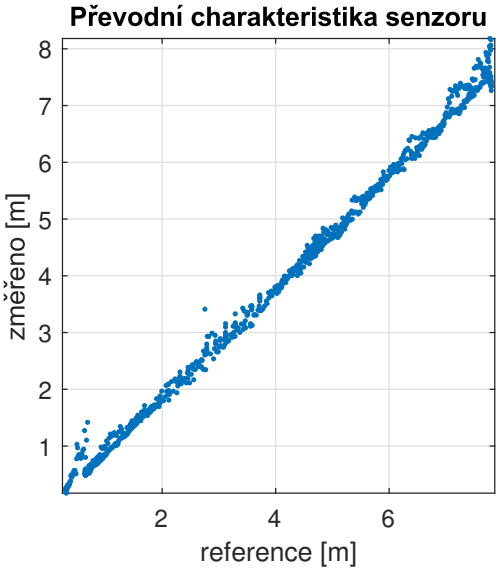


(b) : Převodní charakteristika

Obrázek 5.8: Převodní charakteristika senzoru vzdálenosti při kontrolním měření



(a) : Histogram chyby měření



(b) : Převodní charakteristika

Obrázek 5.9: Opakované kontrolní měření převodní charakteristiky

5.3 Knihovna pro PC

Důležitou součástí práce je soubor programových modulů pro řídicí počítač, který jsem navrhl a implementoval v jazyku C++. Umožňuje propojovat jednotlivé součásti dle schématu na obr. 5.1. Obsahuje knihovny pro ovládání senzorů vzdálenosti, výpočet polohy, udržování formací a řízení helikoptér. Navíc dokáže komunikovat se systémem Vicon a získávat referenční informace o poloze objektů.

5.3.1 Měření vzdálenosti

Komunikaci mezi počítačem a senzorem připojeným přes USB zajišťuje mnou vyvinutá třída nazvaná *DWloc*. Při konstrukci objektu této třídy předáváme jako parametr cestu k USB rozhraní, přes které bude probíhat komunikace. Následně voláme metodu *init()*, ta zajistí nastavení připojeného senzoru a spustí vlákno pro komunikaci přes USB/UART. Senzor odešle po zapnutí svou adresu z EEPROM, v knihovně je poté uložena do parametru *my_address*. Uvedu příklad kódů, který nastaví senzor připojený k PC i ostatní členy senzorové sítě.

```
#include "DWloc.hpp"
// init sensor
DWloc module("/dev/ttyUSB0");
module.init();
// set anchors
module.set_node(a1, RESPONDER, module.my_address, 5);
module.set_node(a2, RESPONDER, module.my_address, 5);
module.set_node(a3, RESPONDER, module.my_address, 5);
module.set_node(a4, RESPONDER, module.my_address, 5);
```

Metoda *set_node()* provede konfiguraci vzdálených senzorů odesláním příslušných příkazů. V tomto případě a_i značí adresu, na kterou budou příkazy odeslány. Senzor bude nastaven do režimu RESPONDER a příkazem RANGE_REPORT_ADR je informován o adrese senzoru připojeného k PC. Více informací o komunikačním protokolu nalezneme v kapitole 5.2.2. Knihovna však sestavení a odeslání příkazů provede interně, zajistí vykonání SEND_WIRELESS_CMD a uživateli stačí operovat s připravenými metodami.

```
// get distance measurement
module.ask_for_ranging_blocking(a1,a3,3,&r1,100);
```

Tento řádek kódu požádá senzorovou síť o měření mezi senzorem s adresou *a1* a *a3*. Třída *DWloc* opět zajistí odeslání potřebných příkazů. Na odeslání zprávy přes UWB (a potvrzení doručení) jsou nastaveny 3 pokusy, výsledek měření bude uložen do proměnné *r1*. Kód vyčká na výsledek měření v blokovacím režimu s časovým limitem 100 ms.

Kromě měření v blokovacím režimu existuje i metoda *ask_for_ranging()*, která pouze iniciuje měření a vykonávání kódu pokračuje dále. Poslední platný záznam vzdálenosti mezi dvěma senzory je uchováván uvnitř objektu třídy *DWloc*. Po přijetí výsledku se hodnoty na pozadí automaticky aktualizují. Vyčítat dostupná data lze v jakýkoli moment metodou *read_latest_distance()*.

Třída obsahuje také metodu pro manuální zaslání příkazu do senzorové sítě *send_wireless_cmd()*. Dále je připravena řada funkcí pro automatické sestavení a odeslání příkazů dle komunikačního protokolu. Můžeme tedy pohodlně komunikovat se senzorem připojeným k PC i se všemi ostatními.

Neméně důležitá je schopnost kompenzace převodní charakteristiky senzorů. Při přijetí výsledku měření ze senzorové sítě dojde automaticky k aplikaci kalibračních konstant. Senzor přes USB tedy odesílá ještě neupravené hodnoty, kompenzace proběhne interně v rámci zpracování přijaté zprávy a uživatel již pracuje s opravenými daty.

5.3.2 Ovládání dronů

Řízení helikoptér Tello probíhá přes WiFi síť. Ke komunikaci s letouny používáme přepis knihovny PyTello [Pin] do jazyka C++, který byl poskytnut vedoucím práce. Na knihovně jsem provedl pouze drobné úpravy. Jednou z nich je možnost ovládat více dronů současně s použitím externích síťových WiFi karet.

Následující útržek kódu provede konstrukci objektu třídy *CTello* zahrnující navázání komunikace s letounem. Parametry konstruktoru jsou komunikační port a identifikátor síťové karty v systému, přes kterou potečou pakety určené tomuto letounu. Více o způsobu připojení nalezneme v kapitole 3.2.3.

```
#include "tello.h"
CTello tello1;
tello1.init(8889, "wlxd0374543f3a6");
tello1.takeOff();
tello1.setStickData(0, roll, pitch, throttle, yaw);
...
tello1.land();
```

Poté je vyslán pokyn k vzletu a nastavení letových parametrů helikoptéry. Funkce pro vzlet a hlavně pro přistání jsem upravil tak, aby byl požadavek odeslán opakovaně. Při testování docházelo ke ztrátám paketů a nevykonání požadované operace.

Letové parametry nastavené funkcí *setStickData()* jsou uloženy do interních proměnných třídy a do sítě jsou odesílány knihovnou v pravidelných intervalech (nastaveno na 50 ms).

5.3.3 Připojení k systému Vicon

Jako zdroj referenční polohy při experimentech používáme lokalizační sytém Vicon, více viz kapitola 3.3. Vicon odesílá polohu sledovaných objektů po místní síti v laboratoři. Při čtení dat ze sítě vycházím z knihovny poskytnuté vedoucím.

Nad existující knihovnou jsem vybudoval nadstavbu umožňující pohodlně sledovat několik objektů současně v odděleném vláknu. Službu zprostředkovává třída *Vicon_master*. Na pozadí přijímá pakety s polohou všech dostupných cílů. Hodnoty těch, které spadají do našeho zájmu, ukládá. Následující kód nastiňuje fungování třídy.

```
#include "vicon_master.hpp"
```

```
#include "dzony.hpp"
Vicon_master vicon;
int VIC_ID1 = vicon.add_target("Tello1");
vicon.connect();
...
mypoint_t vicon_position;
bool valid = vicon.get_position(&vicon_position,VIC_ID1);
```

Metodou *add_target()* přidáváme textové identifikátory objektů, které chceme sledovat. Získáme číselné ID, které v budoucnu používáme pro vyčtení poslední platné polohy. Funkci *get_position* můžeme volat v libovolný moment. Je-li dostupná informace o poloze objektu, uloží se do proměnné *vicon_position* typu *mypoint_t*.

```
struct mypoint_t {
    double x;
    double y;
    double z;
};
```

Mypoint_t je datový typ používaný napříč knihovnou pro reprezentaci souřadnic bodu v prostoru. Definuje ho hlavičkový soubor *dzony.hpp*.

5.3.4 Udržování trojúhelníkové formace

Výpočty související s udržováním formace provádí třída *Triangle*. Na základě poskytnutých dat o vzájemné vzdálenosti agentů nám vrací akční zásah, o kterém předpokládá, že přispěje k zachování požadovaného útvaru.

```
#include "triangle.hpp"
Triangle triangle1(d11,d12,w);
...
triangle1.make_step(r1,r2, timestamp);
...
yaw = triangle1.angle;
roll = triangle1.side_shift;
pitch = triangle.v;
```

V uvedeném kódu voláme konstruktor třídy *Triangle* s parametry definujícími tvar formace. Hodnoty *d11* a *d22* vyjadřují požadovanou vzdálenost sledujícího agenta od senzorů vůdce, *w* je rozpětí senzorů na vůdci. Ve formaci o větším počtu agentů je každý sledující letoun reprezentován vlastním objektem této třídy. Regulace i měření probíhá nezávisle na ostatních.

Krok algoritmu provádíme voláním metody *make_step()*. Algoritmus na základě hodnot o změřené vzdálenosti aktualizuje vnitřní proměnné. K těm poté přistupujeme v momentě odesílání příkazů dronům.

5.3.5 Lokalizace s použitím pevných kotev

Knihovna implementuje algoritmus představený v kapitole 2.3.1 včetně úprav popsaných v kapitole 2.4.1. Sledované objekty jsou reprezentovány třídou *Target*, přičemž je možné provádět sledování několika cílů současně, každý z nich provozuje vlastní instanci Kálmánova filtru. Následující úryvek kódu vytvoří cíl ze senzoru s adresou 8 a zařadí ho ke sledování metodou *add_target()*.

```
#include "DWloc/DWloc.hpp"
#include "DWloc/kalman.hpp" // Target class
DWloc module((char*)"/dev/ttyUSB0");
module.init();
Target dw_target1(8);
int DW_ID1 = module.add_target(dw_target1);
```

Uživatel získá identifikátor *DW_ID1*, který bude následně používat pro vyčítání polohy objektu. Dále je k zahájení procesu lokalizace nutné definovat polohu kotev. A jejich seznam předat jak cíli *dw_target1* pro účely odhadu polohy, tak objektu *module*, který bude řídit proces lokalizace a ve smyčce zajišťovat proměřování vzdáleností.

```
vector<Anchor> anchors;
anchors.push_back(Anchor(9,-2.056209, 3.933867, 1.19));
...
module.add_anchors_list(anchors);
dw_target1.add_anchors_list(anchors);
dw_target1.init();
```

Metoda *init()* vytvoří systémové matice Kálmánova filtru a připraví objekt cíle k odhadu své vlastní polohy. Maticové výpočty provádím za užití knihovny Eigen [GJ+10].

Samotné sledování zahajuje funkce *start_tracking()*, viz kód níže. Objekt *module* spustí nekonečnou smyčku, v rámci které v odděleném vlákně zajišťuje měření vzdálenosti mezi všemi sledovanými agenty a všemi kotvami. Aktualizaci odhadu stavů provádí metodou *update_state()* pro každého agenta.

```
mypoint_t dwloc_position;
module.start_tracking();
module.get_estimated_pos_blocking(&dwloc_position, DW_ID1);
```

Získat poslední známý odhad polohy můžeme mj. funkcí *get_estimated_pos_blocking()*.

Kapitola 6

Provedené experimenty

6.1 Lokalizace s užitím pevných kotev

Metodu lokalizace mobilních agentů založenou na EKF jsme si popsali v kapitole 2.3.1. Se senzory osazenými modulem DWM1000 měříme vzdálenosti mobilních agentů od kotev s definovanou polohou v globálním souřadném systému. Znalost vzdálenosti nám díky EKF umožňuje provádět odhad polohy. V teoretické části byl uveden popis pro lokalizaci v rovině, v experimentech pracujeme i se třetí souřadnicí.

Uspořádání systému je znázorněno na obr. 6.1. Z řídicího PC vysíláme v cyklu požadavky na měření do senzorové sítě UWB. Výsledky zpracováváme a provádíme odhad polohy. Níže si popíšeme výsledky několika experimentů provedených dle tohoto schématu. Referenční hodnoty o poloze získáváme ze systému Vicon, to nám dovoluje současně vyhodnotit přesnost měření senzorů.

Zdůvodnění centralizovaného přístupu

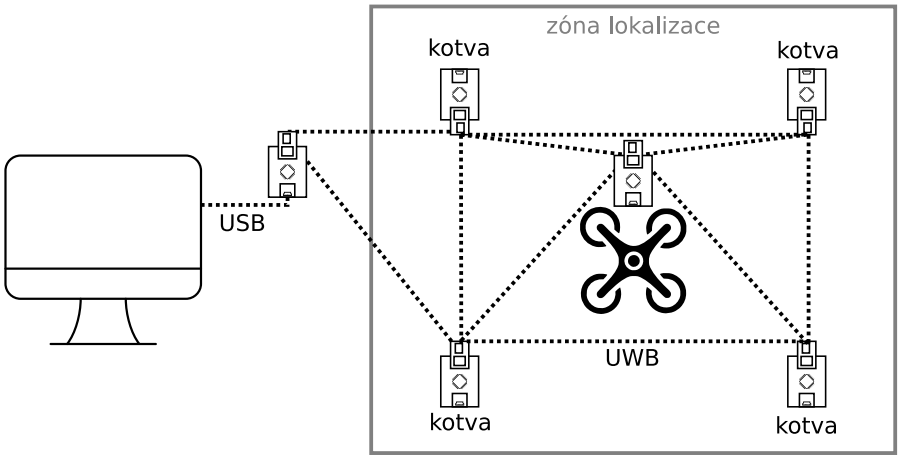
Testoval jsem více přístupů s různou mírou centralizace. Namísto nutnosti měření iniciovat požadavkem z PC lze přepnout senzor mobilního agenta do režimu INITIATOR, popsaného v kapitole 5.2.2. Senzor sám ve smyčce provádí odměry vzdálenosti ke kotvám a výsledky jsou reportovány na nastavenou adresu (adresu senzoru připojeného k PC). Tím můžeme ušetřit část vysílacího času. Reportování výsledků však provádí druhý senzor, než který měření zahájil (viz kapitola 5.2.3). INITIATOR tak nemá informaci o tom, zda byl výsledek do PC úspěšně doručen, nebo dochází k opakovanému vysílání. Načasování je proto těžko proveditelné.

Sledujeme-li více mobilních agentů současně, přestává systém se senzorem v režimu INITIATOR fungovat zcela. Aktivujeme-li dva takové senzory, dochází ke vzájemnému rušení jejich komunikace. Celý proces je tak velmi chybový, dochází často k opakovanému odesílání požadavků a výraznému zpomalení oproti centralizovanému přístupu.

6.1.1 Výsledky experimentů

První scénář, který si popíšeme, testuje vlastnosti Kálmánova filtru po spuštění. Vnitřní proměnné filtru musí být před startem nastaveny na nějakou hodnotu. V našem případě předpokládáme, že se agent nachází v počátku souřadného systému. Po zahájení lokalizace trvá několik iterací algoritmu, než se odhad srovná s realitou. Rychlost náběhu můžeme ovlivnit vhodnou volbou konstant, zejména nastavením kovariančních matic Q a R.

Hodnota R by měla odpovídat rozptylu šumu senzoru. Hodnota Q očekávaným změnám v poloze agenta mezi iteračními kroky algoritmu. Oba údaje však bylo potřeba doladit



Obrázek 6.1: Infrastruktura systému při lokalizaci s užitím pevných kotev

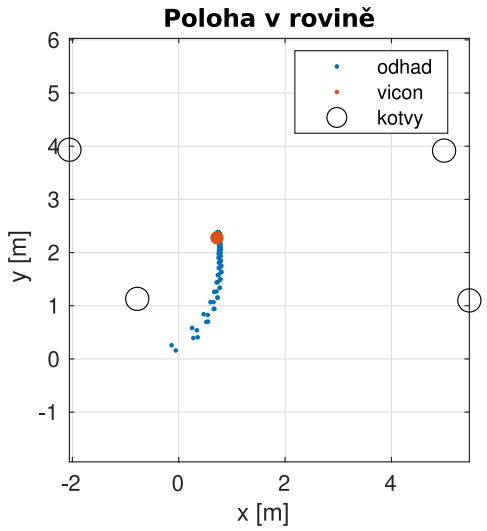
experimentálně. Šum senzoru při experimentech dosahoval vyšších hodnot, než jsme očekávali po měření převodních charakteristik senzoru, navíc se nejedná o normální rozdělení. A očekávaná změna polohy agenta je těžko definovatelná, záleží na počtu sledovaných objektů a charakteru pohybu.

Na obr. 6.2 jsou výsledky prvního experimentu. Agent byl umístěn na neměnné místo a sledovali jsme náběh filtru z počátečních podmínek. K ustálení hodnot došlo přibližně po jedné sekundě od spuštění, přičemž reálná poloha byla od počátku souřadného systému vzdálena necelé 2 m. Po ustálení filtru jsme dosáhli chyby 10 cm (obr. 6.2c). Jedná se o velikost chyby odhadu pozice v rovině. Graf 6.2b ukazuje, že zvlnění odhadnuté polohy v čase je pouze nepatrné a konstanty filtru jsou naladěny přijatelně. Obr. 6.2d přináší zajímavou informaci, a to že všechny čtyři sledované vzdálenosti jsou senzory měřeny jako kratší, než by odpovídalo skutečnosti. Zjištění koresponduje s výsledky testu provedeného týden po kalibraci převodní charakteristiky senzoru (obr. 5.9). Experiment naznačuje, že chyba senzorů nebude odpovídat normálnímu rozdělení s nulovou střední hodnotou, které Kálmánův filtr ideálně předpokládá.

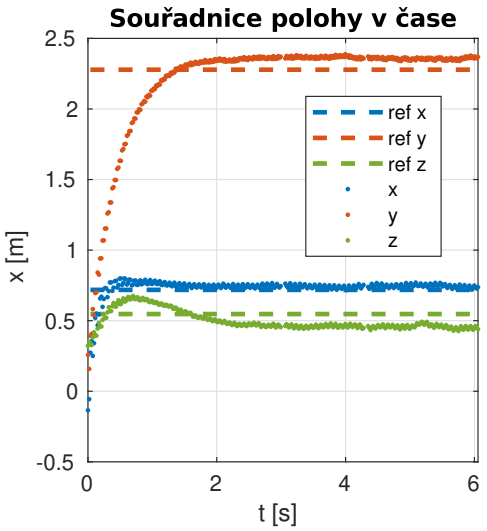
V rámci druhého experimentu (obr. 6.3) jsme zaznamenali trajektorii helikoptéry při manuálním řízení. Obletěli jsme dokola sledovaný prostor (obr. 6.3a), v němž se nenacházely žádné překážky. V prvních několika sekundách můžeme pozorovat náběh Kálmánova filtru. Poté se chyba lokalizace v rovině ustálila na úrovni 30 cm. Chybu v rovině uvádím proto, že kotvy při experimentu jsou umístěny ve výšce od 15 cm do 150 cm. Rozlišovací schopnost algoritmu v souřadnici z je tím potlačena. V případě možnosti umístit kotvy výše by byly výsledky ve všech osách srovnatelné.

Na průběhu naměřených vzdáleností v čase (obr. 6.3d) jsou výrazné dva momenty, kdy došlo ke zvýšené chybovosti. Ani jeden z nich nemá konkrétní příčinu. S největší pravděpodobností došlo k šíření signálu jinou než nejkratší cestou, tedy odrazem od stěn či podlahy. Mezi helikoptérou a kotvami se nevyskytovaly žádné překážky, mohlo však dojít k dílčímu zastínění antény jinou částí dronu. Kromě dvou výrazných míst dochází poměrně často k menšímu kolísání přesnosti, stále však výrazně nad úrovní očekávaného šumu.

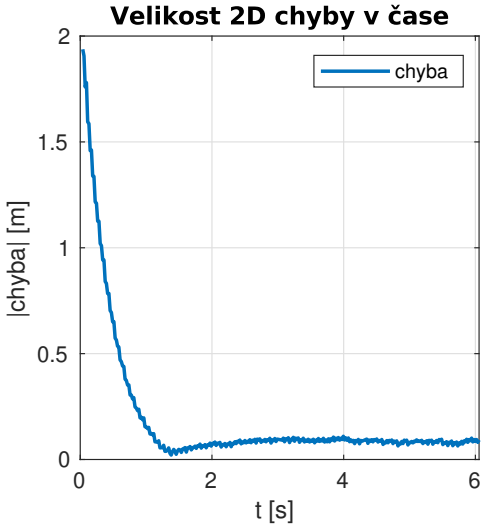
Podobné incidenty, jako byly zaznamenány ve druhém experimentu, nejsou ojedinělou záležitostí. K náhlému zhoršení přesnosti měření vzdálenosti docházelo při testování v podstatě při každém vzletu.



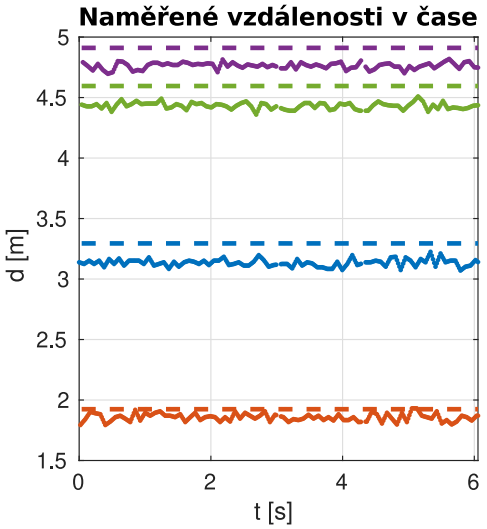
(a) : Umístění kotev a odhad polohy agenta v rovině



(b) : Odhad jednotlivých souřadnic polohy v čase



(c) : Časový vývoj absolutní hodnoty chyby lokalizace v rovině

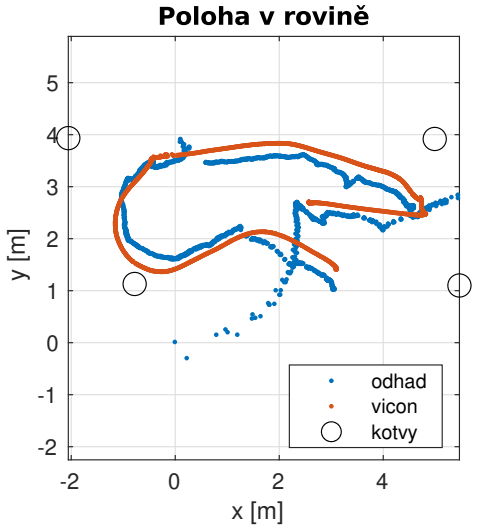


(d) : Naměřené vzdálenosti k jednotlivými kotvám (referenční hodnoty čárkovaně)

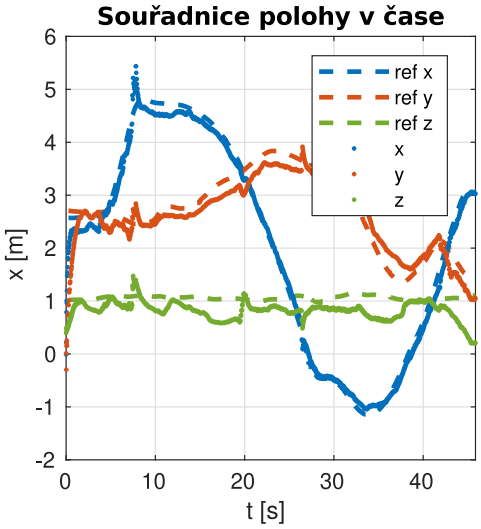
Obrázek 6.2: Výsledky 1. experimentu. Lokalizace s užitím pevných kotev, případ agenta s neměnnou polohou.

Podívejme se na poslední zde prezentovaný experiment. Zkusil jsem zvýšit hodnotu kovariační matice R s cílem odfiltrovat efektivněji popisované chyby. Výsledky jsou na obr. 6.4. Došlo k významnému nárůstu zpoždění filtrace, obr. 6.4b ukazuje, že odhad zaostává o sekundu za reálným stavem. To vedlo k celkovému nárůstu chyby polohy. Přitom agent se pohyboval rychlostí velmi pomalé chůze. Průběh chyby je však hladší a lépe predikovatelný. Zlom v experimentu nastává okolo času 10 s. S helikoptérou jsem se přiblížil ke sloupu. Sloup se nachází mimo oblast vytyčenou kotvami, stál tedy za helikoptérou a nic nebránilo přímému výhledu ke všem kotvám. Přesto však začalo docházet k častým odrazům signálu o stěnu sloupu a degradaci měřených výsledků.

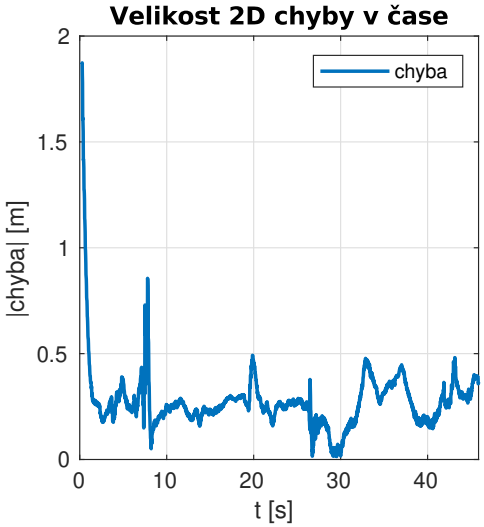
V těchto případech, kdy se nejedná o ojedinělé výstřelky, ale o opakovaně chybná měření po delší časovou periodu, nepomohly ani přehnaně vysoké konstanty matice R či jiné způsoby



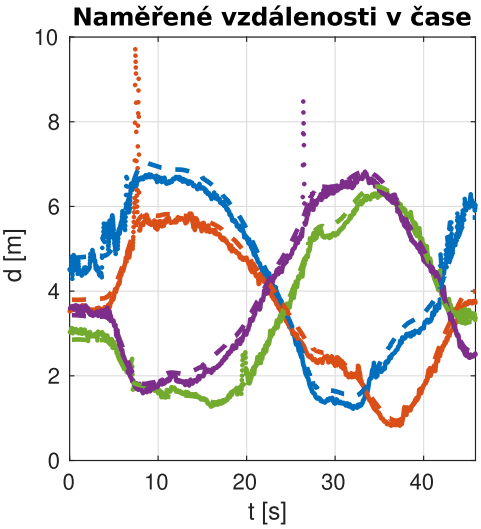
(a) : Umístění kotev a odhad polohy agenta v rovině



(b) : Odhad jednotlivých souřadnic polohy v čase



(c) : Časový vývoj absolutní hodnoty chyby lokalizace v rovině



(d) : Naměřené vzdálenosti k jednotlivými kotvám (referenční hodnoty čárkovaně)

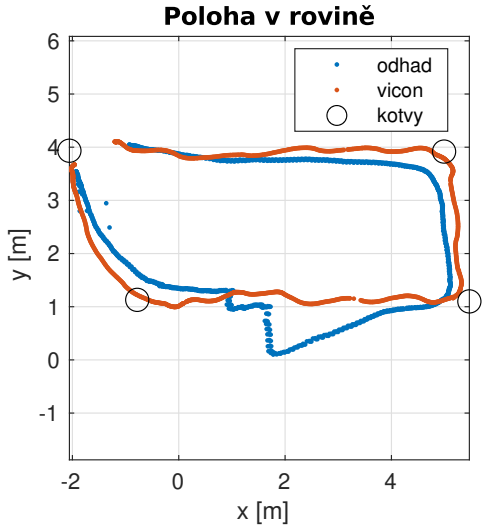
Obrázek 6.3: Výsledky 2. experimentu. Lokalizace s užitím pevných kotev, případ manuálně řízeného letu.

dodatečné filtrace. Ještě markantnější je tento problém v čase 15 s. Vzdálenost ke kotvě vyznačené na obr. 6.4d oranžovou barvou zaznamenala nikoli enormní, avšak zvýšenou chybovost po čas následujících 5 s. Na vývoji chyby lokalizace se to zřetelně projevilo.

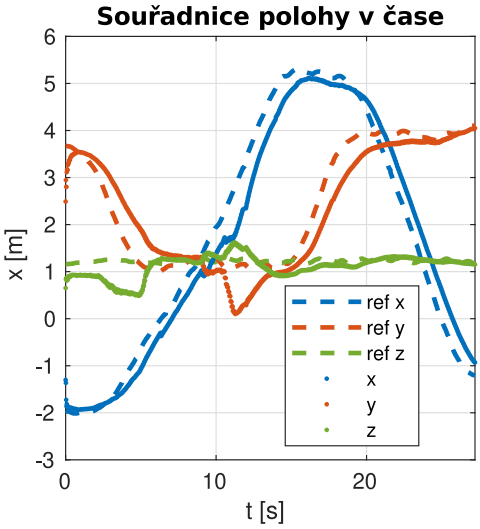
Pozn.: V datech experimentu č. 3 není náběh filtru zaznamenán.

Dodatečné poznatky k přesnosti

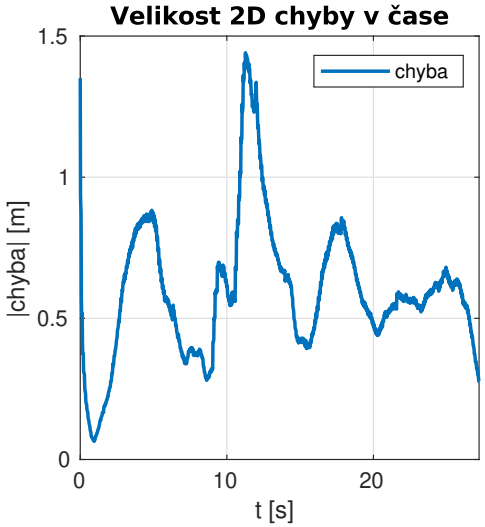
Rád bych představil ještě pár grafů získaných při druhém experimentu. Na obr. 6.5a je histogram chyby měření vzdálenosti od sledovaného agenta ke všem kotvám. Modus chyby leží na hodnotě -25 cm. Obrázek potvrzuje domněnku, že v rámci několika týdnů došlo k posunu převodní charakteristiky senzoru. Pro přesnější měření by bylo patrně nutné provádět



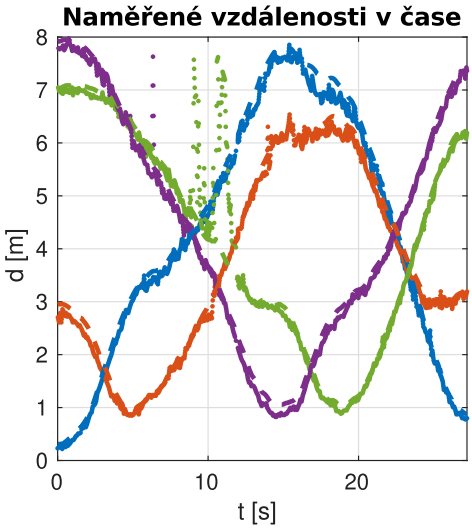
(a) : Umístění kotev a odhad polohy agenta v rovině



(b) : Odhad jednotlivých souřadnic polohy v čase



(c) : Časový vývoj absolutní hodnoty chyby lokalizace v rovině



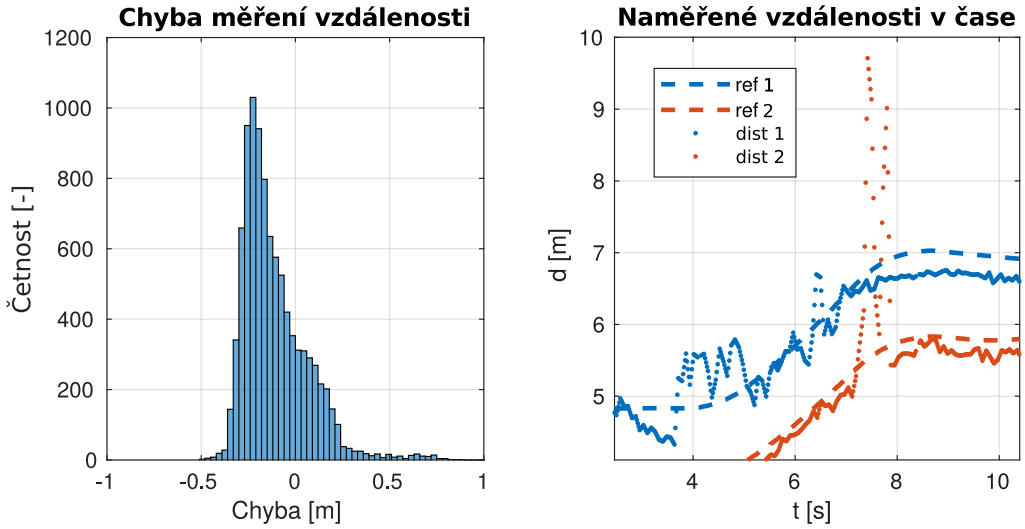
(d) : Naměřené vzdálenosti k jednotlivými kotvám (referenční hodnoty čárkovaně)

Obrázek 6.4: Výsledky 3. experimentu. Lokalizace s užitím pevných kotev, případ sledování agenta s agresivnější filtrací.

kalibraci pravidelně, s patřičně zahřátým senzorem.

Histogram na obr. 6.5a neobsahuje celou škálu zaznamenaných hodnot, v použitém měřítku by nebyly nízké incidence vidět. Ovšem v jednotkách případů přesahovala chyba měření i 4 m. Nejspíš v důsledku odrazu signálu nebo zastínění antény jinou části helikoptéry. Jeden z problematických momentů si přiblížíme na obr. 6.5b. Jde o detail z grafu 6.3d. Nepřesnost měření zde dosahuje násobně vyšších hodnot, než ve většině záznamu. Vzhledem k časovému vývoji dat však nemůže filtr rozpoznat, že se jedná o chybu (obzvláště u modrého průběhu). K podobnému jevu by klidně mohlo dojít reálným pohybem helikoptéry. Změna vzdálenosti je postupná, nejde o jednotlivé body odporující trendu.

Z důvodu odrazu signálů a částečně také zpožděním filtru při pohybu je přesnost lokalizace



(a) : Histogram chyby, detail přiblížený v ose x

(b) : Vzdálenost naměřená k jednotlivým kotvám, problematický moment.

Obrázek 6.5: Výsledky 2. experimentu, podrobnosti měření vzdálenosti.

v 1. experimentu lepší než v experimentech dalších. Dokládá to obr. 6.6.

Frekvence snímání polohy

Jedna iterace celého procesu, tedy vyslání požadavku na měření, zjištění vzdáleností, reportování výsledků do PC a vykonání odhadu polohy agenta, trvala v průměru

$$T_{avg} = 19,2\,\text{ms}. \tag{6.1}$$

Lokalizaci jednoho agenta dokážeme provádět s frekvencí $f \approx 50\,\text{Hz}$. Rychlost odezvy je poměrně konzistentní, 90 % cyklů bylo realizováno v intervalu

$$19\,\text{ms} < T < 20\,\text{ms}. \tag{6.2}$$

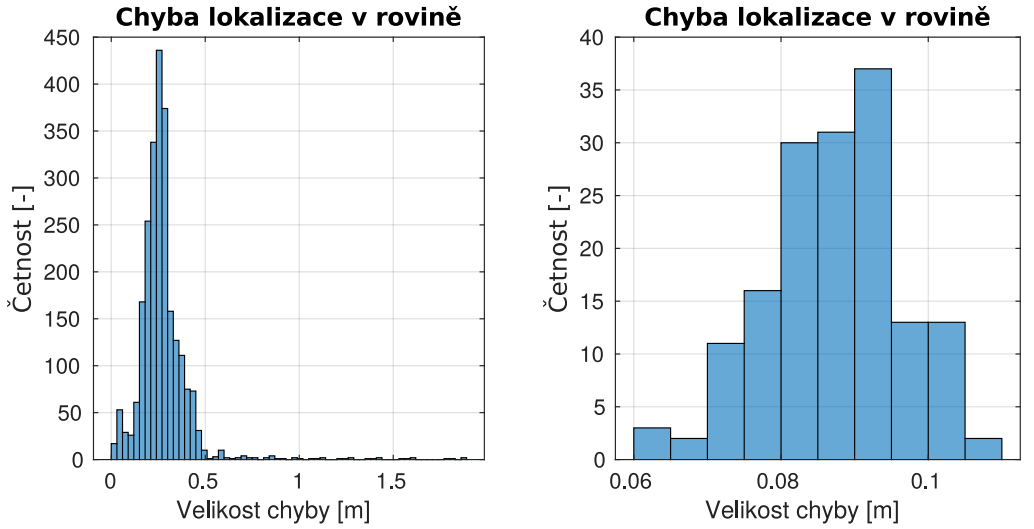
Nárazově však může dojít k výpadku měření vlivem zarušení UWB komunikace z neznámého zdroje. Konkrétně v průběhu posledního experimentu trvalo 5 odměrů $T > 50\,\text{ms}$, z čehož jeden přesáhl hranici $T > 100\,\text{ms}$ (z celkového počtu 1417 naměřených vzdáleností).

Sledování několika agentů současně

Provedl jsem řadu pokusů se sledováním dvou či více objektů současně. Pohyboval-li jsem senzory ve sledované oblasti manuálně, výsledky se nelišily od experimentů uvedených výše. Jediným rozdílem bylo zpomalení celého procesu. Perioda T_{avg} roste úměrně s množstvím agentů.

Situace se změnila při pokusech s více helikoptérami. Náhlé změny v přesnosti měření, které jsme pozorovali při experimentech výše, dosahovaly výrazně vyšší četnosti. Signál se patrně odrážel nejen od stěn místnosti, ale také od ostatních helikoptér v prostoru. Chvílemi docházelo i k zastínění přímé cesty mezi senzory jinou helikoptérou.

Pokusil jsem se také o udržování jednoduché formace s více drony. Záměr jsem s dostupnými helikoptérami v prostoru laboratoře vyhodnotil jako nerealizovatelný. Chyba lokalizace



(a) : Data z 2. experimentu (při letu) **(b) :** Data ze 1. experimentu (statický)

Obrázek 6.6: Srovnání přesnosti lokalizace mezi experimenty

přerostla únosnou mez ve chvíli, kdy si helikoptéry navzájem bránily ve výhledu na kotvy. Problém způsobovaly i odrazy signálu v momentech, kdy přímému výhledu zdánlivě nebránilo nic. Docházelo k častým kolizím mezi letouny a v experimentech jsem nepokračoval s cílem zabránit škodám na majetku a zdraví. Pomoci by mohlo použití jiných modelů helikoptér a testy na rozlehlejším prostoru. Větší vzájemná vzdálenost letounů by možná snížila množství odrazů a zároveň zajistila vyšší bezpečnost experimentů.

6.1.2 Shrnutí výsledků

Ověřili jsme metodu lokalizace s použitím pevných kotev a EKF. V informacích o modulu DWM1000 v kapitole 3.1 jsme uváděli, že jsou použitelné pro konstrukci lokalizačních systémů s přesností až 10 cm. To můžeme potvrdit, povedlo se nám v rámci experimentu se zaručenou přímou viditelnosti lokalizovat objekt s modem chyby polohy v rovině 9,3 cm (obr. 6.6b). Navržený Kálmánův filtr pracoval dle očekávání a poskytoval kvalitní odhad souřadnic agenta. Výsledky by bylo možné ještě zlepšit provedením kalibrace převodní charakteristiky senzorů těsně před experimentem, pravděpodobně dochází k jejímu časovému či teplotnímu driftu.

Při snímání polohy helikoptéry se senzorem umístěným mezi vrtulemi (obr. 3.5) jsme narazili na problém s přímou viditelností. Dostane-li se do cesty mezi anténami senzorů cizí objekt, např. část vrtule, signál se přestane šířit nejkratší možnou cestou a chyba lokalizace vzrůstá (obr. 6.6a). Problém se vyskytoval téměř při každém letu, přestože v prostoru mezi kotvami se kromě helikoptéry nenacházely jiné překážky. Pomoci by mohlo použití jiného modelu dronu, s lepšími možnostmi umístění senzoru. Je také třeba poznamenat, že anténa senzoru na helikoptéře byla umístěna přibližně 10 cm od referenčního bodu systému Vicon. V experimentech zaznamenávajících let (2. a 3.) proto chybu na této úrovni musíme tolerovat.

V každém případě jsme ukázali, že navržená lokalizační metoda i její implementace v knihovně pro PC funguje. Není vhodná pro demonstraci udržování formací s helikoptérami Tello. Může však posloužit s jinými drony či ji lze bez problému použít k záznamu polohy objektů libovolného charakteru.

6.2 Vlastnosti senzoru vzdálenosti

Díky možnosti měřit referenční hodnoty vzdálenosti systémem Vicon při experimentech s pevnými kotvami jsme odhalili několik problematických situací. Senzory vzdálenosti za jistých okolností nevrací správné hodnoty a jejich přesnost významně klesá. Provedl jsem proto několik experimentů s cílem přiblížit sobě i čtenáři, za jakých podmínek senzory pracují korektně. Stručně popíši typové situace.

- Výskyt osoby: při vzdálenosti antén 4 m nebyl průchod osoby na výsledcích znatelný, při 1 m měření znehodnoceno.
- Výskyt helikoptéry Tello: při vzdálenosti antén 4 m zvýšená chybovost, při 1 m měření znehodnoceno.
- Umístění kovové destičky 15×10 cm: měření selhalo při přiblížení k anténě ze všech stran (i zezadu). Signál se šířil odrazem po delší trajektorii.
- Natočení antény: změna orientace má mírný vliv na rozptyl šumu i střední hodnotu výsledku (do 10 cm). V případě, že je poblíž překážka (stěna, podlaha), změna natočení může způsobit odraz.
- Umístění senzorů na tyč s konstantní vzdáleností a postupné pokládání na podlahu: při přiblížení na cca 20 cm od země výsledky zkresleny odrazem.

Podrobněji se přesností senzorů zabýváme v kapitole 6.1 a 5.2.8.

6.3 Udržování trojúhelníkové formace

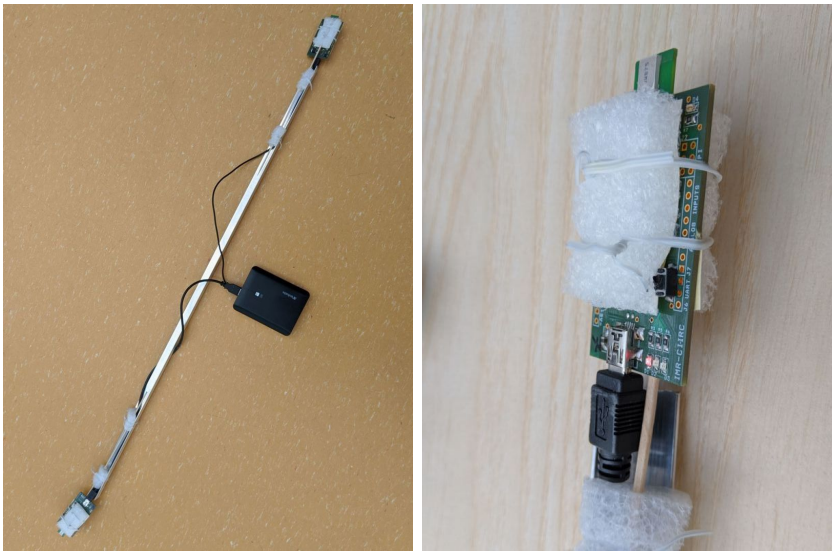
Na modelech helikoptér Tello jsme otestovali algoritmus pro udržování trojúhelníkové formace založený na článku [SAPG12], jehož funkci jsme si osvětlili v kapitole 1.3. Implementace obsahuje řadu modifikací popsaných v kapitole 1.3.1 umožňujících mj. použití pouze jednoho vůdce a několika sledujících agentů.

Na tělo vůdce je třeba umístit dva senzory vzdálenosti s definovaným rozpětím w. Za tímto účelem jsem vyhotovil konstrukci z hliníkového profilu (viz obr 6.7). Její nízká hmotnost umožňuje zavěšení na podvozek větších modelů helikoptér. Návrh je vytvořen pro použití s konkrétním letounem v laboratoři vedoucího práce. Drony Tello, které mám k dispozici na testování, však nejsou žádné další zátěže schopny. V rámci experimentu jsem proto konstrukcí manipuloval manuálně a simuloval roli vůdce. Rozpětí bylo stanoveno

$$w = 94\,\text{cm}. \tag{6.3}$$

Experiment probíhal dle schématu na obr. 6.8. Sledující letouny měří vzdálenost k senzorům vůdce, výsledky posílají do řídicího PC, kde běží algoritmus udržování formace. Počítač zajišťuje ovládání dronů. Popíšeme si dva ukázkové lety. Oba jsou dokumentovány videozáznamem přiloženým k diplomové práci (odkaz v příloze A).

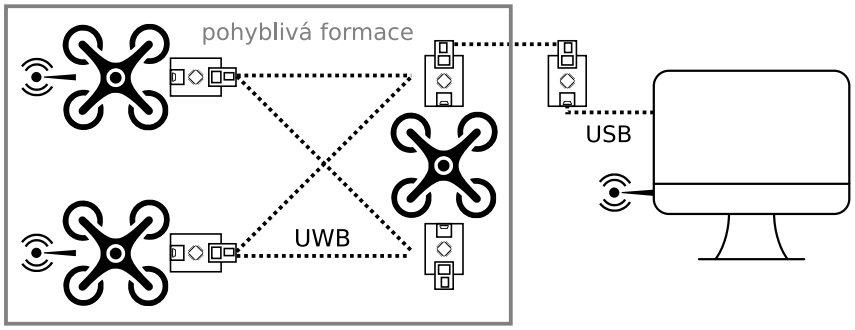
Při prvním letu byl použit jeden vůdčí a jeden sledující agent. Dráha letu byla zvolena tak, aby demonstrovala schopnost algoritmu adaptovat se na změnu směru. Připomínám, že



(a) : Hliníkový profil a zdroj napájení

(b) : Detail uchycení zabraňující zkratu

Obrázek 6.7: Konstrukce pro uchycení senzorů na vůdčí helikoptéru

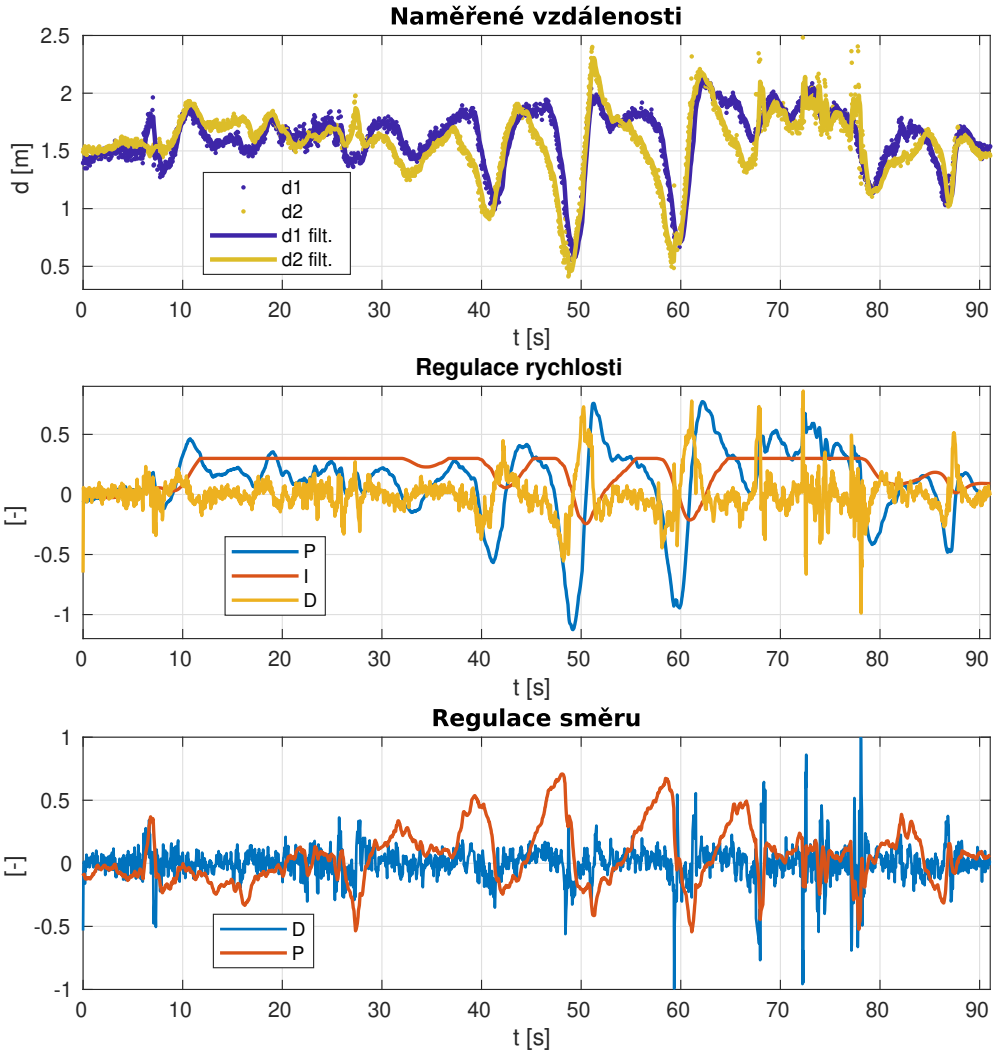


Obrázek 6.8: Infrastruktura systému při testu udržování trojúhelníkové formace na modelech helikoptér

teoreticky návrh počítá pouze s přímočarým pohybem konstantní rychlostí (přičemž moje modifikace umožňují zastavení celého uskupení). Data z průběhu letu zaznamenává obr. 6.9. Pro lepší názornost doporučuji zhlédnout přiložené video (odkaz v příloze A). Graf ukazuje, že prvních 30 sekund letu byly vzdálenosti k oběma senzorům vůdce úspěšně regulovány na nastavenou vzdálenost $d_1 = d_2 = 1,5\,\mathrm{m}$. Následovalo zpomalení vůdce a provedení pozvolného obratu v místnosti o 180°. Vzdálenost od vůdce v průběhu točení kolísala až o $\pm 1\,\mathrm{m}$. Algoritmus však poměrně prudký oblouk akceptoval. Po návratu k přímému pohybu a opětovnému zrychlení vůdce se vzdálenost stabilizovala na požadované hodnotě.

První graf v obr. 6.9 ukazuje naměřené vzdálenosti k oběma senzorům vůdce a průběh vyhlazený klouzavým průměrem délky $n = 8$. Druhý a třetí graf zaznamenává hodnoty regulačních zásahů pro rychlost (osa *pitch*) a změnu směru. Změna směru řídí posun do strany *roll* a po přenásobení koeficientem $k = 0,8$ také rotaci *yaw*. Součet složek P, I, D udává akční zásah regulátoru v rozmezí $-1 < u < 1$.

Průměrná hodnoat chyby tvaru formace mimo úsek oblouku, tedy v časech $t < 40\,\mathrm{s}$ a



Obrázek 6.9: Výsledky experimentu s udržováním trojúhelníkové formace, první let.

$t > 75\,\mathrm{s}$, dosahovala

$$\epsilon_{avg} = 25,8\,\mathrm{cm}, \qquad \delta_{avg} = 11,4\,\mathrm{cm}. \tag{6.4}$$

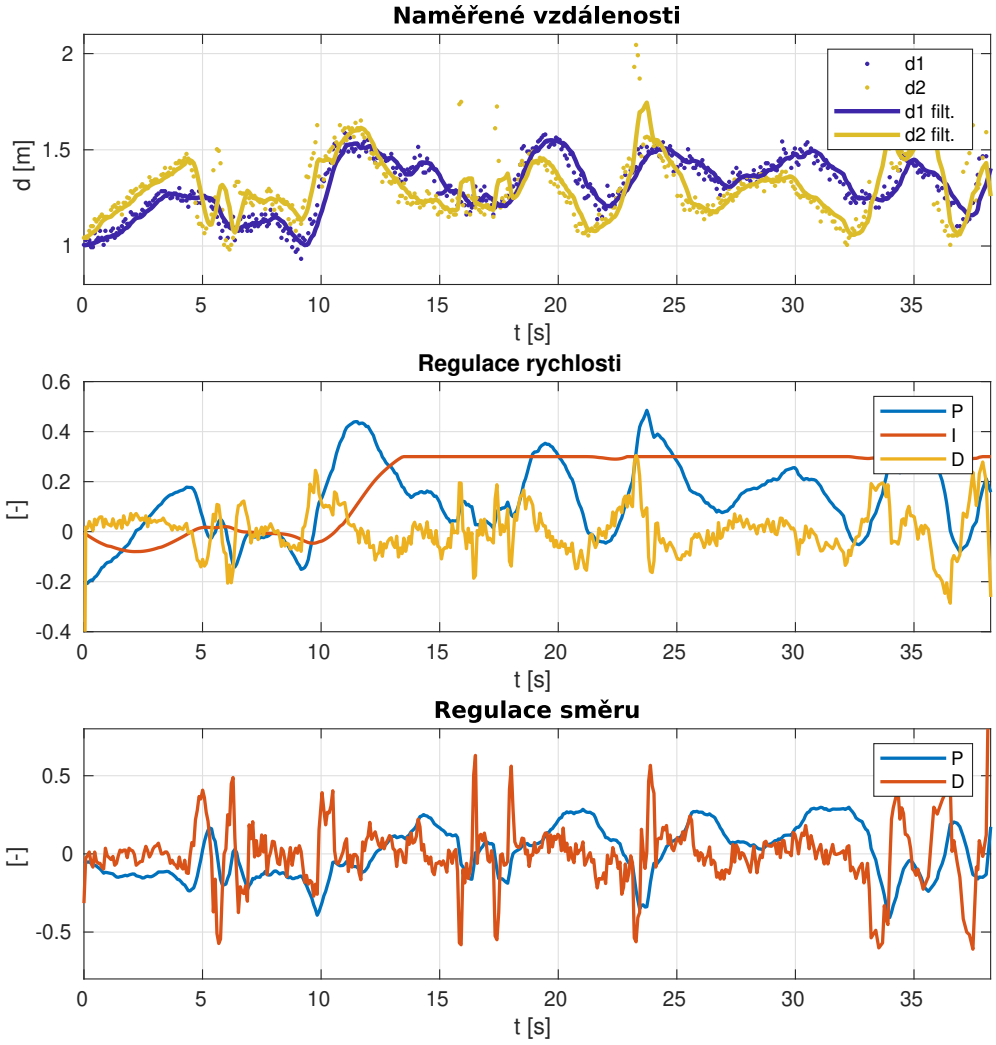
Časový úsek byl stanoven přibližně. Ve zbytku intervalu (po čas otáčení) byly hodnoty

$$\epsilon_{avg} = 68,2\,\mathrm{cm}, \qquad \delta_{avg} = 19,5\,\mathrm{cm}. \tag{6.5}$$

Při druhém letu demonstrujeme možnost zavěšení více sledujících agentů. V tomto případě jeden dron udržuje vzdálenosti $d_1 = d_2 = 1,2\,\mathrm{m}$ a druhý dron $d_1 = d_2 = 2,2\,\mathrm{m}$. Záznamy letových dat přináší obrázky 6.10 a 6.11. Opět doporučuji zhlédnout videozáznam experimentu (odkaz v příloze A). Trajektorie v tomto případě zahrnovala jen pozvolné oblouky, jednalo se o přelet z jednoho konce sálu na druhý s rozjezdem a následným zastavením formace.

Průměrná perioda jednoho cyklu vzrostla z

$$T_{1_{avg}} = 41\,\mathrm{ms} \tag{6.6}$$



Obrázek 6.10: Výsledky experimentu s udržováním trojúhelníkové formace. Druhý let, první agent.

na

$$T_{1_{avg}} = 86\,\text{ms}. \tag{6.7}$$

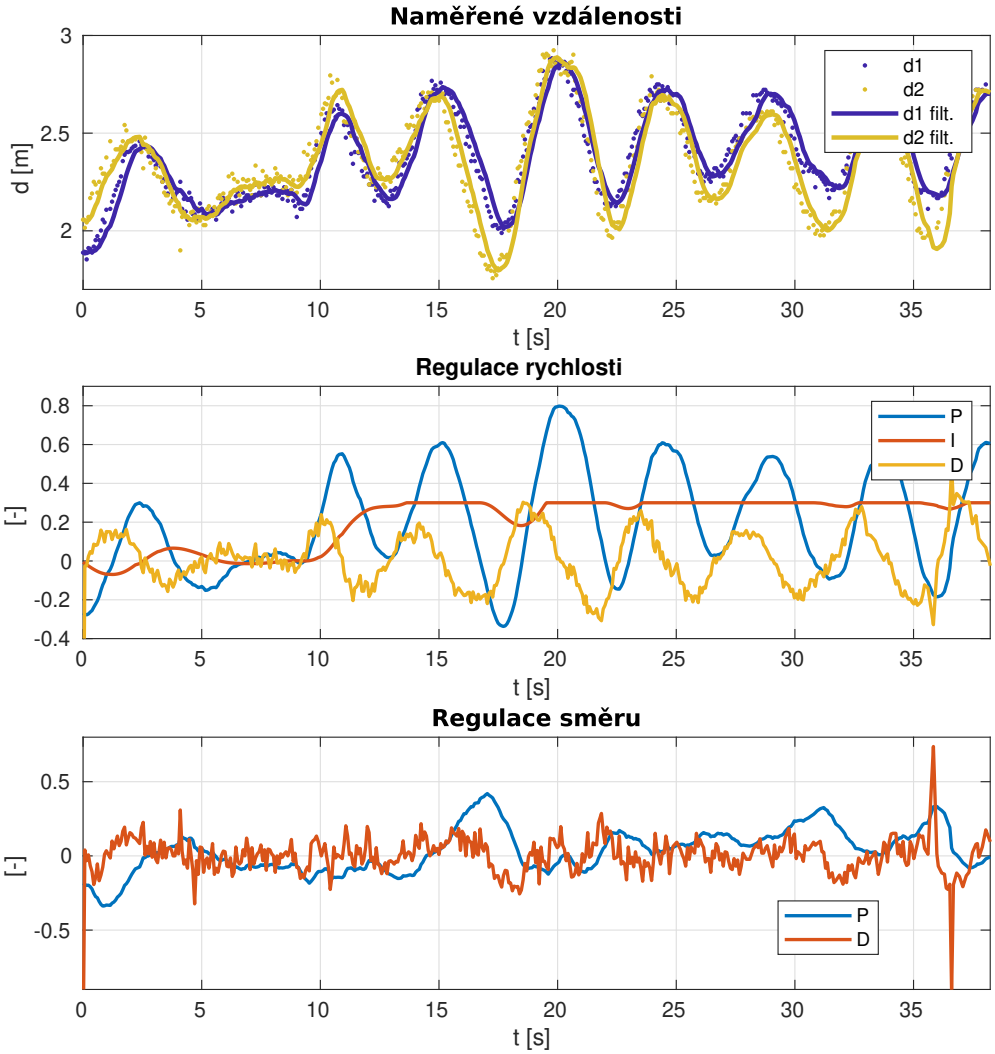
Grafy zároveň dokazují horší schopnost regulovat polohu agenta více vzdáleného od vůdce. Pro zajištění stability bylo potřeba u druhého letounu snížit konstanty k regulaci změny směru. To se projevilo na vyšších výkyvech v naměřené vzdálenosti v obr. 6.11 oproti 6.10. Průměrná hodnota chyb tvaru formace dosáhla pro prvního sledujícího agenta

$$\epsilon_{avg} = 26,5\,\text{cm}, \qquad \delta_{avg} = 12,1\,\text{cm} \tag{6.8}$$

a pro druhého (vzdálenějšího)

$$\epsilon_{avg} = 43,2\,\text{cm}, \qquad \delta_{avg} = 10,2\,\text{cm}. \tag{6.9}$$

Nižší hodnota δ_{avg} druhého agenta neznamená, že by se méně vychyloval z trajektorie, ale s větší vzdáleností má stejné vychýlení menší vliv na změnu δ. Proto je regululace obtížnější.



Obrázek 6.11: Výsledky experimentu s udržováním trojúhelníkové formace. Druhý let, druhý agent.

6.3.1 Shrnutí výsledků

Experimentálně jsme dokázali schopnost algoritmu následovat vůdce s jedním i více sledujícími agenty. Všechny navržené modifikace se ukázaly jako výhodné, s formací bylo možné se zastavit a opětovně pokračovat v posunu. Ovládání osy *roll* umožnilo regulaci výchylky do stran i s velmi nízkou dopřednou rychlostí.

Sledující agenti nemají potíže udržet trend vůdce při přímém pohybu vpřed nebo pozvolné změně směru. Dochází však k výchylkám z požadované pozice a při návrhu tvaru formace je nutné dodržet bezpečnou vzdálenost mezi agenty. Podařilo se dokonce vykonat ostřejší oblouk a otočit se s formací v místnosti do protisměru. V průběhu oblouku bylo třeba postupovat rychlostí velmi pomalé chůze, aby algoritmus výrazná změna směřování nedestabilizovala.

Závěr

V práci jsme se věnovali udržování formací UAV na základě měření vzájemných vzdáleností. Vyvinuli jsme senzor postavený na modulu DecaWave DWM1000 a následně jej použili k ověření navržených algoritmů. Představen byl systém pro udržování trojúhelníkové formace helikoptér a také systém pro lokalizaci mobilních agentů s pevnými kotvami.

Seznámili jsme se s řadou existujících metod pro udržování formací. Hledali jsme vhodný způsob řízení skupiny autonomních letounů s minimální senzorovou výbavou. Využít jsme chtěli pouze informace o vzájemné vzdálenosti agentů. Jako nejvhodnější se jevilo použití algoritmu k udržování trojúhelníkové formace dle článku [SAPG12]. Klade striktní požadavky na tvar formace a charakter pohybu, ale splňuje naše nároky a je podložen simulací. Objasnili jsme si princip algoritmu a představili několik modifikací. Navržené modifikace umožňují mj. použít pouze jednoho vůdce namísto dvou a několik sledujících agentů namísto jednoho. Zároveň dovolují celou formaci zastavit a opětovně uvést v pohyb, původní návrh počítá pouze s rovnoměrným přímočarým pohybem vpřed.

Princip algoritmu jsme nejprve ověřili v simulaci a následně experimentálně s modely helikoptér Ryze Tello. Součástí práce jsou záznamy regulovaných veličin z testovacích letů a také videodokumentace (odkaz v příloze A). Experimenty dokazují schopnost metody následovat vůdce po neznámé trajektorii. Regulace je nejstabilnější, umístíme-li agenty přímo za vůdce (formace není příliš široká). Řiditelnost klesá také s velkou vzdáleností mezi agenty (formace nesmí být ani příliš dlouhá). V experimentech jsme vybavili vůdce dvěma senzory v rozpětí $w = 94\,\mathrm{cm}$ a sledující agenti se pohybovali ve vzdálenosti $1,2\,\mathrm{m}$ až $2,2\,\mathrm{m}$. Formace je zachována i při změně směru, není-li oblouk příliš ostrý a rychlost v zatáčce příliš vysoká. Lepší představu o schopnosti adaptovat se na změnu směřování poskytne videozáznam.

Zabývali jsme se také metodami lokalizace mobilních agentů. Představili jsme systém, jehož vstupní veličinou jsou opět vzájemné vzdálenosti. Několik členů systému (nazýváme je kotvami) má předem známou polohu. Odhad umístění ostatních účastníků provádíme za pomoci EKF (*Extended Kalman Filter*). Kotvy mohou být reprezentovány několika letouny s lepší senzorovou výbavou a přitom definovat souřadný systém menším helikoptérám. Ty v souřadném systému zaujmou definovanou polohu. Při experimentech jsme však narazili na limity modulů DWM1000. K přesnému měření je nutné zabránit odrazům signálu od rušivých elementů. Umístění několika helikoptér Tello se senzory do menšího prostoru mezi kotvami způsobilo nárůst množství odrazů nad únosnou mez a lokalizace přestávala fungovat. Navržený systém je však plně použitelný pro účely lokalizace libovolných objektů. Zajistíme-li přímou viditelnost kotev, můžeme počítat s přesností lokalizace v rovině až $10\,\mathrm{cm}$.

Příloha A

CD

K práci je přiloženo CD s následujícími soubory:

- experiment1.mp4 - videozáznam prvního experimentu s udržováním trojúhelníkové formace, dostupný ke dni odevzdání také na `https://youtu.be/WyMdRwIprwI`.
- experiment2.mp4 - videozáznam druhého experimentu s udržováním trojúhelníkové formace, dostupný ke dni odevzdání také na `https://youtu.be/HflXt1iOiTQ`.
- atmega_firmware.zip - zdrojové kódy firmwaru senzoru vzdálenosti
- knihovna_pc.zip - zdrojové kódy knihovny pro řídicí počítač

Soubory jsou zároveň odevzdány jako přílohy do systému KOS, kromě videozáznamů, které není možné nahrát.

Příloha B

Literatura

[AY11] B. D. O. Anderson and C. Yu. Range-only sensing for formation shape control and easy sensor network localization. In *2011 Chinese Control and Decision Conference (CCDC)*, pages 3310–3315, 2011.

[BKAM13] Yasmine Kheira Benkouider, Mokhtar Keche, and Karim Abed-Meraim. Divided difference kalman filter for indoor mobile localization. In *International Conference on Indoor Positioning and Indoor Navigation*, pages 1–8, 2013.

[CYA11] Ming Cao, Changbin Yu, and Brian D.O. Anderson. Formation control using range-only measurements. *Automatica*, 47(4):776–781, 2011.

[Dec14a] Decawave Ltd. *APS003 Application note: Real Time Location Systems, An Introduction*, 2014. verze 1.1.

[Dec14b] Decawave Ltd. *APS017 Application Note: MAXIMISING RANGE IN DW1000 BASED SYSTEMS*, 2014. verze 1.1.

[Dec15] Decawave Ltd. *DW1000 Datasheet*, 2015. verze 2.09.

[Dec16] Decawave Ltd. *DWM1000 Datasheet*, 2016. verze 1.8.

[Dec17] Decawave Ltd. *DW1000 USER MANUAL*, 2017. verze 2.18.

[Dec21] Decawave Ltd. Dwm1000 module. `https://www.decawave.com/product/dwm1000-module/`, 2021. Navštíveno: 7. dubna 2021.

[DJI19] DJI. Dji forum - whats the difference between tello and tello edu. `https://forum.dji.com/thread-188227-1-1.html`, 2019. Navštíveno: 4. května 2021.

[F-A21] F-Army. arduino-dw1000-ng library. `https://github.com/F-Army/arduino-dw1000-ng`, 2021. Navštíveno: 27. října 2020.

[GIS21] GISGeography. How gps receivers work – trilateration vs triangulation. `https://gisgeography.com/trilateration-triangulation-gps/`, 2021. Navštíveno: 6. května 2021.

[GJ+10] Gaël Guennebaud, Benoît Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010. Navštíveno: prosinec 2020.

[HADB07] Julien M. Hendrickx, Brian D. O. Anderson, Jean-Charles Delvenne, and Vincent D. Blondel. Directed graphs for the analysis of rigidity and persistence in autonomous agent systems. *International Journal of Robust and Nonlinear Control*, 17(10-11):960–981, 2007.

[Kal60] R. E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, 82(1):35–45, 03 1960.

[KBF09] Laura Krick, Mireille E. Broucke, and Bruce A. Francis. Stabilisation of infinitesimally rigid formations of multi-robot networks. *International Journal of Control*, 82(3):423–439, 2009.

[KPLA14] S. Kang, M. Park, B. Lee, and H. Ahn. Distance-based formation control with a single moving leader. In *2014 American Control Conference*, pages 305–310, 2014.

[Nic12] Nick Gammon. Atmega bootloader programmer. `http://www.gammon.com.au/bootloader`, 2012. Navštíveno: 10. října 2020.

[OA17] Kwang-Kyo Oh and Hyo-Sung Ahn. Leader-follower type distance-based formation control of a group of autonomous agents. *International Journal of Control, Automation and Systems*, 15:1738–1745, 2017.

[OPA15] Kwang-Kyo Oh, Myoung-Chul Park, and Hyo-Sung Ahn. A survey of multi-agent formation control. *Automatica*, 53:424–440, 2015.

[Pin] PingguSoft. pytello. `https://bitbucket.org/PingguSoft/pytello/src/master/`. Navštíveno: 4. prosince 2020.

[RSH17] Matthew B Rhudy, Roger A Salguero, and Keaton Holappa. A kalman filtering tutorial for undergraduate students. *International Journal of Computer Science & Engineering Survey*, 8(1):1–18, 2017.

[RSP+14] Francisco Rego, Jorge M. Soares, António Pascoal, A. Pedro Aguiar, and Colin Jones. Flexible triangular formation keeping of marine robotic vehicles using range measurements. *IFAC Proceedings Volumes*, 47(3):5145–5150, 2014. 19th IFAC World Congress.

[Ryz] Ryze Tech. TELLO SDK 1.0.0.0. `https://dl-cdn.ryzerobotics.com/downloads/tello/0228/Tello+SDK+Readme.pdf`. Rok ani verze neuvedena. Navštíveno: 4. května 2021.

[Ryz18] Ryze Tech. TELLO SDK 2.0 User Guide. `https://dl-cdn.ryzerobotics.com/downloads/Tello/Tello%20SDK%202.0%20User%20Guide.pdf`, 2018. verze 1.0.

[Ryz20] Ryze Tech. Tello technical specifications. `https://www.ryzerobotics.com/tello/specs`, 2020. Navštíveno: 4. května 2021.

[SAPG12] Jorge M. Soares, A. Pedro Aguiar, António M. Pascoal, and Marco Gallieri. Triangular formation control using range measurements: An application to marine robotic vehicles. *IFAC Proceedings Volumes*, 45(5):112–117, 2012. 3rd IFAC Workshop on Navigation, Guidance and Control of Underwater Vehicles.

[Tel20] TelloPilots. Low-level protocol. `https://tellopilots.com/wiki/protocol/`, 2020. Navštíveno: 10. listopadu 2020.

[Wik21] Wikipedia.org. Aircraft principal axes. `https://en.wikipedia.org/wiki/Aircraft_principal_axes`, 2021. Navštíveno: 4. května 2021.

[Zig21] Zigbee Alliance. Zigbee alliance homepage. `https://zigbeealliance.org/`, 2021. Navštíveno: 13. dubna 2021.

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Gärtner** Jméno: **Jan** Osobní číslo: **466304**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra řídicí techniky**

Studijní program: **Kybernetika a robotika**

Studijní obor: **Kybernetika a robotika**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Udržování formace UAV na základě měření vzájemných vzdáleností

Název diplomové práce anglicky:

UAV formation keeping based on mutual distance measurement

Pokyny pro vypracování:

1) Prostudujte metody udržování formací autonomních létajících robotů (UAV) využívající měření vzájemných vzdáleností.
2) Seznamte se s rádiovými moduly DWM1000 umožňujícími měření vzdálenosti a zhodnoťte jejich použitelnost pro daný účel.
3) Zvolte existující nebo navrhněte vlastní metodu, která bude vhodná pro budoucí realizaci na laboratorních modelech UAV. Tuto metodu implementujte a otestujte podle možností v simulaci nebo s reálnými modely multi-koptér.
V průběhu řešení problému je přípustné po konzultaci s vedoucím práce definovat omezující podmínky úlohy s ohledem na řešitelnost.
4) Vyhodnoťte, za jakých podmínek je možné formaci udržovat.

Seznam doporučené literatury:

[1] Kwang-Kyo Oh, Myoung-Chul Park, Hyo-Sung Ahn, A survey of multi-agent formation control, Automatica, Volume 53, 2015
[2] Jorge M. Soares, A. Pedro Aguiar, António M. Pascoal, Marco Gallieri,Triangular formation control using range measurements: An application to marine robotic vehicles, IFAC Proceedings Volumes, Volume 45, Issue 5, 2012

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Jan Chudoba, inteligentní a mobilní robotika CIIRC

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **29.01.2021** Termín odevzdání diplomové práce: **21.05.2021**

Platnost zadání diplomové práce:
do konce letního semestru 2021/2022

Ing. Jan Chudoba	prof. Ing. Michael Šebek, DrSc.	prof. Mgr. Petr Páta, Ph.D.
podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

. _____
Datum převzetí zadání

_____
Podpis studenta