VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY
DEPARTMENT OF CONTROL AND INSTRUMENTATION

ŘÍZENÍ ROBOTICKÉ SEKAČKY TRÁVY
CONTROL OF A ROBOTIC LAWNMOWER

DIPLOMOVÁ PRÁCE
MASTER‘S THESIS

AUTOR PRÁCE
AUTHOR
Bc. Antonín Škapa

VEDOUCÍ PRÁCE
SUPERVISOR
Ing. Tomáš Jílek Ph.D.

BRNO 2020

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Kybernetika, automatizace a měření**

Ústav automatizace a měřicí techniky

Student: Bc. Antonín Škapa *ID:* 172136
Ročník: 2 *Akademický rok:* 2019/20

NÁZEV TÉMATU:

Řízení robotické sekačky trávy

POKYNY PRO VYPRACOVÁNÍ:

Cílem diplomové práce je návrh a realizace řízení robotické sekačky trávy využívající RTK GNSS. Pro realizaci práce bude využita komerčně dostupná mechanická platforma sekačky. Předpokládá se využití aktuálně dostupných open-source platforem a knihoven pro řízení mobilních robotů. Práce zahrnuje výběr a ověření parametrů vhodného levného RTK GNSS přijímače.

1. Proveďte rešerši dostupných open-source platforem pro řízení mobilních robotů.
2. Proveďte rešerši dostupných GNSS přijímačů s cenou do 20 tis. Kč.
3. Vyberte vhodnou mechanickou platformu sekačky.
4. Experimentálně ověřte vlastnosti vybraného GNSS přijímače v dané aplikaci.
5. Navrhněte kompletní hardware pro řízení sekačky a vyberte potřebné komponenty.
6. Navrhněte a implementujte software pro řízení sekačky.
7. Realizujte automatickou navigaci a plánování trajektorie pro pohyb sekačky po zahradě.
8. Ověřte funkčnost celého řešení.

DOPORUČENÁ LITERATURA:

BRÄUNL, Thomas. Embedded Robotics. Berlin/Heidelberg: Springer-Verlag, 2006. ISBN 3-540-34318-0.

Termín zadání: 3.2.2020 *Termín odevzdání:* 1.6.2020

Vedoucí práce: Ing. Tomáš Jílek, Ph.D.

doc. Ing. Václav Jirsík, CSc.
předseda oborové rady

Abstrakt

Tato diplomová práce se zabývá vývojem a realizací robotické sekačky se satelitní navigací. Začíná se přípravou platformy pro navigaci mobilního robotu ve venkovním prostředí, návrhem řídicího HW a SW a realizací daného řešení. Jsou zde diskutovány možnosti navigace, jak používané v komerčním sektoru, tak i experimentální. Dále je na základě průzkumu trhu a uživatelských zkušeností vybrán vhodný GNSS přijímač a provedeno jeho měření s různými anténami. Poté je vybrána vhodná open-source řídicí jednotka a je implementován její software. Následně je popsáno řízení z nadřazeného systému palubního počítače a je provedena fyzická realizace. Na konci práce je provedeno oživení a nastíněn další možný vývoj.

Klíčová slova

GNSS, GPS, RTK, L1/L2, ZED-F9P, Pixhawk 2 Cube, ArduRover, navigace mobilního Robotu, ROS, DroneKit

Abstract

This master‘s thesis deals with development and realisation of robotic lawn mower with satelite navigation. It begins with preparation of a platform for outdoor mobile robot navigation and it’s control HW and SW. There are discussed different options of navigation both commercial and experimental. Further on I have chosen the right GNSS receiver based on market research and user experience. The GNSS receiver’s parameters are measured with different antennas. Following with the choice of suitable open-source control unit and it’s software implementation. Furthermore control from a companion computer is described and physical realisation is done. In the end of the thesis activation of the whole mower is performed and described. Lastly there are discussed possible ways of future development.

Keywords

GNSS, GPS, RTK, L1/L2, ZED-F9P, Pixhawk 2 Cube, ArduRover, mobile robot navigation, ROS, DroneKit

ŠKAPA, Antonín. *Řízení robotické sekačky trávy* [online]. Brno, 2020 [cit. 2020-06-01]. Dostupné z: https://www.vutbr.cz/studenti/zav-prace/detail/127102. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky. Vedoucí práce Tomáš Jílek.

Prohlášení

Prohlašuji, že svoji diplomovou práci na téma „Řízení robotické sekačky trávy“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce. Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne .....

.....

(podpis autora)

Poděkování

Děkuji vedoucímu diplomové práce Ing.Tomáši Jílkovi,Ph.D. za odborné vedení, konstruktivní kritiku a účelné rady.

Obsah

Seznam obrázků

Seznam tabulek

Seznam rovnic

1 ÚVOD

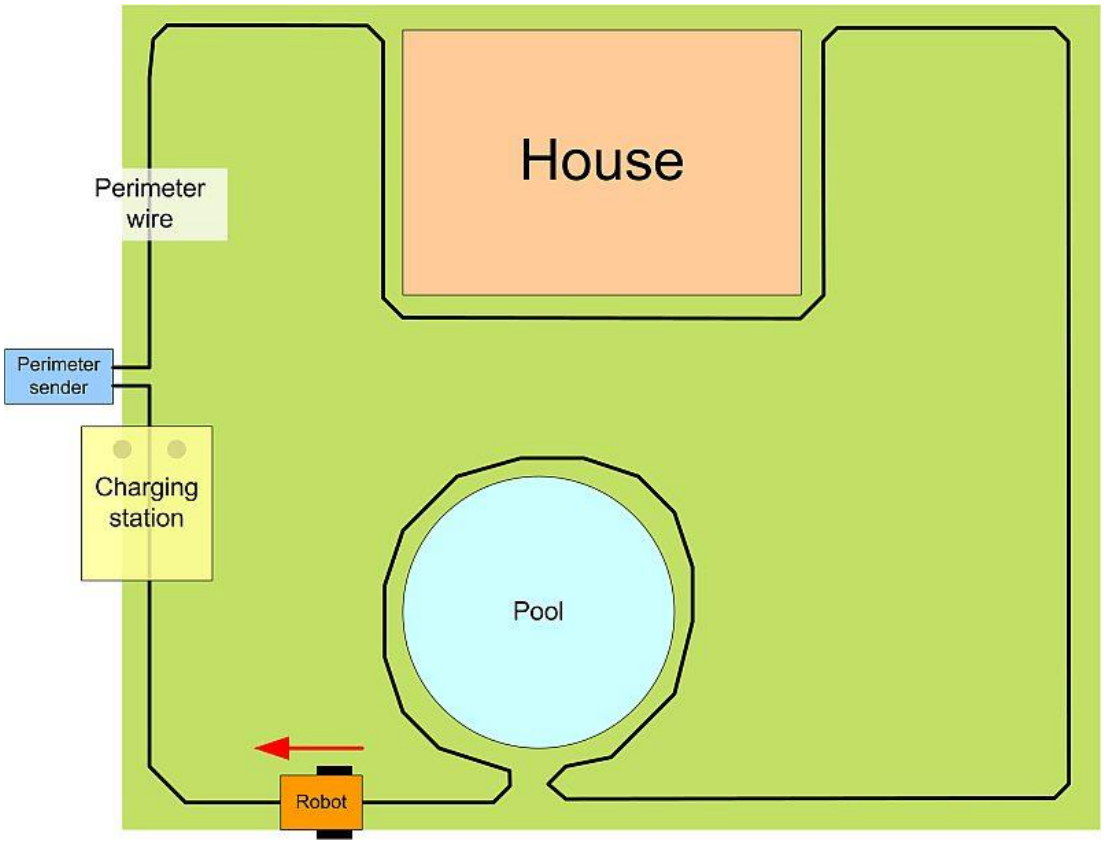
V mé diplomové práci navazuji na semestrální práci zabývající se přípravou platformy pro venkovní robot. Motivací celé práce bylo získat spolehlivou zahradní robotickou sekačku s přesnou lokalizací a možností přizpůsobení. Stroj požadovaných parametrů není na trhu dostupný, a i nejlepší modely zavedených značek zdaleka nesplňují nároky dané aplikace. Budoucí pracovní plocha se vyznačuje několika zónami, mezi kterými musí sekačka přejíždět, mnoha stromy a keři, kterým je třeba se vyhnout a dalšími potenciálními problémy. Rozhodl jsem se tedy sestrojit sekačku naváděnou přesnou GNSS navigací, která všechny tyto nástrahy zvládne. Začínám všeobecným průzkumem možností používaných pro lokalizaci malých zahradních strojů, jak komerčními výrobci, tak i experimentálními možnostmi lokalizace. Dále jsem provedl průzkum trhu dostupných dvoupásmových GNSS přijímačů, včetně uživatelských zkušeností a možnosti přizpůsobení a integrace do mojí aplikace. Poté popisuji mnou sestrojený přístroj na měření GNSS přijímačů a je zjednodušeně naznačeno fungování lokalizace s pomocí korekčních zpráv. Následují porovnání naměřených dat z referenčních GNSS přijímačů a vybraného ZED-F9P. Jednotlivé trajektorie jsou po vyhodnocení a filtraci v MATLABu přeloženy přes sebe a jsou zdůrazněny zajímavé části. Zmíněn je také obslužný software pro PC, a nakonec vybrána mechanická platforma pro realizaci. Zabývám se také hardwarem a sofwarem potřebným pro realizaci, a to jednotkou Pixhawk 2 Cube včetně jejího softwaru, Raspberry Pi, regulátory motorů a vysokoúrovňovým řízením. V tomto bodě jsem narazil na to, že je možné se vydat dvěma různými směry a problém pojmout z různých úhlů. Obě možná řešení jsou diskutována, detailně rozebrána a je naznačen postup a úskalí každého z nich. Na konci je ukázán způsob plánování trajektorie a ověřena funkčnost celého navrženého řešení. V průběhu návrhu a realizace HW a SW bylo několikrát změněno původně zamýšlené řešení, jelikož se empiricky zjišťovaly limity použité elektroniky, nepředvídatelná nekompatibilita softwaru zamýšleného pro použití v Raspberry Pi až po řešení, která „na papíře“ vypadala výborně, v praxi se ukázala jejich nedostatečná robustnost. Nejdůležitější zvraty a problémy, na které jsem narazil jsou v práci popsány. Přes všechny překážky se výsledná realizace podařila a robotická sekačka je již funkční.

2 LOKALIZACE ZAHRADNÍCH STROJŮ VE VENKOVNÍM PROSTŘEDÍ

Lokalizace zahradních strojů se dá pomyslně rozdělit na dvě kategorie. „Přesná“ a přibližná. Kdy v prvním případě se snažíme co nejpřesněji určit polohu stroje v pracovním prostředí například pomocí GNSS, LIDARu, nebo počítačového vidění. Pokud pracujeme s přibližnou polohou – tzn. stačí nám vědět, že se stroj nachází kdekoliv v pracovní oblasti (trávník) a ne mimo ni (pískoviště, chodník), tak lze použít metody jako vodící drát, nebo počítačové vidění (v jednodušší implementaci, kde například pouze vyhodnocujeme texturu povrchu okolo stroje).

2.1 Vodící drát

Navigace pomocí vodícího drátu je nejčastěji používaná metoda navigace. Používá se také zkratka BWF, což značí „burried wire fence“, tedy ohraničení pomocí drátu skrytého pod povrchem. Použití ilustruje obrázek č.1 níže. Pomocí drátu vyznačíme plochu, kde chceme, aby se zahradní stroj pohyboval a oba konce zakončíme ve vysílači hraničního signálu. Vysílač pouští do drátu pulzy o frekvenci zhruba 5–44 kHz, dle výrobce. V robotu jsou pak většinou integrovány dvě cívky (může však být i jedna), pomocí kterých detekujeme jízdu po vodícím drátu – drát je mezi cívkami. Situaci, kdy jedna, nebo druhá přejede nad vodícím drátem detekujeme jako změnu polarity na dané cívce a tím víme, zda robot směřuje do anebo ven z ohraničeného prostoru. Na vodícím drátu se nachází i nabíjecí stanice a robot, který detekuje snižující se napětí v bateriích jednoduše najede na vodící drát a dojede až k nabíjecí stanici. Při normálním provozu se pouze drží uvnitř ohraničeného prostoru a pokud narazí na vodící drát, tak se pod daným úhlem vrací zpět. Toto řešení má hlavní výhody ve své nízké ceně a robustnosti detekce ohraničené plochy pro pohyb. Nevýhodou je neznalost absolutní polohy uvnitř plochy a z toho vyplývající neefektivita pohybu. Další nevýhodou, kterou často zmiňují dlouhodobí majitelé těchto strojů, je přerušení drátu z důvodu koroze, nebo mechanického poškození (přejetí auta, obdělávání zahrady). Takové přerušení se pak složitě vyhledává a zejména v případě velkých a členitých pozemků je to zdlouhavá práce.

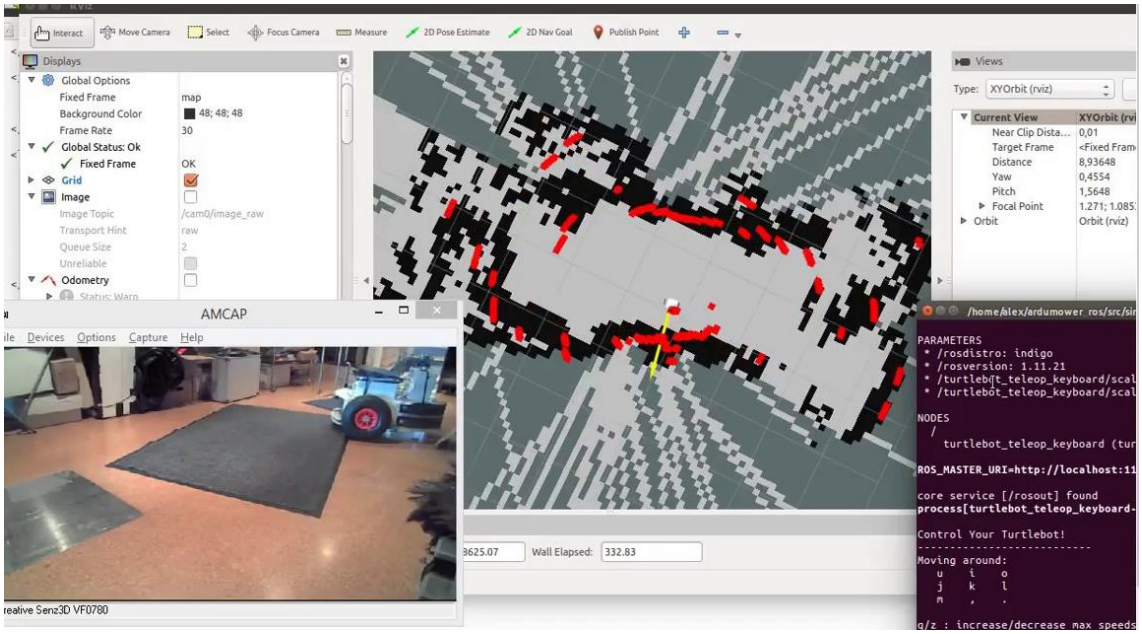


Obr. 1 Nákres navigace pomocí vodícího drátu [1]

2.2 LIDAR

LIDAR znamená „LIght Detection And Ranging). Je to metoda měření vzdálenosti podle doby šíření laserového paprsku mezi snímačem a objektem od kterého se odrazí zpět ke snímači. Obvykle se využívá spektra 1064-1540 nm. [2] Výstupem LIDARu je tzv. „point cloud“ – mračno bodů, ze kterého můžeme rekonstruovat 3D model okolí snímače. Výhodou je možná navigace i uvnitř budov a reakce na aktuální stav okolí (přesunuté předměty, nebo osoby v trajektorii lze snadno bezdotykově detekovat). Nevýhodou je vysoká cena použitelných sensorů a složitá implementace.

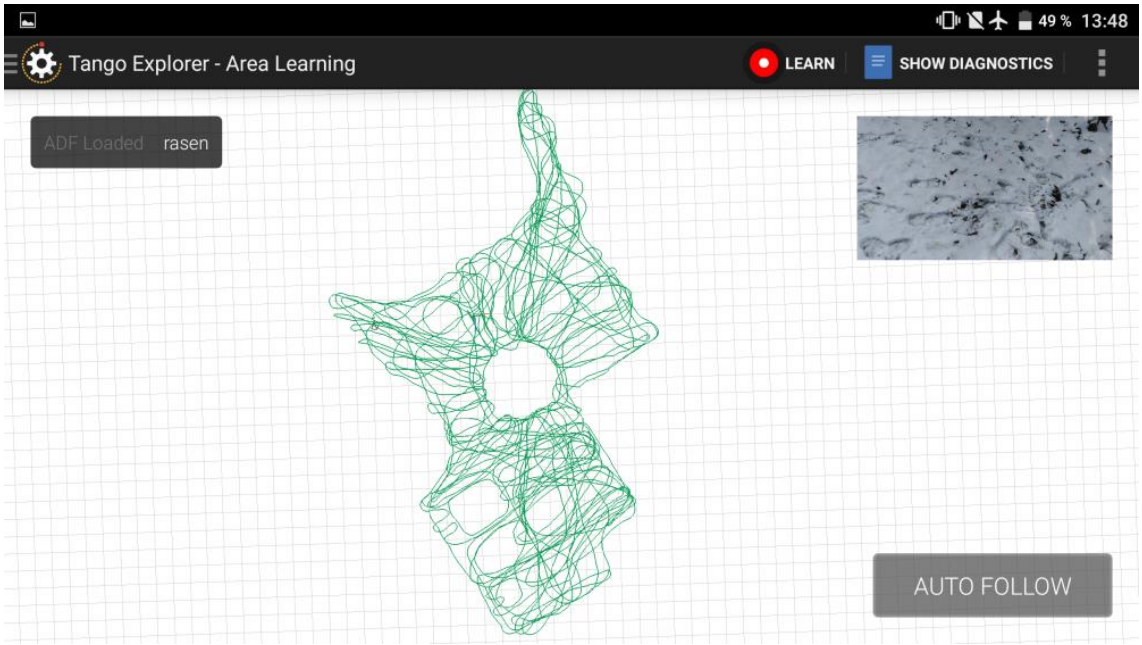
Jediné zadokumentované použití LIDARu v navigaci malých zahradních strojů je doposud pouze použití s ROS na platformě ArduMower. Zařízení je pouze ve fázi prototypu a žádný renomovaný výrobce LIDAR ve svých strojích zatím nepoužívá.



Obr. 2 Navigace robota pomocí LIDARu [3]

2.3 Počítačové vidění [4]

Zadokomentovaná použití počítačového vidění v navigaci zahradních strojů využívají nejčastěji platformu Tango vyvinutou firmou Google. Tango je platforma pro tzv. AR – Augmented Reality, neboli rozšířenou realitu. Originálně platforma sloužila k určení relativní polohy chytrých telefonů a tabletů bez použití GPS. Díky otevřenému SDK bylo umožněno použití i pro navigaci zahradních strojů. Na začátku je nutné zmapovat pracovní oblast a určit pracovní perimetr pomocí orientačních bodů. Výstupem je pak přibližná poloha v pracovním prostoru a velice přesné určení hranic pracovního prostoru.



Obr. 3 Použití frameworku Tango k navigaci robota [4]

2.4 GNSS

Navigování zahradních strojů pomocí satelitní navigace je doménou dražších přístrojů, jako například Stiga AutoClip 550SG. Tato zahradní sekačka atakuje hranici sto tisíc korun, ale přesto, jako většina podobných zařízení, používá GNSS navigaci pouze jako doplňkový zdroj polohy, jelikož z použitého jednoduchého jednokanálového snímače bez korekcí nelze spolehlivě dostat dostatečně přesnou polohu. Poloha je proto použita například jako ochrana před zcizením, protože pokud se detekuje, že je zapnuta mimo vymezený prostor, tak se zablokuje. Další funkcí je rámcová orientace v pracovním perimetru, které levnějším modelům chybí úplně, a proto se pohybují zcela náhodně v rámci pracovního perimetru. Zahradní stroj tak může optimalizovat svůj pohyb v rámci pracovního perimetru, zejména je-li komplikovanější.

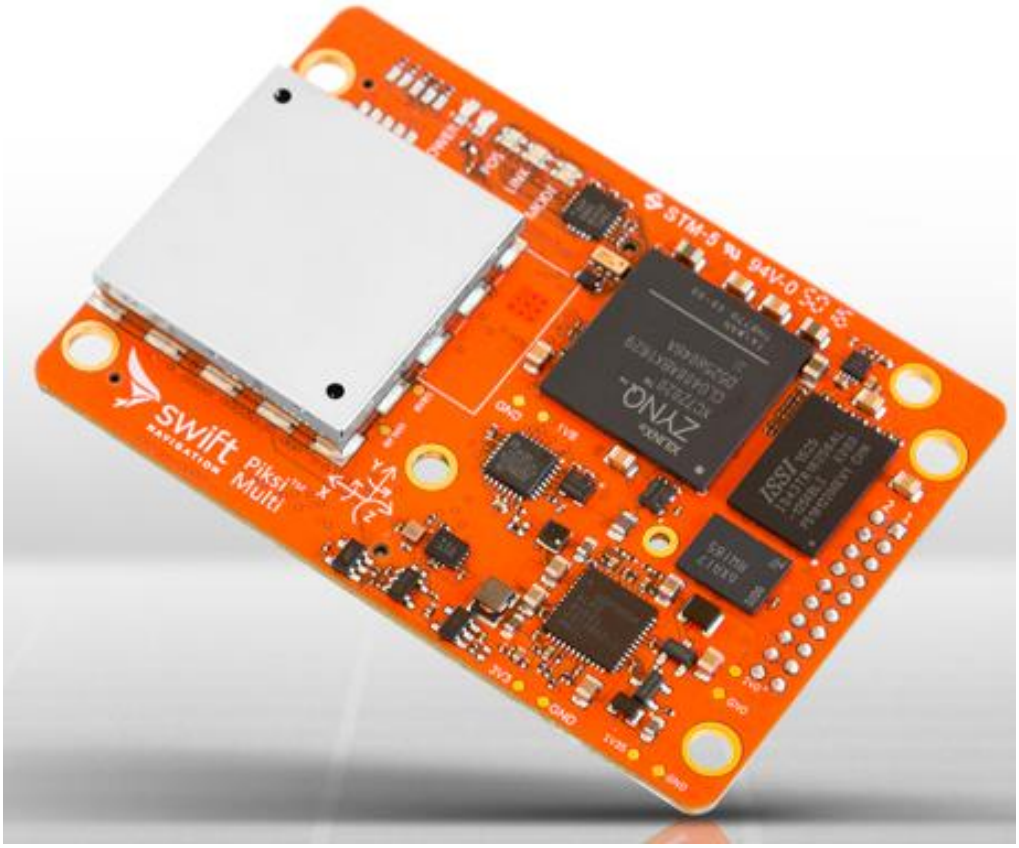
3 DOSTUPNÉ GNSS PŘIJÍMAČE

Pro potřeby spolehlivé navigace sekačky s cenově dostupnými komponenty byly v době vytváření práce na trhu pouze dva přijímače. Piksi a nový U-Blox ZED-F9P. Hlavními sledovanými parametry byly kromě ceny podpora L1+L2 pásem a možnost RTK přímo na čipu. Výhodou je pak možnost současného příjmu dalších GNSS pásem a tím zpřesnění výsledného řešení.

3.1 Piksi [5]

Multipásmový GNSS přijímač s rychlou konvergencí. Vyrábí firma Swift Navigation. Cena za jeden přijímač se pohybuje kolem 30 000 Kč.
Podporované pásma:

- GPS L1/L2.
- GLONASS G1/G2.
- BeiDou B1/B2.
- Galileo E1/E5b.
- Zabudovaný NTRIP klient.
- Větší rozměry.



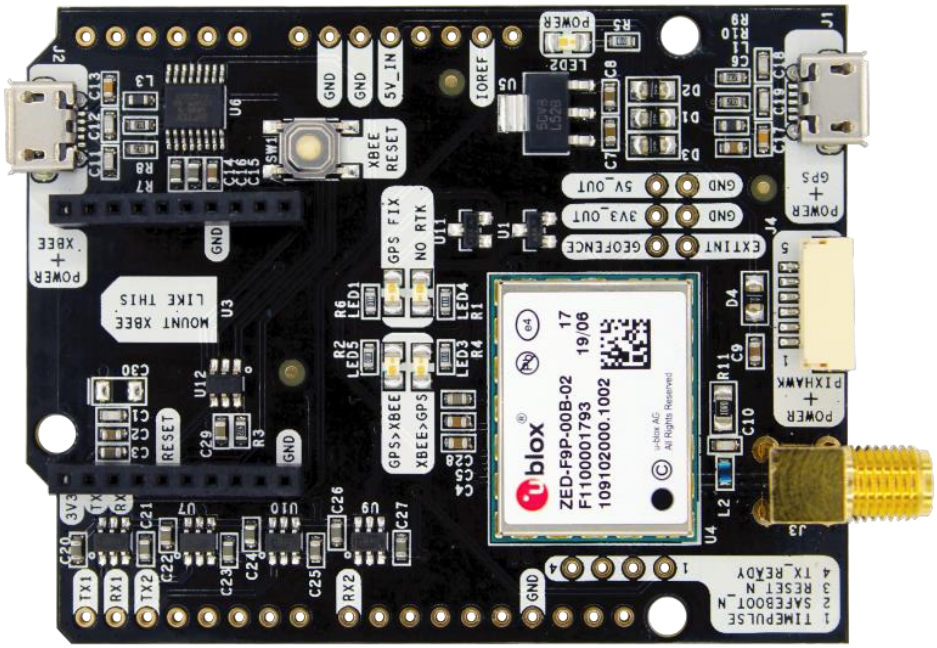
Obr. 4 GNSS přijímač Piksi [6]

3.2 U-Blox ZED-F9P [7]

Nový GNSS přijímač firmy U-Blox. Osazený na desce od firmy ArduSimple ve formě výrobku RTK2B má navíc vyvedeny potřebné konektory. Cena za jeden kus se pohybuje kolem 5000 Kč.

Podporované pásma:

- GPS L1/L2.
- GLONASS G1/G2.
- BeiDou B1/B2.
- Galileo E1/E5b.
- Možnost současného příjmu a zpracování všech podporovaných pásem.
- Možnost kalkulace natočení při použití dvou přijímačů.
- Konektor pro Pixhawk.
- Malé rozměry.
- Příznivá cena.

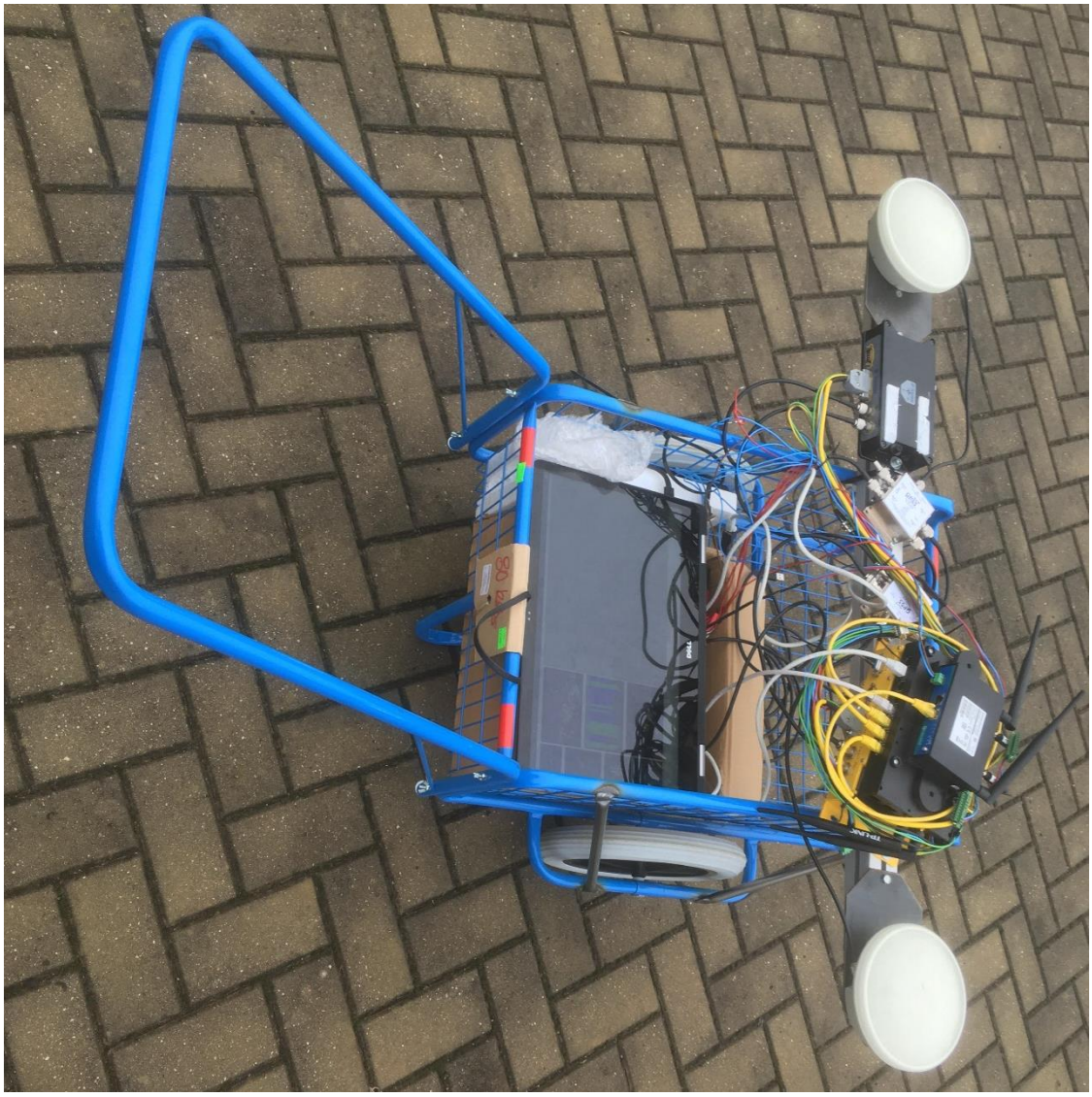


Obr. 5 GNSS přijímač ZED-F9P na desce RTK2B [8]

Vybrán byl přijímač ZED-F9P s tím, že budou v aplikaci použity dva kusy a získá se tak i přesné natočení stroje tzv. „heading“. To, že nemá integrovaného NTRIP klienta v mojí aplikaci vůbec nevadí, jelikož korekce bude dodávat RaspberryPi. Při jeho integraci bylo zjištěno, že integrace headingu je v přijímači velmi nešťastná. Každý čip je schopný upočítat pouze jedno RTK řešení v reálném čase a heading je tak vyřešen způsobem posílání RTCM korekcí z přijímače nakonfigurovaného jako „moving base“ do přijímače nakonfigurovaného jako „rover“. Toto řešení je poněkud nešťastné hned z několika důvodů. V momentě, kdy tuto funkci zapneme, tak se frekvence posílání dat zmenší až na 1 Hz, a navíc se podstatně zhorší přesnost určení pozice, čím se přijímač stane pro toto použití zcela nevhodným.

4 MĚŘENÍ PŘIJÍMAČE ZED-F9P

Měření probíhalo s pomocí přípravků připevněných na malém dvoukolovém vozíku, který umožňoval jednoduchý přesun celého aparátu a měření v různých místech. Měření probíhalo v kampusu VUT kolem budovy FEKTu a Laboratoří profesora Lista. V měření jsou zahrnuty úseky volného prostranství, úseky kolem budov, úsek se stromem a část, kde bylo měřeno vjetí do podzemních garáží (15 metrů dovnitř).



Obr. 6 Zařízení pro měření přijímačů GNSS a antén

4.1 Měřicí zařízení

Měření spočívalo v porovnávání dat z referenčních přijímačů a dvou ZED-F9P. Měření jsem opakoval s použitím několika antén pro účely srovnání.

Složení zařízení:

- Raspberry Pi.
- Trimble BD 982 + BD Trimble 992.
- Ardusimple RTK2B (ZED-F9P) + Ardusimple RTK2B lite (ZED-F9P).
- Datový modem + sim.
- Router Mikrotik.
- 2 * anténní splittery.
- Gelová 12V baterie na napájení.

4.1.1 Popis fungování

Na Raspberry Pi byl nainstalován operační systém Raspbian a nastaveno síťové připojení buď přes připojený router Mikrotik, anebo přes Wi-Fi. Veškerá funkcionalita byla realizována jako systémové služby (daemon), tak aby nebyl problém při souběhu více aplikací a bylo zajištěno správné fungování po připojení napájení bez zásahu přes SSH, nebo jiný uživatelský vstup. Správná funkce celého zařízení byla indikována integrovanou LED diodou. Korekční data pro oba přijímače Trimble jsou přijímána přes GSM modem a routována přes Mikrotik. Funkce jednotlivých daemonů byla následující:

- Logování dat z přijímače Trimble BD 982.
- Logování dat z přijímače Trimble BD 992.
- Logování dat z přijímače ZED-F9P.
- Logování dat z přijímače PIKSI (Příprava).
- Příjem korekčních dat a publikování na UART portu pro ZED-F9P.

4.2 Zaznamenávané zprávy

Přijímač ZED-F9P se dá nakonfigurovat pro vysílání mnoha různých údajů. Po poradě s vedoucím práce jsem zvolil následující zprávy. Některé jsou ve formátu NMEA, jiné zase v proprietárním binárním formátu UBX firmy U-Blox.

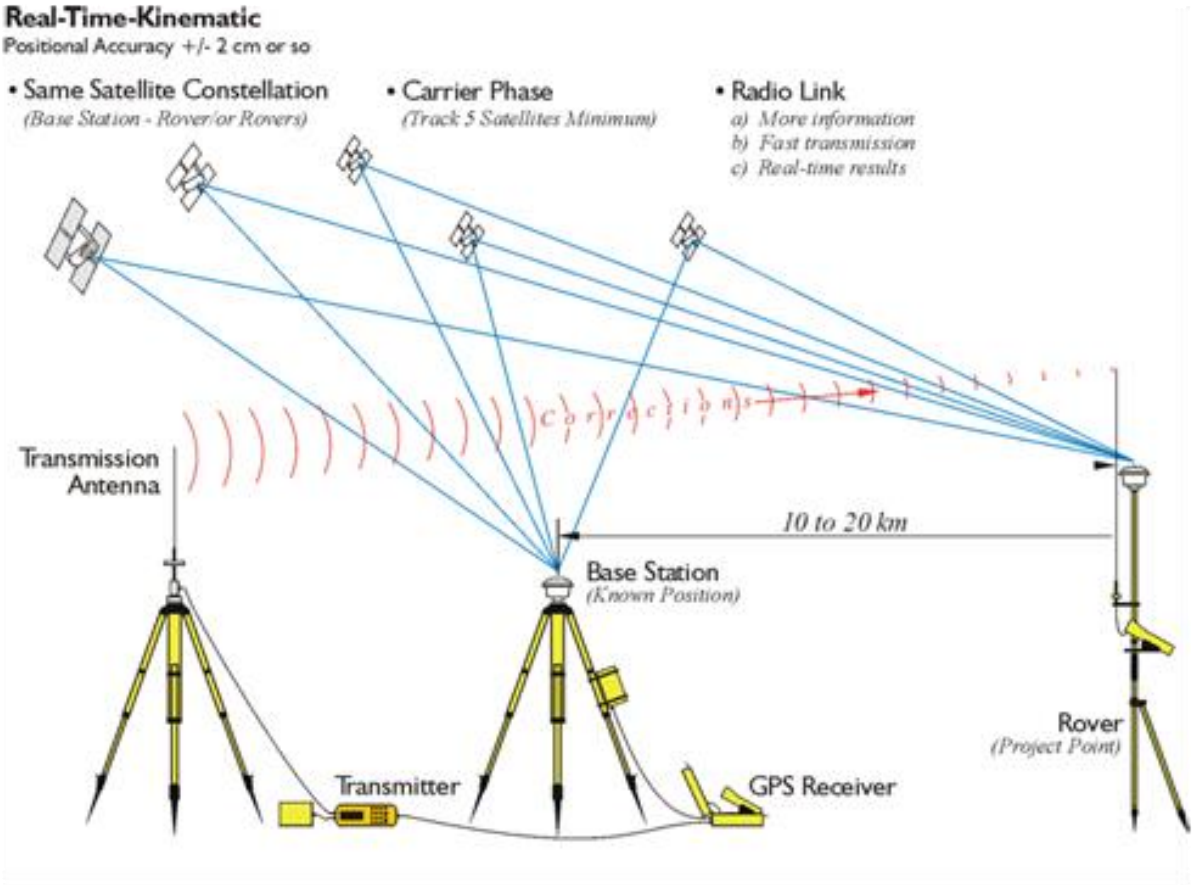
Tabulka 1. Zaznamenávané zprávy

Lat	Zeměpisná šířka.
Lon	Zeměpisná délka.
RELPOSNED	Jednotlivé složky relativního vektoru.
Height	Výška nad hladinou moře.
Heading	Směřování relativního vektoru.
Azimut	Směřování přijímače. Počítáno dle směru pohybu (nefunguje stacionárně).
Chyby	Směrodatné odchylky pro všechny sledované veličiny.
Age of correction	Doba od poslední korekční zprávy.

4.2.1 RTK fix/float [9]

Real Time Kinematics, neboli kinematika v reálném čase je metoda určení polohy na základě GPS, kde pro zpřesnění používáme korekční zprávy z pozemních základových stanic, které přijímáme nejčastěji přes internet protokolem NTRIP. RTK používá algoritmus pro zjištění přesného počtu vln mezi satelity a základovou stanicí a tím získává přesnou polohu. Omezením je hlavně vzdálenost přijímače, pro který chceme určit polohu (tzv. rover) a základové stanice. Ta může být řádově maximálně desítky kilometrů, kde se vzrůstající vzdáleností klesá přesnost určení polohy.

- RTK Fix –Pokud máme kvalitní signál z dostatku satelitů, tak získáme z RTK algoritmu celé číslo vlnové vzdálenosti. V tomto módu jsme schopni určit polohu s přesností na jednotky mm.
- Rtk Float – Pokud je signál méně kvalitní, či nemáme k dispozici dostatek satelitů, tak se můžeme spokojit s řešením, které používá vlnovou vzdálenost s desetinnými místy. Takováto poloha se může lišit až o 5 metrů od skutečné a je velmi obtížné zjistit skutečnou odchylku.



Obr. 7 Znázornění GNSS navigace s pomocí korekčních dat [9]

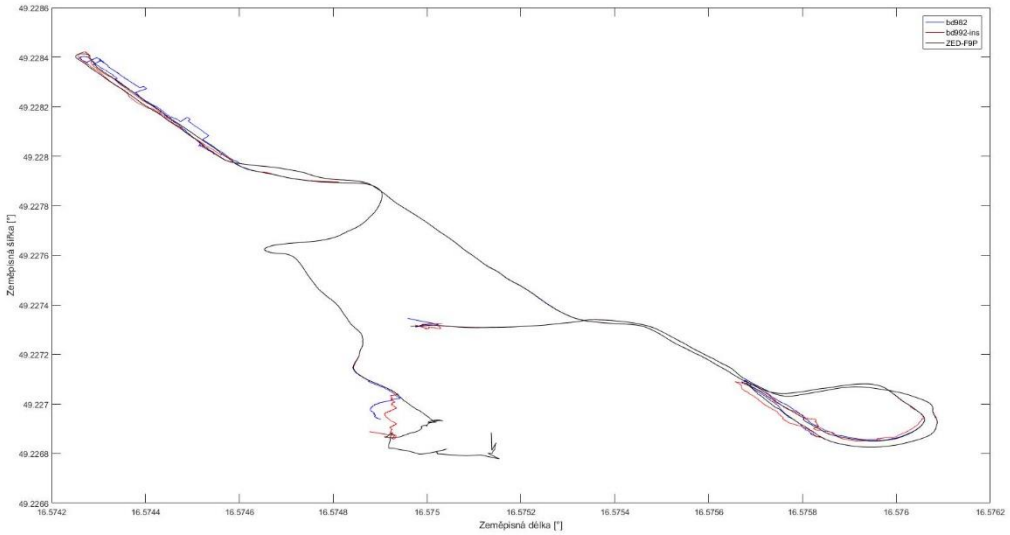
4.3 Měření pomocí dvou přijímačů

Měření pomocí dvou přijímačů, kde probíhal i výpočet úhlu mezi anténami nebylo úspěšné. Chyba polohy takto získané byla neúměrně velká a frekvence dodávaných polohových dat příliš nízká. Pokud bude potřeba měřit natočení pomocí GNSS, bude nutné výpočet realizovat externě a pro řídicí jednotku Pixhawk 2 Cube polohu a natočení emulovat.

4.4 Měření pomocí jednoho přijímače

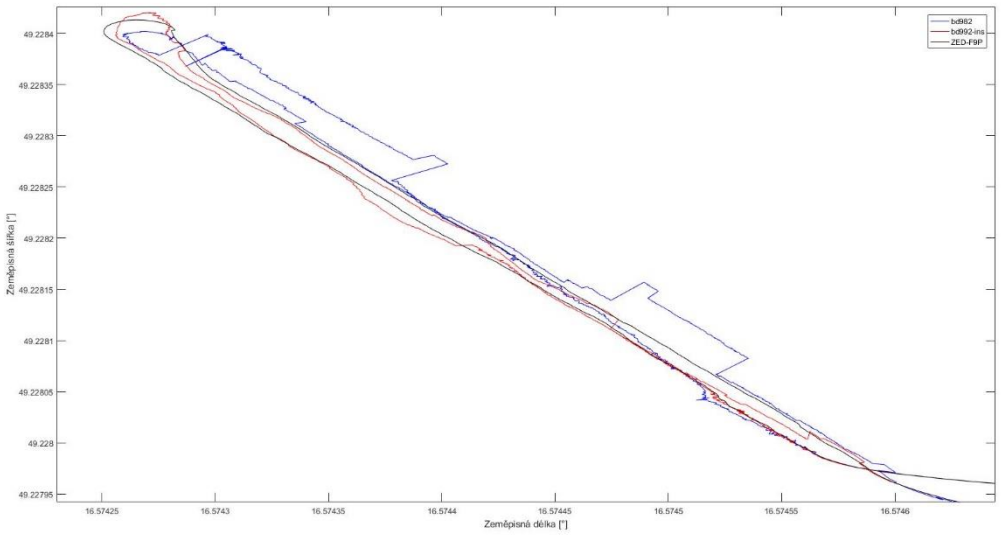
Měření bylo zacíleno na porovnání tří přijímačů v různých podmínkách a s různými anténami. Jedná se vždy o jeden přijímač bez měření úhlu mezi anténami. Po konvergenci řešení polohy vychází odchylky u všech antén ve volném prostoru zanedbatelné (do 2 cm). Zajímavější skutečnosti jsou zřetelné při průjezdu kolem zdi a vjezdu a výjezdu do garáží.

4.5 Trimble anténa



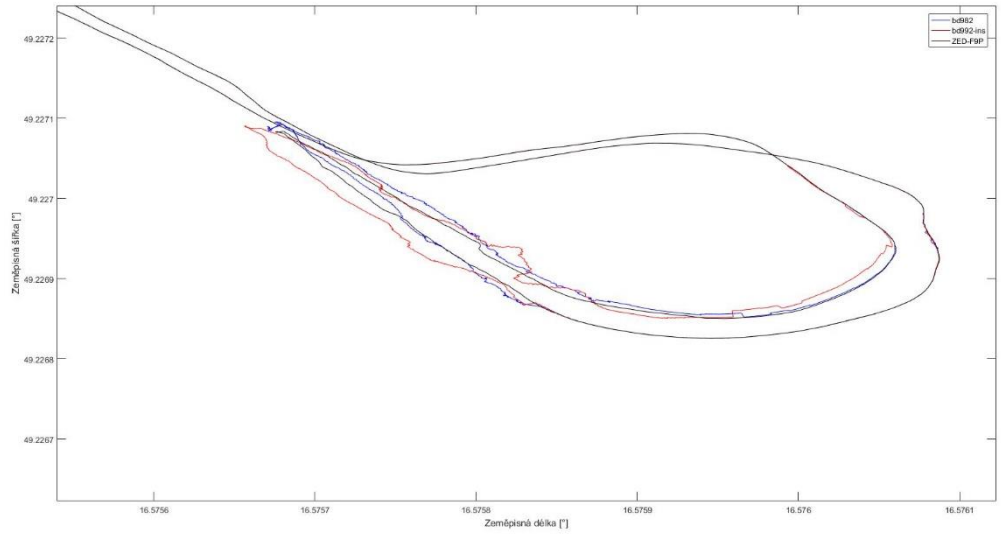
Obr. 8 Měření s anténou Trimble – celý pohled

Na začátku měření (úplně nesmyslná data byla odfiltrována) je zřetelně vidět, že u přijímačů Trimble dochází k rychlejšímu určení přesné polohy i když mají všechny přijímače stejný vstup z antény a stejná korekční data.



Obr. 9 Měření s anténou Trimble – část podél zdi

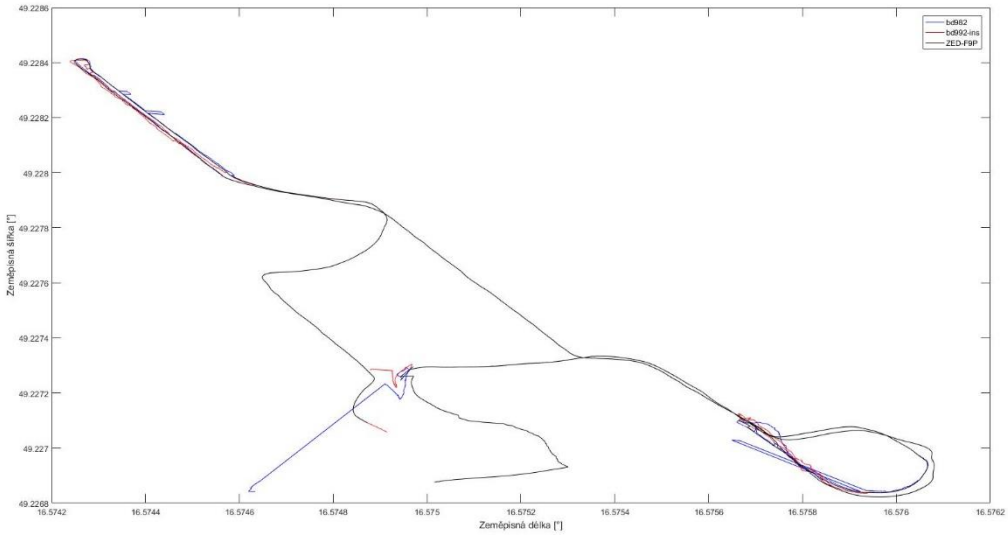
Na této části měření – rovné trajektorii kolem zdi laboratoře profesora Lista, vidíme, že ZED-F9P má nejpřímější trajektorii a oba přijímače Trimble kolem jeho polohy oscilují. U přijímače BD-982 se jedná o odchylky větší a hlavně skokové, což by mohlo při daných požadavcích na přesnost navigace dělat v blízkosti budov problémy.



Obr. 10 Měření s anténou Trimble – část vjezdu do garáží

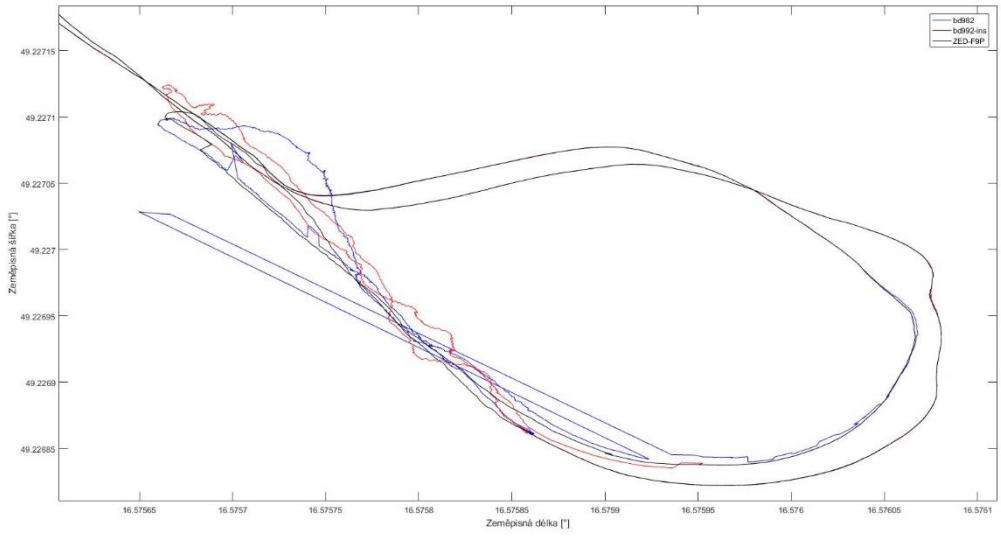
Při této části měření se jedná hlavně o schopnost přijímačů určit polohu z odrazů signálu (podzemní garáže) a také doba konvergence polohy při výjezdu z nich. Při zvážení podmínek měření dávají všechny přijímače rozumnou polohu a při výjezdu konvergují téměř ve stejný čas. Je zde vidět, že na rozdíl od antény stejné konstrukce – JINYUSHI, jsou přijímače Trimble na tuto anténu seřízeny.

4.6 JINYUSHI GNSS survey anténa



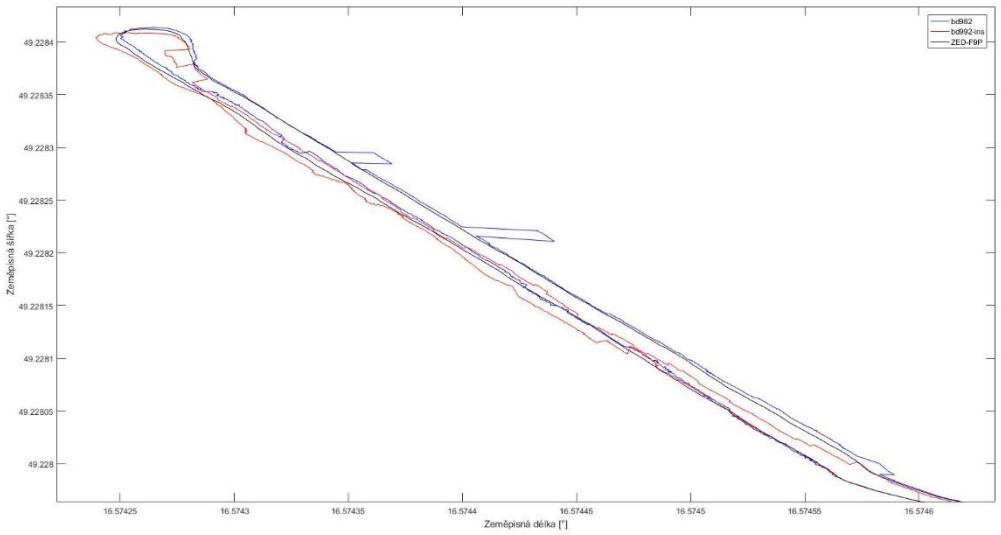
Obr. 11 Měření s anténou JINYUSHI – celý pohled

V tomto měření jasně vidíme, že na začátku měření konvergují všechny přijímače do přesné polohy ve zhruba stejnou dobu. Zajímavější data nám pak nabízejí části měření kolem zdi a v podzemních garážích.



Obr. 12 Měření s anténou JINYUSHI – část vjezdu do garáží

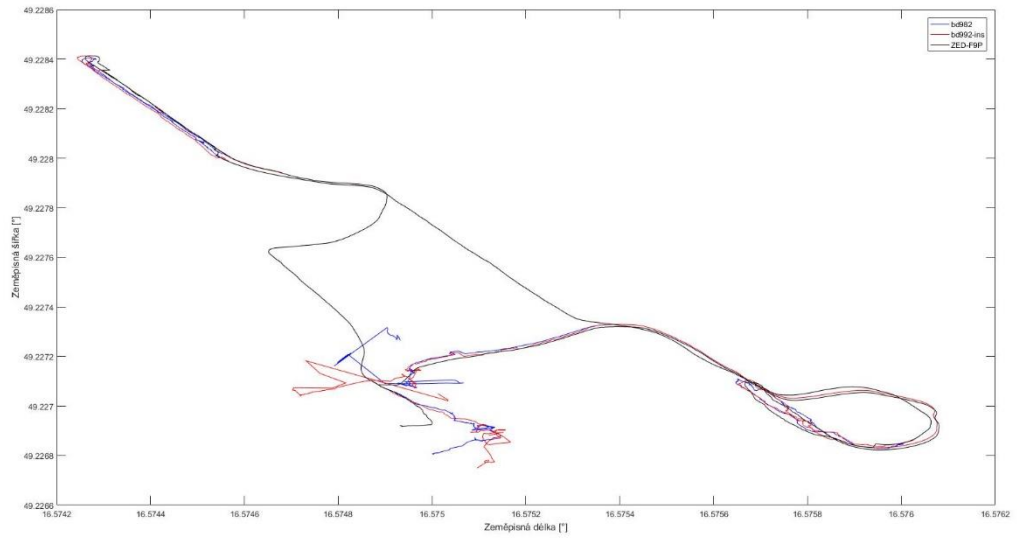
V této části měření s anténou JINYUSHI jdou vidět velké a často i skokové odchylky od skutečné polohy. Doba konvergence správné polohy při výjezdu je také opožděná oproti anténě Trimble. Zajímavé zde je, že nejméně ovlivněným přijímačem je ZED-F9P.



Obr. 13 Měření s anténou JINYUSHI – část podél zdi

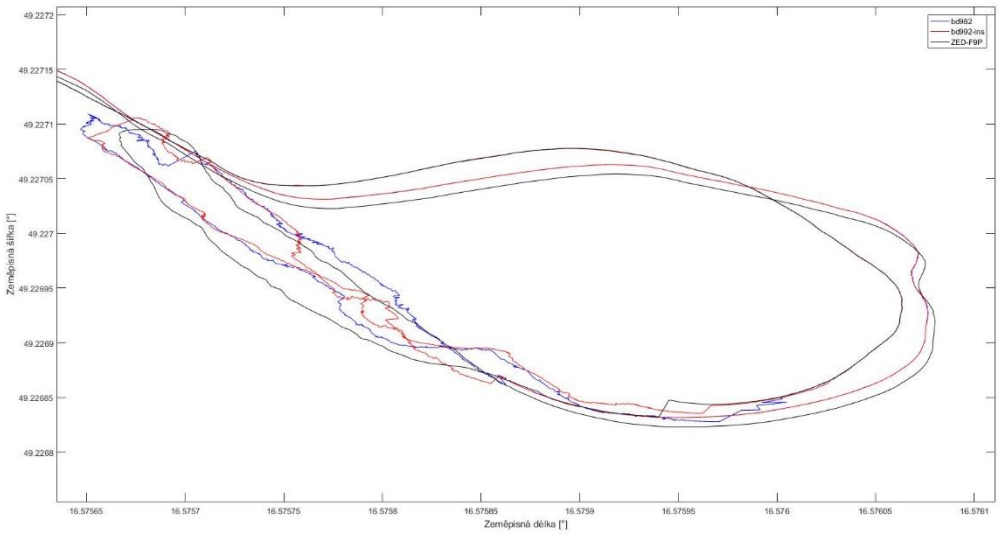
V části měření s anténou JINYUSHI u zdi vidíme, možná kupodivu, u přijímače BD-982 lepší výsledky než s originální anténou Trimble. U přijímače BD-992 jsou výsledky horší a u ZED-F9P porovnatelně dobré, jako u originální antény Trimble.

4.7 ArduSimple OEM GNSS anténa



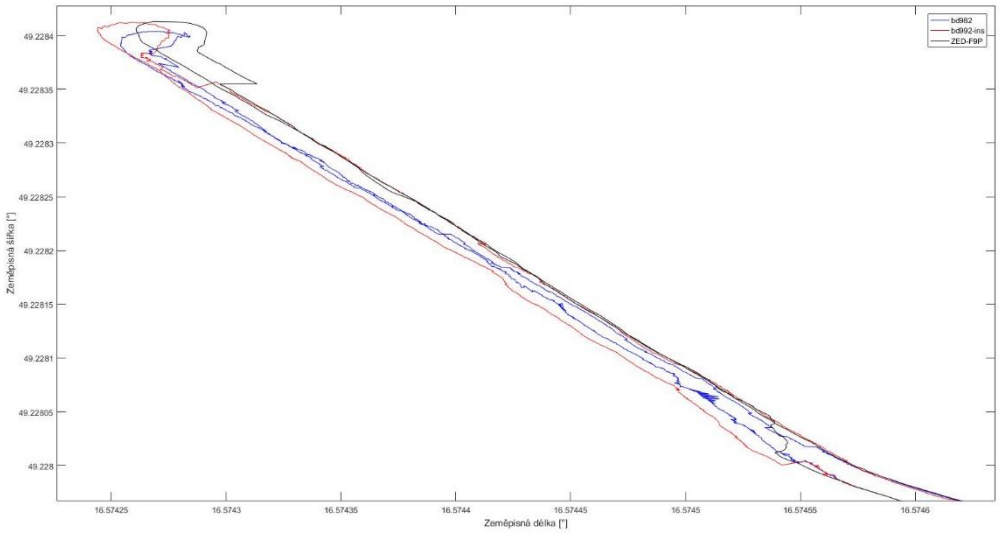
Obr. 14 Měření s anténou ArduSimple – celý pohled

U měření s ArduSimple OEM anténou vidíme na začátku i na konci měření obrovské chyby v určení pozice. Chyby v pozici se vyskytují i v některých částech trajektorie, kde všechny ostatní antény podávají perfektní výkon (otevřené prostranství).



Obr. 15 Měření s anténou ArduSimple – část vjezdu do garáží

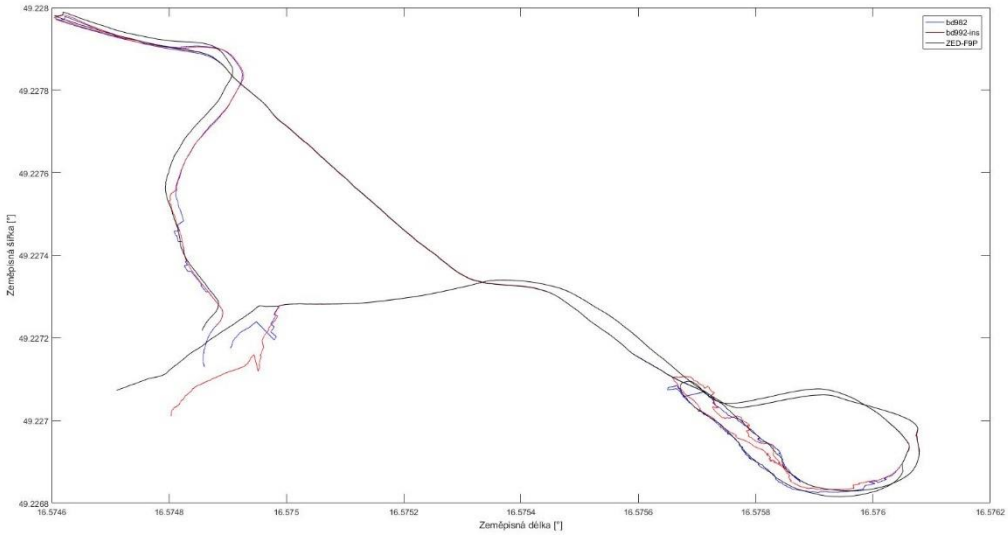
Ačkoli tato anténa nevyniká v otevřeném prostranství, tak v měření pomocí odrazů se dostáváme k rozumným výsledkům. Při vjezdu do garáží s touto anténou ale ztrácíme přesnou polohu jako první a opět jí získáváme mezi posledními.



Obr. 16 Měření s anténou ArduSimple – část u zdi

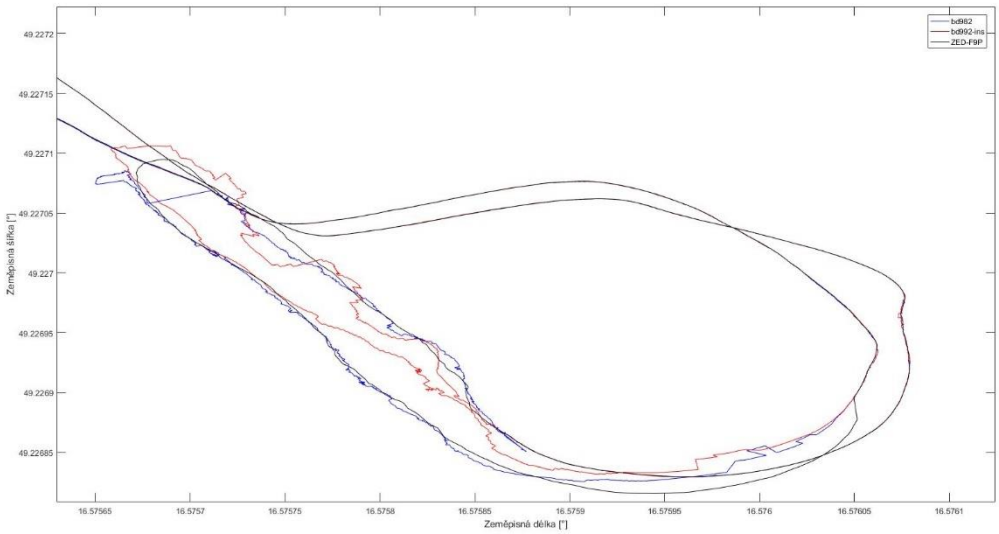
V části měření kolem zdi se s OEM anténou ArduSimple dostáváme k nejhorším výsledkům ze zkoumaných antén. Testujeme zde hlavně schopnost přijímačů potlačit odrazy, pokud mají i přímý výhled na navigační satelity. V této části měření nedodávaly dobrou polohu přijímače Trimble s žádnou anténou, ale přijímač ZED-F9P zde oproti použití s ostatními anténami pohořel také.

4.8 U-Blox



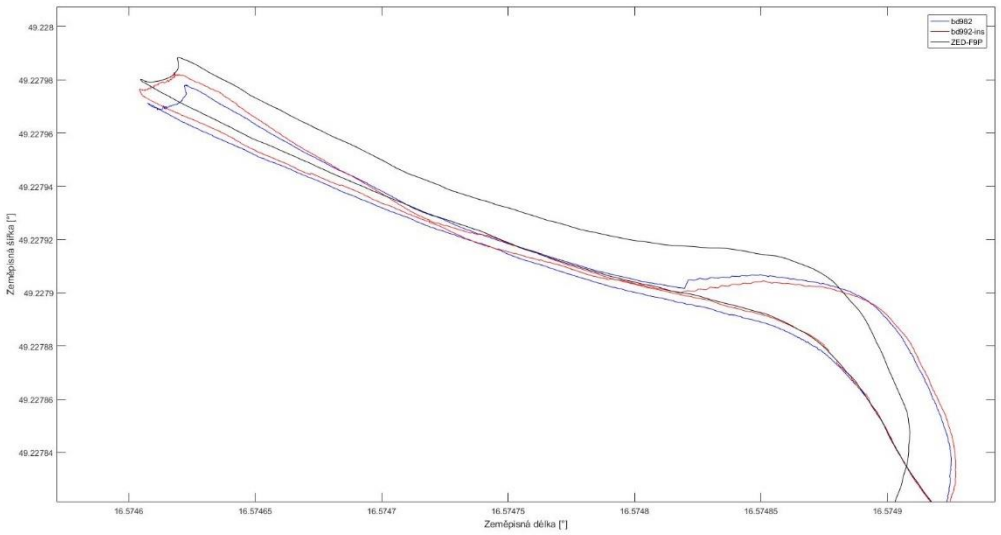
Obr. 17 Měření s anténou U-Blox – celý pohled

U antény U-Blox vidíme, že se trajektorie rozjíždějí na začátku i na konci měření. Posledních zhruba 10 metrů bylo poznamenáno výpadkem korekcí a nebude tedy k nim přihlíženo při hodnocení antény. Průběh měření je jinak velmi dobrý a trajektorie se v podstatě překrývají.



Obr. 18 Měření s anténou U-Blox – část vjezdu do garáží

Při vjezdu do garáží vidíme artefakty na trajektorii díky odrazům, ale v podzemních garážích nám všechny tři přijímače dávají velmi dobrou polohu. Konvergence správné polohy je zde také relativně rychlá.



Obr. 19 Měření s anténou U-Blox – část u zdi

Měření u zdi laboratoře profesora Lista dává s anténou U-Blox suverénně nejlepší polohová data na všech přijímačích. Odchylku na konci tohoto úseku přisuzuji přerušení příjmu korekčních dat.

4.9 Konečný výběr antény pro sekačku

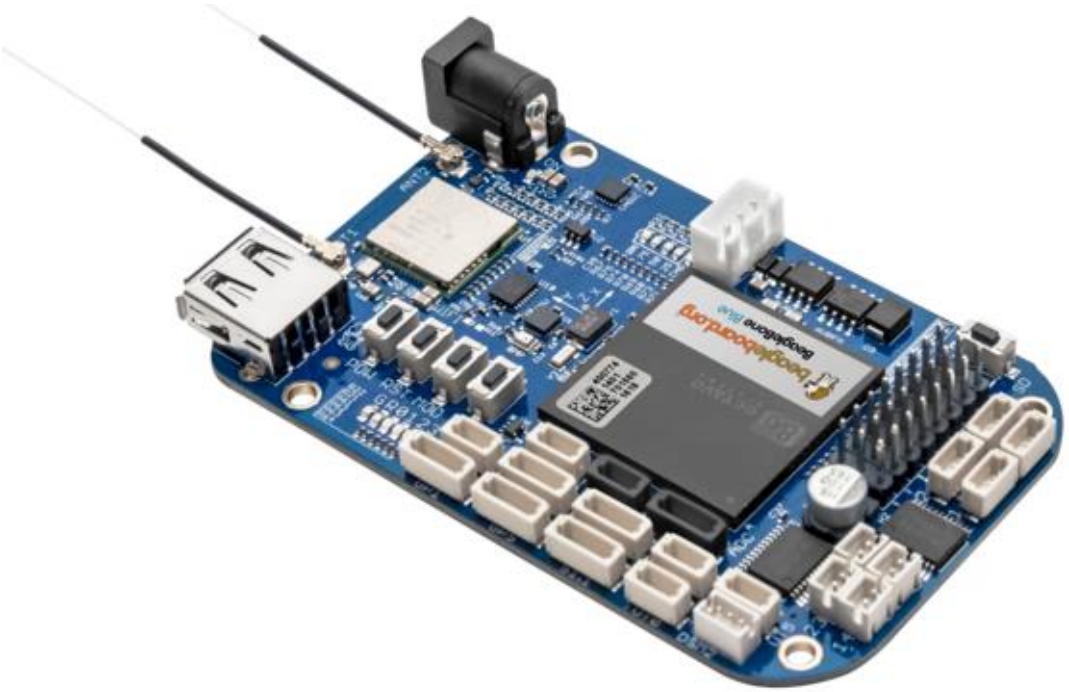
Jako optimální anténa pro danou aplikaci byla zvolena anténa U-Blox, jelikož při měření v otevřeném prostoru má srovnatelně dobré parametry jako ostatní antény, ale při měření kolem zdi, kde jsem zjišťoval schopnost potlačení odrazů, excelovala. Stejné podmínky použití se zamýšlejí i v cílové aplikaci. Přijímač U-Blox ZED-F9P se osvědčil jako vhodný pro tuto aplikaci.

5 OPEN-SOURCE ŘÍDICÍ PLATFORMY

Trendem v současné robotice je mít oddělené řídicí jednotky pro nízkoúrovňové a vysokoúrovňové řízení. Kromě jednodušší implementace je velkou výhodou také realtimové fungování, které je u lepších nízkoúrovňových jednotek dnes standardem. Bezpečnostní mechanismy se také realizují v těchto jednotkách, kde například pokud dojde k zatuhnutí nadřazeného systému, tak se robot bezpečně vrátí na místo startu. Na trhu dnes existuje větší množství takovýchto jednotek lišící se hlavně cenou, zamýšleným použitím, podporou komunity, kompatibilním softwarem, mírou otevřenosti a možnostmi pro připojení dalších periférií.

5.1 Beagle Bone Blue [10]

Beagle Bone Blue je kompletní mikropočítač navržený pro Linux. Je plně open-source a umožňuje na jedné desce provoz například Debianu a PX4/ArduRoveru najednou. Tím eliminujeme potřebu druhého počítače jako nadřízeného systému, ale na druhou stranu běží vše na jedné desce a případě pádu systému přijdeme o veškerou kontrolu nad robotem.



Obr. 20 Beagle Bone Blue [10]

- 1GHz ARM Cortex-A8
- 2×32-bit 200 MHz procesorové jednotky reálného času (PRUs)
- 512 MB DDR3 RAM
- 4 GB 8-bit eMMC flash paměť
- MPU9250 jako akcelerometr, gyroskop a kompas

- BMP280 barometr
- Wifi (802.11bgn)
- 8x Servo konektor, 4x DC motor konektor
- USB 2.0 klient i hostitel
- 11x programovatelné LED + 2 tlačítka
- SPI, I2C, UART

Oproti Pixhawku má podstatně menší podporu v komunitě, menší I/O možnosti a méně kvalitní sensory (a jejich menší počet). Výhodou je integrovaný Wi-Fi čip, díky kterému se můžeme s robotem spojit pomocí SSH, popřípadě tímto způsobem získávat korekce pro RTK GPS a nemusíme se dále zabývat řešením datového spojení a telemetrie.

5.2 Pixhawk 2 [11]

Pro nízkoúrovňové řízení byla vybrána řídicí jednotka Pixhawk 2 Cube, která obsahuje vše potřebné pro moji aplikaci a je zlatým standardem v podobných zařízeních. Verze Pixhawk 2 Cube obsahuje oproti verzi předchozí oddělení inerciální navigace do kostky nad výpočetní deskou a konektory pro ostatní zařízení. Toto uspořádání umožnilo použít pěnové tlumení pro potlačení vysokofrekvenčních vibrací a tím také snížení šumu na snímačích inerciální navigace. Prostor pro snímače inerciální navigace je také odstíněn tepelně, a navíc je teplota uvnitř regulována. Tím je zajištěna možnost provozování i za nízkých teplot a potlačení tepelného vlivu procesoru na výsledky měření sensorů.

Obr. 21 Pixhawk 2 Cube [11]

Výběr řídicí jednotky Pixhawk 2 Cube byl proveden zejména protože podporuje:

- RTK
- Duální GPS
- I2C
- SBUS
- Řízení napájení a nabíjení.
- Výstupy pro modelářské regulátory a serva.
- Sériovou komunikaci s nadřazeným/podřazeným systémem.
- Trojnásobně redundantní inerciální navigace.
    - 3x Akcelerometr
    - 3x Gyroskop
    - 3x Magnetometr
    - 3x Barometr

6 SOFTWARE ŘÍDICÍ JEDNOTKY

Do vybrané řídicí jednotky Pixhawk lze nahrát dva odlišné softwarové rámce, a to PX4 a ArduPilot. Jedná se o konkurenční projekty s podobnou funkcí.

6.1 PX4

PX4 je open source autonomní řídicí platforma, která se vyvinula z projektu Pixhawk na univerzitě v Curychu. Je přímým konkurentem ArduPilota a začíná být upřednostňovanou platformou v bezpilotním letectví, zejména pro svou širokou konfigurovatelnost a přizpůsobitelnost. Pro pozemní stroje se stále jeví lepším řešením ArduPilot, potažmo jeho část ArduRover, jelikož u PX4 není u pozemních vozidel tak aktivní komunita a překotný vývoj.

6.2 ArduPilot

ArduPilot, konkrétně jeho část pro pozemní vozidla ArduRover je open source platformou pro pozemní robotická vozidla. Stále probíhá jeho překotný vývoj a má aktivní komunitu, což ho dělá pro mé použití ideálním. ArduRover umožňuje jednak samostatný provoz, tak provoz s nadřazeným systémem pomocí ovládání protokolem MAVlink, což umožňuje integraci například pomocí ROS. [12]

6.2.1 Rozšířený Kalmanův filtr

Součástí ArduRoveru je takzvaný blok EKF (Extended Kalman Filter), který umožňuje fůzovat data z různých palubních sensorů jako například GPS, kompas, barometr, nebo akcelerometr. Na jeho výstupu jsou veličiny jako poloha vozidla, rychlost, nebo natočení spočítané pomocí všech vstupních sensorů. Výhodou tohoto řešení je možnost potlačit měření s velkou chybou, čímž se robot stává odolnějším proti chybě na jednom ze sensorů. Pokud je hardwarově dostupná více než jedna inerciální navigace (IMU), tak lze spustit více EKF jader/bloků. Každý z nich potom používá jinou sadu IMU a systém pak používá ten EKF, který hlásí nejlepší shodu dat ze všech snímačů. Tato funkcionalita je navržena jako tzv. „black box“ pro uživatele s tím, že několik málo parametrů lze nastavit. Nejčastěji se nastavuje požadovaný počet instancí EKF, dle hardwarové konfigurace řídicí jednotky. V našem případě jsou zajímavé ještě další tři parametry. Nejdůležitějším je asi váha mezi kompasem a GPS při určování natočení, kde výchozí váha je 0,5 (0-1). Následuje typ dat z GPS, které EKF používá. Lze nastavit, zda bude přihlížet k 2D/3D poloze, popř 3D/2D/žádné rychlosti. Posledním je poměr mezi váhou dat z akcelerometru a barometru. Pro toto použití je výchozí hodnota 1 optimální.

6.2.2 Mission Planner

Mission Planner je aplikace dostupná pro Windows a Linux. S pomocí této aplikace lze konfigurovat Pixhawk s nahraným softwarem ArduRover. [13] Jedná se o jakési řídicí centrum, kde můžeme naprogramovat vše od průběhu autonomní mise až po zpracování GPS signálu, či komunikace s přidruženými systémy. Mission Planner obsahuje také simulátor, kde si průběh misí můžeme vyzkoušet nanečisto a vyhnout se případné škodě na hardwaru, nebo okolí. Nejintuitivnější způsob plánování autonomních misí je jejich průběh realizovat v režimu manuálního ovládání a zaznamenané waypointy použít pro opakování mise v autonomním režimu. Do předpisu autonomní mise můžeme nastavit také reakce na různé vnější podněty, jako například detekce překážky čidly, nebo na vnitřní změny, nejčastěji nízké napětí v bateriích.

Obr. 22 Náhled na program Mission Planner [13]

7 MECHANICKÁ PLATFORMA

Výběr mechanické platformy pro realizaci byl těžkým oříškem. Po úvodním rozplánování projektu a jeho nároků bylo zřejmé, že může být zvolena jedna ze tří cest. První cestou bylo použít originální platformu projektu ArduMower, druhou cestou bylo použít návrhu ArduMower ze 3D tištěných dílů a hliníkových profilů. Poslední cestou bylo upravit komerčně dostupnou platformu pro potřeby tohoto projektu. V úvahu se musely vzít zejména parametry jako tzv. „mrtvý čas“ – čas strávený výrobou fyzické konstrukce, který tak nemohl být použit na samotný vývoj elektroniky a softwaru, nebo cena, do které se musely započítat i všechny vícenáklady daného řešení (poštovné dílů z různých částí světa, riziko objednání špatného dílu, zničení součástky při úpravě pro projekt, náročnost na koordinaci dodání všech částí atd.). Jako poslední parametr jsem hodnotil i estetiku výsledného řešení.

7.1 ArduMower Original

Tuto variantu jsem se ze začátku nejvíce zvažoval protože:

- Jsou dostupné plány pro výrobu šasi na CNC fréze.
- Cena platformy je nejnižší ze zvažovaných.
- Veškeré díly jsou snadno dostupné.

Na druhou stranu bylo potřeba zvážit i negativa. Hlavní negativa byla hlavně:

- Nutnost outsourcovat část výroby.
- Problematické zajištění vodotěsnosti.
- Velký podíl ruční práce a tím času na konstrukci.
- Poměrně menší záběr sečení.
- Problematická trakce na podmáčeném trávníku.
- Horší možnost integrace nárazových čidel.
- Neprofesionální vzhled.

Obr. 23 Originální šasi ArduMower [14]

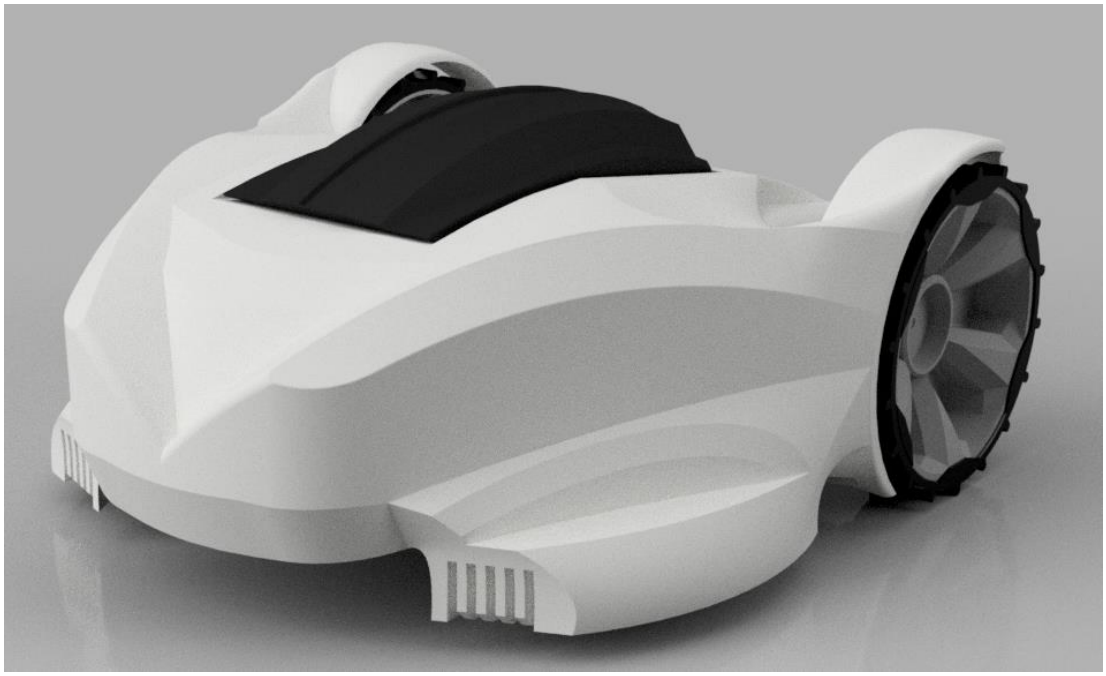
7.1.1 ArduMower Arctic Hare

Jedná se o nový komunitní návrh projektu ArduMower, který má za cíl usnadnit stavbu a dodat výsledné sekačce profesionální vzhled. Tento návrh byl zvažován pro:

- Relativně rychlá stavba s použitím 3D tištěných dílů.
- Možnost vytištění poškozených dílů.
- Dostupnost konstrukčních dílů.
- Rozumná cena.
- Úžasný design.

Jako negativa jsem u toho návrhu zvažoval:

- Možné problémy s tiskem dílů.
- Postupná degradace materiálu vlivem UV záření.
- Menší záběr sečení.
- Možné poškození tištěných dílů teplem z dlouhodobě namáhaných motorů.
- Větší časová náročnost než u úpravy komerční platformy.
- Horší možnost integrace nárazových čidel.



Obr. 24 ArduMower Arctic Hare [15]

Tato možnost byla nakonec zvolena jako záložní, pokud by se nepodařilo sehnat vhodnou komerční platformu.

7.2 Robomow RL 2000 [16]

Jako základní platforma byla nakonec vybrána sekačka Robomow RL 2000, jelikož má bytelnou konstrukci a jeden z největších záběrů (53 cm) ze sekaček běžně dostupných na trhu. Některé sekačky mají záběr třeba i jen 16 cm. V takovém případě je potřeba velice přesná navigace v každém bodu trajektorie, aby nevznikaly nepokosená místa. V našem případě můžeme tedy trajektorii vždy částečně překrývat a tím snížit nároky na přesnost navigace při zachování rozumné délky celkové trajektorie. Velkou výhodou je nutnost pouze menších úprav na konstrukci od zavedeného výrobce a snadné zajištění vodotěsnosti a nárazových čidel. Oproti dvěma předchozím platformám je Robomow RL 2000 podstatně těžší, což mu zajišťuje trakci i na podmáčeném podkladu. Další výhodou je snadná výměna tří žacích nožů a dobrá dostupnost náhradních dílů. Nevýhodami zde byla pouze vyšší cena a nutnost úpravy již navrženého řešení. Poslední výhodou tohoto řešení je nabíjecí stanice, jelikož v budoucnu plánuji integrovat automatické dokování.

Obr. 25 Robomow RL200 a nabíjecí stanice [16]

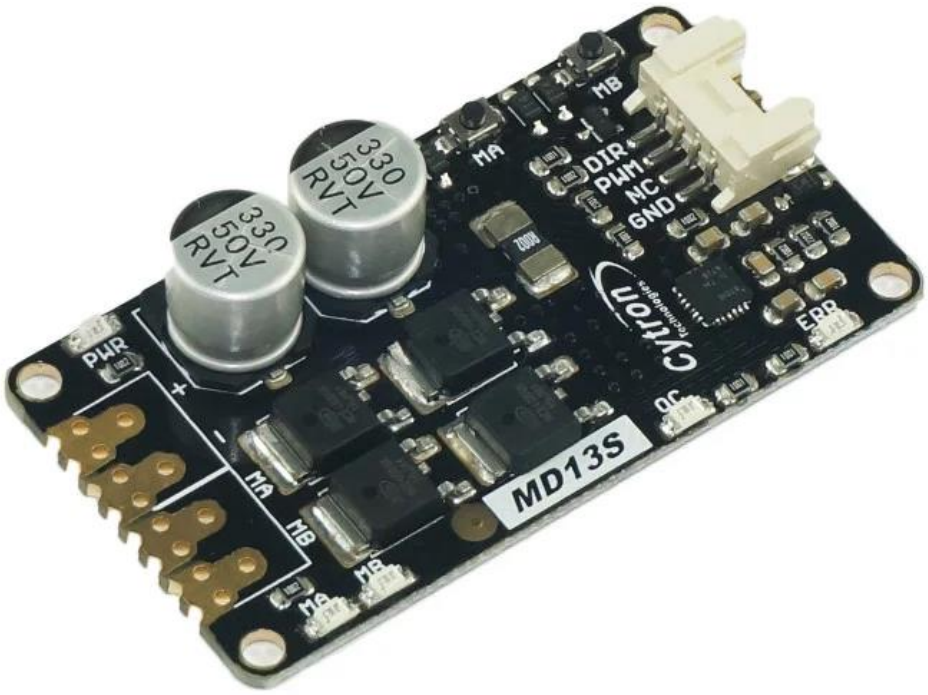
7.3 Podvozek a jeho řízení

Podvozek vybrané robotické sekačky RL 2000 je tříkolový. Dvě zadní kola můžeme ovládat nezávisle na sobě dvěma DC motory s regulátory. Přední kolo je vlečné.

7.3.1 Regulátor CYTRON MD13S

Po zvážení parametrů motorů pohonu – DC motory 24 V/100 W a nízko úrovňové řídicí jednotky Pixhawk Cube byl zvolen regulátor CYTRON MD13S, který svými parametry vysoce převyšuje potřeby dané aplikace a bylo ho možné pořídit z české distribuce, což byl v čase korona krize zásadní faktor. Regulátor má následující parametry:

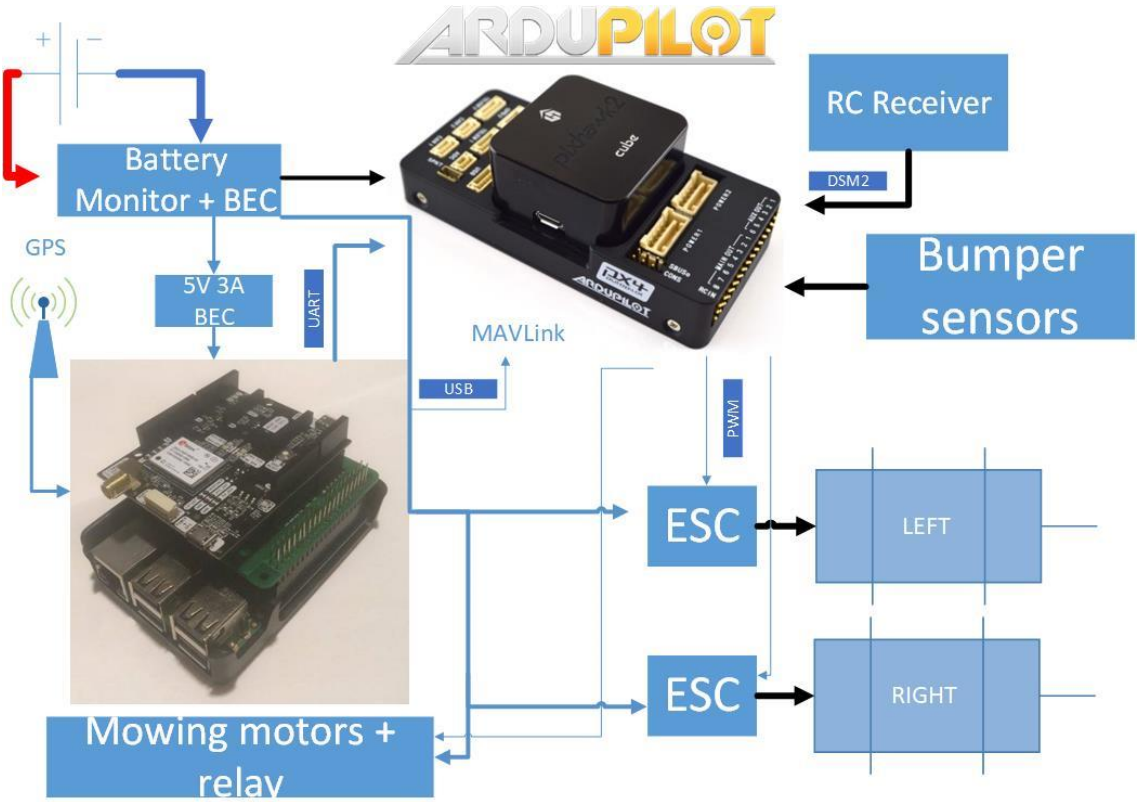
- Podporované napětí: 6-30 V.
- Maximální proud motorové větve: 13 A trvale, 30 A špičkově.
- Obousměrný provoz motorů.
- Vstup: Signál 0-20 KHz pro rychlost + binární vstup pro směr otáčení.
- Výstup: PWM signál přivedený na póly motoru.
- Vysoká efektivita, díky které nepotřebuje chladič.
- Možnost manuálního ovládání.



Obr. 26 Regulátor Cytron MD13S [17]

V Pixhawku je regulátor nakonfigurován jako „Brushed with relay DC regulator“. Typ řízení je zvolen „Skid steering“ tzv. tankové řízení, kde řídíme směr a rychlost otáčení obou kol a tím i výsledný vektor pohybu. Řízení je aktuálně nastaveno tak, že při zatáčení u dopředého pohybu se jeden z motorů pouze zpomalí, či zastaví, ale nikdy nepojede do opačného směru. Umožňovalo by to sice agresivnější zatáčení na menším poloměru, ale experimentálně bylo zjištěno, že rázy vyvíjené motorem při změně směru chodu byly natolik velké, že ohrožovaly strukturální integritu převodovky a konstrukce samotné. Naopak pokud sekačka zjistí, že se odchyluje od úsečky mezi aktuálním a minulým waypointem od více než 45°, tak se zastaví a provede korekci natočení na místě. Zde již probíhá chod motorů na opačné strany a poškození nehrozí.

8 FYZICKÉ PROVEDENÍ ELEKTRONIKY



Obr. 27 Blokové schéma zapojení elektroniky

Na obrázku výše je rozkreslené blokové schéma zapojení elektroniky včetně použitých sběrnic. Baterie je připojena přes bateriový modul dodávaný s Pixhawkem a můžeme díky tomu měřit její napětí a aktuální odebíraný proud. V bateriovém modulu je i integrovaný zdroj pro napájení Pixhawku a příslušenství. Z důvodu vysokého odběru Raspberry Pi a GNSS modulu bylo nutné přidat step-down měnič pro jejich napájení. Sensory nárazu jsou připojeny přímo na vstup pro „safety tlačítko“ Pixhawku. Komunikace mezi modulem Raspberry Pi + ZED F9P probíhá pomocí dvou kanálů. Jednosměrně pomocí UART portu se posílají GNSS data ze ZED-F9P do GPS portu Pixhawku a oboustranné propojení je realizováno pomocí USB mezi Raspberry Pi a Pixhawkem.

8.1 Připojení k síti

Pro potřeby vývoje je připojení realizováno přes Wi-Fi. Po jeho dokončení se pro normální provoz použije USB LTE modem a VPN. Tímto způsobem pak bude možné sekačku monitorovat a ovládat odkudkoli.

9 PALUBNÍ POČÍTAČ

Hlavním „mozkem“ celé sekačky je Raspberry Pi 3B+ s Raspbianem Stretch. RPi zajišťuje příjem korekcí, jejich vyslání ve správném formátu přes UART do GPS F9-P a řízení průběhu mise v Pixhawku. Po zvážení všech možností realizace přicházejících v úvahu jsem zúžil možný výběr na dvě možnosti. Obě řešení stavěly na Pixhawku s ArduRoverem a RPi s linuxovým operačním systémem. První možností bylo řídit kompletní průběh mise z linuxového mikropočítače pomocí vlastního ROS node, MAVROSu a vizualizací pomocí RVIZ. Pixhawk by v tomto schématu byl použit pouze jako fyzické rozhraní a jako zdroj dat z jeho sensorů. Motory podvozku by se ovládaly přímo nastavením úhlové rychlosti jejich otáčení z nadřazeného systému. Pixhawk by převzal kontrolu nad sekačkou pouze pokud by nastala nestandartní situace jako například zatuhnutí nadřazeného systému, nebo náraz sekačky do objektu. Toto řešení bylo přes svoji větší obtížnost realizace na začátku zvoleno, jelikož umožňovalo nejširší další rozvoj softwaru sekačky pomocí ROS balíčků. V průběhu realizace jsem se ale dostal do slepé uličky a rozhodl jsem se tedy využít druhého na začátku zvažovaného řešení. Druhý způsob řešení spočíval v nahrání celého průběhu mise do Pixhawku a defacto využití jeho plného potenciálu pro navigaci a řízení sekačky. Na nadřazeném RPi by tedy zůstalo pouze spouštět a kontrolovat průběh mise, zajistit správné korekce a zajištění telemetrie s logováním. V této variantě bylo třeba v Pixhawku navíc vyladit navigační a řídicí část a pro RPi napsat řídicí skript v Pythonu s pomocí DroneKitu.

9.1 Operační systém

Při vývoji byl upřednostňovaný Raspbian Buster (aktuálně nejnovější verze) a ROS Kinetic z důvodu, že na novějších verzích ROSu (Melodic+) je hlášeno množství problémů s provozováním MAVROSu, který měl zaobstarávat komunikaci s Pixhawkem. Úskalí je v tom, že ROS Kinetic již nemá dostupné zkompilované balíčky a je tedy nutné vše kompilovat ze zdrojových kódů.

9.1.1 Raspbian Buster [18]

Začátek instalace byl tzv. „na zelené louce“ – naformátovaná SD karta. Poté proběhla instalace Raspbian Buster z oficiální distribuce a provedena aktualizace.

```
sudo apt-get update
sudo apt-get upgrade
```

Poté byly nainstalovány balíčky potřebné pro instalaci ROS. U některých balíčků nebyly dostupné závislosti a musely se postupně doinstalovat ručně.

```
sudo apt-get install python-rosdep python-rosinstall-generator python-wstool python-rosinstall build-essential
```

Následně je třeba nainstalovat PIP a rosdep, který inicializujeme a aktualizujeme.

```
sudo pip install -U rosdep rosinstall_generator wstool rosinstall

sudo rosdep init

rosdep update
```

Zbývá nainstalovat samotný ROS

```
rosinstall_generator robot --rosdistro kinetic --deps --wet-only --tar > kinetic-robot-wet.rosinstall

wstool init -j8 src kinetic-robot-wet.rosinstall
```

Instalace poté bohužel zhavaruje při instalaci Catkinu. Toto jsem se snažil vyřešit mnoha způsoby, ale po důkladném prozkoumání internetových fór a dalších zdrojů k ROSu jsem zjistil, že mnoho uživatelů má stejný, či podobný problém a nikde nebyl dostupný ani nástin možného řešení. Prozkoumal jsem možné další cesty a nejschůdněji se jevilo použití předpřipraveného obrazu s ROSem na Ubuntu Xenial od Ubiquiti Robotics.

9.1.2 Ubuntu Xenial [19]

Zde proběhlo stažení předpřipraveného obrazu a nahrání na SD kartu. Po vložení do RPi proběhlo bootování bez problémů a bylo již možné systém používat. Následně jsem provedl aktualizaci systému:

```
sudo apt-get update

sudo apt-get upgrade
```

Zkontroloval jsem funkčnost ROSu a začal jsem kompilovat NTRIP klienta. Kompilace bohužel zhavaruje na nízké verzi LIBC6. Jedná se o knihovnu propojenou s kernelem, rozhodl jsem se tedy pro upgrade na vyšší verzi systému. Postupoval jsem dle doporučeného postupu, ale upgrade bohužel není možný bez odinstalace ROSu, který byl hlavním důvodem pro volbu toho systému. Dalším krokem bylo prozkoumání jiných možností operačních systémů a rozhodl jsem se

pro Raspbian Jessie, jelikož v době vydání ROS Kinetic to byl podporovaný a odzkoušený systém.

9.1.3 Raspbian Jessie [18]

Začátek instalace byl opět zvolen „na zelené louce“ – naformátovaná SD karta. Poté proběhla instalace poslední vydané verze Raspbian Jessie z oficiální distribuce a provedena aktualizace.

```
sudo apt-get update

sudo apt-get upgrade
```

Poté byly nainstalovány balíčky potřebné pro instalaci ROS. U některých balíčků nebyly dostupné závislosti a musely se postupně doinstalovat ručně.

```
sudo apt-get install python-rosdep python-rosinstall-generator python-wstool python-rosinstall build-essential
```

Následně je třeba nainstalovat PIP a rosdep, který inicializujeme a aktualizujeme.

```
sudo pip install -U rosdep rosinstall_generator wstool rosinstall

sudo rosdep init

rosdep update
```

Zbývá nainstalovat samotný ROS

```
rosinstall_generator robot --rosdistro kinetic --deps --wet-only --tar > kinetic-robot-wet.rosinstall

wstool init -j8 src kinetic-robot-wet.rosinstall
```

Nainstalujeme potřebné závislosti:

```
rosdep install --from-paths src --ignore-src --rosdistro kinetic -y
```

Vše zatím probíhá v pořádku a dostáváme se do bodu, kde zhavarovala instalace na Raspbian Buster. Provedeme instalaci Catkinu:

```
./src/catkin/bin/catkin_make_isolated --install -DCMAKE_BUILD_TYPE=Release
```

Catkin při instalaci několikrát zhavaruje. Je nutné se vždy podívat na danou chybu a postupně je jednu po druhé řešit. Postupně se tak iteračně dostáváme do funkčního stavu.

Instalaci dokončíme spuštěním vygenerovaného soboru setup.bash

```
source ~/ros_catkin_ws/install_isolated/setup.bash
```

V tuto chvíli máme funkční ROS a můžeme se vrhnout na další krok – instalaci NTRIP klienta. Vybraný NTRIP klient bohužel opět zhavaruje při kompilaci na nízké verzi LIBC6. I přes snahu doinstalovat vyšší verzi LIBC6 se ani tuto konfiguraci nepodařilo zprovoznit, jelikož aktualizací zase rozbijeme doteď funkční instalaci ROSu. V tuto chvíli jsem se rozhodl provést upgrade systému na vyšší verzi (Stretch), která již obsahuje potřebnou knihovnu LIBC6 a zkusit instalaci překompilovat na tuto verzi.

9.1.4 Raspbian Stretch [18] [20]

Raspbian Stretch nebyl instalován ze zdroje, ale pomocí upgrade systému z verze Jessie.

```
sudo apt-get update
sudo apt-get full-upgrade
```

Následně jsem upravil repozitáře v `/etc/apt/sources.list` a `/etc/apt/sources.list.d/raspi.list` na verzi Stretch.

```
sudo apt-get -y dist-upgrade
```

Jelikož již máme nainstalovány závislosti zbývá překompilovat Catkin.

```
./src/catkin/bin/catkin_make_isolated --install -DCMAKE_BUILD_TYPE=Rel
ease
```

Catkin při instalaci opět několikrát zhavaruje. Je tedy nutné se vždy podívat na danou chybu a postupně je jednu po druhé řešit. Postupně se tak iteračně dostáváme do funkčního stavu.

Instalaci dokončíme spuštěním vygenerovaného soboru setup.bash

```
source ~/ros_catkin_ws/install_isolated/setup.bash
```

Nyní máme funkční ROS a již dostačující verzi LIBC6. Zbývá zkompilovat NTRIP klienta. Toto již proběhlo bez problémů. V tuto chvíli jsem provedl zálohu celé SD karty RPi do formy obrazu disku, aby bylo možné se vrátit do tohoto bodu v čase kdykoliv bude potřeba, popřípadě použít tento obraz v jiné aplikaci, jelikož již jsou hlavní části (ROS, NTRIP klient) funkční a připraveny k použití.

9.2 NTRIP klient

Jako NTRIP klient vhodný pro použití v RPi a zároveň schopný posílat korekční data do GPS po sériové lince ve správném formátu byl zvolen kód od Dirka Stoeckera dostupný na GitHubu. [20] Jedná se o jediný open-source program dostupný na internetu vhodný pro potřeby této aplikace. NTRIP klient potřebuje pro správnou kompilaci LIBC6 minimálně 2.24, což muselo být zjištěno experimentálně. Zdrojový kód naklonujeme z GitHubu do adresáře v domovském adresáři RPi a zkompilujeme:

```
make
make install
```

NTRIP klienta spustíme následujícím příkazem:

```
./ntripclient -m TUBO3 -u UZIVATELSKE_JMENO -p HESLO -s IP_ADRESA_SERV
ERU -r 2111 -D /dev/ttyS0 -B 115200
```

9.2.1 Vytvoření NTRIP daemonu

Vytvoříme soubor ntrip.service, který obsahuje instrukce pro chování daemonu.

```
[Unit]
Description=Startup Script Service
After=multi-user.target
[Service]
Type=simple
Restart=always
RestartSec=10
User=root
ExecStart=/home/pi/ntrip/ntrip.sh
[Install]
WantedBy=multi-user.target
```

Soubor zkopírujeme do */lib/systemd/system* a přidáme práva pro spuštění skriptu, který nám zajistí spuštění programu se správnými parametry. Nakonec službu aktivujeme.

```
cd /home/pi/ntrip/
sudo chmod +x ntrip.sh
cd /home/pi/
sudo mv ntrip.service /lib/systemd/system
```

```
sudo systemctl daemon-reload
sudo systemctl enable ntrip.service
```

9.3 MAVProxy [21]

MAVProxy je jeden ze základních stavebních kamenů u komunikace Pixhawku s nadřazeným systémem. MAVProxy je schopen zastat následující funkce:

- GCS – Pozemní řídicí stanoviště
- Zprostředkování konektivity k Pixhawku přes UDP port
- Zprostředkování konektivity k Pixhawku přes TCP port
- Multiplexace připojení

V mojí instalaci používám připojení pomocí UDP portů, kde jeden UDP port slouží k případnému připojení GCS Mission Planner na Windows PC a druhý lokální UDP port na který se připojí řídicí software (ROS/Dronekit).
Instalace MAVProxy se provádí následovně:

Nejdříve nainstalujeme všechny balíčky, na kterých MAVProxy závisí.

```
sudo apt-get install python3-dev python3-opencv python3-wxgtk3.0 pytho
n3-pip python3-matplotlib python3-pygame python3-lxml python3-yaml
```

Je pravděpodobné, že u některých balíčků nepůjde automaticky vyřešit jejich závislosti a budou se muset doinstalovat ručně.
Po úspěšné instalaci můžeme pomocí PIP nainstalovat samotný MAVProxy.

```
sudo pip install MAVProxy
```

Nakonec je třeba MAVProxy v systému inicializovat.

```
echo "export PATH=$PATH:$HOME/.local/bin" >> ~/.bashrc
```

Tímto posledním krokem získáme funkční instalaci MAVProxy a můžeme program spustit. V mém případě je třeba do příkazu pro spuštění zakomponovat následující:

- Připojení Pixhawku přes USB - /dev/ttyACM0
- Rychlost 115200 baudů
- 1.UDP port lokální (127.0.0.1) 14551
- 2.UDP port pro ovládání přes síť – 14552

Spuštění pak zajistíme tímto příkazem:

```
mavproxy.py --master=/dev/ttyACM0 --baudrate=115200 --out udp:127.0.0.
1:14551 --out udp:192.168.1.104:14552
```

Z toho příkazu vytvoříme shell skript *mavproxy.sh* na spouštění MAVProxy se správnými parametry.

```
#!/bin/bash

mavproxy.py --master=/dev/ttyACM0 --baudrate=115200 --out udp:127.0.0.
1:14551 --out udp:192.168.1.104:14552
```

9.3.1 Vytvoření daemonu

Vytvoříme soubor mavproxy.service, který obsahuje instrukce pro chování daemonu.

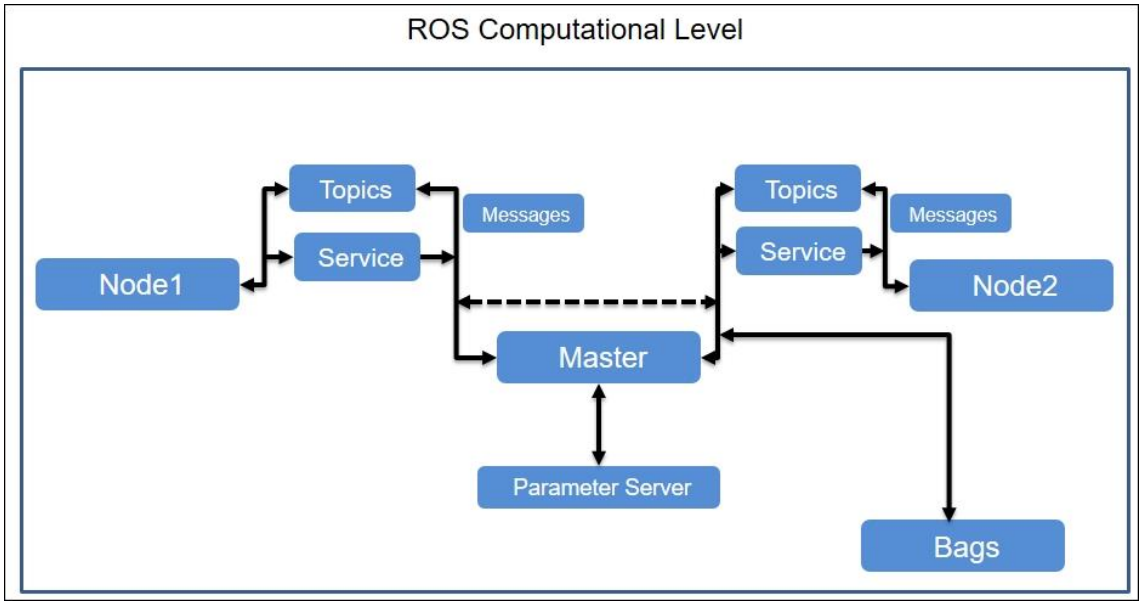
```
[Unit]

Description=Startup Script Service

After=multi-user.target

[Service]

Type=simple

Restart=always

RestartSec=10

User=root

ExecStart=/home/pi/mavproxy.sh

[Install]

WantedBy=multi-user.target
```

Soubor zkopírujeme do */lib/systemd/system* a přidáme práva pro spuštění skriptu, který nám zajistí spuštění programu se správnými parametry. Nakonec službu aktivujeme.

```
cd /home/pi/

sudo chmod +x mavproxy.sh

sudo mv mavproxy.service /lib/systemd/system

sudo systemctl daemon-reload

sudo systemctl enable mavproxy.service
```

9.4 ROS [18]

ROS – neboli Robot Operating System je framework pro vývoj softwaru pro roboty. Je to sbírka nástrojů, knihoven a návrhových vzorů, která má za cíl usnadnit a standardizovat psaní komplexního software k ovládání robotů. Nejedná se o operační systém v pravém slova smyslu, ale o soubor aplikací spouštěných typicky pod Ubuntu. Základním kamenem ROSu je roscore. Jednotlivé programy se poté jmenují uzly (nodes) a fungují na principu klient/server (publisher/subscriber).



Obr. 28 Struktura fungování ROSu [22]

9.4.1 Catkin [23]

Catkin je oficiální kompilační systém ROSu a nahrazuje předchozí systém rosbuild. Catkin používá CMake makra s Python skripty pro přídavnou funkcionalitu nad běžným CMake. V nových verzích ROSu je Catkin schopen sám stáhnout a nainstalovat chybějící balíčky. Obsahuje také podporu pro závislosti jednotlivých balíčků a kompilaci pro jiné platformy, než na kterých běží, což se hodí zejména při vývoji programu pro mikropočítače na desktopovém PC.

9.4.2 MAVROS [24]

MAVROS je balíček systému ROS, který umožňuje komunikaci protokolem MAVLink, který používá Pixhawk, nebo GCS. Jedná se tedy o takový komunikační můstek mezi Pixhawkem a ROSem, popř. ROSem a GCS.
Při instalaci MAVROSu potřebujeme nejdříve pracovní složku Catkinu, kterou již ale máme vytvořenou z instalace ROSu. Přeskočíme tedy na další krok a to je instalace MAVLinku

```
rosinstall_generator --rosdistro kinetic mavlink | tee /tmp/mavros.ros
install
```

Připravíme a natáhneme instalaci MAVROSu.

```
rosinstall_generator --upstream mavros | tee -a /tmp/mavros.rosinstall
```

Zkompilujeme a doinstalujeme závislosti.

```
wstool merge -t src /tmp/mavros.rosinstall
wstool update -t src -j4
rosdep install --from-paths src --ignore-src -y
```

Spustíme instalační skript geolib.

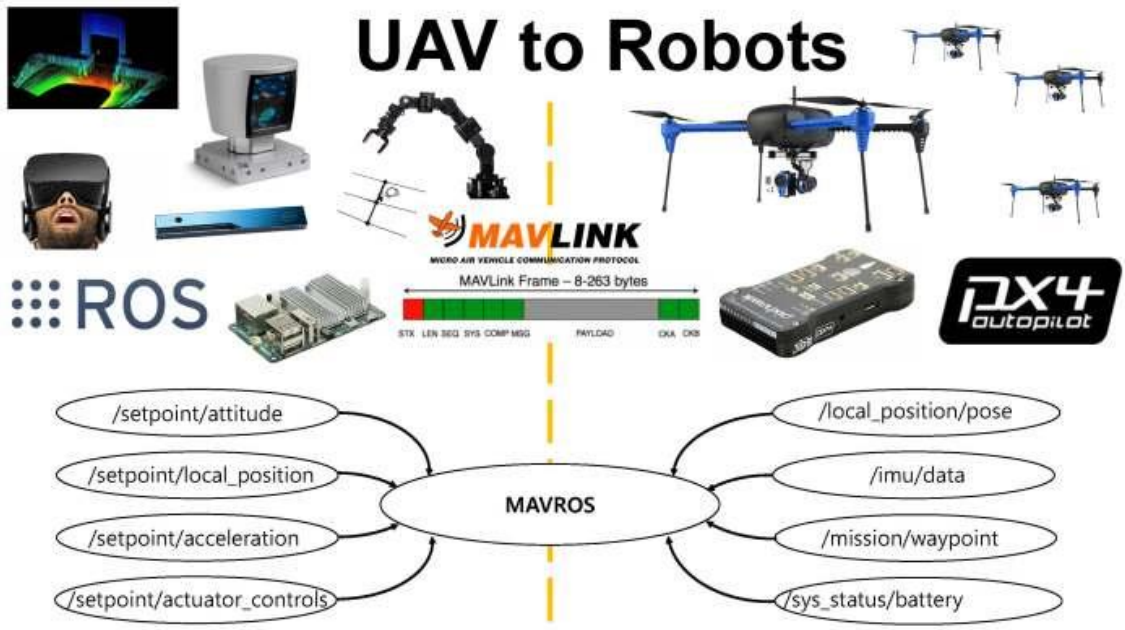
```
./src/mavros/mavros/scripts/install_geographiclib_datasets.sh
```

Spustím Catkin.

```
catkin build
```

A dokončíme instalaci.

```
source devel/setup.bash
```



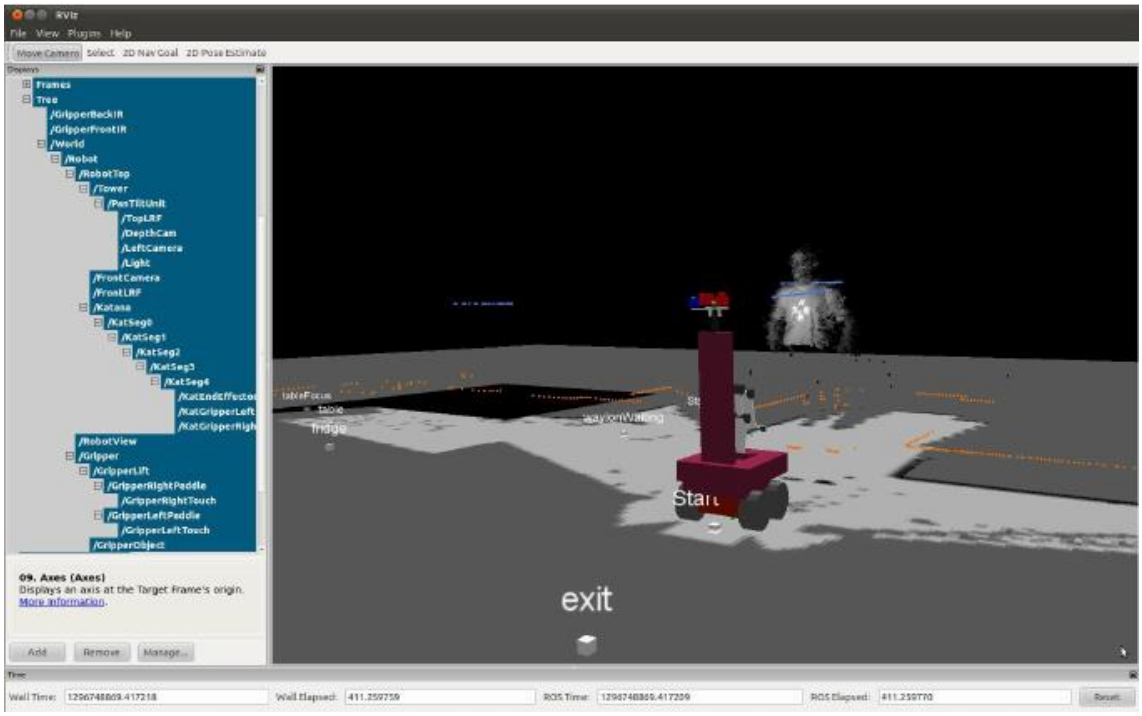
Obr. 29 Způsob fungování MAVROS [25]

9.4.3 ROSPACK

Rospack je konzolový program pro zjišťování informací o balíčcích dostupných na daném zařízení. Umí například prohledat systém a vypsat dostupné balíčky, zjistit jejich závislosti a vygenerovat závislostní strom.

9.4.4 RVIZ

RVIZ je 3D vizualizační nástroj pro ROS. Umožňuje zobrazení robotu a okolního prostředí, dat ze sensorů a nahrávání všech dat pro pozdější použití. Dají se zde například zobrazit data z kamer, laserů atd.



Obr. 30 Náhled na RVIZ [26]

9.5 Směřování dalšího vývoje

Instalace ROSu se i přes mnoho různých obtíží způsobených specifickými nároky ostatního softwaru nakonec podařila. Většina potíží byla způsobena nutností použít starší verzi ROSu, která už nemá mnoho balíčků k dispozici. Při instalaci ze zdrojových kódů jsem se dostal do nekonečné smyčky vzájemných závislostí. Jelikož se mi nepodařilo vyřešit závislosti při instalaci vlastního ROS node pro ovládání sekačky rozhodl jsem se řídicí část napsat v Pythonu s použitím knihovny DroneKit pro standardizaci. Pro potřeby této aplikace je to dostatečné řešení a použitím standardizované knihovny si kód zachová i potřebnou míru standardizace a bude ho s úpravami možné použít jako ROS node.

9.6 DroneKit

DroneKit je ve své podstatě Python knihovna, která se snaží nastolit přístup k autonomnímu robotu jako k API. Umožňuje vytváření Pythonových aplikací, které komunikují s robotem přes protokol MAVLink. DroneKit umožňuje vyšší míru abstrakce tím, že uživatele odstiňuje od protokolu MAVLink. Je tak teoreticky možné napsat jednoduchý program i bez znalosti tohoto protokolu. Další důležitou vlastností je standardizace. Zaváděním standardizovaných metod a přístupů se zjednoduší sdílení kódu, zajistí se větší robustnost výsledného řešení a sníží chybovost. Velmi důležité je programovat „defenzivně“ tzn. Pokud chceme zařízení uvést do „ARMED“ módu, tak nejdříve podmínkou otestujeme prerekvizity, poté odešleme příkaz „přejdi do ARMED módu“ a nakonec otestujeme příkazem „jsi v ARMED módu?“. Tenhle přístup je kromě dobrých programátorských zvyklostí dán hlavně tím, že řídicí jednotka robotu nemusí daný příkaz přijmout, nebo k ní nemusí vůbec dorazit, jelikož MAVLink není navržen jako bezztrátový protokol. V mnoha případech jednotka příkaz nepřijme, ale nedá o tom nijak vědět, na což musíme při programování pamatovat. Když ale jednotka příkaz nepřijme a dá o tom vědět, DroneKit nemusí tuto zprávu vůbec zaznamenat.

9.6.1 Instalace DroneKitu a SITL [27]

K instalaci DroneKitu budeme potřebovat PIP, který již nainstalován je a *python-dev*, který je třeba doinstalovat.

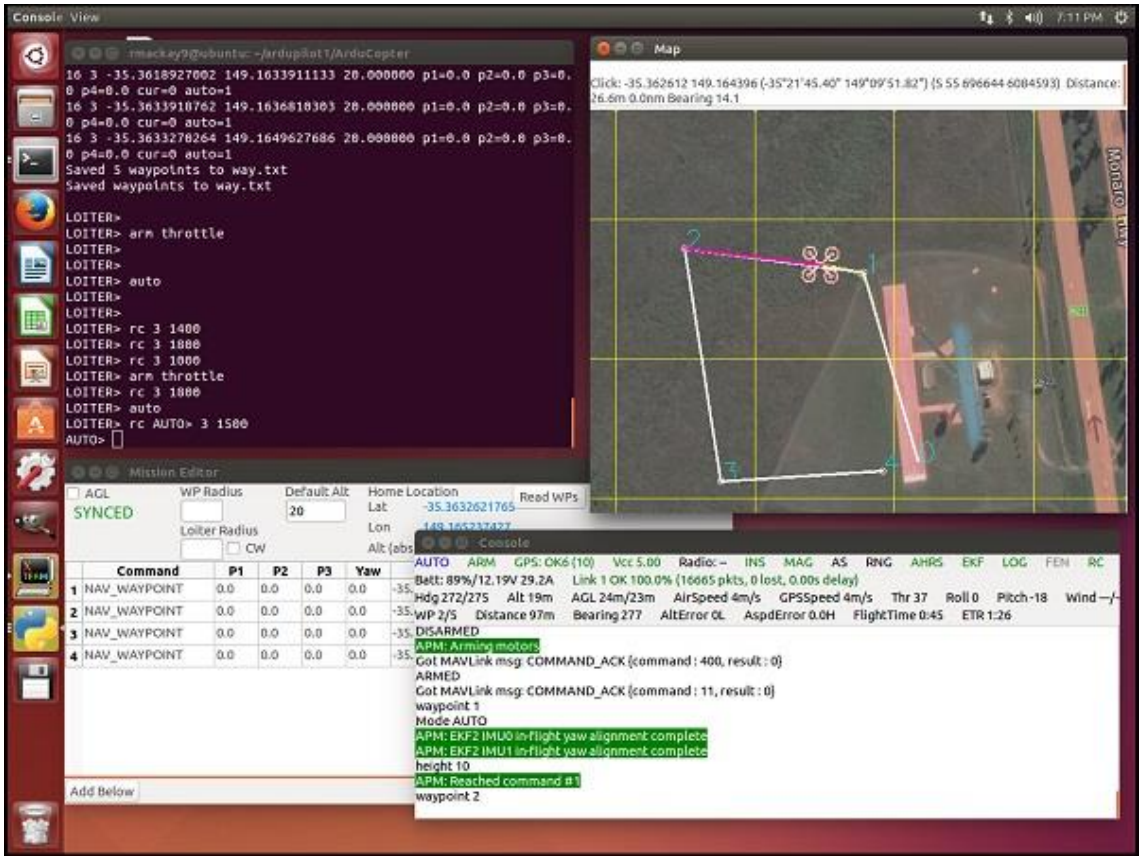
```
sudo apt-get install python-dev
```

Instalaci samotnou spustíme pomocí PIP. Jedná se o takzvaný „one-line installation“, takže tento příkaz za nás zařídí vše potřebné.

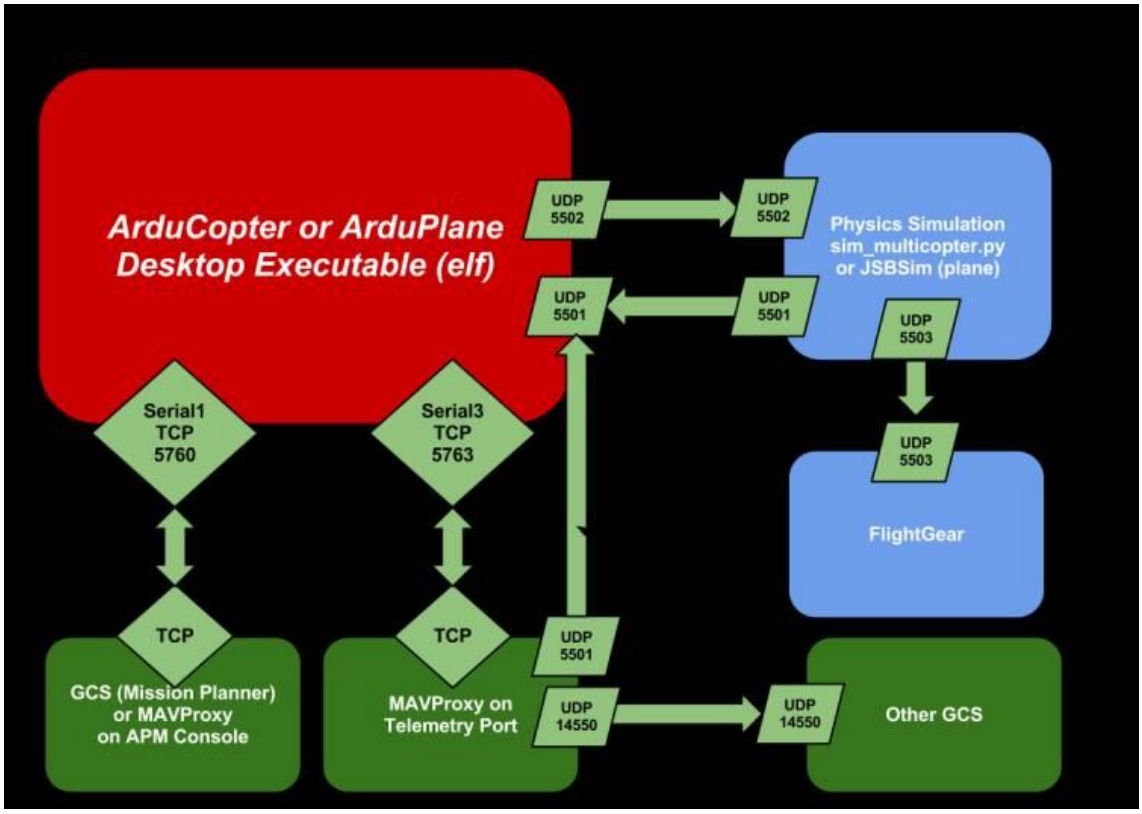
```
sudo pip install dronekit
```

Po instalaci DroneKitu je dobré nainstalovat i simulátor SITL. SITL (software in the loop) nám umožní odzkoušet chování navrženého programu bez fyzického vozidla a tím předejít zbytečným škodám.

```
sudo pip install dronekit-sitl
```



Obr. 31 SITL [28]



Obr. 32 Diagram fungování SITLu [28]

10 OVLÁDÁNÍ STROJE

Ovládání stroje je primárně automatické s přednastavenou misí, ale pro potřeby testování, nebo záložního řízení, či učení mise je připravenou i ovládání pomocí modelářské vysílačky.

10.1 Manuální

Před postupem vývoje do další fáze – autonomní řízení, bylo nutné ověřit chování celého systému v manuálním režimu. Manuální režim jsem zrealizoval pomocí modelářské vysílačky Spektrum DX5e a RC přijímače OrangeRX R617XL s CCPM modulací, která umožňuje data všech kanálů posílat po jedné sběrnici do Pixhawku. Pohyb sekačky je namapovám pouze na levou páčku, která je pružinami držena uprostřed a je tak možné nejelegantněji řídit celý pohyb stroje. Pohybem doleva a doprava se řídí zatáčení a pohybem dopředu jízda dopředu a pohybem směrem k operátorovi couvání. Je tak možné například zároveň jet dopředu a u toho zatáčet. Pokud operátor páčku pustí, tak ta se pomocí pružin sama vystředí a sekačka se zastaví. Motory sečení se ovládají pravou plynovou páčkou, která v nastavené pozici drží. Ovládání režimu AUTOMATICKÝ/MANUÁLNÍ se řídí přepínací páčkou pátého kanálu.

Obr. 33 Spektrum DX5e [29]

10.2 Autonomní

Autonomní ovládání je uděláno pomocí skriptu v Pythonu, který standartně běží v RPi, ale pro potřeby vývoje je možné ho spouštět i v PC a ovládat tak Pixhawk po síti přes UDP port pomocí MAVProxy. Pro vývoj jsem zvolil IDE PyCharm od Jet Brains, které považuji za nejvhodnější IDE pro Python na trhu. Velkou výhodou je možnost skript spouštět přímo z vývojového prostředí, což při vývoji šetří čas.

10.2.1 Spouštění ovládacího skriptu

Samotný ovládací skript se jmenuje mower.py a je přiložen kompletní v příloze. Skript je napsán dle ukázek funkcí DroneKitu. [30] Níže je ukázán pouze úryvek bez funkcí a obslužných rutin zodpovědný za splnění mise v automatickém režimu.

```
print('Upload a mission')
upload_mission("garden.waypoints")
arm()
print("Starting mission")
vehicle.commands.next = 0
# Set mode to AUTO to start mission
vehicle.mode = VehicleMode("AUTO")
while True:
    nextwaypoint = vehicle.commands.next
    print('Distance to waypoint (%s): %s' % (nextwaypoint, distance_to
_current_waypoint()))


    if nextwaypoint == 2:  # Start mowing
        print('Starting mowing')
    if nextwaypoint == numberOfWaypoints-3:  # Stop mowing
        print("Done")
        break;
    time.sleep(1)


print("Close vehicle object")
vehicle.close()
if sitl is not None:
    sitl.stop()
```

Tento skript se spouští pomocí mower.sh, který obsahuje tzv. connect string a jeho obsah vidíme níže. Pokud bychom spustili skript bez něj, tak by se spustil se simulátorem SITL.

```
#!/bin/bash
python mower.py --connect 127.0.0.1:14551
```

10.2.2 Vytvoření daemonu ovládacího skriptu

Vytvoříme soubor mower.service, který obsahuje instrukce pro chování daemonu.

```
[Unit]
Description=Startup Script Service
After=multi-user.target
[Service]
Type=simple
Restart=always
RestartSec=10
User=root
ExecStart=/home/pi/mower.sh
[Install]
WantedBy=multi-user.target
```

Soubor zkopírujeme do */lib/systemd/system* a přidáme práva pro spuštění skriptu, který nám zajistí spuštění programu se správnými parametry. Nakonec službu aktivujeme.

```
cd /home/pi/
sudo chmod +x mower.sh
sudo mv mower.service /lib/systemd/system
sudo systemctl daemon-reload
sudo systemctl enable mower.service
```

10.2.3 Průběh mise

Po zapnutí je po otestování stavu všech důležitých součástí do Pixhawku nahrána mise, která se skládá z bodů na trase určené k projetí. Na začátku mise sekačka najede k prvnímu bodu a spustí samotné sekání. Pokud se v průběhu sekačka něčeho dotkne, což je registrováno pomocí sensorů dotyku po jejím obvodu, tak se

vypnou jak motory sekání, tak i motory pohonu a čeká se na vstup uživatele. Jelikož je trasa naplánována mimo veškeré překážky, tak se tato funkce používá jako bezpečnostní a nepředpokládá se přítomnost cizích předmětů na trase. Po projetí trasy se sekačka vrátí na místo, ze kterého vyjela. Pokud v průběhu mise klesne napětí na baterii pod únosnou mez, tak se mise přeruší a sekačka se také vrátí na místo startu.

10.2.4 Vygenerování mise

Ruční zadávaní všech bodů na trase by bylo zejména u větších ploch velmi náročné. Mission Planner má naštěstí funkce, které nám toto pomohou částečně automatizovat. Nejjednodušší je použít funkci „Grid“, kde ohraničíme danou plochu a specifikujeme rozestupy bodů ve všech osách. Funkce nám potom všechny body vygeneruje automaticky, včetně optimální posloupnosti. Jednotlivé takto vygenerované plochy můžeme spojovat za sebe a utvářet tak komplexní mise.

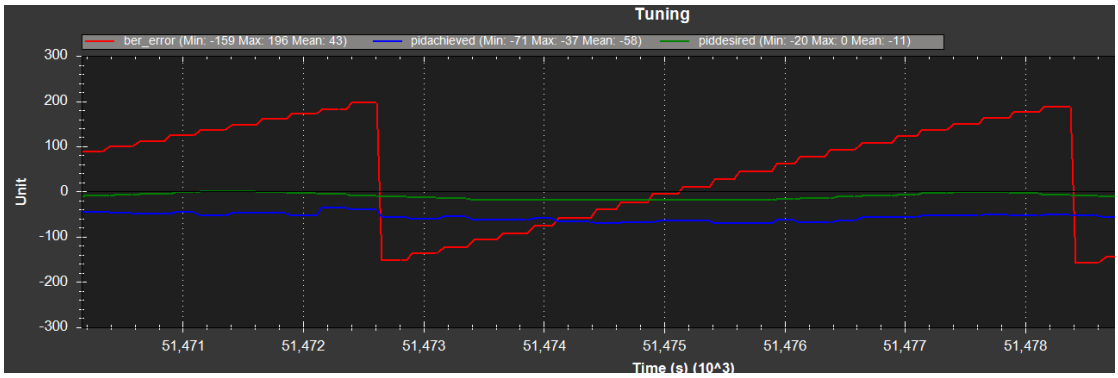
Obr. 34 Náhled na vygenerovanou misi

11 OVĚŘENÍ FUNKNOSTI CELÉHO ŘEŠENÍ

Po zapojení všech dílčích celků dohromady jsem provedl umístění jednotky Pixhawk na spodní stranu obráceného plastového boxu, který je ke konstrukci připevněn pomocí na 3D tiskárně míru vytištěných úchytů, jelikož měření ukazovalo nepřípustné magnetické rušení od běžících motorů sečení. Na výstupní hřídelky motorů jsou nalepeny magnety, a vedle nich na uchycení motorů připojeny Halovy sondy pro budoucí rozšíření přesného měření otáčení kol.

11.1 Oživení

Při oživování celého zařízení bylo nejdříve odzkoušeno chování v manuálním režimu popsaném v předchozí kapitole. Zde vše fungovalo bez problémů a všechny funkce šly pomocí modelářské vysílačky ovládat. Problém nastal v módech GUIDED – přímá cesta do zadaného waypointu z aktuální pozice, nebo AUTO – automatické vykonávaní naprogramované mise. V momentě přepnutí sekačky do těchto módů se stala jedna ze dvou věcí. Pokud byl následující waypoint v levé polorovině před sekačkou (2. kvadrant), tak vše fungovalo do doby, než poloha překmitla do pravé poloroviny (1.kvadrant) a sekačka se začala točit na místě. Pokud byl waypoint kdekoliv jinde, než ve druhém kvadrantu (z pohledu sekačky), tak se začala na místě točit okamžitě. V tuto chvíli dostala sekačka pracovní název „Galileo“ (A přece se točí!). Nejdříve jsem proměřil vliv možných rušení od motorů pohonu, které mohlo ovlivňovat kompas. Tato obava se ukázala jako lichá. Dalším krokem bylo zjistit, zda sekačka vůbec ví, kam má jet. Nastavil jsem tedy Pixhawk pro sběr živých dat a spustil autonomní misi. Sekačka se dle očekávání začala točit dokola. Na grafu níže vidíme červeně vynesenou aktuální odchylku od požadované trajektorie ve stupních, která ukazuje, že navigační část logiky funguje tak, jak má a nedochází ke ztrátě orientace.



Obr. 35 Živá data z průběhu autonomní mise

Modře vidíme požadovaný zásah PID regulátoru zatáčení (popsán v další kapitole) a zeleně skutečnou výslednou akční veličinu. Průběh v tomto grafu vypadal, jako by regulátor reguloval jenom do jedné strany – proto točení. Po poradě s vedoucím práce jsem se dal studia fungování části zdrojového kódu ArduRoveru zodpovědného za otáčení s cílem nalézt chybu zodpovědnou za toto chování.

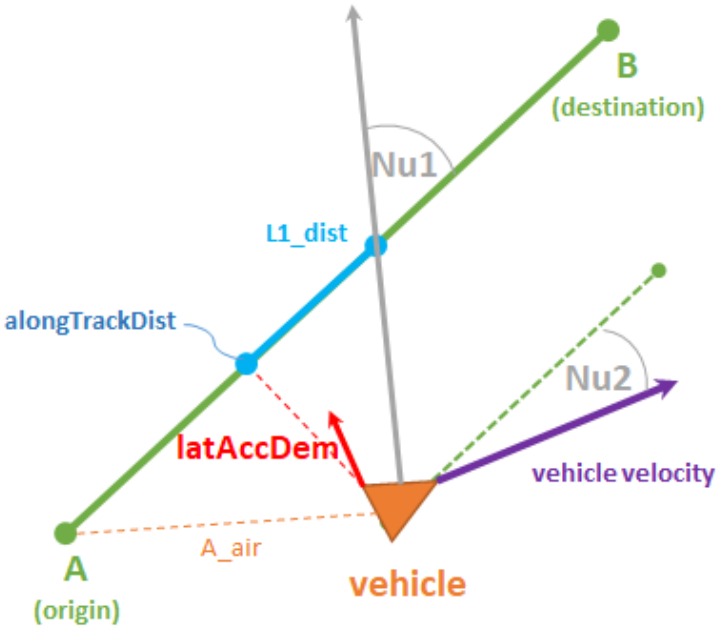
11.1.1 Způsob určení natočení

Ze všeho nejdříve je důležité vůbec pochopit způsob, jakým ArduRover s natočením pracuje, jelikož je velice odlišný od toho, co by se dalo na první pohled předpokládat. První funkce, která řídí natočení je takzvaný L1 regulátor. Zde vzniká prvotní impuls k natočení, který pak prochází dalšími regulátory. Tento regulátor funguje trochu netradičně v tom smyslu, že nepracuje s natočením například v radiánech, ale jeho výstupem je požadované boční zrychlení robota, které ho dostane zpět na úsečku spojující minulý a aktuální waypoint. Celý L1 regulátor se nastavuje dvěma parametry – NAVL1_PERIOD a NAVL1_DAMPING. První nastavuje agresivitu zatáčení podle použitého fyzického robotu a jeho pohonu a druhý nastavuje tlumení.

Jednotlivé části v nákresu pak znamenají:

Tabulka 2. Popis regulátoru L1

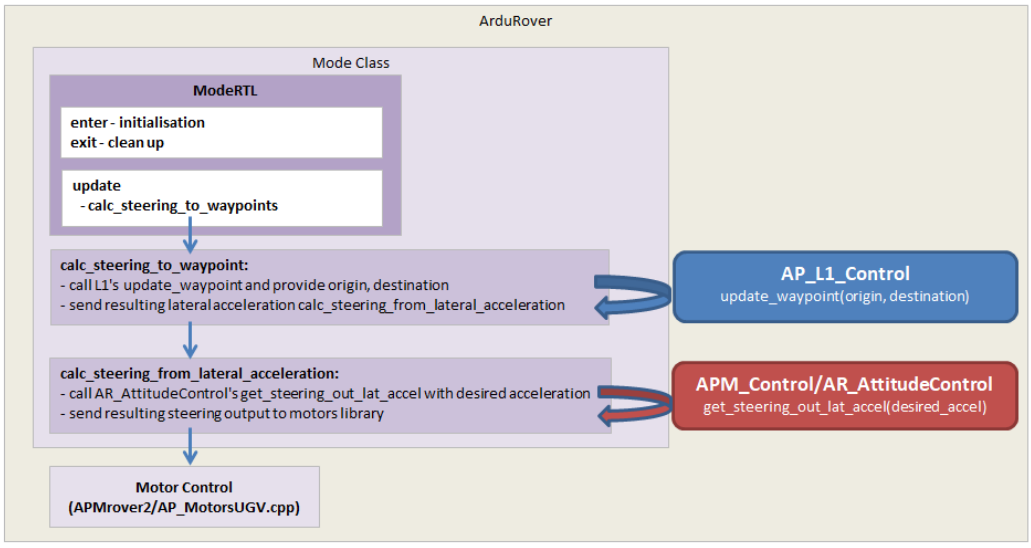
A	Minulý waypoint.
B	Aktuální waypoint.
Nu1	Úhel mezi robotem a bodem L1_dist na úsečce AB.
Nu2	Úhel mezi vektorem rychlosti a natočením úsečky AB
alongTrackDist	Vzdálenost mezi A a nejbližším bodem kolmého průmětu pozice robotu na úsečku AB.
L1_dist	Vzdálenost mezi cílovým bodem L1_dist na úsečce AB a bodem along TrackDist na úsečce AB. L1_dist=0,3*tlumení*agresivita*rychlost.
latAccDem	Výsledný příčné zrychlení.
Vehicle velocity	Aktuální rychlost.
vehicle	Robot – v našem případě sekačka.



Obr. 36 Nákres způsobu určení natočení robotu [31]

$$latAccDem = \frac{4 * tlumení^2 * rychlost^2 * \sin(Nu1 + Nu2)}{\text{L1\_dist}} \quad [1]$$

11.1.2 Funkce pro regulaci zatáčení



Obr. 37 Blokové schéma regulace zatáčení [31]

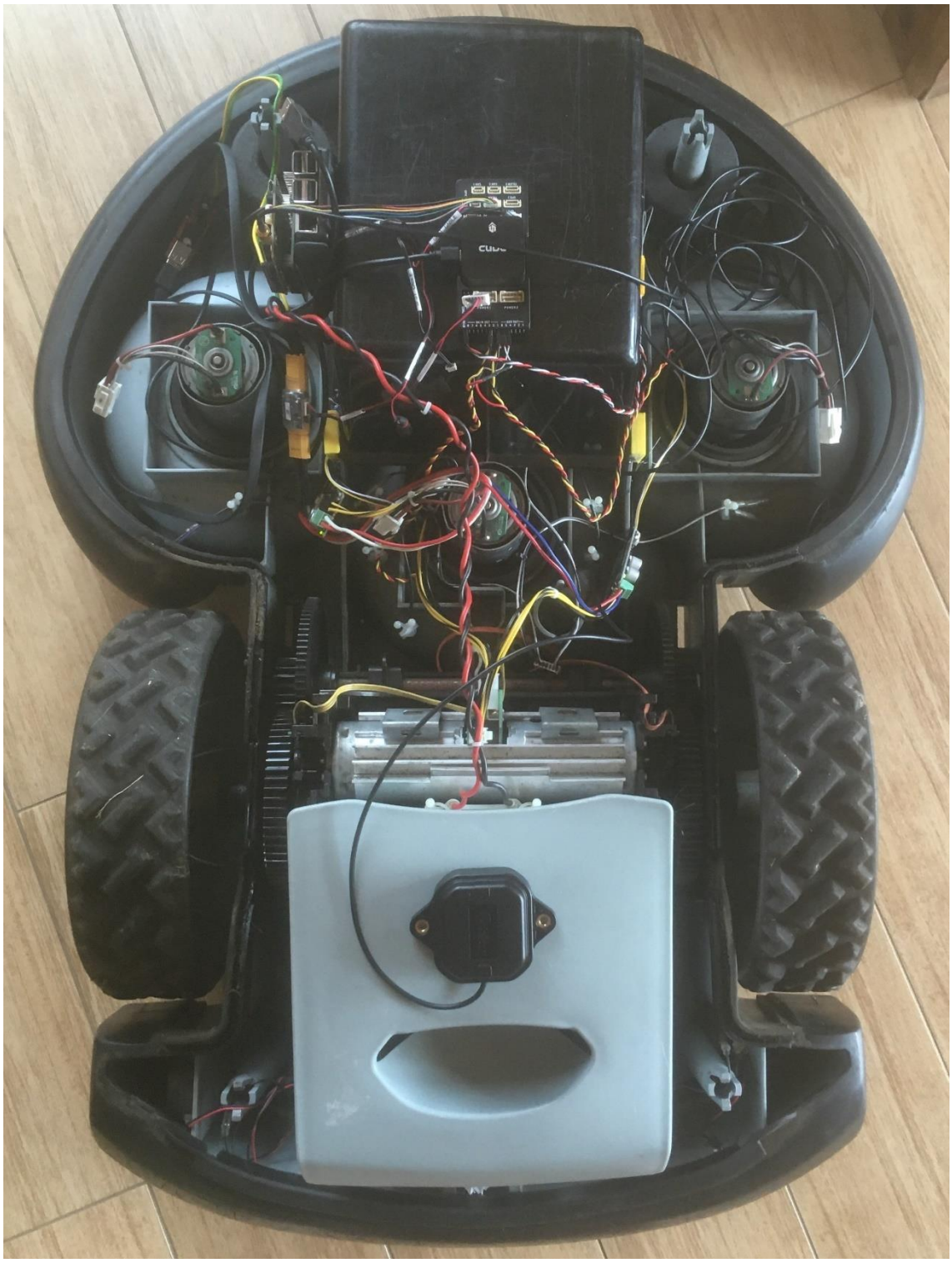
Funkce *calc_steering_to_waypoint*, která získala požadovanou hodnotu příčného zrychlení z regulátoru L1(veličina latAccDem), ho předá do funkce

calc_steering_from_lateral_acceleration, která si zavolá PID regulátor ve funkci *AR_attitudeControl* s funkční hodnotou tohoto příčného zrychlení. Výsledný povel pro řízení je pak poslán dál ke zpracování funkci *AP_MotorsUGV,* která již podle nastaveného podvozku dává správné pokyny motorům/servu.

11.1.3 Vyřešení chyb

Po prověření všech předešlých celků a jejich správného fungování bylo zjištěno, že se v realizaci nacházejí dvě chyby, které se navzájem eliminovaly, a tak bylo ztížení jejich odhalení. První z chyb byla obrácená polarita na levém motoru. Druhou chybou bylo mapování vstupů RC vysílačky v Pixhawku, který tak při jízdě dopředu vydával povel k zatáčení a při zatáčení povel pro jízdu dopředu. Tím, že je použit diferenciální podvozek a byla prohozená polarita na jednom z motorů, tak v manuálním režimu vše fungovalo správně a k problémům docházelo až v automatických režimech. Než jsem dospěl k tomuto řešení, tak byl aktualizován firmware v Pixhawku, nainstalována nejnovější verze Mission Planneru a zkompilována a nahrána nejnovější vývojářská verze ArduRoveru (4.1.0) z GitHubu.

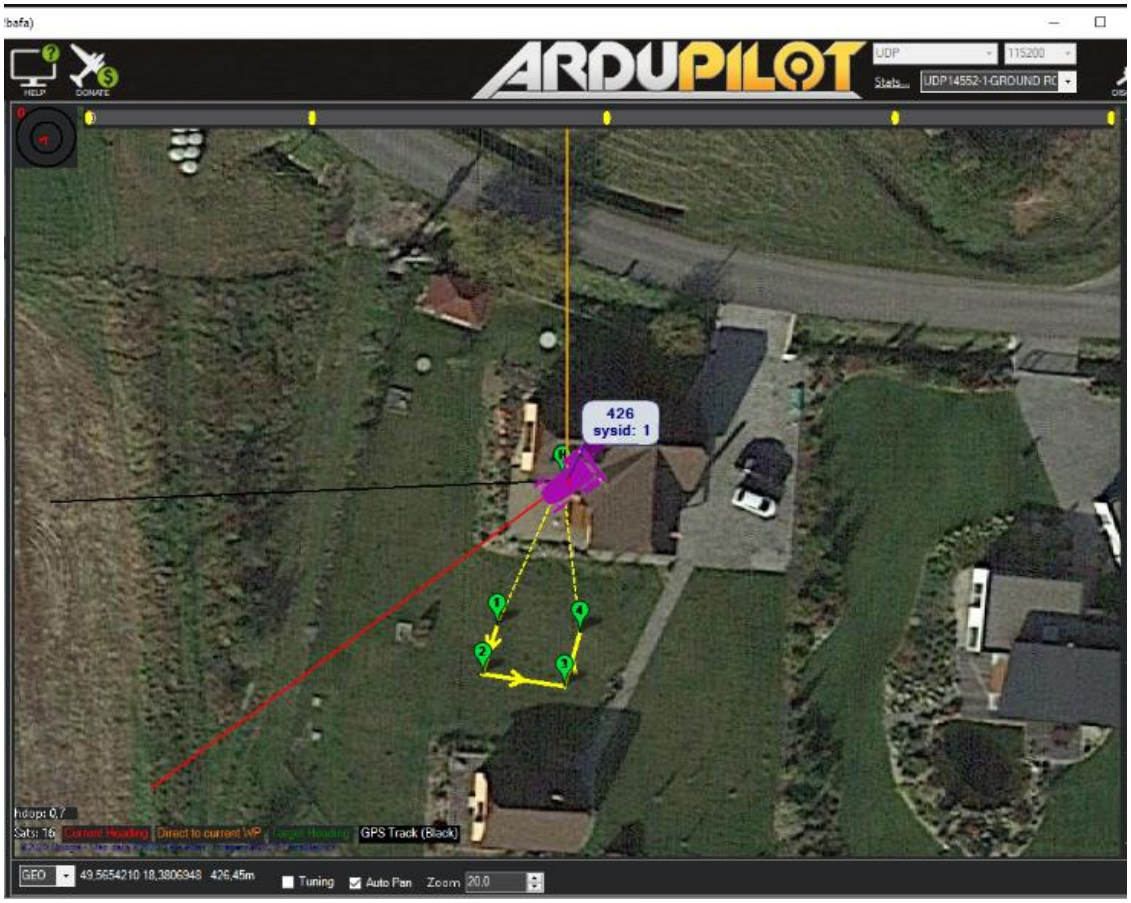
11.2 Finální zapojení



Obr. 38 Odkrytované hotové řešení robotické sekačky

11.3 Určení pozice stroje pomocí GPS RTK

Určení pozice podle GPS RTK probíhá přesně a podle předpokladů. Testování proběhlo přímo na několika bodech budoucí pracovní plochy, kde se porovnávala pozice zjištěná s pozicí skutečnou. Jelikož pro účely navigace sekačky po zahradě není potřeba absolutní přesnosti, tak jsem se v testu obešel bez zeměměřičského vybavení a porovnával, zda se zjištěná poloha zanesená do Google Maps shoduje s aktuální polohou sekačky. Bylo ověřeno, že polohy se odlišují maximálně v řádech jednotek centimetrů, což může být způsobeno jak chybou určení polohy pomocí GPS, tak i nepřesností v mapě, ale pro tuto aplikaci je to více než dostatečné. Bylo potvrzeno, že poloha je přesně určena i kolem budov a stromů, což byla předpokládaná problematická místa.



Obr. 39 Ověření správné polohy v těsné blízkosti budovy

11.4 Autonomní mise

Autonomní mise spouštěné z PC i RPi probíhají dle předpokladů a řešení je tak plně funkční. Začátek i konec mise je korektní a lze i v průběhu převzít manuální řízení. V další verzi plánuji komunikaci mezi RPi a Pixhawkem pomocí UART místo USB kvůli větší robustnosti.

12 ZÁVĚR

V této práci jsem se zabýval vývojem robotické sekačky. Začal jsem rešerší možností na trhu a výběrem vhodného GNSS přijímače pro navigaci malého zahradního robota ve venkovním prostředí. Dále jsem sestrojil měřicí přípravek a nejdříve provedl měření v konfiguraci dvou přijímačů, kde oba přijímače měly dostupné korekce. První přijímač počítal svou polohu a po rozhraní UART ji zasílal do přijímače druhého, který také počítal svou polohu a ze vzájemných poloh probíhal výpočet délky a natočení vektoru mezi přijímači. Bohužel implementace výrobce je poněkud nepřesná a pro naše použití dává nepřiměřeně velkou chybu. Dalším problémem je pokles frekvence dodávaných polohových dat na 1 Hz, což je pro danou aplikaci nedostatečné. Naopak pokud je provozován přijímač pouze jeden, jsou výsledky přímo excelentní, a proto jsem ZED-F9P zvolil pro použití v dané aplikaci. Na základě měření byla s přijímačem spárována anténa U-Blox, která pro přepokládané podmínky použití přinášela nejlepší výsledky. Následoval výběr vhodné open-source platformy pro řízení robotu včetně jejího softwaru. Na trhu je těchto platforem relativně velké množství, ale mnoho z nich má mizivou podporu ve vývojářské komunitě, či nesplňují všechny nároky pro použití v této aplikaci. Vybraná kombinace Pixhawku Cube s ArduRoverem se potvrdila jako výborná volba. Podpůrný program pro PC – Mission Planner se také ukázal jako vhodné řešení. Zvolené řešení podvozku – úprava komerční platformy se pro práci stalo málem osudným, jelikož logistika mnoha dodavatelů kvůli korona krizi vázla a v normální době banální problém jako nahrazení vadného motoru, či zajištění přesného typu motorového regulátoru byl téměř nadlidský úkol. Část palubního počítače – nadřazeného řízení pro Pixhawk, doznala v průběhu realizace největších změn. Bylo nutné zvolit jiný, než původně zamýšlený, způsob řešení, který se ale podařilo zdárně dotáhnout do konce. Výsledkem původního směru vývoje je ale alespoň hotová univerzální platforma pro mobilního robota s ROS a NTRIP klientem. Ke konci práce jsou popsány způsoby ovládání, problémy při oživování a otestování funkčnosti celého řešení. Výsledkem práce je tedy funkční robotická sekačka postavená tak, aby byl umožněn další vývoj. V nejbližší době plánuji přidat přímé měření otáčení jednotlivých kol, které je již v aktuální verzi téměř hardwarově hotové. Dalším krokem bude automatické dokování do nabíjecí stanice a ovládání pomocí vlastního webového rozhraní.

13 LITERATURA

[1] Ardumower Forum, 2013. [Online]. Available: https://www.ardumower.de/index.php/en/induktion.

[2] Wikipedia, „Lidar," 2020. [Online]. Available: https://cs.wikipedia.org/wiki/Lidar.

[3] A. Grau, „ROS driver for Ardumower - demo using DIY lidar," 2019. [Online]. Available: http://grauonline.de/wordpress/?page_id=1233.

[4] Alexander Grau, „CAMERA-BASED POSITION ESTIMATION USING A GOOGLE TANGO PHONE," 2019. [Online]. Available: http://grauonline.de/wordpress/?page_id=2109.

[5] Swift Nav, „PIKSI GNSS MODULE," 2019. [Online]. Available: https://www.swiftnav.com/piksi-multi.

[6] J. Kryštof, „GPS modul pro roboty," 2020. [Online]. Available: https://www.roboticjournal.cz/clanky/2017-01-gps-modul-pro-roboty.

[7] U-Blox, „ZED-F9p," 2019. [Online]. Available: https://www.u-blox.com/en/product/zed-f9p-module.

[8] Direct Industry, „GNSS RTK Receiver," [Online]. Available: https://www.directindustry.com/prod/ardusimple/product-222256-2269707.html.

[9] Penn State Colledge of Earth and Mineral Sciences, „Course GEOG862," [Online]. Available: https://www.e-education.psu.edu/geog862/node/1845. [Přístup získán 2019].

[10] Beagle Board Foundation, „Beagle Bone Blue," 2020. [Online]. Available: https://beagleboard.org/blue.

[11] R. Shop, 2019. [Online]. Available: https://www.robotshop.com/en/pixhawk-21-standard-set.html.

[12] ArduPilot Dev Team, „Companion computers," 2019. [Online]. Available: https://ardupilot.org/rover/docs/common-companion-computers.html.

[13] ArduPilot Dev Team, „Mission Planner Home," 2019. [Online]. Available: https://ardupilot.org/planner/.

[14] ArduMower, „ArduMower Wiki," 2020. [Online]. Available: https://wiki.ardumower.de/index.php?title=Chassis_(English).

[15] R. S. c. R.F., „Ardumower Arctic Hare,“ 2020. [Online]. Available: https://www.robotshop.com/community/forum/t/ardumower-arctic-hare/29421.

[16] Krtkův ráj, [Online]. Available: http://www.krtkuv-raj.cz/zahradni-technika/priprava-noveho-webu/produkty/produkty-zahradni-techniky/07-roboticke-sekacky/roboticke-sekacky-robomower/roboticka-sekacka-rl-2000/.

[17] Cytron Technologies, 2020. [Online]. Available: https://www.cytron.io/p-13amp-6v-30v-dc-motor-driver.

[18] ROS. org, kolektiv autorů, 2020. [Online]. Available: http://wiki.ros.org/kinetic/Installation/Source.

[19] Ubiquiti Robotics, 2020. [Online]. Available: https://downloads.ubiquityrobotics.com/pi.html.

[20] D. Stoecker, „POSIX NTRIP Client,“ 2020. [Online]. Available: https://github.com/nunojpg/ntripclient.

[21] ArduPilot Dev Team, „MAVProxy,“ 2020. [Online]. Available: https://ardupilot.github.io/MAVProxy/html/getting_started/download_and_installation.html.

[22] L. Joseph, „Fundamentals of ROS,“ 2017. [Online]. Available: https://subscription.packtpub.com/book/hardware_and_creative/9781783554713/1/ch01lvl1sec8/fundamentals-of-ros.

[23] ROS.org, kolektiv autorů, „Catkin,“ 2020. [Online]. Available: http://wiki.ros.org/catkin/conceptual_overview.

[24] ROS.org, „MAVROS,“ 2020. [Online]. Available: https://github.com/mavlink/mavros/blob/master/mavros/README.md#installation.

[25] J. Lim, „PX4 Offboard Control Using MAVROS on ROS,“ 2020. [Online]. Available: https://www.codetd.com/article/2918485.

[26] D. W. R. Paulus, „RVIZ,“ 2020. [Online]. Available: https://www.researchgate.net/figure/The-rviz-visualization-tool-showing-various-visualizations-of-sensor-data-In-this-view_fig6_260318037.

[27] DroneKit, „Dronekit Getting Started,“ 2020. [Online]. Available: https://github.com/dronekit/dronekit-python.

[28] ArduPilot dev team, „SITL simulator,“ 2020. [Online]. Available: https://ardupilot.org/dev/docs/sitl-simulator-software-in-the-loop.html.

[29] 3dextratime, „Spektrum DX5e,“ 2020. [Online]. Available: https://www.3dextratime.eu/products/spektrum-dx5e-with-ar610.

[30] DroneKit, „DroneKit Examples,“ 2020. [Online]. Available: https://github.com/dronekit/dronekit-python/tree/master/docs/examples.

[31] ArduPilot Dev Team, „Rover:L1 navigation overview,“ 2020. [Online]. Available: https://ardupilot.org/dev/docs/rover-L1.html.

[32] The Green Head eshop, 2019. [Online]. Available: https://www.thegreenhead.com/2004/12/robomower-rl1000-robotic-lawn-mower-w.php.

[33] RaspberryPi inc, „Updating Raspberry Pi OS,“ [Online]. Available: https://www.raspberrypi.org/documentation/raspbian/updating.md.

[34] PX4 DEV TEAM, „Traxxas Stampede,“ 2020. [Online]. Available: https://docs.px4.io/v1.9.0/en/frames_rover/traxxas_stampede.html.

14 SEZNAM PŘÍLOH

Příloha 1. – řídicí skript pro palubní počítač mower.py
Příloha 2. – konfigurační soubor pro Pixhawk
Příloha 3. – konfigurační soubor pro GPS přijímač ZED-F9P